ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

FACULTAD DE INFORMÁTICA Y ELECTRÓNICA

ESCUELA DE INGENIERÍA EN ELECTRÓNICA, TELECOMUNICACIONES Y REDES

"SIMULACIÓN, MODELAMIENTO Y EVALUACIÓN DE LOS PROTOCOLOS DE ROUTING EN REDES MANET'S CON APLICACIONES DE VIDEO MEDIANTE PLATAFORMAS OPENSOURCE DE EVENTOS DISCRETOS"

Trabajo de titulación presentado para optar al grado académico de:

INGENIERA EN ELECTRÓNICA, TELECOMUNICACIONES Y REDES

AUTORA: KATERYNE KIMBERLYN NIAMA BORJA

TUTOR: ING. ALBERTO ARELLANO AUCANCELA

Riobamba-Ecuador

2017

ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

FACULTAD DE INFORMÁTICA Y ELECTRÓNICA

ESCUELA DE INGENIERÍA ELECTRÓNICA

EN TELECOMUNICACIONES Y REDES

El Tribunal certifica que: El trabajo de titulación: "SIMULACIÓN, MODELAMIENTO Y EVALUACIÓN DE LOS PROTOCOLOS DE ROUTING EN REDES MANET'S CON APLICACIONES DE VIDEO MEDIANTE PLATAFORMAS OPENSOURCE DE EVENTOS DISCRETOS", de responsabilidad de la señorita Kateryne Kimberlyn Niama Borja, ha sido minuciosamente revisado por los miembros del Tribunal del Trabajo de Titulación, quedando autorizada su presentación.

Ing. Washington Luna **DECANO FACULTAD DE INFORMÁTICA Y ELECTRÓNICA**	_____	_____
Ing. Franklin Moreno **DIRECTOR DE ESCUELA DE INGENIERÍA ELECTRÓNICA EN TELECOMUNICACIONES Y REDES**	_____	_____
Ing. Alberto Arellano **DIRECTOR DEL TRABAJO DE TITULACIÓN**	_____	_____
Ing. Edwin Altamirano **MIEMBRO DEL TRIBUNAL**	_____	_____

Yo, Kateyne Kimberlyn Niama Borja declaro ser la autora del presente trabajo de titulación: "SIMULACIÓN, MODELAMIENTO Y EVALUACIÓN DE LOS PROTOCOLOS DE ROUTING EN REDES MANETS CON APLICACIONES DE VIDEO MEDIANTE PLATAFORMAS OPENSOURCE DE EVENTOS DISCRETOS", que fue elaborado en su totalidad por mi persona, bajo la dirección del Ingeniero Alberto Arellano, haciéndome totalmente responsable de las ideas, doctrinas y resultados expuestos en este Trabajo de Titulación y el patrimonio de la misma pertenece a la ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO.

Kateryne Kimberlyn Niama Borja

DEDICATORIA

Esta tesis la dedico a mis padres Alex Niama, Gladys Borja, a mis hermanos y sobrinos por el apoyo incondicional y la confianza que me han dado en el transcurso de mi carrera y haberla culminado.

Su apoyo asido muy importante en mi vida que gracias a ustedes he llegado a dar este paso muy importante en mi vida les amo mucho y esto es gracias a ustedes.

Kateryne K. Niama Borja.

AGRADECIMIENTO

Agradezco de corazón a mis padres Alex Niama y Gladys Borja, a mis hermanos Paulina, Alex y Nathy por sus consejos que me han ayudado a seguir adelante para terminar mi carrera, su apoyo que ha sido incondicional en mi vida para llegar a ser una profesional.

A mis amigas y amigos especialmente a Gisela C., Daniela L., Jhoanna Z., Miguel M., María José V., Ángel B. y Fernanda A. por el apoyado que me ha brindado y a Julio M. por darme su apoyo cada día y acompañarme en cada paso que he dado para terminar mi carrera, por sus consejos y su confianza.

Les agradezco a todos porque les quiero y estimo mucho.

Kateryne K. Niama Borja.

TABLA DE CONTENIDO

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

ÍNDICE DE GRÁFICOS

ÍNDICE DE ANEXOS

RESUMEN

Este trabajo de titulación tuvo como objetivo analizar, modelar y simular los protocolos de enrutamiento en redes móviles ad–hoc (MANET), para aplicaciones de video mediante las plataformas OpenSource de eventos discretos. Las redes MANET, están compuestas por nodos móviles inalámbricos, que no depende de una estructura física, por lo que se tuvo la ventaja de trabajar de una manera rápida, además, permitió tener múltiples saltos desde el nodo origen hasta el nodo destino y su zona de cobertura es limitada. Para cumplir el objetivo planteado, se realizó en primera instancia el estudio de los distintos protocolos de enrutamiento proactivos y reactivos, determinando así que el protocolo proactivo Vector Distancia de Secuencia – Destino (DSDV) y el protocolo reactivo Vector Distancia Bajo Demanda Ad–Hoc (AODV), presentan mejores características y son los más idóneos para implementar en la plataforma de simulación de redes NS-3, la misma que es elegida por sus particularidades y ventajas frente a la plataforma OMNET++. Se plantearon dos escenarios en el simulador NS-3, para el primer escenario se configuró el protocolo Reactivo AODV y el segundo escenario el protocolo Proactivo DSDV, usando el modelamiento en topología de red tipo malla, en cada uno se evaluaron tres parámetros: pérdida de paquetes, jitter y retardo. Posteriormente se realizaron pruebas y análisis para determinar el mejor protocolo para aplicaciones de video, teniendo un porcentaje de perdida de paquetes del 2%, jitter del 5,41% y un retardo del 5,88%, con respecto a una distancia de 20 m, tiempo de 10 segundos y un número de 8 nodos, por consiguiente se concluyó que el protocolo reactivo AODV en una red MANET puede determinar la mejor ruta entre el nodo origen y destino de una manera rápida y fiable obteniendo una red más eficiente.

Palabras claves: <TECNOLOGÍA Y CIENCIAS DE LA INGENIERÍA>**,** <REDES DE COMPUTADORES>**,** < REDES MÓVILES AD–HOC (MANET)>, < VECTOR DISTANCIA BAJO DEMANDA AD–HOC (AODV)>, < VECTOR DISTANCIA DE SECUENCIA–DESTINO (DSDV)>, <NS-3 (SOFTWARE)>, <OMNET++ (SOFTWARE)>, <ANÁLISIS DE TRÁFICO>.

ABSTRACT

This work for titling has the proposal of analyzing, modeling and simulating the routing protocols in ad-hoc mobile networks (MANET), for video applications by using OpenSource platforms of discrete events. MANET networks are composed of wireless nodes that do not depend on a physical structure, so have the advantage of a fast working allowing multiple jumps from the source node to the destination node and its coverage area is limited. To find the objective, the study of different proactive and reactive routing protocols was carried out, determining that the proactive protocol Vector Sequence Distance – Destination (DSDV), and the Vector-Distance Reactive Vector Protocol (AODV), were used to determine the number of proactive and reactive routing protocols, they present better characteristics and are the most suitable to implement in the platform of simulation of networks NS-3 chosen for its particularities and advantages in front of the platform OMNET ++. Two scenarios were proposed in the NS-3 simulator, for the first scenario, the AODV Reactive protocol was set up, and the second scenario was the Proactive DSDV protocol, using modeling in network mesh topology, in each one, three parameters were evaluated: packet loss, jitter and delay. Later, tests and analysis were performed to determine the best protocol for video applications, with a percentage of packet losses of 2%, jitter of 5.41% and a delay of 5.88%, respecting to a distance of 20 m, time of 10 seconds and a number of 8 nodes, therefore it was concluded that the AODV reactive protocol in a MANET network can determine the best route between the source and destination node in a fast and reliable way, obtaining a more efficient network.

Keywords: <TECHNOLOGY AND ENGINEERING SCIENCES>, <COMPUTER NETWORKS>, <AD-HOC MOBILE NETWORKS>, <VECTOR DISTANCE ON DEMAND AD-HOC (AODV) <SEQUENCE-DESTINATION DISTANCE VECTOR (DSDV)>, <NS-3 (SOFTWARE)>, <OMNET ++ (SOFTWARE)>, <TRAFFIC ANALYSIS>

INTRODUCCIÓN

Las redes inalámbricas han tomado impulso debido a las actividades del entorno diario, como los negocios y el comportamiento social, por lo que se ha ido creando diversos usos, que permiten a los usuarios tener comunicación en diferentes escenarios, sin embargo debido a la limitada cobertura de la red base la movilidad de los dispositivos se ve afectada.

Las redes ad-hoc resuelven esta situación debido a su flexibilidad, ya que permiten que sus nodos se desempeñen de forma autónoma, siendo así que pueden actuar como host o como router, debido a que su topología puede cambiar de manera arbitraria y repentina.

Dentro de la clasificación de las redes ad-hoc, se encuentran las redes MANET (Mobile Ad-hoc Network), que son redes inalámbricas que permiten una comunicación de modo autónomo, ya que no necesitan de una infraestructura preestablecida, como es el caso de las redes wifi usuales que necesitan de puntos de acceso. Además las redes MANET son espontáneas, auto-organizadas porque tienen la capacidad de formarse en cualquier momento organizadamente y su comunicación es multi-salto, debido a esto es necesario el uso de un protocolo de enrutamiento, que permita lograr una ruta confiable desde su nodo origen hacia su nodo destino. Sin embargo, los protocolos de enrutamiento tradicionales, no se adaptan a este tipo de red, por lo que necesitan sus protocolos propios, siendo estos los reactivos y proactivos.

En este proyecto de investigación se realizó un análisis de los protocolos más utilizados, como lo son AODV (Ad-hoc On-demand Distance Vector) y el DSDV (Destination Sequence Distance Vector), utilizando la plataforma de simulación NS-3, de modo que se pueda determinar cuál es el protocolo más eficiente y confiable dentro de una red MANET.

ANTECEDENTES

Las primeras redes inalámbricas denominas ad-hoc, fueron desarrolladas en EE.UU, por la Agencia de Investigación de Proyectos Avanzados de Defensa, en los años 70, mismas que operaban en radiofrecuencia UHF (Ultra High Frequency), las pioneras fueron ALOHAnet y PRNET (Packet Radio Networks). (Agurto Armero, Hernandez Rivas, & Lopez Guardado, 2016).

En 1983, esta agencia puso en marcha el proyecto SURAN (Survivable Radio Network), cuya finalidad era desarrollar una red ad-hoc móvil de bajo costo, y que pudiese implementar protocolos packet radio más complejo que PRNET. En los años 90, se iniciaron distintos proyectos en el ámbito militar y de defensa, tales como GloMo (Global Mobile) y NTDR (Near Term Digital Radio). (Batiste, 2011).

En la año 1997, el ejército de EE.UU empezó la implementación de una red inalámbrica multisalto, sin embargo, el desarrollo de las redes ad-hoc tenían como principal propósito, conseguir que los nodos tuvieran la capacidad de moverse, aparecer y desaparecer, este tipo de red se denomina MANET (Mobile Ad-Hoc Network).

Las redes inalámbricas móviles, están evolucionando debido a la importancia en la sociedad, por lo que los estudios se centran en mejorar el funcionamiento de las redes MANET, para lo cual el Grupo de Servicios Telemáticos de la Universidad Politécnica de Cataluña, tratan de enfocar sus estudios en ofrecer calidad de servicio para las redes Ad-Hoc.

En la actualidad las aplicaciones de las redes MANET, se desarrollan en dispositivos móviles personales, este incremento está generando una mayor demanda de las redes ad-hoc como en: áreas militares, redes domésticas, redes de área personal, emergencias, etc, un ejemplo, son los nuevos sistemas de transporte inteligente que utilizan redes vehiculares ad-hoc, donde se busca reducir los índices de accidentes, esta es una muestra del potencial de estas nuevas redes.

Las MANET, son redes formadas a partir de un terminal móvil autónomo, a través de un enlace inalámbrico, sin utilizar una infraestructura fija, su comunicación actúa como una ruta de múltiples saltos, por lo cual es necesario definir protocolos con características específicas que se adapten a estas redes, de modo que se pueda conseguir una implementación de red ad-hoc eficiente.

Se ha investigado en el repositorio de las tesis digitales de la ESPOCH y aún no se ha realizado ningún estudio sobre el tema propuesto.

FORMULACIÓN DEL PROBLEMA

¿Cómo influye la simulación y evaluación de protocolos de routing en redes MANET's con aplicaciones de video?

SISTEMATIZACIÓN DEL PROBLEMA

- ¿Cuáles son los algoritmos de routing que se usarán para las aplicaciones de video?
- ¿Qué plataformas de OpenSource van hacer utilizadas para la simulación?
- ¿Qué parámetros van hacer evaluados en la simulación de redes MANET's?
- ¿Cuál es el protocolo de routing en redes MANET's más adecuado para la transmisión de video?

JUSTIFICACIÓN TEÓRICA

Las redes MANET's, están formadas por un conjunto de estaciones móviles inalámbricas sin ninguna estructura preexistente. Los protocolos que se analizaran y evaluaran son: el protocolo reactivo AODV y el protocolo proactivo DSDV.

Protocolo Reactivo.- Este protocolo, actualiza su tabla de enrutamiento solo si es necesario realizar una transmisión, de esta manera busca la ruta más adecuada de modo que el paquete pueda llegar a su destino.

Protocolo Proactivo.- Este protocolo actualiza constantemente su tabla de enrutamiento, y por ende recopila todos los caminos posibles hacia los múltiples nodos de la red.

Para el desarrollo de esta investigación, se realizarán pruebas de simulación, mediante el manejo de las plataformas de OpenSource de eventos discretos como: OMNET++ y NS-3, software que trabajan bajo Linux y lenguaje C++.

A medida que se realice la programación de los diferentes protocolos, utilizando los algoritmos de enrutamiento, se ira comparando los resultados obtenidos, para así analizar y determinar el mejor protocolo, el cual será evaluado en base a los parámetros de pérdida de paquetes, retardo y

jitter, y así concluir que protocolo de red MANET puede ser más eficiente en aplicaciones de video.

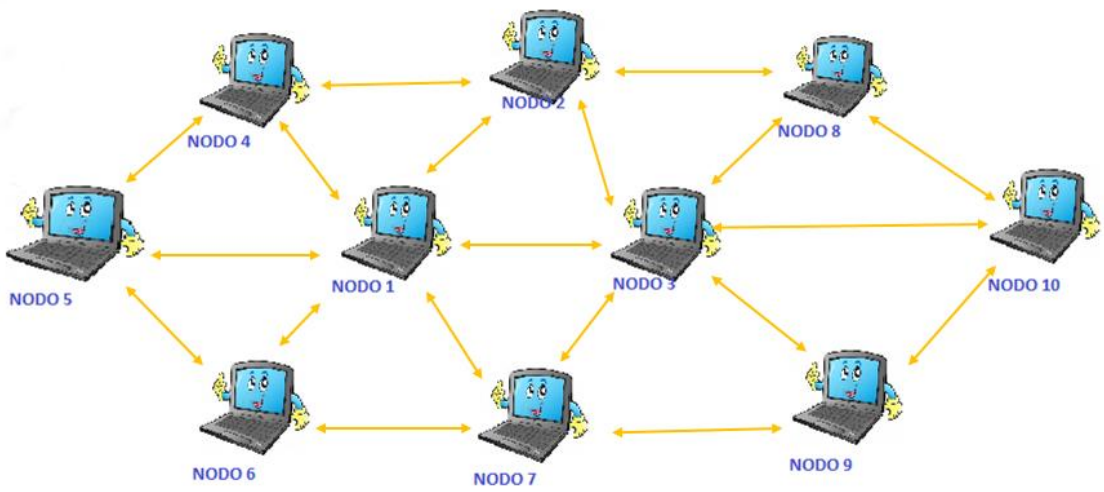
JUSTIFICACIÓN APLICATIVA

Para el estudio se realizara la simulación de redes, ya que se requiere evaluar el desempeño de diferentes plataformas de OpenSource.

Poder simular en un sistema real es costoso debido a los equipos que se necesitan para conformar una red, y por lo que son sistemas que se pueden utilizar en entornos militares, civiles, etc. Es por eso que se utilizará simuladores adecuados que permitan diseñar y experimentar, en condiciones reales, garantizando homogeneidad en las respuestas del sistema y por tanto mayor confiabilidad en las conclusiones que arroje este estudio.

El análisis se realizará en un simulador de eventos discretos, que permite identificar los sucesos que tienen un lugar y un instante en el tiempo. Este tipo de simulador es utilizado para el análisis de sistemas secuenciales, los cuales son frecuentes en las comunicaciones.

En la siguiente figura se muestra la topología propuesta para la configuración y programación de los protocolos de las redes MANET´s.



Topología de red con 10 nodos

Realizado por Kateryne K. Niama. 2017

OBJETIVOS

OBJETIVO GENERAL

- Simular, modelar y evaluar los protocolos de routing en redes MANET's con aplicaciones de video mediante plataformas OpenSource de eventos discretos.

OBJETIVOS ESPECÍFICOS

- Realizar el estudio del arte de la simulación y la evaluación de protocolos routing en redes MANET's.
- Seleccionar las plataformas de simulación más apropiadas para configurar y programar los protocolos elegidos.
- Realizar las simulaciones de los protocolos seleccionados con las plataformas indicadas.
- Evaluar los resultados para determinar el algoritmo más adecuado.

CAPITULO I

1 MARCO TEÓRICO

1.1 Redes inalámbricas

A medida que avanza la tecnología, una de las innovaciones más prometedoras de las ciencias informáticas son las redes inalámbricas por el gran aporte que representa en las empresas, y en la sociedad; por lo que son ampliamente investigadas.

Las redes inalámbricas, permiten que varios host que no se encuentran conectados entre sí, puedan compartir información de una manera móvil, es decir, que facilitan la comunicación entre computadoras sin necesidad de encontrarse en un mismo sitio, pero si dentro de un misma red. Figura 1.1.

Figura 1. 1 Redes Inalámbricas

Fuente http://blogluzdey.blogspot.com/2015/07/4.html

1.1.1 Tipos de redes inalámbricas

Modo infraestructura.- Estas redes tiene un infraestructura preestablecida, necesitan un punto de acceso para poder comunicarse entre hots. Se clasifican en:

- WLAN.- Redes Inalámbricas de área local
- WPAN.- Redes inalámbricas Personales
- WMAN.- Redes de área metropolitana Inalámbricas

- WWAN.- Redes inalámbricas de aérea ancha.

Ventajas:

- Conexión de varios dispositivos.
- Escalabilidad de redes cableadas.
- Mayor alcance

Modo Ad-hoc.- No requieren de una infraestructura, los nodos son autónomos y no depende de una administración centralizada. Se clasifican en:

- MANET's.- Redes móviles Ad-hoc.
- WSN.- Redes de sensores
- VANET.- Redes vehiculares Ad-hoc
- WMN.- Redes inalámbricas Mesh.

Ventajas:

- No es necesario conexión a internet.
- La información llega a su destino sin tener que pasar por otro equipo.
- Los ordenadores pueden compartir información entre sí, mientras se encuentren en la misma red.
- No necesitan de un control central.

1.2 Redes inalámbricas Ad-Hoc

Es una red que no necesita de una dirección central, no requieren de infraestructura fija, por lo que los nodos pueden ser clientes o servidores para tener una comunicación. Al ser inalámbrica, dependerá del rango de alcance ya que puede ser limitado e impedir la trasmisión de datos. Como se muestra en la figura 2.1.

Figura 2.1 Redes Inalámbricas Ad-hoc

Realizado por Kateryne K. Niama. 2017

1.3 Red MANET

Una red MANET, está implementada por un conjunto de nodos móviles inalámbricos, cada nodo se comunica con protocolos de encaminamiento mediante el reenvío de datos hacia sus nodos vecinos, que se conectan por medio de enlaces sin utilizar una estructura física siempre que esté en su rango de cobertura, es decir, que son sistemas distribuidos de manera compleja que comprenden de nodos que pueden auto-organizarse de manera libre, son flexibles y sencillamente desplegables.

En la Figura 3.1 se puede observar una red de nodos móviles, donde el área de los círculos representa la cobertura de las interfaces inalámbricas de cada nodo.

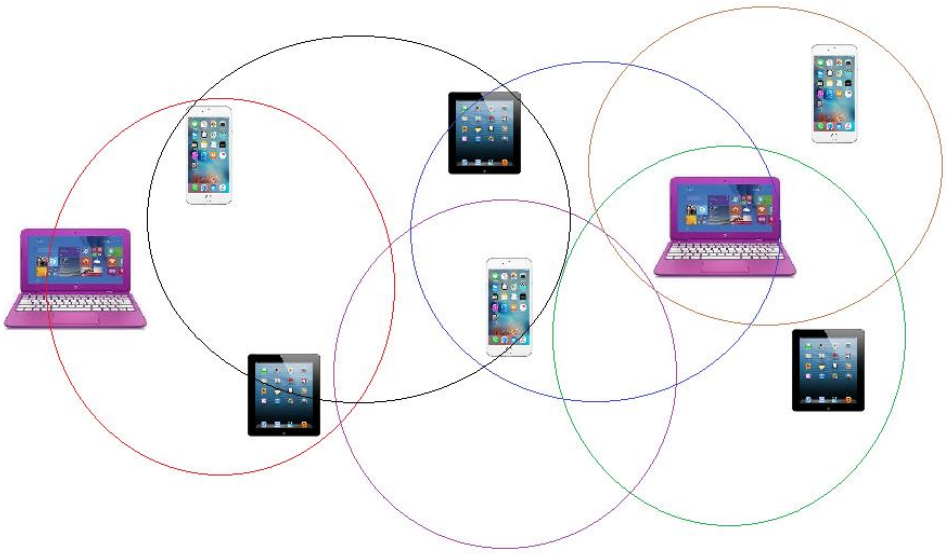


Figura 3.1 Ejemplo de una red MANET

Realizado por Kateryne K. Niama. 2017

1.3.1 Características de la Red MANET

1. **Topología dinámica, nodos móviles:** Sus nodos pueden moverse libremente, es decir la red puede cambiar aleatoriamente y de manera impredecible, por lo tanto se puede formar nuevos enlaces con otros hots que están en su mismo rango de cobertura.

2. **Multi-Salto:** La información se da por múltiples saltos hasta llegar a su destino.

3. **Escalabilidad:** La red puede crecer y tener varios nodos.

4. **Limitaciones energéticas:** Algunos nodos funcionan con baterías con vida limitada, es por eso la optimización de protocolos para que la red sea eficiente.

5. **Ausencia de infraestructura.-** No existe una administración central, los equipos pueden ser hots o routers de una manera aleatoria.

6. **Seguridad:** Este tipo de redes son más vulnerables a posibles ataques que las redes alámbricas, debido a la capacidad de procesamiento de los nodos y la seguridad física.

1.3.2 Aplicaciones de las redes MANET

Debido al avance en los últimos años que ha tenido las redes MANET, son empleadas en las siguientes áreas:

1. **Áreas militares.-** Estas son las primeras en surgir en las redes MANET, donde permite el enlace entre diferentes dispositivos, una ventaja es que tiene la capacidad de comunicarse con la red, siempre y cuando esté en su zona de cobertura, se utilizan en operaciones militares o de rescate.

2. **Entornos civiles.-** Se puede construir una red MANET, donde permite compartir información entre usuarios.

3. **Redes domésticas:** Los equipos se comunican para intercambiar información.

4. **Redes de área personal.-** Redes conformadas por elementos de uso personal.

5. **Emergencias.-** Para una urgencia la red MANET puede expandirse ágilmente sin necesidad de instalar un equipo central que cubra su rango de enlace, esto permite que los nodos actúen solos brindando una conectividad con sus nodos vecinos, solucionando una situación de manera rápida y eficaz, como: en desastres naturales, terremotos, etc.

6. **Vanet.-** Son redes móviles vehiculares que permiten disminuir el número de accidentes de tráfico, también admite ver información que se esté compartiendo entre diferentes vehículos.

1.3.3 Algoritmos de Enrutamiento Ad-hoc

Los algoritmos de enrutamiento, son los encargados de determinar la ruta de un coste mínimo por el cual va hacer enviado el paquete hasta llegar a su destino. Los más utilizados en las redes MANET son los siguientes:

1.3.3.1 Algoritmo de Estado de Enlace

Estado de enlace, es la conexión con el router que identifica cuáles son sus nodos vecinos y a que distancia se encuentran, asignando un costo en cada conexión dado que cada nodo crea un mapa de la red donde su información es solo enviada hacia los enlaces conectados localmente,

manteniendo sus tablas de enrutamiento sin bucles, este tipo de algoritmo envía actualizaciones mediante multicast.

1.3.3.2 Algoritmo Vector-Distancia

Vector Distancia, se basa en el número de saltos, es decir que se determina mediante el recorrido que hace el paquete entre los nodos hasta llegar a su destino, la ruta selecciona es la que tiene menos números de saltos ya que es la más óptima para enviar la información, también comunica a sus nodos vecinos que usen el mismo protocolo del cambio de red.

1.3.3.3 Source Routing

Permite a su remitente enviar un paquete con una ruta establecida que debe seguir a través de la red para llegar a su destino.

1.3.4 Protocolos de enrutamiento en las redes Ad-Hoc

Los protocolos de enrutamiento en Ad-hoc, se determinan como unicast, ya que se centran en que el paquete llegue desde su origen hacia su destino, una clasificación de los mismos está dividido en 3 grupos, proactivos, reactivos e híbridos, como se muestra en la figura 4.1.

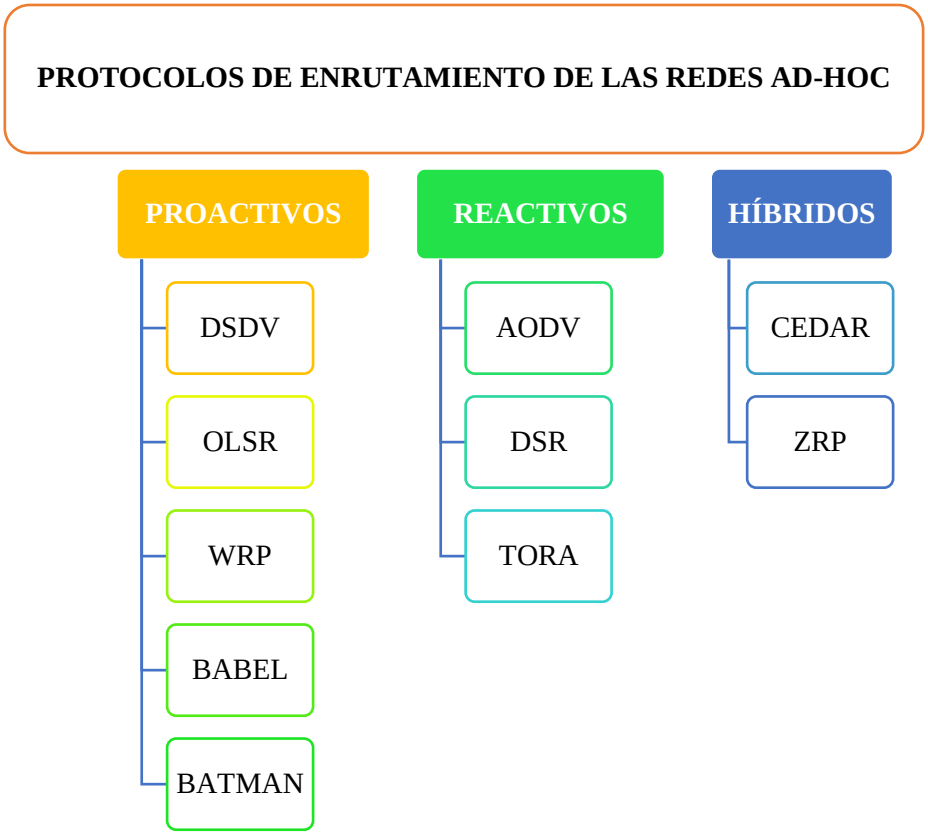


Figura 4.1 Tipos de protocolos de enrutamiento ad-hoc.

Realizado por Kateryne K. Niama. 2017

Tabla 1.1 Significado de los Protocolos de Enrutamiento

PROTOCOLOS PROACTIVOS	
DSDV	Destination-Sequenced Distance-Vector
OLSR	Optimised Link State Routing
WRP	Wireless Routing Protocol
BABEL	A loop-free distance-vector routing protocol
BATMAN	Better Approach to Mobile Ad hoc Networking
PROTOCOLOS REACTIVOS	
AODV	Ad-hoc On-Demand Distance Vector
DSR	Dynamic Source Routing
TORA	Temporally Ordered Routing
PROTOCOLOS HÍBRIDOS	
CEDAR	Core-Extraction Distributed Ad Hoc Routing
ZRP	Zone Based Routing Protocol

Realizado por Kateryne K. Niama. 2017

1.3.4.1 Protocolo de enrutamiento Reactivo

El protocolo de enrutamiento reactivo, tiene como propósito disminuir la carga de los protocolos proactivos, su tabla de enrutamiento se actualiza cuando es necesario, es decir, cuando el nodo fuente necesita transmitir un paquete busca la ruta más conveniente para ser comunicado con su nodo emisor, confirmando su entrega de paquete enviado, teniendo una baja velocidad de respuesta por los cambios en la topología de la red, recordando que son nodos móviles inalámbricos, es por eso que no existe una sobrecarga, los protocolos que pertenecen al enrutamiento reactivos siendo los más utilizados los siguientes:

- **Ad-Hoc On-Demand Distance Vector (AODV)**

AODV, es un protocolo reactivo, que trabaja bajo el algoritmo Vector-Distancia, la ventaja de este protocolo es que no genera tráfico en la red para su comunicación ya que su tabla de enrutamiento se actualiza solo cuando es necesario enviar un paquete a sus nodos. Figura 5.1.

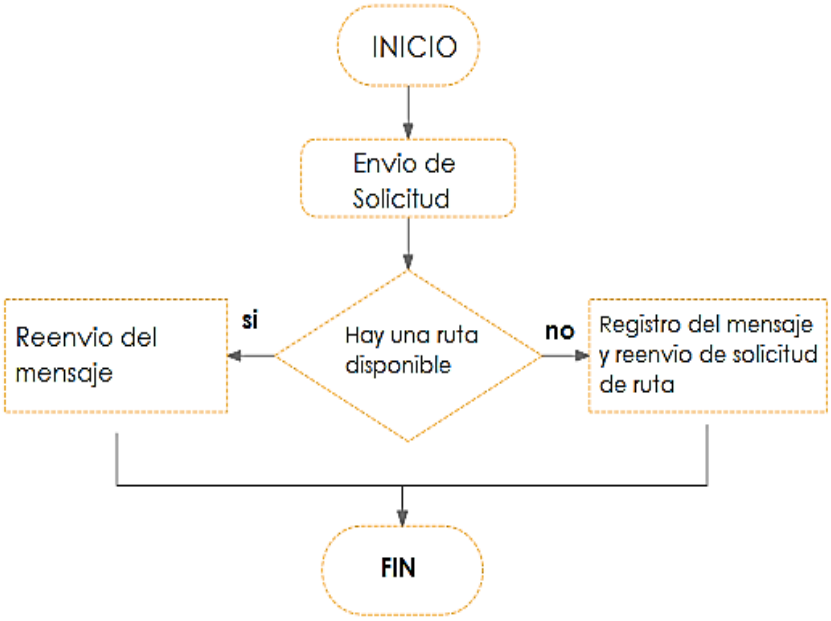


Figura 5.1 Diagrama de proceso de envió de solicitud en AODV

Realizado por Kateryne K. Niama. 2017

Para el descubrimiento de la ruta, el nodo emisor envía un paquete a su nodo receptor en modo Broadcast, el emisor envía mensajes RREQ (Route Request) a todos sus nodos vecinos conectados y cualquier nodo que conozca la ruta hacia el nodo fuente puede contestar con un mensaje RREP (Route Replies) al emisor, indicando cual es la ruta más corta y rápida que necesita para ser enviada la información.

Mediante la comunicación entre el nodo emisor y receptor, puede ocurrir que algún nodo vecino haya cambiado su posición ya que son nodos móviles, esto causa que un enlace se haya roto y que la ruta no se pueda utilizar, en ese caso el nodo del enlace roto debe informar al resto de nodos con un mensaje RERR (Route Errors) que deben reencaminar el paquete por otra ruta y el nodo emisor decide si se inicia el proceso de envió de información o se cancela la petición, recordando que los mensajes son enviados mediante UDP. Figura 6.1.

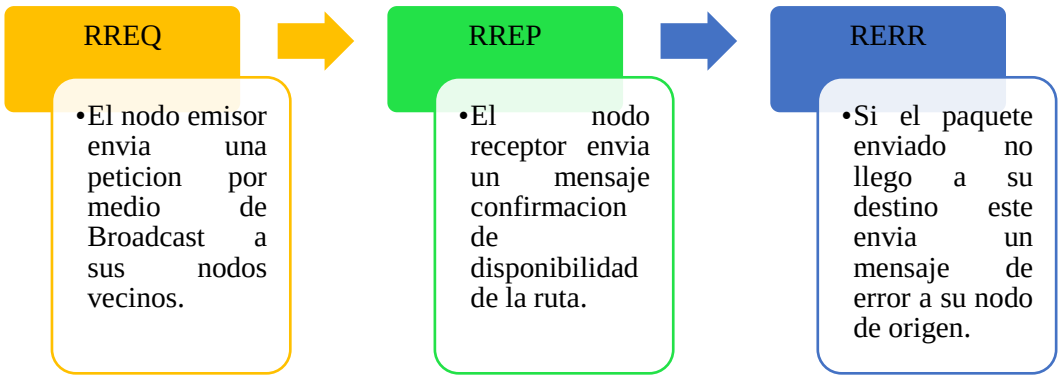


Figura 6.1 Proceso de funcionamiento del protocolo de enrutamiento de AODV

Realizado por Kateryne K. Niama. 2017

- **Dynamic Source Routing (DSR)**

DSR, es un protocolo para direcciones IPv4, al ser un protocolo reactivo funciona bajo demanda. No utiliza mensajes de actualización al enviar paquetes, por lo que tiene poca sobrecarga. Este protocolo trabaja mediante el descubrimiento de rutas y el mantenimiento de las mismas.
El primero, descubre el camino que debe seguir el paquete desde el nodo origen al destino, solo cuando existe una petición, denominada flooding route request (RREQ), la cual es enviado por los nodos vecinos y retransmitida en modo difusión, y permite determinar cuál es la mejor ruta.

Al llegar el paquete a su destino, se envía un mensaje RREP, para confirmar la recepción del mismo. Los nodos vecinos intermedios utilizan únicamente el camino que se encuentra en la cabecera de esta manera conocen el nodo al que deben reenviar el paquete.
Debido a su funcionamiento minimiza la sobrecarga, ya que envía mensajes periódicos, solo en caso de ser necesario él envió de información.

- **Temporally-Ordered Routing Algorithm (TORA)**

TORA, es un protocolo utilizado en grandes redes, puede trabajar como proactivo o reactivo. Los nodos almacenan información solo de sus nodos vecinos, como lo realiza un protocolo reactivo, es una ventaja cuando la topología de red es cambiante, pero al mismo tiempo puede estará trabajando como proactivo, utilizando las tablas de enrutamiento de los nodos que se encuentran conectados en la red.

No utiliza la ruta más corta, sino que proporciona diferentes trayectos para que le paquete llegue desde su origen hasta el destino. Lo que hace es fijar un peso a cada nodo, y el paquete es enviado desde un nodo a otro si es de mayor a menor peso. No usa rutas del camino más corto, algo inusual en la mayoría de los protocolos, lo que hace es fijar unas alturas a cada nodo, y al no utilizar rutas idóneas, minimiza la sobrecarga de la red.

1.3.4.2 Protocolo de enrutamiento Proactivo

El protocolo de enrutamiento proactivo, mantiene sus rutas siempre disponibles, para que los datos sean enviados a sus nodos móviles que comparten información periódicamente, al ser enviado el paquete, este escoge la mejor ruta no precisamente la más corta, confirmando su entrega al destino, es por eso que las tablas de enrutamiento de sus nodos siempre están actualizados, generando esto una inundación de paquetes de control de tráfico de red, aun sin ser enviado el paquete, esto puede generar un exceso en la red, el consumo energético y que aumente su ancho de banda, este protocolo tiene periódicamente una verificación de sus rutas, los protocolos que pertenecen al enrutamiento proactivo siendo los más utilizados los siguientes:

- **Destination-Sequenced Distance Vector Routing (DSDV)**

DSDV, es un protocolo proactivo, que igual que AODV trabaja bajo el algoritmo Vector-Distancia, a diferencia del otro protocolo este genera tráfico al enviar una petición ya que su tabla de enrutamiento se está actualizando constantemente sin necesidad de ninguna petición.

El protocolo escoge la ruta más corta mediante el número de saltos dependiendo de a que distancia se encuentre el nodo receptor, realizando actualizaciones en sus tablas de enrutamiento cuando envía información a sus nodos más cercanos.

La ruta más óptima, es la que tiene el número de menor saltos, si el enlace por el que viaja el paquete no funciona este asigna un número de salto con un valor infinito lo que indica que es una ruta inalcanzable.

- **Optimized Link State Routing Protocol (OLSR)**

OLSR, es un protocolo de enrutamiento proactivo, cada nodo dispone de una actualización de la información de su tabla de enrutamiento, del estado y disposición de todos los nodos conectados a la misma red, donde periódicamente se envían mensajes de HELLO.
Funciona bien en redes con gran número de nodos, y en una topología aleatoria e impredecible.

Mensajes

Los paquetes UDP se transmiten por la red, por el puerto de comunicación 698. Los mensajes enviados tienen un número que permite a los nodos reconocer la información más actualizada.
Para el funcionamiento del protocolo hay 3 tipos de mensajes:

HELLO.- Este mensaje se encarga de realizar una actualización de nodos conectados en la red, y de señalización y elección de MPRs.

Topology Control.- Permite que cada nodo tenga información actualizada de la topología de red de modo que puedan calcular las tablas de enrutamiento.

MID.- Descubre las múltiples interfaces en un nodo.

- **Wireless Routing Protocol (WRP)**

WRP, es un protocolo vector distancia, utiliza el algoritmo Bellman-Ford el cual permite calcular los caminos entre los nodos vecinos. En una red MANET, este protocolo presenta características que reducen los bucles en el camino y ayudan en la entrega de los paquetes de una manera fiable.

Al igual que DSDV, WRP tiene actualizadas las tablas de estado de la red, como son la entrada para cada nodo destino, siguiente salto, distancia y el nodo antecesor, enviando un paquete de actualización, de esta manera cada nodo conoce la ruta de acceso a los nodos destinos.
Este protocolo utiliza un grupo de tablas para actualizar la información, tabla de distancia, ruteo, costo de enlace.

- **A loop-free distance-vector routing protocol (BABEL)**

BABEL, es un protocolo nuevo, utiliza un algoritmo vector distancia, eficaz para ser utilizado en redes tipo infraestructura y ad-hoc. Se basa en las características del protocolo DSDV y AODV. Es un protocolo proactivo pero puede adaptarse a características reactivas.

- **Better Approach to Mobile Ad-hoc Networks (BATMAN)**

BATMAN, es un protocolo de enrutamiento que se encuentra en investigación con el objetivo de suplir a OLSR, está enfocado en las redes móviles inalámbricas. BATMAN solo tiene conocimiento del siguiente salto, de esta manera los nodos solo almacenan y conservan esta información para cada nodo destino.

Este protocolo, selecciona el nodo del siguiente salto para utilizarlo como entrada hacia el nodo destino, por lo que no calcula la ruta completa, ya que encuentra otros nodos y define la mejor ruta en base a los nodos vecinos, además de que envía un mensaje que se distribuye por toda la red de modo que mantiene informados de la existencia de nuevos nodos accesibles y cambios en la topología, por lo que se considera como un protocolo rápido y eficiente.

1.3.4.3 Protocolo de Enrutamiento Hibrido

El protocolo proactivo tiene menor latencia y tiene mayor sobrecarga en la red, ancho de banda y energía mientras que el protocolo reactivo tiene mayor latencia y menor sobrecarga en la red, para este tipo de escenarios se utiliza un protocolo de enrutamiento híbrido, que es una implementación entre un protocolo proactivo y un reactivo, los protocolos que pertenecen al enrutamiento híbridos siendo los más utilizados los siguientes:

- **Core-Extraction Distributed Ad Hoc Routing (CEDAR)**

CEDAR, es un protocolo de enrutamiento para redes ad-hoc, busca las mejores rutas para satisfacer los requerimientos de calidad de servicio en la comunicación en una red MANET.
Tiene tres componentes, establecimiento y mantenimiento de una infraestructura auto organizado.

- **Zone Routing Protocol (ZRP)**

ZRP, protocolo de encaminamiento híbrido, combinada las características de un protocolo proactivo y un reactivo, y lograr conservar las tablas de enrutamiento actualizadas sin causar una sobrecarga en la red.

Para establecer una comunicación se la realiza a través del protocolo de enrutamiento IARP (Intrazone Routing Protocol), la que permite actualizar la información de los nodos vecinos de una forma anticipada, como lo hace un protocolo proactivo.

ZRP separa la red en zonas, por lo que la comunicación entre diferentes zonas sea realiza por el protocolo IERP (Interzone Routing Protocol), que trabaja como protocolo reactivo. Cuando ocurre un cambio en la topología de la red, y se necesita un camino hacia el nuevo nodo usando IERP, se utiliza BRP (Bordercast Resolution Protocol), el mismo que envía un mensaje de petición de ruta a sus nodos periféricos, de modo que pueda informar el camino entre el nodo origen y el destino. Este protocolo puede utilizar múltiples trayectorias. Figura 7.1.

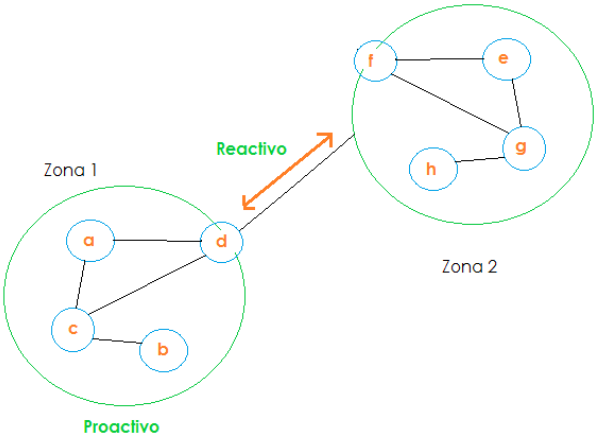


Figura 7.1 Protocolo ZRP

Realizado por Kateryne K. Niama. 2017

1.4 Herramientas de simulación de red

Son software, que permiten la simulación de escenarios reales, donde el usuario puede interactuar creando o modificando escenarios previamente diseñados adaptándose a diferentes necesidades. Además nos permite estudiar cómo trabaja una red, configurando los parámetros necesarios, un simulador permite, diseñar, analizar, verificar el rendimiento de los algoritmos y protocolos, con la posibilidad de crear redes básicas hasta complejas, para luego ser implementados físicamente si así se requiere.

Los simuladores tienen como ventaja que son utilizados para uso académico, y si la red está bien diseñada, pueden brindar una idea del funcionamiento de la red real. Los simuladores son

desarrollados mediante varios lenguajes de programación como C++, phyton, java, HTML. Dentro de las herramientas más utilizadas tenemos las siguientes:

1.4.1 Simulador de red OPNET

En la figura 8.1, se puede ver la interfaz gráfica del simulador OPNET, que es una herramienta para simular el comportamiento y el rendimiento de cualquier tipo de red. La principal diferencia con otros simuladores reside en su potencia y versatilidad. Este simulador hace posible trabajar con el modelo OSI, desde la capa 7 hasta la modificación de los parámetros físicos más esenciales.

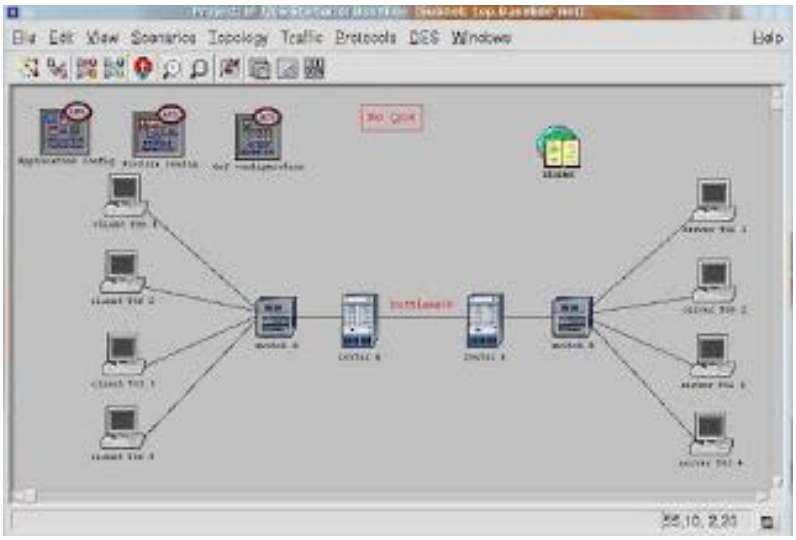


Figura 8.1 Interfaz de OPNET

Realizado por: http://blog-del-linformatico.blogspot.com/2008/09/simulador-opnet.html

1.4.2 Simulador de red NS-2

Network Simulator 2, es un simulador de eventos discretos de código abierto, desarrollado en C++, con limitadas características como la interfaz gráfica, está orientado a programación, es un software manejado para simular diferentes tipos de redes sean estas cableadas o inalámbricas.
Trabaja con scripts OTcl, que permiten diseñar el modelo de red a simular, los componentes y las características de comportamiento de la misma, así como protocolos y fuentes de tráfico. Figura 9.1.

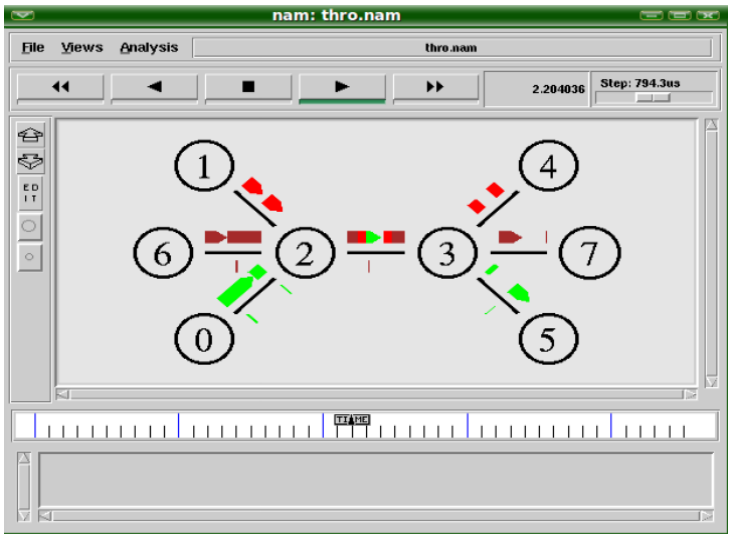


Figura 9.1 Interfaz de NS-2

Realizado por: https://academiccollegeprojects.com/ns2-projects/

1.4.3 Simulador NS-3

NS-3 es una versión mejorada de NS-2, cabe recalcar que ns-3 y ns-2 son dos simuladores distintos y no es compatibles el uno con el otro. El simulador está desarrollado con lenguaje de programación C++, y lenguaje Python que también es una ayuda en la parte de programación, NS-3 genera un archivo tipo pcap, donde puede ser leído con la ayuda del programa Wireshark donde nos permite interpretar la información enviada en la red, para visualizar en modo gráfico es necesario instalar NAM (Network Animador), es fácil de utilizar y también nos da resultados obtenidos de la red configurada. La figura 10.1

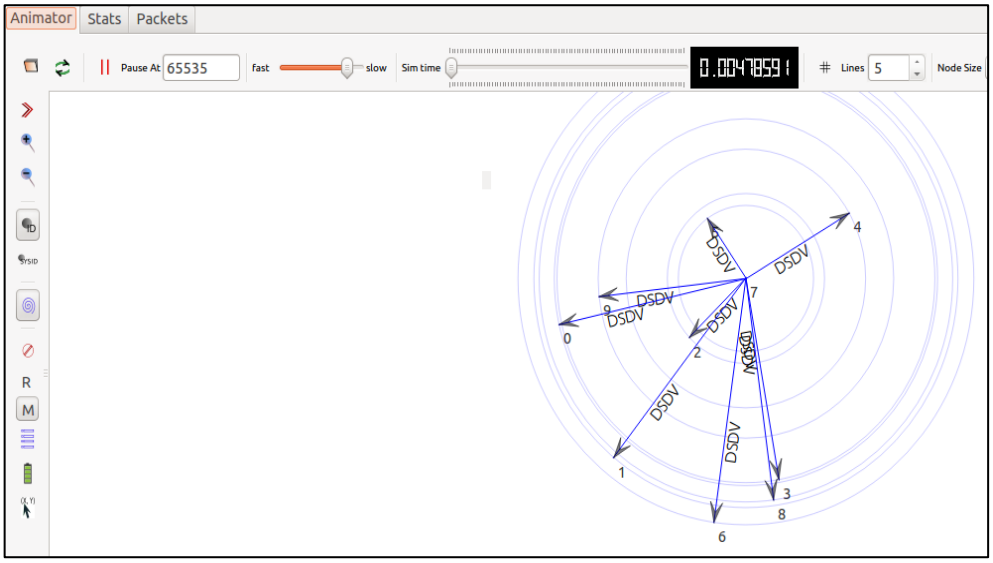


Figura 10.1 Interfaz Gráfica de NS-3

Realizado por Kateryne K. Niama. 2017

Características Generales de NS (Network Simulator)

Requerimientos del sistema.- NS es un paquete compuesto por un conjunto de componentes requeridos y otros tantos opcionales, este paquete contiene un script de instalación para configurar, compilar e instalar estos componentes. Para instalar este software se requiere cumplir con las especificaciones, (Morales Mosquera & Tabares, 2011).

Sistema operativo.- Plataformas Unix y Windows.

Interfaz de usuario.- Tiene un editor por código abierto, se diseña y se configuran, los protocolos y los elementos de la red.

1.4.4 Simulador OMNET++

La interfaz gráfica de OMNET++, fue creado en el año 2003 por András Varga quien es el principal programador del software quien estudio en la Universidad Técnica de Budapest, el simulador es un software libre.

Es un simulador eficiente de eventos discretos enfocado a las redes, está compuesto por un kernel del simulador, con soporte GUI: gráfico (tkenv) o texto (cmdenv), Simulation Class Library, además cuenta con una arquitectura flexible, soporta plataformas en MAC os, Windows y UNIX y su lenguaje de programación es C++. Figura 11.1

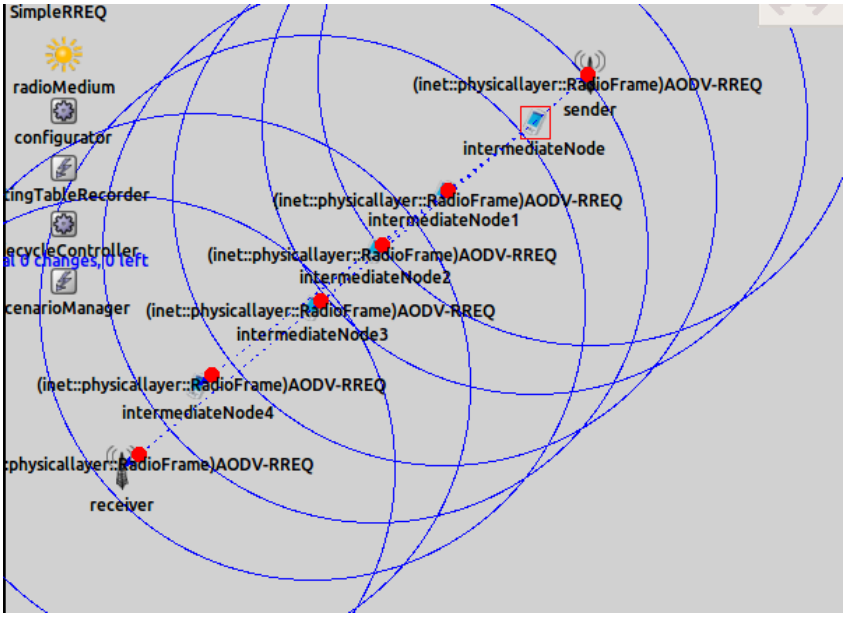


Figura 11.1 Interfaz de OMNET++

Realizado por Kateryne K. Niama. 2017

Puede ser utilizado para modelar el tráfico de información sobre las redes, los protocolos de red, las redes de colas, multiprocesadores y otros sistemas de hardware distribuido, además para validar arquitecturas de hardware y evaluar el rendimiento de sistemas complejos. (Santos Díaz, 2013).

Ventajas

- Es un software libre
- Es para uso académico e investigativo
- Fácil de crear un módulo
- Disponible para Windows y UNIX
- Lenguaje de programación en C++

Desventajas

- Es necesario saber programar en lenguaje .NED
- Modo gráfico es rígido
- Pocos modelos de equipos y enlaces
- Es complejo su uso

1.5 Video Streaming

El streaming o transmisión, es una forma de distribuir contenido multimedia a través de una red, para que el usuario pueda reproducirlo a la misma vez que se lo descarga, el video es responsable del 30% del tráfico total de internet. Como se puede ver en la figura 12.1

Los problemas que pueden aparecer en la transmisión de un video en tiempo real son:

- Pérdida de paquetes.- Cuando se pierden paquetes en un stream, la calidad en recepción se ve afectada. En un streaming de video, esto significaría una pérdida de fotogramas, cuando esto sucede, el usuario reproduce el siguiente fotograma que ha llegado correctamente, y por lo tanto se pierde información, resultando en una calidad mala del video. (Arroyo Ruiz, 2015)

- Jitter.- Es la variación en el tiempo de llegada de los paquetes, los tiempos de los mismos varían, lo que puede producir que algunas imágenes se congelen hasta que llega la siguiente, que podría reproducirse por un periodo de tiempo menor que las demás. (Arroyo Ruiz, 2015)

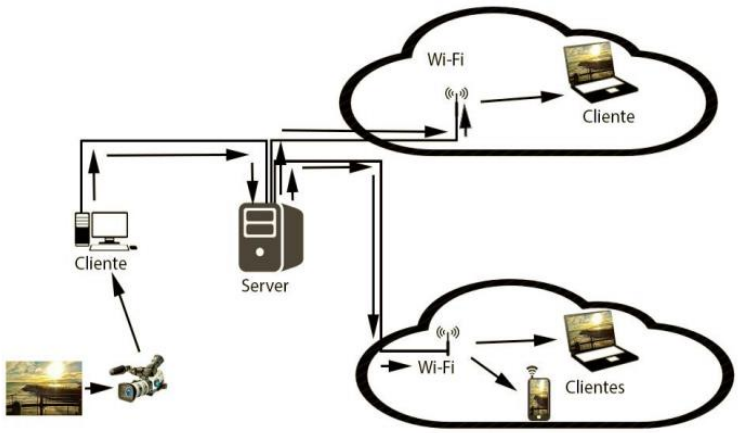


Figura 12.1 Esquema Streaming de video.

Realizado por https://repositorio.unican.es/xmlui/bitstream/handle/10902/6670/376267.pdf?sequence=1

1.5.1 Factores de Calidad de Servicio (QoS)

La entrega de señales de voz, video y fax desde un punto a otro no se puede considerar realizada con un éxito total a menos que la calidad de las señales transmitidas satisfaga al usuario. Entre los factores que afectan a la calidad se encuentran los siguientes:

- Requerimientos de ancho de banda (bandwidth): la velocidad de transmisión de la infraestructura de red y su topología física.
- Latencia o retardo (delay): Es el tiempo que se demora él envió del paquete desde el origen hasta el destino a través de la red.
- Variación en el tiempo (Jitter): variación en los tiempos de llegada entre los paquetes. Figura 14.1
- Perdidas de paquetes: cuando un paquete se pierde en la red. Figura 13.1

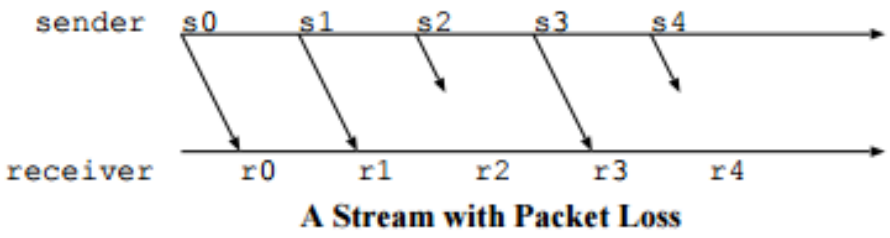


Figura 13.1 Stream con pérdida de paquetes.

Realizado por https://repositorio.unican.es/xmlui/bitstream/handle/10902/6670/376267.pdf?sequence=1

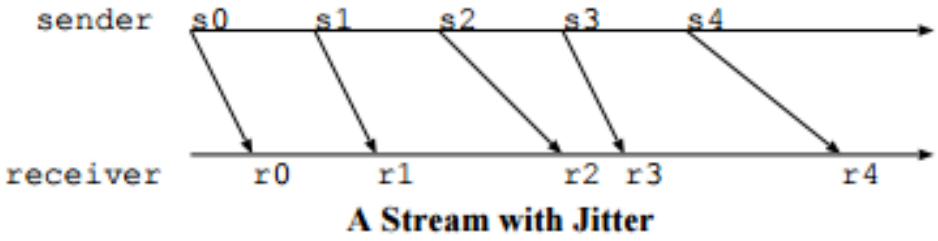


Figura 14.1 Stream con Jitter.

Realizado por https://repositorio.unican.es/xmlui/bitstream/handle/10902/6670/376267.pdf?sequence=1

CAPÍTULO II

2 MARCO METODOLÓGICO

2.1 Desarrollo del trabajo de titulación

En la figura 1.2, se describe el procedimiento a seguir para el desarrollo de la investigación.

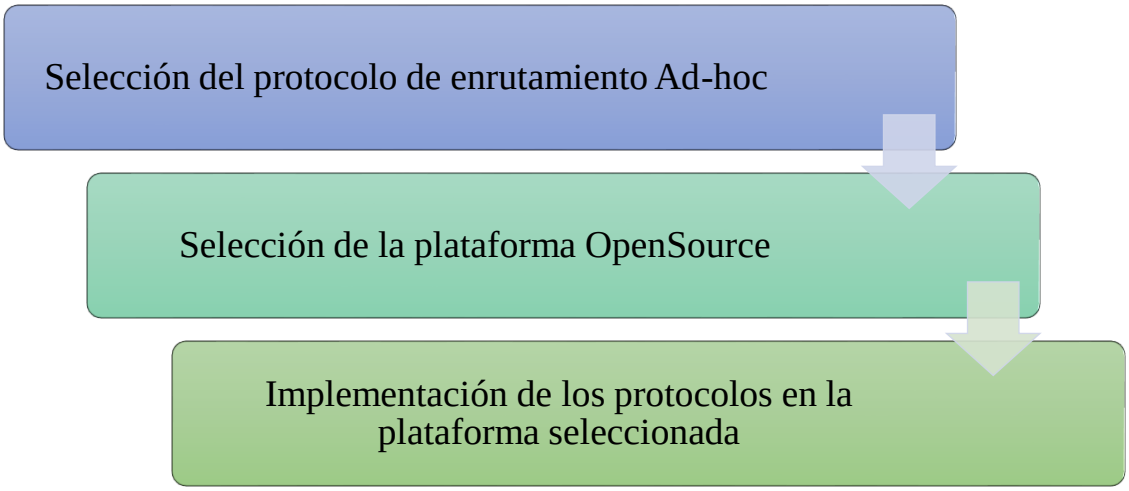


Figura 1.2 Proceso para el desarrollo del proyecto.

Realizado por Kateryne K. Niama. 2017

En primera instancia, se realizó un estudio previo de los protocolos de enrutamiento comúnmente utilizados en las redes MANET, como son los proactivos y reactivos. Para posteriormente realizar el estudio de las plataformas de simulación OMNET++ y NS-3, mediante la comparación de sus características y accesibilidad. Finalmente, en la plataforma seleccionada se procede a la configuración lógica y simulación de los protocolos de enrutamiento, donde se realiza el análisis de tráfico evaluando 3 variables: perdida de datos, jitter y retardo, con la ayuda del software analizador de tráfico, se obtendrán los resultados para evaluar que protocolo es el más adecuado para aplicaciones de video.

2.1.1 Selección del protocolo de enrutamiento Ad-hoc

Los protocolos de enrutamiento Ad-hoc se clasifican en protocolos proactivos y reactivos. Figura 2.2. Son diseñados específicamente para este tipo de redes inalámbricas, en este apartado se abordara el detalle del funcionamiento de los protocolos para poder establecer una comparación entre ellos, por las características que presentan.

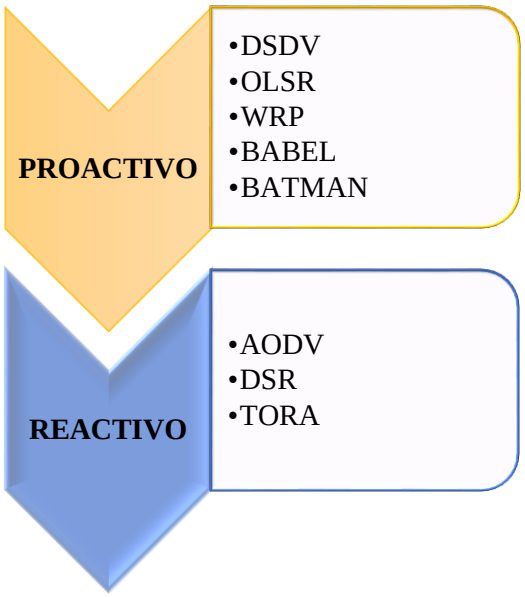


Figura 2.2 Protocolos seleccionados

Realizado por Kateryne K. Niama. 2017

Características de los protocolos de enrutamiento:

Métricas de enrutamiento.- Es una medida que establece la elección de la ruta.

Algoritmo de enrutamiento.- Se encarga de determinar la ruta de un coste mínimo por el cual va hacer enviado el paquete hasta llegar a su destino.

Libre de lazos.- Establece si el protocolo es idóneo de impedir los lazos en la red.

Escalabilidad.- Ve si el protocolo admite la inserción de un nuevo nodo sin alterar el rendimiento de la red.

Confiabilidad.- Garantiza la entrega del paquete a su destino.

Balance de carga.- Evita que exista sobrecarga en la red distribuyendo la carga y el tráfico.

Control de congestión.- Es la técnica que permite descubrir y corregir los errores que se presenta en l red, que pueden ser causadas debido a que le paquete no llego a su destino.

Implementación.- Plataforma donde se puede programar los protocolos de enrutamiento.

Para determinar que protocolo es más óptimo para redes MANET, se describe cuáles son los tipos de protocolos de enrutamiento ad-hoc tanto proactivos como reactivos, así como las principales características que presentan cada uno de ellos. Posteriormente al estudio, se realizó un análisis tomando en cuenta las características más significativas como: la implementación en las plataformas de simulación para redes inalámbricas, la confiabilidad del protocolo en la entrega del paquete a su destino, su escalabilidad admite la inclusión de nuevos nodos sin alterar la red, su métrica de enrutamiento nos ayuda a encontrar la ruta más rápida y corta, y por último el control de congestión que nos permitirá corregir errores en la red cuando el paquete no ha sido enviado correctamente. [10]

2.1.1.1 Comparación entre protocolos proactivos

Para realizar la comparación entre los diferentes protocolos proactivos, DSDV, OLSR, WRP, BABEL y BATMAN, se realizó un análisis donde se evalúan las características que estos presentan de manera que se pueda escoger al protocolo más eficiente para trabajar con redes MANET. Como se puede visualizar en la tabla 1.2, la clasificación de los protocolos comúnmente utilizados.

Tabla 1.2 Comparación entre los protocolos Proactivos

PROTOCOLOS / ***CARACTERÍSTICAS***	**DSDV**	**OLSR**	**WRP**	**BABEL**	**BATMAN**
MÉTRICA DE ENRUTAMIENTO	Camino más corto	Camino más corto	Camino más corto	Calidad del enlace	Próximo mejor salto para cada destino
ALGORITMO DE ENRUTAMIENTO	Vector Distancia	Estado de Enlace	Vector distancia	Vector distancia	Estado de enlace
ESCALABILIDAD	No	No	No	Si	Depende de la pérdida de paquete
CONFIABILIDAD	No	Si	No	No	Si
BALANCE DE CARGA	No	No	No	No	No
LIBRE DE LAZOS	Si	Si	Si	Si	Si
CONTROL DE CONGESTIÓN	Si	No	No	No	Si
MÚLTIPLES PLATAFORMA	Linux Ns-3	Linux Android Ns-3	Linux	Linux	Linux Android

Realizado por Kateryne K. Niama. 2017

Como se puede observar en la tabla 2.1, el protocolo OLSR es mejor en comparación a los otros protocolos proactivos (DSDV, WRP, BABEL y BATMAN), pero debido a los estudios que ya se han realizado sobre OLSR, se selecciona al protocolo proactivo DSDV con el motivo de extender su estudio y ya que presenta buenas características técnicas, este protocolo escoge la ruta más corta por número de saltos para realizar él envió de datos, además de que tiene similitudes con el protocolo AODV. (Rodriguez Pineda, 2015).

2.1.1.2 Comparación entre protocolos reactivos

Posterior al análisis de las características de los protocolos proactivos, se procedió a realizar la misma comparación entre protocolos reactivos, con el objetivo de seleccionar el más eficiente para redes MANET, se evalúan los mismos parámetros, como se muestra detalladamente en la tabla 2.2.

Tabla 2. 2 Comparación entre los protocolos Reactivos

PROTOCOLOS / ***CARACTERÍSTICAS***	**AODV**	**DSR**	**TORA**
MÉTRICA DE ENRUTAMIENTO	Camino más rápido y más corto	Camino más corto	Conteo de saltos
ALGORITMO DE ENRUTAMIENTO	Vector distancia	Source routing	Reversión de enlaces
ESCALABILIDAD	No	No	No
CONFIABILIDAD	Si	Si	Si
BALANCE DE CARGA	No	No	No
LIBRE DE LAZOS	Si	Si	Si
CONTROL DE CONGESTIÓN	Si	No	No
MÚLTIPLES PLATAFORMA	Linux Matlab Omnet++ Ns-3	Linux Ns-3	Implementación en hardware Ns-2.34

Realizado por Kateryne K. Niama. 2017

Como se puede observar en la tabla 2.2, el protocolo AODV presenta mejores características en comparación con los otros protocolos reactivos. Del análisis realizado se determina que el protocolo reactivo AODV, es el más utilizado para redes inalámbricas, ya que es óptimo y estable, y además es un protocolo de Vector Distancia que busca la ruta más corta y rápida para transmisión de información al destino.

Para el desarrollo del trabajo de investigación, se seleccionó un protocolo de cada categoría Proactivo y Reactivo, para realizar el estudio de los protocolos en relación a un análisis de tráfico, con el objetivo de establecer el más eficiente para implementar en la plataforma seleccionada**.**

En la tabla 3.2 muestra las principales características de los protocolos AODV y DSDV, los que fueron seleccionados para realizar el análisis de los protocolos de enrutamiento ad-hoc. Los mismos que proporcionaran un estudio eficiente para el uso dentro de las redes MANET.

Tabla 3.2 Características de los protocolos de enrutamiento

Protocolos / **Características**	**AODV**	**DSDV**
Métrica de enrutamiento	Camino más rápido y Más corto	Camino más corto
Algoritmo de enrutamiento	Vector distancia	Vector distancia
Libre de lazos	Si	Si
Escalabilidad	No	No
Confiabilidad	Si	No
Balance de carga	No	No
Control de congestión	Si	Si
Múltiples Plataformas	Ns-3 Omnet++ Matlab Linux	Ns-3 Linux

Realizado por Kateryne K. Niama. 2017

2.1.2 Selección de la plataforma OpenSource

En el presente trabajo de titulación, se evalúa los siguientes elementos claves a la hora de elegir una herramienta de simulación.

- **Especificación de Hardware**

En la tabla 4.2 se observa las características del hardware que se recomienda utilizar para la configuración de los simuladores de red sean estos OMNET++ o NS-3.

Tabla 4.2 Especificaciones del Hardware

Parámetro	**Hardware**
Equipo	Laptop Toshiba, Satélite
Memoria RAM	12 GB DDR3
Procesador	I5-3210m 2.5Ghz
Disco Duro	100 GB

Fuente: https://dspace.unl.edu.ec/jspui/bitstream/123456789/11585/1/Rodr%C3%ADguez%20Pineda,%20Gabriela%20Maribel.pdf

- **Requisitos Software**

Para realizar la instalación de la máquina virtual y utilizar el sistema operativo Ubuntu, se debe cumplir con los requisitos de software descritos en la tabla 5.2, ya que este sistema es necesario para el manejo de las plataformas de simulación de redes.

Tabla 5.2 Requisitos de Software

Parámetro	**Software**
Máquina Virtual	Virtual Box 5.1.10
Memoria RAM	2 GB
Disco Duro	50 GB

Realizado por Kateryne K. Niama. 2017

Segundo se instala las plataformas de simulación OMNET++ o NS-3, pero no simultáneamente ya que los programas necesitan cumplir con los requisitos de software y hardware para su buen funcionamiento. En la tabla 6.2, se puede ver las especificaciones del software que se utilizaran, para la simulación de las redes.

Tabla 6.2 Especificaciones del software

Parámetro	Software	Versión
Sistema Operativo	XUBUNTU	14.04
Simulador de red	NS-3 OMNET++	3.23 3.4

Fuente: https://dspace.unl.edu.ec/jspui/bitstream/123456789/11585/1/Rodr%C3%ADguez%20Pineda,%20Gabriela%20Maribel.pdf

- **Plataforma de simulación de redes MANET**

Para seleccionar la plataforma de simulación, se realizó una comparación de los programas comúnmente utilizados para redes MANET adquiridos de estudios realizados por otros proyectos de investigación que se basan en el análisis y simulación de protocolos de enrutamiento ad-hoc. El uso e instalación de los programas permitió establecer bajo criterios propios los parámetros de la tabla 7.2, que detalla las características principales de Omnet++ y Ns-3, de igual manera se pudo establecer los niveles de valorización que se muestran en la tabla 2.8, que permitirán analizar y comparar entre plataformas y así establecer cuál es la más indicada para simular los protocolos de enrutamiento de redes MANET. [8], [10]

Tabla 7.2 Características de las plataformas OMNET++ y NS-3

Simuladores / **Características**	**OMNET++**	**NS-3**
Uso investigativo	Alto	Alto
Tipo de licencia	GNU	GNU
Curva de aprendizaje	Alto	Alto
Múltiples Plataformas	Windows Mac Unix	Windows Mac Unix Linux
Interfaz gráfica	Medio	Medio
Visualización de resultados	Medio	Alto
Aplicación de módulos	Alto	Alto
Soporte a redes inalámbricas	Medio	Alto

Lenguaje de programación	C++ .NED	C++ Pyton NetAnim
Flexibilidad	Alta	Baja
Complejidad	Alta	Media
Artículos Publicados	3,900	142,00

Realizado por Kateryne K. Niama. 2017

Para decidir que plataforma es la más adecuada para el desarrollo del trabajo de titulación, se realizó una tabla de porcentajes para valorar las características de cada software, como se muestra en la tabla 8.2.

Tabla 8.2 Nivel de valorización

Nivel de valorización	**Porcentaje %**
Alto	70 - 100
Medio	36 - 69
Bajo	0 - 35
Tipo de licencia	
GNU	51 - 100
GPLv2	0 - 50
Múltiples plataformas	
4 o más	51 -100
menos de 4	0 - 50
Lenguaje de programación	
3 o más	51 - 100
menos de 3	0 -50
Número de artículos publicados	
140,000 o más	51 - 100
menos de 140,00	0 - 50

Realizado por Kateryne K. Niama. 2017

Como se observa en la tabla 9.2 se ha valorado cada característica de cada plataforma con un porcentaje referente a la tabla 8.2.

Tabla 9.2 Valorización de las plataformas

PLATAFORMAS / CARACTERÍSTICAS	OMNET++ (%)	NS-3 (%)
Uso investigativo	85	85
Tipo de licencia	90	90
Curva de aprendizaje	85	85
Múltiples plataformas	45	75
Interfaz gráfica	75	60
Visualización de resultados	65	90
Aplicación de modelos	80	80
Soporte a redes inalámbricas	60	80
Lenguaje de programación	25	60
Flexibilidad	80	30
Complejidad	85	50
Artículos publicados	15	75
PROMEDIO	**65,83 %**	**71,67 %**

Realizado por Kateryne K. Niama. 2017

La tabla 9.2 muestra el total alcanzado en el análisis de las características de cada uno, dando así que NS-3 alcanza un total de 71,67% en comparación a la plataforma OMNET++ 65,83%, este resultado permite concluir que NS-3 es el más eficiente y óptimo para el estudio del presente trabajo, debido a sus características que obtiene valores altos lo cual da más confiabilidad que OMNET++, lo que permitió su selección para el análisis de los protocolos de enrutamiento AODV y DSDV.

2.1.3 Implementación de los protocolos en la plataforma seleccionada

2.1.3.1 Topología para redes MANET

La topología es la forma física o virtual que está diseñada una red, es decir, es la interconexión de dispositivos como: servidor, terminales o nodos entre otros.

Para redes MANET, su topología es en malla, debido a que los nodos están conectados unos con otros para poder establecer una comunicación óptima, y de esta manera realizar él envió del

paquete desde su nodo origen al nodo destino utilizando multi-saltos. Por esta razón la topología a implementarse es de tipo malla, ya que no necesita de un nodo central. Si falla un nodo no afecta a la red ya que todos están conectados entre sí, la información se puede enviar por diferentes caminos ya que estos son redundantes, como se muestra en la figura 3.2

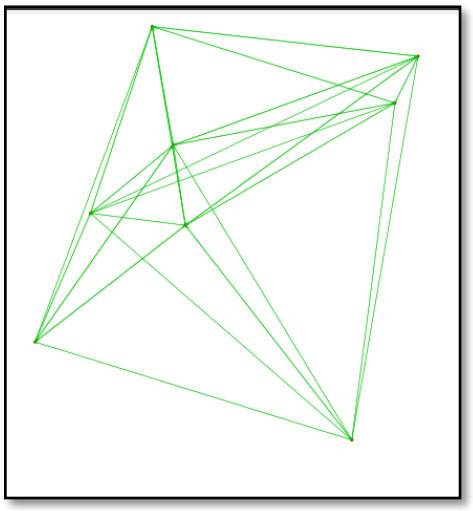


Figura 3.2 Topología en Malla

Realizado por Kateryne K. Niama. 2017

La topología en malla tiene las siguientes ventajas y desventajas:

Ventajas

- La red es robusta y confiable.
- Tiene enlaces redundantes

Desventajas

- Su instalación y mantenimiento es costosa.
- Su instalación y configuración es difícil.

2.1.3.2 Configuración de los protocolos en redes MANET

Para realizar el desarrollo de los protocolos en redes MANET en el simulador NS-3, se debe seguir el siguiente proceso como se muestra en la figura 4.2.

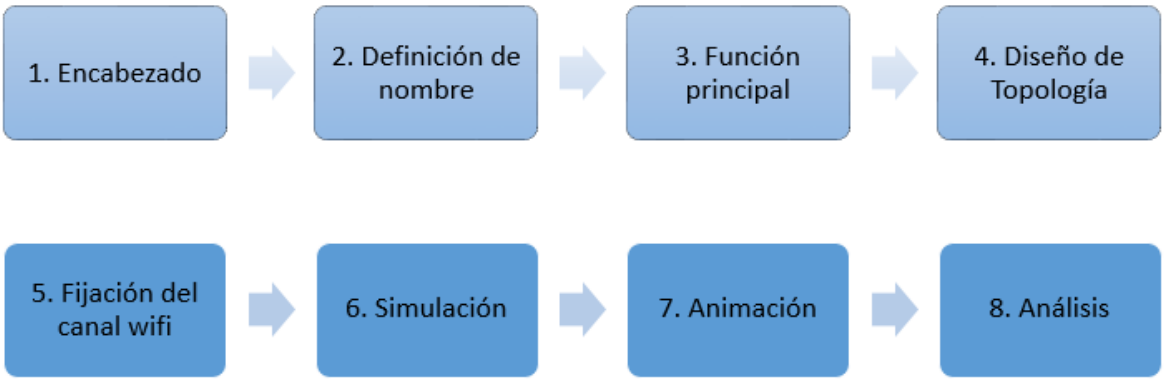


Figura 4.2 Diagrama de flujo para el diseño de los protocolos establecidos

Realizado por Kateryne K. Niama. 2017

- **Configuración del Protocolo AODV**

A continuación se detalla la configuración de cada una de las etapas, como se muestra en la figura 4.2.

1. **Encabezado de la programación**

Para el encabezado de la programación se debe incluir las siguientes librerías:

```
//MODULOS DE INCLUSION
#include <fstream>
#include "ns3/aodv-module.h"
#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/internet-module.h"
#include "ns3/mobility-module.h"
#include "ns3/point-to-point-module.h"
#include "ns3/wifi-module.h"
#include "ns3/v4ping-helper.h"
#include "ns3/netanim-module.h"
#include "ns3/flow-monitor-module.h"
#include <iostream>
#include <cmath>
```

2. Definición de nombre NS3

El espacio o definición de nombre ayuda a la agrupación de funciones que facilite el manejo de variables.

```
//DEFINICION DE NOMBRE
using namespace ns3;
```

3. Función principal

Las siguientes líneas de código muestran la función principal de la simulación, es el tipo de dato que se va a crear, el número de nodos, la distancia de cada nodo en la red y los paquetes que serán transmitidos.

```
//FUNCION PRINCIPAL
int main (int argc, char **argv)
{
Aodv test;
if (!test.Configure (argc, argv))
NS_FATAL_ERROR ("Configuration failed. Aborted.");

test.Run ();
test.Report (std::cout);
return 0;
}
Aodv::Aodv () :
nodos (10), //creación de los nodos
step (100), // distancia de los nodos
totalTime (20), // paquetes transmitidos
pcap (true),
printRoutes (true)
{
}
```

4. Clases para el diseño de la topología

NodeContainer: Es el que contiene los nodos donde permite crear, gestionar y acceder a cualquier nodo, que ayuda a ejecutar la simulación, que se está declarando un NodeContainer nodes.

NetDeviceContainer: Se debe tener una lista de los netdevice creados, al igual que se hizo con NodeContainer

Creación de los nodos:

```
//NETWORK
NodeContainer nodes;
NetDeviceContainer devices;
Ipv4InterfaceContainer interfaces;
```

El tipo de comunicación que se va a utilizar en la programación del algoritmo de enrutamiento es el estándar 802.11b que es Wifi.

5. **Fijación del canal Wifi**

```
//ESTABLECER CANAL WIFI UTILIZANDO HELPER
Void
Aodv::CreateDevices ()
{
NqosWifiMacHelper wifiMac = NqosWifiMacHelper::Default ();
wifiMac.SetType ("ns3::AdhocWifiMac");
YansWifiPhyHelper wifiPhy = YansWifiPhyHelper::Default ();
YansWifiChannelHelper wifiChannel = YansWifiChannelHelper::Default ();
wifiPhy.SetChannel (wifiChannel.Create ());
WifiHelper wifi = WifiHelper::Default ();
wifi.SetRemoteStationManager ("ns3::ConstantRateWifiManager", "DataMode", StringValue
("OfdmRate6Mbps"), "RtsCtsThreshold", UintegerValue (0));
devices = wifi.Install (wifiPhy, wifiMac, nodes);
```

- **Direccionamiento IP**

El tipo de dirección que se utiliza en la programación en IP versión cuatro, en las siguientes líneas de comando se muestran las direcciones para los nodos asignados se inicia en uno y aumenta de manera sucesiva hasta llegar a su último nodo.

```
void
Aodv::InstallInternetStack ()
{
AodvHelper aodv;
InternetStackHelper stack;
stack.SetRoutingHelper (aodv);
stack.Install (nodes);
Ipv4AddressHelper address;
address.SetBase ("10.0.0.0", "255.0.0.0");
interfaces = address.Assign (devices);
```

6. Simulación

Para que se ejecute el programa necesitamos de los siguientes parámetros.

```
//SIMULADOR
Simulator::Stop (Seconds (totalTime));
Simulator::Run ();
flowMonitor ->SerializeToXmlFile("aodvfow.xml", true, true);
Simulator::Destroy ();
}
```

7. Animación en NetAnim

Para la animación de la red programada se usa un módulo NetAnim.

```
//ANIMACION
AnimationInterface anim ("aodv.xml");
anim.SetMobilityPollInterval(Seconds(1));
anim.EnablePacketMetadata(true);
```

En la figura 5.2 se puede observar la interfaz de NetAnim

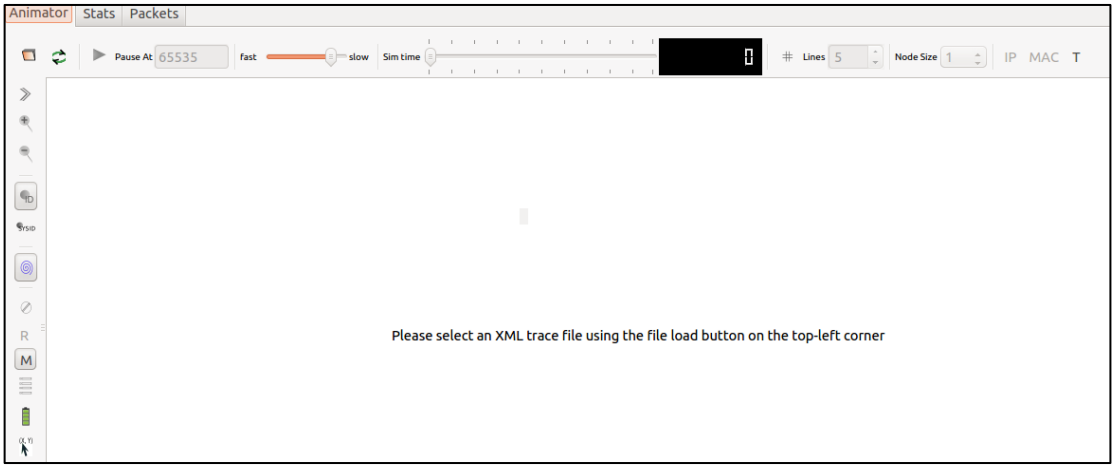


Figura 5.2 NetAnim

Realizado por Kateryne K. Niama. 2017

8. Análisis

Para el análisis de tráfico se usa el módulo FlowMonitor.

Parámetros para el uso de FlowMonitor

```
Ptr<FlowMonitor> flowMonitor;
FlowMonitorHelper flowHelper;
flowMonitor = flowHelper.InstallAll ();
```

En la figura 6.2 se puede observar que mediante el comando FlowMonitor podemos obtener resultados del análisis que se realizó.

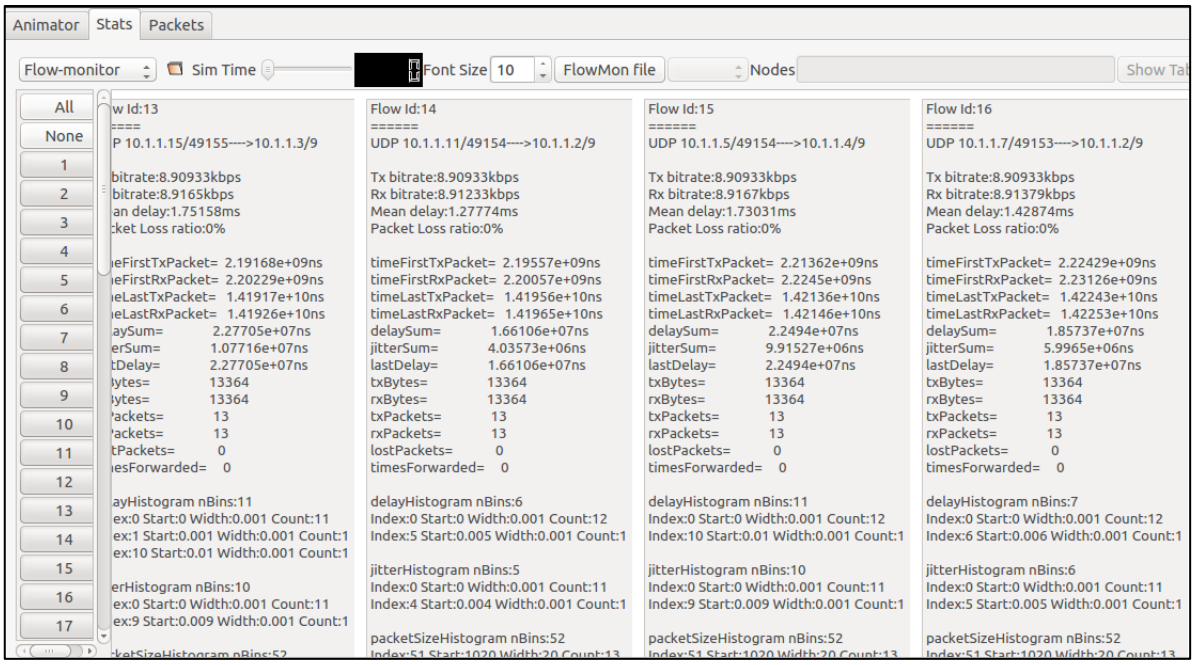


Figura 6.2 Resultados mediante FlowMonitor

Realizado por Kateryne K. Niama. 2017

Configuración del Protocolo DSDV:

1. **Encabezado de la programación:**

```
#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/applications-module.h"
#include "ns3/mobility-module.h"
#include "ns3/point-to-point-module.h"
#include "ns3/config-store-module.h"
#include "ns3/wifi-module.h"
#include "ns3/internet-module.h"
#include "ns3/dsdv-helper.h"
#include "ns3/flow-monitor-module.h"
#include "ns3/netanim-module.h"
#include <iostream>
#include <cmath>
```

2. **Definición de nombre NS3**

```
using namespace ns3;
```

3. **Función principal**

```
int main (int argc, char **argv)
{
Dsdv2 test;
uint32_t nodos = 10; //creacion de los nodos
uint32_t nSinks = 4;
double totalTime = 25.0;
std::string rate ("8kbps");
std::string phyMode ("DsssRate11Mbps");
uint32_t nodeSpeed = 10; // en m/s //velocidad del nodo
std::string appl = "all";
uint32_t periodicUpdateInterval = 15;
uint32_t settlingTime = 6;
double dataStart = 1.0;
bool printRoutingTable = true;
std::string CSVfileName = "Dsdv2.csv";
```

4. **Clases para el diseño de la topología**

Creación de los nodos:

```
//network
NodeContainer nodes;
NetDeviceContainer devices;
Ipv4InterfaceContainer interfaces;
```

5. **Fijación del canal Wifi**

```
void
Dsdv2::CreateDevices (std::string tr_name)
{
NqosWifiMacHelper wifiMac = NqosWifiMacHelper::Default ();
wifiMac.SetType ("ns3::AdhocWifiMac");
YansWifiPhyHelper wifiPhy = YansWifiPhyHelper::Default ();
YansWifiChannelHelper wifiChannel;
wifiChannel.SetPropagationDelay ("ns3::ConstantSpeedPropagationDelayModel");
wifiChannel.AddPropagationLoss ("ns3::FriisPropagationLossModel");
wifiPhy.SetChannel (wifiChannel.Create ());
WifiHelper wifi;
wifi.SetStandard (WIFI_PHY_STANDARD_80211b);
wifi.SetRemoteStationManager ("ns3::ConstantRateWifiManager", "DataMode", StringValue
(m_phyMode), "ControlMode",
StringValue (m_phyMode));
devices = wifi.Install (wifiPhy, wifiMac, nodes);

AsciiTraceHelper ascii;
wifiPhy.EnableAsciiAll (ascii.CreateFileStream (tr_name + ".tr"));
wifiPhy.EnablePcapAll (tr_name);
}
```

- **Direccionamiento IP:**

```
void
Dsdv2::InstallInternetStack (std::string tr_name)
{
DsdvHelper dsdv;
dsdv.Set ("PeriodicUpdateInterval", TimeValue (Seconds (m_periodicUpdateInterval)));
dsdv.Set ("SettlingTime", TimeValue (Seconds (m_settlingTime)));
InternetStackHelper stack;
stack.SetRoutingHelper (dsdv); // has effect on the next Install ()
stack.Install (nodes);
Ipv4AddressHelper address;
address.SetBase ("10.1.1.0", "255.255.255.0");
interfaces = address.Assign (devices);
if (m_printRoutes)
{
Ptr<OutputStreamWrapper> routingStream = Create<OutputStreamWrapper> ((tr_name +
".routes"), std::ios::out);
dsdv.PrintRoutingTableAllAt (Seconds (m_periodicUpdateInterval), routingStream);
}
```

6. **Simulación**

```
Simulator::Stop (Seconds (m_totalTime));
Simulator::Run ();
flowMonitor ->SerializeToXmlFile("dsdvfow.xml", true, true);
Simulator::Destroy ();
```

7. **Animación en NetAnim:**

```
AnimationInterface anim("dsdv2.xml");
anim.SetMobilityPollInterval(Seconds (1));
anim.EnablePacketMetadata(true);
```

8. Análisis

```
Ptr<FlowMonitor> flowMonitor;
FlowMonitorHelper flowHelper;
flowMonitor = flowHelper.InstallAll ();
```

El código completo de los protocolos de enrutamiento se encuentra en el anexo D y E.

CAPÍTULO III

3 EVALUACIÓN Y COMPARACIÓN DE RESULTADOS

3.1 Simulación y análisis

Para realizar el análisis de los protocolos de enrutamiento de redes MANET, se hicieron pruebas basándose en diferentes escenarios con respecto al número de nodos, tiempo y distancia. En cada escenario se evalúa 3 parámetros: pérdida de paquetes, retardo y jitter, los mismos que han sido seleccionados basados en estudios anteriores, siendo estos de gran importancia, para determinar que una red sea eficiente y confiable.

La recomendación para el análisis de estos parámetros, se puede observar en la tabla 1.3, donde se puede ver cada parámetro con su grado de importancia, para ofrecer una calidad de servicio en una transmisión de datos en redes inalámbricas.

Tabla 1. 3 Parámetros a ser medidos

Parámetros	Valor máximo
Pérdida de paquetes	5%
Jitter	10 ms
Retardo	7 ms

Fuente: http://www.scielo.org.co/scielo.php?script=sci_arttext&pid=S1692-17982014000100002

3.1.1 Simulación del primer escenario con el protocolo AODV

De estudios que se han realizado anteriormente, se ha sacado un promedio en relación a la distancia para el desarrollo de cada escenario, donde en los tres casos se analizó con respecto a los nodos, la distancia y el tiempo.

En la figura 1.3 se observa la topología en malla con 5 nodos.

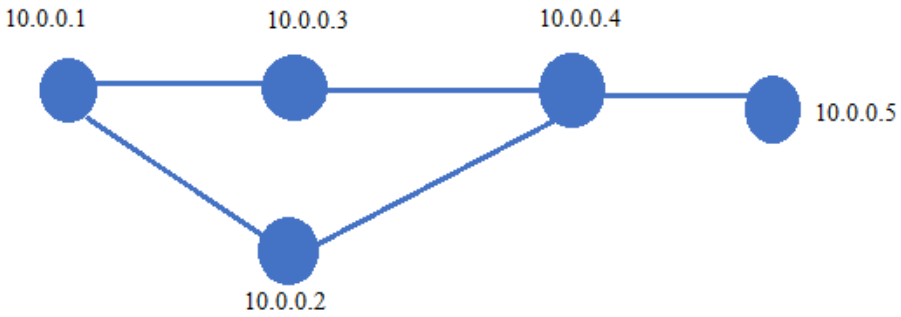


Figura 1.3 Topología con 5 nodos

Realizado por Kateryne K. Niama. 2017

Caso 1:

Se analizó los siguientes parámetros: nodos, distancia y tiempo, como se muestra en la tabla 2.3.

Tabla 2. 3 Parámetros del escenario 1 - caso 1

Parámetros	**Cantidad**
Nodos	10
Distancia	25 m
Tiempo	10 s

Realizado por Kateryne K. Niama. 2017

En la figura 2.3 se muestra el diseño del primer caso con respecto al primer escenario, donde se puede observar que todos los nodos creados se encuentran en su zona de cobertura.

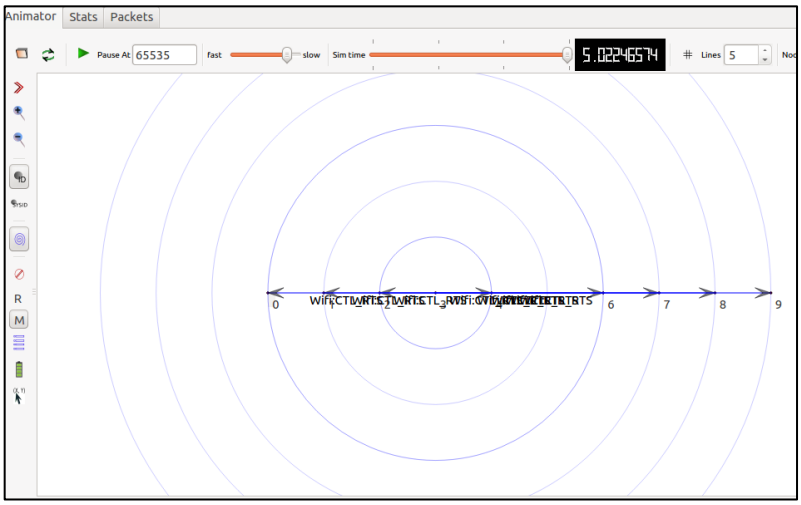


Figura 2. 3 Diseño del escenario 1 - caso 1

Realizado por Kateryne K. Niama. 2017

En la figura 3.3 se observa los resultados obtenidos del primer caso con sus respectivos parámetros.

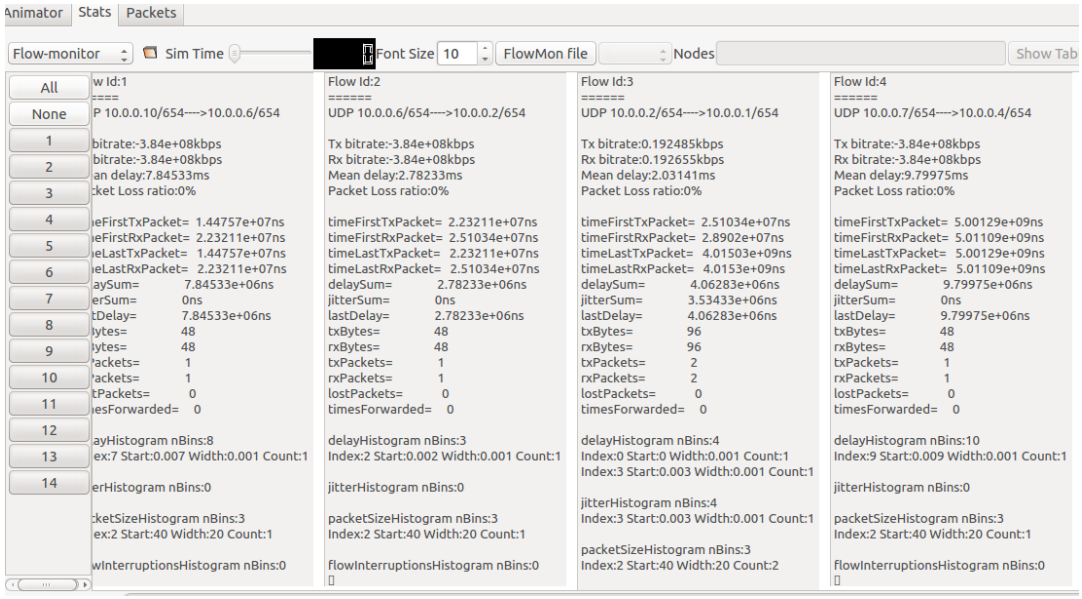


Figura 3. 3 Resultados del escenario 1 - caso 1

Realizado por Kateryne K. Niama. 2017

En la tabla 3.3 se puede observar la tabla de enrutamiento de la primera del escenario 1 – caso 1.

Tabla 3.3 Tabla de enrutamiento de la primera prueba del escenario 1 - caso 1

Dirección origen	**Dirección destino**	**Pérdida de paquete (%)**	**Retardo (ms)**	**Jitter (ms)**
10.0.0.1	10.0.0.2	21	46	-
10.0.0.1	10.0.0.3	23	2	44
10.0.0.1	10.0.0.4	18	2	0
10.0.0.1	10.0.0.5	15	32	30
10.0.0.1	10.0.0.7	22	2	30
10.0.0.1	10.0.0.8	25	2	0
10.0.0.1	10.0.0.9	15	2	0
10.0.0.1	10.0.0.10	21	2	0

Realizado por Kateryne K. Niama. 2017

Se realizó un número de 10 pruebas para sacar un promedio de como en 10 segundos afecta a la red, como se muestra en la tabla 4.3.

Tabla 4.3 Análisis del escenario 1 - caso 1

Número de pruebas	Pérdida de paquetes (%)	Retardo (ms)	Jitter (ms)
1	20	11,25	14,86
2	18	8,782	12,064
3	21	9,39	14,065
4	17	9,8	12,582
5	22	10,403	15,72
6	21	9,793	15,678
7	25	12,773	14,514
8	24	14,817	16,432
9	18	10,821	12,901
10	19	12,628	14,645
Promedio	**21**	**11,05**	**14,35**

Realizado por Kateryne K. Niama. 2017

Como se puede observar en la gráfico 1.3 el promedio del caso 1 se obtuvo una pérdida de paquetes del 21%, un retardo de 11,05 ms y un jitter de 14,35 ms.

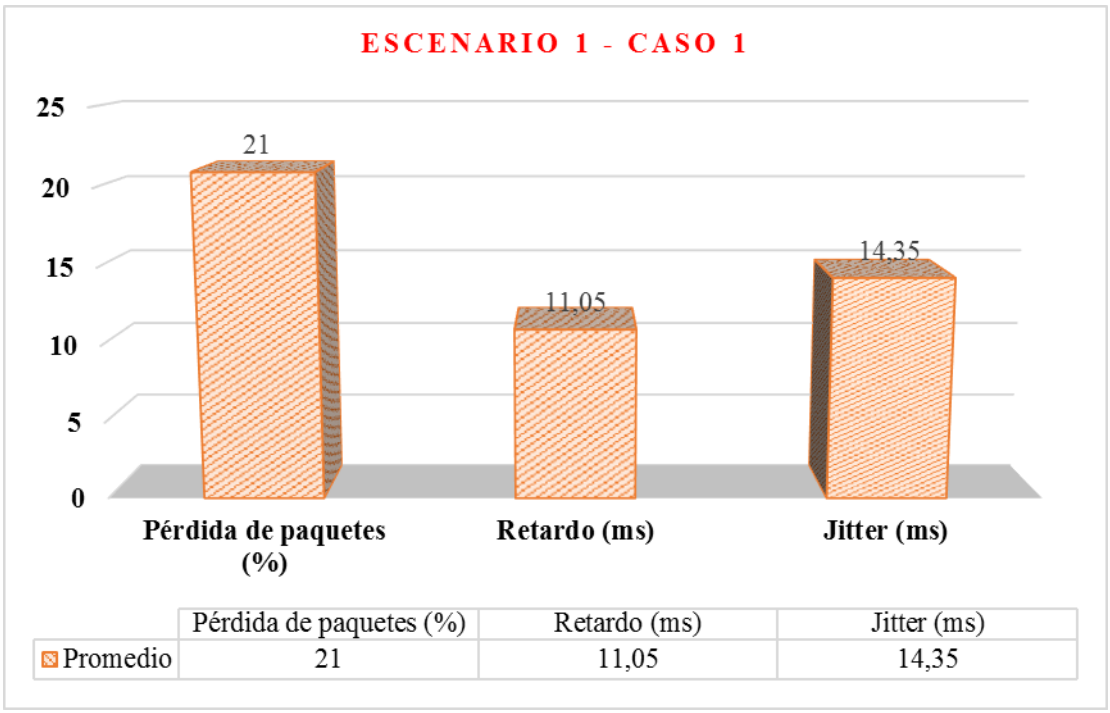


Gráfico 1.3 Análisis del escenario 1 - caso 1

Realizado por Kateryne K. Niama. 2017

Caso 2:

Se analizó los siguientes parámetros: nodos, distancia y tiempo, como se muestra en la tabla 5.3.

Tabla 5.3 Parámetros del escenario 1 - caso 2

Parámetros	Cantidad
Nodos	15
Distancia	30 m
Tiempo	10 s

Realizado por Kateryne K. Niama. 2017

En la figura 4.3 se puede observar el diseño del segundo caso con respecto al primer escenario, algunos nodos no se encuentran en su zona de cobertura debido a la distancia, con lo que se produce un mayor número de pérdida de paquetes.

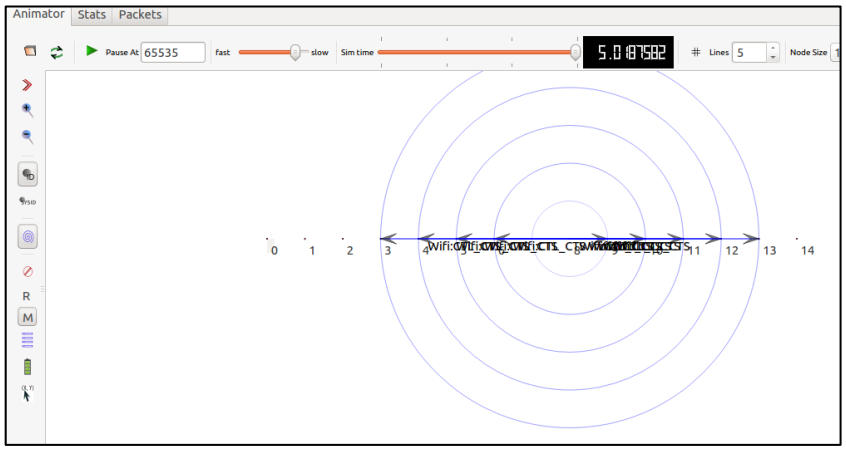


Figura 4.3 Diseño del escenario 1 - caso 2

Realizado por Kateryne K. Niama. 2017

En la figura 5.3 se observa los resultados obtenidos del segundo caso con sus respectivos parámetros.

Figura 5.3 Resultados del escenario 1 - caso 2

Realizado por Kateryne K. Niama. 2017

En la tabla 6.3 se puede observar la tabla de enrutamiento de la primera del escenario 1 – caso 2.

Tabla 6.3 Tabla de enrutamiento de la primera prueba del escenario 1 - caso 2

Dirección origen	Dirección destino	Pérdida de paquete (%)	Retardo (ms)	Jitter (ms)
10.0.0.1	10.0.0.2	38	83	-
10.0.0.1	10.0.0.3	37	6	77
10.0.0.1	10.0.0.4	35	25	19
10.0.0.1	10.0.0.5	40	16	9
10.0.0.1	10.0.0.6	37	5	11
10.0.0.1	10.0.0.8	43	19	14
10.0.0.1	10.0.0.9	33	10	9
10.0.0.1	10.0.0.10	30	25	15
10.0.0.1	10.0.0.11	39	4	21
10.0.0.1	10.0.0.12	37	22	18
10.0.0.1	10.0.0.13	40	31	9
10.0.0.1	10.0.0.14	39	4	27
10.0.0.2	10.0.0.15	41	39	35

Realizado por Kateryne K. Niama. 2017

Se realizó un número de 10 pruebas para sacar un promedio de como en 10 segundos afecta a la red, como se muestra en la tabla 7.3.

Tabla 7.3 Análisis del escenario 1 - caso 2

Número de pruebas	Pérdida de paquetes (%)	Retardo (ms)	Jitter (ms)
1	40	22,33	22,8
2	37	15,403	15,072
3	36	15,063	9,065
4	41	22,525	19,582
5	37	15,403	15,072
6	43	31,186	25,678
7	39	19,773	20
8	35	14,817	16,432
9	38	19,082	21,901
10	40	22,33	22,8
Promedio	**39**	**19,79**	**18,84**

Realizado por Kateryne K. Niama. 2017

Como se puede observar en el gráfico 2.3 tenemos una pérdida de paquetes del 39%, un retardo de 19,79 ms y un jitter de 18,84 ms.

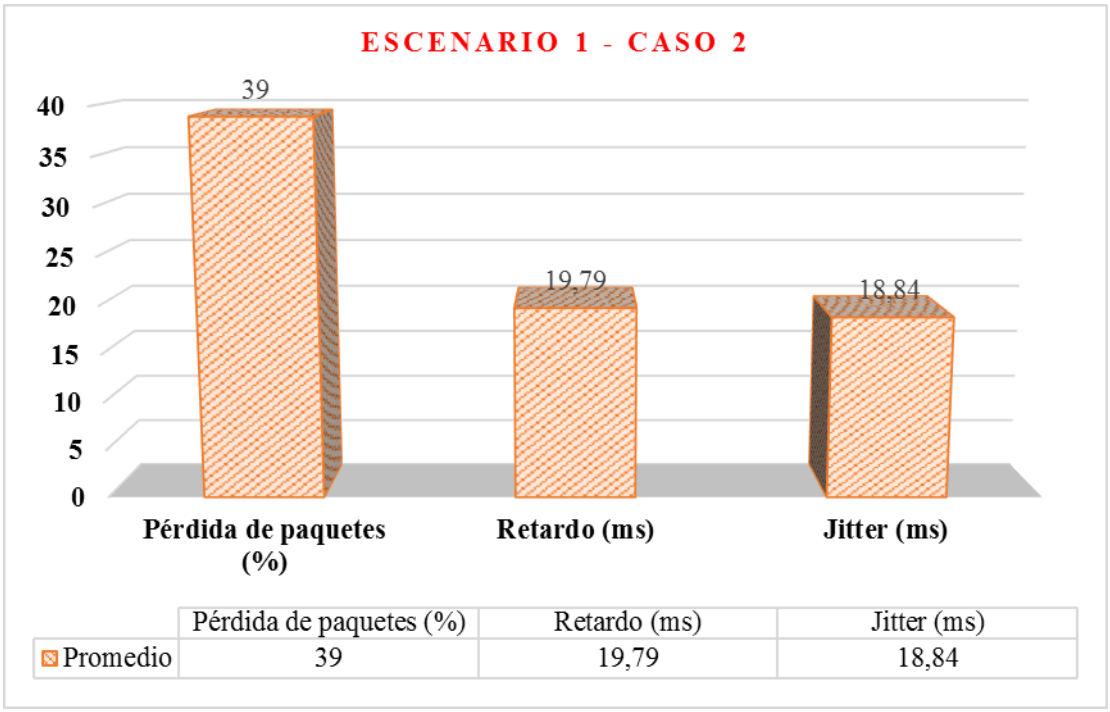


Gráfico 2.3 Análisis del escenario 1 - caso 2

Realizado por Kateryne K. Niama. 2017

Caso 3:

En el tercer y último se analizó los siguientes parámetros: nodos, distancia y tiempo, como se muestra en la tabla 8.3.

Tabla 8.3 Parámetros del escenario 1 - caso 3

Parámetros	Cantidad
Nodos	8
Distancia	20 m
Tiempo	10 s

Realizado por Kateryne K. Niama. 2017

En la figura 6.3 se puede observar el diseño del tercer caso con respecto al primer escenario, donde se puede observar que todos los nodos creados se encuentran en su zona de cobertura.

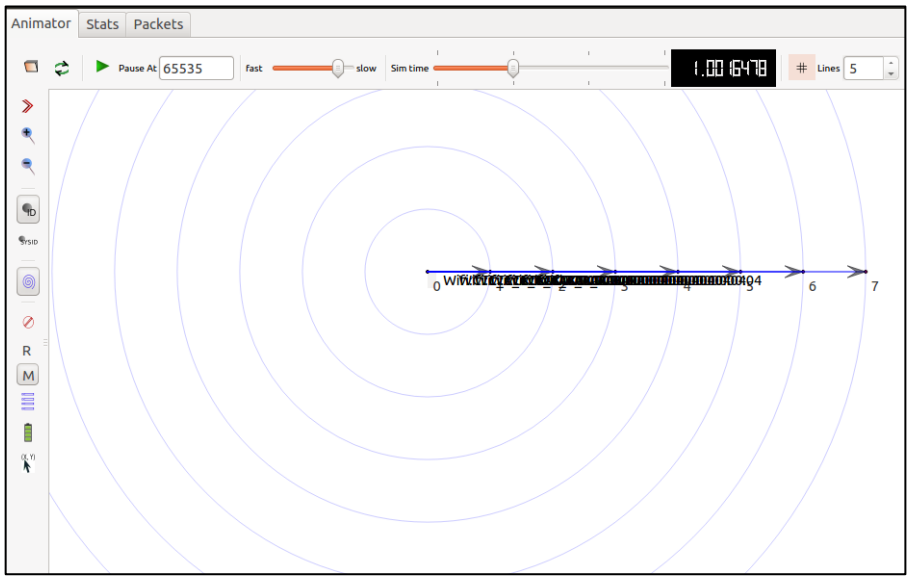


Figura 6.3 Diseño del escenario 1 - caso 3

Realizado por Kateryne K. Niama. 2017

En la figura 7.3 se observa los resultados obtenidos del tercer caso con sus respectivos parámetros.

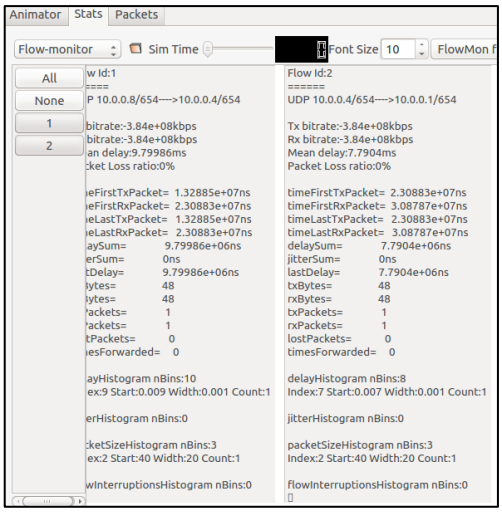


Figura 7.3 Resultado del escenario 1 - caso 3

Realizado por Kateryne K. Niama. 2017

En la tabla 9.3 se puede observar la tabla de enrutamiento de la primera del escenario 1 – caso 3.

Tabla 9.3 Tabla de enrutamiento de la primera prueba del escenario 1 - caso 3

Dirección origen	**Dirección destino**	**Pérdida de paquete (%)**	**Retardo (ms)**	**Jitter (ms)**
10.0.0.1	10.0.0.2	2	50	-
10.0.0.1	10.0.0.3	3	0,97	49,03
10.0.0.1	10.0.0.4	3	1,38	0,41
10.0.0.1	10.0.0.5	1	0,9	0,48
10.0.0.1	10.0.0.6	1	0,76	0,14
10.0.0.1	10.0.0.7	3	1,06	0,3
10.0.0.1	10.0.0.8	2	1	0,06

Realizado por Kateryne K. Niama. 2017

Se realizó un número de 10 pruebas para sacar un promedio de como en 10 segundos afecta a la red, como se muestra en la tabla 10.3.

Tabla 10.3 Análisis del escenario 1 - caso 3

Número de pruebas	Pérdida de paquetes (%)	Retardo (ms)	Jitter (ms)
1	2	5,900	5,440
2	2	5,900	5,440
3	2	5,900	5,440
4	1	5,867	5,373
5	1	5,867	5,373
6	1	5,867	5,373
7	2	5,900	5,440
8	1	5,867	5,373
9	1	5,867	5,373
10	2	5,900	5,440
Promedio	**2**	**5,88**	**5,41**

Realizado por Kateryne K. Niama. 2017

Como se puede observar en el gráfico 3.3 se tiene una pérdida de paquetes del 2%, un retardo de 5,88 ms y un jitter de 5,41 ms., lo que implica a menor distancia con menor número de nodos; mejor será su retardo y jitter.

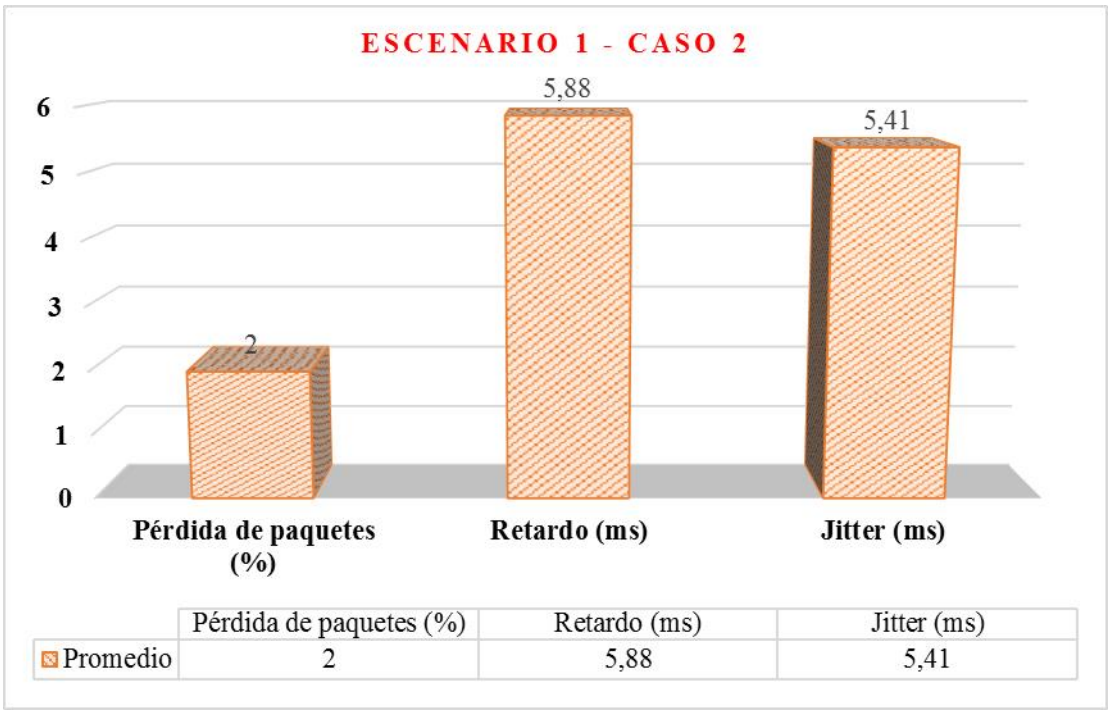
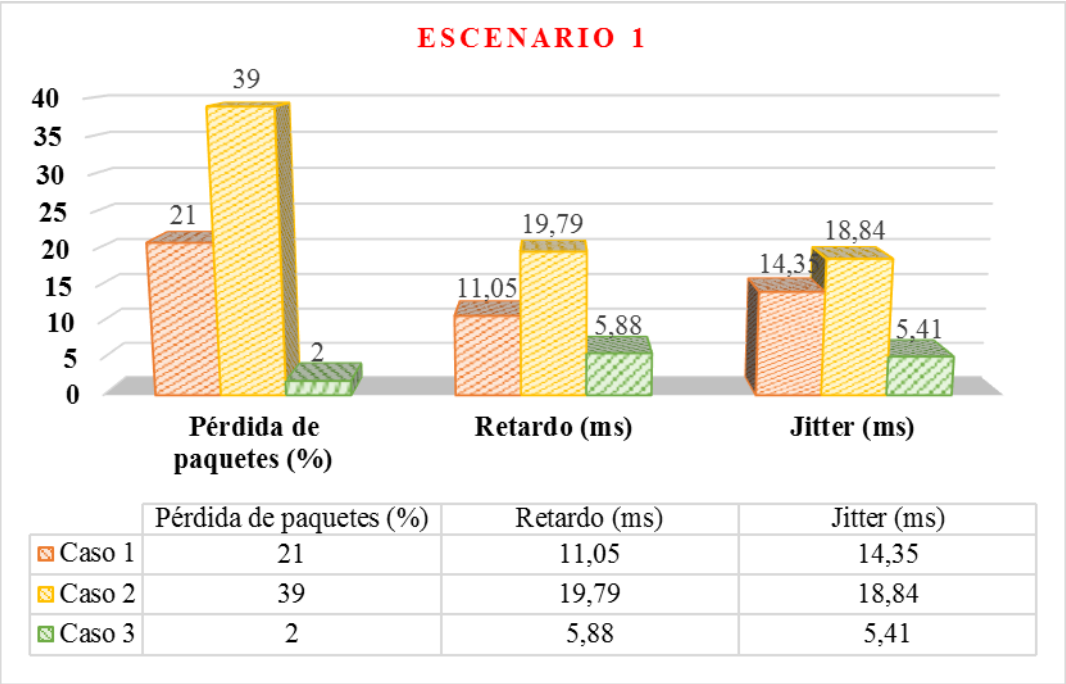


Gráfico 3.3 Análisis del escenario 1 - caso 3

Realizado por Kateryne K. Niama. 2017

Comparación de los 3 casos en el escenario 1:

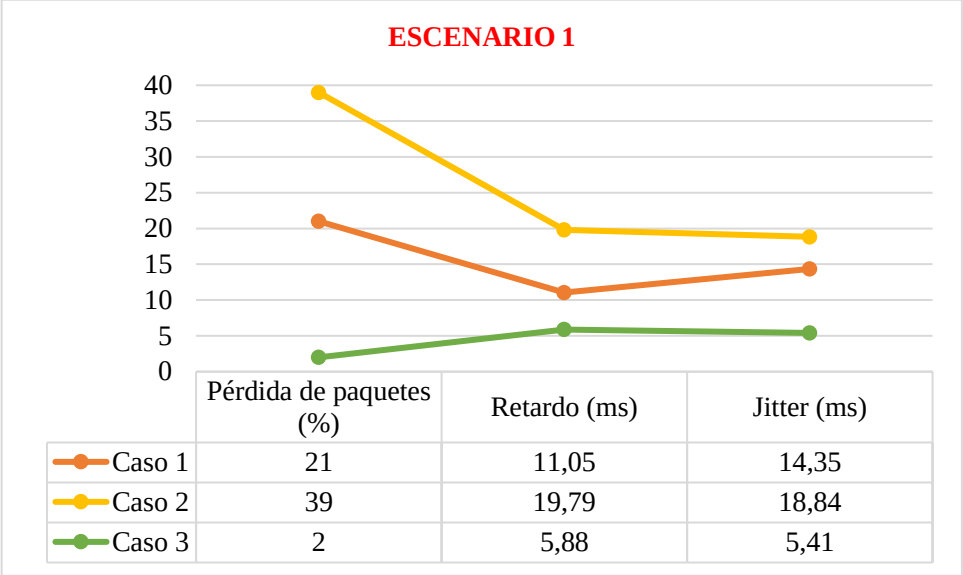


	Pérdida de paquetes (%)	Retardo (ms)	Jitter (ms)
Caso 1	21	11,05	14,35
Caso 2	39	19,79	18,84
Caso 3	2	5,88	5,41

Gráfico 4.3 Comparación del escenario 1 – 3 casos

Realizado por Kateryne K. Niama. 2017

Como se puede observar en el gráfico 4.3 y 5.3, en el caso 3 existe una pérdida de paquetes de 2%, retardo de 5,88 ms y jitter de 5,41, debido que este diseño de red puede ser el más óptimo y estable utilizando el protocolo AODV, haciendo referencia y comparando con la tabla 3.1, de calidad de servicio se puede determinar que cumple con los parámetros para ofrecer una conexión eficaz.



	Pérdida de paquetes (%)	Retardo (ms)	Jitter (ms)
Caso 1	21	11,05	14,35
Caso 2	39	19,79	18,84
Caso 3	2	5,88	5,41

Gráfico 5.3 Comparación de los 3 casos en el escenario 1

Realizado por Kateryne K. Niama. 2017

3.1.2 Simulación del primer escenario con el protocolo DSDV

Para los tres casos se analizó con respecto a los siguientes parámetros: nodos, la distancia y el tiempo.

Caso 1:

Se analizó los siguientes parámetros: nodos, distancia y tiempo, como se muestra en la tabla 11.3

Tabla 11. 3 Parámetros del escenario 2 - caso 1

Parámetros	**Cantidad**
Nodos	10
Distancia	25 m
Tiempo	10 s

Realizado por Kateryne K. Niama. 2017

En la figura 8.3 se puede observar el diseño del primer caso con respecto al segundo escenario, donde se puede observar que todos los nodos creados se encuentran en su zona de cobertura.

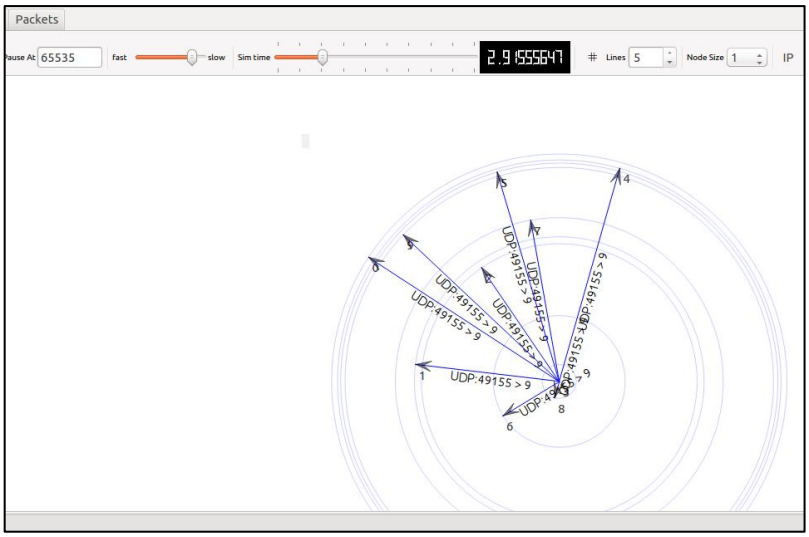


Figura 8. 3 Diseño del escenario 2 - caso 1

Realizado por Kateryne K. Niama. 2017

En la figura 9.3 se observa los resultados obtenidos del primer caso con sus respectivos parámetros

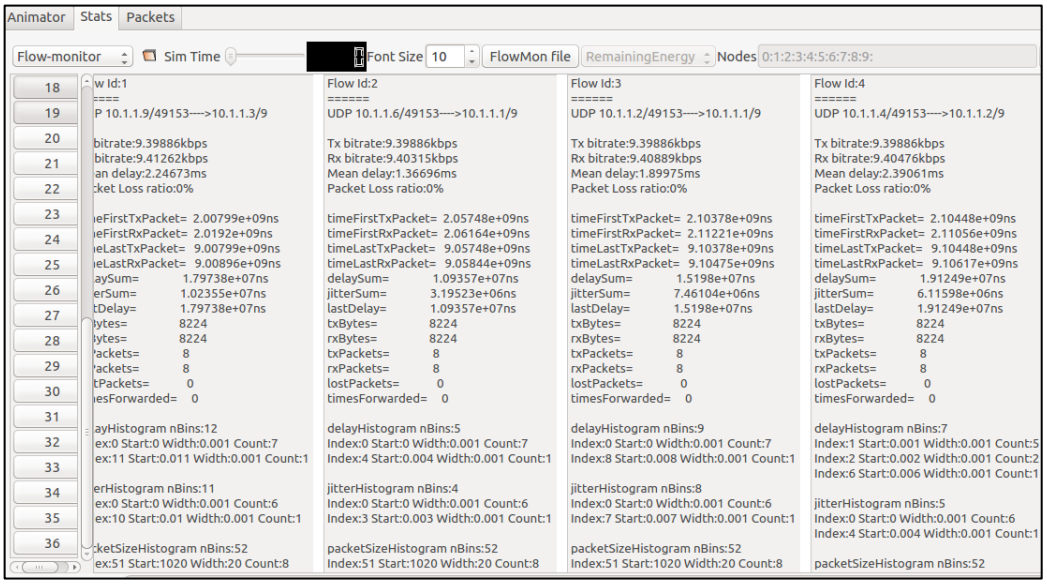


Figura 9. 3 Resultados del escenario 2 - caso 1

Realizado por Kateryne K. Niama. 2017

En la tabla 12.3 se puede observar la tabla de enrutamiento de la primera del escenario 2 – caso 1.

Tabla 12. 3 Tabla de enrutamiento de la primera prueba del escenario 2 - caso 1

Dirección origen	Dirección destino	Pérdida de paquete (%)	Retardo (ms)	Jitter (ms)
10.0.0.1	10.0.0.2	16	35	-
10.0.0.1	10.0.0.3	18	10	25
10.0.0.1	10.0.0.4	17	6	4
10.0.0.1	10.0.0.5	19	11	5
10.0.0.1	10.0.0.7	18	3	8
10.0.0.1	10.0.0.8	17	5	2
10.0.0.1	10.0.0.9	19	7	2
10.0.0.1	10.0.0.10	19	2	5

Realizado por Kateryne K. Niama. 2017

Se realizó un número de 10 pruebas para sacar un promedio de como en 10 segundos afecta a la red, como se muestra en la tabla 13.3.

Tabla 13. 3Análisis del escenario 2 - caso 1

Número de pruebas	Pérdida de paquetes (%)	Retardo (ms)	Jitter (ms)
1	18	7,988	6,240
2	17	5,674	3,440
3	19	11,894	8,041
4	20	14,760	10,760
5	18	7,988	6,240
6	19	11,894	8,041
7	18	7,988	6,240
8	19	11,894	8,041
9	18	7,988	6,240
10	19	11,894	8,041
Promedio	**19**	**10,00**	**7,13**

Realizado por Kateryne K. Niama. 2017

Como se puede observar en el gráfico 6.3, tenemos una pérdida de paquetes del 19%, un retardo de 10 ms y un jitter de 7,13 ms.

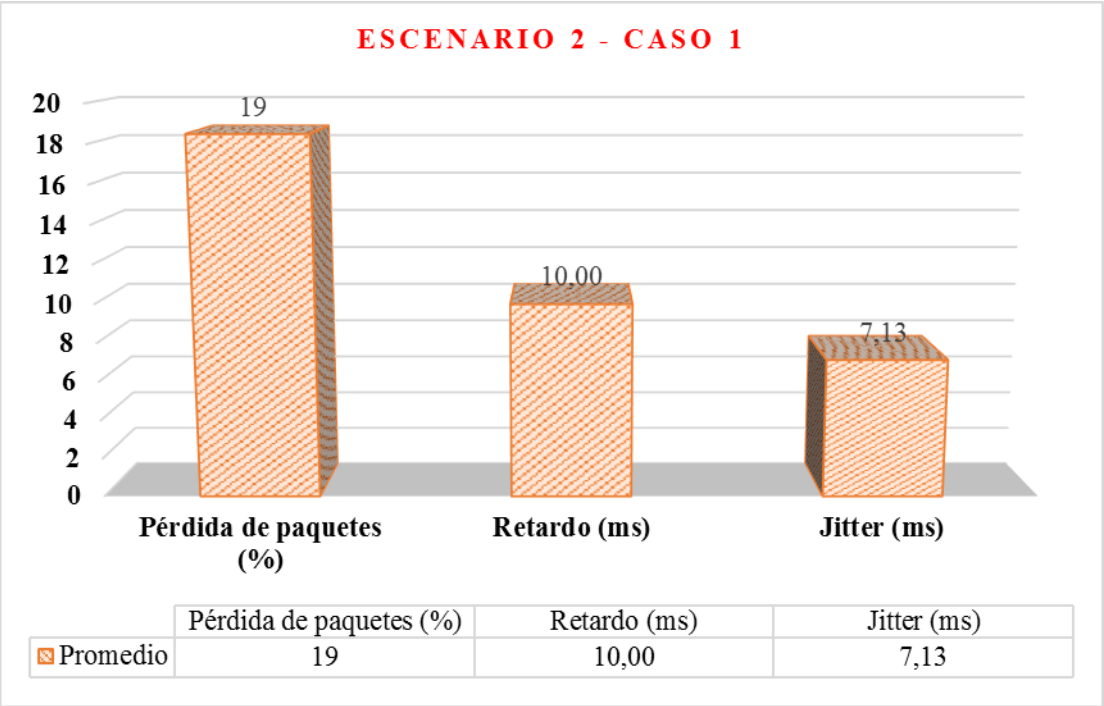


Gráfico 6.3 Análisis del escenario 2 - caso 1

Realizado por Kateryne K. Niama. 2017

Caso 2:

Se analizó los siguientes parámetros: nodos, distancia y tiempo, como se muestra en la tabla 14.3.

Tabla 14. 3 Parámetros del escenario 2 - caso 2

Parámetros	**Cantidad**
Nodos	15
Distancia	30 m
Tiempo	10 s

Realizado por Kateryne K. Niama. 2017

En la figura 10.3 se puede observar el diseño del segundo caso con respecto al segundo escenario, donde se puede observar que todos los nodos creados se encuentran en su zona de cobertura.

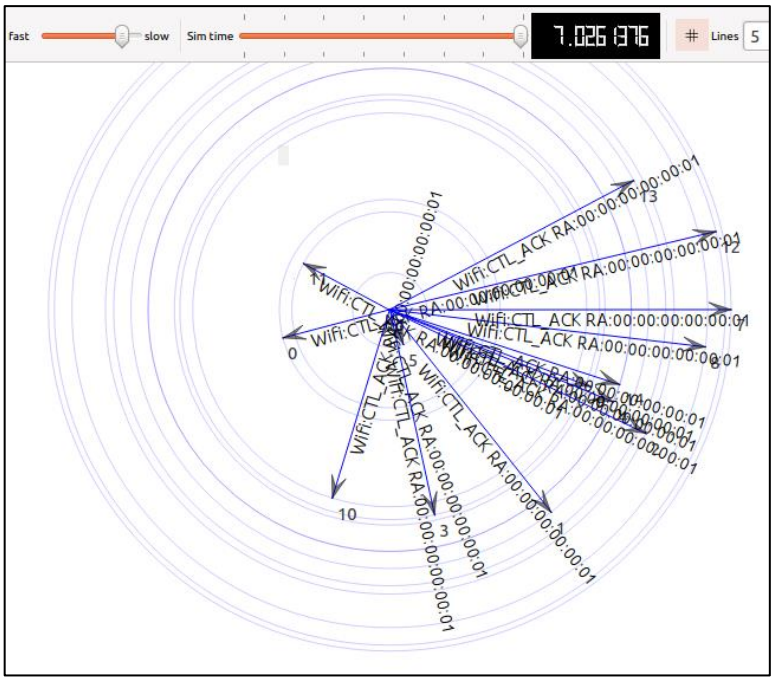


Figura 10. 3 Diseño del escenario 2 - caso 2

Realizado por Kateryne K. Niama. 2017

En la figura 11.3 se observa los resultados obtenidos del segundo caso con sus respectivos parámetros.

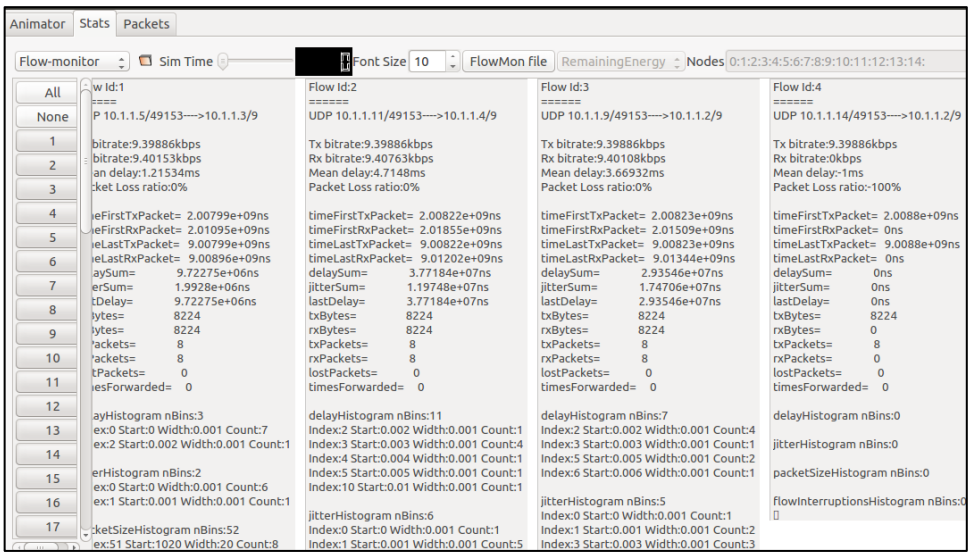


Figura 11. 3 Resultados del escenario 2 - caso 2

Realizado por Kateryne K. Niama. 2017

En la tabla 15.3 se puede observar la tabla de enrutamiento del segundo caso en el escenario 2.

Tabla 15. 3 Tabla de enrutamiento de la primera prueba del escenario 2 - caso 2

Dirección origen	Dirección destino	Pérdida de paquete (%)	Retardo (ms)	Jitter (ms)
10.0.0.1	10.0.0.2	38	116	-
10.0.0.1	10.0.0.3	36	10	106
10.0.0.1	10.0.0.4	38	4	6
10.0.0.1	10.0.0.5	40	15	11
10.0.0.1	10.0.0.7	42	11	4
10.0.0.1	10.0.0.8	45	25	14
10.0.0.1	10.0.0.9	43	13	12
10.0.0.1	10.0.0.10	45	35	22
10.0.0.1	10.0.0.11	44	15	20
10.0.0.1	10.0.0.12	46	13	2
10.0.0.1	10.0.0.13	45	25	12
10.0.0.1	10.0.0.14	45	21	4
10.0.0.1	10.0.0.15	48	23	2

Realizado por Kateryne K. Niama. 2017

Se realizó un número de 10 pruebas para sacar un promedio de como en 10 segundos afecta a la red, como se muestra en la tabla 16.3.

Tabla 16. 3 Análisis del escenario 2 - caso 2

Número de pruebas	Pérdida de paquetes (%)	Retardo (ms)	Jitter (ms)
1	43	25,080	16,540
2	43	25,080	16,540
3	40	22,060	19,000
4	39	21,789	17,480
5	48	27,988	26,240
6	44	27,308	25,890
7	45	30,480	29,080
8	45	30,480	29,080
9	41	22,578	19,488
10	39	21,789	17,480
Promedio	**43**	**25,46**	**21,68**

Realizado por Kateryne K. Niama. 2017

Como se puede observar en el gráfico 7.3, tenemos una pérdida de paquetes del 43%, un retardo de 25,46 ms y un jitter de 21,68 ms.

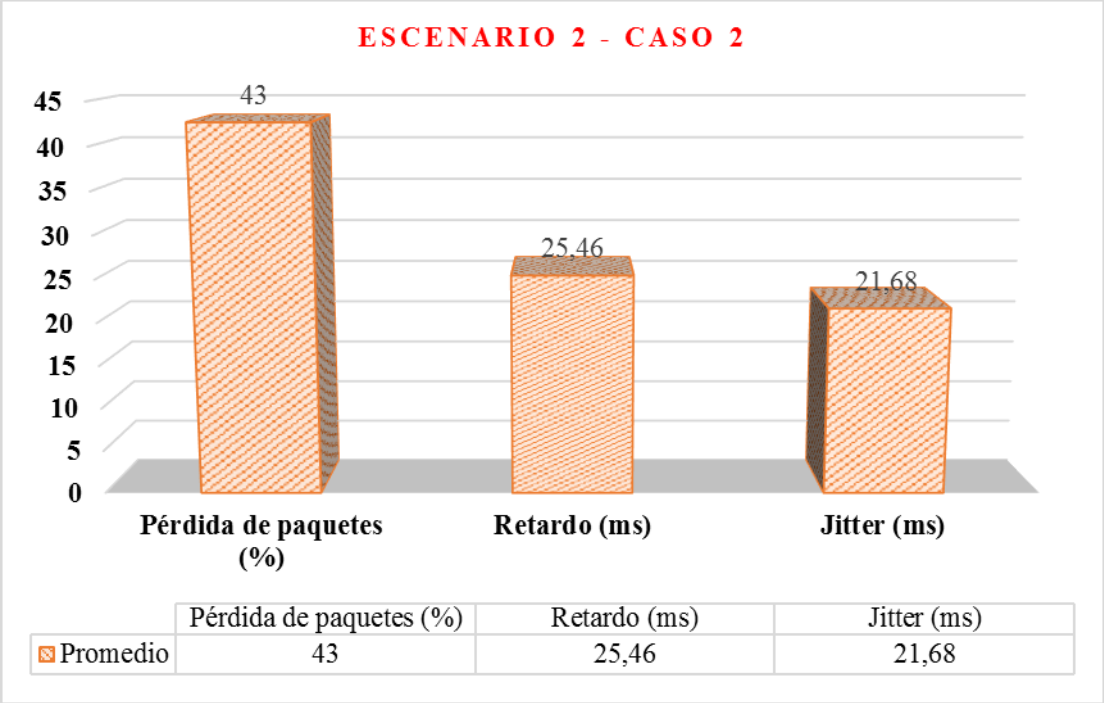


Gráfico 7.3 Análisis del escenario 2 - caso 2

Realizado por Kateryne K. Niama. 2017

Caso 3:

En el tercer y último se analizó los siguientes parámetros: nodos, distancia y tiempo, como se muestra en la tabla 17.3

Tabla 17. 3 Análisis del escenario 2 - caso 3

Parámetros	Cantidad
Nodos	8
Distancia	20 m
Tiempo	10 s

Realizado por Kateryne K. Niama. 2017

En la figura 12.3 se puede observar el diseño del tercer caso con respecto al segundo escenario, donde se puede observar que todos los nodos creados se encuentran en su zona de cobertura.

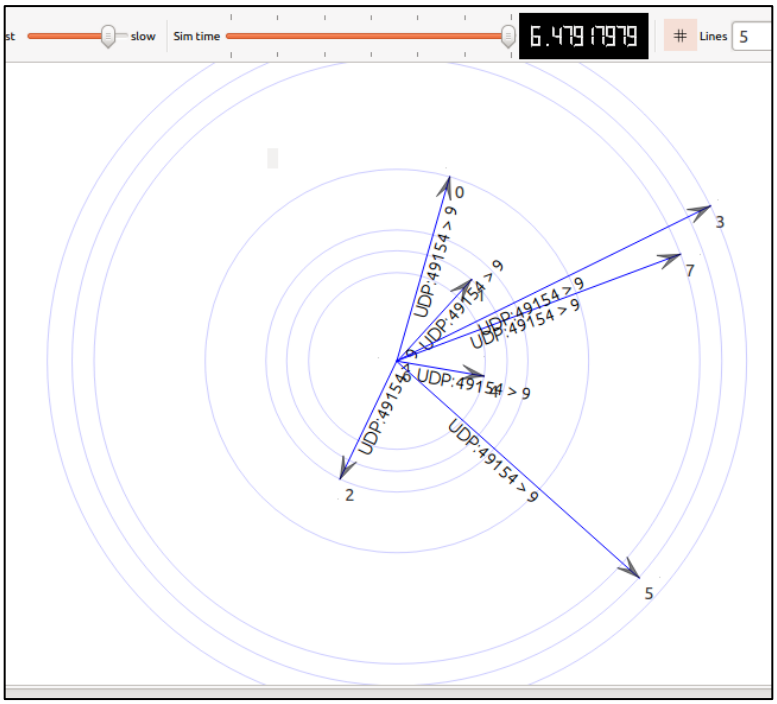


Figura 12. 3 Diseño del escenario 2 - caso 3

Realizado por Kateryne K. Niama. 2017

En la figura 13.3 se observa los resultados obtenidos del tercer caso con sus respectivos parámetros.

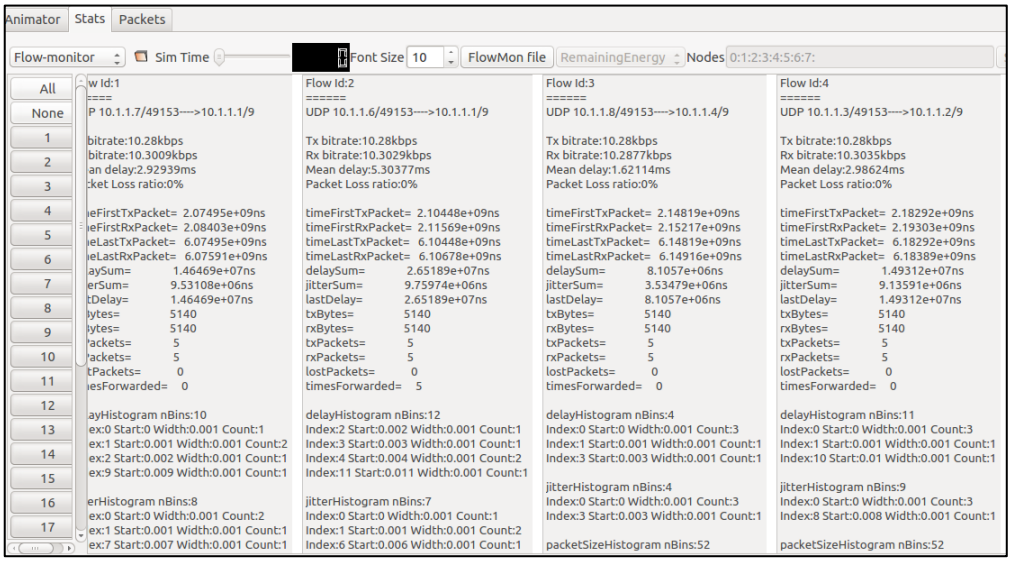


Figura 13.3 Resultados del escenario 2 - caso 3

Realizado por Kateryne K. Niama. 2017

En la tabla 18.3 se puede observar la tabla de enrutamiento del tercer caso en el escenario 2.

Tabla 18. 3 Tabla de enrutamiento de la primera prueba del escenario 2 - caso 2

Dirección origen	Dirección destino	Pérdida de paquete (%)	Retardo (ms)	Jitter (ms)
10.0.0.1	10.0.0.2	2	17	-
10.0.0.1	10.0.0.3	4	2	15
10.0.0.1	10.0.0.4	6	18	16
10.0.0.1	10.0.0.5	5	2	16
10.0.0.1	10.0.0.6	7	5	3
10.0.0.1	10.0.0.7	3	9	4
10.0.0.1	10.0.0.8	6	14	5

Realizado por Kateryne K. Niama. 2017

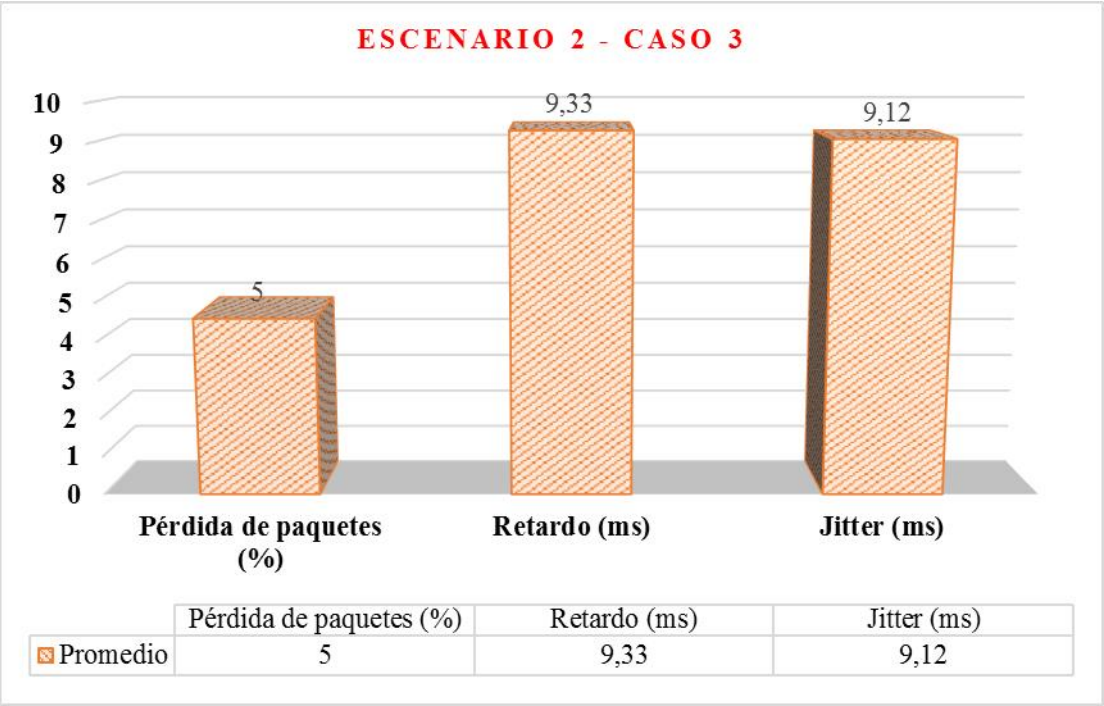
Se realizó un número de 10 pruebas para sacar un promedio de como en 10 segundos afecta a la red, como se muestra en la tabla 19.3.

Tabla 19. 3 Análisis del escenario 2 - caso 2

Número de pruebas	Pérdida de paquetes (%)	Retardo (ms)	Jitter (ms)
1	5	9,57	8,43
2	6	11,71	12,80
3	5	9,57	8,43
4	6	11,71	12,80
5	5	9,57	8,43
6	4	8,87	9,70
7	2	5,80	6,15
8	3	7,50	6,70
9	3	7,50	6,70
10	7	11,54	11,08
Promedio	**5**	**9,33**	**9,12**

Realizado por Kateryne K. Niama. 2017

Como se puede observar en el gráfico 8.3, tenemos una pérdida de paquetes del 5%, un retardo de 9,33 ms y un jitter de 9,12 ms.

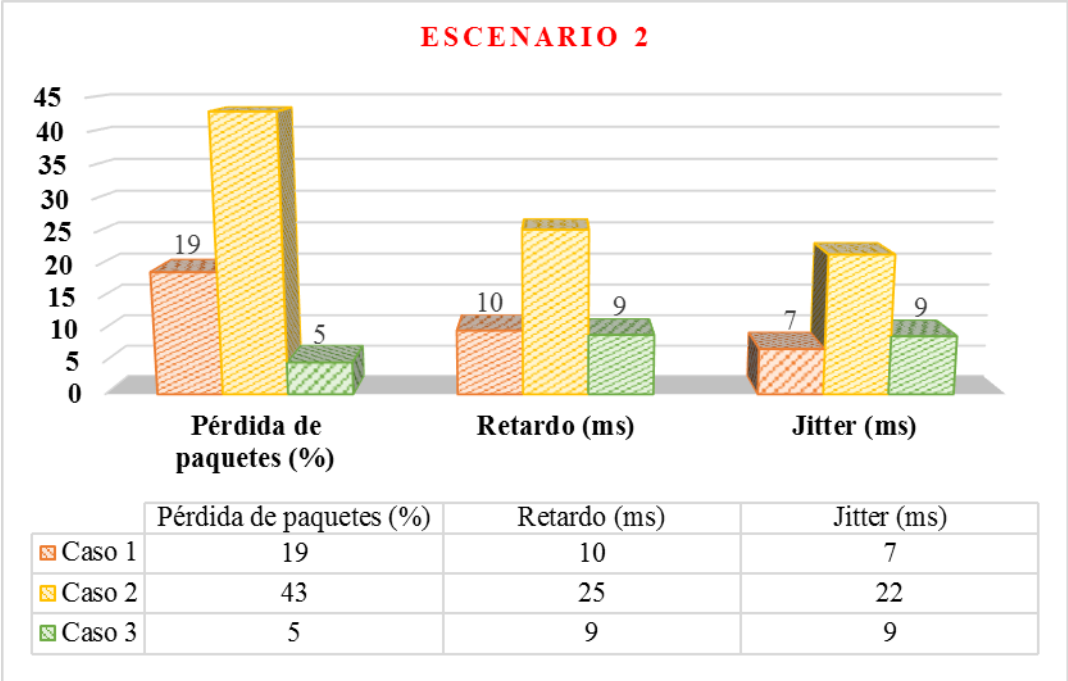


	Pérdida de paquetes (%)	Retardo (ms)	Jitter (ms)
Promedio	5	9,33	9,12

Gráfico 8.3 Análisis del escenario 2 - caso 3

Realizado por Kateryne K. Niama. 2017

Comparación de los 3 casos en el escenario 2:

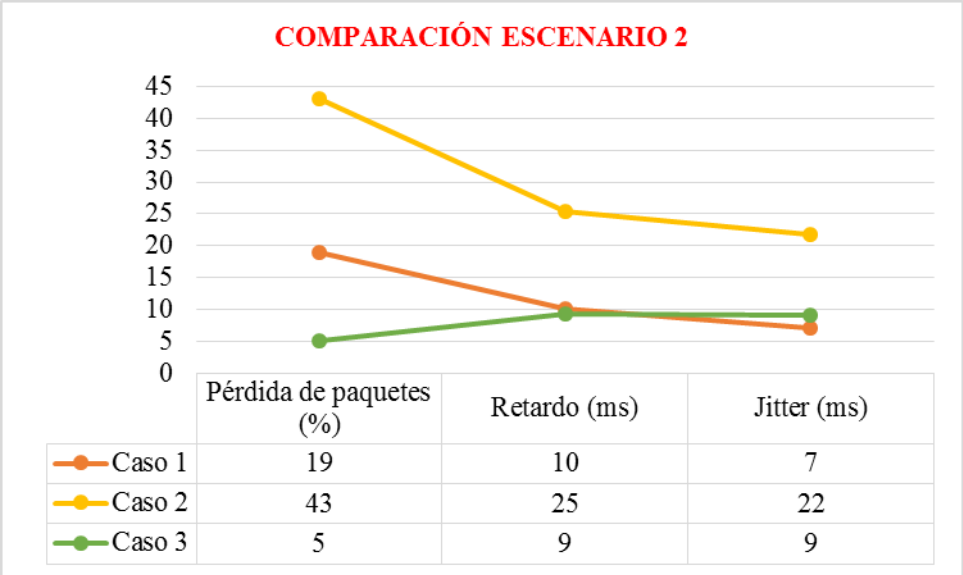


	Pérdida de paquetes (%)	Retardo (ms)	Jitter (ms)
Caso 1	19	10	7
Caso 2	43	25	22
Caso 3	5	9	9

Gráfico 9.3 Comparación del escenario 2 – 3 casos

Realizado por Kateryne K. Niama. 2017

Como se puede observar en el gráfico 9.3 y 10.3, en el caso 3 existe una pérdida de paquetes de 5%, retardo de 9,33 ms y jitter de 9.12, se puede considerar que este diseño de red es el más óptimo y estable utilizando el protocolo DSDV, haciendo referencia y comparando con la tabla 3. 1, de calidad de servicio.

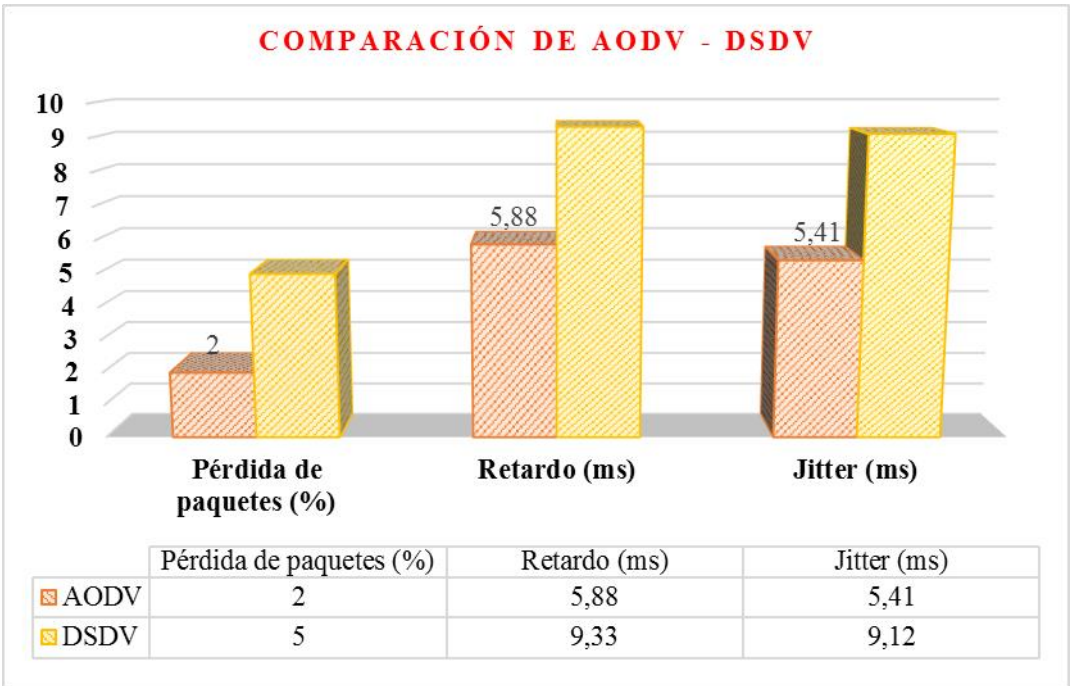


	Pérdida de paquetes (%)	Retardo (ms)	Jitter (ms)
Caso 1	19	10	7
Caso 2	43	25	22
Caso 3	5	9	9

Gráfico 10.3 Comparación de los 3 casos en el escenario 2

Realizado por Kateryne K. Niama. 2017

Comparación de AODV y DSDV



	Pérdida de paquetes (%)	Retardo (ms)	Jitter (ms)
AODV	2	5,88	5,41
DSDV	5	9,33	9,12

Gráfico 11.3 Comparación AODV - DSDV

Realizado por Kateryne K. Niama. 2017

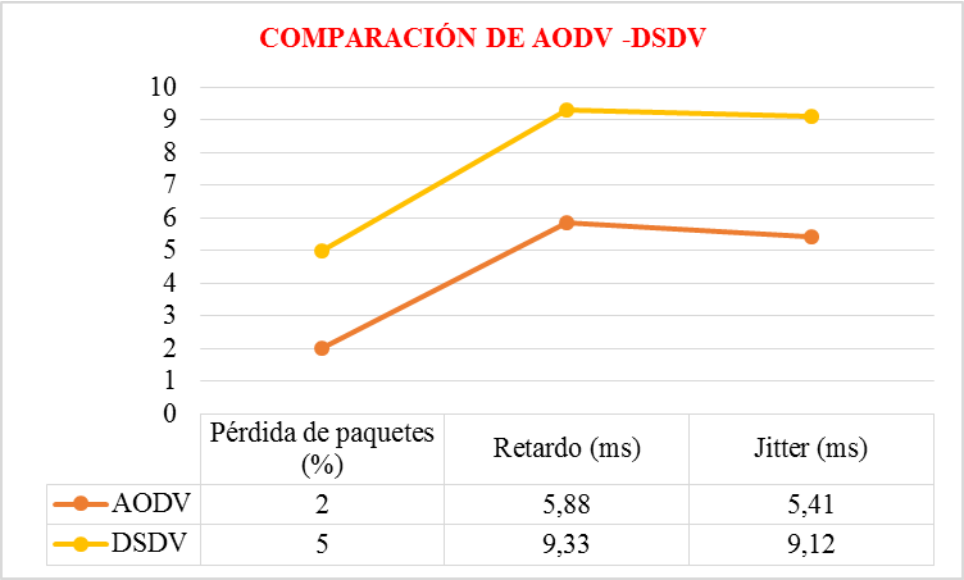


Gráfico 12.3 Comparación AODV - DSDV

Realizado por Kateryne K. Niama. 2017

- **Comparación entre protocolo AODV y DSDV**

Como se puede observar en el gráfico 11.3 y 12.3, utilizando el mismo escenario para el protocolo AODV y DSDV (8 nodos, distancia de 20 m y un tiempo de 10 segundos), se puede analizar y comparar bajo los parámetros propuestos que AODV es un protocolo que presenta mejores resultados y características siendo así más óptimo y estable, ya que tiene menos pérdida de paquetes 2%, jitter de 5,88 ms y un retardo de 5,41 ms, debido que busca la ruta más corta de una manera rápida.

CONCLUSIONES

- Luego de las pruebas realizadas, se pudo determinar que el protocolo AODV es el más adecuado para aplicaciones de video ya que obtuvo una pérdida de paquetes del 2%, y utiliza un algoritmo de Vector Distancia, debido a que escoge su ruta más óptima mediante el número de saltos desde su origen hacia su destino.

- De las plataformas de simulación de redes inalámbricas ad-hoc, OMNET++ obtuvo 65,83% en comparación a NS-3 con 71,67%, por esto se eligió a NS-3 debido a las características técnicas que presenta el software entre la más relevante que es una plataforma que brinda soporte en redes inalámbricas, y la visualización de resultados es más adherido a la realidad.

- La red Ad-hoc trabaja con topología tipo malla, por lo cual presenta varias ventajas entre ellas es que es robusta y confiable, ya que tiene múltiples rutas para él envió del paquete de modo que si una de las conexiones se altera busca otro camino para enviar el paquete hasta llegar al destino.

- Debido a que el protocolo AODV, tiene como característica fundamental buscar la ruta más corta por medio de multi-saltos, se considera un protocolo óptimo, ya que de esta manera tiene un menor tiempo de retardo 5,88 ms, jitter 5,41 ms y una pérdida de paquetes del 2%, y se eficiente en la trasmisión de datos de video.

RECOMENDACIONES

- Se recomienda que para la instalación de las plataformas de simulación, debe cumplir con los requisitos de hardware óptimos como un mejor procesador y suficiente memoria RAM, debido a que utiliza muchos recursos para su instalación y ejecución.

- Para el correcto funcionamiento de las plataformas de simulación OMNET++ y NS-3, se necesita que las versiones estén acorde al sistema operativo, ya que el programa necesita de paquetes extras para su funcionamiento óptimo.

- Se recomienda utilizar el protocolo reactivo AODV, para la correcta transmisión de datos de video, sobre redes MANET, debido a que este protocolo mediante multi-saltos busca la ruta más corta, para realizar él envió de datos, además su tabla de enrutamiento se actualiza automáticamente y cuando es necesario, lo que nos permite un menor tráfico en la red.

- Las redes ad-hoc, debido al tipo de implementación su zona de cobertura es limitada, por lo que se recomienda que la distancia entre nodos no sobrepase este rango, de esta manera poder tener una comunicación eficiente.

GLOSARIO

AODV	Ad-hoc On-Demand Distance Vector
BABEL	Better Approach To Mobile Ad-hoc Networking
BATMAN	Better Approach To Mobile Ad-hoc Networking
CEDAR	Core-Extraction Distributed Ad Hoc Routing
DSDV	Destination-Sequenced Distance-Vector
DSR	Dynamic Source Routing
GLOMO	Global Mobile
HTML	HyperText Markup Language
IEEE	Institute of Electrical and Electronics Engineers
IP	Internet Protocol
ITS	Intelligent Transport Systems
MANET	Mobile Ad-hoc Network
MOS	Mean Opinion Score
MPLS	Multiprotocol Label Switching
NAM	Network Animator
NS	Simulator Network
NSF-CISE	National Science Foundation, Computer & Information Science & Engineering
NTDR	Near Term Digital Radio
OLSR	Optimised Link State Routing
OSI	Open System Interconnection
RAM	Random Access Memory
RERR	Route Errors
RREP	Route Replies
RREQ	Route Request
SURAN	Survivable Radio Network
TCP	Transmission Control Protocol
TORA	Temporally Ordered Routing
UDP	User Datagram Protocol
UHF	Ultra High Frequency
VANET	Vehicular ad-hoc network
WLAN	Wireless Local Area Network
WMAN	Wireless metropolitan area networks

WMN	Wireless Mesh Network
WPAN	Wireless Personal Area Network
WRP	Wireless Routing Protocol
WSN	Redes de sensores
WWAN	Wireless Wide Area Network
ZRP	Zone Routing Protocol

BIBLIOGRAFÍA

[1] Abreu, M. *Evaluación de desempeño de VoIP en redes MANET.* Cuba. (2014). Obtenido de http://www.scielo.org.co/scielo.php?script=sci_arttext&pid=S1692-17982014000100002

[2] Arroyo Ruiz, J. A. *Combinación de técnicas de simulación y virtualización para analizar el comportamiento de aplicaciones reales.* Trabajo de grado, Cantabria. (2015). Recuperado el 12 de 03 de 2017, de https://repositorio.unican.es/xmlui/bitstream/handle/10902/6670/376267.pdf?sequence=1

[3] Batiste, A.. *Sevilla Mesh*. (05 de 03 de 2011). Recuperado el 11 de 03 de 2017, de Red inalámbrica abierta y libre de Sevilla: https://sevillamesh.wordpress.com/

[4] Ekram Hossain, T. I. *Introduction to Network Simulator NS2.* New York. (2012). Second Edition.

[5] García Quiroz, N. *SIMULACIÓN DE TRÁFICO EN REDES INALAMBRICAS MEDIANTE NS2.* Pereira. (2010).

[6] Hidalgo Barranco, F. S. *ESTUDIO DEL COMPORTAMIENTO DE LA CONEXIÓN A INTERNET DESDE REDES MÓVILES AD HOC CON PROTOCOLO AODV6.* Málaga. (2006).

[7] Hurtado, J. P., & Siguenza, E. P. *ANÁLISIS DL USO DE NODOS MÓVILES VIRTUALES PRA PROCESOS DE ENCAMINAMIENTO EN REDES VEHICULARES AD-HOC.* Cuenca. (2013).

[8] Jain, R. *Una encuesta de red Herramientas de simulación: Estado actual y desarrollos futuros.* (24 de 11 de 2008). Recuperado el 12 de 03 de 2017, de http://www.cse.wustl.edu/~jain/cse567-08/ftp/simtools/

[9] Morales Mosquera, A., & Tabares, C. A. *DISEÑO Y SIMULACIÓN DE UNA RED MESH EN EL MUNICIPIO DE SANTA ROSA DE CABAL.* Tesis de grado, Pereira. (2011). Recuperado el 12 de 03 de 2017, de http://ribuc.ucp.edu.co:8080/jspui/bitstream/handle/10785/1490/CDMIST49.pdf?sequence=1

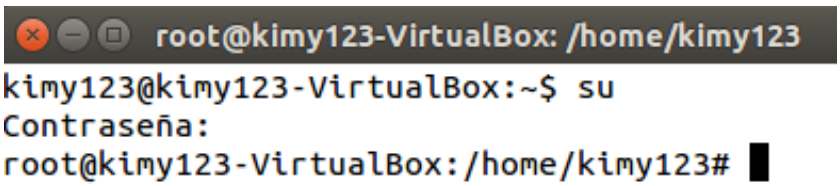
[10] Rodriguez Pineda, G. M. *"Análisis y Simulación de protocolos de enrutamiento adecuados en diferentes escenarios para redes AdHoc, mediante la herramienta Ns-3".* UNIVERSIDAD NACIONAL DE LOJA, Loja. (2015). Obtenido de https://dspace.unl.edu.ec/jspui/bitstream/123456789/11585/1/Rodr%C3%ADguez%20Pineda,%20Gabriela%20Maribel.pdf

[11] Ruiz , F., & Solera, M. *Simulación de Protocolos de Enrutamiento para Redes Móviles Ad-Hoc Mediante la Herramienta de Simulación NS-3.* Loja. (2014).

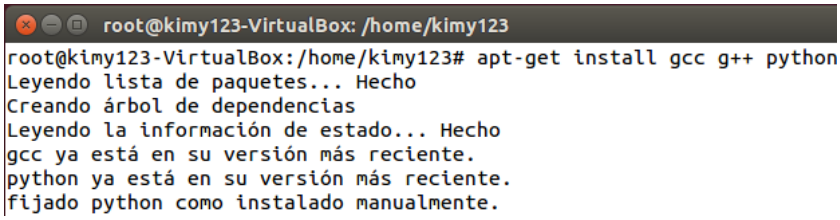
[12] Santos Díaz, N. *MODELO DE SIMULACIÓN PARA REDES DE COMUNICACIÓN DE SISTEMAS DE DETECCIÓN DE FUEGO BASADO EN OMNET++*. Cataluya. (2013). Obtenido de https://upcommons.upc.edu/bitstream/handle/2099.1/19619/ProyectoFinCarrera_NoeliaSantosDiaz.pdf?sequence=4

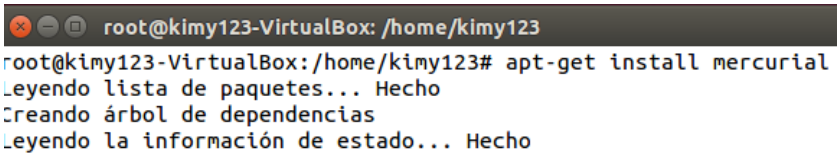
ANEXOS

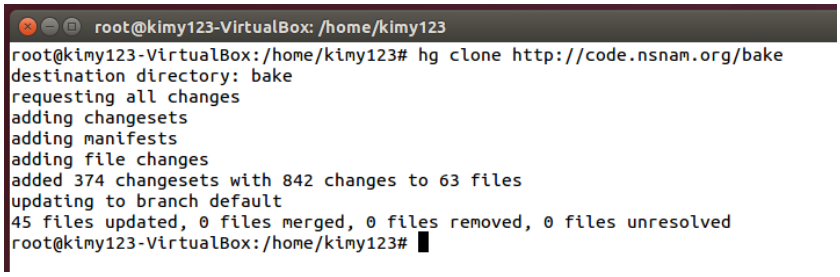
ANEXO A INSTALACIÓN DE NS-3

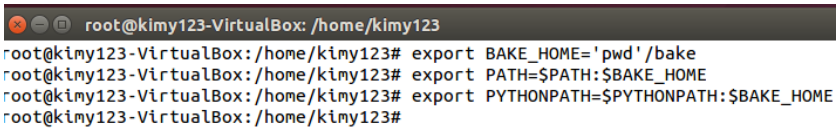


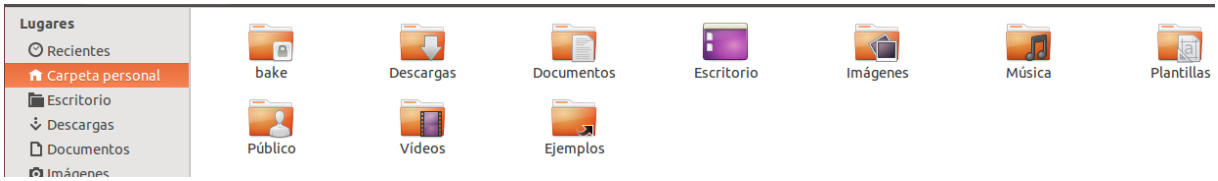
1. Instalación y uso de Bake



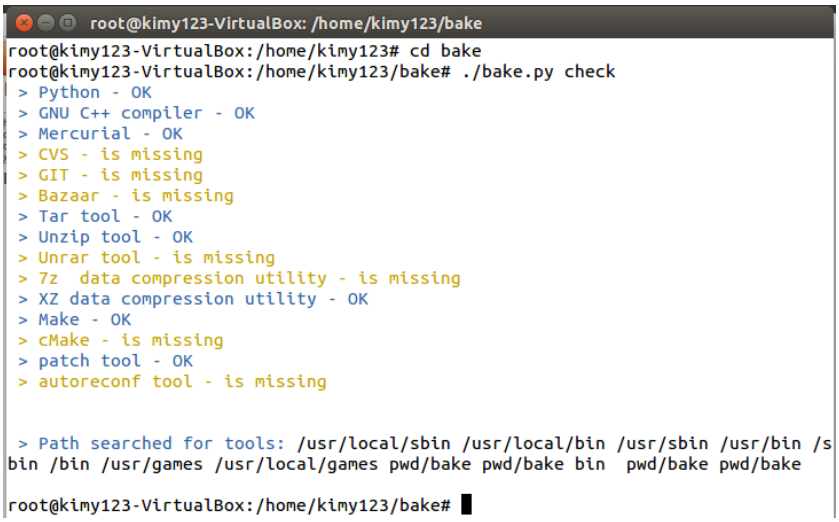


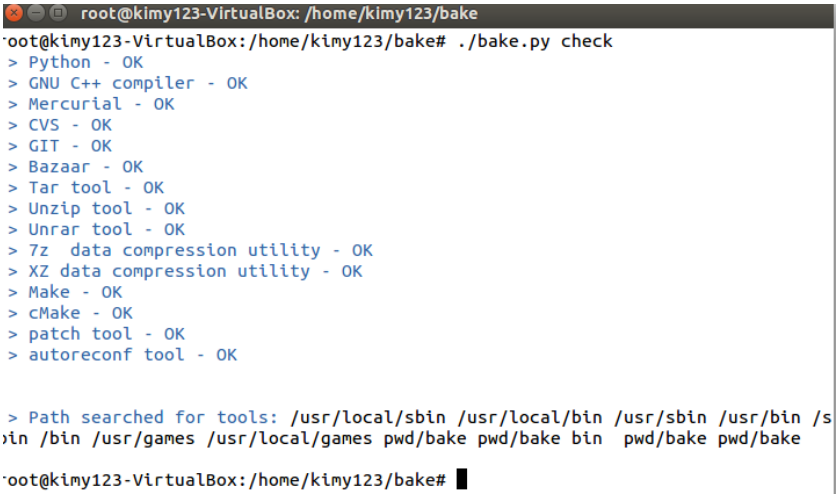






2. Instalación de los paquetes necesarios para NS-3





3.- Descarga de NS-3

```
root@kimy123-VirtualBox: /home/kimy123/bake
root@kimy123-VirtualBox:/home/kimy123/bake# ./bake.py configure -e ns-3.23
root@kimy123-VirtualBox:/home/kimy123/bake#
```

```
root@kimy123-VirtualBox: /home/kimy123/bake

root@kimy123-VirtualBox:/home/kimy123/bake# ./bake.py show
module: qt4 (enabled)
  No dependencies!
```

```
root@kimy123-VirtualBox: /home/kimy123/bake
    pygraphviz (optional:True)
    pygoocanvas (optional:True)
module: netanim-3.106 (enabled)
  depends on:
    qt4 (optional:False)
    g++ (optional:False)
module: ns-3.23 (enabled)
  depends on:
    netanim-3.106 (optional:True)
    pybindgen-0.17.0.886 (optional:True)
    pyviz-prerequisites (optional:True)

-- System Dependencies --
 > g++ - OK
 > pygoocanvas - Missing
   >> The pygoocanvas is not installed, try to install it.
   >> Try: "sudo apt-get install python-pygoocanvas", if you have sudo rights.
 > pygraphviz - Missing
   >> The pygraphviz is not installed, try to install it.
   >> Try: "sudo apt-get install python-pygraphviz", if you have sudo rights.
 > python-dev - Missing
   >> The python-dev is not installed, try to install it.
   >> Try: "sudo apt-get install python-dev", if you have sudo rights.
 > qt4 - Missing
   >> Didn't find:   QT 4, download and install it from http://qt.nokia.com/down
loads/
   >> Try: "sudo apt-get install qt4-dev-tools libqt4-dev", if you have sudo rig
hts.
```

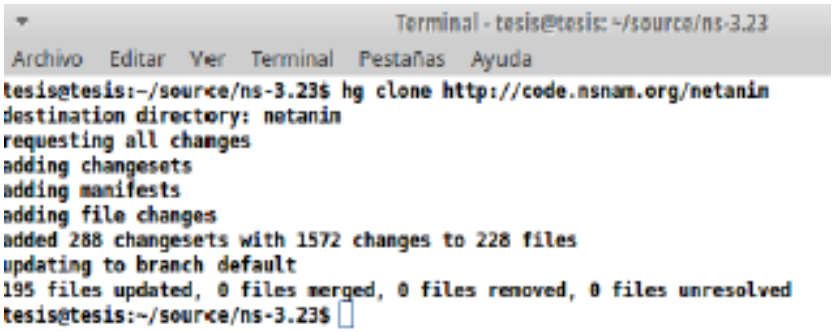
```
-- System Dependencies --
 > g++ - OK
 > pygoocanvas - OK
 > pygraphviz - OK
 > python-dev - OK
 > qt4 - OK
root@kimy123-VirtualBox:/home/kimy123/bake#
```

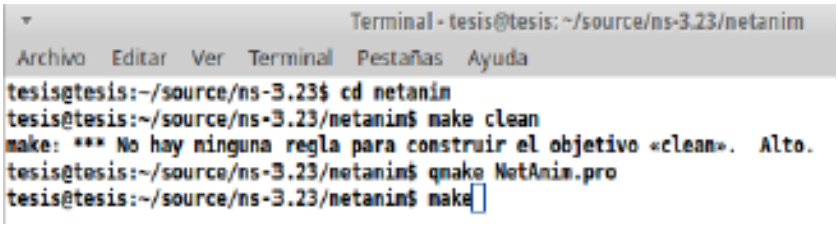
```
Archivo  Editar  Ver  Terminal  Pestañas  Ayuda
tesis@tesis:~$ bake.py deploy
Downloading, building and installing the selected modules and dependencies.
Please, be patient, this may take a while!
 >> Downloading pybindgen-0.17.0.886 - OK
 >> Searching for system dependency g++ - OK
 >> Searching for system dependency qt4 - OK
 >> Searching for system dependency python-dev - OK
 >> Searching for system dependency pygraphviz - OK
 >> Searching for system dependency pygoocanvas - OK
 >> Downloading netanim-3.106 - OK
 >> Downloading ns-3.23 - OK
 >> Building pybindgen-0.17.0.886 - Problem
 > Problem: Optional dependency, module "pybindgen-0.17.0.886" failed
   This may reduce the  functionality of the final build.
   However, bake will continue since "pybindgen-0.17.0.886" is not an essential dependency.
   For more information call bake with -v or -vvv, for full verbose mode.

 >> Building netanim-3.106 - OK
 >> Building ns-3.23 - OK
tesis@tesis:~$
```

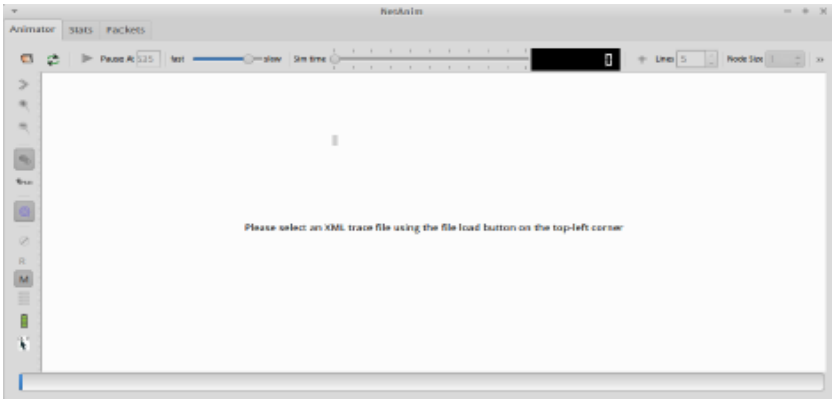
```
PASS: Example src/stats/examples/gnuplot-aggregator-example
PASS: Example src/stats/examples/gnuplot-helper-example
PASS: Example src/uan/examples/uan-rc-example
PASS: Example src/uan/examples/uan-cw-example
PASS: Example src/virtual-net-device/examples/virtual-net-device
PASS: Example src/wave/examples/wave-simple-80211p
PASS: Example src/wave/examples/wave-simple-device
PASS: Example src/wimax/examples/wimax-simple
PASS: Example src/wimax/examples/wimax-ipv4
PASS: Example src/wimax/examples/wimax-multicast
PASS: Example examples/wireless/wifi-ap.py
PASS: Example examples/wireless/mixed-wireless.py
PASS: Example examples/routing/simple-routing-ping6.py
PASS: Example examples/tutorial/first.py
PASS: Example src/bridge/examples/csma-bridge.py
PASS: Example src/core/examples/sample-simulator.py
PASS: Example src/flow-monitor/examples/wifi-olsr-flowmon.py
130 of 130 tests passed (130 passed, 0 skipped, 0 failed, 0 crashed, 0 valgrind errors)

***  Note: ns-3 tests are currently disabled. Enable them by adding
***  "--enable-tests" to ./waf configure or modifying your .ns3rc file.
```

```
Terminal - tesis@tesis: ~/source/ns-3.23
Archivo  Editar  Ver  Terminal  Pestañas  Ayuda
tesis@tesis:~/source/ns-3.23$ ./waf --run hello-simulator
Waf: Entering directory `/home/tesis/source/ns-3.23/build'
Waf: Leaving directory `/home/tesis/source/ns-3.23/build'
'build' finished successfully (2.033s)
Hello Simulator
tesis@tesis:~/source/ns-3.23$
```

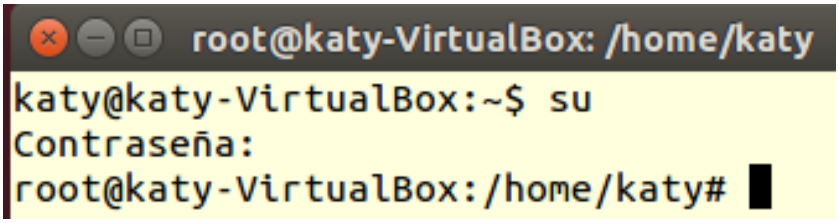
4.- Instalación de NetAnim



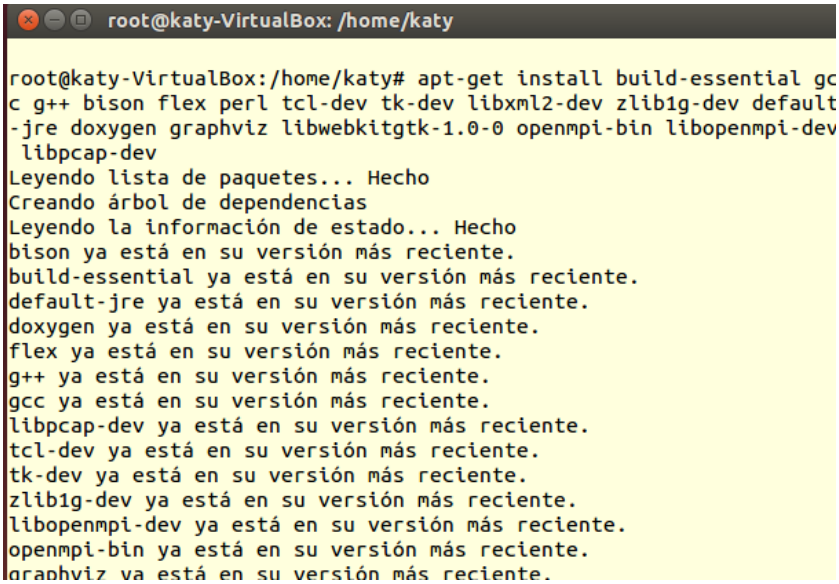


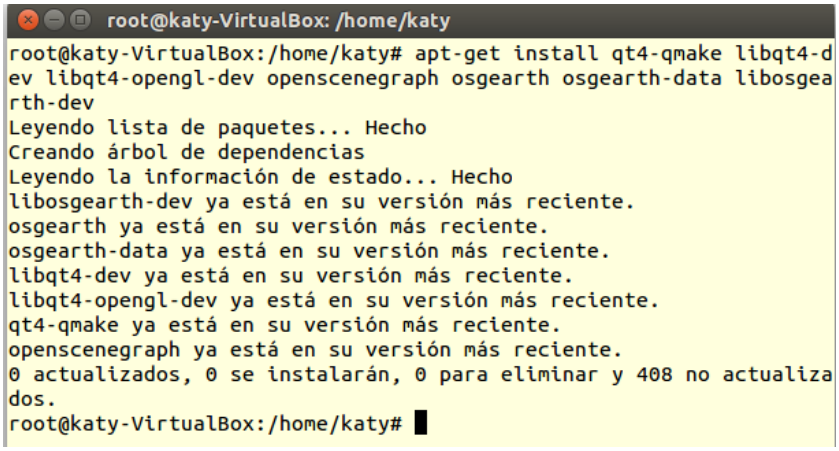


ANEXO B INSTALACIÓN DE OMNET++

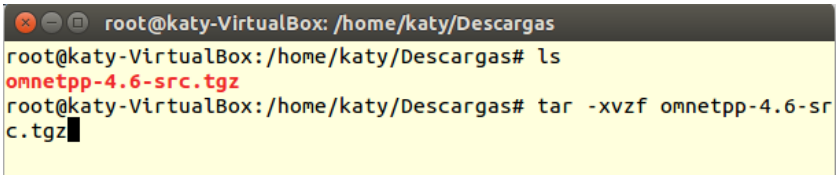


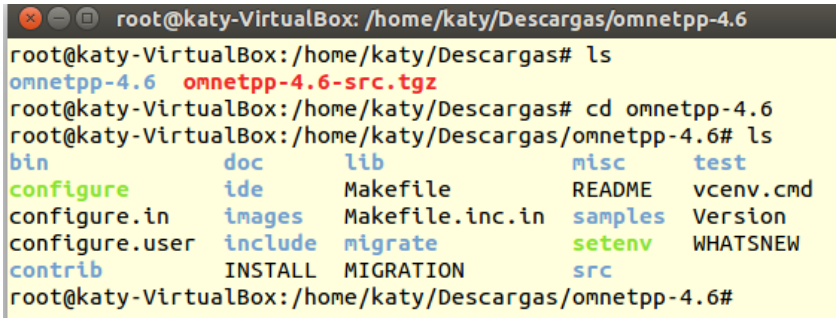
1. Instalación de los paquetes necesarios para OMNET++



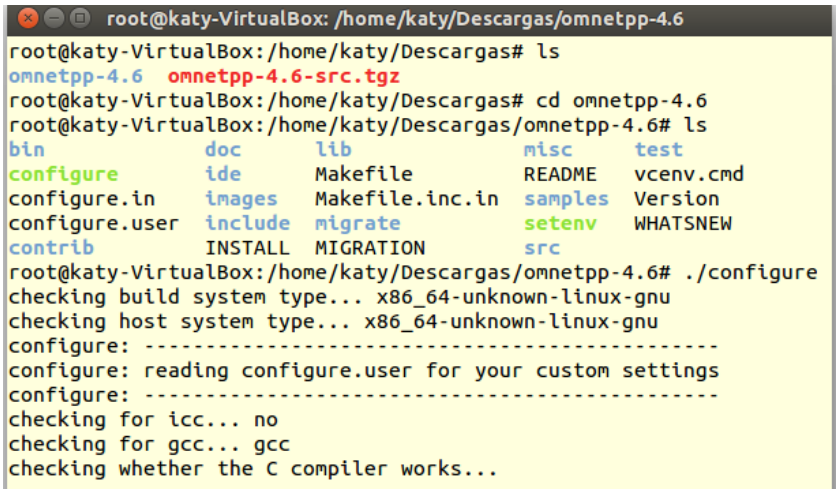


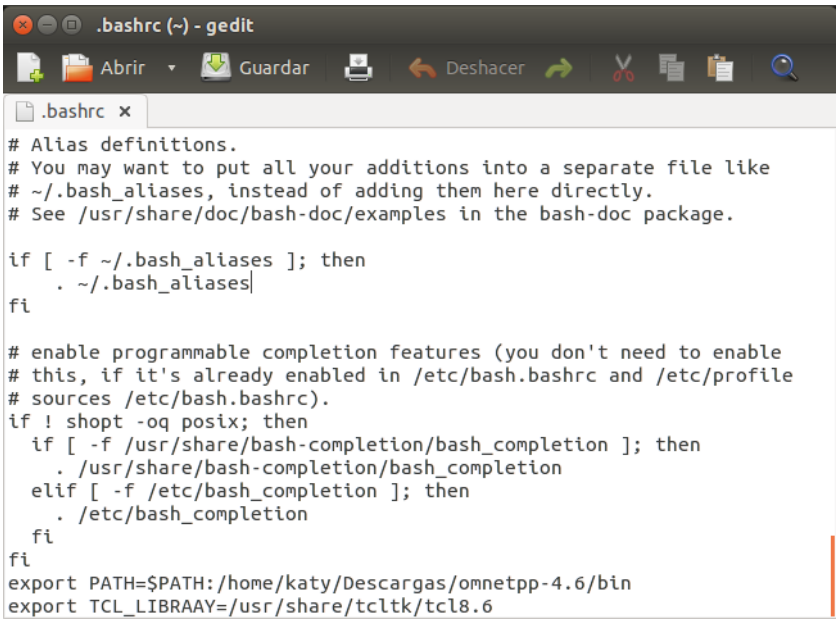
2. Descarga de OMNET++ 4.6

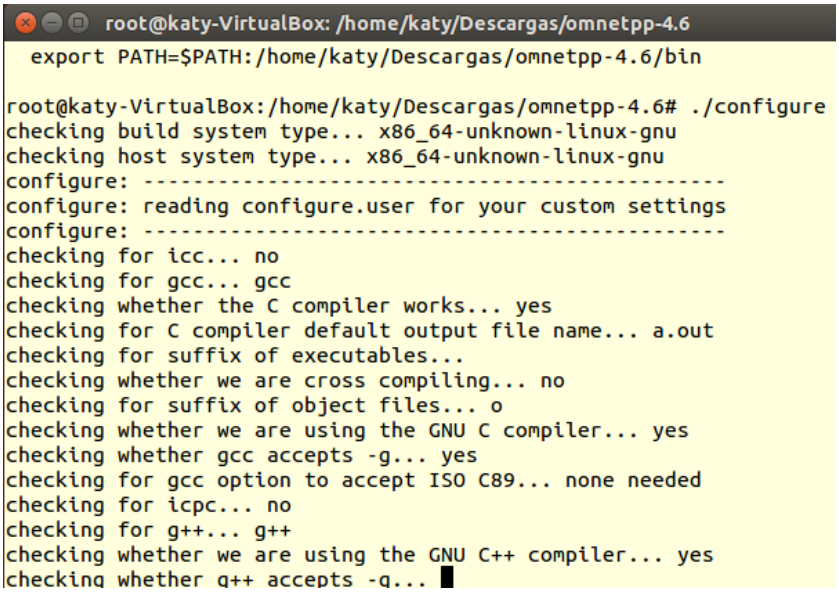


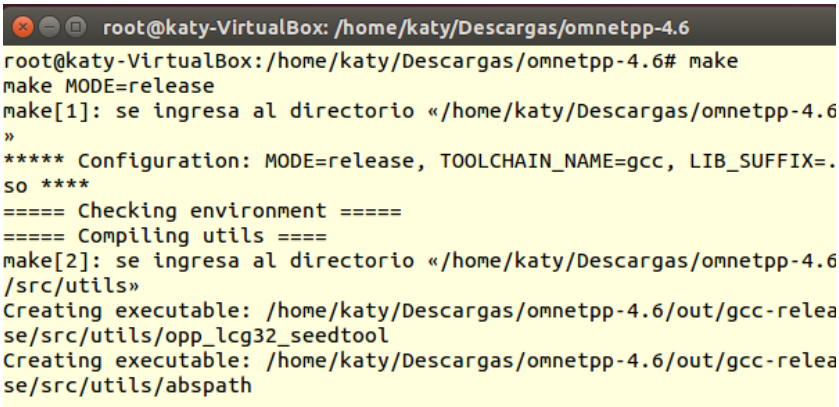


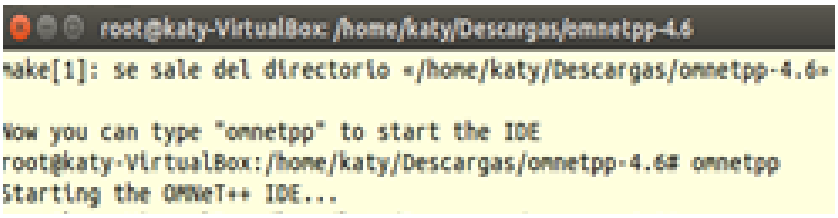
3. Instalación de OMNET++ 4.6



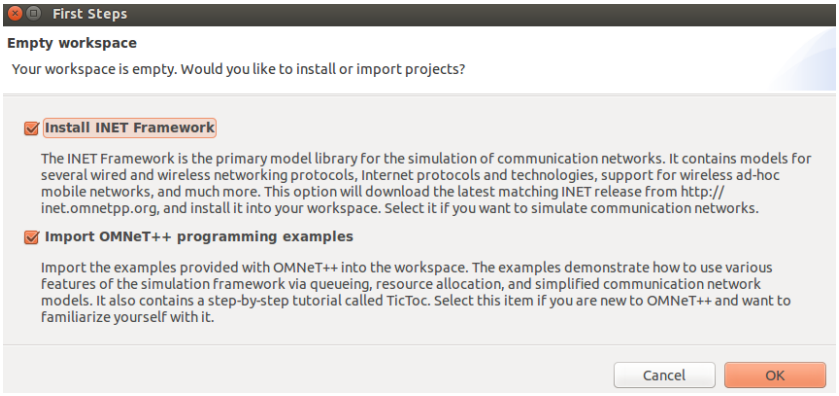

.bashrc (~) - gedit
Abrir
Guardar
Deshacer
.bashrc
# Alias definitions.
# You may want to put all your additions into a separate file like
# ~/.bash_aliases, instead of adding them here directly.
# See /usr/share/doc/bash-doc/examples in the bash-doc package.
if [-f ~/.bash_aliases]; then
. ~/.bash_aliases
fi
# enable programmable completion features (you don't need to enable
# this, if it's already enabled in /etc/bash.bashrc and /etc/profile
# sources /etc/bash.bashrc).
if ! shopt -oq posix; then
if [-f /usr/share/bash-completion/bash_completion]; then
. /usr/share/bash-completion/bash_completion
elif [-f /etc/bash_completion]; then
. /etc/bash_completion
fi
fi
export PATH=$PATH:/home/katy/Descargas/omnetpp-4.6/bin
export TCL_LIBRAAY=/usr/share/tcltk/tcl8.6

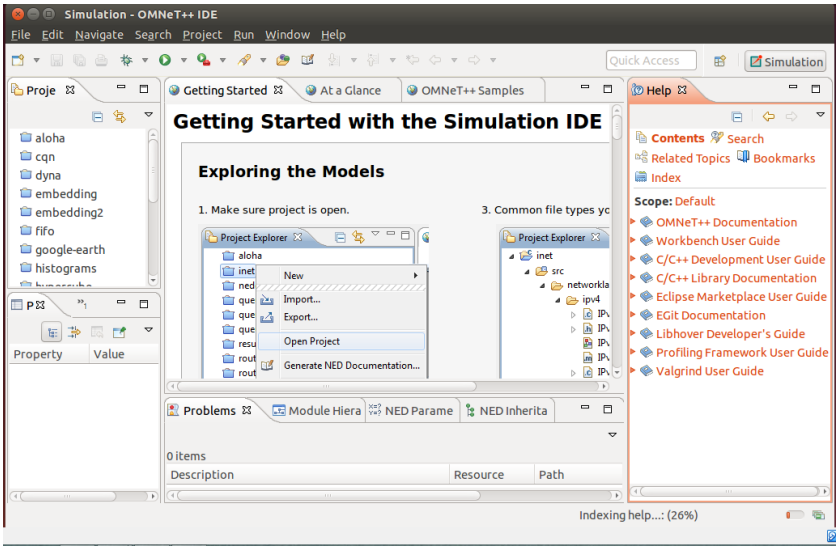

root@katy-VirtualBox: /home/katy/Descargas/omnetpp-4.6
export PATH=$PATH:/home/katy/Descargas/omnetpp-4.6/bin
root@katy-VirtualBox:/home/katy/Descargas/omnetpp-4.6# ./configure
checking build system type... x86_64-unknown-linux-gnu
checking host system type... x86_64-unknown-linux-gnu
configure: --
configure: reading configure.user for your custom settings
configure: --
checking for icc... no
checking for gcc... gcc
checking whether the C compiler works... yes
checking for C compiler default output file name... a.out
checking for suffix of executables...
checking whether we are cross compiling... no
checking for suffix of object files... o
checking whether we are using the GNU C compiler... yes
checking whether gcc accepts -g... yes
checking for gcc option to accept ISO C89... none needed
checking for icpc... no
checking for g++... g++
checking whether we are using the GNU C++ compiler... yes
checking whether g++ accepts -g...


root@katy-VirtualBox: /home/katy/Descargas/omnetpp-4.6
root@katy-VirtualBox:/home/katy/Descargas/omnetpp-4.6# make
make MODE=release
make[1]: se ingresa al directorio «/home/katy/Descargas/omnetpp-4.6
»
***** Configuration: MODE=release, TOOLCHAIN_NAME=gcc, LIB_SUFFIX=.
so ****
===== Checking environment =====
===== Compiling utils ====
make[2]: se ingresa al directorio «/home/katy/Descargas/omnetpp-4.6
/src/utils»
Creating executable: /home/katy/Descargas/omnetpp-4.6/out/gcc-relea
se/src/utils/opp_lcg32_seedtool
Creating executable: /home/katy/Descargas/omnetpp-4.6/out/gcc-relea
se/src/utils/abspath



4. Instalación de INET FRAMEWORK


First Steps
Empty workspace
Your workspace is empty. Would you like to install or import projects?
Install INET Framework
The INET Framework is the primary model library for the simulation of communication networks. It contains models for several wired and wireless networking protocols, Internet protocols and technologies, support for wireless ad-hoc mobile networks, and much more. This option will download the latest matching INET release from http://inet.omnetpp.org, and install it into your workspace. Select it if you want to simulate communication networks.
Import OMNeT++ programming examples
Import the examples provided with OMNeT++ into the workspace. The examples demonstrate how to use various features of the simulation framework via queueing, resource allocation, and simplified communication network models. It also contains a step-by-step tutorial called TicToc. Select this item if you are new to OMNeT++ and want to familiarize yourself with it.
Cancel
OK


Simulation - OMNeT++ IDE
File Edit Navigate Search Project Run Window Help
Quick Access
Simulation
Getting Started
At a Glance
OMNeT++ Samples
Getting Started with the Simulation IDE
Exploring the Models
1. Make sure project is open.
3. Common file types yc
Project Explorer
aloha
inet
New
Import...
Export...
Open Project
Generate NED Documentation...
Help
Contents
Search
Related Topics
Bookmarks
Index
Scope: Default
OMNeT++ Documentation
Workbench User Guide
C/C++ Development User Guide
C/C++ Library Documentation
Eclipse Marketplace User Guide
EGit Documentation
Libhover Developer's Guide
Profiling Framework User Guide
Valgrind User Guide
Problems
Module Hiera
NED Parame
NED Inherita
0 items
Description
Resource
Path
Property
Value
Indexing help...: (26%)

ANEXO C REQUISITOS PARA LA INSTALACIÓN DE LAS PLATAFORMAS

Parámetro	**Hardware**
Equipo	Lapto hp
Memoria RAM	4 GB
Procesador	Intel (R) Core (TM) i5 CPU

Parámetro	**Hardware**
Máquina Virtual	Virtual Box 5.1.10
Memoria RAM	2.00 GB
Disco Duro	50.00 GB

Parámetro	**Software**	**Versión**
Sistema Operativo	XUBUNTU	14.04
Simulador de red	NS-3 OMNET++	3.23 3.4

ANEXO D CÓDIGO DEL PROTOCOLO AODV

```
//MODULOS DE INCLUSION
#include <fstream>
#include "ns3/aodv-module.h"
#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/internet-module.h"
#include "ns3/mobility-module.h"
#include "ns3/point-to-point-module.h"
#include "ns3/wifi-module.h"
#include "ns3/v4ping-helper.h"
#include "ns3/netanim-module.h"
#include "ns3/flow-monitor-module.h"
#include <iostream>
#include <cmath>

//DEFINICION DE NOMBRE
using namespace ns3;

class Aodv
{
public:
Aodv ();
/// Configure script parameters, \return true on successful configuration
bool Configure (int argc, char **argv);
/// Run simulation
void Run ();
/// Report results
void Report (std::ostream & os);

private:
// parameteros
/// Number of nodes
uint32_t nodos;
/// Distance between nodes, meters
double step;
```

```
/// Simulation time, seconds
double totalTime;
/// Write per-device PCAP traces if true
bool pcap;
/// Print routes if true
bool printRoutes;

// network
NodeContainer nodes;
NetDeviceContainer devices;
Ipv4InterfaceContainer interfaces;

private:
void CreateNodes ();
void CreateDevices ();
void InstallInternetStack ();
void InstallApplications ();
};

//FUNCION PRINCIPAL
int main (int argc, char **argv)
{
Aodv test;
if (!test.Configure (argc, argv))
NS_FATAL_ERROR ("Configuration failed. Aborted.");

test.Run ();
test.Report (std::cout);
return 0;
}

//-----------------------------------------------------------------------------
Aodv::Aodv () :
nodos (10), //creacion de los nodos
step (10), // distancia de los nodos
totalTime (25), // paquetes transmitidos
pcap (true),
```

```
printRoutes (true)
{
}

bool
Aodv::Configure (int argc, char **argv)
{
// Enable AODV logs by default. Comment this if too noisy
// LogComponentEnable("AodvRoutingProtocol", LOG_LEVEL_ALL);

SeedManager::SetSeed (12345);
CommandLine cmd;

cmd.AddValue ("pcap", "Write PCAP traces.", pcap);
cmd.AddValue ("printRoutes", "Print routing table dumps.", printRoutes);
cmd.AddValue ("nodos", "Number of nodes.", nodos);
cmd.AddValue ("time", "Simulation time, s.", totalTime);
cmd.AddValue ("step", "Grid step, m", step);

cmd.Parse (argc, argv);
return true;
}

void
Aodv::Run ()
{
//Config::SetDefault ("ns3::WifiRemoteStationManager::RtsCtsThreshold", UintegerValue (1));
// enable rts cts all the time.
CreateNodes ();
CreateDevices ();
InstallInternetStack ();
InstallApplications ();

std::cout << "Starting simulation for " << totalTime << " s ...\n";

Ptr<FlowMonitor> flowMonitor;
FlowMonitorHelper flowHelper;
```

```
flowMonitor = flowHelper.InstallAll ();

//ANIMACION
AnimationInterface anim ("aodv.xml");
anim.SetMobilityPollInterval(Seconds(1));
anim.EnablePacketMetadata(true);

//SIMULADOR
Simulator::Stop (Seconds (totalTime));
Simulator::Run ();

flowMonitor ->SerializeToXmlFile("aodvfow.xml", true, true);

Simulator::Destroy ();
}

void
Aodv::Report (std::ostream &)
{
}

void
Aodv::CreateNodes ()
{
std::cout << "Creating " << (unsigned)nodos << " nodes " << step << " m apart.\n";
nodes.Create (nodos);
// Name nodes
for (uint32_t i = 0; i < nodos; ++i)
{
std::ostringstream os;
os << "node-" << i;
Names::Add (os.str (), nodes.Get (i));
}

//ESTABLECER CANAL WIFI UTILIZANDO HELPER
void
Aodv::CreateDevices ()
```

```
{
NqosWifiMacHelper wifiMac = NqosWifiMacHelper::Default ();
wifiMac.SetType ("ns3::AdhocWifiMac");
YansWifiPhyHelper wifiPhy = YansWifiPhyHelper::Default ();
YansWifiChannelHelper wifiChannel = YansWifiChannelHelper::Default ();
wifiPhy.SetChannel (wifiChannel.Create ());
WifiHelper wifi = WifiHelper::Default ();
wifi.SetRemoteStationManager ("ns3::ConstantRateWifiManager", "DataMode", StringValue
("OfdmRate6Mbps"), "RtsCtsThreshold", UintegerValue (0));
devices = wifi.Install (wifiPhy, wifiMac, nodes);

if (pcap)
{
wifiPhy.EnablePcapAll (std::string ("aodv"));
}
}

void
Aodv::InstallInternetStack ()
{
AodvHelper aodv;
// you can configure AODV attributes here using aodv.Set(name, value)
InternetStackHelper stack;
stack.SetRoutingHelper (aodv); // has effect on the next Install ()
stack.Install (nodes);
Ipv4AddressHelper address;
address.SetBase ("10.0.0.0", "255.0.0.0");
interfaces = address.Assign (devices);

// move node away
Ptr<Node> node = nodes.Get (nodos/2);
Ptr<MobilityModel> mob = node->GetObject<MobilityModel> ();
Simulator::Schedule (Seconds (totalTime/3), &MobilityModel::SetPosition, mob, Vector (1e5,
1e5, 1e5));
}
```

ANEXO E CÓDIGO DEL PROTCOLO DSDV

```
#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/applications-module.h"
#include "ns3/mobility-module.h"
#include "ns3/point-to-point-module.h"
#include "ns3/config-store-module.h"
#include "ns3/wifi-module.h"
#include "ns3/internet-module.h"
#include "ns3/dsdv-helper.h"
#include "ns3/flow-monitor-module.h"
#include "ns3/netanim-module.h"
#include <iostream>
#include <cmath>

using namespace ns3;

NS_LOG_COMPONENT_DEFINE ("Dsdv2");

class Dsdv2
{
public:
Dsdv2 ();
void CaseRun (uint32_t nodos,
uint32_t nSinks,
double totalTime,
std::string rate,
std::string phyMode,
uint32_t nodeSpeed,
uint32_t periodicUpdateInterval,
uint32_t settlingTime,
double dataStart,
bool printRoutes,
std::string CSVfileName);

private:
```

```
//network
NodeContainer nodes;
NetDeviceContainer devices;
Ipv4InterfaceContainer interfaces;

private:
void CreateNodes ();
void CreateDevices (std::string tr_name);
void InstallInternetStack (std::string tr_name);
void InstallApplications ();
void SetupMobility ();
void ReceivePacket (Ptr <Socket> );
Ptr <Socket> SetupPacketReceive (Ipv4Address, Ptr <Node> );
void CheckThroughput ();

};

int main (int argc, char **argv)
{
Dsdv2 test;
uint32_t nodos = 10; //creacion de los nodos
uint32_t nSinks = 4;
double totalTime = 25.0;
std::string rate ("8kbps");
std::string phyMode ("DsssRate11Mbps");
uint32_t nodeSpeed = 10; // en m/s //velocidad del nodo
std::string appl = "all";
uint32_t periodicUpdateInterval = 15;
uint32_t settlingTime = 6;
double dataStart = 1.0;
bool printRoutingTable = true;
std::string CSVfileName = "Dsdv2.csv";


CommandLine cmd;
cmd.AddValue ("nodos", "Number of wifi nodes[Default:30]", nodos);
cmd.AddValue ("nSinks", "Number of wifi sink nodes[Default:10]", nSinks);
```

```
cmd.AddValue ("totalTime", "Total Simulation time[Default:100]", totalTime);
cmd.AddValue ("phyMode", "Wifi Phy mode[Default:DsssRate11Mbps]", phyMode);
cmd.AddValue ("rate", "CBR traffic rate[Default:8kbps]", rate);
cmd.AddValue ("nodeSpeed", "Node speed in RandomWayPoint model[Default:10]",
nodeSpeed);
cmd.AddValue ("periodicUpdateInterval", "Periodic Interval Time[Default=15]",
periodicUpdateInterval);
cmd.AddValue ("settlingTime", "Settling Time before sending out an update for changed
metric[Default=6]", settlingTime);
cmd.AddValue ("dataStart", "Time at which nodes start to transmit data[Default=50.0]",
dataStart);
cmd.AddValue ("printRoutingTable", "print routing table for nodes[Default:1]",
printRoutingTable);
cmd.AddValue ("CSVfileName", "The name of the CSV output file name[Default:Dsdv2.csv]",
CSVfileName);
cmd.Parse (argc, argv);

std::ofstream out (CSVfileName.c_str ());
out << "SimulationSecond," <<
"ReceiveRate," <<
"PacketsReceived," <<
"NumberOfSinks," <<
std::endl;
out.close ();

SeedManager::SetSeed (12345);

Config::SetDefault ("ns3::OnOffApplication::PacketSize", StringValue ("1000"));
Config::SetDefault ("ns3::OnOffApplication::DataRate", StringValue (rate));
Config::SetDefault ("ns3::WifiRemoteStationManager::NonUnicastMode", StringValue
(phyMode));
Config::SetDefault ("ns3::WifiRemoteStationManager::RtsCtsThreshold", StringValue
("2000"));

test = Dsdv2 ();
test.CaseRun (nodos, nSinks, totalTime, rate, phyMode, nodeSpeed, periodicUpdateInterval,
settlingTime, dataStart, printRoutingTable, CSVfileName);
```

```
return 0;
}

Dsdv2::Dsdv2 ()
: bytesTotal (0),
packetsReceived (0)
{
}

void
Dsdv2::ReceivePacket (Ptr <Socket> socket)
{
NS_LOG_UNCOND (Simulator::Now ().GetSeconds () << " Received one packet!");
Ptr <Packet> packet;
while ((packet = socket->Recv ()))
{
bytesTotal += packet->GetSize ();
packetsReceived += 1;
}
}

void
Dsdv2::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 << std::endl;

out.close ();
packetsReceived = 0;
Simulator::Schedule (Seconds (1.0), &Dsdv2::CheckThroughput, this);
}
```

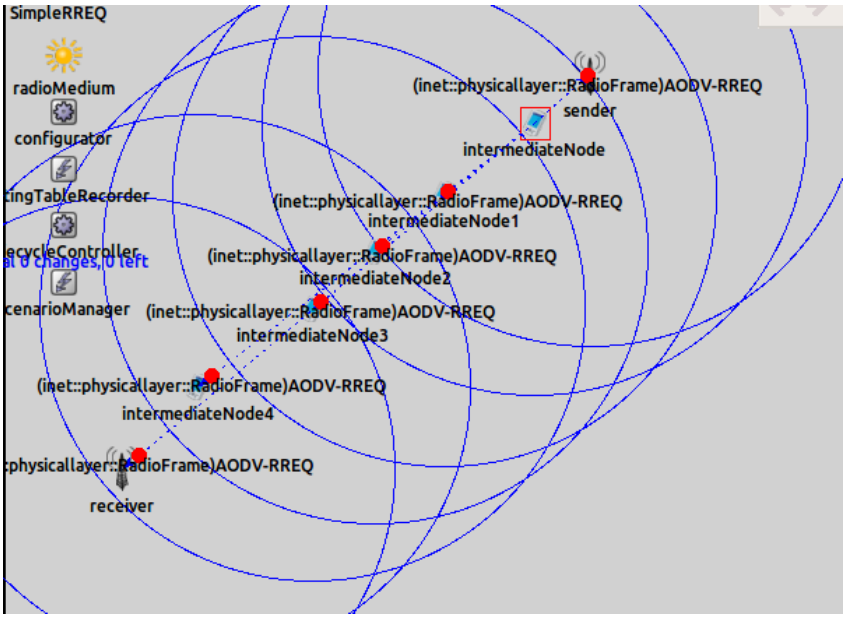
```
Ptr <Socket>
Dsdv2::SetupPacketReceive (Ipv4Address addr, Ptr <Node> node)
{

TypeId tid = TypeId::LookupByName ("ns3::UdpSocketFactory");
Ptr <Socket> sink = Socket::CreateSocket (node, tid);
InetSocketAddress local = InetSocketAddress (addr, port);
sink->Bind (local);
sink->SetRecvCallback (MakeCallback ( &Dsdv2::ReceivePacket, this));

return sink;
}

void
Dsdv2::CaseRun (uint32_t nodos, uint32_t nSinks, double totalTime, std::string rate,
std::string phyMode, uint32_t nodeSpeed, uint32_t periodicUpdateInterval, uint32_t
settlingTime,
double dataStart, bool printRoutes, std::string CSVfileName)
{
std::stringstream ss;
ss << m_nodos;
std::string t_nodes = ss.str ();

std::stringstream ss3;
ss3 << m_totalTime;
std::string sTotalTime = ss3.str ();

std::string tr_name = "Dsdv2" + t_nodes + "Nodes_" + sTotalTime + "SimTime";
std::cout << "Trace file generated is " << tr_name << ".tr\n";

CreateNodes ();
CreateDevices (tr_name);
SetupMobility ();
InstallInternetStack (tr_name);
InstallApplications ();
```

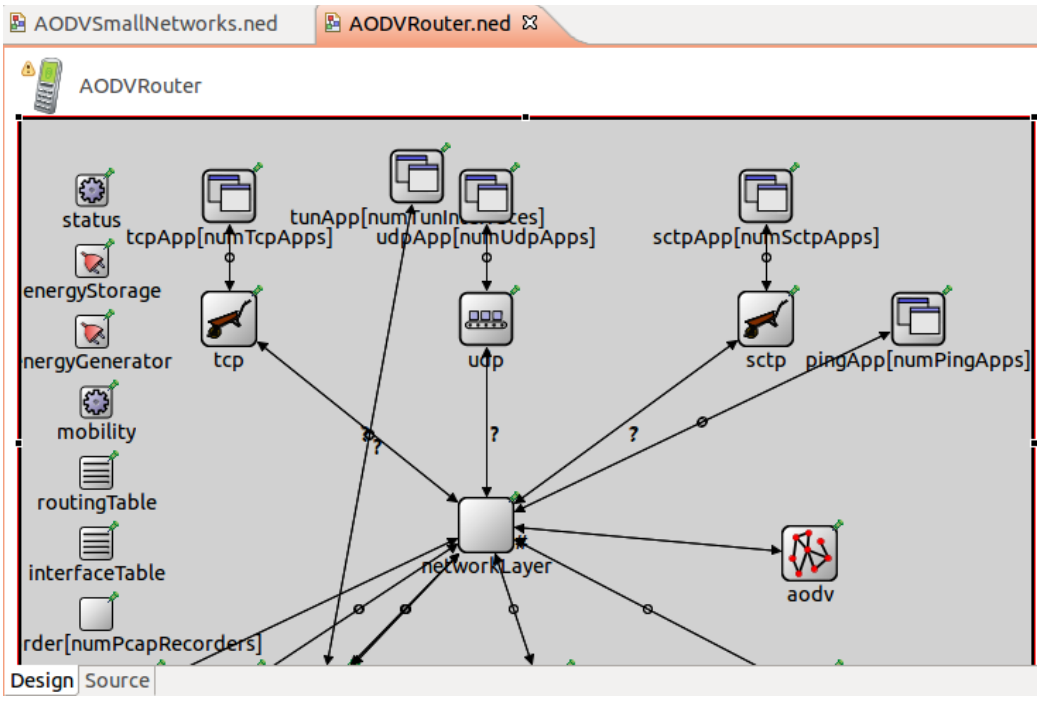
```
std::cout << "\nStarting simulation for " << m_totalTime << " s ...\n";

CheckThroughput ();

Ptr<FlowMonitor> flowMonitor;
FlowMonitorHelper flowHelper;
flowMonitor = flowHelper.InstallAll ();

AnimationInterface anim("dsdv2.xml");
anim.SetMobilityPollInterval(Seconds(1));
anim.EnablePacketMetadata(true);

Simulator::Stop (Seconds (m_totalTime));
Simulator::Run ();
flowMonitor ->SerializeToXmlFile("dsdvfow.xml", true, true);
Simulator::Destroy ();
}

void
Dsdv2::CreateNodes ()
{
std::cout << "Creating " << (unsigned) m_nodos << " nodes.\n";
nodes.Create (m_nodos);
NS_ASSERT_MSG (m_nodos > m_nSinks, "Sinks must be less or equal to the number of
nodes in network");
}

void
Dsdv2::SetupMobility ()
{
MobilityHelper mobility;
ObjectFactory pos;
pos.SetTypeId ("ns3::RandomRectanglePositionAllocator");
pos.Set ("X", StringValue ("ns3::UniformRandomVariable[Min=0.0|Max=1000.0]"));
pos.Set ("Y", StringValue ("ns3::UniformRandomVariable[Min=0.0|Max=1000.0]"));

std::ostringstream speedConstantRandomVariableStream;
```

```
speedConstantRandomVariableStream << "ns3::ConstantRandomVariable[Constant="
<< m_nodeSpeed
<< "]";

Ptr <PositionAllocator> taPositionAlloc = pos.Create ()->GetObject <PositionAllocator> ();
mobility.SetMobilityModel ("ns3::RandomWaypointMobilityModel", "Speed", StringValue
(speedConstantRandomVariableStream.str ()),
"Pause", StringValue ("ns3::ConstantRandomVariable[Constant=2.0]"), "PositionAllocator",
PointerValue (taPositionAlloc));
mobility.SetPositionAllocator (taPositionAlloc);
mobility.Install (nodes);
}

void
Dsdv2::CreateDevices (std::string tr_name)
{
NqosWifiMacHelper wifiMac = NqosWifiMacHelper::Default ();
wifiMac.SetType ("ns3::AdhocWifiMac");
YansWifiPhyHelper wifiPhy = YansWifiPhyHelper::Default ();
YansWifiChannelHelper wifiChannel;
wifiChannel.SetPropagationDelay ("ns3::ConstantSpeedPropagationDelayModel");
wifiChannel.AddPropagationLoss ("ns3::FriisPropagationLossModel");
wifiPhy.SetChannel (wifiChannel.Create ());
WifiHelper wifi;
wifi.SetStandard (WIFI_PHY_STANDARD_80211b);
wifi.SetRemoteStationManager ("ns3::ConstantRateWifiManager", "DataMode", StringValue
(m_phyMode), "ControlMode",
StringValue (m_phyMode));
devices = wifi.Install (wifiPhy, wifiMac, nodes);

AsciiTraceHelper ascii;
wifiPhy.EnableAsciiAll (ascii.CreateFileStream (tr_name + ".tr"));
wifiPhy.EnablePcapAll (tr_name);
}

void
Dsdv2::InstallInternetStack (std::string tr_name)
```

```
{
DsdvHelper dsdv;
dsdv.Set ("PeriodicUpdateInterval", TimeValue (Seconds (m_periodicUpdateInterval)));
dsdv.Set ("SettlingTime", TimeValue (Seconds (m_settlingTime)));
InternetStackHelper stack;
stack.SetRoutingHelper (dsdv); // has effect on the next Install ()
stack.Install (nodes);
Ipv4AddressHelper address;
address.SetBase ("10.1.1.0", "255.255.255.0");
interfaces = address.Assign (devices);
if (m_printRoutes)
{
Ptr<OutputStreamWrapper> routingStream = Create<OutputStreamWrapper> ((tr_name +
".routes"), std::ios::out);
dsdv.PrintRoutingTableAllAt (Seconds (m_periodicUpdateInterval), routingStream);
}
}

void
Dsdv2::InstallApplications ()
{
for (uint32_t i = 0; i <= m_nSinks - 1; i++ )
{
Ptr<Node> node = NodeList::GetNode (i);
Ipv4Address nodeAddress = node->GetObject<Ipv4> ()->GetAddress (1, 0).GetLocal ();
Ptr<Socket> sink = SetupPacketReceive (nodeAddress, node);
}
for (uint32_t clientNode = 0; clientNode <= m_nodos - 1; clientNode++ )
{
for (uint32_t j = 0; j <= m_nSinks - 1; j++ )
{
OnOffHelper onoff1 ("ns3::UdpSocketFactory", Address (InetSocketAddress
(interfaces.GetAddress (j), port)));
onoff1.SetAttribute ("OnTime", StringValue ("ns3::ConstantRandomVariable[Constant=1.0]"));
onoff1.SetAttribute ("OffTime", StringValue
("ns3::ConstantRandomVariable[Constant=0.0]"));
```

ANEXO F PROGRAMACIÓN DEL PROTCOLO AODV EN OMNET++






```
package inet.examples.aodv;

import inet.common.lifecycle.LifecycleController;
import inet.common.scenario.ScenarioManager;
import inet.networklayer.configurator.ipv4.IPv4NetworkConfigurator;
import inet.networklayer.ipv4.RoutingTableRecorder;
```

```
import inet.node.aodv.AODVRouter;
import inet.physicallayer.idealradio.IdealRadioMedium;

network ShortestPath
{
    parameters:
        @display("bgb=961,662");
    submodules:
        radioMedium: IdealRadioMedium {
            parameters:
                @display("p=50,50");
        }
        configurator: IPv4NetworkConfigurator {
            parameters:
                addDefaultRoutes = false;
                addStaticRoutes = false;
                addSubnetRoutes = false;
                config = xml("<config><interface hosts='*' address='145.236.x.x' netmask='255.255.0.0'/></config>");
                @display("p=50,100");
        }
        routingTableRecorder: RoutingTableRecorder {
            parameters:
                @display("p=50,150");
        }
        lifecycleController: LifecycleController {
            parameters:
                @display("p=50,200");
        }
        scenarioManager: ScenarioManager {
            parameters:
                script = default(xml("<scenario/>"));
                @display("p=50,250");
        }
        sender: AODVRouter {
            parameters:
                @display("i=device/pocketpc_s;r=,,#707070;p=659,69");
```

```
    }
    A: AODVRouter {
      parameters:
        @display("i=device/pocketpc_s;r=,,#707070;p=569,127");
    }
    B: AODVRouter {
      parameters:
        @display("i=device/pocketpc_s;r=,,#707070;p=489,188");
    }
    C: AODVRouter {
      parameters:
        @display("i=device/pocketpc_s;r=,,#707070;p=386,273");
    }
    receiver: AODVRouter {
      parameters:
        @display("i=device/pocketpc_s;r=,,#707070;p=87,458");
    }
    D: AODVRouter {
      parameters:
        @display("i=device/pocketpc_s;r=,,#707070;p=313,341");
    }
    E: AODVRouter {
      parameters:
        @display("i=device/pocketpc_s;r=,,#707070;p=207,394");
    }
  connections allowunconnected:
}
```