Universidad Nacional del Nordeste
Facultad de Ciencias Exactas y Naturales y Agrimensura

Maestría en Sistemas de Redes y Telecomunicaciones

TESIS

ASPECTOS DE LA CALIDAD DE SERVICIO QUE INFLUYEN SOBRE LOS PROTOCOLOS DE ENRUTAMIENTO EN LAS REDES AD-HOC

DIRECTOR DE TESIS

Magister en redes de datos

REINALDO JOSE RAMON SCAPPINI

MAESTRANDO

Licenciado en sistemas

VICTOR JOSE KAUFHOLD

Corrientes, 22 de Noviembre 2018

Índice general

Indice de figuras

Índice de tablas

Abreviaturas

Abreviaturas	Descripción	Significado
ACA	Autonomous Component Architecture	Arquitectura de Componentes Autónomos
ADHOC	Mobile Ad-Hoc Network	Red Inalámbrica Móvil
AODV	Ad-hoc On Demand Vector	Ad-hoc en vector demanda
banda ISM	Industrial, Scientific and Medical	Industrial, científico y médico.
CCK	Complementary Code Keying	Codificación de código complementario
CSMA/CA	Carrier sense multiple access with collision avoidance	Acceso múltiple por detección de portadora y prevención de colisiones
DiffServ	Differentiated Services	Servicios diferenciados
DSCP	Differentiated Services Code Point	Punto de Codificación de Servicios Diferenciados
DSDV	Destination Sequenced Distance Vector	Destino Secuenciado con vector distancia
DSR	Dynamic Source Routing	Enrutamiento dinámico fuente
DSSS	Direct Sequence Spread Spectrum	Espectro extendido de secuencia directa
FHSS	Frequency Hopping Spread Spectrum	Frecuencia de salto Spread Spectrum
GPS	Global Positioning System	Sistema de Posicionamiento Global
GUI	graphical user interface	Interfaz gráfica de usuario
IEEE	Institute of Electrical and Electronics Engineers	Instituto de Ingeniería Eléctrica y Electrónica
IntServ	Integrated Services	Servicios Integrados
IP	Internet Protocol	Protocolo de Internet
IPV4	Internet Protocol Version 4	Protocolo de Internet versión 4
IPV6	Internet Protocol Version 6	Protocolo de Internet versión 6
IRTF	Internet Research Task Force	Fuerza de Tareas de Investigaciones de Internet
LAN	Local Area Network	Red de área local
LLC	Logical Link	Control lógico de enlace
LTE	Long Term Evolution	Evolución a largo plazo
MAC	Media Access Control	Control de acceso al medio
MANET	Mobile ad hoc network	Red ad hoc móvil
MIMO	Múltiple Input Múltiple Output	Múltiple entrada múltiple salida
MPLS	Multiprotocol Label Switching	Conmutación de etiquetas multiprotocolo
MPR	MultiPoint Relay	Retransmisores multipunto
NIC	Network Interface Card	Tarjeta de interfaz de red
NS3	Network Simulator 3	Simulador de Redes versión 3

Abreviatura	Descripción	Significado
NSF-CISE	National Science Foundation, Computer & InformationScience & Engineering	Fundación Nacional de Ciencias, Informática y Ingeniería y Ciencias de la Información
OFDM	Orthogonal Frequency-Division Multiplexing	Multiplexación por división de frecuencia ortogonal
OLSR	Optimized link State Routing Protocol	Protocolo de enrutamiento de estado de enlace optimizado
OPNET	Optimized Network Engineering Tools	Herramientas optimizadas para ingeniería de red
OSI	Open System Interconnection	Modelo de interconexión de sistemas abiertos
Otcl	Object Tool Command Language	Lenguaje de herramientas de comando orientado a objetos
PCAP	Packet capture data	Paquete de captura de datos
PHB	Per Hop Behavior	Comportamiento por Salto
PHY	physical layer	Capa física
PLCP		Procedimiento de Convergencia de Capa Física
QAM	Quadrature Amplitude Modulation	Modulación de amplitud en cuadratura
QoS	quality of service	Calidad de servicio
REER	Route Error	Error de ruta
RREP	Route Reply	Respuesta de ruta
RREQ	Route Request	Requerimiento de ruta
RSVP	Resource Reservation Protocol	Protocolo de Reserva de Recursos
Tcl	Tool Command Language	Lenguaje de herramientas de comando
TCP	Transmission control protocol	Protocolo de control de transmisión
UDP	User Datagram Protocol	Protocolo de datagramas de usuario
Wi-Fi	Marca comercial	Tecnología que permite la interconexión inalámbrica

Resumen

Este trabajo estudia algunos aspectos de la calidad de servicio que influyen sobre los protocolos de enrutamiento en las redes ad hoc. Para una implementación de enrutamiento orientado a la calidad de servicio eficiente, los requisitos de calidad de servicio deben especificarse en el protocolo de enrutamiento. Una calidad de servicio determinada se solicita especificando sus requisitos en términos de una o más métricas. Este trabajo propone que esas métricas hagan referencia a la perdida de paquetes, el retardo de paquetes y la variación de retardo de paquetes. Para poder establecer una relación entre las métricas propuestas y los tipos de enrutamiento (proactivo y reactivo) en las redes ad hoc se realizó una simulación de ambos tipos de protocolos variando la cantidad de nodos en la red, utilizando diferentes velocidades en los nodos, y aplicando los protocolos de enrutamiento. Como conclusión se pudo comprobar a través de simulación que, en el caso de la demora de extremo a extremo, y la variación del retardo los protocolos AODV y DSDV van empeorando su prestación a medida que aumenta el número de nodos. El protocolo OLSR se mantiene con una prestación aproximadamente constante. En cuanto a la perdida de paquetes los protocolo DSDV es el que ha tenido un comportamiento más modesto. Los protocolos AODV y OLSR tienen comportamientos similares para redes con poca cantidad de nodos. Se destaca el comportamiento estable del protocolo OLSR para redes de 50 y más nodos.

Abstract

This paper studies some aspects of the quality of service that influence the routing protocols in ad hoc networks. For an implementation of routing oriented to the efficient quality of service, the quality of service requirements must be specified in the routing protocol. A given quality of service is requested by specifying its requirements in terms of one or more metrics. This paper proposes that these metrics refer to packet loss, packet delay and packet delay variation. In order to establish a relationship between the proposed metrics and the types of routing (proactive and reactive) in the ad hoc

networks, a simulation of both types of protocols was carried out by varying the number of nodes in the network, using different speeds in the nodes, and applying the routing protocols. In conclusion, it was possible to verify through simulation that, in the case of end-to-end delay, and the variation of the delay, the AODV and DSDV protocols worsen their performance as the number of nodes increases. The OLSR protocol is maintained with an approximately constant performance. Regarding the loss of packets, the DSDV protocol is the one that has had a more modest behavior. The AODV and OLSR protocols have similar behaviors for networks with a small number of nodes. The stable behavior of the OLSR protocol for networks of 50 and more nodes is highlighted.

Keywords

Ad Hoc, AODV, OLSR, DSV, QoS, ns3

Introducción

Este trabajo estudia algunos aspectos de la calidad de servicio que influyen sobre los protocolos de enrutamiento en las redes ad hoc.

En el nivel de capa de red se han desarrollado arquitecturas que plantean modelos para implementar calidad de servicio en una red Ad Hoc. Uno de estos modelos es el modelo de servicios diferenciados (DiffServ) (Blake, y otros, 1998), que utiliza un mecanismo que está orientado a la priorización de tráfico.

El modelo no establece un canal virtual sobre la red y por lo tanto no realiza ninguna reserva de servicios de red, adoptando cada nodo diferentes comportamientos de acuerdo con sus características disponibles y de acuerdo a su fabricante.

La diferenciación de servicios se lleva a cabo mediante la definición de comportamientos específicos para cada clase de tráfico entre dispositivos de interconexión, a este hecho se le conoce como Comportamiento por Salto (PHB - Per Hop Behavior). De las tres clases de servicio que define el modelo de servicios diferenciados (DiffServ) (Blake, y otros, 1998) esta el servicio de Reenvío Asegurado que asegura un trato preferente, pero no garantiza caudales, retardos, etc.

La selección de una ruta adecuada según los requerimientos de tráfico (Lin, y otros, 1999), teniendo en cuenta la información sobre el estado de la red, es uno de los objetivos importantes del enrutamiento orientado a la calidad de servicio. El protocolo de enrutamiento orientado a la calidad de servicio debe encontrar rutas que consuman recursos mínimos de acuerdo con las métricas de calidad de servicio relevantes.

Para una implementación de enrutamiento orientado a la calidad de servicio eficiente, los requisitos de calidad de servicio deben especificarse en el protocolo de enrutamiento.

En consecuencia, pueden utilizarse como restricciones en el descubrimiento y selección de rutas. Normalmente, una aplicación puede solicitar una calidad de servicio particular especificando sus requisitos en términos de una o más de las siguientes métricas.

a) El rendimiento mínimo o capacidad (bps) que generalmente se traduce como el rendimiento de datos de la aplicación deseada (Lin, y otros, 1999);

b) El retardo máximo tolerable que se especifica como el máximo retardo tolerable de origen a destino para la transmisión de paquetes de datos (Chlamtac, y otros, 2003);

c) La variación de retardo máxima tolerable (Murazzo, y otros, 2009);

d) Relación de pérdida de paquetes tolerable máxima (PLR) (%), que es el porcentaje aceptable del total de paquetes enviados, que el agente de la capa de transporte no recibe en el nodo de destino del paquete (Kaur, y otros, 2013);

Hipótesis planteada

Se pueden calcular los valores de las métricas propuestas (tasa de descarte de paquetes, demora en la llegada de paquetes y variación de la demora en la llegada de paquetes) a partir de simulaciones de los protocolos de redes ad hoc, a los efectos de expresar las restricciones a los procesos de descubrimiento y selección de rutas establecidos por esos protocolos para modelar la orientación a la calidad de servicio.

Objetivos

1) Ofrecer un informe del estado del arte de las implementaciones para su simulación, de protocolos de enrutamiento para redes ad-hoc seleccionados para esta tesis

2) Seleccionar para este estudio protocolos de enrutamiento reactivos y proactivos a los efectos de visualizar sus ventajas y diferencias con respecto a métricas propuestas. Para ello se utilizarán los protocolos de enrutamiento proactivo OLSR (Optimized Link

State Routing) y DSDV (Destination Sequenced Distance Vector) y el protocolo de enrutamiento reactivo AODV (Ad-hoc On Demand Vector), y

3) Simular redes ad-hoc en el Simulador de Red (NS - Network Simulator), con licencia GNU, a los efectos de evaluar las métricas de pérdidas de paquetes, el retardo, variación del retardo, que se producen en la aplicación de los protocolos mencionados, y en distintos escenarios, que se configuraran como de 10, 15, 25,35 y 50 nodos.

4) Definir parámetros y métricas pertinentes.

Materiales y métodos

Para el objetivo de brindar una reseña de simuladores de red que se están utilizando en la actualidad se realizó una investigación bibliográfica y búsqueda en internet de los diversos sitios y manuales de los paquetes de software pertinentes.

Para alcanzar el objetivo de realizar los procesos de simulación con protocolos de enrutamiento y nodos establecidos previamente, se utilizó el siguiente equipo y software, los cuales exhiben las siguientes características

Procesador	Intel Core i3 – 3120M
CPU	2,50 GHz
Sistema operativo	Windows 10 – Ubuntu 17.04
Simulador de Red	NS3 (Network Simulator 3) version 3.26
Animación de conexiones de red	NetAnim 3.107

Para realizar la simulación presentada en el trabajo se utiliza el simulador de red ns3 (Network Simulator) que es un simulador de eventos discretos y para realizar la colecta de datos se utilizan scripts Python y C ++ .

La simulación se ejecuta durante 200 segundos simulados, de los cuales los primeros 50 segundos se utilizan para el tiempo de puesta en marcha.

Se realizaron 90 simulaciones haciendo variar la cantidad de nodos, se tomaron en grupos de 10, 15, 25, 35 y 50 nodos. A cada uno de los grupos se le aplico los protocolos de enrutamiento previstos. Los nodos se mueven según el modelo RandomWaypointMobilityModel con una velocidad de 1 m/s en algunos casos y hasta 20 m/s en otros casos, y sin tiempo de pausa, dentro de una región de 300m x 1500 m. El módulo Wi-Fi del simulador NS3 es utilizado en modo Ad Hoc con una tasa de 2 Mb / s (802.11b). La potencia de transmisión se establece en 7.5 dBm.

Los protocolos de enrutamiento que se utilizaron son los siquientes: AODV (Ad Hoc On Demand Vector), OLSR (Optimized Link State Routing), DSDV (Destination Sequenced Distance Vector).

Para definir los parámetros y métricas de la simulación de utilizaron los manuales del software utilizados, los cuales fueron explicados en el capítulo de la discusión.

Resultados obtenidos

En función de los objetivos propuestos para este trabajo se puede mencionar que se han logrado los objetivos planteados.

En referencia al estado del arte de los diversos paquetes de software con los cuales se puede lograr una simulación de las redes ad hoc, en el apartado de discusión del presente trabajo se mencionan varias herramientas de simulación. También se detallan los sitios en internet oficiales de cada uno de los productos de software mencionados a los efectos de servir de material de consulta para cada uno de ellos.

En cuanto a la simulación de los protocolos propuestos en una red configurada al efecto, se realizó el trabajo con el simulador de redes 3 (ns3), logrando correr los protocolos propuestos a modo de simulación en redes de hasta 50 nodos. En estas redes simuladas se logró tomar mediciones de las métricas propuestas llegándose a conclusiones sobre las mismas y los protocolos relacionados. También se implementó

en la instalación y puesta en funcionamiento del simulador de red en una máquina virtual con sistema operativo.

En cuanto al objetivo de brindar los parámetros de configuración pertinente se ofrece en el apartado de discusión los detalles de los mismo y su explicación.

Discusión

1. El estándar o normativa IEEE 802.11

1.1. Breve descripción de la evolución del estándar 802.11 (Wikipedia, 2018)

A los efectos de implementar una red inalámbrica se puede seleccionar entre una variedad de propuestas. Cada solución propuesta intenta responder a ciertas necesidades y por lo general, cada propuesta pertenece a una cierta empresa que apostó por esta (Gast, 2005).

Las empresas necesitan que sus productos sean compatibles con los de las otras, de aquí que surja la necesidad de tener un estándar a seguir.

El primer estándar de la familia 802.11 se desarrolló en 1997, el cual consiste en una familia de especificaciones desarrolladas por el IEEE y hacen referencia a las redes LAN inalámbricas.

Los primeros productos se comercializaron en 1999, con el nombre de productos que cumplen la norma Wi-Fi. La norma especificaba dos tasas de transmisión de 1 y 2 Mbit/s sobre infrarrojos (IR) o sobre radiofrecuencia en la banda ISM de 2,4 GHz.

Aunque la transmisión por infrarrojos sigue incluida en la norma no hay en el mercado productos que la utilicen. Permite usar codificación DSSS (Direct Sequence Spread Spectrum) o FHSS (Frequency Hopping Spread Spectrum).

En el año 1999 se publican los estándares 802.11a y 802.11b, aunque la revisión b fue una de los más extendidos en el momento de su llegada. Estos dos primeros estándares

unificados para todo el mercado, presentaban varios problemas de inter-operatividad con el primer estándar 802.11. Se utilizan las bandas de 2,4GHz y de 5GHz.

La revisión 802.11a a la norma original fue ratificada en 1999 y funciona en la banda de 5GHz utilizando 52 subportadoras OFDM (Orthogonal Frequency-Division Multiplexing).

Esta revisión tiene una velocidad teórica máxima de 54 Mbit/s, con velocidades reales de aproximadamente 20 Mbit/s. La velocidad de datos se reduce a 48, 36, 24, 18, 12, 9 o 6 Mbit/s en caso necesario. 802.11a tiene 12 canales no solapados, 8 para red inalámbrica y 4 para conexiones punto a punto.

La revisión 802.11b de la norma original funciona en la banda de 2.4GHz. Esta revisión tiene una velocidad teórica máxima de transmisión de 11 Mbit/s, pero debido al espacio ocupado por la codificación del protocolo CSMA/CA, en la práctica la velocidad máxima de transmisión es de aproximadamente 5.9 Mbit/s para TCP y 7.1 Mbit/s para UDP.

Los dispositivos 802.11b deben mantener la compatibilidad con la norma original IEEE 802.11 (utilizando DSSS). Aunque también utiliza una técnica de ensanchado de espectro basada en DSSS, en realidad la extensión 802.11b introduce CCK (Complementary Code Keying) para llegar a velocidades de 5,5 y 11 Mbit/s (tasa de bit en capa física). La norma también admite el uso de PBCC (Packet Binary Convolutional Coding) como opcional.

Los equipos 802.11a y 802.11b no pueden operar entre ellos.

En el año 2003 se aprueba la revisión 802.11g, que funciona en la banda de 2,4GHz pero añadiendo características más avanzadas, como la multiplexación por división de frecuencias ortogonales, conocida como OFDM, o una innovadora modulación de amplitud en cuadratura, la 64QAM.

Este nuevo estándar no aumentó la velocidad de transmisión de datos, que heredaba los 54Mbps que ya se conseguían con el estándar 802.11a, pero sí se logró que los dispositivos compatibles con él fuesen más baratos de fabricar debido a su mayor sencillez.

La velocidad teórica máxima de transmisión es de 54 Mbit/s, sin embargo, la velocidad real de transferencia es de aproximadamente 22 Mbit/s, similar a la de la norma 802.11a.

En el año 2009 se aprueba la revisión 802.11n que contempla la posibilidad de emplear distintas antenas. Concretamente, hasta cuatro antenas entre el transmisor y el receptor, lo que mejoraba no sólo el rendimiento de la transmisión inalámbrica sino también la robustez, pues podía dejar de funcionar cualquiera de las antenas intermedias y seguir funcionando, de forma más lenta y con peor cobertura, pero, aun así, seguía operativa.

El estándar 802.11n llegó admitiendo las dos bandas de frecuencia que se habían usado hasta ahora. Un dispositivo con 802.11n podía conectarse tanto a redes de 2,4GHz como a redes de 5GHz, y ampliaba el ancho de banda por primera vez desde el desarrollo del estándar. Los 20MHz del inicio se multiplicaban por dos, llegando a los 40Hz, aún con OFDM y 64QAM, pero elevando la velocidad de transmisión hasta los 600Mbps, 11 veces más que el máximo permitido por el estándar 802.11g, reemplazado desde entonces.

La velocidad real de transmisión podría llegar a los 300 Mbit/s (lo que significa que las velocidades teóricas de transmisión serían aún mayores), y debería ser hasta 10 veces más rápida que una red bajo las normas 802.11a y 802.11g, y unas 40 veces más rápida que una red bajo la norma 802.11b.

También se espera que el alcance de operación de las redes sea mayor con esta nueva norma gracias a la tecnología MIMO Múltiple Input Múltiple Output, que permite utilizar

varios canales a la vez para enviar y recibir datos gracias a la incorporación de varias antenas.

En el año 2013 se aprueba la revisión 802.11ac, que implementa la conectividad dual, y que trajo consigo otro importante aumento de velocidad gracias, nuevamente, a una ampliación del ancho de banda.

El estándar 802.11ac pasa a un máximo de 160Mhz, que se atribuye a la utilización de los distintos flujos (streams) MIMO que se especifican en el estándar.

Como añadido a este salto en el ancho de banda, la modulación de amplitud en cuadratura se amplió hasta cuatro veces. El 64QAM dejó paso al 256QAM aumentando así una mayor compactación de los datos, y el envío de más información a través del mismo túnel. Más túneles, más anchos, y más información.

Todo ello, unido al reajuste de señal (beamforming), ya presente en el 802.11n pero más estandarizado y aumentando la calidad de la señal y la velocidad de las conexiones. Beam Forming es una manera de manejar la señal de radiofrecuencia a través de un punto de acceso que utiliza múltiples antenas para transmitir la misma señal. Funciona enviando múltiples señales y analizando las señales de respuesta (feedback) de los dispositivos clientes.

Así, la infraestructura de la red inalámbrica puede ajustar estas señales enviadas y determinar cuál es el mejor camino que deberían tomar para alcanzar un dispositivo cliente. Con el estándar 802.11ac, la velocidad máxima de transmisión de datos en redes 802.11 alcanzó los 7Gbps.

El último estándar aprobado, el 802.11ax, comprende una nueva mejora en robustez y velocidad. Sobre todo, esto último, ya que lleva la velocidad máxima hasta los 10,53Gbps.

Para lograrlo, el 802.11ax emplea la misma banda de frecuencia que otros estándares anteriores, la de 5GHz, y empleará el nuevo MIMO-OFDA. Con este nuevo avance sobre los múltiples canales de entrada y salida, el 802.11ax será capaz de optimizar la transferencia de datos y aumentar la velocidad, se establece hasta ocho canales 256QAM de hasta 160MHz.

1.2. Aspectos de la familia IEEE 802 a parametrizar en el simulador

Como concepto general, la norma IEE 802.11 define los dos niveles inferiores de la arquitectura OSI (el físico y de enlace) de las redes inalámbricas. Es decir, define como se utilizará el canal físico (aire) a través del cual se envía y se recibe la información, y los protocolos utilizados para ello (Wikipedia, 2018).

Las especificaciones IEEE 802 se centran en las dos capas más bajas del modelo OSI porque incorporan componentes físicos y de enlace de datos. Todas las redes 802 tienen un componente de acceso al medio (MAC) y físico (PHY). El componente MAC es un conjunto de reglas para determinar cómo acceder al medio y enviar datos, pero los detalles de la transmisión y la recepción se dejan a la capa física (PHY).

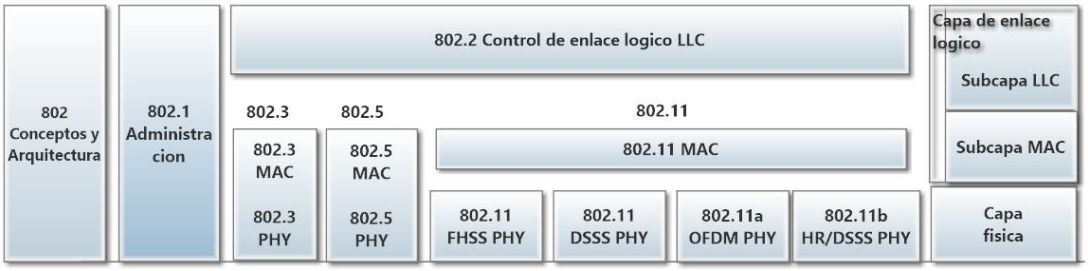

Figura 1 - la familia IEEE 802 y su relación con el modelo OSI (Gast, 2005)

Las especificaciones individuales en la serie de normas 802 están identificadas por un segundo número. Por ejemplo, 802.3 es la especificación para una red de acceso múltiple con detección de portadora con detección de colisión (CSMA / CD), que está relacionada con Ethernet (y con frecuencia se llama erróneamente) Ethernet, y 802.5 es la especificación Token Ring.

Otras especificaciones describen otras partes de la pila de protocolo 802. Como, por ejemplo, 802.2 especifica una capa de enlace común, el Control de Enlace Lógico (LLC), que puede ser utilizado por cualquier tecnología LAN de capa inferior. Las características de gestión y administración para 802 redes se especifican en 802.1. Entre las muchas provisiones de 802.1 están el puente (802.1d) y las LAN virtuales, o VLAN (802.1q).

La norma 802.11 es la especificación de una capa de enlace que puede usar la encapsulación 802.2 LLC. La especificación básica 802.11 incluye el componente MAC 802.11 y dos capas físicas: una capa física de espectro expandido de salto de frecuencia (FHSS) y una capa de enlace de espectro expandido de secuencia directa (DSSS). Las revisiones posteriores a 802.11 agregaron capas físicas adicionales. La normativa 802.11b especifica una capa de secuencia directa de alta velocidad (HR / DSSS). La normativa 802.11a describe una capa física basada en multiplexación por división de frecuencia ortogonal (OFDM).

Para lograr que la norma 802.11 acceda a la red móvil se incorporan una serie de características adicionales en el componente MAC, que agregan complejidad al componente.

El uso de las ondas de radio como una capa física también requiere un componente PHY relativamente complejo. La norma 802.11 divide el componente PHY en dos componentes genéricos: el Procedimiento de Convergencia de Capa Física (PLCP), para asignar las tramas MAC al medio, y un sistema Dependiente del Medio Físico (PMD) para transmitir esas tramas. El componente PLCP abarca el límite de las capas MAC y física, como se muestra en la Figura 2. En 802.11, el PLCP agrega una cantidad de campos a la trama cuando se transmite "en el aire".

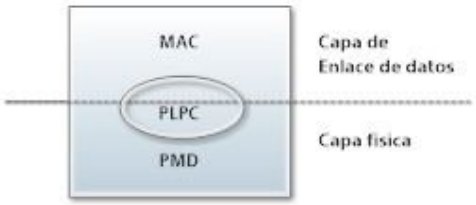

Figura 2 - Componente físico en la norma 802

2. Caracterización de las redes Ad Hoc

2.1. Aspectos generales de las redes ad hoc

Las redes inalámbricas están conectadas a través de enlaces inalámbricos, como frecuencias de radio, o bien, rayos infrarrojos en casos mucho menos frecuente. Han surgido redes inalámbricas espaciales nativas con el propósito de superar las restricciones que son obligatorias para las redes cableadas, ya que permiten instalaciones de red más rápidas y gran movilidad.

Esta red comprende nodos móviles que realizan transferencias entre ellos a través de un medio inalámbrico sin una disposición instalada en forma fija.

Las Redes Ad Hoc se proyectan para operar en ambientes hostiles e irregulares, tales como campos de batalla o zonas de desastre natural, donde se requiere que de forma rápida se puedan desplegar y establecer procesos de comunicación efectivos entre los diferentes elementos o unidades involucradas; así mismo estas redes pueden operar en aeropuertos, campus universitarios, zonas de congresos académicos, etc. donde se necesite un intercambio efectivo de información entre sus nodos.

La red móvil ad hoc no tiene ninguna disposición de instalación y está formada por nodos inalámbricos que viajan con dinamismo y sin restricciones de límite y tienen por ello unas particularidades que las diferencian de las demás (Soyinka, 2010).

Entre las características que cabe destacar (Toh, 2001) (Corson, y otros, 1999) (Sarkar, y otros, 2008) (Chlamtac, y otros, 2003):

a) Debido a la naturaleza inalámbrica de la red ad hoc, los enlaces presentan una capacidad más reducida que los enlaces de las redes cableadas, teniendo como consecuencia, un limitado ancho de banda para su funcionamiento.

b) Las redes ad hoc no precisan la existencia de una infraestructura de red disponible para poder operar, ellas pueden tener de forma autónoma su propio conjunto de protocolos de enrutamiento, mecanismos de gestión de red y procesos de establecimiento de la comunicación e intercambio de información, como consecuencia, no dependen de una infraestructura instalada previamente.

c) En una red ad hoc los nodos se gestionan de forma autónoma, permitiendo que estos se muevan libremente en cualquier dirección y en cualquier momento de forma independiente respecto de los demás nodos, por lo tanto, cuentan con una topología dinámica.

d) Como resultado de efectos de interferencia y canal oculto, de la diversidad de potencia que presentan los dispositivos, de la topología dinámica, entre otros, las redes ad hoc presentan una elevada variabilidad en las condiciones de propagación del canal radio eléctrico.

e) Por lo general, las redes ad hoc realizan sus comunicaciones a través en enlaces de múltiples saltos, cada uno de ellos con diferentes condiciones de propagación y efectuados a través de los diferentes nodos que conforman la red.

f) Se considera que los diferentes nodos operan en los ambientes de alta movilidad e inalámbricos, característicos en las redes ad hoc, por lo tanto, basan su fuente de energía en baterías, las cuales tienen una vida útil limitada.

2.2. Consideraciones técnicas de las redes ad hoc

El escenario presentado por las redes ad hoc permite completar el ambiente de conectividad de las diversas redes existentes. Para configurar y operar una red ad hoc

se hace necesario tener en cuenta algunos desafíos técnicos y tecnológicos para poder obtener beneficios prácticos de su aplicación.

Entre las consideraciones técnicas a resolver en este tipo de redes se encuentran (Toh, 2001):

a) Uno de los aspectos más interesantes y desafiantes en las redes ad hoc se refiere a la búsqueda e implementación de un protocolo de enrutamiento que sea lo suficientemente eficaz y eficiente para operar en las condiciones de alta movilidad y bajo ancho de banda que presentan las redes ad hoc.

b) Uno de los aspectos relevantes actualmente en el entorno de las comunicaciones es la calidad de servicio que debe ofrecer la red a los usuarios que hacen uso de ella y por ende las redes ad hoc no pueden ser indistintas a ello.

El aspecto de calidad de servicio en una red ad hoc es mucho más crítico que en otras redes, pues es mucho más difícil garantizar niveles o límites de retardo de red, variación en el retardo o probabilidad de pérdida de paquetes, etc., cuando se opera en un entorno altamente dinámico y cuyos enlaces de comunicación inalámbricos son más vulnerables a diferentes efectos de propagación, niveles elevados de potencia de los vecinos, condiciones del entorno, etc. tornando bastante critico el concepto de calidad de servicio.

c) Se debe considerar mecanismos que permitan que el número de usuarios de la red se incremente y que la red continué suministrando el nivel de servicio que se espera sin degradarse ni colapsar la operatividad de la red, es decir, se debe contemplar escalabilidad de la implementación de red.

2.3. Aplicaciones y escenarios de uso para las redes ad-hoc

Podemos distinguir dos tipos de escenarios para los cuales se prevé el uso de redes ad-hoc: las redes ad-hoc puras, donde no existe infraestructura, y las redes híbridas, que combinan una red ad-hoc con nodos que proporcionan acceso a redes como Internet.

Organismos como el IRTF (Internet Research Task Force) han identificado que tales arquitecturas híbridas constituyen los escenarios de uso común con mayor potencial para las redes ad-hoc (Yang, y otros, 2003).

2.3.1. Redes ad-hoc puras

Las redes ad-hoc puras se presentan en entornos donde la ausencia de infraestructura coloca a la alternativa ad-hoc como única opción para proporcionar comunicaciones entre dispositivos. En otros casos, si bien es posible la existencia de infraestructura, pero supone un coste que con una red ad-hoc resulta nulo. Algunos ejemplos son los siguientes:

a) Las aplicaciones colaborativas entre usuarios con dispositivos como computadoras portátiles, tabletas o teléfonos móviles, se han convertido en herramientas de trabajo imprescindibles para varios sectores de la sociedad actual.

La conectividad ad-hoc permite que los terminales de cualquier grupo de personas puedan construir una red instantáneamente cuando éstas estén concentradas, como por ejemplo en conferencias o reuniones. Así se posibilita el intercambio de información entre los asistentes, o el uso de aplicaciones colaborativas.

b) Las operaciones de rescate y situaciones de emergencia donde, en algunos casos, establecer comunicaciones mediante los canales habituales como Internet puede no ser posible debido a que tal infraestructura no existe, no es accesible o

incluso ha resultado dañada debido a algún accidente o catástrofe. En tal caso, resulta apropiada una solución ad-hoc.

c) La comunicación entre vehículos tiene como finalidad dotar con la capacidad de establecer comunicación ad-hoc a vehículos de carretera. Esto permite que tales vehículos puedan intercambiarse información, por ejemplo, para notificar accidentes, cortes de tráfico en la carretera o cualquier tipo de evento imprevisto que deba ser conocido por los conductores.

d) Las redes de sensores es otra aplicación de las redes sin infraestructura, y consiste en el uso de una colección de sensores para la realización de medidas. Tales dispositivos pueden crear una red ad-hoc para comunicarse entre sí y transmitir su información.

2.3.2. Redes híbridas

Las redes híbridas disponen de nodos de infraestructura junto a nodos puramente ad-hoc. Dos ejemplos de uso de tales arquitecturas son las redes corporativas y las redes domésticas.

a) Las redes corporativas son un tipo de redes destinadas a cubrir uno o más edificios cercanos, en ámbitos como una empresa o un campus universitario. En este escenario existirían los nodos de infraestructura junto a dos tipos de nodos ad-hoc: nodos fijos (alimentados por red eléctrica y con las características de una computadora personal) y nodos móviles, con poca batería.

Cualquiera de estos dos tipos de dispositivos ad-hoc puede actuar como equipo enrutador, pero los primeros disponen de características apropiadas para ser usados con mayor prioridad. Por otro lado, la infraestructura de la red radicaría en puntos de acceso y puertas de enlace para su salida a Internet.

b) La creación de redes domesticas se fundamentan en que Internet está llegando de una forma progresiva a dispositivos de características muy distintas, de modo que la capilaridad de la red crece. Varias líneas de investigación en domótica apuntan a viviendas cuyos dispositivos están conectados a Internet, de modo que son configurables y accesibles de forma remota (Domínguez, y otros, 2006).

Estas redes hacen referencia tanto a computadoras y teléfonos como a electrodomésticos dotados de una cierta inteligencia, e incluso a objetos cuya vinculación con las redes podría parecer impensable. El uso de interfaces inalámbricas de transmisión y el soporte para conectividad ad-hoc puede resultar una solución clave en estos escenarios.

3. Arquitectura orientada a la calidad de servicios

Los servicios proporcionados por las redes de comunicaciones móviles han experimentado un persistente desarrollo, lo que ha supuesto un gran esfuerzo científico y técnico para garantizar la calidad de esos servicios brindados (Marrone , y otros, 2011).

En el caso de las redes móviles ad-hoc, este esfuerzo es incluso más relevante debido a la complejidad del entorno de aplicación de estas redes, inherentemente dinámico.

La calidad de servicio puede ser descrita como un conjunto de parámetros que describe la calidad de una corriente específica de los datos (Sun, 2014).

Diversas aplicaciones y medios de comunicación comparten está necesidad ya que en la actualidad es indispensable garantizar un adecuado rendimiento, seguridad, beneficios y resultados eficientes para el usuario como aspectos de la calidad de servicio ofrecida (Reid, y otros, 2004).

En el nivel de capa de red se han desarrollado arquitecturas que plantean modelos para implementar calidad de servicio en una red Ad Hoc.

3.1. Arquitectura de Servicios Integrados (IntServ - Integrated Services)

El modelo de Servicios Integrados se diferencia principalmente por la utilización de la reserva de recursos en la red por cada flujo de tráfico, indicando las características que resultarán necesarias como (ancho de banda, retardo, variación de retardo, etc.) (Braden, y otros, 1994)

Para la provisión del servicio cada equipo enrutador del núcleo de la red debe mantener una tabla con el estado de reserva por cada flujo. Un flujo es una secuencia de paquetes IP desde un único transmisor, destinado a un único receptor que pertenece de un único usuario y requiere la misma calidad de servicio.

Para realizar la configuración de la red con una arquitectura de servicios integrados (IntServ) se utiliza un protocolo de reserva de recursos de red como un mecanismo de señalización para llevar los parámetros de calidad de servicio desde el emisor hasta el receptor y poder realizar reservas de recursos a lo largo del trayecto.

Se pueden mencionar como protocolos de reserva de recursos de red el protocolo RSVP (Protocolo de reserva de recursos) (Zhang, y otros, 1997), el protocolo ST2+ (Protocolo de flujo de internet version 2 (Internet Stream Protocol Version 2 (ST2) Protocol Specification - Version ST2+) (Delgrossi, y otros, 1995) , el protocolo YESSIR (Pan, y otros, 1999), el protocolo de Señalización por Ticket (TSP - Ticket Signaling Protocol) (Reid, y otros, 2004), el protocolo de reserva dinámico (DRP - Dynamic Reservation Protocol) (Reid, y otros, 2004) , el protocolo Boomerang (Fehér, y otros, 2000).

La arquitectura de servicios integrados (IntServ) define tres clases de servicios:

Los servicios garantizados son los que brindan condiciones seguras y garantizadas en una comunicación entre extremos. Cada equipo enrutador dentro del trayecto debe ofrecer las garantías solicitadas, aunque a veces esto no es posible por las características del medio físico en que se encuentre.

El servicio de carga controlada es el que provee la misma calidad de servicio que un flujo recibiría si la red estuviera aliviada, pero asegurando que el servicio se conservaría aun cuando la red estuviera sobrecargada, es decir un buen tiempo de respuesta, pero sin garantías estrictas ya que se pueden producir retardos grandes.

El servicio de mejor esfuerzo no garantiza ningún servicio que no se provea debido a la falta de recursos disponibles en la red.

La ventaja de la arquitectura de servicios integrados (IntServ) es la simplicidad, que facilita que toda la red mantenga una política de administración integrada. La posibilidad de crear reglas de calidad de servicios para flujos discretos.

La desventaja de la arquitectura de servicios integrados (IntServ) es que todos los elementos deben mantener el estado e intercambiar mensajes de señalización por cada flujo que manejen, es decir que se necesitan mensajes periódicos de actualización para mantener las sesiones, lo que aumenta el tráfico en la red siendo susceptible a pérdidas de paquetes, por lo tanto, todos los nodos intermedios deben tener RSVP en sus funciones.

3.2. La arquitectura de servicios diferenciados (Diff-Serv - Diferentiated Services)

El modelo de servicios diferenciados (DiffServ) (Blake, y otros, 1998), utiliza un mecanismo de asignación de los flujos múltiples en los niveles de servicio, solucionando de esta forma los problemas de escalabilidad del modelo de servicios integrados, es decir, que está orientado a la priorización de tráfico.

El modelo no establece un canal virtual sobre la red y por lo tanto no realiza ninguna reserva de servicios de red, adoptando cada nodo diferentes comportamientos de acuerdo con sus características disponibles y de acuerdo a su fabricante.

La diferenciación de servicios se lleva a cabo mediante la definición de comportamientos específicos para cada clase de tráfico entre dispositivos de interconexión, a este hecho se le conoce como Comportamiento por Salto (PHB - Per Hop Behavior).

En este modelo se introduce el concepto de PHB, el cual define cuánto tráfico le corresponde a un paquete en particular. En las cabeceras de los paquetes IP, el PHB no es indicado como tal, sino que se usan técnicas de marcado de paquetes mediante los valores del campo Punto de Codificación de Servicios Diferenciados (DSCP - Differentiated Services Code Point) (Nichols, y otros, 1998).

El campo Punto de Codificación de Servicios Diferenciados (DSCP) hace referencia al segundo byte en la cabecera de los paquetes IP que se utiliza para diferenciar la calidad en la comunicación que requieran los datos que se transportan.

De esta manera se soluciona el problema del mayor procesamiento en el mantenimiento de la información de un flujo particular.

La arquitectura de servicios diferenciados (DiffServ) define tres clases de servicios (Blake, y otros, 1998):

El servicio de Reenvío Acelerado es el de mayor calidad ya que ofrece un servicio equivalente a una línea dedicada virtual y debe garantizar un caudal y tasa mínima de pérdida de paquetes y un retardo medio.

El servicio de Reenvío Asegurado asegura un trato preferente, pero no garantiza caudales, retardos, etc. Se definen cuatro clases posibles pudiéndose asignar a cada

clase una cantidad de recursos de red en los equipos enrutadores (ancho de banda, espacio en buffers, etc.).

El servicio de Mejor Esfuerzo se caracteriza por tener a cero los tres primeros bits del DSCP, en este caso los dos bits restantes pueden utilizarse para marcar una prioridad, este servicio no ofrece ningún tipo de garantías.

La ventaja de la arquitectura de servicios diferenciados (DiffServ) es que los flujos múltiples se asignan a un nivel de servicio único por lo que las garantías de calidad de servicio cualitativa se proporcionan a aplicaciones con diferentes niveles de servicio.

La desventaja de la arquitectura de servicios diferenciados (DiffServ) es que los servicios no están estrictamente garantizados, ya que al no haber reserva de ancho de banda entre extremos, cualquier nodo mal configurado que no soporte el PHB propuesto puede descartar flujos de clases establecidas.

3.3. Objetivos del enrutamiento orientado a la calidad de servicio

La selección de una ruta adecuada según los requerimientos de tráfico (Lin, y otros, 1999), teniendo en cuenta la información sobre el estado de la red, es uno de los objetivos importantes del enrutamiento orientado a la calidad de servicio.

Se puede mencionar a la maximización de la utilización de los recursos de la red como otro objetivo importante del enrutamiento orientado a la calidad de servicio.

Por lo tanto, los esquemas de enrutamiento orientados a la calidad de servicio deben presentar soluciones para los mecanismos de distribución de métricas y el algoritmo de selección de ruta.

En general, los protocolos de enrutamiento orientados a la calidad de servicio buscan rutas con recursos suficientes para satisfacer los requisitos de la calidad de servicio de un flujo.

La información sobre la disponibilidad de recursos se gestiona mediante una función de gestión de recursos que utiliza un protocolo de enrutamiento orientado a la calidad de servicio en su búsqueda de rutas factibles con la calidad de servicios requerida.

El protocolo de enrutamiento orientado a la calidad de servicio debe encontrar rutas que consuman recursos mínimos de acuerdo con las métricas de calidad de servicio relevantes.

Para una operación exitosa de enrutamiento orientada a la calidad de servicio, la información de topología se puede mantener en los nodos. La información de la topología debe actualizarse con frecuencia mediante el envío de mensajes de actualización del estado del enlace, que consumen valiosos recursos de red, como el ancho de banda y la alimentación de la batería.

Una red ad hoc se caracteriza porque su topología cambia dinámicamente, lo que puede hacer que la información de la topología se vuelva imprecisa. Este compromiso afecta el rendimiento del protocolo de enrutamiento orientado a la calidad de servicio.

El camino de red que satisface los requisitos de calidad de servicio se debe volver a calcular cada vez que una ruta deja de funcionar, lo que es habitual en la red ad hoc.

El protocolo de enrutamiento orientado a la calidad de servicio debe responder rápidamente en caso de interrupciones de ruta y volver a calcular la ruta inválida u omitir el enlace que no funciona sin degradar el nivel de calidad de servicio.

3.4. Especificaciones de requisitos de calidad de servicio

Para una implementación de enrutamiento orientado a la calidad de servicio eficiente, los requisitos de calidad de servicio deben especificarse en el protocolo de enrutamiento. En consecuencia, pueden utilizarse como restricciones en el descubrimiento y selección de rutas. Normalmente, una aplicación puede solicitar una

calidad de servicio particular especificando sus requisitos en términos de una o más de las siguientes métricas.

a) El rendimiento mínimo o capacidad (bps) que generalmente se traduce como el rendimiento de datos de la aplicación deseada (Lin, y otros, 1999);

b) El retardo máximo tolerable que se especifica como el máximo retardo tolerable de origen a destino para la transmisión de paquetes de datos (Chlamtac, y otros, 2003);

c) La variación de retardo máxima tolerable (Murazzo, y otros, 2009);

d) Relación de pérdida de paquetes tolerable máxima (PLR) (%), que es el porcentaje aceptable del total de paquetes enviados, que el agente de la capa de transporte no recibe en el nodo de destino del paquete (Kaur, y otros, 2013);

3.5 Las métricas en el enrutamiento orientado a la calidad (Becker, 2007)

El enrutamiento es uno de los problemas centrales para el intercambio de datos entre nodos en redes.

Para el enrutamiento orientado a la calidad de servicio, no es suficiente encontrar una ruta desde un origen a uno o varios destinos. Esta ruta también tiene que satisfacer una o más restricciones de calidad de servicio, como por ejemplo ancho de banda o retraso en la entrega de paquetes.

Para garantizar estas restricciones después de encontrar una ruta, se realizan reservas de recursos en los nodos participantes. Sin embargo, en las redes ad hoc no se pueden dar garantías o predicciones debido a las características de este tipo de redes.

Proporcionar una orientación a calidad de servicios en redes móviles ad-hoc es mucho más difícil que en la mayoría de los otros tipos de redes. En primer lugar, debido a la naturaleza de los enlaces de radio, las reservas en los enlaces pueden influirse

mutuamente en un rango de 2 saltos y, por lo tanto, complicar el cálculo y la gestión del ancho de banda y las restricciones de retardo. Además, incluso con las reservas, la disponibilidad de recursos no siempre puede garantizarse debido al aspecto dinámico de la red.

La calidad de servicio en redes informáticas se refiere a la prestación de servicios garantizados en la capa de red, definida en forma de contratos de rendimiento entre la aplicación y el proveedor de servicios. Para negociar dicho contrato, la aplicación define los requisitos de calidad de servicio que contienen información suficiente sobre el tipo y nivel de servicio requerido.

Para especificar los requisitos de calidad de servicio, se utilizan métricas. En las redes, una métrica asocia un valor numérico con una ruta, por lo que se pueden comparar diferentes rutas.

Las métricas de calidad de servicio se pueden clasificar en diferentes grupos, por ej. métricas aditivas, métricas multiplicativas y métricas cóncavas.

Sea $d(n_i, n_j)$ una métrica para el enlace (n_i, n_j) y sea $p = (n_1, n_2, \dots, n_m)$ un camino entre los nodos n_1 y n_m.

Entonces las métricas se denominan como sigue:

Métrica aditiva: $d(p) = d(n_1, n_2) + d(n_2, n_3) + \cdots + d(n_{m-1}, n_m)$

Métrica Multiplicativa: $d(p) = d(n_1, n_2) \times d(n_2, n_3) \times \cdots \times d(n_{m-1}, n_m)$

Métrica Cóncava: $d(p) = min\big(d(n_1, n_2); d(n_2, n_3); \cdots; d(n_{m-1}, n_m)\big)$

Las métricas utilizadas en este trabajo pueden clasificarse como sigue:

La probabilidad de pérdida de paquetes (métrica multiplicativa) se refiere a la probabilidad de que un paquete se pierda en su camino hacia el nodo de destino, por ejemplo. Debido a colisiones, cambios de topología o señales de radio débiles.

La demora en la entrega de paquetes (métrica aditiva) indica el tiempo entre el envío de un paquete desde el nodo de origen y la recepción de este paquete en el nodo de destino.

La variación en la demora en la entrega de paquetes (métrica aditiva) (Gangwar, y otros, 2011) se refiere a la probabilidad de que un paquete se pierda en su camino hacia el nodo de destino, por ejemplo. Debido a colisiones, cambios de topología o señales de radio débiles.

Además de estas métricas, hay otras métricas para redes orientadas a la calidad de servicio:

El ancho de banda (métrica cóncava) se refiere el ancho de banda a lo largo de una determinada ruta y está limitado por el enlace con el ancho de banda más bajo a lo largo de esta ruta.

La métrica “costo” no pertenece a ninguno de los grupos anteriores, ya que es más abstracto y puede ser definido por cualquier función.

El número de saltos (métrica aditiva) representa el número de enlaces en una ruta.

La fluctuación (métrica aditiva) denota la variación entre el tiempo de recepción esperado y real de un paquete.

La energía (métrica aditiva) toma en cuenta la energía necesaria para enviar un paquete desde la fuente hasta el destino.

Alternativamente, la energía también puede manejarse como una métrica cóncava, donde un (móvil) no tiene que proporcionar un cierto nivel de energía, para ser considerado como parte de una ruta.

Otras métricas de QoS incluyen, por ejemplo, la intensidad de la señal (métrica cóncava) y la distancia (métrica aditiva).

Los protocolos de enrutamiento orientados a la calidad de servicio de utilizan subconjuntos de estas métricas.

4. Protocolos de ruteo en las redes ad-hoc

4.1. Introducción

En el ámbito de las redes ad hoc (Murthy, y otros, 2004) el protocolo de enrutamiento debe ser capaz de manejar un gran número de nodos con recursos limitados. El inconveniente principal asociado con el protocolo de enrutamiento hace referencia a la aparición y desaparición de nodos en varias ubicaciones.

Una de las principales ventajas de las redes inalámbricas es la de permitir la movilidad de los usuarios mientras siguen conectados a la red. Sin embargo, en las redes con infraestructura, esa movilidad está limitada por el área de cobertura de la estación base o punto de acceso.

En determinadas situaciones, esta área de cobertura puede ser muy pequeña o, simplemente, podría no existir una estación base, lo cual, no permitirá tener comunicación entre los nodos.

Además, existen distintos escenarios donde podría ser necesario desplegar una red en forma rápida, como pueden ser cuando existen catástrofes naturales, operaciones de rescate, guerras. etc.

Las redes ad-hoc (Sarkar, y otros, 2008) son muy adecuadas para resolver estas situaciones porque se pueden conformar dinámicamente mediante la colaboración de todos los nodos que la integran. No hay necesidad de que exista un acuerdo previo en lo que respecta al rol específico que cada nodo debe asumir, es descentralizada.

Toda la actividad de la red, incluyendo el descubrimiento de la topología de la misma y el envío de mensajes, es realizada por los nodos.

Este tipo de redes tienen una topología dinámica y arbitraria que puede cambiar rápidamente y de manera impredecible debido a que los nodos son libres de moverse aleatoriamente.

La principal consecuencia de esta movilidad es que los enlaces pueden formarse y romperse con mucha frecuencia, lo que implica que la red debe ser auto-configurable y auto-organizada.

Obviamente, el camino entre un origen y un determinado destino que atraviesa varios nodos intermedios puede, repentinamente, modificarse. Cuando esto sucede, la red debe ser capaz de encontrar un nuevo camino en el menor tiempo posible y todos los nodos que la componen deben cooperar para lograrlo. Esta funcionalidad requiere que cada uno de los nodos sea capaz de reenviar datos en nombre de otros nodos, es decir, que funcionen como un equipo enrutador.

En una red inalámbrica del tipo infraestructura, la comunicación entre los nodos que la componen es de un solo salto. Todos los nodos se comunican, directamente, con la estación base. Pero, en una red ad-hoc, puede suceder que dos nodos determinados no se puedan conectar directamente, aunque si lo pueden hacer utilizando nodos intermedios, es decir, la comunicación entre los nodos puede comprender saltos múltiples.

Es por esta característica que se requiere la presencia de un protocolo de enrutamiento que permita encontrar las posibles rutas entre un origen y un destino. Este protocolo debe ser ejecutado en todos los nodos que conforman la red.

En las redes ad-hoc, no existe una única estrategia de enrutamiento posible o válida, sino que cada uno de los distintos enfoques de enrutamiento puede resultar especialmente adecuado para un tipo de escenario en concreto (Singh Som, y otros, 2012) .

Se deben considerar los patrones de movilidad de los nodos, las características de los dispositivos participantes y el tipo de tráfico que éstos deben intercambiar para determinar las condiciones específicas que guiaran la estrategia de enrutamiento a utilizar, estas condiciones pueden diferir significativamente según los casos.

Es necesario reducir la sobrecarga de mensajes de enrutamiento a pesar del número creciente de nodos. Otra cuestión importante es mantener la tabla de enrutamiento pequeña, ya que la tasa a la que aumenta la tabla de enrutamiento afecta a los paquetes de control enviados en la red y, a su vez, afecta a las sobrecargas en los enlaces.

4.2. Clasificación de los protocolos de ruteo

Los protocolos de enrutamiento para redes ad-hoc se pueden clasificar de la siguiente forma: planos (flat), jerárquicos (hierarchical) y de posición geográficamente asistidos (Geographic position assisted) (Kuosmanen, 2002).

En el ruteo plano, o ruteo uniforme, todos los nodos son idénticos en lo que, a su rol y responsabilidad, dentro de la red, se refiere. En cambio, en el ruteo jerárquico, existen nodos que tienen una responsabilidad diferente en el funcionamiento del protocolo de ruteo. Por último, en los protocolos de posición geográfica asistida, los nodos pueden ser ayudados por algún dispositivo especial, como puede ser un dispositivo GPS.

Dentro del ruteo plano, los protocolos se pueden dividir en tres grupos según la forma en que los nodos obtienen y mantienen la información de ruteo: proactivos (controlados por tabla), reactivos (por demanda) e híbridos.

Los protocolos proactivos intentan mantener la información de ruteo a todos los demás nodos de la red consistente y actualizada, de tal forma, que si un nodo necesita una ruta la obtiene inmediatamente.

Como mantienen la información en tablas, también se los conoce como controlados por tabla (table-driven). Estas tablas son enviadas periódicamente a todos sus vecinos. Aun cuando no sean utilizadas, las rutas a todos los destinos posibles se obtienen y mantienen en cada uno de los nodos.

Por otra parte, en un protocolo de enrutamiento reactivo, las rutas son buscadas, únicamente, cuando se necesitan.

El proceso de descubrimiento de ruta es iniciado por el origen y, si se encuentra una ruta hacia el destino deseado, esta se mantiene hasta que el destino deja de ser accesible.

Los protocolos proactivos tienen la ventaja de un menor retardo de extremo a extremo porque las rutas deberán estar disponibles al momento de querer establecer una conexión, pero tienen como desventaja, que generan una gran cantidad de tráfico para mantener las tablas actualizadas, por lo que, pueden presentar problemas de escalabilidad en redes grandes.

Por último, los protocolos híbridos intentan combinar los méritos de estos dos tipos de protocolos y, además, mitigar en gran medida sus limitaciones.

4.3. Protocolos de encaminamiento proactivo

Los protocolos de enrutamiento proactivo (Raut, y otros, 2013) también se conocen como los protocolos de enrutamiento controlado por tabla, utilizan algoritmos de enrutamiento de estado de enlace que inundan la red con la información de enlace a sus vecinos, con una frecuencia determinada.

El objetivo de este intercambio periódico de información de enlace es mantener las tablas de los nodos permanentemente actualizadas, de forma que cuando un nodo necesite enviar datos a otro, ya disponga de la información necesaria para alcanzar a su destino.

En términos generales, este intercambio periódico de información de enlace entre los nodos implica la existencia de una sobrecarga en la red.

Sin embargo, la disponibilidad de rutas en cualquier momento permite que el retardo extremo a extremo asociado a la transmisión de un paquete sea bajo, puesto que cuando un nodo necesite enviar un paquete, ya conoce la ruta hacia dónde debe enviarlo.

En este trabajo de simulación se van a utilizar como representantes de los protocolos proactivos, los protocolos DSDV (Destination-Sequenced Distance Vector Routing) y OLSR (Optimized Link State Routing).

4.4. Protocolos de enrutamiento reactivo

Los protocolos de enrutamiento reactivos o bajo demanda (Raut, y otros, 2013) se diseñaron para reducir los costos del mantenimiento de la información de enlace en cada nodo, en el que incurren los protocolos proactivos.

En oposición al funcionamiento de los protocolos de enrutamiento proactivo, los protocolos reactivos pretenden buscar las rutas bajo demanda entre un nodo origen y un destino.

De este modo, cuando un nodo debe enviar un paquete, inicia un mecanismo que le permite averiguar qué camino debe seguir ese paquete. Una desventaja de este planteamiento es la demora adicional necesaria para descubrir una ruta. Sin embargo, la sobrecarga en el proceso de enrutamiento se puede reducir hasta cero si ningún nodo debe mandar información.

Utiliza el algoritmo de enrutamiento de vector de distancias y establece la ruta a un destino determinado solo cuando un nodo lo solicita iniciando el proceso de descubrimiento de ruta.

Este protocolo funciona en el momento de descubrimiento de rutas y el mecanismo de mantenimiento de rutas. Los protocolos de enrutamiento reactivo tienen el inconveniente de retrasar la búsqueda de rutas hacia un nuevo destino.

En este trabajo de simulación se utiliza como representante de los protocolos reactivos, el protocolo AODV (Ad hoc On-Demand Distance Vector Routing).

4.5. Protocolo OLSR - Optimized Link State Routing protocol

El protocolo de enrutamiento de estado de enlace optimizado (OLSR - Optimized Link State Routing protocol) (Clausen, y otros, 2003) se basa en el algoritmo de estado de enlace. Al ser un protocolo de enrutamiento proactivo, tiene la ventaja de tener la ruta inmediatamente disponible dentro de la tabla de enrutamiento estándar cuando sea necesario.

Su funcionamiento se basa en la elección de nodos especiales denominados retransmisores multipunto (MPR - MultiPoint Relay) que son los encargados de distribuir el tráfico de control por toda la red, indicando los enlaces que existen entre sus nodos.

Debido a la naturaleza de la optimización, se produce una duplicación de inundación mínima en una red altamente conectada.

Cada nodo en la red selecciona un conjunto de nodos vecinos para retransmitir los paquetes y este conjunto de nodos se denomina retransmisores multipunto de ese nodo.

En lugar de la inundación pura, el protocolo OLSR emplea la retransmisión multipunto (MPR) en la red para reducir la sobrecarga posible, la inundación de la transmisión y el intervalo de tiempo para la transmisión de mensajes de control. Este esquema mejora sustancialmente la estrategia de inundación clásica, reduciendo el tráfico de control y, por tanto, consumiendo menos ancho de banda

Solo los MPR reenvían los paquetes de control de tal manera que la información llegue a toda la red y estos MPR sean responsables de declarar la información del estado de enlace.

Cada nodo difunde periódicamente una lista de sus vecinos de un salto para seleccionar los MPR con la ayuda del mensaje de saludo. Los cálculos de ruta se realizan por MPR desde el origen hasta el nodo de destino.

El protocolo OLSR admite tres mecanismos: detección de vecinos, inundación eficiente del tráfico de control y suficiente información de topología.

El protocolo OLSR utiliza dos tipos de mensajes de control: “Hello” y Control de topología (TC). Los mensajes de saludo (“Hello”) se utilizan para encontrar la información sobre el estado del enlace y los vecinos del nodo, mientras que los mensajes de control de topología (TC) se utilizan para la transmisión de información sobre los propios vecinos anunciados incluyen al menos la lista de selección de MPR.

4.6. Protocolo AODV - On-demand Distance Vector routing

El protocolo de enrutamiento vector distancia bajo demanda para redes Ad-hoc (AODV - On-demand Distance Vector routing) (Perkins, y otros, 2003) es un protocolo reactivo, por lo tanto, las rutas se establecen siempre que sea necesario.

Su funcionamiento es bajo demanda, es decir, los nodos, solamente, tienen las rutas que están usando o que han usado recientemente. Se basa en el mecanismo a pedido del descubrimiento de ruta y el mantenimiento de la ruta, más el uso del enrutamiento salto a salto y el número de secuencia de destino.

Provee rutas libres de bucles, aun mientras se están reparando enlaces que dejaron de funcionar, al utilizar un contador llamado número de secuencia de destino (destination sequence number).

Este contador está asociado a cada entrada en la tabla de rutas y está determinado por el nodo destino. La métrica utilizada para comparar dos rutas es la cuenta de saltos (Hop Count).

La tabla de enrutamiento consiste en la información sobre el siguiente salto al destino y un número de secuencia recibido del destino.

Para encontrar una ruta entre dos nodos cualquiera de la red, AODV utiliza un proceso de descubrimiento de ruta por difusión (broadcast).

Este protocolo define tres tipos de mensajes: Requerimiento de ruta (RREQ - Route Request), Respuesta de ruta (RREP - Route Reply) y Error de ruta (REER - Route Error), que son enviados utilizando el protocolo UDP.

Cuando un nodo quiere establecer una conexión con un destino en particular, primero debe verificar que no tiene una ruta válida a ese destino. Si existe una, la utiliza y AODV no hace nada.

En caso contrario, debe iniciar el proceso de descubrimiento de ruta (Path Discovery). El descubrimiento de la ruta se realiza mediante la difusión del mensaje broadcast RREQ a sus vecinos con el número de secuencia de destino solicitado, lo que evita el problema de bucle

Cada uno de los vecinos que reciba este mensaje puede tomar una de las dos acciones: si tiene una ruta válida hacia ese destino, le contesta al origen con un RREP, y, si no, debe reenviar el RREQ, también en forma broadcast, después de incrementar el contador de saltos (Hop Count).

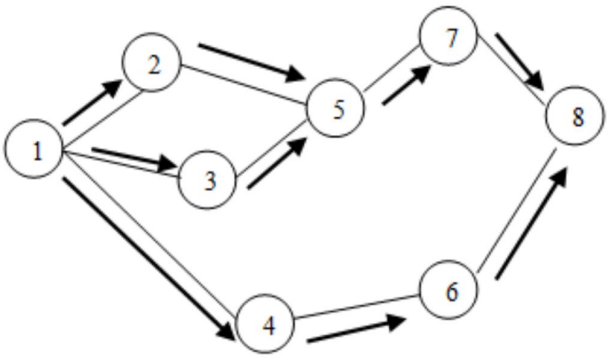

Figura 3 - Propagación del mensaje requerimiento de ruta (RREQ)

Para saber si una ruta hacia un destino es válida (o “fresca”), el nodo debe verificar que el valor del campo número de secuencia de destino (destination sequence number), asociado a la entrada en la tabla de ruteo, es mayor o igual al valor de ese campo en el mensaje RREQ. La respuesta, sea de parte del nodo destino o de un nodo intermedio, es mediante un mensaje RREP.

Cada nodo que reenvía un RREQ debe almacenar, entre otra información, la dirección del nodo del cual recibió el pedido. Al hacer esto, automáticamente, se va estableciendo el camino de regreso hacia el origen (reverse-path). De esta forma, cuando un nodo recibe el correspondiente RREP para esa ruta, sabe a qué vecino se lo debe enviar en forma unicast. Además del camino de regreso (reverse-path), se establece el camino hacia adelante (forward-path).

Cuando RREQ llega al nodo de destino, responde por el paquete RREP.

Mientras el RREP viaja de regreso al origen, cada nodo por donde pasa este mensaje debe almacenar la dirección del vecino del cual lo recibió. Un nodo puede recibir un mismo RREP más de una vez.

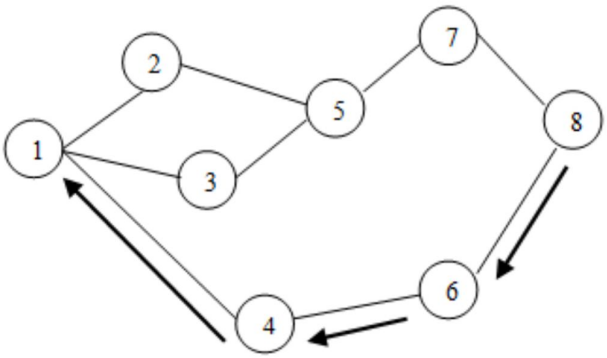

Figura 4 - Camino que toma el mensaje respuesta de ruta (RREP)

Sólo debe retransmitir el primero que recibió. Los demás los debe descartar, salvo que el número de secuencia de destino (destination sequence number) del RREP recibido sea mayor que el enviado anteriormente o, si son iguales, que tenga una mejor métrica. El origen puede comenzar a transmitir los datos tan pronto como reciba el primer RREP y puede, luego, actualizar su información de ruteo si aprende una mejor ruta.

Cuando una ruta deja de funcionar, el nodo envía un mensaje RERR a los vecinos.

Utiliza el mensaje HELLO periódicamente para informar al vecino que el enlace al host está activo. Al recibir el mensaje HELLO, el nodo actualiza la vida útil de la información del nodo en la tabla de enrutamiento.

Al ser un protocolo de enrutamiento plano, el protocolo AODV no necesita ningún sistema administrativo central para manejar el proceso de enrutamiento. También reduce la sobrecarga del mensaje de tráfico de control a costa de una mayor latencia en la búsqueda de nuevas rutas.

Existen varios temporizadores que se utilizan para ir limpiando las entradas del camino hacia atrás (reverse-path) o del camino hacia adelante (forward-path). También existen temporizadores para eliminar de la tabla de rutas todas aquellas rutas que no hayan sido utilizadas en el último periodo de tiempo. La ruta se considera inválida y se elimina.

Si un nodo detecta que un enlace de una ruta activa deja de funcionar, envía un mensaje REER a todos los vecinos que utilizan ese enlace en forma activa.

4.7. Protocolo DSDV - Destination Sequenced Distance Vector

El protocolo de enrutamiento vector distancia con destino secuenciado (DSDV - Destination Sequenced Distance Vector) (Murthy, y otros, 2004) es un protocolo de enrutamiento proactivo, controlado por tablas, que brinda una solución para la ruta más corta entre dos nodos.

Utiliza un número de secuencia de destino para cada entrada de la tabla de enrutamiento de toda la red. Este valor es utilizado para distinguir las rutas viejas de las nuevas y así evitar la formación de bucles de enrutamiento.

Cada nodo en la red mantiene una tabla de ruteo en la cual están registrados todos los posibles destinos dentro de la red y la cantidad de saltos para alcanzar cada uno de esos destinos.

La tabla de enrutamiento se actualiza periódicamente en toda la red para mantener la coherencia en la tabla.

Las actualizaciones de las rutas se pueden hacer de dos maneras: basadas en tiempo (time-driven) o basadas en eventos (event-driven).

Para mantener la vista actualizada de la red, las tablas se intercambian en intervalos de tiempo regulares. Un nodo, cada cierto tiempo, transmite las actualizaciones, que contienen toda su tabla de ruteo, a sus vecinos. También, se envían actualizaciones cada vez que se produce un cambio significativo en su tabla de ruteo.

Para reducir la cantidad de información transportada durante la transmisión de los paquetes de información de enrutamiento, se definen dos tipos de mensajes.

Un tipo de mensaje que lleva toda la información de enrutamiento disponible se llama volcado completo, donde se envía la tabla de rutas completa y otro, llamado volcado incremental el cual transporta información que ha cambiado desde el último volcado completo. Un volcado completo requiere múltiples unidades de datos de protocolo de red (NPDU), mientras que el volcado incremental solo requiere uno para que quepa en toda la información.

Los paquetes de datos son enviados utilizando las tablas de ruteo existentes en cada nodo. Estas tablas contienen la lista de todos los destinos posibles y la cantidad de saltos a cada uno de ellos. Además, cada entrada tiene asociado el correspondiente número de secuencia de destino (destination sequence number), el cual es generado por el nodo destino.

Un nodo puede recibir distintas actualizaciones desde diferentes vecinos. Las rutas recibidas las debe comparar con las que tiene en su tabla de ruteo. Si alguna ruta recibida no existe en su tabla de ruteo, la debe agregar. Pero, si existe, el valor recibido del campo número de secuencia de destino, que está asociado a un destino en particular, se debe comparar con el existente en la tabla de ruteo para el mismo destino.

La ruta con el valor más nuevo será utilizada y las que tienen un valor más viejo son descartadas. Si ese valor es el mismo, la ruta con mejor métrica será la elegida. Las métricas de las rutas recibidas y que son elegidas para agregar a la tabla de ruteo serán incrementadas en uno.

En este protocolo, las actualizaciones llevan a una alta sobrecarga de control durante la alta movilidad debido a enlaces que dejan de funcionar. Otro inconveniente es que el nodo tiene que esperar un mensaje de actualización de la tabla iniciado por el mismo nodo de destino para obtener información sobre un nodo de destino en particular

5. Simuladores de red

Los escenarios de redes completas en tiempo real se implementan y establecen después de una tarea compleja. Es complicado y costoso implementar un componente completo del banco de pruebas que contenga múltiples redes (computadoras, enrutadores y enlaces de datos) para validar y verificar un determinado protocolo de red o un algoritmo de red específico para redes cableadas e inalámbricas.

Con la finalidad de verificar, probar y analizar los resultados de una red se utiliza una metodología generalizada basada en la simulación de red. Una simulación de red es una técnica en la que el programa modela el comportamiento de una red, ya sea calculando la interacción entre las diferentes entidades de la red (hosts, paquetes, etc.) utilizando fórmulas matemáticas, o capturando y reproduciendo realmente observaciones de una red de producción.

La simulación de la red se realiza mediante la herramienta de simulación de red que se conoce como simulador de red. Un simulador de red es una pieza de software o hardware que predice el comportamiento de una red sin una red real presente.

Durante los últimos años, se han desarrollado diversas herramientas de simulación de red en el campo de la comunicación.

Se mencionan aquí varias herramientas de simulación de red, como NS3, NS2, OMNet ++, NetSim, OPNET, REAL, J-Sim, QualNet.

5.1 Simulador de red 3 (ns3 – network simulator 3) (NS3 Official Website)

El proyecto simulador de red 3 (ns3) se inició a mediados de 2006 (Carneiro, 2010). Es un simulador de red de eventos discretos, enfocado principalmente para la investigación y el uso educativo.

El simulador de red 3 (ns3) es un software libre bajo la licencia GNU GPLv2, que está disponible públicamente para su uso.

La herramienta ns3 fomenta el desarrollo de modelos de simulación que sean lo suficientemente realistas como para permitir que ns3 se utilice como un emulador de red en tiempo real que pueda interconectarse con el mundo real y que permita que muchas implementaciones de protocolos del mundo real sean reutilizadas dentro de ns3.

El núcleo de simulación de ns3 admite la investigación tanto en redes IP como no IP. La mayoría de sus usuarios se centran principalmente en simulaciones inalámbricas / IP que incluyen modelos para las capas 1 y 2 y una variedad de protocolos de enrutamiento estáticos o dinámicos, como OLSR (enrutamiento optimizado de estado de enlace) y AODV (enrutamiento bajo demanda basado en vectores de distancias) para aplicaciones basadas en IP (protocolo de internet).

NS3 también admite un programador en tiempo real que facilita una serie de casos de uso de "simulación en el bucle" para interactuar con los sistemas reales. Por ejemplo, los usuarios pueden enviar y recibir paquetes generados por NS3 en dispositivos de red reales, y NS3 puede funcionar como un entorno de interconexión entre máquinas virtuales.

5.2. Simulador de red 2 (ns2 - network simulator 2) (Karl, 2005)

El proyecto de simulador de red 2 se inició en 1996 . Es un simulador de red de eventos discretos de código abierto. Es capaz de simular redes inalámbricas y cableadas. El simulador de red 2 está escrito en código C ++, y se utiliza para modelar el comportamiento de los nodos de simulación. Cuenta con programas escritos en codito OTcl que gestionan la simulación y especifican la topología de la red (Weingartner, y otros, 2009).

Esta arquitectura del simulador se hizo originalmente para evitar recompilaciones innecesarias si se realizan cambios en la configuración de la simulación.

La frecuente recopilación de programas generaba gran consumo de tiempo y ralentizaba todo el ciclo de investigación.

El usuario describe una topología de red escribiendo guiones OTcl, y luego el programa NS principal simula la topología con parámetros específicos. En ns-2, el animador de red (NAM) se utiliza para la vista gráfica de la red.

5.3 OMNet++ (Omnet)

El simulador OMNeT ++ es un entorno y biblioteca de simulación de C ++ extensible, modular y basada en componentes.

Se admiten la funcionalidad específica de dominio para redes de sensores, redes inalámbricas ad-hoc, redes punto a punto, protocolos de Internet, conmutador óptico y red de área de almacenamiento (Varga, y otros, 2008). OMNeT ++ es un entorno de tiempo de ejecución gráfico IDE basado en eclipse.

Las extensiones se manejan en simulación en tiempo real, emulación de red, programación alternativa (Java, C #, C) e integración de base de datos.

5.4 Simulador de red (NetSim – network simulator) (NetSim)

El software NetSim es un simulador de eventos discreto que fue desarrollado por Tetcos. Se utiliza principalmente para la experimentación de laboratorio de red.

El simulador de red se utiliza a modo de interfaz entre el usuario y las bibliotecas de protocolos con las que cuenta el programa y el núcleo de simulación.

Cuenta con facilidades para simular las principales tecnologías como la inalámbrica (LAN, Wi-Max, MANET, WSN, Wi-Fi), MPLS, QoS, VoIP, TCP, IP, etc.

Las bibliotecas de protocolo NetSim están disponibles en código C abierto para su modificación.

5.5 OPNET - Herramientas optimizadas para ingeniería de red (Opnet)

El simulador OPNET – Herramientas optimizadas para ingeniería de red (Optimized Network Engineering Tools) cuenta con a una interfaz gráfica para el diseño de las diversas topología.

El proceso de simulación a través de este entorno se realiza a través de módulos de visualización y provee recopilación de datos de rendimiento de cada simulación.

Este proyecto es un programa de modelos de simulación de eventos discretos de alta fidelidad para tecnologías como IPv6, LTE, MPLS, UMTS, 802.16 (WiMax).

Este software se utiliza para la simulación, el análisis y el diseño de redes, protocolos, dispositivos y aplicaciones (modelado de terreno, sistema en el bucle, visualizador de red 3D, transacción de aplicaciones, modelos de transacciones de aplicaciones).

5.6. REAL - Un enfoque de ingeniería para redes de computadoras (Keshav, S. ;)

El proyeto REAL - Un enfoque de ingeniería para redes de computadoras (An Engineering Approach to Computer Networking), está diseñado para estudiar el comportamiento dinámico de los esquemas de control de flujo y congestión en redes de datos de conmutación de paquetes.

Sus facilidades incluyen la posibilidad de emular varios protocolos de control de flujo y la programación de cinco.

El simulador toma como entrada del escenario con topología, protocolos y parámetros de carga de trabajo y control.

El diseño modular del sistema permite agregar nuevos módulos al sistema.

Se proporciona el código fuente para que los usuarios interesados puedan modificar el simulador. La documentación en línea y el código fuente son parte de la distribución.

Con la simulación se crean estadísticas de salida, como la cantidad de paquetes enviados por cada fuente, la demora en la cola en cada punto de la cola y la cantidad de paquetes descartados y descartados.

La interfaz gráfica permite a los usuarios crear rápidamente escenarios de simulación con interfaces de arrastrar y dibujar.

5.7. JavaSim (J-Sim)

El proyecto JavaSim es un entorno de desarrollo de aplicaciones basado en la arquitectura de software basada en componentes, Arquitectura de Componentes Autónomos (ACA - Autonomous Component Architecture) (Kacer, 2001).

El simulador JavaSim es una biblioteca basada en objetos para simulación orientada a procesos de tiempo discreto.

El área de aplicación principal es la simulación de la red de colas. JavaSim está escrito lenguaje Java.

También admite la interfaz de secuencias de comandos Perl, Tcl (lenguaje de herramientas de comando - Tool Command Language) o Python para la integración.

5.8. QualNet (QualNet)

El software de simulación de red QualNet es una herramienta de planificación, prueba y capacitación que modela el comportamiento de una red de comunicaciones real. Proporciona un entorno enfocado en diseñar protocolos, crear y animar escenarios de red y analizar su rendimiento.

Este simulador es una herramienta basada en una interfaz gráfica para el diseño y visualización de redes.

En el modo de diseño, se puede configurar varias conexiones de red, subredes, definir patrones de movilidad de redes inalámbricas de nodos mediante operaciones intuitivas de clic y arrastre.

El usuario puede personalizar el protocolo de QualNet, el tráfico de la capa de aplicación y los servicios que se ejecutan para la red.

En el modo de visualización, el usuario puede realizar una visualización y análisis en profundidad de un escenario de red y puede generar gráficos dinámicos.

5.9 Resumen de simuladores

Simulador	**Interfaz**			**Emulación**	**código abierto**	**Comercial**	**Lenguaje de programación**	**Plataforma**	**Ultima versión**	**Sitio oficial**
	GUI	**CLI**	**Analizador**							
NS3	×	√	NetAnim	√	√	×	C++, Python	Windows, Linux, Mac OS, Free BSD	NS3.29 (Sept 2018)	http://www.nsnam.org/ns-3-13/download/
NS2	×	√	NAM	√	√	×	C++, Otcl	Windows, Linux, Mac OS, Free BSD	NS2.35 (Nov 2011)	http://www.isi.edu/nsnam/ns/ns-build.html
OMNeT++	√	×	√	√	√	×	C++	Windows, Linux, Mac OS	OMNeT ++ 5.4.1 (Junio 2018)	Omnet https://omnetpp.org/download/
NetSim	√	×	√	Net- Patrol	×	√	C,C++, Java	Windows	NetSim v11 (Ago 2018)	TETCOS https://tetcos.com/index.html
OPNET	√	×	√	√	×	√	C, C++,	Windows	Version 18.7.1 (Junio 2018)	OPNET Technologies Inc. https://support.riverbed.com/content/support/software/steelcentral-npm/modeler-index.html
REAL	√	√	-	×	√	×	C (CLI), Java (GUI)	Sun OS, Linux, Windows	Real5.0 (Agosto 1997)	S. Keshav Cornell University http://www.cs.cornell.edu/skeshav/real/overview.html
JavaSim	√	√	√	Partial	√	×	Java, TCL	Windows, Linux,	Version 1.3 (Jun. 2008)	https://sites.google.com/site/jsimofficial/downloads
QualNet	√	√	√	√	×	√	C++	Windows, Linux	-	Scalable Network Technologies Inc. http://web.scalable-networks.com/content/qualnet

Tabla 1 - Clasificación de herramientas de simulación (Patel, y otros, 2015)

6. Herramienta de simulación (ns3 – network simulator 3)

6.1. Requerimientos considerados al seleccionar el simulador utilizado

El software de simulación seleccionado cumple con la mayoría de los siguientes requerimientos.

a) A los efectos de facilitar la adquisición del software, se consideró que el simulador ns3 es libre y está disponible en forma gratuita.
b) El software simula los protocolos Wi-Fi más comunes (802.11 a/b/g), los siguientes protocolos de enrutamiento para redes ad Hoc: AODV, OLSR, DSDV; y además, la pila de protocolos TCP/IP.
c) El simulador contempla la pérdida de paquetes, el retardo, la variación del retardo, que son las métricas propuestas para este trabajo.
d) La especificación de la potencia de los nodos, la configuración de modelos de propagación, la forma de movilidad de los nodos, así como el posicionamiento de los mismos y la extensión del área de movilidad se realizan con procedimientos relativamente sencillos.
e) El trabajo complementario que se realiza alrededor del software como por ejemplo la creación de nuevos módulos, tutoriales, propuestas de mejoras, análisis de problemas realizados por una comunidad de usuarios son características importantes del software de simulación seleccionado.

6.2. Introducción al ns3

El software ns3 es un simulador de red de eventos discretos que se ha convertido en el sucesor de ns2. El desarrollo de ns-3, fue patrocinado en sus inicios por la NSF-CISE (National Science Foundation, Computer & InformationScience & Engineering).

Principalmente fue desarrollado por investigadores de la Universidad de Washington, del Instituto Tecnológico de Georgia y del grupo de investigación Planète en INRIA. La primera liberación de ns-3.1 fue hecha en junio de 2008 (Wikipedia, 2018).

El simulador ns3 tiene su punto de partida en el trabajo de Mathieu Lacage y Thomas Henderson en el simulador Yet Another Network Simulator (yans) (Lacage, y otros, 2006).

Luego de un análisis del simulador ns2, se detectó un conjunto de deficiencias de diseño que se juzgaron lo suficientemente importantes como para iniciar un nuevo simulador desde cero; entre estas deficiencias se destacaban la falta de versatilidad, debido básicamente a la dependencia entre los modelos, el deficiente uso de las técnicas de programación orientada a objetos, y el rígido acoplamiento entre C++ y OTcl (Patel, y otros, 2015) .

A diferencia de su predecesor, ns3 está desarrollado exclusivamente en C++, aunque permite el interfaz con lenguajes de alto nivel como Python. Los antiguos scripts para ns2 (desarrollados en OTcl) no funcionan en ns3.

El simulador ns3 cuenta con modelos para los elementos que conforman una red de computadoras, por ejemplo, dispositivos de red que representan los dispositivos físicos que conectan un nodo con el canal de comunicación.

Esto puede ser una simple tarjeta de red Ethernet o algo más complejo como un dispositivo inalámbrico IEEE 802.11.

El simulador de red 3 (ns3) cuenta con las implementaciones de los protocolos AODV, OLSR, DSDV. Este software incorpora los aspectos principales de ns2 (licencia libre, desarrollo abierto, colaboración amplia de la comunidad académica), a la vez que trata de superar las carencias y desperfectos de diseño (ampliamente compartidos por su comunidad de usuarios).

La cantidad de documentación en forma de tutoriales, detalles de la API, artículos, con la que cuenta el simulador de red 3 (ns3) es un elemento a destacar, además, es considerado desde su origen un simulador con un buen diseño, potente y flexible.

6.3. Instalación de ns3

El sitio web oficial del simulador es www.nsnam.org, donde están disponibles archivos comprimidos o “tarballs”, como así también, se puede acceder a un repositorio de código fuente. Este software al ser libre, puede ser copiado al entorno de trabajo del investigador muy fácilmente.

Para su instalación se requiere de una máquina de propósito general que tenga instalado Debian/Ubuntu así como los compiladores de C++ (g++).

También se puede utilizar una máquina virtual en Windows como WMware o Virtual Box para instalar sobre ellos, un sistema operativo Debian/Ubuntu.

Una vez descargada la versión deseada de ns3 de su sitio oficial, se obtiene un archivo comprimido con extensión “.tar.bz2” nombrado ns-allinone-3.x, donde la x se refiere a la versión; en este trabajo se utiliza la versión 3.26 (Febrero 2016).

Posteriormente se accede a la consola de Linux, y se posiciona el directorio en la carpeta descompactada del software, a través del comando:

```
$cd /ruta/ns-allinone-3.26
```

Para comenzar la instalación se teclea:

```
$./build.py
```

6.4. Clases Fundamentales de ns3

Es necesario familiarizarse con los diversos módulos y clase de C++ que están implementados en el simulador de red, y que son fundamentales para simular cualquier tipo de topología.

Todas las funcionalidades de ns3 desde la creación de nodos hasta la instalación de aplicaciones están implementadas en clases de C++ que se encuentran en el código del programa.

6.4.1. La clase Node (nodo)

En el ambiente de Internet, un dispositivo informático que se conecta a una red se denomina host o bien, dispositivo o sistema final. Debido a que ns-3 es un simulador de red, no específicamente un simulador de Internet, no se utiliza el término host, ya que está estrechamente relacionado con Internet y sus protocolos.

En el simulador de redes 3 los dispositivos finales o host son llamados nodos. Esta abstracción está representada por la clase Node. La clase Node suministra métodos para administrar la representación de los dispositivos computacionales en simulación.

Se debe pensar en la clase Node como una computadora a la cual se le adicionan funcionalidades, o sea, aplicaciones, pilas de protocolos y tarjetas periféricas con sus respectivos controladores asociados; de esta forma se habilita la computadora para realizar un trabajo determinado (ns-3 project, 2017).

6.4.2. La clase Application (aplicación)

Por lo general, el software de computadora se divide en dos clases genéricas. El software del sistema, también conocido como sistema operativo, organiza diversos recursos informáticos, como memoria, ciclos de procesador, discos, redes, etc., de acuerdo con algún modelo informático. El software del sistema generalmente no usa esos recursos para completar tareas que benefician directamente a un usuario.

Un usuario normalmente ejecutaría una aplicación que adquiere y utiliza los recursos controlados por el software del sistema para lograr algún objetivo.

En el ambiente de simulación de ns-3 no existe un concepto de sistema operativo y, especialmente, ningún concepto de niveles de privilegio o llamadas al sistema. Se

cuenta, sin embargo, con la idea de una aplicación. Al igual que las aplicaciones de software se ejecutan en computadoras para realizar tareas en el "mundo real", las aplicaciones ns-3 se ejecutan en nodos ns-3 para impulsar simulaciones en el mundo simulado.

En el simulador de redes 3 una aplicación es la abstracción básica para un programa de usuario que genere algunas actividades para ser simuladas; se representa por la clase Application, esta clase provee un método para administrar la representación de nuestra versión de aplicación de nivel de usuario en simulaciones (ns-3 project, 2017).

6.4.3. La clase Channel (canal)

En general, una computadora se puede conectar a una red. A menudo, los medios sobre los cuales fluyen los datos en estas redes se denominan canales. Cuando se conecta un cable Ethernet al conector de la red, se está conectando la computadora a un canal de comunicación Ethernet.

En el mundo simulado de ns-3, se conecta un nodo a un objeto que representa un canal de comunicación. En este caso, la abstracción de subred de comunicación básica se denomina canal y está representada en C ++ por la clase Channel.

En el mundo simulado de ns3, se conecta un nodo a un objeto, representando un canal de comunicación.

La clase Channel suministra métodos para administrar objetos en la red de comunicaciones y conectar nodos a ellos.

Una especialización de canal puede representar algo tan simple como un cable o tan complicado como un espacio tridimensional lleno de obstrucciones en el caso de una red inalámbrica.

Se usan versiones especializadas de canal, entre ellas, PointToPointChannel, CsmaChannel Y WifiChannel (ns-3 project, 2017) .

6.4.4. La clase NetDevice (dispositivo de red)

Para conectar una computadora a una red específica, se debe contar con un tipo específico de cable de red y un dispositivo de hardware llamado (en terminología de computadoras) una tarjeta periférica que debe instalarse en la computadora. Si la tarjeta periférica implementa alguna función de red, se denomina tarjeta de interfaz de red o NIC (Network Interface Card). En la actualidad, la mayoría de las computadoras vienen con el hardware de interfaz de red integrado.

Una NIC no funcionará sin un controlador de software para controlar el hardware. En Unix (o Linux), una pieza de hardware periférico se clasifica como un dispositivo. Los dispositivos se controlan mediante controladores de dispositivos y los dispositivos de red (NIC) se controlan mediante controladores de dispositivos de red conocidos colectivamente como dispositivos de red (netdevices). En Unix y Linux, se hace referencia a estos dispositivos de red a través de nombres como eth0.

En el mundo simulado de ns3, el dispositivo de red o Netdevice abarca tanto el software controlador como el hardware simulado. Un Netdevice se instala en un nodo para habilitar la comunicación del nodo con otros nodos a través de canales.

La abstracción NetDevice se representa en el simulador ns3 con la clase NetDevice, la cual proporciona métodos para administrar conexiones a nodos y canales.

Como una NIC Ethernet está diseñada para trabajar en una red Ethernet, el Wi-Fi NetDevice, por ejemplo, está diseñado para trabajar en un Wi-Fi Channel (ns-3 project, 2017).

6.5. Topologías de Ayuda

En una red simulada que sea de gran tamaño se necesita ordenar muchas conexiones entre nodos, dispositivos de red y canales, además de asignar direcciones IP y protocolos entre otros requisitos. Para realizar estas tareas, el simulador de red ns3 proporciona topologías de ayuda que combinan distintas operaciones para hacer más fácil el uso de un modelo determinado.

Ejemplos de topologías de ayuda son:

a) NodeContainer que proporciona una forma conveniente para crear, administrar y acceder a cualquier objeto nodo que haya sido creado para correr una simulación.
b) InternetStackHelper instala una pila de protocolos en los nodos.
c) Ipv4AddressHelper administra la asignación de direcciones IP.
d) Ipv4InterfaceContainer en ns3 hace una asociación entre una dirección IP y un dispositivo usando un objeto Ipv4Interface. Muchas veces, se necesita una lista de los dispositivos de red creados. El Ipv4InterfaceContainer proporciona esta facilidad.

6.6. Clases y módulos del simulador utilizados

La biblioteca de ns3 está organizada en módulos; algunos módulos que se utilizaron se describen a continuación:

a) AODV, DSDV y OLSR: se utilizan para implementar los protocolos de enrutamiento de la red ad Hoc.
b) Applications: aplicaciones.
c) core: En el módulo core se encuentran clases base como la clase objeto, la cual es fundamental en cualquier script de simulación.

d) Internet y network : las bibliotecas Internet y network son usadas para instalar las pilas de protocolos, las direcciones IP, habilitar las clases nodo, paquete, canal, netdevice, socket, aplicaciones, entre otras, todas fundamentales para el correcto funcionamiento de la red.

e) Mobility: El módulo mobility facilita la instalación de diversos modelos de movilidad en los nodos.

f) Propagation: En modulo propagation se encuentran varios modelos de pérdidas de propagación y retardo necesarios para acercar el comportamiento de la red a un entorno real.

g) Wifi: En wifi están todas las clases esenciales para el montaje y configuración de la Red Ad Hoc Móvil.

6.6.1. Modelos de pérdidas de propagación y retardo

En el simulador ns3 existe el módulo “propagation”, dentro del cual se encuentran implementadas varias clases referidas a modelos de pérdidas de propagación y retardo. Las clases utilizadas en este trabajo se explica a continuación:

a) Clase ConstantSpeedPropagationDelayModel

La llamada a esta clase implica que exista un retardo de propagación constante en el canal.

b) Clase FriisPropagationLossModel

El modelo de Friis (de espacio libre) permite calcular la potencia recibida a cierta distancia en condiciones ideales, es decir, sin obstáculos de ninguna naturaleza.

$$\Pr(d) = \frac{PtGtGr\lambda^2}{(4\pi)^2 d^2 L}$$

Donde:

Pt y Pr: potencias transmisora y receptora.

Gt y Gr: ganancias de las antenas transmisora y receptora respectivamente.

L: representa las pérdidas del sistema

λ: es la longitud de onda

Es común seleccionar en las simulaciones Gt=Gr=1 y L=1.

Se puede decir que el modelo de espacio libre representa un rango de comunicación como un círculo alrededor del transmisor. Si el receptor está dentro del círculo, se reciben los paquetes, si está fuera se pierden.

6.6.2. Posicionamiento de los nodos

Para el posicionamiento de los nodos se utiliza la clase positionallocator y de ésta el método principalmente: RandomRectanglePositionAllocator; el cual se explica a continuación:

RandomRectanglePositionAllocator

Mediante este método de posicionamiento se crea un rectángulo de dimensiones especificadas; los nodos se posicionan dentro de éste aleatoriamente, en el rango definido por el usuario.

6.6.3. Modelos de movilidad

El modelo de movilidad es el conjunto de características del movimiento descrito por un nodo móvil dentro de un entorno de estudio. Cada modelo de movilidad intenta representar y ajustarse a alguna situación real que pueda darse en alguna aplicación o entorno de estudio concreto. Cada uno de los terminales móviles se moverá dentro del área de estudio siguiendo el modelo de movilidad seleccionado.

6.6.4. Modelo RandomWaypoint

En este modelo denominado de destino aleatorio o RandomWaypoint (RWP) los nodos de una red Ad Hoc se desplazan en línea recta y con velocidad constante entre dos puntos elegidos al azar dentro de un espacio limitado para los movimientos.

Así, en el caso de un espacio bidimensional, para cada nuevo movimiento, el nodo determina las coordenadas (x e y) del siguiente destino mediante una variable aleatoria uniformemente distribuida entre 0 (origen de coordenadas) y el límite máximo permitido para los desplazamientos en cada dirección (xmáx e ymáx, respectivamente).

Una vez que se alcanza un destino y previamente a elegir el siguiente, el modelo RWP permite pausas, normalmente caracterizadas con un valor constante (Pause). Igualmente, en la mayor parte de las implementaciones, la velocidad constante de cada trayecto se suele decidir también a través de una distribución uniforme en el intervalo (0, Vmáx], siendo Vmáx (expresada en m/s) la velocidad máxima permitida para los nodos (Gupta, y otros, 2013).

El modelo RandomWaypoint (RWP) ofrece una enorme simplicidad y “parsimonia”, esto es, una escasa necesidad de parametrización (limitada básicamente a los valores de Pause, Vmáx y Vmín). Asimismo, al no estar enfocado a ningún escenario de aplicación concreto, su alto grado de abstracción lo convierte en un modelo generalista, idóneo para efectuar pruebas genéricas, al menos iniciales, ante nuevas propuestas de protocolos o mecanismos de gestión de recursos en redes MANET. Estas ventajas, junto con la facilidad y extensión de su implementación, lo han convertido en el modelo más común en la literatura sobre redes Ad Hoc (Kaur, y otros, 2012).

Sin embargo, cabe señalar que son escasos los escenarios reales en los que los nodos de comunicación se mueven de la manera “errática” o a la deriva como lo define el modelo RandomWaypoint (RWP) ; además, este modelo no considera la presencia de obstáculos en el área de desplazamiento ni el hecho de que, en muchos escenarios de

redes Ad Hoc, la movilidad se encuentra fuertemente determinada por la presencia de rutas prefijadas (calles, pasillos, senderos); y tampoco considera posibles correlaciones en los movimientos de los nodos, los cuales se desplazan sin tener en cuenta a los demás. Esta circunstancia no toma en cuenta la evidencia de que en la mayoría de las aplicaciones en donde la ubicación del nodo concuerda con la de una persona, el movimiento sigue pautas fuertemente grupales (provocadas por la formación de pelotones, unidades de salvamento) (Kaur, y otros, 2012).

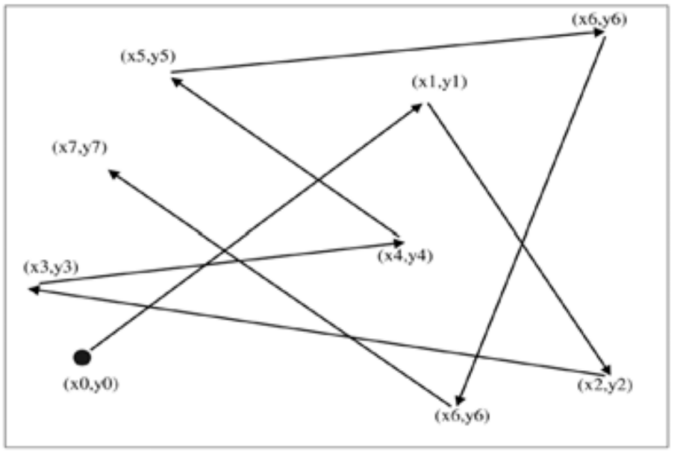

Figura 5 - Patrón de viaje de un nodo usando Random Waypoint

7. Articulación y configuración de la Red Ad Hoc Móvil

Para realizar este trabajo se utilizó el simulador NS-3.26. La simulación se ejecuta durante 200 segundos simulados, de los cuales los primeros 50 se utilizan para el tiempo de puesta en marcha. La cantidad de nodos se tomará en grupos de 10, 15, 25, 35 y 50 nodos.

Los nodos se mueven según el modelo RandomWaypointMobilityModel con una velocidad de 20 m / s y sin tiempo de pausa dentro de una región de 300m x 1500 m. El WiFi es en modo Ad Hoc con una tasa de 2 Mb / s (802.11b). La potencia de transmisión se establecerá en 7.5 dBm.

Para llevar a cabo la articulación y la configuración de la red se necesita, en primer lugar, crear los contenedores de nodos, contenedores de dispositivos, y contenedores de interfaces Ipv4, los cuales son necesarios para almacenar los nodos, dispositivos y direcciones IP, para luego realizar las asociaciones.

Posteriormente se procede a la configuración de las capas PHY y MAC utilizando las clases provistas por ns3. En las siguientes líneas de código: se crea la capa MAC, se configura la red en modo Ad Hoc usando el estándar 802.11b, se crea la capa PHY, se establece el canal añadiéndole el retardo de propagación y las pérdidas, se fija la potencia de transmisión, y finalmente se instala lo realizado en los dispositivos.

```
// se define un contenedor de nodos de red

NodeContainer adhocNodes;

// se crean 5 nodos dentro del contenedor

 adhocNodes.Create (5);

// por defecto se crea como NQoSWifiMac

WifiMacHelper wifiMac ;

// se especifica MAC para soportar una red AdHoc

wifiMac.SetType ("ns3::AdhocWifiMac");

// se especifica PHY con parámetros por defecto

YansWifiPhyHelper wifiPhy = YansWifiPhyHelper::Default ();

// se define un Canal con parámetros por defecto

YansWifiChannelHelper wifiChannel;

// se especifica canal con modelo de velocidad constante de retardo

wifiChannel.SetPropagationDelay ("ns3::ConstantSpeedPropagationDelayModel");
```

```
// se especifica canal con modelo Friis (de espacio libre) de pérdidas de propagación
wifiChannel.AddPropagationLoss ("ns3::FriisPropagationLossModel");
// se especifica una potencia de dBm
double txp = 7.5;
// Nivel de transmisión mínimo disponible (dbm)
wifiPhy.Set ("TxPowerStart",DoubleValue (txp));
// Nivel de transmisión máximo disponible (dbm)
wifiPhy.Set ("TxPowerEnd", DoubleValue (txp));
// Crear un objeto canal
wifiPhy.SetChannel (wifiChannel.Create ());
// definir topologia de ayuda del tipo wifi
WifiHelper wifi;
// especificar la norma 802.11b para la topología de ayuda
wifi.SetStandard (WIFI_PHY_STANDARD_80211b);
// especificar topología de ayuda:
//   gestión de tasa constante,
//  la capa física definida previamente
wifi.SetRemoteStationManager   ("ns3::ConstantRateWifiManager",
            "DataMode", StringValue (phyMode),
            "ControlMode", StringValue (phyMode));
// ****************************************************
```

```
// se define un contenedor de dispositivos de red
// y instalan en los nodos con la capa física y de acceso al medio definidas
anteriormente
// ******************************************************
NetDeviceContainer adhocDevices = wifi.Install (wifiPhy, wifiMac, adhocNodes);
```

A continuación, se instala la pila de protocolos de Internet, IP, UDP, y se añade, además, en este ejemplo que se muestra, el protocolo de enrutamiento OLSR.

```
 InternetStackHelper stack;
 OLSRHelper OLSR;
 stack.SetRoutingHelper (OLSR);
 stack.Install (adhocNodes);
```

Para finalizar la configuración de la red, se realiza la asignación de las direcciones IP en los nodos. Las siguientes líneas declaran un objeto addresshelper (ayudante para direccionamiento IP) diciéndole que debe comenzar la asignación de direcciones IP desde la red 10.5.1.0 usando la máscara 255.255.255.0. Por defecto la dirección asignada comenzará con un incremento de uno monótonamente, o sea, la primera dirección asignada será la 10.5.1.1, la segunda 10.5.1.2 y así sucesivamente hasta instalar las direcciones IP en todos los nodos creados.

```
Ipv4AddressHelper address;
address.SetBase ("10.5.1.0", "255.255.255.0");
interfaces = address.Assign (adhocDevices);
```

7.1. Parámetros de la simulación y recolección de datos

```
/* -*-  Mode: C++; c-file-style: "gnu"; indent-tabs-mode:nil; -*- */
/* ***********************************************************
 * Unne - Universidad Nacional del Nordeste
 * Maestría en sistemas y redes de telecomunicación
 * 2018
 *
 * ********************************************************** */
/*
 * Copyright (c) 2011 University of Kansas
 *
 * 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

 *

 * Author: Justin Rohrer <rohrej@ittc.ku.edu>

 *

 * James P.G. Sterbenz <jpgs@ittc.ku.edu>, director

 * ResiliNets Research Group  http://wiki.ittc.ku.edu/resilinets

 * Information and Telecommunication Technology Center (ITTC)

 * and Department of Electrical Engineering and Computer Science

 * The University of Kansas Lawrence, KS USA.

 *

 * Work supported in part by NSF FIND (Future Internet Design) Program

 * under grant CNS-0626918 (Postmodern Internet Architecture),

 * NSF grant CNS-1050226 (Multilayer Network Resilience Analysis and
Experimentation on GENI),

 * US Department of Defense (DoD), and ITTC at The University of Kansas.

 */



/*

 * This example program allows one to run ns-3 DSDV, AODV, or OLSR under

 * a typical random waypoint mobility model.
```

```
*
* By default, the simulation runs for 200 simulated seconds, of which
* the first 50 are used for start-up time.  The number of nodes is 50.
* Nodes move according to RandomWaypointMobilityModel with a speed of
* 20 m/s and no pause time within a 300x1500 m region.  The WiFi is
* in ad hoc mode with a 2 Mb/s rate (802.11b) and a Friis loss model.
* The transmit power is set to 7.5 dBm.
*
* It is possible to change the mobility and density of the network by
* directly modifying the speed and the number of nodes.  It is also
* possible to change the characteristics of the network by changing
* the transmit power (as power increases, the impact of mobility
* decreases and the effective density increases).
*
* By default, OLSR is used, but specifying a value of 2 for the protocol
* will cause AODV to be used, and specifying a value of 3 will cause
* DSDV to be used.
*
* By default, there are 10 source/sink data pairs sending UDP data
* at an application rate of 2.048 Kb/s each.    This is typically done
* at a rate of 4 64-byte packets per second.  Application data is
```

```
 * started at a random time between 50 and 51 seconds and continues
 * to the end of the simulation.
 *
 * The program outputs a few items:
 * - packet receptions are notified to stdout such as:
 *   <timestamp> <node-id> received one packet from <src-address>
 * - each second, the data reception statistics are tabulated and output
 *   to a comma-separated value (csv) file
 * - some tracing and flow monitor configuration that used to work is
 *   left commented inline in the program
 */

#include <fstream>
#include <iostream>
#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/internet-module.h"
#include "ns3/mobility-module.h"
#include "ns3/wifi-module.h"
#include "ns3/aodv-module.h"
#include "ns3/olsr-module.h"
```

```
#include "ns3/dsdv-module.h"
#include "ns3/dsr-module.h"
#include "ns3/applications-module.h"
#include "ns3/flow-monitor-module.h"
#include "ns3/dsr-fs-header.h"

using namespace ns3;
using namespace dsr;

NS_LOG_COMPONENT_DEFINE ("manet-routing-compare");

class RoutingExperiment
{
public:
  RoutingExperiment ();
  void Run (int nSinks, double txp, std::string CSVfileName);
  //static void SetMACParam (ns3::NetDeviceContainer & devices,
  //                         int slotDistance);
  std::string CommandSetup (int argc, char **argv);

private:
```

```
  Ptr<Socket> SetupPacketReceive (Ipv4Address addr, Ptr<Node> node);
  void ReceivePacket (Ptr<Socket> socket);
  void CheckThroughput ();

  uint32_t port;
  uint32_t bytesTotal;
  uint32_t packetsReceived;

  std::string m_CSVfileName;
  int m_nSinks;
  std::string m_protocolName;
  double m_txp;
  bool m_traceMobility;
  uint32_t m_protocol;
};

RoutingExperiment::RoutingExperiment ()
  : port (9),
    bytesTotal (0),
    packetsReceived (0),
    m_CSVfileName ("manet-routing.output.csv"),
```

```
    m_traceMobility (false),
    // ********************************************************
    // *** protocolo a utilizar. 1.OLSR - 2.AODV - 3.DSDV - 4.DSR **
    // ********************************************************
    // m_protocol (2) // AODV  // rutina original
    m_protocol (3) // DSDV
{
}


static inline std::string
PrintReceivedPacket (Ptr<Socket> socket, Ptr<Packet> packet, Address
senderAddress)
{
  std::ostringstream oss;

  oss << Simulator::Now ().GetSeconds () << " " << socket->GetNode ()->GetId ();

  if (InetSocketAddress::IsMatchingType (senderAddress))
    {
      InetSocketAddress addr = InetSocketAddress::ConvertFrom (senderAddress);
      oss << " received one packet from " << addr.GetIpv4 ();
```

```
    }
  else
    {
      oss << " received one packet!";
    }
  return oss.str ();
}


void
RoutingExperiment::ReceivePacket (Ptr<Socket> socket)
{
  Ptr<Packet> packet;
  Address senderAddress;
  while ((packet = socket->RecvFrom (senderAddress)))
    {
      bytesTotal += packet->GetSize ();
      packetsReceived += 1;
      NS_LOG_UNCOND (PrintReceivedPacket (socket, packet, senderAddress));
    }
}
```

```
void
RoutingExperiment::CheckThroughput ()
{
  double kbs = (bytesTotal * 8.0) / 1000;
  bytesTotal = 0;

  std::ofstream out (m_CSVfileName.c_str (), std::ios::app);

  out << (Simulator::Now ()).GetSeconds () << ","
      << kbs << ","
      << packetsReceived << ","
      << m_nSinks << ","
      << m_protocolName << ","
      << m_txp << ""
      << std::endl;

  out.close ();
  packetsReceived = 0;
  Simulator::Schedule (Seconds (1.0), &RoutingExperiment::CheckThroughput, this);
}
```

```
Ptr<Socket>
RoutingExperiment::SetupPacketReceive (Ipv4Address addr, Ptr<Node> node)
{
  TypeId tid = TypeId::LookupByName ("ns3::UdpSocketFactory");
  Ptr<Socket> sink = Socket::CreateSocket (node, tid);
  InetSocketAddress local = InetSocketAddress (addr, port);
  sink->Bind (local);
  sink->SetRecvCallback (MakeCallback (&RoutingExperiment::ReceivePacket, this));

  return sink;
}

std::string
RoutingExperiment::CommandSetup (int argc, char **argv)
{
  CommandLine cmd;
  cmd.AddValue ("CSVfileName", "The name of the CSV output file name",
m_CSVfileName);
  cmd.AddValue ("traceMobility", "Enable mobility tracing", m_traceMobility);
  cmd.AddValue ("protocol", "1=OLSR;2=AODV;3=DSDV;4=DSR", m_protocol);
  cmd.Parse (argc, argv);
```

```
  return m_CSVfileName;
}

int
main (int argc, char *argv[])
{
  RoutingExperiment experiment;
  std::string CSVfileName = experiment.CommandSetup (argc,argv);

  // blank out the last output file and write the column headers
  // blank out the last output file and write the column headers
  std::ofstream out (CSVfileName.c_str ());
  out << "SimulationSecond," <<
  "ReceiveRate," <<
  "PacketsReceived," <<
  "NumberOfSinks," <<
  "RoutingProtocol," <<
  "TransmissionPower" <<
  std::endl;
  out.close ();
```

```
  // ***************************************
  // *** cantidad de sumideros /potencia de dBm
  // ***************************************
  // int nSinks = 10; // rutina original
  int nSinks = 25;


  double txp = 7.5;


  experiment.Run (nSinks, txp, CSVfileName);
}


void
RoutingExperiment::Run (int nSinks, double txp, std::string CSVfileName)
{
  Packet::EnablePrinting ();
  m_nSinks = nSinks;
  m_txp = txp;
  m_CSVfileName = CSVfileName;


  // ***************************************
```

```
// *** cantidad de nodos
// ************************************
// int nWifis = 50; // rutina original
int nWifis = 50;


// ************************************
// ** tiempo de simulacion
// ************************************
// double TotalTime = 200.0; // rutina original
double TotalTime = 200.0;
std::string rate ("2048bps");
std::string phyMode ("DsssRate11Mbps");
std::string tr_name ("manet-routing-compare");
int nodeSpeed = 20; //in m/s
int nodePause = 0; //in s
m_protocolName = "protocol";


Config::SetDefault  ("ns3::OnOffApplication::PacketSize",StringValue ("64"));
Config::SetDefault ("ns3::OnOffApplication::DataRate",  StringValue (rate));


// ************************************
```

```
  // *** Set Non-unicastMode rate to unicast mode
  // ************************************
  Config::SetDefault ("ns3::WifiRemoteStationManager::NonUnicastMode",StringValue
(phyMode));

  NodeContainer adhocNodes;
  adhocNodes.Create (nWifis);

  // ************************************
  // *** setting up wifi phy and channel using helpers
  // ************************************
  WifiHelper wifi;
  wifi.SetStandard (WIFI_PHY_STANDARD_80211b);

  YansWifiPhyHelper wifiPhy =  YansWifiPhyHelper::Default ();
  YansWifiChannelHelper wifiChannel;
  wifiChannel.SetPropagationDelay ("ns3::ConstantSpeedPropagationDelayModel");
  wifiChannel.AddPropagationLoss ("ns3::FriisPropagationLossModel");
  wifiPhy.SetChannel (wifiChannel.Create ());

  // ************************************
```

```
// *** Add a mac and disable rate control
// ************************************
WifiMacHelper wifiMac;
wifi.SetRemoteStationManager ("ns3::ConstantRateWifiManager",
                              "DataMode",StringValue (phyMode),
                              "ControlMode",StringValue (phyMode));

wifiPhy.Set ("TxPowerStart",DoubleValue (txp));
wifiPhy.Set ("TxPowerEnd", DoubleValue (txp));

wifiMac.SetType ("ns3::AdhocWifiMac");
NetDeviceContainer adhocDevices = wifi.Install (wifiPhy, wifiMac, adhocNodes);

MobilityHelper mobilityAdhoc;
int64_t streamIndex = 0; // used to get consistent mobility across scenarios

ObjectFactory pos;
pos.SetTypeId ("ns3::RandomRectanglePositionAllocator");
pos.Set ("X", StringValue ("ns3::UniformRandomVariable[Min=0.0|Max=300.0]"));
pos.Set ("Y", StringValue ("ns3::UniformRandomVariable[Min=0.0|Max=1500.0]"));
```

```
  Ptr<PositionAllocator> taPositionAlloc = pos.Create ()->GetObject<PositionAllocator>
();
  streamIndex += taPositionAlloc->AssignStreams (streamIndex);


  std::stringstream ssSpeed;
  ssSpeed << "ns3::UniformRandomVariable[Min=0.0|Max=" << nodeSpeed << "]";
  std::stringstream ssPause;
  ssPause << "ns3::ConstantRandomVariable[Constant=" << nodePause << "]";
  mobilityAdhoc.SetMobilityModel ("ns3::RandomWaypointMobilityModel",
                                  "Speed", StringValue (ssSpeed.str ()),
                                  "Pause", StringValue (ssPause.str ()),
                                  "PositionAllocator", PointerValue (taPositionAlloc));
  mobilityAdhoc.SetPositionAllocator (taPositionAlloc);
  mobilityAdhoc.Install (adhocNodes);
  streamIndex += mobilityAdhoc.AssignStreams (adhocNodes, streamIndex);


  AodvHelper aodv;
  OlsrHelper olsr;
  DsdvHelper dsdv;
  DsrHelper dsr;
  DsrMainHelper dsrMain;
```

```
Ipv4ListRoutingHelper list;
InternetStackHelper internet;


switch (m_protocol)
  {
  case 1:
    list.Add (olsr, 100);
    m_protocolName = "OLSR";
    break;
  case 2:
    list.Add (aodv, 100);
    m_protocolName = "AODV";
    break;
  case 3:
    list.Add (dsdv, 100);
    m_protocolName = "DSDV";
    break;
  case 4:
    m_protocolName = "DSR";
    break;
  default:
```

```
    NS_FATAL_ERROR ("No such protocol:" << m_protocol);
  }


if (m_protocol < 4)
  {
    internet.SetRoutingHelper (list);
    internet.Install (adhocNodes);
  }
else if (m_protocol == 4)
  {
    internet.Install (adhocNodes);
    dsrMain.Install (dsr, adhocNodes);
  }


NS_LOG_INFO ("assigning ip address");


Ipv4AddressHelper addressAdhoc;
addressAdhoc.SetBase ("10.1.1.0", "255.255.255.0");
Ipv4InterfaceContainer adhocInterfaces;
adhocInterfaces = addressAdhoc.Assign (adhocDevices);
```

```
  OnOffHelper onoff1 ("ns3::UdpSocketFactory",Address ());
  onoff1.SetAttribute ("OnTime", StringValue
("ns3::ConstantRandomVariable[Constant=1.0]"));
  onoff1.SetAttribute ("OffTime", StringValue
("ns3::ConstantRandomVariable[Constant=0.0]"));

  for (int i = 0; i < nSinks; i++)
    {
      Ptr<Socket> sink = SetupPacketReceive (adhocInterfaces.GetAddress (i),
adhocNodes.Get (i));

      AddressValue remoteAddress (InetSocketAddress (adhocInterfaces.GetAddress
(i), port));
      onoff1.SetAttribute ("Remote", remoteAddress);

      Ptr<UniformRandomVariable> var = CreateObject<UniformRandomVariable> ();
      ApplicationContainer temp = onoff1.Install (adhocNodes.Get (i + nSinks));
      temp.Start (Seconds (var->GetValue (100.0,101.0)));
      temp.Stop (Seconds (TotalTime));
    }

  std::stringstream ss;
```

```
ss << nWifis;

std::string nodes = ss.str ();


std::stringstream ss2;

ss2 << nodeSpeed;

std::string sNodeSpeed = ss2.str ();


std::stringstream ss3;

ss3 << nodePause;

std::string sNodePause = ss3.str ();


std::stringstream ss4;

ss4 << rate;

std::string sRate = ss4.str ();


// ********************************************

// ** para incorporar trace file

// ********************************************

NS_LOG_INFO ("Configure Tracing.");

tr_name = tr_name + "_" + m_protocolName +"_" + nodes + "nodes_" + sNodeSpeed
+ "speed_" + sNodePause + "pause_" + sRate + "rate";
```

```
  // // AsciiTraceHelper ascii; // original

  // // Ptr<OutputStreamWrapper> osw = ascii.CreateFileStream ( (tr_name +
".tr").c_str()); // original

  // wifiPhy.EnableAsciiAll (osw);


  // AsciiTraceHelper asciiTrace; // original

  // Ptr<OutputStreamWrapper> osw = asciiTrace.CreateFileStream ( (tr_name +
".tr").c_str()); // original


  // wifiPhy.EnableAsciiAll (osw);

  // ****************************************


  AsciiTraceHelper ascii;

  MobilityHelper::EnableAsciiAll (ascii.CreateFileStream (tr_name + ".mob"));


  // ****************************************

  // ** para incorporar flow monitor

  // ****************************************

  Ptr<FlowMonitor> flowmon;

  FlowMonitorHelper flowmonHelper;

  flowmon = flowmonHelper.InstallAll ();
```

```
  NS_LOG_INFO ("Run Simulation.");


  CheckThroughput ();


  Simulator::Stop (Seconds (TotalTime));
  Simulator::Run ();


  // ******************************************
  // ** para incorporar flow monitor
  // ******************************************
  flowmon->SerializeToXmlFile ((tr_name + ".flowmon").c_str(), false, false);


  Simulator::Destroy ();
}
```

7.2. Procedimiento en Python para leer los datos desde el archivo xml

```
from __future__ import division
import sys
import os
```

```
try:
    from xml.etree import cElementTree as ElementTree
except ImportError:
    from xml.etree import ElementTree


def parse_time_ns(tm):
    if tm.endswith('ns'):
        return long(tm[:-4])
    raise ValueError(tm)




class FiveTuple(object):
    __slots__ = ['sourceAddress', 'destinationAddress', 'protocol', 'sourcePort',
'destinationPort']
    def __init__(self, el):
        self.sourceAddress = el.get('sourceAddress')
        self.destinationAddress = el.get('destinationAddress')
        self.sourcePort = int(el.get('sourcePort'))
        self.destinationPort = int(el.get('destinationPort'))
        self.protocol = int(el.get('protocol'))
```

```
class Histogram(object):
    __slots__ = 'bins', 'nbins', 'number_of_flows'
    def __init__(self, el=None):
        self.bins = []
        if el is not None:
            #self.nbins = int(el.get('nBins'))
            for bin in el.findall('bin'):
                self.bins.append( (float(bin.get("start")), float(bin.get("width")),
int(bin.get("count"))) )


class Flow(object):
    __slots__ = ['flowId', 'delayMean', 'packetLossRatio', 'rxBitrate', 'txBitrate',
                 'fiveTuple', 'packetSizeMean', 'probe_stats_unsorted',
                 'hopCount', 'flowInterruptionsHistogram', 'rx_duration']
    def __init__(self, flow_el):
        self.flowId = int(flow_el.get('flowId'))
        rxPackets = long(flow_el.get('rxPackets'))
        txPackets = long(flow_el.get('txPackets'))
        tx_duration = float(long(flow_el.get('timeLastTxPacket')[:-4]) -
long(flow_el.get('timeFirstTxPacket')[:-4]))*1e-9
```

```
        rx_duration = float(long(flow_el.get('timeLastRxPacket')[:-4]) -
long(flow_el.get('timeFirstRxPacket')[:-4]))*1e-9
        self.rx_duration = rx_duration
        self.probe_stats_unsorted = []
        if rxPackets:
            self.hopCount = float(flow_el.get('timesForwarded')) / rxPackets + 1
        else:
            self.hopCount = -1000
        if rxPackets:
            self.delayMean = float(flow_el.get('delaySum')[:-4]) / rxPackets * 1e-9
            self.packetSizeMean = float(flow_el.get('rxBytes')) / rxPackets
        else:
            self.delayMean = None
            self.packetSizeMean = None
        if rx_duration > 0:
            self.rxBitrate = long(flow_el.get('rxBytes'))*8 / rx_duration
        else:
            self.rxBitrate = None
        if tx_duration > 0:
            self.txBitrate = long(flow_el.get('txBytes'))*8 / tx_duration
        else:
```

```
            self.txBitrate = None
        lost = float(flow_el.get('lostPackets'))
        #print "rxBytes: %s; txPackets: %s; rxPackets: %s; lostPackets: %s" %
(flow_el.get('rxBytes'), txPackets, rxPackets, lost)
        if rxPackets == 0:
            self.packetLossRatio = None
        else:
            self.packetLossRatio = (lost / (rxPackets + lost))

        interrupt_hist_elem = flow_el.find("flowInterruptionsHistogram")
        if interrupt_hist_elem is None:
            self.flowInterruptionsHistogram = None
        else:
            self.flowInterruptionsHistogram = Histogram(interrupt_hist_elem)


class ProbeFlowStats(object):
    __slots__ = ['probeId', 'packets', 'bytes', 'delayFromFirstProbe']

class Simulation(object):
    def __init__(self, simulation_el):
```

```
self.flows = []
FlowClassifier_el, = simulation_el.findall("Ipv4FlowClassifier")
flow_map = {}
for flow_el in simulation_el.findall("FlowStats/Flow"):
    flow = Flow(flow_el)
    flow_map[flow.flowId] = flow
    self.flows.append(flow)
for flow_cls in FlowClassifier_el.findall("Flow"):
    flowId = int(flow_cls.get('flowId'))
    flow_map[flowId].fiveTuple = FiveTuple(flow_cls)


for probe_elem in simulation_el.findall("FlowProbes/FlowProbe"):
    probeId = int(probe_elem.get('index'))
    for stats in probe_elem.findall("FlowStats"):
        flowId = int(stats.get('flowId'))
        s = ProbeFlowStats()
        s.packets = int(stats.get('packets'))
        s.bytes = long(stats.get('bytes'))
        s.probeId = probeId
        if s.packets > 0:
```

```
                s.delayFromFirstProbe =
parse_time_ns(stats.get('delayFromFirstProbeSum')) / float(s.packets)
            else:
                s.delayFromFirstProbe = 0
            flow_map[flowId].probe_stats_unsorted.append(s)



def main(argv):
    file_obj = open(argv[1])
    print "Reading XML file ",

    sys.stdout.flush()
    level = 0
    sim_list = []
    for event, elem in ElementTree.iterparse(file_obj, events=("start", "end")):
        if event == "start":
            level += 1
        if event == "end":
            level -= 1
            if level == 0 and elem.tag == 'FlowMonitor':
                sim = Simulation(elem)
```

```
            sim_list.append(sim)
            elem.clear() # won't need this any more
            sys.stdout.write(".")
            sys.stdout.flush()
    print " done."



    for sim in sim_list:
        for flow in sim.flows:
            t = flow.fiveTuple
            proto = {6: 'TCP', 17: 'UDP'} [t.protocol]
            print "FlowID: %i (%s %s/%s --> %s/%i)" % \
                (flow.flowId, proto, t.sourceAddress, t.sourcePort, t.destinationAddress,
t.destinationPort)
            if flow.txBitrate is None:
                print "\tTX bitrate: None"
            else:
                print "\tTX bitrate: %.2f kbit/s" % (flow.txBitrate*1e-3,)
            if flow.rxBitrate is None:
                print "\tRX bitrate: None"
            else:
```

```
            print "\tRX bitrate: %.2f kbit/s" % (flow.rxBitrate*1e-3,)
        if flow.delayMean is None:
            print "\tMean Delay: None"
        else:
            print "\tMean Delay: %.2f ms" % (flow.delayMean*1e3,)
        if flow.packetLossRatio is None:
            print "\tPacket Loss Ratio: None"
        else:
            print "\tPacket Loss Ratio: %.2f %%" % (flow.packetLossRatio*100)


if __name__ == '__main__':
    main(sys.argv)
```

8. Análisis comparativo de las métricas propuestas

Con el fin de comparar 3 protocolos en cada una de las situaciones diagramadas, se evaluaron las siguientes métricas: Perdida de paquetes, demora de paquetes y variación en la demora de paquetes.

8.1. Descarte de paquetes

La tasa de descarte de paquetes es un factor preponderante en lo que refiere a servicios en tiempo real, se expresa como el porcentaje de paquetes descartados en el receptor previo al inicio del flujo de datos.

Cuando las pérdidas superan cierto umbral (del orden del 3 %) o cuando se dan en ráfagas ya dejan de ser útiles las técnicas de corrección de errores.

Para un análisis más amplio del descarte de paquetes, en primer lugar, se realizaron simulaciones del mismo protocolo, variando el número de nodos de la red y también la movilidad de los nodos.

En un escenario los nodos se mueven a una velocidad de 1m/s y en otro escenario los nodos se mueven a una velocidad de 20 m/s.

Este procedimiento tiene por objetivo determinar la influencia de la movilidad de los nodos sobre cada protocolo en estudio, teniendo en cuenta la densidad de la red, es decir la cantidad de nodos presentes.

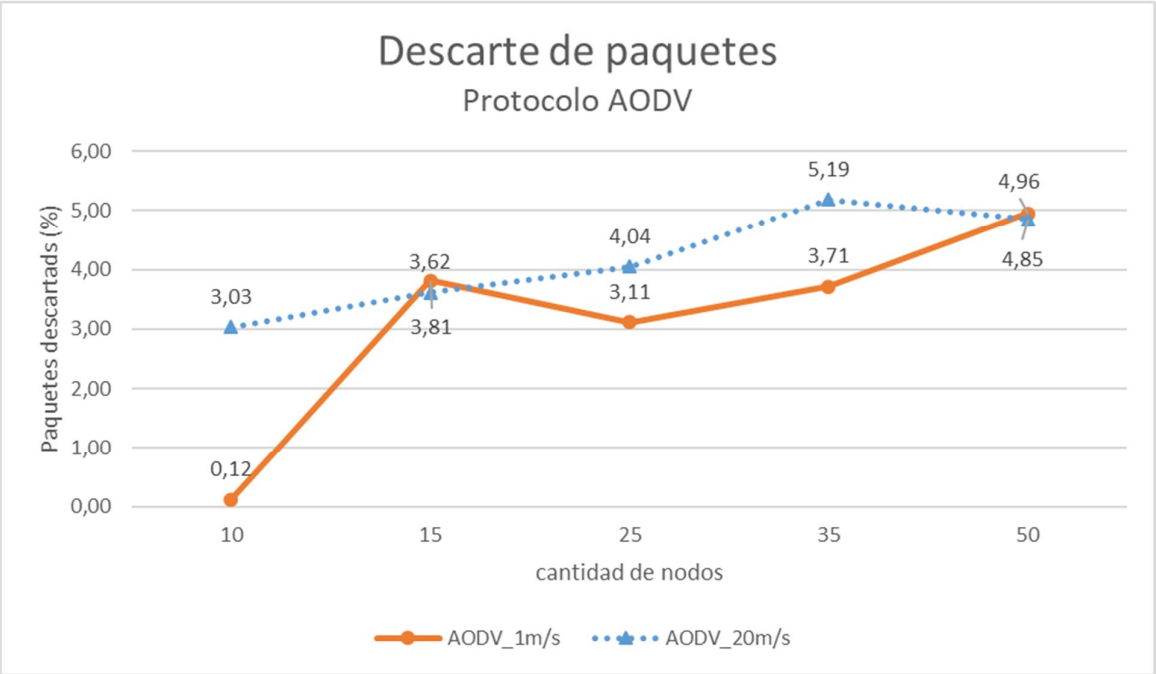

Figura 6 - Tasa de paquetes descartados vs número de nodos para el protocolo

La Figura 6 muestra comportamiento del protocolo AODV cuando se va variando la cantidad de nodos presentes en la red, considerando 2 velocidades distintas de los nodos, a saber: 1 m/s y otra movilidad a 20 m/s.

Cuando se toma en cuenta una velocidad de nodo menor, de 1 m/s del protocolo casi no descarta paquetes, con una red de 10 nodos. Se puede apreciar con la misma red de 10 nodos, el protocolo descarta casi un 3 %. Cuando se va incrementando la cantidad de nodos, en este caso a partir de una red de 15 nodos, el protocolo se va comportando de una manera similar sin importar la velocidad de los nodos, incluso se puede observar que, en una red de 50 nodos, el descarte aproximadamente coincide entre nodos que se están moviendo a velocidades de 1 m/s y otros que se están moviendo a 20 m/s.

El mayor procesamiento de los protocolos reactivos como AODV se relaciona principalmente con el descubrimiento de las nuevas rutas y con las actualizaciones de las rutas utilizables.

Por este motivo lo tanto estos protocolos reactivos funcionan en redes con tráfico ligero y baja movilidad, y se adaptan perfectamente a las redes más grandes con poco ancho de banda y sobrecarga de almacenamiento.

Sin embargo, si la red incluye mucho tráfico con gran cantidad de destinos con alta movilidad, estos protocolos reactivos no reaccionan bien. Esta situación dará como resultado que una gran cantidad de rutas dejaran de funcionar, lo que resultará en descubrimientos de rutas repetidas e informes de errores en la red.

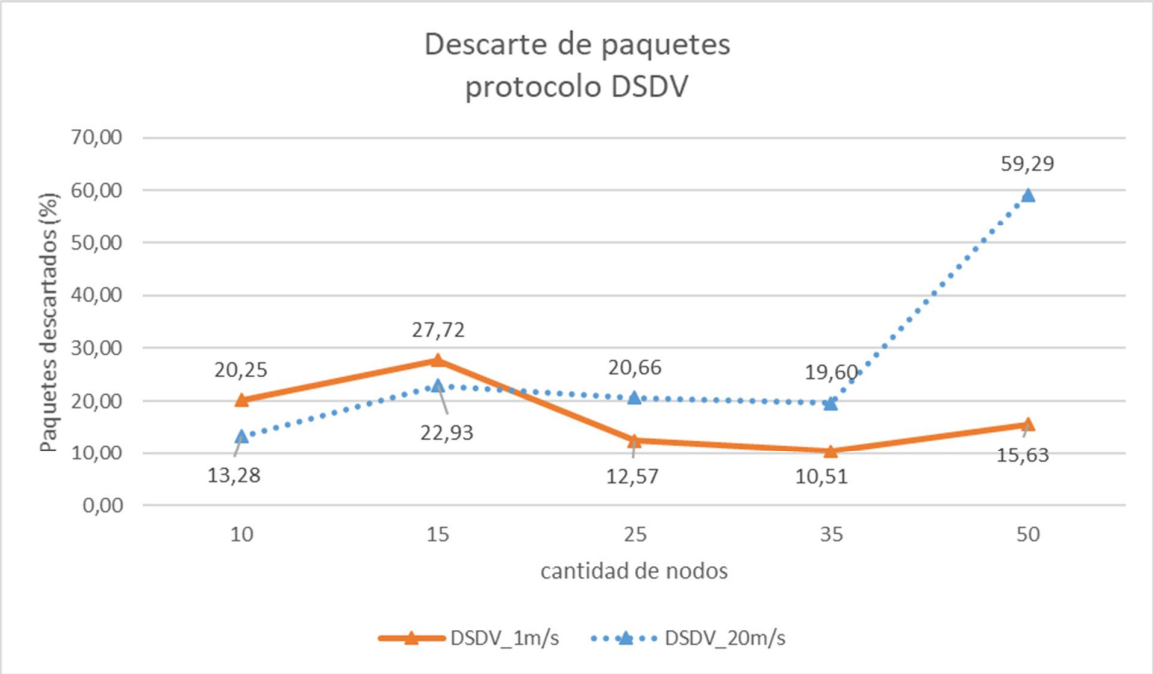

Figura 7 -Tasa de paquetes descartados vs número de nodos para el protocolo DSDV

La Figura 7 muestra comportamiento del protocolo DSDV cuando se va variando la cantidad de nodos presentes en la red, considerando 2 velocidades distintas de los nodos, a saber: 1 m/s y otra movilidad a 20 m/s.

Si tomamos en cuenta una red de 10 nodos, el protocolo se comporta aproximadamente de forma similar, a una velocidad de nodo de nodo de 1 m/s como a una velocidad de

nodo de 20 m/s. Se observa un peor comportamiento del protocolo a partir de una red con 25 nodos cuando los nodos se están moviendo a una velocidad mayor.

A partir de una red de 50 nodos el comportamiento del protocolo es pronunciadamente peor, llegando a una métrica de descarte de paquetes del 50%. El protocolo DSDV se basa en entradas de ruta que van quedando obsoletas con la mayor movilidad de los nodos, y sufre la ausencia de rutas actualizadas recientes

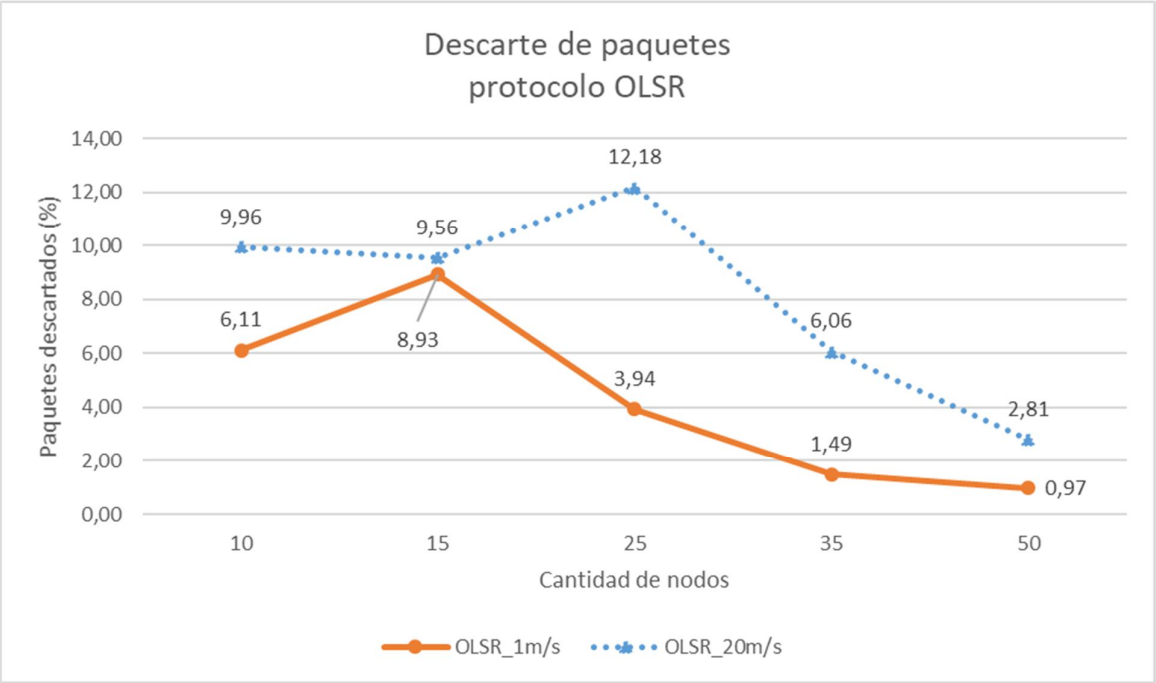

Figura 8 - Tasa de paquetes descartados vs número de nodos para el protocolo OLSR

La Figura 8 muestra comportamiento del protocolo OLSR cuando se va variando la cantidad de nodos presentes en la red, considerando 2 velocidades distintas de todos los nodos de cada una de las redes, a saber: 1 m/s y otra movilidad a 20 m/s.

Si tomamos en cuenta una red de 10 nodos, el protocolo se comporta diferente ante la velocidad de los nodos en cuanto al descarte de paquetes. Hasta una red compuesta por 25 nodos, el comportamiento del protocolo se observa inestable respecto a la velocidad de los nodos y la densidad de la red.

A partir de una red mayor de 25 nodos el comportamiento del protocolo comienza a mejorar, para todas las velocidades en que se están moviendo los nodos, desde 1 m/s hasta 20 m/s.

Debido que el protocolo OLSR es proactivo, el mismo reduce la sobrecarga de control utilizando retransmisores multipunto (MPR - MultiPoint Relay) que propagan las actualizaciones del estado del enlace, y también se obtiene la eficiencia en comparación con el protocolo clásico del estado del enlace cuando el conjunto de retransmisores multipunto seleccionado es lo más pequeño posible.

Pero el inconveniente de este procedimiento radica en que se debe mantener la tabla de enrutamiento para todas las rutas posibles, por lo que no hay diferencia en las redes pequeñas, pero cuando aumenta el número de nodos móviles, también aumenta la sobrecarga de los mensajes de control.

Este inconveniente restringe la escalabilidad del protocolo OLSR. El protocolo OLSR funciona de la manera más eficiente en las redes con mayor número de nodos como se puede apreciar en el gráfico de la Figura 8 .

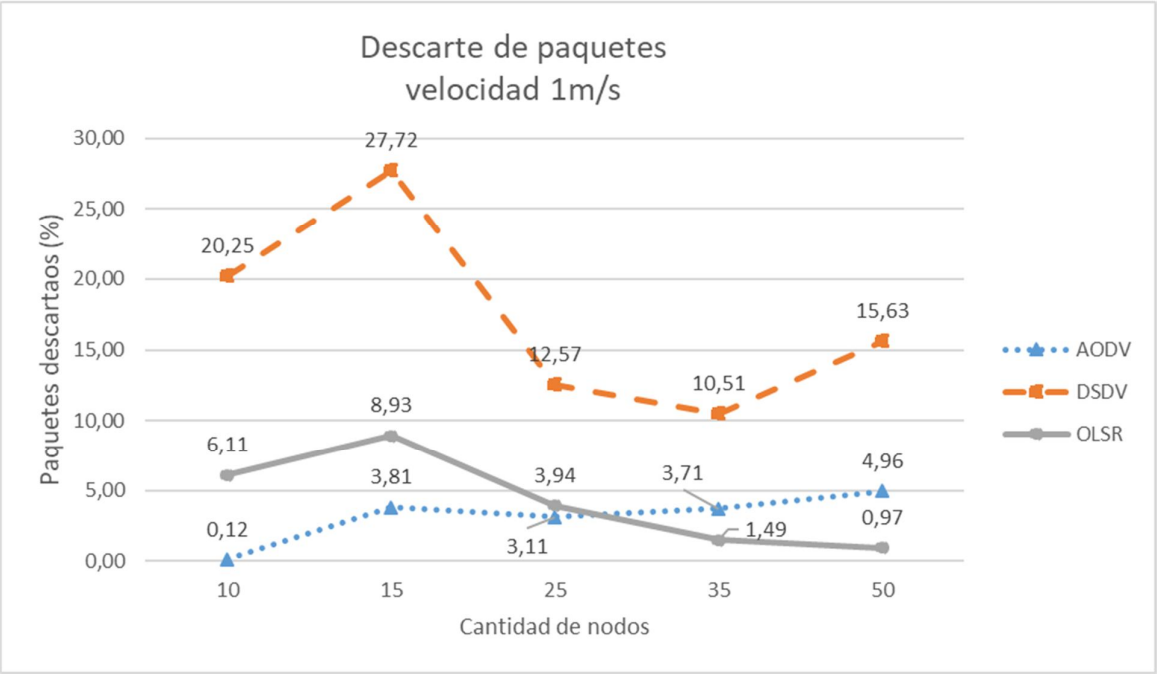

Figura 9 - La tasa de paquetes descartados vs número de nodos para el protocolo los protocolos AODV, DSDV, OLSR

La Figura 9 muestra comportamiento de los protocolos AODV, DSDV, OLSR cuando se va variando la cantidad de nodos presentes en la red, considerando movilidad de todos los nodos de cada una de las redes de hasta 1 m/s.

Si tomamos en cuenta una red de 10 nodos hasta una red de 50 nodos, en todos los casos la tasa de paquetes descartados por el protocolo DSDV es mayor que los protocolos AODV y OLSR.

El protocolo DSDV no ofrece rutas tan estables como los demás protocolos puesto que muchas veces se basa en entradas de ruta obsoletas en ausencia de rutas actualizadas recientes.

El protocolo OLSR ofrece rutas más estables puesto que tiene constantemente actualizada su información de topología a través de sus nodos retransmisores multipunto y AODV es un protocolo reactivo que inicia un proceso de descubrimiento de ruta cuando necesita alguna ruta nueva.

El protocolo OLSR funciona de la manera más eficiente en las redes con mayor número de nodos como se puede apreciar en el gráfico de la Figura 9 , debido a su estructura de nodos de retransmisión multipunto para mantener actualizada la información de topología.

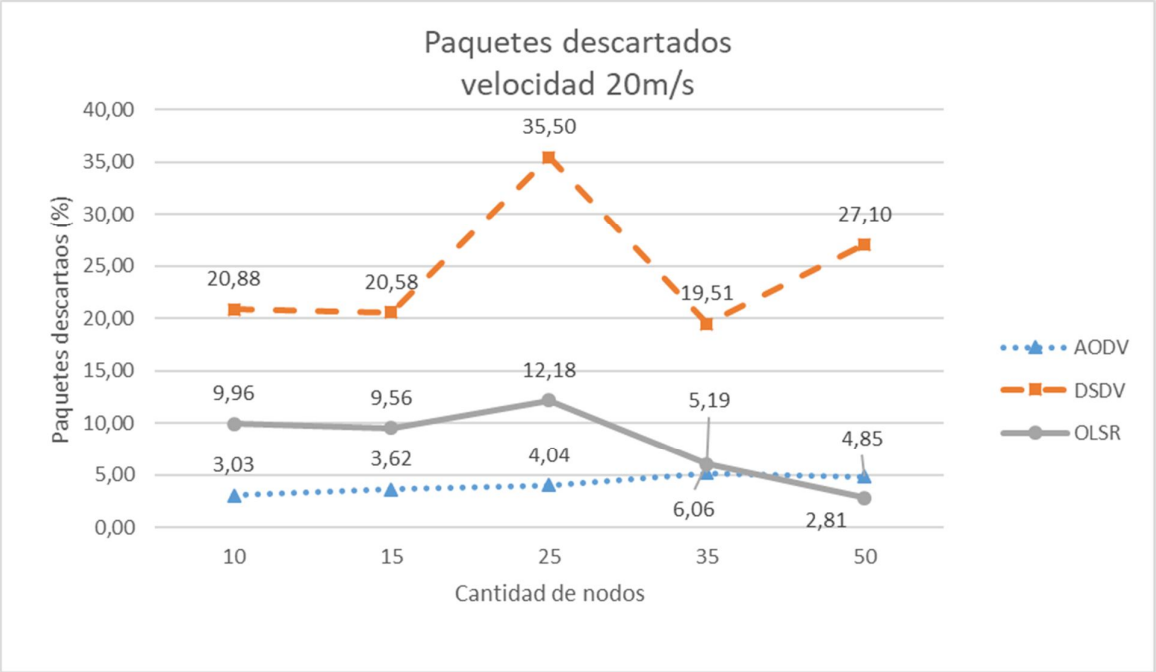

Figura 10 - La tasa de paquetes descartados vs número de nodos para el protocolo los protocolos AODV, DSDV, OLSR

La Figura 10 Figura 8 muestra comportamiento de los protocolos AODV, DSDV, OLSR cuando se va variando la cantidad de nodos presentes en la red, considerando movilidad de todos los nodos de cada una de las redes de hasta 1 m/s.

Si tomamos en cuenta una red de 10 nodos hasta una red de 50 nodos, en todos los casos la tasa de paquetes descartados por el protocolo DSDV es mayor que los protocolos AODV y OLSR.

Esta diferencia se debe a que el protocolo OLSR tiene constantemente actualizada su información de topología a través de sus nodos retransmisores multipunto y AODV es un protocolo reactivo que inicia un proceso de descubrimiento de ruta cuando necesita

alguna ruta nueva, al contrario, DSDV que se basa en entradas de ruta obsoletas en ausencia de rutas actualizadas recientes.

En este caso las rutas se vuelven obsoletas más rápidamente puesto que los nodos se están moviendo a 20 m/s dejando de funcionar las rutas almacenadas en las tablas de rutas de los nodos.

El protocolo OLSR funciona de la manera más eficiente en las redes con mayor número de nodos como se puede apreciar en el gráfico de la Figura 10 , debido a su estructura de nodos de retransmisión multipunto para mantener actualizada la información de topología.

8.2 Retardo o demora en la entrega de paquetes

La demora en la entrega de paquetes o retardo es el tiempo, promedio, que tardan los paquetes de datos en llegar desde el origen al destino teniendo en cuenta todos los posibles retardos que sufre el paquete en su camino, como pueden ser: el almacenamiento en memoria durante el proceso de descubrimiento de ruta, encolamiento en las interfaces, retardos de retransmisión en la capa MAC, propagación y tiempos de transferencia.

Para un análisis más amplio de la demora en la entrega de paquetes, en primer lugar, se realizaron simulaciones del mismo protocolo, variando el número de nodos de la red y también la movilidad de los nodos.

En un escenario los nodos se mueven a una velocidad de 1m/s y en otro escenario los nodos se mueven a una velocidad de 20 m/s. Este procedimiento tiene por objetivo determinar la influencia de la movilidad de los nodos sobre cada protocolo en estudio, teniendo en cuenta la densidad de la red, es decir la cantidad de nodos presentes

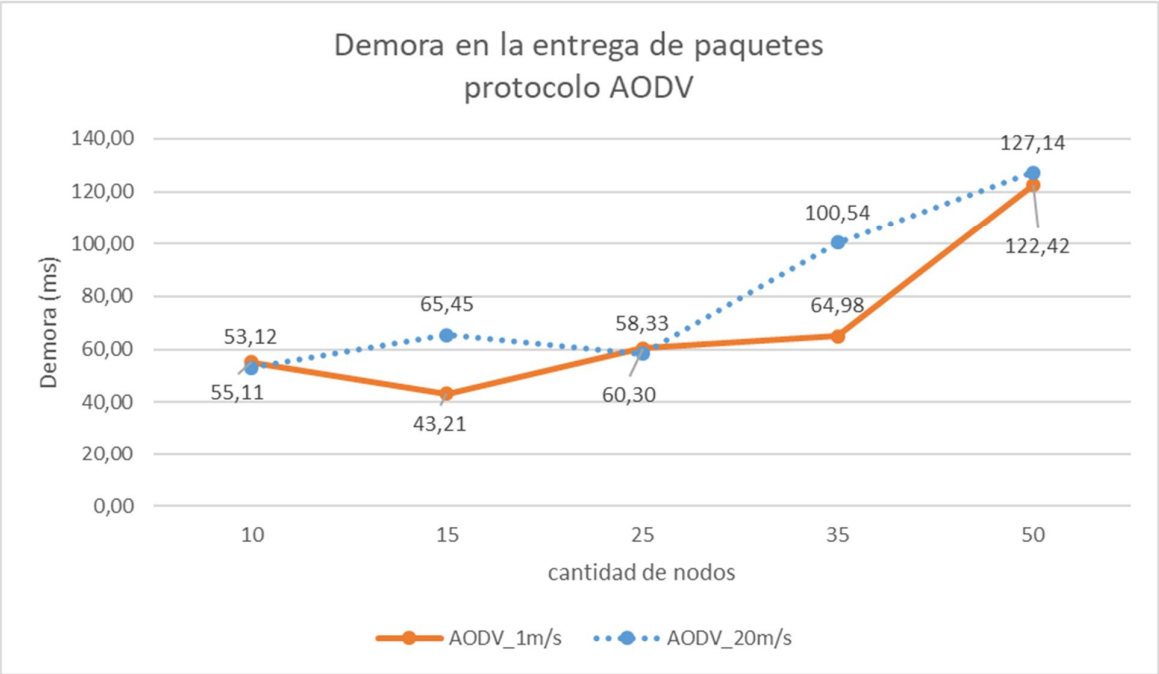

Figura 11 - Demora en la entrega de paquetes vs cantidad de nodos

En los gráficos de la Figura 11 , se puede observar el retardo promedio de los paquetes de datos, desde que son enviados por el origen hasta que son recibidos en el destino.

Se puede observar que el protocolo AODV en redes de aproximadamente 10 nodos se comporta en forma similar con nodos moviéndose a 1 m/s como a 20 m/s.

Este comportamiento se presenta aproximadamente similar hasta redes de 25 nodos aproximadamente.

Con redes más grandes que 25 nodos, el protocolo empeora su comportamiento en forma muy marcada, con respecto a la métrica de demora en la entrega de paquetes. Este empeoramiento es independiente de la velocidad en que se están moviendo los nodos de la red.

El protocolo AODV es necesario descubrir primero la ruta para enviar los datos reales, por lo que la latencia de búsqueda afecta al protocolo AODV.

Por este motivo el protocolo AODV puede sufrir una gran sobrecarga al establecer las nuevas rutas en la red con alta movilidad y retransmisión de paquetes que se agrava aún más, si el entorno de comunicación deficiente.

Su escalabilidad es limitada por el esfuerzo que debe realizar para descubrir nuevas rutas debido a que es un protocolo reactivo.

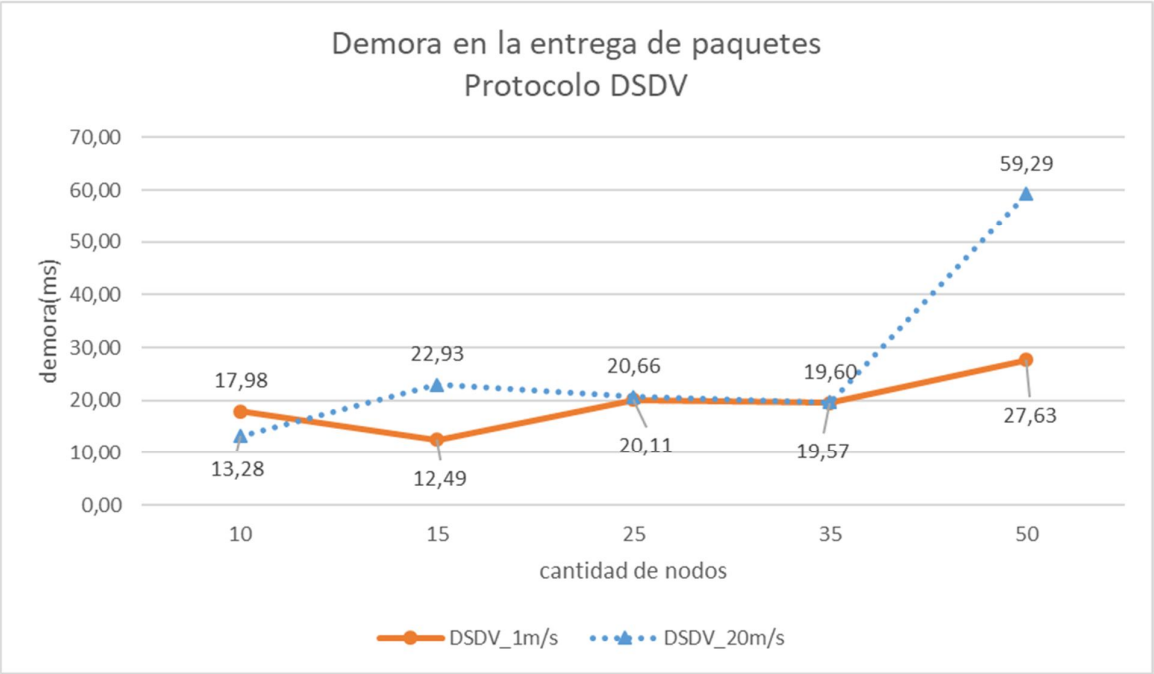

Figura 12 - Demora en la entrega de paquetes vs número de nodos para el protocolo DSDV

La Figura 12 muestra comportamiento del protocolo DSDV cuando se va variando la cantidad de nodos presentes en la red, considerando 2 velocidades distintas de todos los nodos de cada una de las redes, a saber: 1 m/s y otra movilidad a 20 m/s.

Se puede observar que el protocolo DSDV en redes de aproximadamente 10 nodos se comporta en forma similar con nodos moviéndose a 1 m/s como a 20 m/s.

Este comportamiento se presenta aproximadamente similar hasta redes de 35 nodos aproximadamente.

Con redes más grandes que 35 nodos, el protocolo empeora su comportamiento en forma muy marcada, con respecto a la métrica de demora en la entrega de paquetes. Este empeoramiento es más pronunciado cuando se observa las redes cuyos se están moviendo a mayor velocidad.

El protocolo DSDV es un protocolo de enrutamiento proactivo, es decir que utiliza una tabla de enrutamiento para actualizar la información de rutas con otros nodos. El tiempo de envío de los mensajes de actualización depende de los cambios de topología o bien de un periodo determinado de tiempo.

Sin ningún cambio de topología, este protocolo envía los mensajes de actualización a sus vecinos cada 15 segundos. Al cambiar la topología, también envía mensajes de actualización, independientemente del período de tiempo. En este proceso de actualización participan todos los nodos de la red, es por este motivo que al incrementarse los nodos de la red más allá de un número, en este caso 35 nodos, el protocolo empeora su tiempo de demora de entrega de paquetes.

A esto se debe agregar que con nodos moviéndose a mayor velocidad mayor cantidad de enlaces dejan de funcionar, haciéndose necesaria una actualización más frecuente de la topología.

Por este motivo, es aún mayor el tiempo de demora para redes de más de 50 nodos, cuando los nodos se están moviendo más rápidamente.

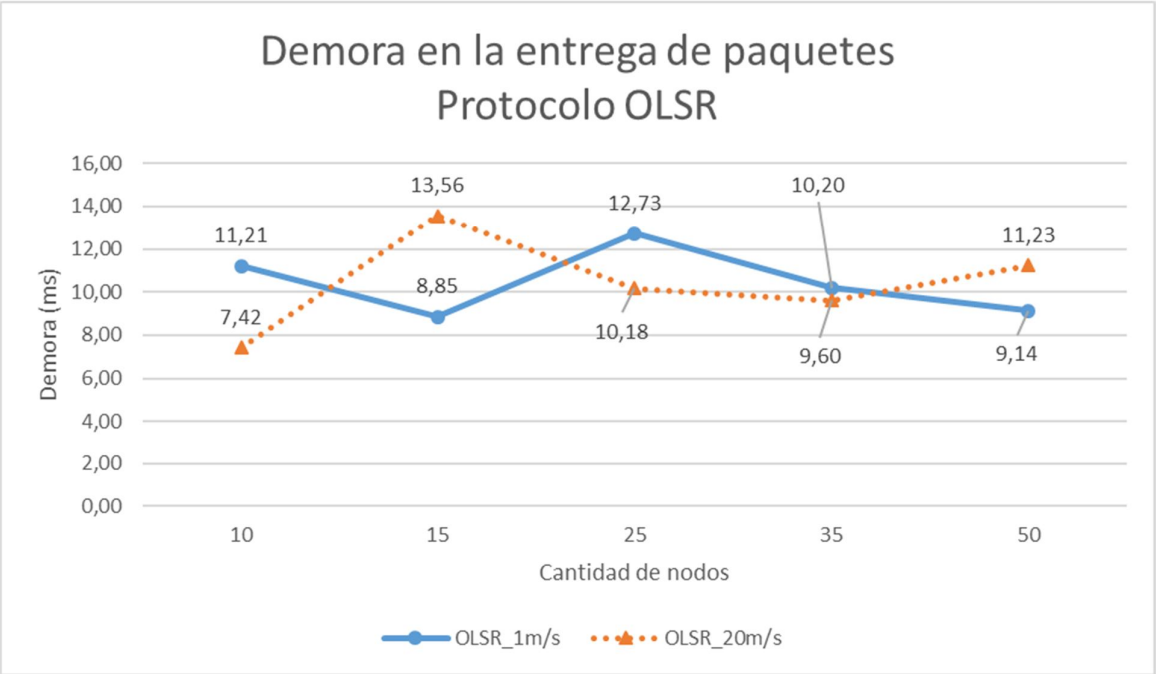

Figura 13 - Demora en la entrega de paquetes vs número de nodos para el protocolo OLSR.

La Figura 13 muestra comportamiento del protocolo OLSR cuando se va variando la cantidad de nodos presentes en la red, considerando 2 velocidades distintas de todos los nodos de cada una de las redes, a saber: 1 m/s y otra movilidad a 20 m/s.

Se puede observar que el protocolo OLSR tiene un comportamiento similar en redes de 10 nodos hasta 50 nodos, incluso considerando la velocidad de nodo como variable desde 1 m/s hasta 20 m/s de los nodos que componen las redes.

Debido que el protocolo OLSR es proactivo, mantiene actualizada la información de topología, sin embargo, el protocolo reduce la sobrecarga de control utilizando retransmisores multipunto (MPR - MultiPoint Relay), los cuales propagan las actualizaciones del estado del enlace.

Cuando una ruta deja de funcionar, debido a la movilidad del nodo, y es necesario encontrar nuevas rutas, el protocolo OLSR siempre tiene a mano información de topología actualizada, por lo tanto, las nuevas rutas se pueden calcular inmediatamente

cuando se informa de un salto de ruta, no ocasionando mayores demoras en la entrega de paquetes.

Por este motivo no se observan mayores diferencias de demora en la entrega de paquetes, en las simulaciones realizadas.

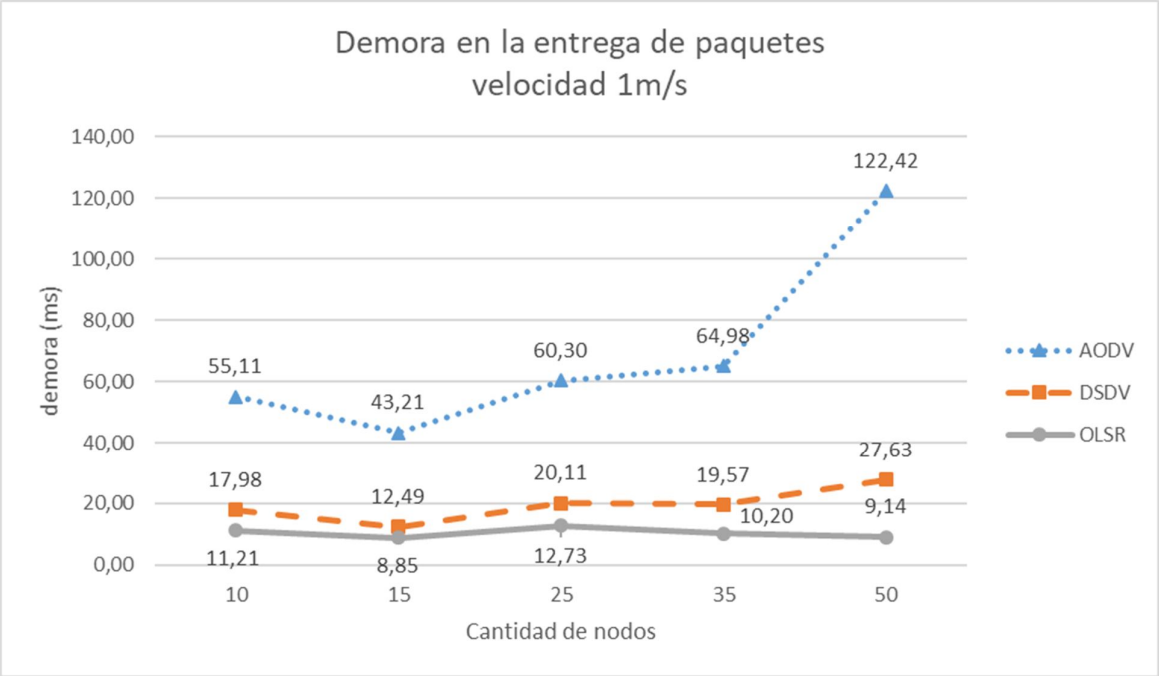

Figura 14 - La demora en la entrega de paquetes vs número de nodos para el protocolo los protocolos AODV, DSDV, OLSR

La Figura 14 muestra comportamiento de los protocolos AODV, DSDV, OLSR cuando se va variando la cantidad de nodos presentes en la red, considerando movilidad de todos los nodos de cada una de las redes de hasta 1 m/s.

Si tomamos en cuenta una red de 10 nodos hasta una red de 50 nodos, en todos los casos el comportamiento del protocolo AODV es peor que el de los protocolos restantes.

A partir de redes de 35 nodos o más el comportamiento del protocolo AODV comienza a deteriorase significativamente.

Esta diferencia se debe a que el protocolo OLSR tiene constantemente actualizada su información de topología a través de sus nodos retransmisores multipunto y DSDV es

un protocolo proactivo que tiene información sobre la topología almacenada, sin embargo, DSDV se basa en entradas de ruta obsoletas cuando no cuenta con rutas actualizadas recientes.

El protocolo AODV debe iniciar el protocolo de descubrir rutas cuando necesita enviar paquetes de datos. Para ello inicia un procedimiento de inundación con paquetes de requerimiento de ruta, motivo por el cual cuando la red crece en número de nodos, el comportamiento del protocolo empeora drásticamente.

El protocolo OLSR funciona de la manera más eficiente que otros protocolos, en las redes con mayor número de nodos como se puede apreciar en el gráfico de la Figura 14 , debido a su estructura de nodos de retransmisión multipunto para mantener actualizada la información de topología.

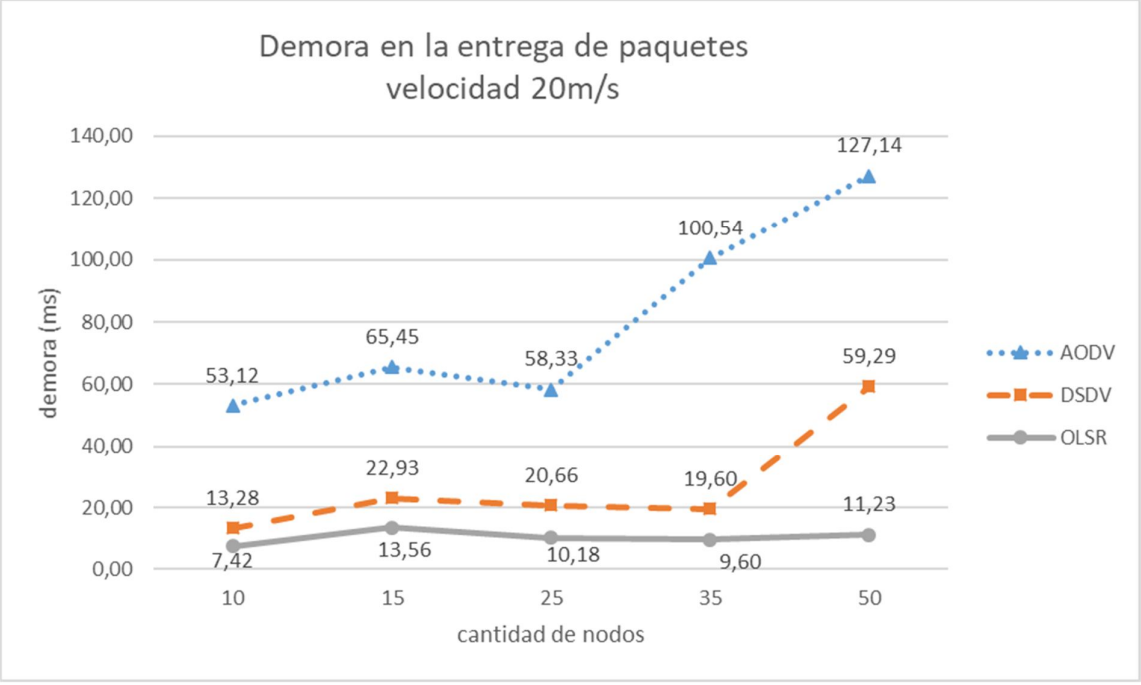

Figura 15 - La demora en la entrega de paquetes vs número de nodos para el protocolo los protocolos AODV, DSDV, OLSR

La Figura 15 muestra comportamiento de los protocolos AODV, DSDV, OLSR cuando se va variando la cantidad de nodos presentes en la red, considerando movilidad de todos los nodos de cada una de las redes de hasta 20 m/s.

Si tomamos en cuenta una red de 10 nodos hasta una red de 50 nodos, en todos los casos el comportamiento del protocolo AODV es peor que el de los protocolos restantes.

A partir de redes de 25 nodos o más el comportamiento del protocolo AODV comienza a deteriorase significativamente.

Esta diferencia se debe a que el protocolo OLSR tiene constantemente actualizada su información de topología a través de sus nodos retransmisores multipunto y DSDV es un protocolo proactivo que tiene información sobre la topología almacenada, sin embargo, DSDV que se basa en entradas de ruta obsoletas cuando no cuenta con rutas actualizadas recientes.

El hecho de que los nodos se estén moviendo, hace que mayor número de rutas dejen de funcionar, lo que desencadena antes el proceso de inundación para la búsqueda de nuevas rutas que utilizar el protocolo AODV.

El protocolo AODV debe iniciar el protocolo de descubrir rutas cuando necesita enviar paquetes de datos. Para ello inicia un procedimiento de inundación con paquetes de requerimiento de ruta, motivo por el cual cuando la red crece en número de nodos, el comportamiento del protocolo empeora.

El protocolo OLSR funciona de la manera más eficiente que los otros protocolos, en las redes con mayor número de nodos como se puede apreciar en el gráfico de la Figura 15 , debido a su estructura de nodos de retransmisión multipunto para mantener actualizada la información de topología.

8.3 Variación del retardo

Otro de los factores que afectan la calidad percibida por el usuario es la variación del retardo entre las llegadas de paquetes subsiguientes, se denomina también jitter, el cual es causado por una variación temporal de las condiciones de la red, cambios de ruta, etc. Para que un protocolo sea eficiente, debe ser lo más bajo posible.

El receptor debería recibir los paquetes a intervalos constantes, para poder regenerar de forma adecuada la señal original. Dado que el jitter es inevitable, los receptores disponen de un buffer de entrada, con el objetivo de suavizar el efecto de la variación de las demoras.

Este buffer recibe los paquetes a intervalos variables, y los entrega a intervalos constantes.

El uso de buffers en el receptor también ayuda al problema de la llegada de tramas fuera de orden, chequeando los números de secuencia de éstas.

Es de hacer notar que este buffer agrega una demora adicional al sistema, ya que debe retener paquetes para poder entregarlos a intervalos constantes. Cuanta más variación de demoras (jitter) exista, más grande deberá ser el buffer, y por lo tanto, mayor demora se introducirá al sistema.

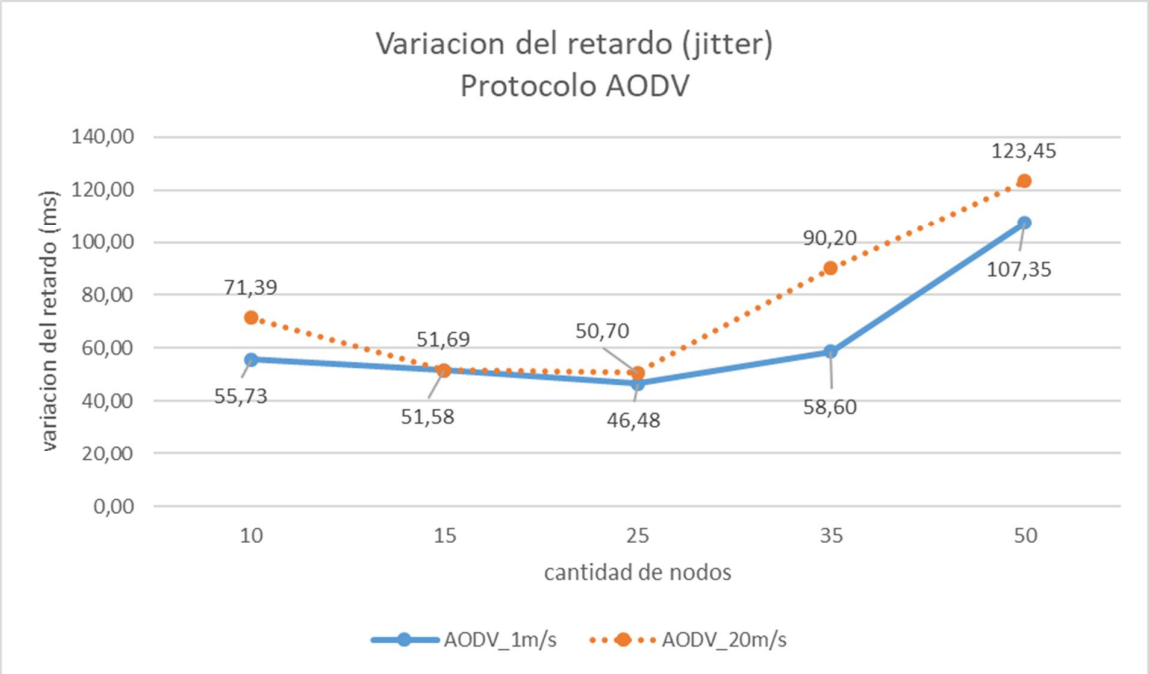

Figura 16 - Variación de la demora en la llegada de paquetes vs número de nodos para el protocolo AODV

La Figura 16 muestra comportamiento del protocolo AODV cuando se va variando la cantidad de nodos presentes en la red, considerando 2 velocidades distintas de todos los nodos de cada una de las redes, a saber: 1 m/s y otra movilidad a 20 m/s.

Se puede observar que el protocolo AODV tiene un comportamiento similar en redes de 10 nodos hasta 25 nodos, incluso considerando la velocidad de nodo como variable desde 1 m/s hasta 20 m/s de los nodos que componen las redes.

Con redes más grandes que 25 nodos, el protocolo empeora su comportamiento en forma muy marcada, con respecto a la métrica de variación en la demora en la entrega de paquetes. Este empeoramiento es más pronunciado cuando se observa las redes cuyos nodos se están moviendo a mayor velocidad.

El protocolo AODV utiliza como uno de sus criterios de selección de ruta el trayecto que posea una menor demora en la llegada de paquetes, sin tener en cuenta la carga de esas rutas, en lo que se refiere a tráfico. Esto ocasiona que las rutas más frecuentemente seleccionadas rápidamente se desbalanceen en cuanto a carga de tráfico, ocasionando problemas de variación en la llegada de los paquetes a su destino.

Este problema se acentúa cuando la cantidad de nodos en la red crece, verificándose un mayor tráfico en las rutas disponibles. Si a esto le agregamos un mayor tráfico de control debido al descubrimiento de rutas ocasionado por rutas que dejan de funcionar debido a la movilidad de los nodos, se puede verificar en el grafico el empeoramiento del comportamiento del protocolo para esta métrica.

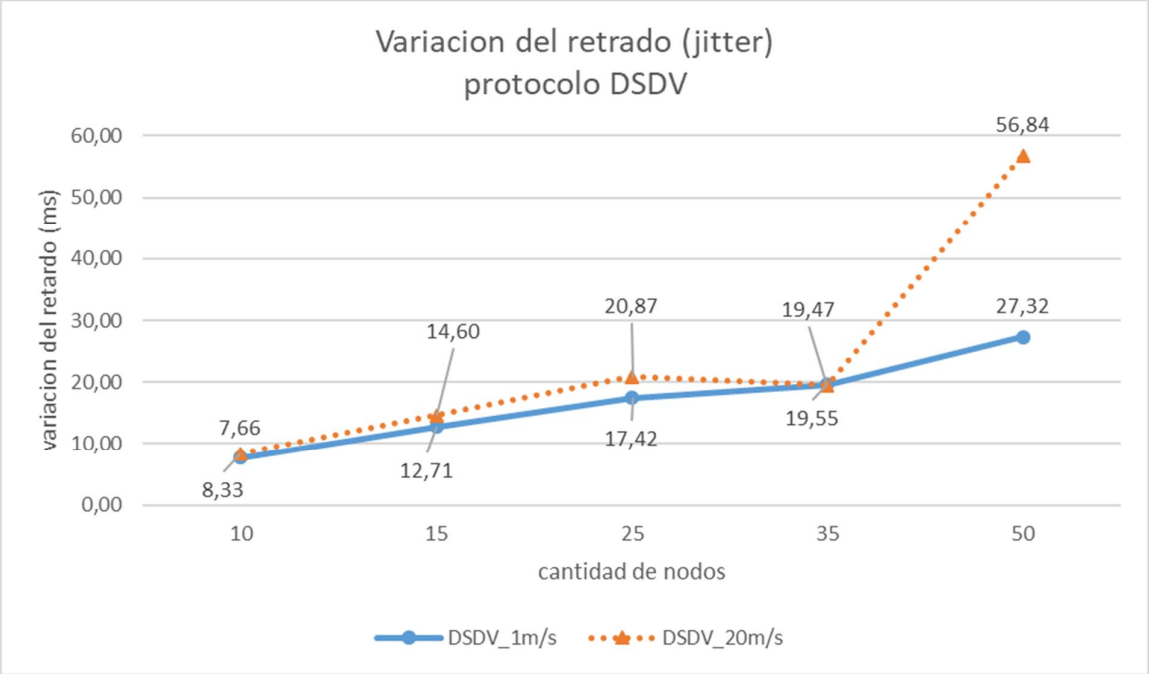

Figura 17 - Variación de la demora en la llegada de paquetes vs número de nodos para el protocolo DSDV

La Figura 17 muestra comportamiento del protocolo DSDV cuando se va variando la cantidad de nodos presentes en la red, considerando 2 velocidades distintas de todos los nodos de cada una de las redes, a saber: 1 m/s y otra movilidad a 20 m/s.

Se puede observar que el protocolo DSDV tiene un comportamiento similar en redes de 10 nodos hasta 35 nodos, incluso considerando la velocidad de nodo como variable desde 1 m/s hasta 20 m/s de los nodos que componen las redes.

Con redes más grandes que 35 nodos, el protocolo empeora su comportamiento en forma muy marcada, con respecto a la métrica de variación en la demora en la entrega de paquetes. Este empeoramiento es más pronunciado cuando se observa las redes cuyos nodos se están moviendo a mayor velocidad.

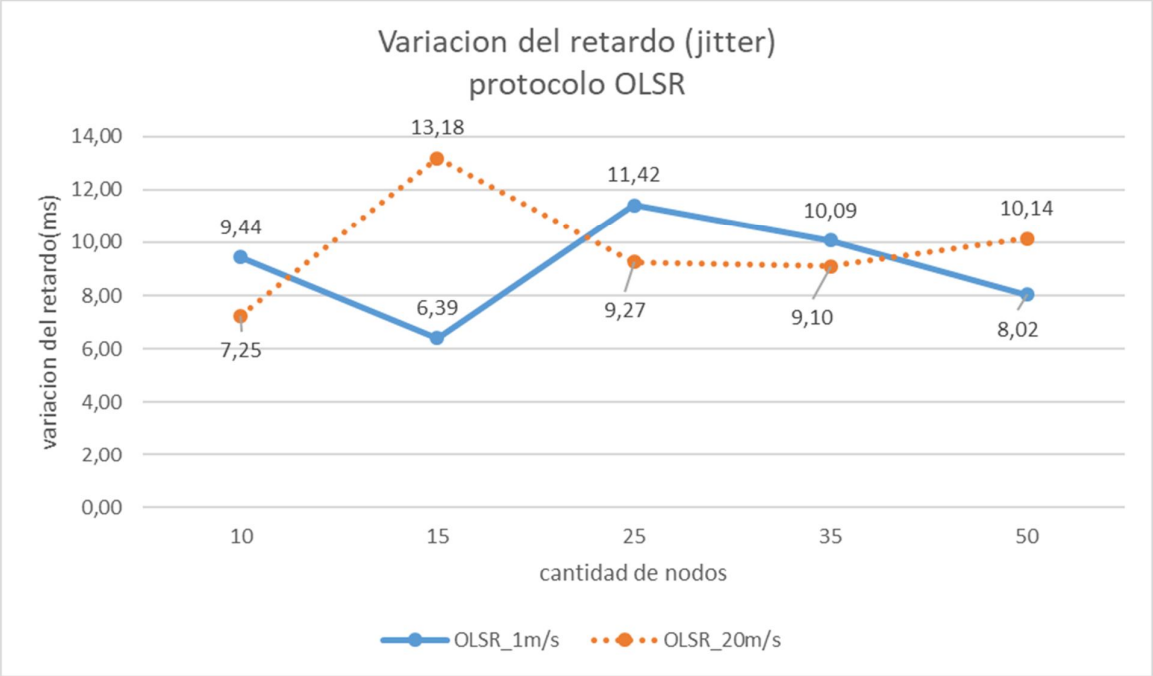

Figura 18 - Variación de la demora en la llegada de paquetes vs número de nodos para el protocolo OLSR

La Figura 18 muestra comportamiento del protocolo OLSR cuando se va variando la cantidad de nodos presentes en la red, considerando 2 velocidades distintas de todos los nodos de cada una de las redes, a saber: 1 m/s y otra movilidad a 20 m/s.

Se puede observar que el protocolo OLSR tiene un comportamiento similar en redes de 10 nodos hasta 50 nodos, incluso considerando la velocidad de nodo como variable desde 1 m/s hasta 20 m/s de los nodos que componen las redes.

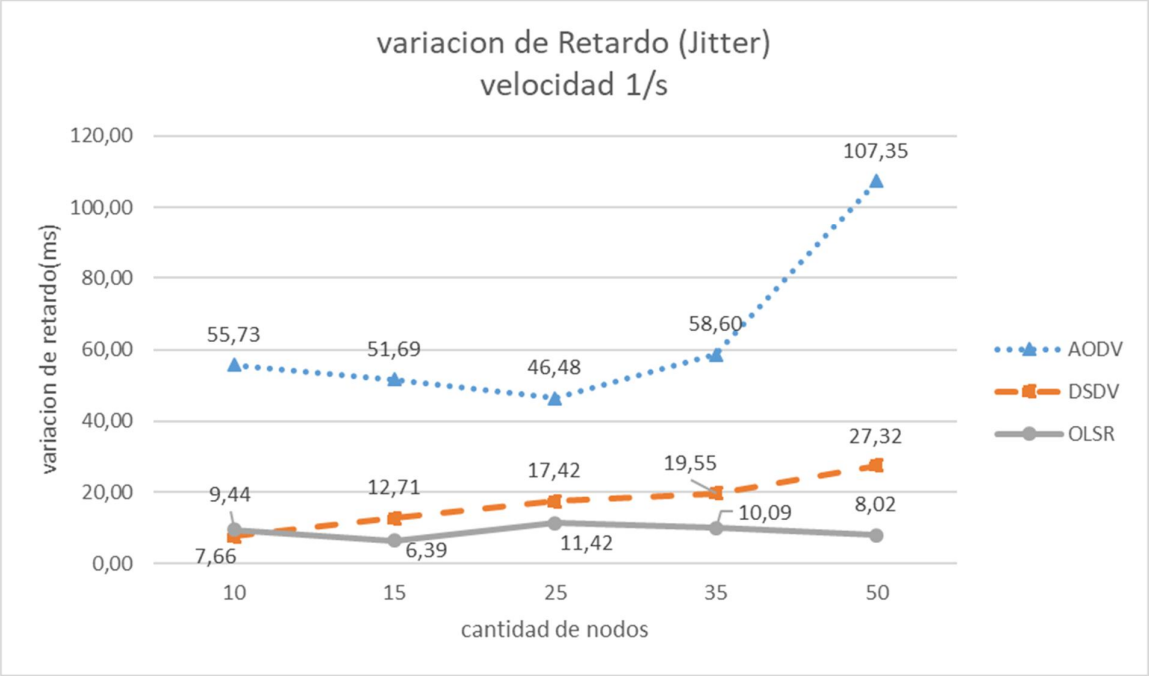

Figura 19 - La variación en la demora en la llegada de paquetes vs número de nodos para el protocolo los protocolos AODV, DSDV, OLSR

La Figura 19 muestra comportamiento de los protocolos AODV, DSDV, OLSR cuando se va variando la cantidad de nodos presentes en la red, considerando movilidad de todos los nodos de cada una de las redes de hasta 20 m/s.

Si tomamos en cuenta una red de 10 nodos hasta una red de 50 nodos, en todos los casos el comportamiento del protocolo AODV es peor que el de los protocolos restantes.

A partir de redes de 35 nodos o más el comportamiento del protocolo AODV comienza a deteriorase significativamente.

El protocolo AODV utiliza como uno de sus criterios de selección de ruta el trayecto que posea el retardo mínimo, independientemente de la carga de esas rutas. Esto ocasiona que las rutas más frecuentemente seleccionadas rápidamente se desbalanceen en cuanto a carga de tráfico, comparado con los demás protocolos, ocasionando problemas de variación en la llegada de los paquetes a su destino.

La diferencia observada entre OSLR y DSDV se debe a que el protocolo OLSR tiene constantemente actualizada su información de topología a través de sus nodos retransmisores multipunto y DSDV es un protocolo proactivo que tiene información sobre la topología almacenada, sin embargo, DSDV se basa en entradas de ruta obsoletas cuando no cuenta con rutas actualizadas recientes.

El hecho de que los nodos se estén moviendo, hace que mayor número de rutas dejen de funcionar, lo que desencadena antes el proceso de inundación para la búsqueda de nuevas rutas que utiliza el protocolo AODV.

El protocolo AODV debe iniciar el protocolo de descubrir rutas cuando necesita enviar paquetes de datos. Para ello inicia un procedimiento de inundación con paquetes de requerimiento de ruta, motivo por el cual cuando la red crece en número de nodos, el comportamiento del protocolo empeora.

El protocolo OLSR funciona de la manera más eficiente que los otros protocolos, en las redes con mayor número de nodos como se puede apreciar en el gráfico de la Figura 19 , debido a su estructura de nodos de retransmisión multipunto para mantener actualizada la información de topología.

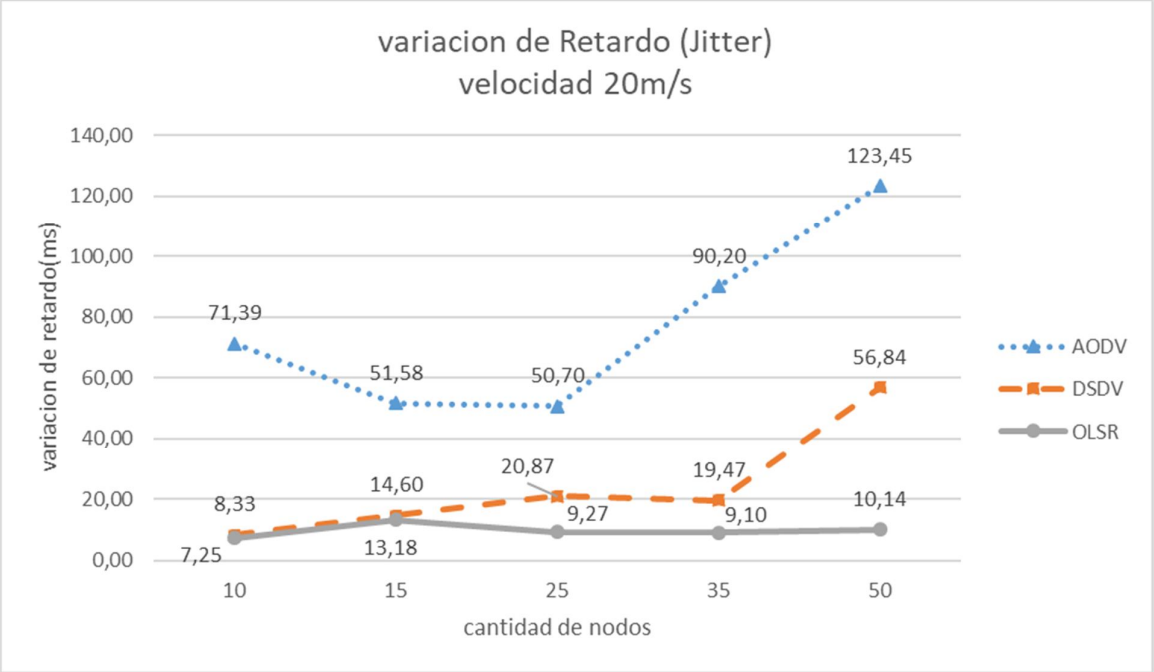

Figura 20 - La variación en la demora en la llegada de paquetes vs número de nodos para el protocolo los protocolos AODV, DSDV, OLSR

La Figura 20 muestra comportamiento de los protocolos AODV, DSDV, OLSR cuando se va variando la cantidad de nodos presentes en la red, considerando movilidad de todos los nodos de cada una de las redes de hasta 20 m/s.

Si tomamos en cuenta una red de 10 nodos hasta una red de 50 nodos, en todos los casos el comportamiento del protocolo AODV es peor que el de los protocolos restantes.

A partir de redes de 35 nodos o más el comportamiento del protocolo AODV comienza a deteriorase significativamente.

El protocolo AODV utiliza como uno de sus criterios de selección de ruta el trayecto que posea el retardo mínimo, independientemente de la carga de esas rutas. Esto ocasiona que las rutas más frecuentemente seleccionadas rápidamente se desbalanceen en cuanto a carga de tráfico, comparado con los demás protocolos, ocasionando problemas de variación en la llegada de los paquetes a su destino.

La diferencia observada entre OSLR y DSDV se debe a que el protocolo OLSR tiene constantemente actualizada su información de topología a través de sus nodos

retransmisores multipunto y DSDV es un protocolo proactivo que tiene información sobre la topología almacenada, sin embargo, DSDV que se basa en entradas de ruta obsoletas cuando no cuenta con rutas actualizadas recientes.

El hecho de que los nodos se estén moviendo, hace que mayor número de rutas dejen de funcionar, lo que desencadena antes el proceso de inundación para la búsqueda de nuevas rutas que utilizar el protocolo AODV, agravando el problema de desbalanceo de rutas mencionado que afecta la variación de llegada de paquetes a destino.

El protocolo AODV debe iniciar el protocolo de descubrir rutas cuando necesita enviar paquetes de datos. Para ello inicia un procedimiento de inundación con paquetes de requerimiento de ruta, motivo por el cual cuando la red crece en número de nodos, el comportamiento del protocolo empeora.

El protocolo OLSR funciona de la manera más eficiente que los otros protocolos, en las redes con mayor número de nodos como se puede apreciar en el gráfico de la Figura 20 , debido a su estructura de nodos de retransmisión multipunto para mantener actualizada la información de topología.

9. Clases y módulos del simulador utilizados

Clase ConstantSpeedPropagationDelayModel

```
ConstantSpeedPropagationDelayModel::GetTypeId (void)
{
 static TypeId tid = TypeId
 ("ns3::ConstantSpeedPropagationDelayModel")
   .SetParent<PropagationDelayModel> ()
   .SetGroupName ("Propagation")
   .AddConstructor<ConstantSpeedPropagationDelayModel> ()
   .AddAttribute ("Speed", "The propagation speed (m/s) in the propagation medium
          being considered. The default value is the propagation
          speed of light in the vacuum.",
        DoubleValue (299792458),
        MakeDoubleAccessor (&ConstantSpeedPropagationDelayModel::m_speed),
        MakeDoubleChecker<double> ())
 ;
 return tid;
}
```

Clase FriisPropagationLossModel

```
 FriisPropagationLossModel::GetTypeId (void)
 {
 static TypeId tid = TypeId ("ns3::FriisPropagationLossModel")
  .SetParent<PropagationLossModel> ()
  .SetGroupName ("Propagation")
  .AddConstructor<FriisPropagationLossModel> ()
  .AddAttribute ("Frequency",
     "The carrier frequency (in Hz) at which propagation occurs (default is 5.15 GHz).",
     DoubleValue (5.150e9),
     MakeDoubleAccessor (&FriisPropagationLossModel::SetFrequency,
         &FriisPropagationLossModel::GetFrequency),
     MakeDoubleChecker<double> ())
  .AddAttribute ("SystemLoss", "The system loss",
     DoubleValue (1.0),
     MakeDoubleAccessor (&FriisPropagationLossModel::m_systemLoss),
     MakeDoubleChecker<double> ())
  .AddAttribute ("MinLoss",
     "The minimum value (dB) of the total loss, used at short ranges. Note: ",
     DoubleValue (0.0),
     MakeDoubleAccessor (&FriisPropagationLossModel::SetMinLoss,
         &FriisPropagationLossModel::GetMinLoss),
     MakeDoubleChecker<double> ())
 ;
 return tid;
 }
```

Clase RandomRectanglePositionAllocator

```
 RandomRectanglePositionAllocator::GetTypeId (void)
 {
```

```
static TypeId tid = TypeId ("ns3::RandomRectanglePositionAllocator")
 .SetParent<PositionAllocator> ()
 .SetGroupName ("Mobility")
 .AddConstructor<RandomRectanglePositionAllocator> ()
 .AddAttribute ("X",
       "A random variable which represents the x coordinate of a position in a random
        rectangle.",
       StringValue ("ns3::UniformRandomVariable[Min=0.0|Max=1.0]"),
       MakePointerAccessor (&RandomRectanglePositionAllocator::m_x),
       MakePointerChecker<RandomVariableStream> ())
 .AddAttribute ("Y",
       "A random variable which represents the y coordinate of a position in a random
       rectangle.",
      StringValue ("ns3::UniformRandomVariable[Min=0.0|Max=1.0]"),
      MakePointerAccessor (&RandomRectanglePositionAllocator::m_y),
      MakePointerChecker<RandomVariableStream> ());
return tid;
}
```

Clase RandomWaypointMobilityModel

```
RandomWaypointMobilityModel::GetTypeId (void)
{
 static TypeId tid = TypeId ("ns3::RandomWaypointMobilityModel")
 .SetParent<MobilityModel> ()
 .SetGroupName ("Mobility")
 .AddConstructor<RandomWaypointMobilityModel> ()
 .AddAttribute ("Speed",
    "A random variable used to pick the speed of a random waypoint model.",
    StringValue ("ns3::UniformRandomVariable[Min=0.3|Max=0.7]"),
    MakePointerAccessor (&RandomWaypointMobilityModel::m_speed),
    MakePointerChecker<RandomVariableStream> ())
 .AddAttribute ("Pause",
    "A random variable used to pick the pause of a random waypoint model.",
    StringValue ("ns3::ConstantRandomVariable[Constant=2.0]"),
    MakePointerAccessor (&RandomWaypointMobilityModel::m_pause),
    MakePointerChecker<RandomVariableStream> ())
 .AddAttribute ("PositionAllocator",
    "The position model used to pick a destination point.",
    PointerValue (),
    MakePointerAccessor (&RandomWaypointMobilityModel::m_position),
    MakePointerChecker<PositionAllocator> ());

 return tid;
}
```

Conclusiones

El presente trabajo muestra los resultados que se obtuvieron al evaluar la relación de las métricas propuestas como una indicación de la orientación a la calidad de servicio y el comportamiento de los protocolos de enrutamiento proactivos y reactivos, tomando

como casos específicos de los mismos, a los protocolos OLSR, DSDV y AODV en un entorno de red inalámbrica ad hoc.

Además del número de nodos de las redes, si introdujo la variable movilidad de los nodos para analizar el efecto de la velocidad de los nodos sobre el comportamiento de los protocolos y como se ve reflejada esta variable en las métricas propuestas.

En estas simulaciones redes se tomaron como métricas la tase de pérdida de paquetes, la demora en la llegada de paquetes y la variación de la demora en la llegada de paquetes. Para lograr los objetivos planteados, se utilizó un simulador de eventos discretos, denominado simulador de redes 3, con el fin de poder realizar el análisis de las métricas establecidas para los protocolos propuestos.

De esta manera, se puede concluir los siguientes aspectos de los protocolos que se ven reflejados en las métricas propuestas:

El descarte de paquetes: La tasa de paquetes descartados señala al protocolo DSDV como el algoritmo de enrutamiento con rendimiento más modesto en cuanto esta métrica. Presenta un comportamiento inestable con respecto a esta métrica, en redes de 10 a 50 nodos.

Los protocolos OLSR y AODV se comportan en forma similar en redes ad hoc de un rango de 50 nodos, observándose que el protocolo AODV se adapta mejor a redes con 25 nodos o menos. El protocolo OLSR obtiene mejores valores de la métrica en redes mayores de 35 nodos.

La demora en la llegada de paquetes: El tiempo de demora para la llegada de paquetes a su destino clasifica al protocolo AODV como el algoritmo enrutamiento con rendimiento más modesto en cuanto esta métrica. Presenta un comportamiento inestable con respecto a esta métrica, en redes de hasta 35 nodos, después de dicho número de nodos, el protocolo empeora drásticamente su comportamiento. Si los nodos de la red

tienen una movilidad de aproximadamente 20 m/s o más, la cantidad de nodos critica se ubica en 25 nodos.

Los protocolos OLSR y DSDV se comportan en forma similar en redes ad hoc de un rango de 50 nodos, obteniendo valores ligeramente mejores para esta métrica el protocolo OLSR.

Si se considera una movilidad de los nodos de aproximadamente 20 m/s o mayor, el protocolo DSDV empeora su comportamiento en forma pronunciada en redes de 35 nodos o mayores. El protocolo OLSR obtiene mejores valores de la métrica en todos los casos, aunque esta mejoría sea leve en ocasiones.

La variación de la demora en la llegada de paquetes: La variación del tiempo de demora para la llegada de paquetes a su destino muestra al protocolo AODV como el algoritmo enrutamiento con rendimiento más modesto en cuanto esta métrica. Presenta un comportamiento inestable con respecto a esta métrica, en redes de hasta 35 nodos, después de dicho número de nodos, el protocolo empeora drásticamente su comportamiento. Si los nodos de la red tienen una movilidad de aproximadamente 20 m/s o más, la cantidad de nodos critica se ubica en 25 nodos.

Los protocolos OLSR y DSDV se comportan en forma similar en redes ad hoc de un rango de 50 nodos, obteniendo valores ligeramente mejores para esta métrica el protocolo OLSR.

Si se considera una movilidad de los nodos de aproximadamente 20 m/s o mayor, el protocolo DSDV empeora su comportamiento en forma pronunciada en redes de 35 nodos o mayores.

El protocolo OLSR se muestra como adecuado para redes de 15 nodos o mayores, donde obtiene mejores valores de la métrica incluso cuando se introduce nodos con una velocidad aproximada de 20 m/s.

Bibliografía

Becker, Philipp. 2007. QoS Routing Protocols for Mobile Ad-hoc . [Online] Agosto 6, 2007.

Blake, S., et al. 1998. RFC2475 - An Architecture for Differentiated Services. [Online] Diciembre 1998. [Cited: Junio 15, 2018.]

Braden, R., Clark, D. and Shenker, S. 1994. RFC1633 - Integrated Services in the Internet Architecture: an Overview. [Online] Julio 1994. [Cited: Noviembre 20, 2018.] https://tools.ietf.org/pdf/rfc1633.pdf.

Carneiro, Gustavo. 2010. NS-3: Network Simulator 3. [Online] Abril 20, 2010. https://www.nsnam.org/tutorials/NS-3-LABMEETING-1.pdf.

Chlamtac, Imrich, Conti, Marco and Liu, Jennifer J.-N. 2003. Mobile ad hoc networking: imperatives and challenges. [Online] 2003. [Cited: Marzo 10, 2018.] http://www.grc.upv.es/docencia/rinmalaga/biblio_BASICA/Mobile ad hoc networking imperatives and challenges.pdf.

Clausen, T. and Jacquet, P. 2003. RFC3626 - Optimized Link State Routing Protocol (OLSR). [Online] Octubre 2003. [Cited: Setiembre 12, 2018.] https://tools.ietf.org/pdf/rfc3626.pdf.

Corson, S. and Macker, J. 1999. RFC2501 - Mobile Ad hoc Networking (MANET) - Routing Protocol Performance Issues and Evaluation Considerations. [Online] Enero 1999. [Cited: Mayo 10, 2018.] https://tools.ietf.org/pdf/rfc2501.pdf.

Delgrossi, L. and Berger, L. . 1995. RFC1819 - Internet Stream Protocol Version 2 (ST2) Protocol Specification - Version ST2+. [Online] Agosto 1995. [Cited: Marzo 10, 2018.] https://tools.ietf.org/pdf/rfc1819.pdf.

Domínguez, Hugo Martín and Sáez Vacas, Fernando. 2006. Domótica: Un enfoque sociotécnico. [Online] Junio 2006. [Cited: Noviembre 20, 2018.] http://lsi.vc.ehu.es/pablogn/investig/dom%C3%B3tica/libro_domotica.pdf.

Fehér, Gábor, Németh, Krisztián and Maliosz, Markosz. 2000. Boomerang – A Simple Protocol for Resource Reservation in IP networks. [Online] Noviembre 2000. [Cited: Marzo 12, 2018.] http://www.cs.columbia.edu/~pingpan/siglite-papers/boomerang_RTAS.ps.

Gangwar, Sanjeev and Kumar, Krishan. 2011. Mobile Ad Hoc Networks: A detailed Survey of QoS Routing Protocols. [Online] Noviembre 2011. https://pdfs.semanticscholar.org/e225/ b5beee020d6baba8bb5a9163a21aa2bb1d0e.pdf.

Gast, Matthew S. 2005. *802.11 Wireless Networks: The Definitive Guide.* Sebastopol, USA : O' Reilly Media, 2005.

Gupta, Anuj K., Sadawarti, Harsh and Verma, Anil K. 2013. Performance Analysis of MANET Routing Protocols in Different Mobility Models. [Online] 05 2013. [Cited: 09 10, 2018.] http://www.mecs-press.org/ijitcs/ijitcs-v5-n6/IJITCS-V5-N6-10.pdf.

J-Sim, Sitio oficial. [Online] https://sites.google.com /site/ jsimofficial.

Kacer, Jaroslav. 2001. J-Sim – A Java-based Tool for Discrete Simulations. [Online] Septiembre 2001. https://www.kiv.zcu.cz/site/documents/verejne/vyzkum/ publikace/technicke-zpravy/ 2001/tr-2001-05.pdf.

Karl, Michael. 2005. A Comparison of the architecture of network simulators NS-2 and TOSSIM. [Online] Enero 2005. https://pdfs.semanticscholar.org/e00d/01fb88f167a80bbd2c9afe68d389575827be.pdf.

Kaur, Amandeep and Mahajan, Rajiv. 2013. A survey of QoS Routing Solutions for mobile ad hoc networks. [Online] Julio 2013. [Cited: Febrero 17, 2018.] https://pdfs.semanticscholar.org/fefd/4c1777dd8ac80525aa9c5ef7e6b9664a6645.pdf.

Kaur, Sandeep, et al. 2012. Mobility models in Adhoc networks. [Online] 11 2012. [Cited: Mayo 10, 2018.] http://www.rroij.com/open-access/mobility-models-in-adhoc-networks-58-64.pdf.

Keshav, S. ;. An Engineering Approach to Computer Networking - The REAL Network Simulator. [Online] http://www.cs.cornell.edu/skeshav/real/.

Kuosmanen, Petteri. 2002. Classification of Ad Hoc Routing Protocols. [Online] 2002. [Cited: Noviembre 15, 2018.] http://www.netlab.tkk.fi/opetus/s38030/k02/Papers/12-Petteri.pdf.

Lacage, Mathieu and Henderson, Thomas. 2006. Yet Another Network Simulator. [Online] Junio 07, 2006. [Cited: Agosto 10, 2018.] https://hal.inria.fr/file/index/docid/78318/filename/yans-rr.pdf.

Lin, Chunhung Richard and Liu, Jain-Shing . 1999. QoS Routing in Ad Hoc Wireless Networks. [Online] Agosto 1999. [Cited: Marzo 12, 2018.] http://erdos.csie.ncnu.edu.tw/~ccyang/WirelessNetwork/Papers/MANET/AdHocQoS-10.pdf.

Marrone , Luis Armando, et al. 2011. Administración de QoS en MANET. [Online] Mayo 2011. [Cited: 05 10, 2018.] http://sedici.unlp.edu.ar/handle/10915/19605.

Murazzo, Maria A., Rodríguez, Nelson R. and Martínez, Matías. 2009. Evaluacion y simulacion del rendimiento de los protocolos de ruteo para manet bajo restricciones de QoS. [Online] 2009. [Cited: Octubre 10, 2018.] http://sedici.unlp.edu.ar/bitstream/handle/10915/19663/Documento_completo.pdf?sequence=1.

Murthy, Siva Ram and Manoj, B. S. 2004. *Ad Hoc Wireless Networks. Architectures and Protocols.* New York : Pearson Education, Inc, 2004.

NetSim, Sitio oficial. Sitio oficial NetSim. [Online] https://www.tetcos.com.

Nichols, K., et al. 1998. RFC2474 - Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers. [Online] Diciembre 1998. [Cited: Mayo 20, 2018.] https://tools.ietf.org/pdf/rfc2474.pdf.

NS3 Official Website. NS3 Official Website. [Online] http://www.nsnam.org.

ns-3 project. 2017. ns-3 Tutorial. [Online] Marzo 14, 2017. [Cited: Mayo 15, 2018.] https://www.nsnam.org/docs/release/3.26/tutorial/html/index.html.

Omnet , Sitio oficial. Sitio oficial Omnet. [Online] http://www.omnetpp.org.

Opnet, sitio oficial. [Online] https://www.openet.com/.

Pan, Ping and Schulzrinne, Henning. 1999. YESSIR: A Simple Reservation Mechanism for the internet. [Online] Abril 1999. [Cited: Marzo 15, 2018.] http://www1.cs.columbia.edu/~pingpan/papers/yessir.pdf.

Patel, Rajan and Kamboj, Pariza. 2015. Investigation of Network Simulation Tools and Comparison Study: NS3 vs NS2. [Online] Diciembre 2015. https://pdfs.semanticscholar.org/60c6/95d1e7496076432b920b06d046b976d093c0.pdf.

Perkins, C., Belding-Royer, E. and Das, S. 2003. RFC3561 - Ad hoc On-Demand Distance Vector (AODV) Routing. [Online] Julio 2003. [Cited: Noviembre 19, 2018.] https://tools.ietf.org/pdf/rfc3561.pdf.

QualNet , Sitio oficial. [Online] https://web.scalable-networks.com/qualnet-network-simulator-software.

Raut, Surendra H. and Ambulgekar, Hemant P. 2013. Proactive and Reactive Routing Protocols in Multihop Mobile Ad hoc Network. [Online] Abril 2013. [Cited: Noviembre 19, 2018.] https://pdfs.semanticscholar.org/4704/ 972232ae2bf64df8e4fd3dcc93b414fbebdb.pdf.

Reid, Daniel and Katchabaw, Michael. 2004. Internet QoS: Past, Present, and Future. [Online] June 2004. [Cited: Junio 12, 2018.] http://www.csd.uwo.ca/faculty/katchab/pubs/tr_internetqos.pdf.

Sarkar, Subir Kumar, Basavaraju, T G and Puttamadappa, C. 2008. *Ad Hoc Mobile Wireless Networks - Principles, Protocols, and Applications.* Boca Raton : Auerbach Publications, 2008.

Singh Som, Davesh and Singh, Dhananjaya. 2012. Performance Analysis and Simulation of AODV, DSR and TORA Routing Protocols in MANETs. [Online] Agosto 2012. [Cited: Julio 23, 2018.] https://pdfs.semanticscholar.org/9aef/ 71813d76680893ea3db462d89dc2ad246dbd.pdf.

Soyinka, Wale. 2010. *Wireless Network Administration - A Beginner's Guide.* New York : McGraw-Hill, 2010.

Sun, Zhili. 2014. *Satellite Networking - Principles and Protocols.* Chichester, Inglaterra : John Wiley & Sons Ltd, 2014.

Toh, Chai Keong. 2001. Maximum Battery Life Routing to Support Ubiquitous Mobile Computing in Wireless Ad Hoc Networks. [Online] Junio 2001. [Cited: Setiembre 10, 2018.] https://pdfs.semanticscholar.org/169f/121daa66603e1f9cfef10e963b5d6483a538.pdf.

Varga, András and Hornig, Rudolf. 2008. An overview of the OMNeT++ simulation. [Online] 2008. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.231.4511&rep=rep1&type=pdf.

Weingartner, Elias, vom Lehn, Hendrik and Wehrle, Klaus. 2009. A performance comparison of recent network. [Online] Julio 2009. https://www.researchgate.net/publication/224574793_A_Performance_Comparison_of_Recent_Network_Simulators.

Wikipedia. 2018. IEEE 802.11. [Online] 10 19, 2018. [Cited: 10 30, 2018.] https://en.wikipedia.org/wiki/IEEE_802.11.

—. 2018. ns (simulador). [Online] Agosto 9, 2018. [Cited: Setiembre 10, 2018.] https://es.wikipedia.org/wiki/Ns_(simulador).

Yang, L., et al. 2003. Common Wireless Ad Hoc Network Usage Scenarios. [Online] Octubre 2003. [Cited: Noviembre 20, 2018.] https://tools.ietf.org/pdf/draft-irtf-yang-ans-scenarios-00.pdf.

Zhang, L., et al. 1997. RCF2205 - Resource ReSerVation Protocol (RSVP). [Online] September 1997. [Cited: Mayo 12, 2018.] https://tools.ietf.org/pdf/rfc2205.pdf.