



UNIVERSIDAD NACIONAL DE LOJA

FACULTAD DE LA ENERGÍA, LAS INDUSTRIAS Y LOS RECURSOS NATURALES NO RENOVABLES

CARRERA DE INGENIERÍA EN ELECTRÓNICA Y
TELECOMUNICACIONES

“IMPLEMENTACIÓN DE UN SIMULADOR DE REDES MEDIANTE SOFTWARE LIBRE PARA EL LABORATORIO DE TELECOMUNICACIONES DEL AEIRNNR”

TESIS DE GRADO PREVIO A OPTAR POR
EL TÍTULO DE INGENIERA EN
ELECTRÓNICA Y TELECOMUNICACIONES

AUTORA:

Maritza Elizabeth Palacios Morocho.

DIRECTOR:

Ing. Paulo Alberto Samaniego Rojas, Mg. Sc.

LOJA-ECUADOR

1859

2017



CERTIFICACIÓN

Ing.

Paulo Alberto Samaniego Rojas, Mg. Sc.

DIRECTOR DEL TRABAJO DE FIN DE TITULACIÓN

Certifico:

Haber dirigido, asesorado, revisado y corregido el presente trabajo de tesis de grado, en su proceso de investigación cuyo tema versa en “**IMPLEMENTACIÓN DE UN SIMULADOR DE REDES MEDIANTE SOFTWARE LIBRE PARA EL LABORATORIO DE TELECOMUNICACIONES DEL AEIRNNR**” previa a la obtención del título de Ingeniera en Electrónica y Telecomunicaciones, realizado por **Maritza Elizabeth Palacios Morocho**, la misma que cumple con la reglamentación y políticas de investigación, por lo que autorizo su presentación y posterior sustentación y defensa.

Loja, Mayo 2017

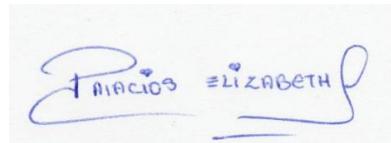


Ing. Paulo Alberto Samaniego Rojas, Mg. Sc.
DIRECTOR DEL TRABAJO DE FIN DE TITULACIÓN

AUTORÍA

Yo, **MARITZA ELIZABETH PALACIOS MOROCHO**, declaro ser autora del presente trabajo de tesis y eximo expresamente a la Universidad Nacional de Loja y a sus representantes jurídicos de posibles reclamos o acciones legales, por el contenido de la misma. Adicionalmente acepto y autorizo a la Universidad Nacional de Loja, la publicación de mi tesis en el Repositorio Institucional- Biblioteca Virtual

Firma:

A handwritten signature in blue ink. The first part reads "PALACIOS" and the second part reads "ELIZABETH". There is a small heart symbol between the two names.

Cédula: 1103805519

Fecha: 10 de enero de 2018

CARTA DE AUTORIZACIÓN DE TESIS POR PARTE DE LA AUTORA, PARA LA CONSULTA, REPRODUCCIÓN PARCIAL O TOTAL, Y PUBLICACIÓN ELECTRÓNICA DEL TEXTO COMPLETO.

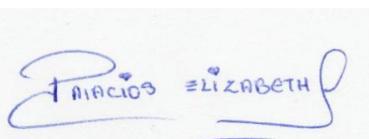
Yo, MARITZA ELIZABETH PALACIOS MOROCHO, declaro ser autora de la tesis titulada: “IMPLEMENTACIÓN DE UN SIMULADOR DE REDES MEDIANTE SOFTWARE LIBRE PARA EL LABORATORIO DE TELECOMUNICACIONES DEL AEIRNNR” como requisito para optar por grado de: INGENIERA EN ELECTRÓNICA Y TELECOMUNICACIONES; autorizo al Sistema Bibliotecario de la Universidad Nacional de Loja para que con fines académicos, muestre al mundo la producción intelectual de la Universidad, a través de la visibilidad de su contenido de la siguiente manera en el Repositorio Digital Institucional.

Los usuarios pueden consultar el contenido de este trabajo en el RDI, en las redes de información del país y del exterior, con las cuales tenga convenio la universidad.

La Universidad Nacional de Loja, no se responsabiliza por el plagio o copia de la tesis que realice un tercero.

Para constancia de esta autorización, en la ciudad de Loja, a los diez días del mes de enero del dos mil dieciocho.

Firma:



Autor: Maritza Elizabeth Palacios Morocho.

Cédula: 1103805519

Dirección: Loja, Pitas II (Av. Pablo Palacio y Viena).

Correo Electrónico: lizzy6d91@gmail.com

Teléfono: 072615611 **Celular:** 0980381450.

DATOS COMPLEMENTARIOS

Director de Tesis: Ing. Paulo Alberto Samaniego Rojas, Mg. Sc.

Tribunal de Grado: Ing. Juan Gabriel Ochoa Aldeán, Mg. Sc

Ing. Juan Manuel Galindo Vera, Mg. Sc.

Ing. Daniel Armando Jaramillo Chamba, Mg. Sc.

DEDICATORIA

A:

Mi madre Julia Morocho, por el esfuerzo y sacrificio que hizo para apoyarme en todo momento, por el amor que día a día me ha demostrado, quiero que sepa que todo esto se lo debo a usted.

Mis hermanos, Milton, Betty, Silvana y Cristian porque a pesar de ser tan diferentes siempre hemos estado ahí, el uno para el otro, porque son más fuertes los lazos que nos unen.

“Los amo”

AGRADECIMIENTO

A la Universidad Nacional de Loja, a la Facultad de la Energía, las Industrias y los Recursos Naturales No Renovables, a la Carrera de ingeniería en Electrónica y Telecomunicaciones y a cada uno de los docentes, que fueron parte de mi formación profesional.

Quiero expresar mi agradecimiento al Ing. Paulo Alberto Samaniego Rojas quien con sus acertadas correcciones y sugerencias me guió a la culminación de mi trabajo de titulación.

Así mismo deseo agradecer al Ing. Juan Gabriel Ochoa Aldeán por haberme guiado desde el inicio de mi trabajo de titulación hasta la finalización del mismo.

A mi gran amiga y consejera María Fernanda y a mi prima Andrea, que se ha convertido en una hermana más, gracias por enseñarme a ser siempre positiva.

A mis sobrinos, cuñada e igualmente a mis demás familiares y a todas esas personas que se ganaron un espacio en mi corazón, ya que sus palabras, actos de aliento me impulsaron a ser la persona que soy.

ÍNDICE

CERTIFICACIÓN	II
AUTORÍA	III
CARTA DE AUTORIZACIÓN	IV
DEDICATORIA.....	V
AGRADECIMIENTO	VI
ÍNDICE.....	VII
ÍNDICE DE FIGURAS	X
ÍNDICE DE TABLAS.....	XIV
ACRÓNIMOS	XV
1. TÍTULO	1
2. RESUMEN.....	2
2.1. Abstract	3
3. INTRODUCCIÓN	4
4. REVISIÓN DE LITERATURA.....	6
4.1. MODELO OSI.....	6
4.2. REDES CABLEADAS.....	6
4.2.1. Estándar IEEE 802.3 (Ethernet).	7
4.2.1.1 Formato de trama	9
4.2.1.2 Método de transmisión de datos CSMA/CD	9
4.2.1.3 Topología.	11
4.3 REDES INALÁMBRICAS	13
4.3.1 Redes de área personal inalámbrica.....	13
4.3.2 Redes de área local inalámbrica	14
4.3.2.1 Wi-Fi	15
4.3.3 Redes inalámbricas de área metropolitana	16
4.3.4 Redes inalámbricas de área extendida	17
4.3.5 Estándares de conexión	18
4.3.5.1 Estándar IEEE 802.11a	18
4.3.5.2 Estándar IEEE 802.11b	18
4.3.5.3 Estándar IEEE 802.11g	20
4.3.5.4 Estándar IEEE 802.11n	21

4.3.5.5	<i>Estándar IEEE 802.11ac</i>	21
4.4	REDES MALLADAS.....	22
4.4.1	Estándar IEEE 802.11s	22
4.4.2	Arquitectura de la red IEEE 802.11s	22
4.4.3	Protocolo Hibrido de Malla Inalámbrica (HWMP).....	23
4.4.3.1	<i>Modo Reactivo</i>	24
4.4.3.2	<i>Modo Proactivo</i>	25
4.4.3.3	<i>Airtime Link Metric (ALM)</i>	26
4.5	SIMULADOR DE REDES 3 (NS-3).....	27
4.5.1	Biblioteca NS-3	27
4.5.2	Modelos de pérdidas de propagación.	29
4.5.3	Estructura de un script en NS-3	29
5.	MATERIALES Y MÉTODOS	34
5.1	Descripción de los scripts utilizados.....	34
5.1.1	Instalación de NS-3	34
5.1.2	Instalación del módulo Openflow-Shiwch.	38
5.1.3	Estándar IEEE 802.3 – Ethernet.....	41
5.1.3.1	<i>Topología tipo bus</i>	41
5.1.3.2	<i>Topología tipo estrella</i>	47
5.1.4	Estándar IEEE 802.11b/g – WIFI.....	51
5.1.4.1	<i>Modo Infraestructura</i>	51
5.1.5	Estándar IEEE 802.11s	58
5.2	Descripción del Hardware Utilizado.....	63
5.3	Descripción del Software Utilizado.....	63
5.3.1	Sistema Operativo.....	63
5.3.2	Simulador de redes.	64
5.3.3	Herramientas de Análisis y Animación.....	65
5.3.3.1	<i>Herramientas de Análisis de datos -Wireshark</i>	65
5.3.3.2	<i>Herramientas de Animación- PyViz</i>	67
5.3.3.3	<i>Herramientas de Animación –NetAnim</i>	68
5.4	Diagrama de flujo	69
5.4.1	Diagrama general de un script en NS-3	69
6.	RESULTADOS	70

6.1	Estándar IEEE 802.3	70
6.1.1	Topología tipo bus	70
6.1.2	Topología tipo estrella	73
6.2	Estándar IEEE 802.11 b/g	78
6.2.1	Modo Infraestructura	78
6.2.1.1	<i>Modo Infraestructura – IEEE 802.11b</i>	78
6.2.1.2	<i>Modo Infraestructura – IEEE 802.11g</i>	83
6.2.2	Modo AD-HOC	88
6.2.2.1	<i>Modo AD-HOC – IEEE 802.11b</i>	88
6.2.2.2	<i>Modo AD-HOC – IEEE 802.11g</i>	93
6.3	Estándar IEEE 802.11s	97
6.4	Análisis de los resultados	115
7.	DISCUSIÓN	117
8.	CONCLUSIONES	120
9.	RECOMENDACIONES	121
10.	BIBLIOGRAFÍA	122
11.	ANEXOS	127
	ANEXO 1: INSTALACIÓN DE NS-3	127
	ANEXO 2: INSTALACIÓN DE OPENFLOW-SWITCH	131
	ANEXO 3: MODO DE INGRESO	135
	ANEXO 4: ESTÁNDAR IEEE 802.3 – ETHERNET - TOPOLOGÍA TIPO BUS....	137
	ANEXO 5: ESTÁNDAR IEEE 802.3 – ETHERNET - TOPOLOGÍA TIPO ESTRELLA.	142
	ANEXO 6: ESTÁNDAR IEEE 802.11b – WIFI - MODO INFRAESTRUCTURA...	154
	ANEXO 7: ESTÁNDAR IEEE 802.11g – WIFI - MODO INFRAESTRUCTURA...	161
	ANEXO 8: ESTÁNDAR IEEE 802.11b – WIFI - MODO AD-HOC	168
	ANEXO 9: ESTÁNDAR IEEE 802.11g – WIFI - MODO AD-HOC	175
	ANEXO 10: ESTÁNDAR IEEE 802.11s	182
	ANEXO 11: ALM	188
	ANEXO 12: MODELO DEL REPORTE DEL ESTÁNDAR IEEE 802.11s	190
	ANEXO 13: TABLAS DE LOS RESULTADOS DE LAS PRUEBAS DEL PROTOCOLO IEEE 802.11s EN DIFERENTES MALLAS, VARIANDO LA DISTANCIA ENTRE NODOS	192

ÍNDICE DE FIGURAS

Figura 1. Redes cableadas	7
Figura 2 Trama IEEE 802.3.....	9
Figura 3 Funcionamiento de CSMA/CD.....	10
Figura 4 Forma gráfica del funcionamiento de CSMA/CD.	10
Figura 5 Topología tipo bus.	11
Figura 6 Topología estrella: superior con hub y inferior con switch.	12
Figura 7 Clasificación de redes inalámbricas según su cobertura	13
Figura 8 Redes inalámbricas de área personal	14
Figura 9 Redes de área personal inalámbrica	14
Figura 10 Especificaciones IEEE 802.11	15
Figura 11 Modo Insfraestructura	16
Figura 12 Modo ad-hoc	16
Figura 13 Redes inalámbricas de área metropolitana.....	17
Figura 14 Redes inalámbricas de área extendida	17
Figura 15 Funcionamiento CSMA/CD.....	19
Figura 16 Espectro ensanchado por secuencia directa	20
Figura 17 Comparación del uso del ancho de banda entre FDM y OFDM.....	21
Figura 18 Arquitectura de la Red Mesh	22
Figura 19 Basic Service Set (BSS)	23
Figura 20 Modo Reactivo.....	24
Figura 21 Modo Proactivo.....	25
Figura 22 Bibliotecas de NS-3	28
Figura 23 Diagrama de Aplicaciones	33
Figura 24 Ingreso en modo root	34
Figura 25 Instalación de pre-requisitos	35
Figura 26 Descarga y descomprensión del archivo que contiene a NS-3	35
Figura 27 Construcción de NS-3	36
Figura 28 Mensaje de construcción exitosa.....	36
Figura 29 Resultado del primer ejemplo	37
Figura 30 Descarga y descompresión del módulo.....	38
Figura 31 Instalación del módulo.....	39
Figura 32 Construcción finalizada.....	39
Figura 33 Mensaje de OpenFlow como no habilitado.....	40
Figura 34 Módulo activado.	40
Figura 35 Esquema de la Topología tipo Bus	41
Figura 36 Topología tipo estrella	47
Figura 37 Modo infraestructura.....	51
Figura 38 Topología estándar IEEE 802.11s.....	58
Figura 39 Sistema operativo Ubuntu	63
Figura 40 Versión de Ubuntu utilizada.	64

Figura 41 Versiones de NS-3	64
Figura 42 Proceso de instalación de wireshark	65
Figura 43 Configuración de wireshark	66
Figura 44 Programa Wireshark	66
Figura 45 Herramienta PyViz.....	67
Figura 46 Sentencias para ejecutar PyViz	67
Figura 47 Herramienta de visualización NetAnim.....	68
Figura 48 Proceso para ejecutar NetAnim	68
Figura 49. Diagrama de flujo general de un script en NS-3.....	69
Figura 50 Topología tipo bus – Herramienta PyViz	70
Figura 51 Topología tipo bus – Herramienta NetAnim	70
Figura 52 Paquete enviados del nodo 0 al nodo 1	71
Figura 53 Envió de paquetes del nodo 1 al nodo 10.....	71
Figura 54 Envió de paquetes del nodo 10 al nodo 1.....	71
Figura 55 Resultado de archivo .tr – Topología tipo bus	71
Figura 56 Archivos generados – Topología tipo bus.....	72
Figura 57 Resultado de archivo .pcap – Herramienta Wireshark – Nodo 5.....	72
Figura 58 Topología tipo estrella mediante la herramienta Pyviz.....	73
Figura 59 Topología tipo estrella – Herramienta NetAnim.....	74
Figura 60 Direcciones Asignadas	74
Figura 61 Envió de paquetes en topología estrella. –Herramienta NetAnim	75
Figura 62 Diagrama de envio de paquetes desde el nodo 6 a los diferentes nodos.....	75
Figura 63 Resultado de archivo . tr – Topología estrella	76
Figura 64 Archivos generados –Topología tipo estrella.....	76
Figura 65 Resultado de archivo .pcap – Topología tipo estrella – Nodo 6	77
Figura 66 Modo Infraestructura – Herramienta NetAnm.....	78
Figura 67 Asociación del AP con las estaciones	78
Figura 68 Archivos generados – Modo infraestructura.....	79
Figura 69 Resultado archivo .pcap – Modo infraestructura - AP	79
Figura 70 Resultado archivo .pcap – Modo infraestructura – Estación 1	80
Figura 71 Resultado archivo .pcap – Modo infraestructura – Estación 2	80
Figura 72 Resultado paquetes enviados	81
Figura 73 Configuración modo infraestructura, estaciones separadas 55 m	81
Figura 74 Comportamiento de la red en función de la variación de la distancia.	82
Figura 75 Configuración modo infraestructura, estaciones separadas 68 m	82
Figura 76 Configuración modo infraestructura, estaciones separadas 71 m	82
Figura 77 Archivos generados – Modo infraestructura.....	83
Figura 78 Resultado archivo .pcap – Modo Infraestructura - AP.....	83
Figura 79 Resultado archivo . pcap – Modo infraestructura – Estación 1	84
Figura 80 Resultado archivo .pcap – Modo infraestructura – Estación 2	84
Figura 81 Resultado paquetes enviados.	85
Figura 82 Configuración modo infraestructura, estaciones separadas 10 m	85

Figura 83 Comportamiento de la red en función de la distancia.....	86
Figura 84 Configuración modo infraestructura, estaciones separadas 26 m	86
Figura 85 Configuración modo infraestructura, estaciones separadas 28 m	87
Figura 86 Modo Ad –Hoc – Herramienta PyViz.....	88
Figura 87 Archivos generados – Modo Ad - Hoc	88
Figura 88 Envió de paquete – Modo Ad-Hoc - IEEE 802.11b	88
Figura 89 Diagrama de paquetes enviados – Modo Ad-Hoc - IEEE 802.11b.	89
Figura 90 Resultado de archivo . tr – Modo Ad – Hoc –IEEE 802.11b.....	90
Figura 91 Resultados de archivo .pcap – Modo Ad – Hoc -Estación 1- IEEE 802.11b.....	90
Figura 92 Resultado archivo .pcap – Modo Ad – Hoc - Estación 2-- IEEE 802.11b....	91
Figura 93 Resultado paquetes enviados – Modo Ad-Hoc - IEEE 802.11b.....	91
Figura 94 Configuración modo ad-hoc. Estaciones separadas 50 m.....	92
Figura 95 Comportamiento de la red en función de la distancia – Modo Ad-Hoc - IEEE 802.11b.	92
Figura 96 Configuración modo ad-hoc. Estaciones separadas 61 m – IEEE 802.11b ..	92
Figura 97 Configuración modo ad-hoc. Estaciones separadas 64 m – IEEE 802.11b..	92
Figura 98 Archivos generados – Modo Ad –Hoc –IEEE 802.11g.....	93
Figura 99 Resultado archivo .tr – Modo Ad-Hoc –IEEE 802.11g.....	93
Figura 100 Resultado archivo .pcap – Modo Ad – Hoc – Estación 1 – IEEE 802.11g.	94
Figura 101 Resultado archivo .pcap – Modo Ad – Hoc – Estación 2 – IEEE 802.11g.	95
Figura 102 Resultado paquetes enviados – Modo Ad –Hoc –IEEE 802.11g.....	95
Figura 103 Configuración modo ad-hoc. Estaciones separadas 22 m – Modo Ad –Hoc – IEEE 802.11g.	96
Figura 104 Comportamiento de la red en función de la distancia – Modo Ad –Hoc –IEEE 802.11g.	96
Figura 105 Configuración modo ad-hoc. Estaciones separadas 24 m – Modo Ad –Hoc – IEEE 802.11g.	96
Figura 106 Configuración modo ad-hoc. Estaciones separadas 25 m – Modo Ad –Hoc – IEEE 802.11g.	96
Figura 107 Archivos generados – Red Mesh	97
Figura 108 Resultado archivo .pcap – Red Mesh.....	98
Figura 109 Reporte para el MP: 00:00:00:00:00:01.....	98
Figura 110 Mensajes PREQ y PREP	99
Figura 111 Reporte del nodo 1	100
Figura 112 Asociación desde el nodo 1 - NetAnime.....	100
Figura 113 Reporte del nodo 2	101
Figura 114 Asociaciones del nodo 3	101
Figura 115 Cálculo de ALM	102
Figura 116 Métrica ALM del nodo 0 hacia el nodo 1	103
Figura 117 Métrica ALM del nodo 0 hacia el nodo 2	104
Figura 118 Métrica ALM del nodo 1 hacia el nodo 0	105
Figura 119 Métrica ALM del nodo 1 hacia el nodo 3	106

Figura 120 Métrica ALM del nodo 2 hacia el nodo 0	107
Figura 121 Métrica ALM del nodo 2 hacia el nodo 3	108
Figura 122 Métrica ALM del nodo 3 hacia el nodo 1	109
Figura 123 Métrica ALM del nodo 3 hacia el nodo 2	110
Figura 124 Paquetes recibidos y perdidos en una malla de 2x2 con una separación de 100m entre nodos	111
Figura 125 Reporte del nodo 37	111
Figura 126 Asociaciones del nodo 3	112
Figura 127 Paquetes recibidos vs paquetes perdidos.	113
Figura 128 Tasa de paquetes perdidos – Nodos separados 100m	114
Figura 129 Throughput – nodos separados 100m	114
Figura 130: Malla formada por 4 nodos variando la distancia entre nodos.	193
Figura 131 Malla formada por 10 nodos variando la distancia entre nodos.	194
Figura 132 Malla formada por 15 nodos variando la distancia entre nodos.	195
Figura 133 Malla formada por 30 nodos variando la distancia entre nodos.	196
Figura 134 Malla formada por 36 nodos variando la distancia entre nodos.	197
Figura 135 Malla formada por 49 nodos variando la distancia entre nodos.	198

ÍNDICE DE TABLAS

Tabla 1 Tecnología IEEE 802.3 (<i>Ethernet</i>)	8
Tabla 2 Campos de la trama IEEE 802.3	9
Tabla 3 Parámetros de ALM	26
Tabla 4 Variaciones en las configuraciones de los diferentes estándares.	57
Tabla 5 Características del equipo.....	63
Tabla 6 Resultados obtenidos al variar el número de nodos de la red.	112
Tabla 7 Resultados del protocolo IEEE 802.11s para una malla formada por 4 nodos.	
.....	193
Tabla 8 Resultados del protocolo IEEE 802.11s para una malla formada por 10 nodos.	
.....	194
Tabla 9 Resultados del protocolo IEEE 802.11s para una malla formada por 15 nodos	
.....	195
Tabla 10 Resultados del protocolo IEEE 802.11s para una malla formada por 30 nodos	
.....	196
Tabla 11 Resultados del protocolo IEEE 802.11s para una malla formada por 36 nodos.	
.....	197
Tabla 12 Resultados del protocolo IEEE 802.11s para una malla formada por 49 nodos.	
.....	198

ACRÓNIMOS

ALM	
Métrica de tiempo de aire.	18
AP	
Punto de acceso.	30
BSS	
Conjunto de servicio básico.	38
CIEYT	
Carrera de Ingeniería en Electrónica y Telecomunicaciones.	20
CSMA/CA	
Acceso múltiple con escucha de portadora y evasión de colisiones.	33
CSMA/CD	
Acceso múltiple con escucha de portadora y detección de colisiones.	22
DIFS	
Duración del espacio entre tramas.	34
DSSS	
Espectro ensanchado por secuencia directa.	33
FDM	
Acceso múltiple por división de frecuencias.	36
FEIRNNR	
Facultad de Ingeniería en Eléctronica y Telecomunicaciones	18
GNU	
Licencia pública general.	20
GPLv2	
Licencia pública general versión 2.	20
HWMP	
Protocolo híbrido de malla inalámbrica.	18
IEEE	
Instituto de Ingeniería Eléctrica y Electrónica.	18
LAN	
Redes de área local.	22
LTS	
Soporte a largo plazo.	18
MAC	
Control de acceso al medio.	39
MAP	
Punto de acceso de malla.	38
MIMO	
Múltiples entradas, multiples salidas.	36
MP	

Punto de malla.	38
<i>MPP</i>	
Punto de portal de malla.	38
<i>NS-3</i>	
Simulador de redes versión 3	18
<i>OFDM</i>	
Acceso múltiple por división de frecuencias ortogonales.	33
<i>OSI</i>	
Interconexión de sistemas abiertos.	39
<i>PC</i>	
Estación de trabajo.	21
<i>RANN</i>	
Anuncio de la raíz.	39
<i>RERR</i>	
Error de ruta.	39
<i>RREP</i>	
Respuesta de ruta.	39
<i>RREQ</i> ,	
Solicitud de ruta.	39
<i>TCP</i>	
Protocolo de control de transmisión.	47
<i>UDP</i>	
Protocolo de datagrama de usuario.	47
<i>WLAN</i>	
Redes de área local inalámbrica.	28
<i>WMAN</i>	
Redes de área metropolitana.	28
<i>WPAN</i>	
Redes de área personal inalámbrica.	28
<i>WWAN</i>	
Redes de área extendida.	28

1. TÍTULO

**“IMPLEMENTACIÓN DE UN SIMULADOR DE REDES
MEDIANTE SOFTWARE LIBRE PARA EL
LABORATORIO DE TELECOMUNICACIONES DEL
AEIRNNR”**

2. RESUMEN

En el presente trabajo se realizó la implementación del simulador de redes mediante software libre para el laboratorio de telecomunicaciones del AEIRNNR y la configuración de diferentes estándares de redes de datos.

Inicialmente, se detalla la instalación del simulador de red 3 (NS-3, por sus siglas en inglés) en el sistema operativo Ubuntu 14.04 LTS con la finalidad de implementarlo en el laboratorio de telecomunicaciones del AEIRNNR, ya que el mismo servirá de base para las futuras generaciones de alumnos para un estudio técnico e investigativo sobre redes cableadas e inalámbricas.

Por otra parte, se realizó la configuración de las redes que cumplan con los estándares:

- 1) IEEE 802.3 (Ethernet), para evaluar su desempeño tanto en la topología tipo bus como estrella.
- 2) IEEE 802.11 b/g (WI-FI), para evaluar su desempeño tanto en modo infraestructura como en modo Ad-Hoc.
- 3) IEEE 802.11s para evaluar el desempeño del protocolo estandarizado HWMP junto a su métrica tiempo de aire (ALM, por sus siglas en inglés).

Las configuraciones antes mencionadas permiten estudiar de manera más profunda el funcionamiento de las mismas en sus diferentes topologías y así constatar lo aprendido en los salones de clases, de igual modo determinar las ventajas y desventajas de dichas redes de datos, todo esto sin la necesidad de requerir una inversión económica en la adquisición de equipos de red que se utilizarían para el estudio de este tipo de redes; en las redes *mesh*, se compara su eficiencia al aumentar el número de nodos a una distancia específica en este caso se realizaron pruebas hasta con 100 nodos.

Con el fin de un mejor entendimiento del funcionamiento de cada una de estas redes de datos, se utilizó tanto herramientas gráficas como PyViz y NetAnim, como también un analizador de protocolos de red denominado Wireshark, además de Microsoft Excel.

2.1. Abstract

In the present work was made the implementation of the network simulator using free software for the telecommunications laboratory of the FEIRNNR and the configuration of different network standards of data.

Initially, the installation of the network simulator (NS3, for the acronym in English) is detailed in the operating system Ubuntu 14.04 LTS to the purpose of the implement it in the laboratory of telecommunications of the AEIRNNR, then it will serve as a basis for future generations of students for a technical and investigative study on wired and wireless networks.

On the other part, it was made the configuration of the networks that comply with the standards:

- 1) IEEE 802.3 (Ethernet), to evaluate their performance both topology type bus and star.
- 2) IEEE 802.11 b/g (WI-FI), to evaluate their performance both mode infrastructure and AD-HOC.
- 3) IEEE 802.11s, to evaluate their performance of the standardized protocol HWMP together their metric time of air (ALM, for the acronym in English).

The configurations mentioned allow to study in a more profound way the operation of different topologies of them and verify has been learned in the classrooms, likewise determine the advantages and disadvantages of this networks. It'll all without the need to require an economic investment in the acquisition of network equipment that would be used for the study of this type networks; in mesh networks, their efficiency is compared by the increasing the number of nodes at a specify distance in this case the tests were performed up to 100 nodes.

To the purposes of had better understand the operation of each at these data networks, it was used graphics tools such PyViz and NetAnim, and a network protocol analyzer called Wireshark, also Microsoft Excel.

3. INTRODUCCIÓN

Actualmente el laboratorio de telecomunicaciones de la Carrera de Ingeniería en Electrónica y Telecomunicaciones (CIEYT) anexo a la Facultad de Energía, las Industrias y los Recursos Naturales no Renovables (FEIRNNR) de la Universidad Nacional de Loja (UNL), no cuenta con una herramienta que permita a través de la simulación; pronosticar el comportamiento de redes de datos tanto inalámbricas como cableadas.

La idea del presente proyecto tiene como finalidad la implementación de un simulador de redes de software libre en el laboratorio de telecomunicaciones, ya que el mismo logrará reforzar y profundizar los conocimientos de los estudiantes en el campo de las redes de datos.

La importancia del simulador radica en la predicción de la respuesta a una infinidad de eventos que pueden afectar el desempeño de la red de datos una vez implementada, por otra parte, permite mejorar la calidad de sus aplicaciones y servicios, además se emplea la simulación como una herramienta primordial tanto para el diseño como para la implementación de redes de datos.

Cabe mencionar que las simulaciones son relevantes debido a que permiten saber el comportamiento de diferentes parámetros dentro de la red, como lo es el retardo, paquetes transmitidos, paquetes perdidos y al mismo tiempo permite evaluar su desempeño con la recreación de escenarios reales. Finalmente representan un método de gran eficiencia en el momento de la enseñanza dado que el estudiante puede comprender de forma clara y profunda. Paralelamente si nos enfocamos en el campo de la investigación nos permite economizar tiempo y dinero.

El simulador escogido es NS-3 ya que se basa en software libre, además de los beneficios que trae consigo la licencia pública general (*GNU*, por sus siglas en inglés) y la licencia pública general de *GNU* versión 2 (*GPLv2*, por sus siglas en inglés), estas licencias aseguran que todos los programas bajo las mismas se pueden copiar, distribuir y modificar por cualquier usuario de forma libre, a fin de que existan mejoras de los

mismos, disminuyendo así los errores. Cabe mencionar que la adquisición de este tipo de software libre no representa costo económico alguno.

Otro aspecto importante que justifica la conveniencia de la investigación es el continuo desarrollo de las redes de datos, dado que actualmente son un campo de investigación con mucho potencial, además la simulación de dichas redes nos permite evaluar a fondo el comportamiento de las mismas y realizar su respectivo análisis con la finalidad de compararlas y medir su desempeño. Así mismo, esta investigación servirá como base para que futuras generaciones de estudiantes tengan una guía para la utilización de este simulador y continúen con la investigación sobre los diferentes tipos de redes de datos.

Para la realización de esta investigación se plantearon los siguientes objetivos:

- Instalar el simulador de redes NS-3 en el laboratorio de Telecomunicaciones sobre una estación de trabajo (PC) con sistema operativo Linux distribución Ubuntu 14.04 LTS.
- Configurar NS-3 para simular redes cableadas que cumplan con el estándar IEEE 802.3 (Ethernet).
- Configurar NS-3 para simular redes inalámbricas que cumplan con el estándar IEEE 802.11 b/g (WI-FI).
- Configurar el simulador NS3 con la finalidad de analizar el protocolo de la red MESH: 802.11s, HWMP.

4. REVISIÓN DE LITERATURA

4.1. MODELO OSI

El modelo de Interconexión de Sistemas Abiertos es un modelo de referencia, está compuesto por siete capas: (Tanenbaum, 2003)

- Física.
- Enlace de datos.
- Red.
- Transporte.
- Sesión.
- Presentación.
- Aplicación.

En la capa física se transmiten los bits por el canal de comunicación. En la capa de enlace es el medio de comunicación entre la capa física y la de red, esta fragmenta los datos en tramas pequeñas para su correcta transmisión. La capa de red controla la subred es decir determina la ruta que sigue el paquete para llegar a su destino. La capa de transporte separa la capa de sesión de la capa de red, además se encarga de recibir los datos provenientes de las capas superiores, dividirlos si su tamaño lo amerita y enviarlos a la capa de red. La capa de sesión permite establecer sesiones entre usuarios diferentes. La capa de presentación se encarga de la sintaxis y semántica de la información. La capa de aplicación contiene todos los protocolos que el usuario utiliza. (Tanenbaum, 2003).

4.2. REDES CABLEADAS

Las redes cableadas (véase figura 1) son aquellas en la que los datos se transmiten, desde un dispositivo a otro a través de cables de datos conocidos como cables Ethernet. Son utilizadas cuando el usuario necesita transferir grandes cantidades de información a altas velocidades como archivos multimedia. (Colegio de la UNLPAM, 2011).



Figura 1. Redes cableadas.
Fuente: (Colegio de la UNLPAM, 2011).

4.2.1. Estándar IEEE 802.3 (Ethernet).

El estándar IEEE 802.3 también conocido como Ethernet fue publicado por primera vez en el año 1983; generalmente aplicado en redes de área local (*LAN*, por sus siglas en inglés), emplea el método de transmisión de datos denominado Acceso múltiple con detección de portadora y detección de colisiones (*CSMA/CD*, por sus siglas en inglés) (Castillo, 2005). La tabla 1 muestra las normas de trasmisión, las velocidades de transmisión, el tipo de cable, la distancia máxima y la topología permitida en el estándar IEEE 802.3 (Barbieri, s.f)

Tabla 1*Normas de Transmisión de IEEE 802.3 (Ethernet)*

Norma de transmisión	Velocidad de Transmisión	Tipo de cable	Distancia máxima	Topología
10Base2	10 Mbps	Coaxial	185 m	Bus (Conector T)
10BaseT	10 Mbps	Par trenzado	100 m	Estrella (Hub o Switch)
10BaseF	10 Mbps	Fibra óptica	2000 m	Estrella (Hub o Switch)
100BaseT4	100 Mbps	Par trenzado (categoría 3UTP)	100 m	Estrella, Half Duplex(hub) y Full Duplex (switch)
100BaseTx	100 Mbps	Par trenzado (categoría 5UTP)	100 m	Estrella, Half Duplex(hub) y Full Duplex (switch)
100BaseFx	100 Mbps	Fibra óptica	2000 m	No permite el uso de hubs
1000BaseT	1000 Mbps	4 pares trenzado (categoría 5e ó 6 UTP)	100 m	Estrella, Full Duplex (switch)
1000BaseSx	1000 Mbps	Fibra óptica (multimodo)	550 m	Estrella, Full Duplex (switch)
1000BaseLx	1000 Mbps	Fibra óptica (monomodo)	5000 m	Estrella, Full Duplex (switch)

Nota: Recuperado de (Barbieri, s.f.).

4.2.1.1 Formato de trama

En el estándar IEEE 802.3 la trama se encuentra formada por diferentes campos, como se muestra en la figura 2.



Figura 2 Trama IEEE 802.3

Fuente: (Molero, s.f.)

La tabla 2 mostrada a continuación se describe cada uno de estos campos.

Tabla 2
Campos de la trama IEEE 802.3

Campo	Descripción
Preámbulo	Campo de 7B
Inicio	Campo de 1B
Dirección de destino	Campo de 6B
Dirección de origen	Campo de 6B
Longitud datos	Campo de 2B
Datos	Varía entre 0 y 1500B
Relleno	Equivale al valor faltante para que la trama tenga un tamaño mínimo a 64B
CRC	Codifica el control de error de la trama

Nota: Recuperado de (Molero, s.f.)

4.2.1.2 Método de transmisión de datos CSMA/CD

CSMA/CD funciona de la siguiente forma (véase figura 3 y figura 4): (Tanenbaum, 2003).

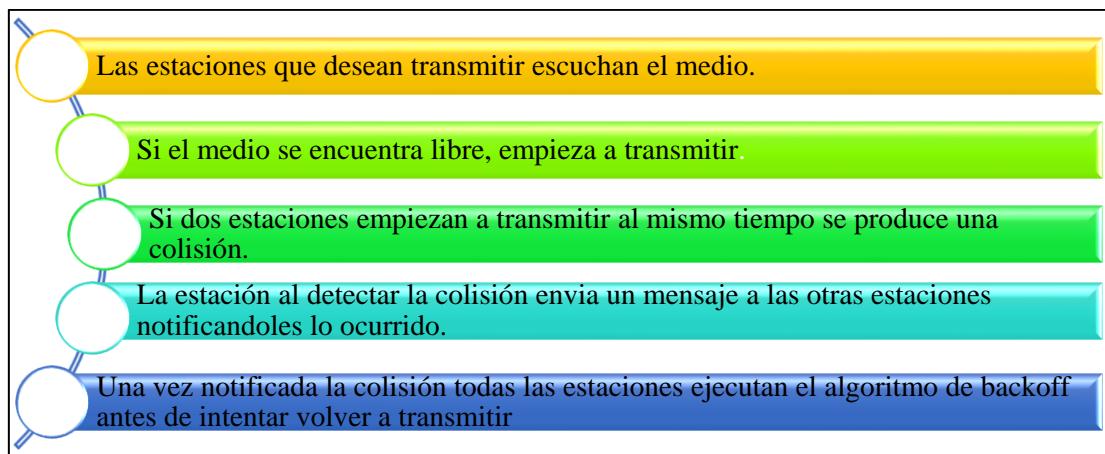


Figura 3 Funcionamiento de CSMA/CD.

Fuente: (Tanenbaum, 2003).

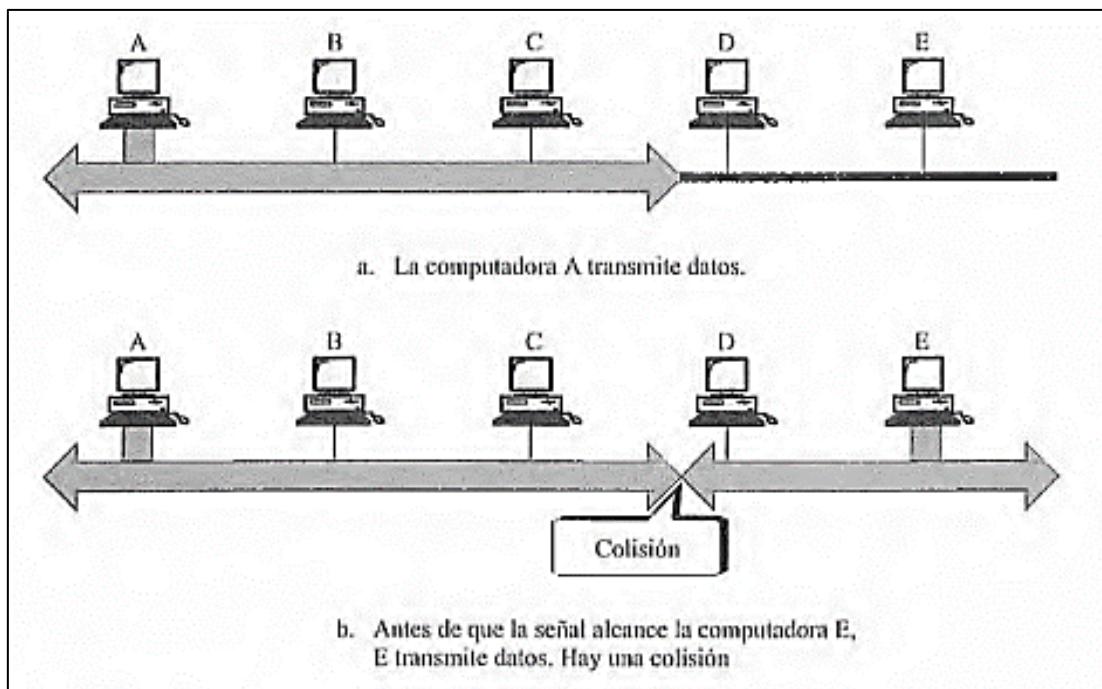


Figura 4 Forma gráfica del funcionamiento de CSMA/CD.

Fuente: (Forouzan, 2001).

4.2.1.3 Topología.

Se emplea dos tipos de topología: bus, estrella.

Bus

En este tipo de topología (véase figura 5) se puede unir varios segmentos hasta que lleguen a una longitud máxima equivalente a 2.5 km, entre sus características se puede mencionar que no pueden existir más de dos repetidores entre dos estaciones y solo se pueden colocar 1024 estaciones como máximo, su desventaja principal radica en que el posible corte de un segmento causaría la perdida de la comunicación con todas las estaciones conectadas a dicho segmento (Universidad de Oviedo, s.f.).

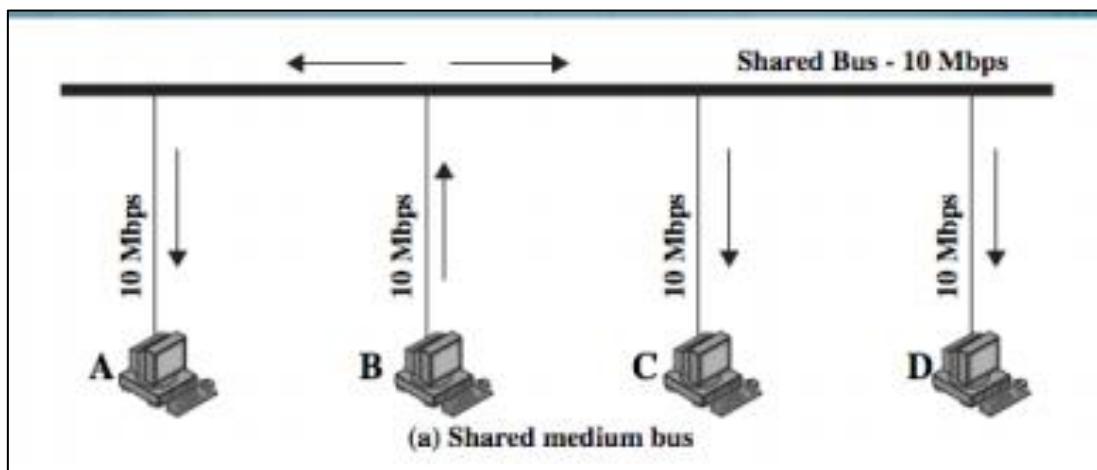


Figura 5 Topología tipo bus.

Fuente: (Universidad de Granada, s.f.)

Estrella

En este tipo de topología (véase figura 6) las estaciones se encuentran conectadas a través de un *hub* o *switch*, la ventaja que presenta es que si se daña algún segmento solo se pierde la comunicación con la estación a la que pertenece dicho segmento, solo cuando el *hub* o el *switch* dejan de funcionar se pierde la comunicación de todas las estaciones (Universidad de Oviedo, s.f.).

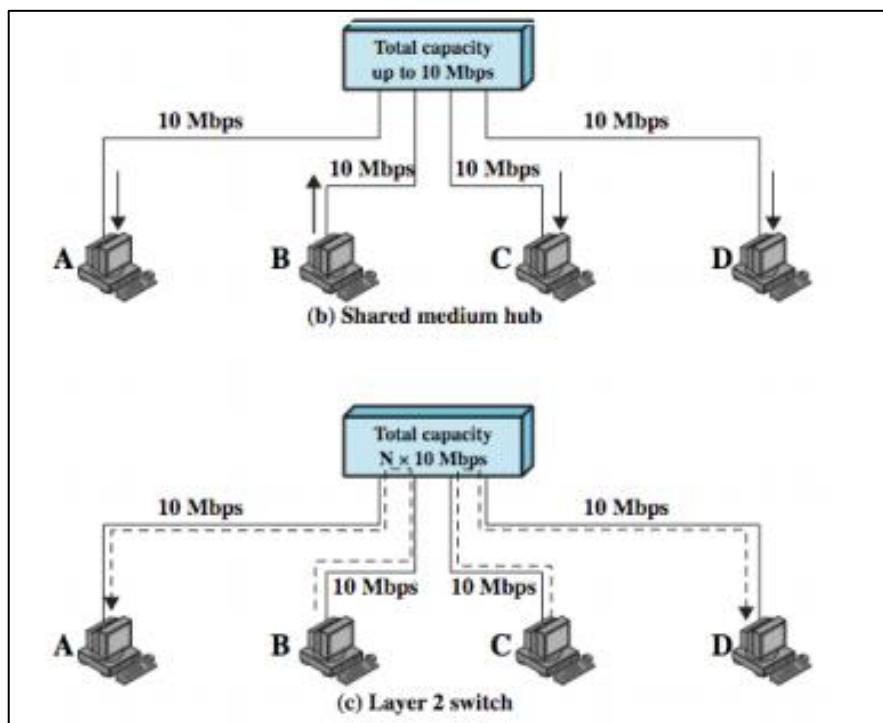


Figura 6 Topología estrella: superior con hub y inferior con switch.
Fuente: (Universidad de Granada, s.f.)

4.3 REDES INALÁMBRICAS

Las redes inalámbricas lograron un gran impacto en la forma de trabajar en el día a día de las personas ya que permiten la movilidad del usuario. Este tipo de redes utilizan las ondas electromagnéticas para conectar diferentes dispositivos a grandes distancias, fueron pensadas para cubrir las áreas de difícil acceso, escenarios en los cuales las redes cableadas presentan dificultad para su implementación y para lograr una mayor cobertura de las mismas (Gralla, 2006).

Las redes inalámbricas se las puede clasificar tomando en cuenta el área de cobertura, se establecen 4 grandes grupos (véase figura 7) (Vilcaguano, 2007):

- ✚ Redes de área personal inalámbrica (WPAN, por sus siglas en inglés).
- ✚ Redes de área local inalámbrica (WLAN, por sus siglas en inglés).
- ✚ Redes de área metropolitana inalámbrica (WMAN, por sus siglas en inglés).
- ✚ Redes de área extendida inalámbrica (WWAN, por sus siglas en inglés).

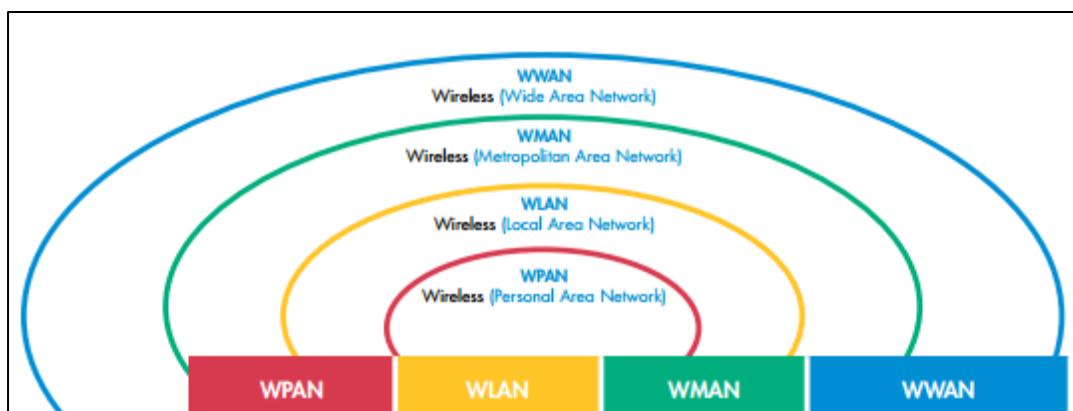


Figura 7 Clasificación de redes inalámbricas según su cobertura

Fuente: (Mifsud, 2013)

4.3.1 Redes de área personal inalámbrica

Las redes de área personal inalámbrica (véase figura 8) son redes pequeñas, que permiten la comunicación entre dispositivos de uso personal a distancias cortas, aproximadamente 10m, las mismas que se encuentran conformadas hasta por 8 equipos,

en este tipo de redes tenemos los enlaces infrarrojos, Bluetooth y ZigBee. Poseen bajo consumo, fácil integración y una topología en malla (Caro, s.f).

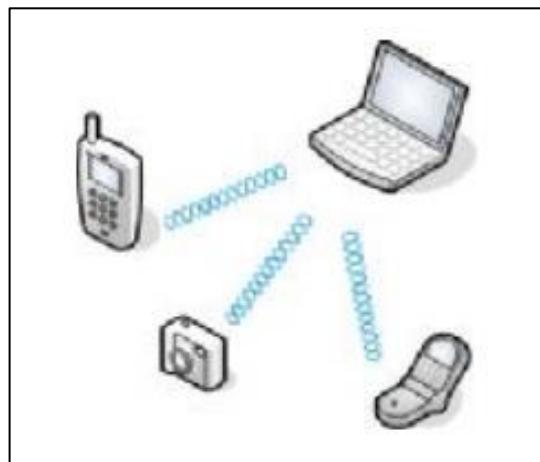


Figura 8 Redes inalámbricas de área personal
Fuente: (Novoa, 2007).

4.3.2 Redes de área local inalámbrica

Ofrece una conexión inalámbrica entre varios dispositivos dentro de un área local (véase figura 9), utiliza como medio de transmisión las emisiones de radiofrecuencia. Presenta varias ventajas entre las que se puede mencionar la movilidad, una fácil instalación y configuración además de una reducción en los costos. En este tipo de redes tenemos las siguientes: Wi-Fi, Home RF, Hiper LAN, (Vilcaguano, 2007).

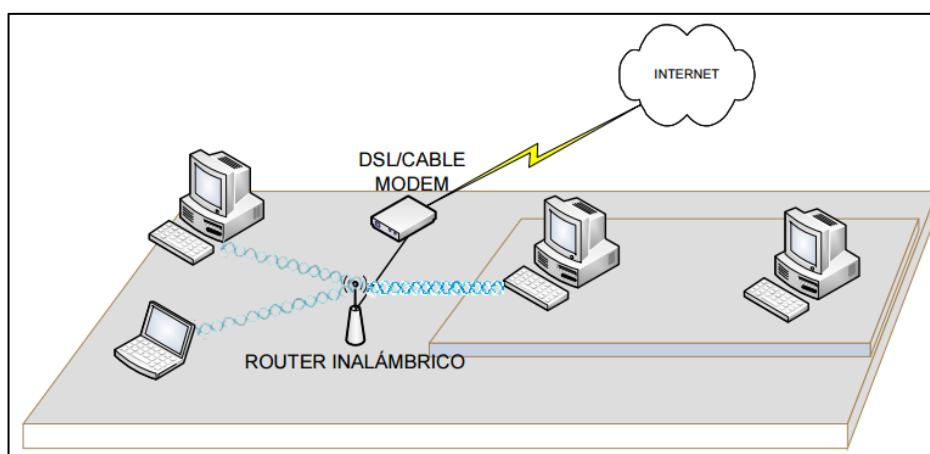


Figura 9 Redes de área personal inalámbrica
Fuente: (Novoa, 2007)

4.3.2.1 Wi-Fi

Wi-fi es un grupo de estándares definidos para redes inalámbrica los mismos que se encuentran basados en las especificaciones IEEE 802.11 (véase figura 10) y cuenta con variantes con diferentes modificaciones y mejoras. (Pellejero, 2006).

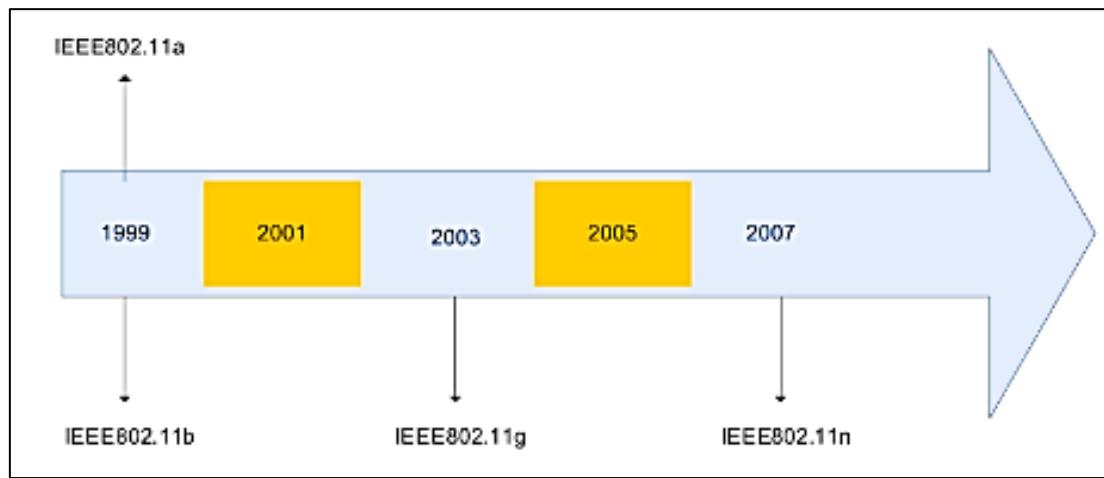


Figura 10 Especificaciones IEEE 802.11

Fuente: (Pellejero, 2006)

Pueden operar en modo infraestructura (véase figura 11) que consiste en un sistema compuesto por una red cableada en la cual se añade un sistema de WLAN para esto se emplea un punto de acceso (*AP*, por sus siglas en inglés), que comúnmente se utiliza como elemento centralizador y gestor de los terminales que se encuentran dentro de su zona de cobertura, el otro modo de operación es el ad-hoc (véase figura 12) que no es más que la interconexión de ordenadores con WLAN sin usar un AP, este modo no posee infraestructura cableada y es muy utilizado en entornos cerrados donde lo único que se busca es lograr un intercambio de información óptimo (Roig, 2003).

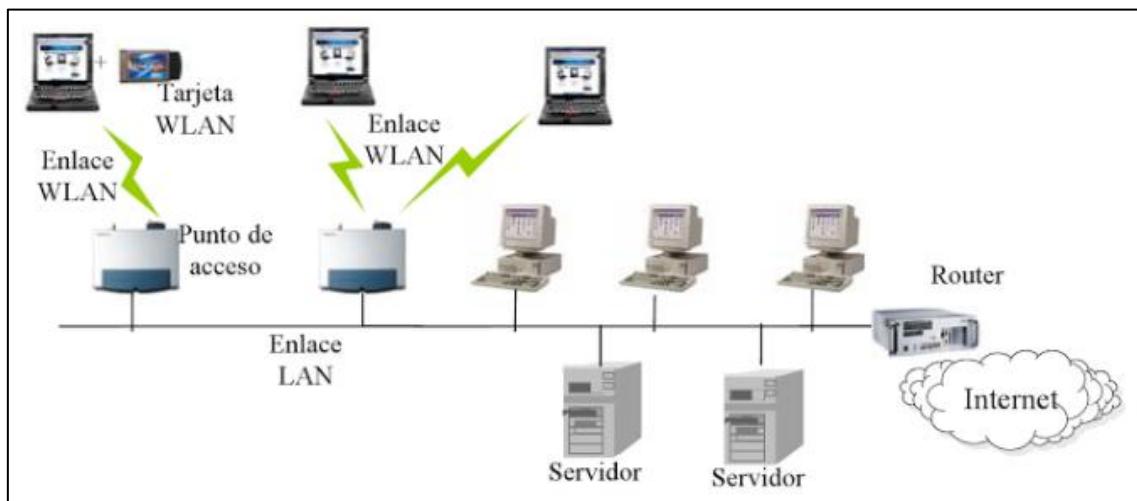


Figura 11 Modo Insfraestructura

Fuente: (Roig, 2003)



Figura 12 Modo ad-hoc

Fuente: (Roig, 2003)

4.3.3 Redes inalámbricas de área metropolitana

Estás redes tienen un área de cobertura geográfica extensa (véase figura 13), aproximadamente al tamaño de una ciudad, es una red de alta velocidad e integra los servicios de transmisión mediante medios inalámbricos: video voz y datos (Ruiz, 2007).

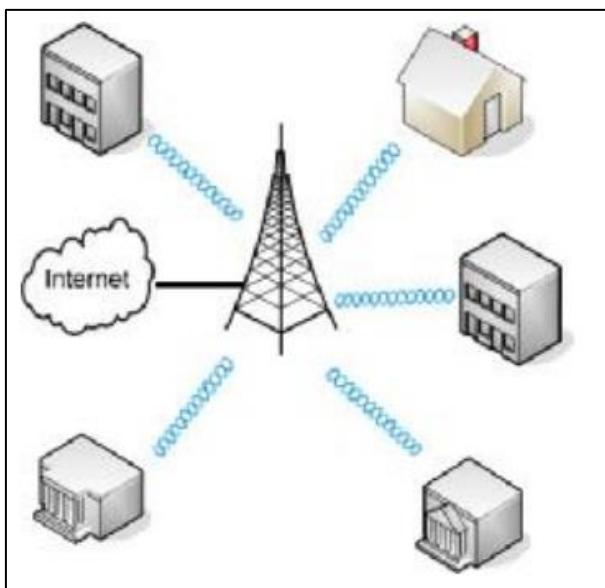


Figura 13 Redes inalámbricas de área metropolitana

Fuente: (Novoa, 2007).

4.3.4 Redes inalámbricas de área extendida

Está redes cubren áreas geográficas extensas (véase figura 14) como países o continentes, una de las ventajas es la movilidad en otras palabras permite a los usuarios conectados en esta red no perder la conexión dentro de toda esa área (Sosa, 2011).

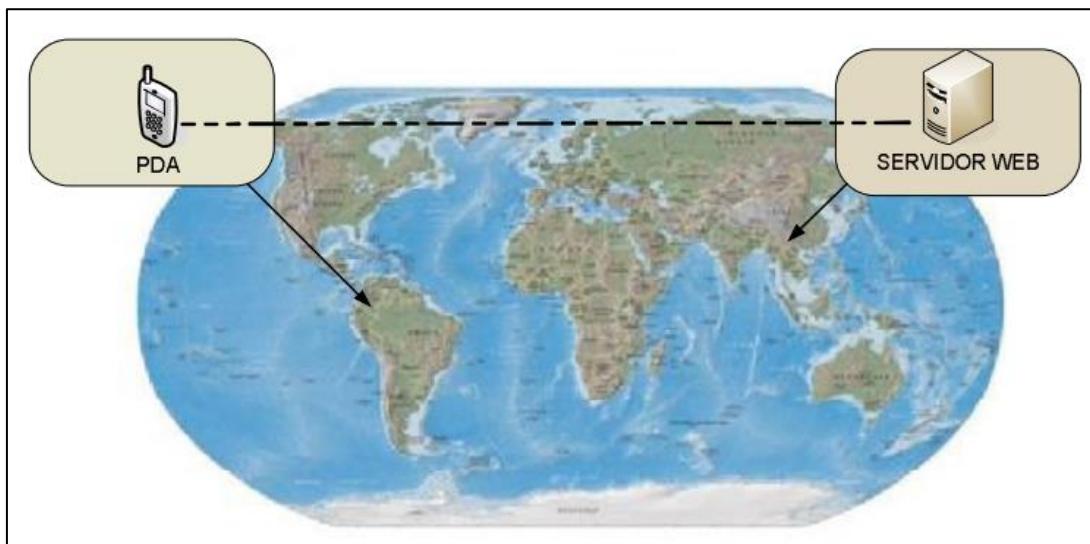


Figura 14 Redes inalámbricas de área extendida

Fuente: (Novoa, 2007)

4.3.5 Estándares de conexión

4.3.5.1 Estándar IEEE 802.11a

El estándar IEEE 802.11a, posee una tasa de transferencia de hasta 54 Mbps, emplea una selección adaptativa de la velocidad, en otras palabras, permite disminuir la tasa de transferencia basándose en las condiciones del medio. Trabaja en la frecuencia de 5 Ghz y emplea la técnica de modulación OFDM (Escudero, 2007).

4.3.5.2 Estándar IEEE 802.11b

El estándar IEEE 802.11b, posee una tasa de transferencia de 11Mbps, utiliza una tasa dinámica que permite el ajuste de forma automática tomando en cuenta las condiciones de ruido, en otras palabras, esta va ir disminuyendo a 5.5 Mbps, 2 Mbps hasta llegar a 1 Mbps, trabaja en la banda de frecuencia de 2,4 Ghz y usa el método de acceso CSMA/CA o RTS/CTS, tiene un alcance de 30m en interiores y emplea la técnica DSSS en la capa de enlace (Griera, 2008).

✚ Acceso múltiple por escucha de portadora o eliminación de colisiones - CSMA/CA

Acceso múltiple por escucha de portadora o eliminación de colisiones (CSMA/CA, por sus siglas en inglés) es muy empleado en las redes inalámbricas ya que no se puede detectar las colisiones, la figura 15 muestra su funcionamiento (Griera, 2008).

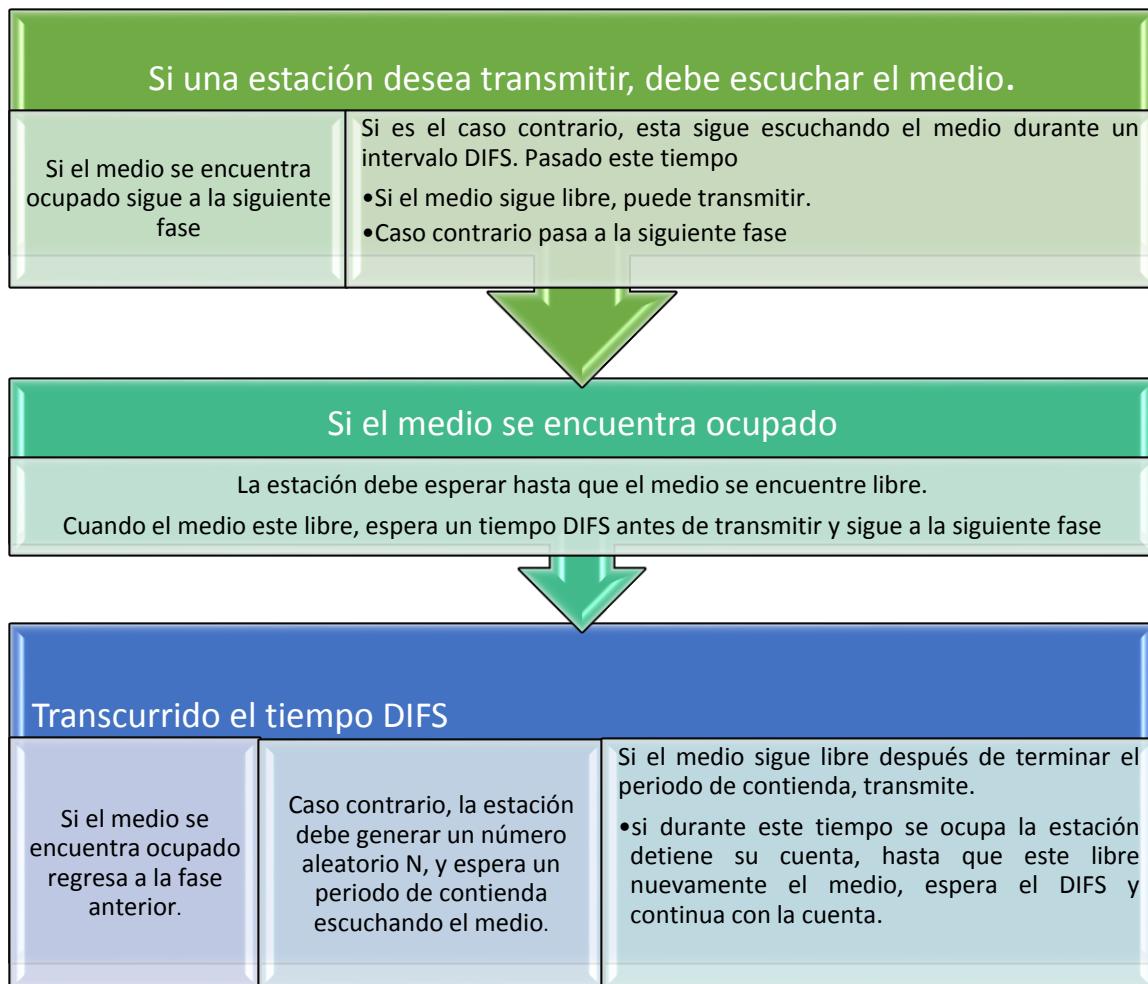


Figura 15 Funcionamiento CSMA/CD

Fuente: (Griera, 2008).

✚ Espectro ensanchado por secuencia directa - DSSS

Espectro ensanchando por secuencia directa (*DSSS*, por sus siglas en inglés) (véase figura 16), es un método empleado para la codificación del canal. En sí genera un patrón redundante para cada uno de los bits que se desea transmitir, a este patrón se lo conoce como chip, de esta manera modula digitalmente la portadora, todo esto con el fin de que la señal resultante de este proceso sea un espectro que tiene una gran semejanza con el ruido, cabe mencionar que mientras mayor sea la longitud del chip, la probabilidad de que los datos enviados sean recuperados es mayor, así que cualquier otro receptor al que no esté destinado la recepción de la señal será considerado como ruido y será rechazada (Kruegle, 2007).

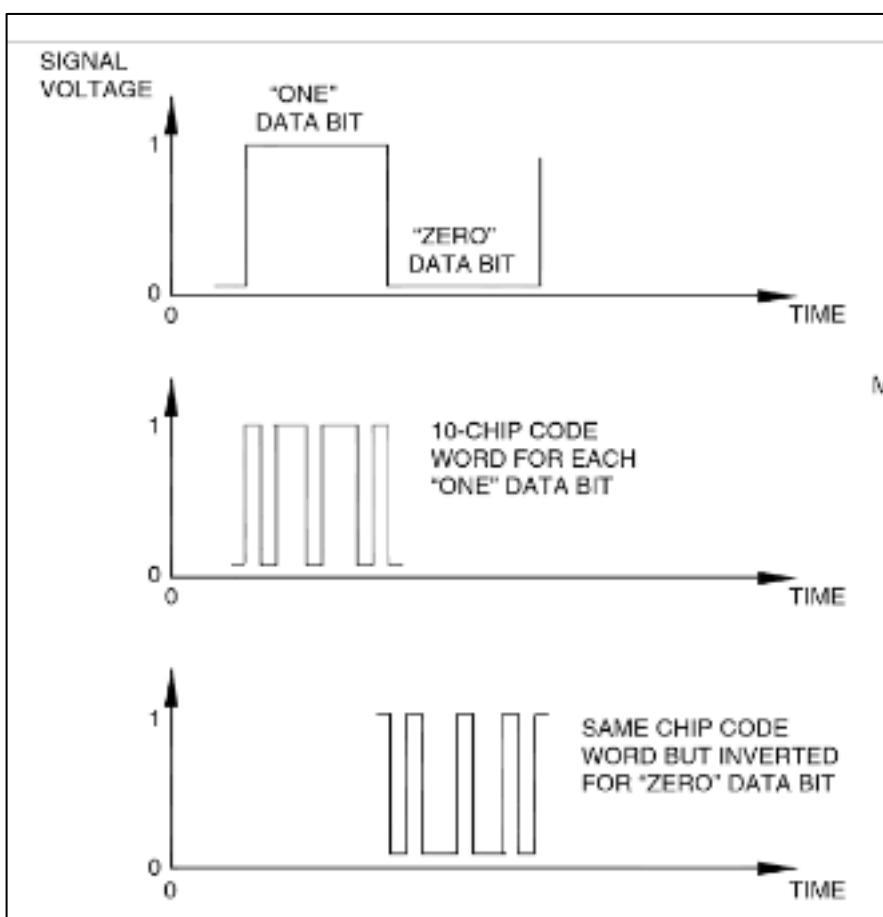


Figura 16 Espectro ensanchado por secuencia directa
Fuente: (Kruegle, 2007)

4.3.5.3 Estándar IEEE 802.11g

El estándar 802.11g, posee una tasa de transferencia de 54Mbps, trabaja en la banda de frecuencia de 2,4 Ghz, compatible con IEEE 802.11b y usa la técnica de modulación DSSS junto con la de OFDM (Pellejero, 2006).

OFDM

Modulación por división ortogonal de frecuencia (OFDM) (véase figura 17), es muy empleada ya que presenta un gran ahorro en el uso del ancho de banda, está formada por varias subportadoras las mismas que son ortogonales entre sí, y usan una fracción del ancho de banda para enviar los datos (Avila, 2015).

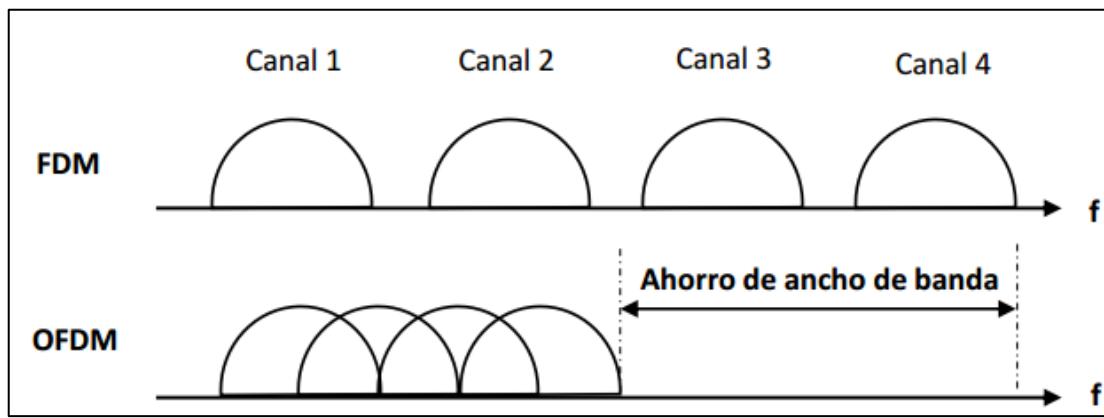


Figura 17 Comparación del uso del ancho de banda entre FDM y OFDM
 Fuente: (Avila, 2015).

4.3.5.4 Estándar IEEE 802.11n

El estándar IEEE 802.11n emplea la tecnología múltiple entradas, múltiples salidas (*MIMO*, por sus siglas en inglés), en otras palabras, el transmisor y el receptor tienen varias antenas. Tiene una velocidad de transmisión de datos aproximada a 600 Mbps a frecuencias de 20 o 40 Ghz.

4.3.5.5 Estándar IEEE 802.11ac

El estándar IEEE 802.11ac actualmente se encuentra en proceso de desarrollo, el mismo que tiene como objetivo proveer a las redes WLAN un rendimiento elevado en la frecuencia de 5Ghz. Teóricamente tendrá un rendimiento de 1Gbps en una WLAN compuesta por varias estaciones y un rendimiento de 500Mbps en un enlace único (Andréu, 2011).

4.4 REDES MALLADAS

4.4.1 Estándar IEEE 802.11s

IEEE 802.11s también conocido redes *mesh*, en este tipo de topología todos los nodos se encuentran interconectados entre sí, lo que conlleva a que existan diferentes caminos para que se pueda transmitir la información de un nodo determinado hacia otro (Conner, 2006).

4.4.2 Arquitectura de la red IEEE 802.11s

La figura 18 muestra la arquitectura de la red *mesh* 802.11s : (Espiga, 2012)

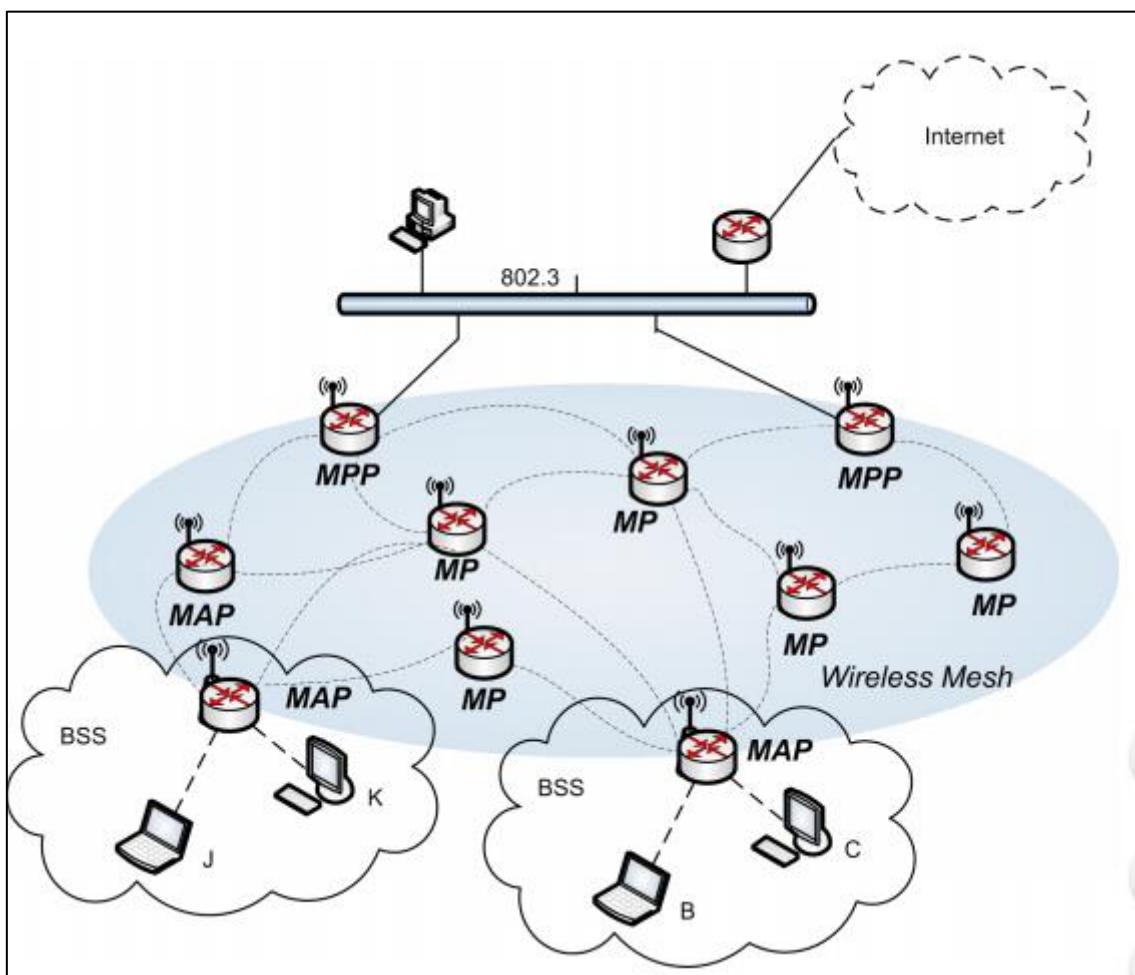


Figura 18 Arquitectura de la Red Mesh

Fuente: (Espiga, 2012).

Se observa que se encuentra formada por varios elementos que serán explicados a continuación:

- *Basic Service Set (BSS, por sus siglas en inglés)*, (véase figura 19) esta se forma a partir de dos estaciones, y si también se encuentra conformado por un AP se denomina BSS de infraestructura.

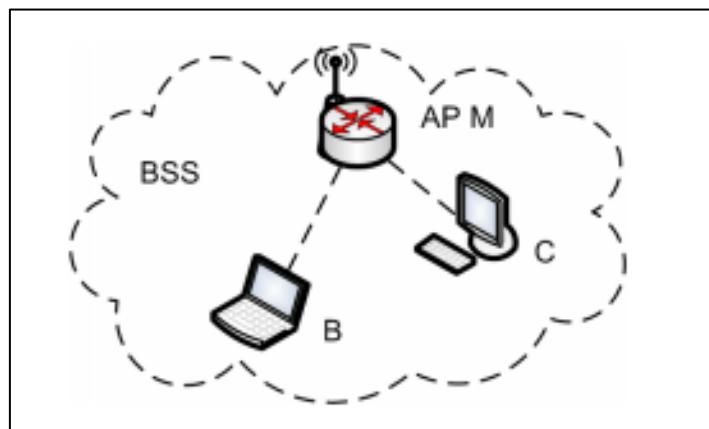


Figura 19 Basic Service Set (BSS)

Fuente: (Espiga, 2012).

- *Mesh Point (MP, por sus siglas en inglés)*, su función es ejecutar los servicios de la red *mesh* inalámbrica.
- *Mesh Access Point (MAP, por sus siglas en inglés)*, este realiza las mismas funciones que un MP pero con funcionalidades de AP.
- *Mesh Portal Point (MPP, por sus siglas en inglés)*, representa un portal de malla, también posee las funciones de un MP y son los encargados de permitir el ingreso y salida de las tramas (Espiga, 2012).

4.4.3 Protocolo Híbrido de Malla Inalámbrica (HWMP)

El estándar IEEE 802.11s tiene definido por defecto el empleo del protocolo híbrido de malla inalámbrica (*HWMP*, por sus siglas en inglés) el mismo que utiliza la métrica Airtimer Link Metric (*ALM*, por sus siglas en inglés).

Este protocolo trabaja a nivel de la capa de enlace es decir la capa 2 del modelo OSI, combina dos tipos de encaminamiento, el modo reactivo y el modo proactivo, este protocolo posee un bajo consumo cuando se descubre rutas empleando el modo reactivo y tiene la eficiencia que brinda el modo proactivo. Solo conoce sus vecinos es decir el conocimiento sobre la topología que tiene es parcial (Municio, s.f.).

Para descubrir la mejor ruta a utilizar cada uno de los nodos, usando las tramas de gestión ayuda para el cálculo de la métrica. Las tramas de gestión son:

- ✚ *Route Request (RREQ, por sus siglas en inglés)*
- ✚ *Route Reply (RREP, por sus siglas en inglés)*
- ✚ *Route Error (RERR, por sus siglas en inglés)*
- ✚ *Root Announcement (RANN, por sus siglas en inglés)*

4.4.3.1 Modo Reactivo

Las rutas son establecidas bajo demanda, en otras palabras, si un nodo origen desea enviar datos hacia un nodo destino y no cuenta con la ruta hacia dicho destino, se inicia el descubrimiento de la ruta (Aguirre, 2014). Emplea el método vector distancia, y para el descubrimiento de la ruta emplea el método RREQ/RREP, usa las direcciones MAC (véase figura 20).

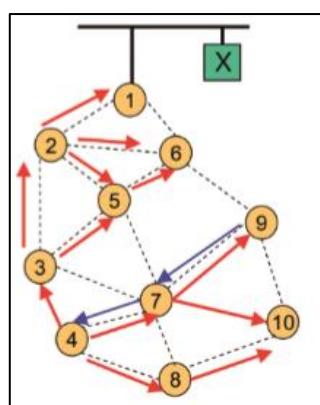


Figura 20 Modo Reactivo

Fuente: (Espiga, 2012)

4.4.3.2 Modo Proactivo

En el modo proactivo (véase figura 21), existe siempre un nodo raíz el mismo que inicia cada cierto intervalo de tiempo la selección de rutas, en otras palabras, siempre existirá una ruta actual entre todos los nodos, esto elimina el tiempo de latencia inicial al enviar los datos. La ruta proactiva se la puede configurar de dos maneras (Blengio, 2008):

- PREQ proactivo: Emplea los mensajes RREQ/RREP, para el establecimiento de la ruta (Blengio, 2008).
- Mecanismo RANN: Emplea los mensajes RRAN para distribuir la información de los próximos saltos hacia el nodo raíz, pero no establecen ruta alguna, la ruta es establecida por mensajes *unicast*, basadas en la información del próximo salto (Blengio, 2008).

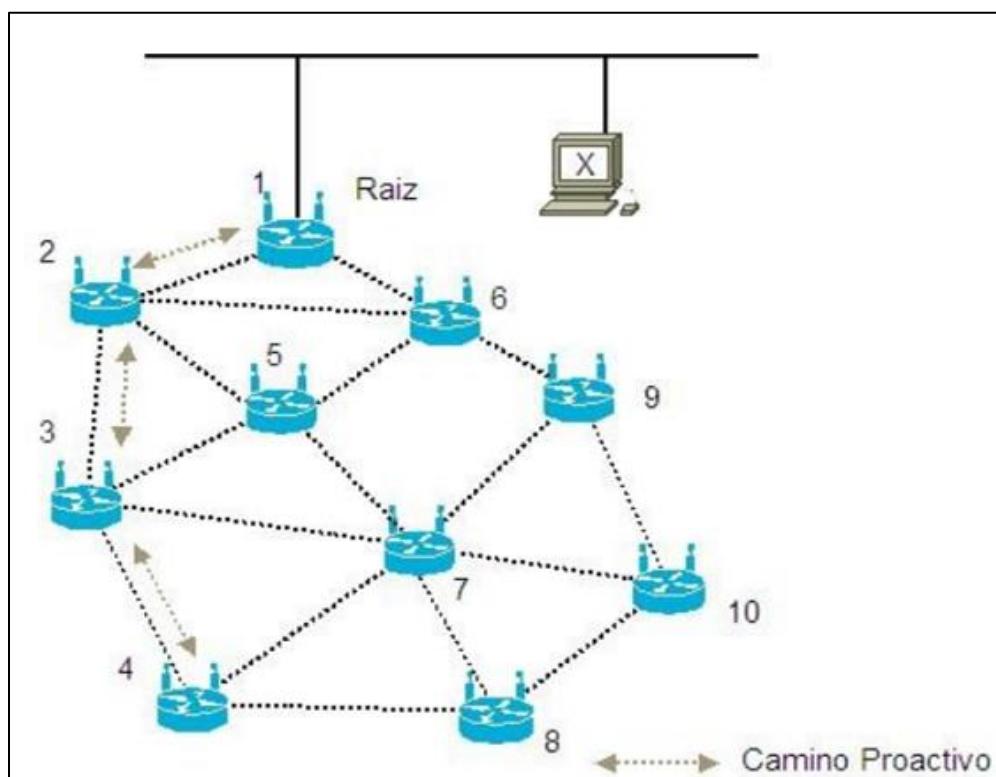


Figura 21 Modo Proactivo
Fuente: (Blengio, 2008).

4.4.3.3 Airtime Link Metric (ALM)

Estima la cantidad de recursos que se utilizan en el canal, tomando en cuenta factores como la tasa de pérdidas de paquetes, el ancho de banda (Espiga, 2012).

$$C_a = \left(O_{ca} + O_p + \frac{B_t}{r} \right) * \left(\frac{1}{1 - e_{pt}} \right) \quad Ec. 1$$

Tabla 3
Parámetros de ALM

	802.11a	802.11 b/g	Descripción
O_{ca}	75 us	335 us	Sobrecarga del acceso al canal
O_p	110 us	364 us	Sobrecarga del protocolo
B_t	8192	8192	Bits en la trama de prueba

Nota: Recuperado de (Espiga, 2012).

4.5 SIMULADOR DE REDES 3 (NS-3)

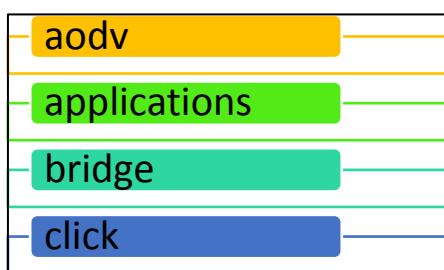
El simulador de redes 3 (*NS-3*, por sus siglas en inglés) es una herramienta que se emplea para simular eventos discretos de una red, especialmente si se necesita simular escenarios en las capas 2 y 4 del modelo de interconexión de sistemas abiertos (*OSI*, por sus siglas en inglés), es el más usado a nivel educativo (Morales, 2013).

NS-3 presenta algunas características que lo distinguen de los otros simuladores, a continuación, se mencionará las de mayor importancia: (Morales, 2013).

- ✚ Diseñado como un conjunto de bibliotecas que se pueden combinar entre sí y también con otras bibliotecas de software externos.
- ✚ Es modular, presenta varios animadores externos para el análisis de datos y herramientas de visualización
- ✚ Desarrollo en software C ++ y / o Python.
- ✚ Mayormente utilizado en sistemas Linux, existe soporte para FreeBSD, Cygwin (para Windows), y se encuentra en desarrollo el apoyo de Visual Studio nativo de Windows (ns-3 project, 2010-2017).

4.5.1 Biblioteca NS-3

La biblioteca NS-3 (véase figura 22) se divide en varios módulos: (NS-3 project, 2017).



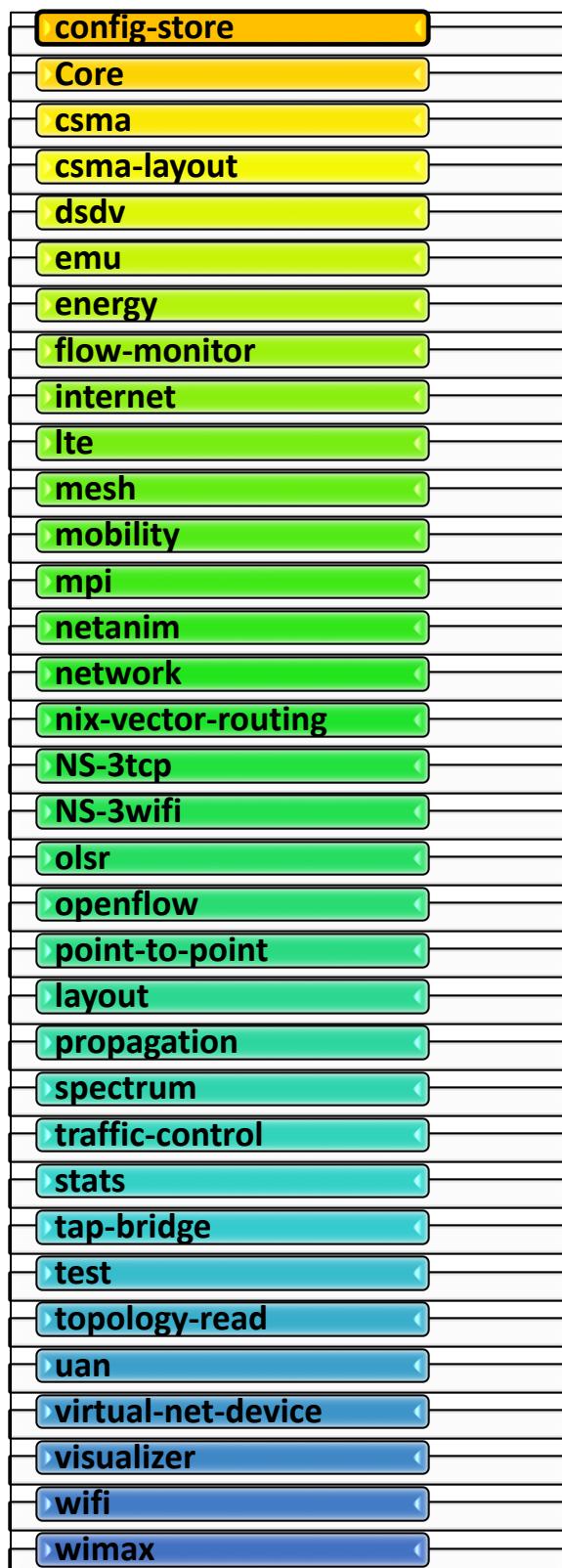


Figura 22 Bibliotecas de NS-3

Fuente: Autor

4.5.2 Modelos de pérdidas de propagación.

A continuación, se listará los distintos modelos de pérdidas de propagación que posee NS-3 para configurar recrear las interferencias en el medio real. (NS-3 project, 2017).

- **Fixed Rss Loss Model:** Este modelo configura la señal que se recibe RSS.
- **Friis Propagation Loss Model:** Este modelo emplea la intensidad que se recibe de las antenas tomando en cuenta la distancia existente entre ellas.
- **Jakes Propagation Loss Model:** Este modelo configura, el número de rayos, de oscilaciones y la frecuencia de Doppler.
- **Nakagami Propagation Loss Model:** Este modelo toma en cuenta las propiedades inalámbricas del canal inalámbrico.
- **Random Propagation Loss Model:** Este modelo se usa cuando un parámetro tiene un rango de valores.
- **Log Distance Propagation Loss Model:** Este modelo calcula la intensidad de la señal que es recibida de la antena.
- **Three Log Distance Propagation Loss Model:** Este modelo es similar a Log Distance Propagation Loss Model, se diferencia en que usa tres campos para la distancia.

4.5.3 Estructura de un script en NS-3

Un script consta de una serie de sentencias que son almacenadas en un archivo de texto, las mismas que son interpretadas una a una en el momento de la ejecución del mismo. En este apartado se explicará a detalle la estructura que se debe seguir para la creación de un nuevo *script* de NS-3 escrito con un lenguaje de programación C++ (NS-3 project, 2017).

Texto informativo

En estas líneas se especifica toda la información necesaria, como que tipo de código fuente, derechos de autor, licencia de distribución del software, así como detalles sobre la topología y el funcionamiento del script.

```
/*-*-Mode: C++; c-file-style:"gnu"; indent-tabs-mode:ni;-*-*/
```

Por lo general los scripts de NS-3 llevan el siguiente texto informativo.

```
/*
* Este programa es software libre; usted puede redistribuirlo y / o modificarlo
* bajo los términos de la Licencia Pública General GNU versión 2 como
* publicada por la Free Software Foundation;
*
* Este programa se distribuye con la esperanza de que sea útil,
* pero SIN NINGUNA GARANTÍA; ni siquiera la garantía implícita de
* COMERCIALIZACIÓN o IDONEIDAD PARA UN PROPÓSITO PARTICULAR. Vea
* la
* Licencia Pública General de GNU para más detalles.
*
* Debería haber recibido una copia de la Licencia Pública General de GNU
* junto con este programa; si no, escriba a la Free Software
* Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 EE.UU.
*/
```

Módulos de inclusión

En todo script antes construir el código se debe comenzar con los módulos de inclusión, los mismos que permiten la escritura del *script* de manera más eficiente.

```
#include "NS-3/netanim-module.h"
#include "NS-3/core-module.h"
#include "NS-3/network-module.h"
#include "NS-3/csma-module.h"
```

Definición de nombres

Se define el uso del espacio de nombres NS-3, esto permite un manejo de forma global de todas las variables.

```
using namespace ns3;
```

Registros

Se emplea para poder tener un documento que contendrá la información de cada uno de los procesos, es decir declara un componente de registro denominado *FirstScriptExample*, este permite activar y desactivar la consola de mensajes de registro por referencia al nombre.

```
NS_LOG_COMPONENT_DEFINE("FirstScriptExample");
```

Función Principal

Aquí se encuentra las secuencias de comandos principales del script, de la misma manera que en un programa con lenguaje C++, se debe declarar la función principal. En esta parte del código se debe realizar la activación de los componentes registro.

```
int main (int argc, char *argv[])
{
    LogComponentEnable ("UdpEchoClientApplication", LOG_LEVEL_INFO);
    LogComponentEnable ("UdpEchoServerApplication", LOG_LEVEL_INFO);
    ....
}
```

Topología

Dentro de la topología se define:

 NodeContainer:

Este es un ayudante de topología que permite de manera sencilla crear, gestionar y acceder a cualquier nodo.

 **PointToPointHelper:**

Es un ayudante en la construcción de la topología, permite conectar los nodos entre sí, en este caso un enlace punto a punto.

 **NetDeviceContainer:**

Es un ayudante de topología que permite agrupar los dispositivos de red.

 **InternetStackHelper:**

Es un ayudante de topología que permite la instalación de protocolos en los nodos.

 **Ipv4AddressHelper:**

Es un ayudante de topología que permite asociar los dispositivos de red con direcciones IP.

 **Aplicación**

Es la capa en la que se configura como se transmiten los datos, por ejemplo, UDP o TCP (véase figura 23).

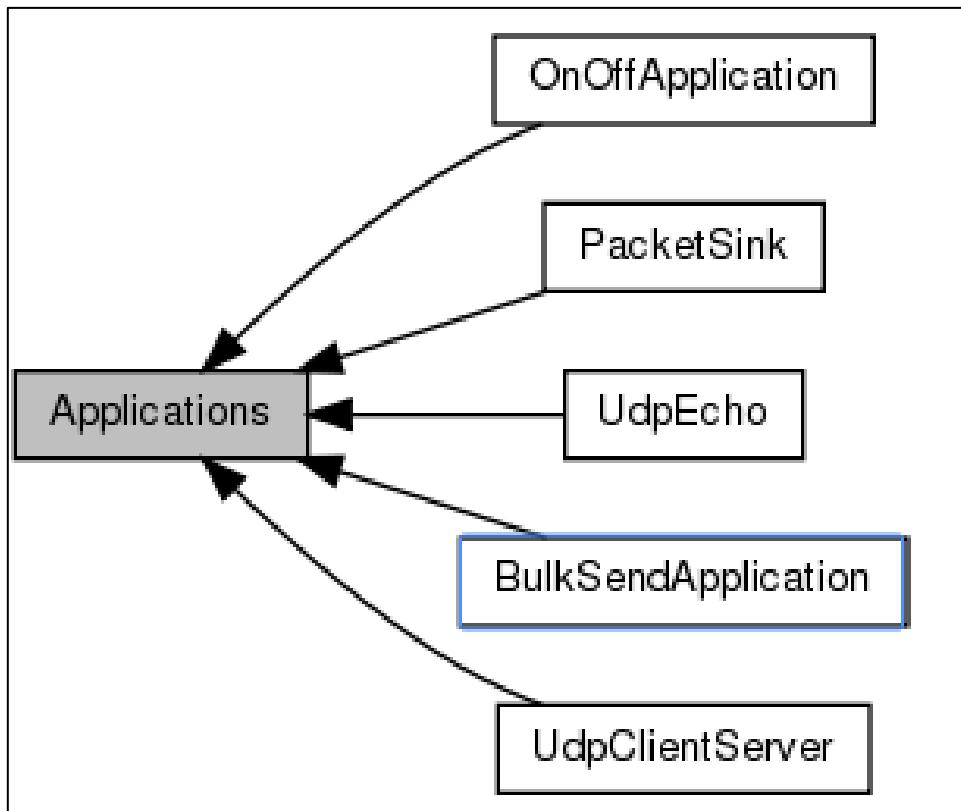


Figura 23 Diagrama de Aplicaciones

Fuente: (ns-3 project, 2017).

💡 Ejecución

Es en sí, la ejecución del programa escrito, luego de ser ejecutado el programa se debe destruir todos los objetos auxiliares creados durante la simulación del mismo.

```
Simulator::Run();
Simulator::Destroy
```

5. MATERIALES Y MÉTODOS

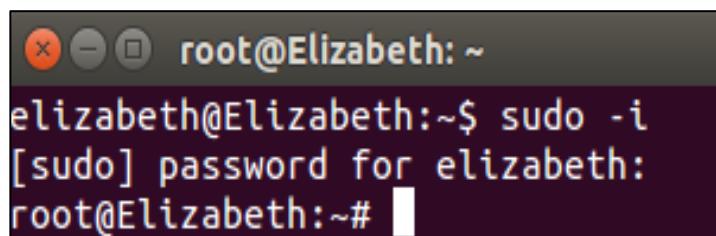
5.1 Descripción de los scripts utilizados.

Antes de la descripción de los scripts utilizados se indica los pasos para instalación del simulador, el módulo de openflow-switch, y el sistema operativo empleado:

5.1.1 Instalación de NS-3.

La instalación se realiza desde el terminal de la PC, la misma posee un sistema operativo Ubuntu, a continuación, los pasos a seguir:

1. Se abre el terminal y se ingresa en modo root, para no tener ningún problema con los permisos al momento de instalar.



A screenshot of a terminal window titled "root@Elizabeth: ~". The window shows the command "elizabeth@Elizabeth:~\$ sudo -i" being typed, followed by "[sudo] password for elizabeth:". The password field is obscured by a redacted area. The prompt then changes to "root@Elizabeth:~#". The window has standard Linux terminal window controls (close, minimize, maximize) at the top left.

Figura 24 Ingreso en modo root

Fuente: Autor

2. Seguidamente se realiza la instalación de los pre-requisitos, en algunos casos se informará que utilizará espacio adicional en el disco y preguntará si desea continuar, se debe escribir “S” y continuar instalando los restantes pre-requisitos.

```

root@Elizabeth: ~
Preparando para desempaquetar .../g++_4%3a4.8.2-1ubuntu6_amd64.deb ...
Desempaquetando g++ (4:4.8.2-1ubuntu6) ...
Procesando disparadores para man-db (2.6.7.1-1ubuntu1) ...
Configurando libstdc++-4.8-dev:amd64 (4.8.4-2ubuntu1~14.04.3) ...
Configurando g++-4.8 (4.8.4-2ubuntu1~14.04.3) ...
Configurando g++ (4:4.8.2-1ubuntu6) ...
update-alternatives: utilizando /usr/bin/g++ para proveer /usr/bin/c++ (c++) en modo automático
W: Duplicate sources.list entry http://dl.google.com/linux/chrome/deb/stable/main
amd64 Packages (/var/lib/apt/lists/dl.google.com_linux_chrome_deb_dists_stable_main_binary-amd64_Packages)
W: Tal vez quiera ejecutar «apt-get update» para corregir estos problemas
root@Elizabeth:~# apt-get install gcc g++ python python-dev
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
g++ ya está en su versión más reciente.
gcc ya está en su versión más reciente.
python ya está en su versión más reciente.
Se instalarán los siguientes paquetes extras:
  libexpat1-dev libpython-dev libpython2.7 libpython2.7-dev
  libpython2.7-minimal libpython2.7-stdlib python2.7 python2.7-dev
  python2.7-minimal
Paquetes sugeridos:
  python2.7-doc binfmt-support
Se instalarán los siguientes paquetes NUEVOS:
  libexpat1-dev libpython-dev libpython2.7-dev python-dev python2.7-dev
Se actualizarán los siguientes paquetes:
  libpython2.7 libpython2.7-minimal libpython2.7-stdlib python2.7
  python2.7-minimal
5 actualizados, 5 se instalarán, 0 para eliminar y 155 no actualizados.
Necesito descargar 27,0 MB de archivos.
Se utilizarán 35,1 MB de espacio de disco adicional después de esta operación.
¿Desea continuar? [S/n] S
Des:1 http://pe.archive.ubuntu.com/ubuntu/ trusty-updates/main python2.7 amd64 2.7.6-8ubuntu0.3 [197 kB]
1% [1 python2.7 168 kB/197 kB 85%]

```

Figura 25 Instalación de pre-requisitos

Fuente: Autor

3. A continuación, se procede a crear y abrir la carpeta en donde se instalará el simulador, posteriormente se descarga el archivo que contiene el simulador y se procede a descomprimirlo.

```

root@Elizabeth:~# cd
root@Elizabeth:~# mkdir ns3
root@Elizabeth:~# cd ns3
root@Elizabeth:~/ns3# wget http://www.nsnam.org/release/ns-allinone-3.22.tar.bz2
--2017-01-22 00:01:37-- http://www.nsnam.org/release/ns-allinone-3.22.tar.bz2
Resolviendo www.nsnam.org (www.nsnam.org)... 143.215.76.161
Conectando con www.nsnam.org (www.nsnam.org)[143.215.76.161]:80... conectado.
Petición HTTP enviada, esperando respuesta... 302 Found
Ubicación: https://www.nsnam.org/release/ns-allinone-3.22.tar.bz2 [siguiente]
--2017-01-22 00:01:38-- https://www.nsnam.org/release/ns-allinone-3.22.tar.bz2
Conectando con www.nsnam.org (www.nsnam.org)[143.215.76.161]:443... conectado.
Petición HTTP enviada, esperando respuesta... 200 OK
Longitud: 24558182 (23M) [application/x-bzip2]
Grabando a: "ns-allinone-3.22.tar.bz2"

100%[=====] 24.558.182 290KB/s en 2m 12s

2017-01-22 00:03:51 (181 KB/s) - "ns-allinone-3.22.tar.bz2" guardado [24558182/24558182]

root@Elizabeth:~/ns3# tar xjf ns-allinone-3.22.tar.bz2
root@Elizabeth:~/ns3#

```

Figura 26 Descarga y descompresión del archivo que contiene a NS-3

Fuente: Autor

- Se abre la carpeta “ns-allinone-3.22” y se construye NS-3, con el comando siguiente: ./build.py --enable-examples --enable-tests.

Aparece lo siguiente en el terminal:

```

:851/2426] command $(PYTHON) bindings/python/ns3modulegen-modular.py src/wave/bindings/modulegen__gcc_LP64.py -> build/src/wave/bindings/ns3modulegen.log
:852/2426] command $(PYTHON) bindings/python/ns3modulegen-modular.py src/wifi/bindings/modulegen__gcc_LP64.py src/wifi/bindings/modulegen_cu_tutorialgen.py -> build/src/wifi/bindings/ns3module.cc build/src/wifi/bindings/ns3module.h build/src/wifi/bindings/ns3modulegen.log
:853/2426] command $(PYTHON) bindings/python/ns3modulegen-modular.py src/wimax/bindings/modulegen__gcc_LP64.py -> build/src/wimax/bindings/ns3modulegen.log
:854/2426] command $(PYTHON) bindings/python/ns3modulegen-modular.py src/utils/bindings/modulegen__gcc_LP64.py -> build/src/utils/bindings/ns3modulegen.log
:855/2426] cxx: utils/test-runner.cc -> build/utils/test-runner.cc.1.o
:856/2426] cxx: utils/bench-simulator.cc -> build/utils/bench-simulator.cc.2.o
:857/2426] cxx: utils/bench-packets.cc -> build/utils/bench-packets.cc.3.o
:858/2426] cxx: examples/introduction-demos/intercepted-doxygen.cc -> build/examples/introduction-demos/intercepted-doxygen.cc.4.o
:859/2426] cxx: scratch/openflow-switch.cc -> build/scratch/openflow-switch.cc.2.o
:860/2426] cxx: examples/udp-client-server/udp-trace-client-server.cc -> build/examples/udp-client-server/udp-trace-client-server.cc.2.o
:861/2426] cxx: examples/udp-client-server/udp-client-server.cc -> build/examples/udp-client-server/udp-client-server.cc.1.o
:862/2426] cxx: examples/ipv6/lossy-routing.ipv6.cc -> build/examples/ipv6/lossy-routing.ipv6.cc.0.o
:863/2426] cxx: examples/ipv6/fragmentation-ipv6.cc -> build/examples/ipv6/fragmentation-ipv6.cc.0.o
:864/2426] cxx: examples/ipv6/test.ipv6.cc -> build/examples/ipv6/test.ipv6.cc.0.o
:865/2426] cxx: examples/ipv6/radvd-two-prefix.cc -> build/examples/ipv6/radvd-two-prefix.cc.4.o
:866/2426] cxx: examples/ipv6/forwarding-tutorial/forwarding.ipv6.cc -> build/examples/ipv6/forwarding-tutorial/forwarding.ipv6.cc.0.o
:867/2426] cxx: examples/ipv6/cmpv6-redirect.cc -> build/examples/ipv6/cmpv6-redirect.cc.1.o
:868/2426] cxx: examples/tutorial/seventh.cc -> build/examples/tutorial/seventh.cc.8.o
:869/2426] cxx: examples/tutorial/fifth.cc -> build/examples/tutorial/fifth.cc.8.o
:870/2426] cxx: examples/tutorial/fourth.cc -> build/examples/tutorial/fourth.cc.5.o
:871/2426] cxx: examples/tutorial/first.cc -> build/examples/tutorial/first.cc.1.o
:872/2426] cxx: examples/wireless/simple-simulations/wifi-example-sim.cc -> build/examples/wireless/simple-simulations/wifi-example-sim.cc.1.o
:873/2426] cxx: examples/wireless/rate-adaptation-distance.cc -> build/examples/wireless/rate-adaptation-distance.cc.21.o
:874/2426] cxx: examples/wireless/power-adaptation-distance.cc -> build/examples/wireless/power-adaptation-distance.cc.19.o
:875/2426] cxx: examples/wireless/wifi-sleep.cc -> build/examples/wireless/wifi-sleep.cc.18.o
:876/2426] cxx: examples/wireless/wifi-hidden-terminal.cc -> build/examples/wireless/wifi-hidden-terminal.cc.16.o
:877/2426] cxx: examples/wireless/odm-ht-validation.cc -> build/examples/wireless/odm-ht-validation.cc.14.o
:878/2426] cxx: examples/wireless/odm-validation.cc -> build/examples/wireless/odm-validation.cc.13.o
:880/2426] cxx: examples/wireless/wifi-simple-adhoc-grid.cc -> build/examples/wireless/wifi-simple-adhoc-grid.cc.9.o
:881/2426] cxx: examples/wireless/wifi-wired-bridging.cc -> build/examples/wireless/wifi-wired-bridging.cc.5.o
:882/2426] cxx: examples/wireless/simple-mdu-aggregation.cc -> build/examples/wireless/simple-mdu-aggregation.cc.6.o
:883/2426] cxx: examples/wireless/wifi-adhoc.cc -> build/examples/wireless/wifi-adhoc.cc.2.o
:885/2426] cxx: examples/routing/mixed-global-routing.cc -> build/examples/routing/mixed-global-routing.cc.7.o
:886/2426] cxx: examples/routing/simple-global-routing.cc -> build/examples/routing/simple-global-routing.cc.5.o
:887/2426] cxx: examples/routing/static-routing-stash32.cc -> build/examples/routing/static-routing-stash32.cc.2.o

```

Figura 27 Construcción de NS-3

Fuente: Autor

Una vez terminado el proceso y si la construcción es exitosa, entonces se mostrará el siguiente mensaje: “*“Build finished successfully”*”

```

Waf: Leaving directory '/root/ns3/ns-allinone-3.22/ns-3.22/build'
'build' finished successfully (1h3m11.433s)

Modules built:
antenna           aody          applications
bridge            buildings      config-store
core              csma          csma-layout
dsdv              dsr           energy
fd-net-device     flow-monitor internet
lr-wpan           lte            mesh
mobility          mpi            netanim (no Python)
network           nix-vector-routing olsr
point-to-point    point-to-point-layout propagation
sixlowpan         spectrum      stats
tap-bridge        test (no Python) topology-read
uan               virtual-net-device visualizer
wave              wifi           wimax

Modules not built (see ns-3 tutorial for explanation):
brite              click          openflow

```

Figura 28 Mensaje de construcción exitosa

Fuente: Autor

5. Se abre la carpeta “ ns-3.22 ” y se configura WAF (herramienta de construcción) con el comando: ./waf -d debug --enable-examples --enable-tests configure.

Se comprueba la correcta instalación del programa corriendo el siguiente ejemplo: *./waf --run first*

Apareciendo lo siguiente:

```
At time 2s client sent 1024 bytes to 10.1.1.2 port 9  
At time 2.00369s server received 1024 bytes from 10.1.1.1 port 49153  
At time 2.00369s server sent 1024 bytes to 10.1.1.1 port 49153  
At time 2.00737s client received 1024 bytes from 10.1.1.2 port 9
```

```
root@elizabeth:~/ns3/ns-allinone-3.22# cd ns-3.22/  
root@elizabeth:~/ns3/ns-allinone-3.22/ns-3.22# ./waf --run first  
Waf: Entering directory '/root/ns3/ns-allinone-3.22/ns-3.22/build'  
Waf: Leaving directory '/root/ns3/ns-allinone-3.22/ns-3.22/build'  
'build' finished successfully (16.972s)  
At time 2s client sent 1024 bytes to 10.1.1.2 port 9  
At time 2.00369s server received 1024 bytes from 10.1.1.1 port 49153  
At time 2.00369s server sent 1024 bytes to 10.1.1.1 port 49153  
At time 2.00737s client received 1024 bytes from 10.1.1.2 port 9  
root@elizabeth:~/ns3/ns-allinone-3.22/ns-3.22# █
```

Figura 29 Resultado del primer ejemplo

Fuente: Autor

5.1.2 Instalación del módulo Openflow-Shiwch.

El módulo *Openflow-Switch* no viene por defecto en la instalación de NS-3, este debe ser instalado si se desea realizar simulaciones en donde se ve involucrado un *switch* y debido a que más adelante se empleará una topología tipo estrella en el estándar IEEE 802.3 es necesario hacerlo.:

Las siguientes líneas de código nos permitirán realizar la instalación:

1. Se abre el terminal y se ingresa en modo root a la carpeta donde se encuentra instalado el simulador para descargar el modulo, y descomprimirlo:

```
root@Elizabeth:~/ns3/ns-allinone-3.22/ns-3.22# cd /root/ns3/ns-allinone-3.22/ns-3.22
root@Elizabeth:~/ns3/ns-allinone-3.22/ns-3.22# export BOOSTDIR=/home/non-privileged-user/boost
root@Elizabeth:~/ns3/ns-allinone-3.22/ns-3.22# wget http://downloads.sourceforge.net/project/boost/boost/1.57.0/boost_1_57_0.tar.bz2
--2017-01-22 01:08:04--  http://downloads.sourceforge.net/project/boost/boost/1.57.0/boost_1_57_0.tar.bz2
Resolviendo downloads.sourceforge.net (downloads.sourceforge.net)... 216.34.181.59
Conectando con downloads.sourceforge.net (downloads.sourceforge.net)[216.34.181.59]:80... conectado.
Petición HTTP enviada, esperando respuesta... 302 Found
Ubicación: https://ufpr.dl.sourceforge.net/project/boost/boost/1.57.0/boost_1_57_0.tar.bz2 [siguiente]
--2017-01-22 01:08:05--  https://ufpr.dl.sourceforge.net/project/boost/boost/1.57.0/boost_1_57_0.tar.bz2
Resolviendo ufpr.dl.sourceforge.net (ufpr.dl.sourceforge.net)... 200.236.31.2, 2801:82:80ff:8000::3
Conectando con ufpr.dl.sourceforge.net (ufpr.dl.sourceforge.net)[200.236.31.2]:443...
.. conectado.
Petición HTTP enviada, esperando respuesta... 200 OK
Longitud: 60821561 (58M) [application/octet-stream]
Grabando a: "boost_1_57_0.tar.bz2"

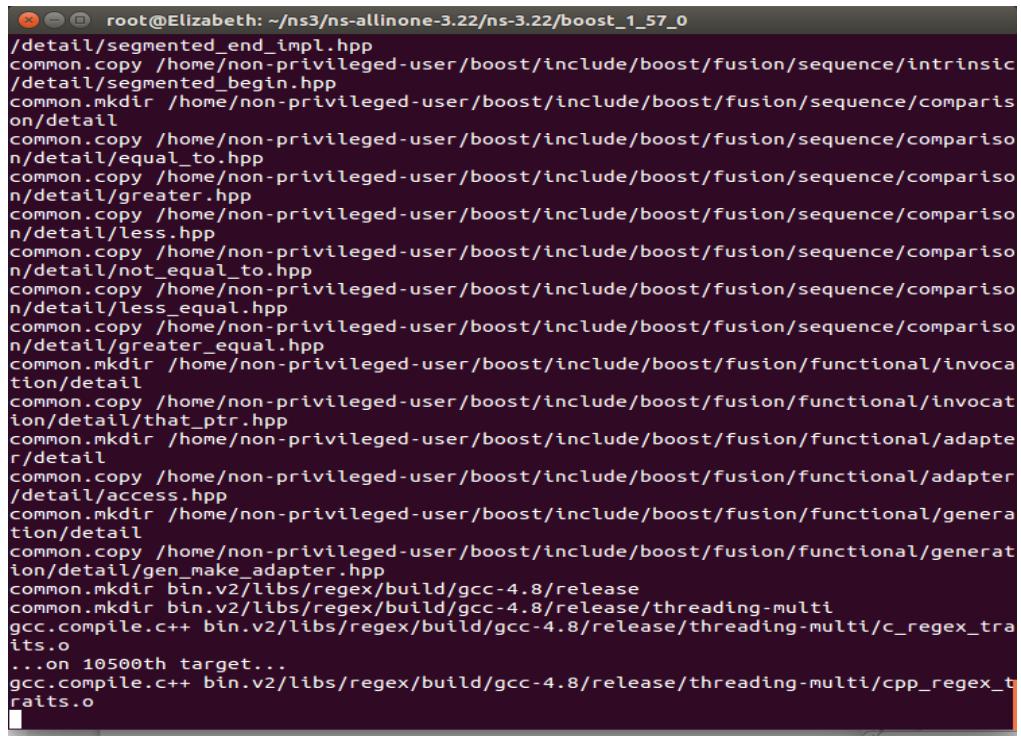
100%[=====] 60.821.561  259KB/s  en 4m 22s
2017-01-22 01:12:29 (226 KB/s) - "boost_1_57_0.tar.bz2" guardado [60821561/60821561]

root@Elizabeth:~/ns3/ns-allinone-3.22/ns-3.22# tar jxf boost_1_57_0.tar.bz2
root@Elizabeth:~/ns3/ns-allinone-3.22/ns-3.22#
```

Figura 30 Descarga y descompresión del módulo.

Fuente: Autor

2. Se ingresa a la carpeta boost_1_57_0 que se crea al descomprimir el módulo y se procede a la instalación.



```

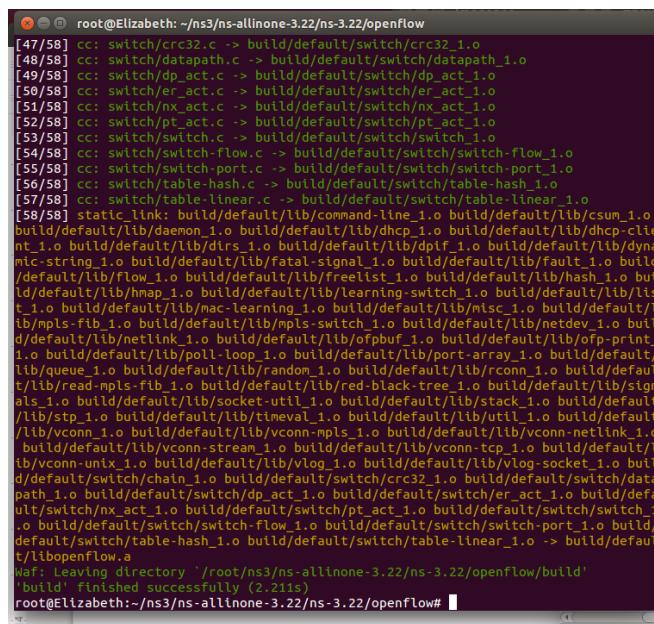
root@Elizabeth: ~/ns3/ns-allinone-3.22/ns-3.22/boost_1_57_0
/d/detail/segmented_endImpl.hpp
common.copy /home/non-privileged-user/boost/include/boost/fusion/sequence/intrinsic
/d/detail/segmented_begin.hpp
common.mkdir /home/non-privileged-user/boost/include/boost/fusion/sequence/comparis
on/detail
common.copy /home/non-privileged-user/boost/include/boost/fusion/sequence/compariso
n/detail/equal_to.hpp
common.copy /home/non-privileged-user/boost/include/boost/fusion/sequence/compariso
n/detail/greater.hpp
common.copy /home/non-privileged-user/boost/include/boost/fusion/sequence/compariso
n/detail/less.hpp
common.copy /home/non-privileged-user/boost/include/boost/fusion/sequence/compariso
n/detail/not_equal_to.hpp
common.copy /home/non-privileged-user/boost/include/boost/fusion/sequence/compariso
n/detail/less_equal.hpp
common.copy /home/non-privileged-user/boost/include/boost/fusion/sequence/compariso
n/detail/greater_equal.hpp
common.mkdir /home/non-privileged-user/boost/include/boost/fusion/functional/invoca
tion/detail
common.copy /home/non-privileged-user/boost/include/boost/fusion/functional/invocat
ion/detail/that_ptr.hpp
common.mkdir /home/non-privileged-user/boost/include/boost/fusion/functional/adapte
r/detail
common.copy /home/non-privileged-user/boost/include/boost/fusion/functional/adapter
/d/detail/access.hpp
common.mkdir /home/non-privileged-user/boost/include/boost/fusion/functional/genera
tion/detail
common.copy /home/non-privileged-user/boost/include/boost/fusion/functional/generat
ion/detail/gen_make_adapter.hpp
common.mkdir bin.v2/libs/regex/build/gcc-4.8/release
common.mkdir bin.v2/libs/regex/build/gcc-4.8/release/threading-multi
gcc.compile.c++ bin.v2/libs/regex/build/gcc-4.8/release/threading-multi/c_regex_traits.o
...on 10500th target...
gcc.compile.c++ bin.v2/libs/regex/build/gcc-4.8/release/threading-multi/cpp_regex_t
raits.o

```

Figura 31 Instalación del módulo.

Fuente: Autor

3. Una vez instalado se procede a ejecutar la configuración y construcción del módulo. Si la construcción es exitosa, entonces se mostrará el siguiente mensaje: "Build finished successfully"



```

root@Elizabeth: ~/ns3/ns-allinone-3.22/ns-3.22/openflow
[47/58] cc: switch/crc32.c -> build/default/switch/crc32_1.o
[48/58] cc: switch/datapath.c -> build/default/switch/datapath_1.o
[49/58] cc: switch/dp_act.c -> build/default/switch/dp_act_1.o
[50/58] cc: switch/er_act.c -> build/default/switch/er_act_1.o
[51/58] cc: switch/nx_act.c -> build/default/switch/nx_act_1.o
[52/58] cc: switch/pt_act.c -> build/default/switch/pt_act_1.o
[53/58] cc: switch/switch.c -> build/default/switch/switch_1.o
[54/58] cc: switch/switch-flow.c -> build/default/switch/switch-flow_1.o
[55/58] cc: switch/switch-port.c -> build/default/switch/switch_port_1.o
[56/58] cc: switch/table-hash.c -> build/default/switch/table-hash_1.o
[57/58] cc: switch/table-linear.c -> build/default/switch/table-linear_1.o
[58/58] static_link: build/default/lib/command-line_1.o build/default/lib/csum_1.o
build/default/lib/daemon_1.o build/default/lib/dhcp_1.o build/default/lib/dhcp-clie
nt_1.o build/default/lib/dirs_1.o build/default/lib/dpif_1.o build/default/lib/dyna
mic-string_1.o build/default/lib/fatal-signal_1.o build/default/lib/fault_1.o build
/default/lib/flow_1.o build/default/lib/freelist_1.o build/default/lib/hash_1.o bu
ild/default/lib/hmap_1.o build/default/lib/learning-switch_1.o build/default/lib/lis
t_1.o build/default/lib/mac-learning_1.o build/default/lib/misc_1.o build/default/l
ib/mpls-fib_1.o build/default/lib/ncls-switch_1.o build/default/lib/netdev_1.o bu
ild/default/lib/netlink_1.o build/default/lib/ofpbuf_1.o build/default/lib/ofp-print
_1.o build/default/lib/poll-loop_1.o build/default/lib/port-array_1.o build/default/
lib/queue_1.o build/default/lib/random_1.o build/default/lib/rconi_1.o build/defau
lt/lib/read-mpls-fib_1.o build/default/lib/red-black-tree_1.o build/default/lib/sign
als_1.o build/default/lib/socket-util_1.o build/default/lib/stack_1.o build/defaul
t/lib/tcp_1.o build/default/lib/timerval_1.o build/default/lib/util_1.o build/defaul
t/lib/vconn_1.o build/default/lib/vconn-mpls_1.o build/default/lib/vconn-netlink_1.o
build/default/lib/vconn-stream_1.o build/default/lib/vconn-tcp_1.o build/default/l
ib/vconn-unix_1.o build/default/lib/vlog_1.o build/default/lib/vlog-socket_1.o buil
d/default/switch/chain_1.o build/default/switch/crc32_1.o build/default/switch/data
path_1.o build/default/switch/dp_act_1.o build/default/switch/switch_1.o build/defau
lt/switch/nx_act_1.o build/default/switch/pt_act_1.o build/default/switch/switch_1
.o build/default/switch/switch-flow_1.o build/default/switch/switch-port_1.o build/b
uild/default/switch/table-hash_1.o build/default/switch/table-linear_1.o -> build/def
ault/libopenflow.a
War: Leaving directory '/root/ns3/ns-allinone-3.22/ns-3.22/openflow/build'
'build' finished successfully (2.211s)
root@Elizabeth:~/ns3/ns-allinone-3.22/ns-3.22/openflow# 

```

Figura 32 Construcción finalizada

Fuente: Autor

4. Se regresa a la carpeta de NS-3 para hacer que NS-3 reconozca a *openflow* como una de sus librerías. Una vez finalizado debe aparecer lo siguiente en el terminal.



```
NS-3 OpenFlow Integration : not enabled (OpenFlow not enabled (see option --with-openflow))
```

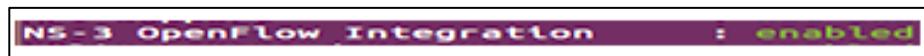
Figura 33 Mensaje de OpenFlow como no habilitado

Fuente: Autor

Como se observa *Openflow* no se encuentra habilitado, a continuación se procederá a habilitarlo, para esto se debe abrir la carpeta: root/NS-3/ns-allinone-3.22/ns-3.22/openflow/include y copiar la carpeta “*openflow*” que se encuentra dentro de la carpeta *include* abierta anteriormente.

5. Seguidamente se abre la carpeta: root/NS-3/ns-allinone-3.22/ns-3.22/build, pegar la carpeta “*openflow*” dentro de la carpeta *build* abierta y volver a configurar y construir la carpeta *openflow*.

Una vez terminado, ya se debe encontrar habilitado el modulo y debe aparecer lo siguiente:



```
NS-3 OpenFlow Integration : enabled
```

Figura 34 Módulo activado.

Fuente: Autor

5.1.3 Estándar IEEE 802.3 – Ethernet.

5.1.3.1 Topología tipo bus.



Figura 35 Esquema de la Topología tipo Bus

Fuente: Autor

La figura 35 muestra una red conformada por 11 nodos representados desde n0 hasta n10, los nodos n0 y n1 se encuentran configurados como una red punto a punto y desde n1 a n9 se encuentra configurada como una red tipo bus con el método de acceso CSMA.

El código empleado se detalla a continuación:

1. Se carga los módulos de archivos de inclusión

```
#include "NS-3/netanim-module.h"  
.....  
#include "NS-3/ipv4-global-routing-helper.h"
```

2. Luego de los módulos de inclusión, se sigue con el espacio de nombres NS-3 el mismo que contiene las declaraciones de NS-3, este se lo representa con:

Using namespace ns3

3. Se define el componente de registro con:

```
NS_LOG_COMPONENT_DEFINE(TOPOLOGIA_BUS);
```

Cabe mencionar que se puede adherir otras variantes de componentes de registro como:

NS LOG ERROR para el registro de mensajes de error, *NS LOG WARN*: para el registro de mensajes de advertencia, *NS LOG DEBUG* para el registro de mensajes de depuración ad-hoc relativamente raros, *NS LOG INFO* para el registro de mensajes informativos sobre el progreso del programa, *NS LOG FUNCTION* que registra un mensaje describiendo cada función llamada, *NS LOG LOGIC* es un registro de mensajes que describen el flujo lógico dentro de una función, *NS LOG ALL* representa el registro de todos los datos.

4. Los mensajes log no se encuentran habilitados por defecto, las líneas de código mostradas a continuación muestran como se los habilita.

```
LogComponentEnable ( " UdpEchoClientApplication " , LOG LEVEL INFO );  
LogComponentEnable ( " UdpEchoServerApplication " , LOG LEVEL INFO );
```

5. Se define el número de dispositivos por los que estará conformada la red CSMA, este puede variar modificando el valor de nodosCsma.

```
uint32_t nodosCsma = 9;
```

6. Se procede a declarar un contenedor de nodos con el nombre de Nodosp2p, este es un vector que guarda punteros a objetos de la clase nodo.

```
NodeContainer Nodosp2p;
```

7. Se crea los dos nodos que se encontrarán dentro del contenedor de nodos. Son dos nodos ya que se realizará un enlace punto a punto con ellos.

```
Nodosp2p.Create ( 2 );
```

8. Se declara un contenedor de nodos con el nombre Nodoscsma, además uno de los Nodosp2p formará parte de los Nodoscsma.

```
NodeContainer Nodoscsma;  
Nodoscsma.Add (Nodosp2p.Get (1));
```

9. Se declara el objeto “*Helper*” correspondiente a las redes punto a punto, este permite emplear métodos que facilitan la creación de redes punto a punto.

```
PointToPointHelper pointToPoint;
```

Tanto la velocidad de los datos como el *delay* del canal se pueden modificar en este caso se estableció que los dispositivos transmitan a una tasa de 10Mbps y el canal tendrá un *delay* de 2ms. Este valor de velocidad de transferencia corresponde a una de las velocidades establecidas en el estándar IEEE802.3. Se lo modifica de la siguiente manera:

```
pointToPoint.SetDeviceAttribute ("DataRate", StringValue ("10Mbps"));  
pointToPoint.SetChannelAttribute ("Delay", StringValue ("2ms"));
```

10. Se crea una instancia para hacer el seguimiento de los dispositivos que componen la red punto a punto, luego se debe instalar estos dispositivos en los Nodosp2p.

```
NetDeviceContainer Dispositivosp2p;  
Dispositivosp2p = pointToPoint.Install (Nodosp2p);
```

Se puede utilizar la misma velocidad de transmisión como el *delay* configurados para los dispositivos Csma de la red punto a punto, si ese fuera el caso solo se necesita colocar la siguiente línea y se procede al siguiente paso:

```
CsmaHelper csma;
```

Pero si desea variar tanto la velocidad de transmisión como el *delay* se puede configurar solo en el canal ya que una red CSMA no permite que en una misma red los dispositivos trabajen a velocidades distintas. En este caso se estableció una velocidad de 100Mbps que también corresponde a una de las velocidades establecidas en IEEE802.3 y un *delay* de 1ns escogido aleatoriamente.

```
csma.SetChannelAttribute ("DataRate", StringValue ("100Mbps"));  
csma.SetChannelAttribute ("Delay", TimeValue (NanoSeconds (1)));
```

Para la simulación realizada se tomó en cuenta solo la primera configuración.

11. Se crea una instancia para hacer el seguimiento de los dispositivos que componen la red CSMA e instalar estos dispositivos en los Nodoscsma.

```
NetDeviceContainer Dispositivoscsma;  
Dispositivoscsma = csma.Install (Nodoscsma);
```

Se procede a establecer la pila de protocolos ya que se encuentran creados los nodos, dispositivos y canales. Estos solo se instalan en la pila, en uno de los Nodosp2p ya que anteriormente el nodo faltante fue incluido dentro de los Nodoscsma.

```
InternetStackHelper stack;  
stack.Install (Nodosp2p.Get (0));  
stack.Install (Nodoscsma);
```

12. Se asigna las direcciones IP a las interfaces de los dispositivos de la red punto a punto.

```

Ipv4AddressHelper ipv4;
ipv4.SetBase ("192.168.1.0", "255.255.255.0");
Ipv4InterfaceContainer Interfacesp2p;
Interfacesp2p = ipv4.Assign (Dispositivosp2p).

```

13. Se asigna las direcciones IP a las interfaces de los dispositivos de la red CSMA

```

ipv4.SetBase ("192.168.2.0", "255.255.255.0");
Ipv4InterfaceContainer Interfacescsma;
Interfacescsma = ipv4.Assign (Dispositivoscsm).

```

14. Se instala el servidor en uno de los nodos Csma, y se define el número de puerto del servidor.

```

UdpEchoServerHelper echoServer (9); // Número de puerto del servidor

```

15. Se instala el cliente en el primero nodo de la red punto a punto es decir en el que solo pertenece a la red punto a punto, ya que el otro nodo fue incluido en la red Csma.

```

UdpEchoClientHelper echoClient (Interfacescsma.GetAddress (nCsma), 9);
ApplicationContainer clientApps = echoClient.Install (Nodosp2p.Get (0));

```

En este caso se puede establecer los valores para los diversos atributos como la cantidad máxima de paquetes, el intervalo en que son enviados y el tamaño del mismo.

```

echoClient.SetAttribute ("MaxPackets", UintegerValue (1));
echoClient.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
echoClient.SetAttribute ("PacketSize", UintegerValue (1024));

```

16. Se debe crear el enrutamiento con:

```
Ipv4GlobalRoutingHelper::PopulateRoutingTables();
```

17. Se habilita el rastreo.

```
AnimationInterface anim ("topologíabus.xml");
```

```
.....
```

```
csma.EnablePcapAll ("topologiabus", true);
```

18. Se ejecuta la simulación.

```
Simulator::Run();
```

5.1.3.2 Topología tipo estrella.

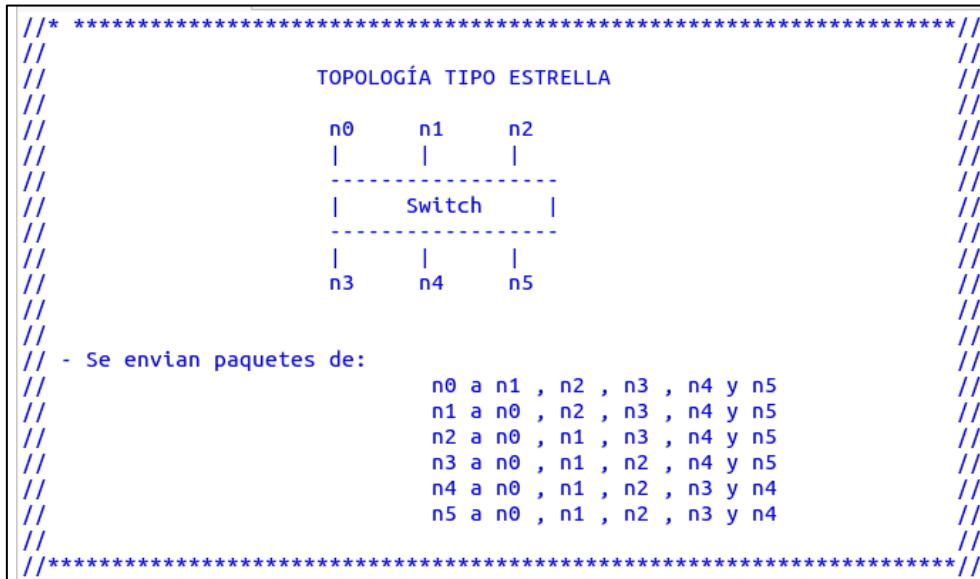


Figura 36 Topología tipo estrella

Fuente: Autor

La figura 36 muestra una red con una topología tipo estrella, esta se encuentra conformada por 6 nodos, etiquetados desde n0 hasta n5 respectivamente.

Se debe seguir la misma estructura que con la topología tipo bus, el código empleado se detalla a continuación:

1. Se debe cargar los archivos de inclusión, en este script se usa el módulo *openflow*, para simular un *switch*.

```
#include <iostream>
.....
#include "NS-3/openflow-module.h"
```

2. Luego de los módulos de inclusión sigue el espacio de nombres ns-3 el mismo que contiene las declaraciones de ns-3 esto se lo representa con:

Using namespace ns3

3. Se define el componente de registro con:

```
NS_LOG_COMPONENT_DEFINE ("OpenFlowCsmaSwitchExample");
```

4. Se declaran la variable para realizar un seguimiento

```
bool use_drop = false;
```

5. Se habilita los Logs.

```
LogComponentEnable ("OpenFlowCsmaSwitchExample",  
LOG_LEVEL_INFO);  
LogComponentEnable ("OpenFlowInterface", LOG_LEVEL_INFO);  
LogComponentEnable ("OpenFlowSwitchNetDevice",  
LOG_LEVEL_INFO);
```

6. Se crean los nodos que se conectarán al switch, en este caso son 6 los nodos.

```
NS_LOG_INFO ("Creación de nodos.");  
NodeContainer terminales;  
terminales.Create (6);
```

7. Se crea un contenedor para el switch

```
NodeContainer Switchcsma;  
Switchcsma.Create (1);
```

8. Se construye la topología con la ayuda del Helper, igual que en la topología tipo bus se puede modificar la velocidad de transmisión de los datos como el delay.

```
CsmaHelper csma;
```

9. Se crea la conexión entre los nodos y el *switch* empleando un ciclo for.

```
for (int i = 0; i < 6; i++)  
{  
    NetDeviceContainer link = csma.Install (NodeContainer (terminals.Get (i),  
    csmaSwitch));
```

10. Se crea el *netdevice* del *switch* para poder enviar paquetes a cada nodo, se para esto se usa el *OpenFlowSwitchHelper*.

```
OpenFlowSwitchHelper swtch;  
.....  
swtch.Install (switchNode, switchDevices, controller);
```

11. Se procede a crear la pila de protocolos.

```
InternetStackHelper internet;  
internet.Install (terminales)
```

12. Se asigna las direcciones IP a las interfaces de los dispositivos.

```
NS_LOG_INFO ("Asignación de direcciones IPs");  
Ipv4AddressHelper ipv4;  
ipv4.SetBase ("10.1.1.0", "255.255.255.0");  
ipv4.Assign (terminalDevices);
```

13. Se crea el nivel de aplicaciones, para esto se emplea la clase *OnOffHelper*, que no es más que un ayudante para que sea más cómodo en el momento de crear una instancia de una *NS-3 :: OnOffApplication* en los nodos. Se especifica la velocidad de transmisión a utilizarse la misma que será constante.

```
OnOffHelper onoff ("NS-3::UdpSocketFactory",
```

```
Address (InetSocketAddress (Ipv4Address ("192.168.1.2"), port)));
onoff.SetConstantRate (DataRate ("100Mb/s"));
ApplicationContainer app = onoff.Install (terminales.Get (0));
```

14. Se genera el tráfico a un solo destino basándose en un patrón de OnOff, el cual será configurado a continuación.

```
onoff.SetAttribute ("Remote",
    AddressValue (InetSocketAddress (Ipv4Address
        ("192.168.1.3"), port)));
app = onoff.Install (terminales.Get (0));
app.Start (Seconds (1.1));
app.Stop (Seconds (10.0));
```

15. Se habilita el rastreo

```
AnimationInterface anim ("topoestrella.xml");
anim.SetConstantPosition (terminales.Get(0), 0.0, 0.0);
.....
```

16. Se ejecuta la simulación.

```
Simulator::Run ();
```

Como se puede observar en los dos scripts explicados anteriormente, todos llevan la misma estructura por lo tanto en los scripts a continuación, no será necesario explicarlos detalladamente ya que la mayoría de sentencias fueron explicadas previamente.

5.1.4 Estándar IEEE 802.11b/g – WIFI.

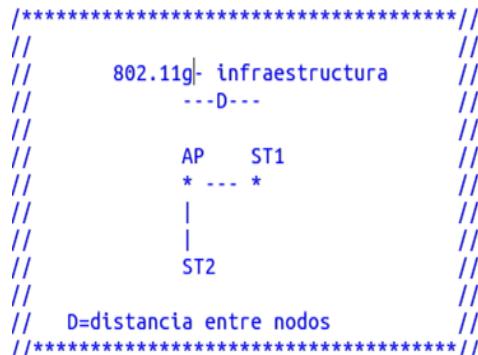


Figura 37 Modo infraestructura

Fuente: Autor

La figura 37 muestra la configuración de una red en modo infraestructura, la misma que se encuentra conformada por un AP y dos estaciones, etiquetadas como AP, ST1 y ST2 respectivamente.

5.1.4.1 Modo Infraestructura

1. Se debe cargar los archivos de inclusión

```
#include "NS-3/wifi-module.h"  
....  
#include "NS-3/netanim-module.h".
```

2. Luego de los módulos de inclusión sigue el espacio de nombres ns-3 el mismo que contiene las declaraciones de ns-3 esto se lo representa con:

Using namespace ns3

3. Se define el componente de registro con:

```
NS_LOG_COMPONENT_DEFINE ("OpenFlowCsmaSwitchExample");
```

4. Se declaran las variables empleadas en el script.

```
uint32_t distancia = 550;
```

En este caso hay una parte muy importante que es, el establecimiento de la tasa de transmisión, así como la codificación del canal a emplearse, esto se lo puede determinar con la línea de código mostrada a continuación:

```
std::string phyMode ("DsssRate11Mbps");
```

5. Se establece el valor inicial de las variables declaradas anteriormente, esto se lo realiza dentro del programa principal de la manera siguiente:

```
CommandLine cmd;  
cmd.AddValue ("tamapaquete", "tamaño del paquete enviado",  
tamapaquete);
```

6. Se modifica los valores que tienen por defecto los objetos en NS-3, en este caso se modifica: *FragmentationThreshold* con el fin de deshabilitar la fragmentación de las tramas desde un umbral determinado, *RtsCtsThreshold* para deshabilitar RTS / CTS de las tramas desde un umbral escogido, *NonUnicastMod*, permite establecer el valor de la velocidad de transmisión de datos no unicast igual al unicast.

```
Config::SetDefault ("NS-  
3::WifiRemoteStationManager::FragmentationThreshold", StringValue  
("2200"));
```

7. Se crean el número de nodos que se crea necesario, los mismos que representarán los dispositivos en este caso son 2 los nodos creados.

```
NodeContainer nodo;  
nodo.Create (2);
```

8. Se crean el número de nodos que se crea necesario, los mismos que representarán los *AP* en este caso es 1 nodo el creado.

```
NodeContainer ap;
```

```
ap.Create (1);
```

9. Se construye la topología con la ayuda del *Helper*.

```
WifiHelper wifi;
```

Se habilita los Logs.

```
wifi.EnableLogComponents ();
```

Se establece el estándar wifi que se va a emplear

```
wifi.SetStandard (WIFI_PHY_STANDARD_80211b);
```

Se determina atributos de la estación como la velocidad con la que se enviará los datos y RTC en este caso se empleó una velocidad constante

```
wifi.SetRemoteStationManager ("NS-3::ConstantRateWifiManager",  
"DataMode",StringValue (phyMode),  
"ControlMode",StringValue (phyMode));
```

10. Se emplea otro ayudante para crear y administrar los objetos PHY :

```
YansWifiPhyHelper wifiPhy;
```

Entre muchos de los atributos que este ayudante permite modificar, se modifica la ganancia de transmisión como de recepción.

```
wifiPhy.Set ("RxGain", DoubleValue (0));
```

11. Para modificar los parámetros del canal a usarse se emplea:

```
YansWifiChannelHelper wifiChannel;
```

Este ayudante permite determinar el modelo de la pérdida de propagación a través de un medio de transmisión y el modelo de retardo de propagación.

```
FriisPropagationLossModel;
```

```
ConstantSpeedPropagationDelayModel;
```

12. Para configurar el modelo MAC se emplea el siguiente ayudante:

```
WifiMacHelper wifiMac ;
```

Se configura los atributos tanto del AP como de la estación en este caso denominada nodo, en el script actual ambos tienen SSID ns-3-ssid wifi y no tienen activada calidad de servicio.

```
Ssid ssid = Ssid ("wifi ");  
wifiMac.SetType ("NS-3::StaWifiMac",  
"Ssid", SsidValue (ssid),  
"ActiveProbing", BooleanValue (false));
```

13. Se modifica la movilidad de los nodos con el ayudante:

```
MobilityHelper mobility;
```

Se establece la posición de los nodos en el plano x, y; en función de la variable distancia, y que estos permanecerán constantes durante toda la simulación.

14. Se procede a crear la pila de protocolos.

```
InternetStackHelper internet;  
internet.Install (ap);  
internet.Install (nodo);
```

15. Se asigna las direcciones IP a las interfaces de los dispositivos.

```
NS_LOG_INFO ("Asignación de direcciones IPs");  
Ipv4AddressHelper ipv4;  
ipv4.SetBase ("192.168.1.0", "255.255.255.0");  
ipv4.Assign (devices);
```

16. Se crea el nivel de aplicaciones

```
TypeId tid = TypeId::LookupByName ("NS-3::UdpSocketFactory");  
Ptr<Socket> recvSink = Socket::CreateSocket (nodo.Get (1), tid);
```

17. Se habilita el rastreo

```
wifiPhy.EnablePcap ("/root/NS-3/ns-allinone-3.22/ns  
3.22/src/resultados/infra_b/infra_ap", apDevice);
```

18. Se ejecuta la simulación.

```
Simulator::Run ();
```

Como se observa se necesita configurar principalmente:

- WifiChannel
- WifiPhy
- WifiMac

- WiFiDevice
- Movilidad

WifiChannel: recibe la señal de los dispositivos en un mismo canal wifi. Se configura principalmente la propagación modelo de pérdida y el modelo de retardo de propagación.

WifiPhy: envía y recibe la señal inalámbrica de WifiChannel, principalmente se configura la tasa de error.

WifiMac: tiene que ver con la arquitectura del dispositivo, es decir determinar si es ad-hoc o AP – STA, si trabaja con Qos o sin ella.

WifiDevice: se configura en el estándar WI-FI que deseé: 802.11b, 802.11g.

A continuación, se muestra en la tabla 4 las variaciones de las configuraciones para IEEE 802.11g modo infraestructura y IEEE 802.11b/g modo ad-hoc, en referencia al código explicado anteriormente.

Tabla 4

Variaciones en las configuraciones de los diferentes estándares.

		WifiPhy	WifiDevice	WifiMac
IEEE 802.11b				
MODO	***** // 802.11g- infraestructura // ---D--- // AP ST1 // * --- * // ST2 // D=distancia entre nodos *****	std::string phyMode ("DsssRate11Mbps");	wifi.SetStandard (WIFI_PHY_STANDA RD_80211b);	Aquí se configura el nodo que realizara la función de AP-STA.
IEEE 802.11g				
MODO	***** // 802.11g- infraestructura // ---D--- // AP ST1 // * --- * // ST2 // D=distancia entre nodos *****	std::string phyMode ("ErpOfdmRate54 Mbps");	wifi.SetStandard (WIFI_PHY_STANDA RD_80211g);	Aquí se configura el nodo que realizara la función de AP-STA
IEEE 802.11b				
MODO AD-HOC		std::string phyMode ("DsssRate11Mbps");	wifi.SetStandard (WIFI_PHY_STANDA RD_80211b);	wifiMac.GetType ("NS-3::AdhocWifiMac");
IEEE 802.11g				
MODO AD-HOC	***** // 802.11b- ad-hoc // ---D--- // n0 n1 // * --- * // D=distancia entre nodos *****	std::string phyMode ("ErpOfdmRate54 Mbps");	wifi.SetStandard (WIFI_PHY_STANDA RD_80211g);	wifiMac.GetType ("NS-3::AdhocWifiMac");

Fuente: Autor

5.1.5 Estándar IEEE 802.11s

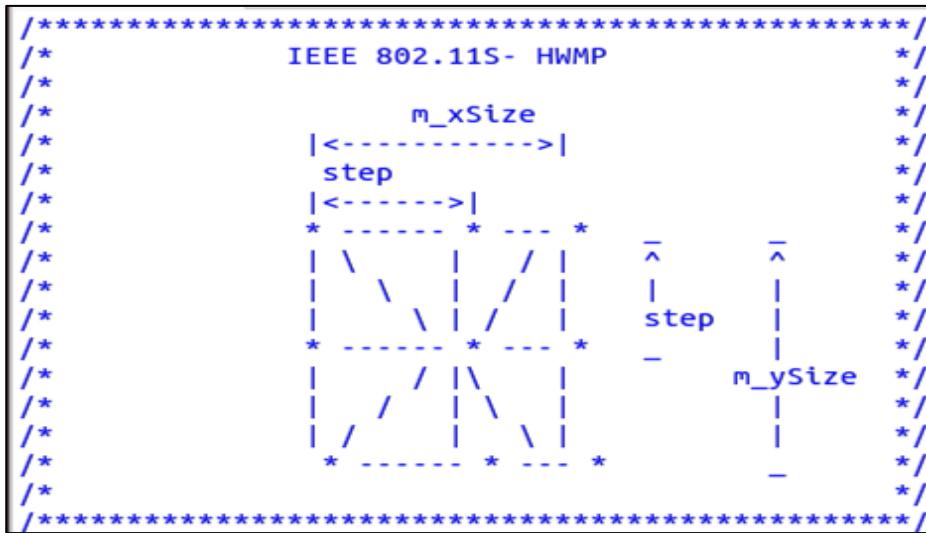


Figura 38 Topología estándar IEEE 802.11s

Fuente: Autor

La figura 38 muestra la configuración de una red mallada $n \times m$ donde n y m son el número de nodo por el que se encuentra formada la malla, n representa los nodos ubicados de manera horizontal y m representa los nodos ubicados de manera vertical, dichos nodos se encuentran separados entre sí una distancia $step$. En este caso se hace ping desde el primer nodo al último nodo.

1. Se debe cargar los archivos de inclusión

```
#include "NS-3/mesh-module.h"  
....  
#include "NS-3/netanim-module.h".
```

2. Luego de los módulos de inclusión sigue el espacio de nombres ns-3 el mismo que contiene las declaraciones de ns-3 esto se lo representa con:

Using namespace ns3

3. Se define el componente de registro con:

```
NS_LOG_COMPONENT_DEFINE ("mesh-hwmp");
```

4. Se declaran las variables empleadas en el script.

```
double m_step;
```

En este caso m_step es la distancia existente entre los nodos.

5. Se establece el valor inicial de las variables declaradas anteriormente, esto se lo realiza dentro del programa principal de la manera siguiente:

```
CommandLine cmd;
```

```
cmd.AddValue ("step", "Separación entre los nodos", m_step);
```

6. Se crean el número de nodos en forma de malla, que se crea necesario los mismos que representaran los Nodos.

```
nodes.Create (m_ySize*m_xSize);
```

7. Se habilita los Logs.

```
NS_LOG_DEBUG ("Tiempo de simulación: " << m_totalTime << " s");
```

8. Se emplea un ayudante para crear y administrar los objetos PHY :

```
YansWifiPhyHelper wifiPhy;
```

9. Para modificar los parámetros del canal a usarse se emplea:

```
YansWifiChannelHelper wifiChannel;
```

En este caso se emplea los modelos que vienen por defecto LogDistancePropagationLossModel y ConstantSpeedPropagationDelayModel con una pérdida de referencia de 46.6777 dB.

10. Se crea un ayudante de malla, instala las pilas y crea todos los protocolos necesarios e instala en los dispositivos de punto de malla:

```
mesh = MeshHelper::Default();
```

11. Se establece el número de interfaces, el valor predeterminado es el punto de malla de una sola interfaz

```
mesh.SetNumberOfInterfaces (m_nIfaces);
```

12. Se instala los protocolos mesh.

```
meshDevices = mesh.Install (wifiPhy, nodes);
```

13. Se modifica la movilidad de los nodos con el ayudante, a una topología de red estática:

```
MobilityHelper mobility;
```

Se establece la posición de los nodos en el plano x, y en función de la variable distancia, y que estos permanecerán constantes durante toda la simulación.

14. Se procede a crear la pila de protocolos.

```
InternetStackHelper internetStack;  
internetStack.Install (nodes);
```

15. Se asigna las direcciones IP a las interfaces de los dispositivos.

```
NS_LOG_INFO ("Asignación de direcciones IPs");
```

```
Ipv4AddressHelper ipv4;  
ipv4.SetBase ("192.168.1.0", "255.255.255.0");  
ipv4.Assign (meshDevices);
```

16. Se crea el nivel de aplicaciones

```
UdpEchoServerHelper echoServer (9);  
ApplicationContainer serverApps = echoServer.Install (nodes.Get (0));  
ApplicationContainer clientApps = echoClient.Install (nodes.Get  
(m_xSize*m_ySize-1));
```

17. Se habilita el rastreo

```
Ptr<FlowMonitor> monitor;  
FlowMonitorHelper flowmon_helper;
```

18. Se estable un contador para que para que se imprima un reporte según la secuencia deseada.

```
float i;  
for ( .....)  
{  
.....  
}
```

19. Se activa el reporte para cada uno de los MP.

```
for (NetDeviceContainer::Iterator i = meshDevices.Begin (); i !=  
meshDevices.End (); ++n)  
{
```

```
std::ostringstream os;
os << "/root/NS-3/ns-allinone-3.22/ns-3.22/RES/reporter-" << n << "en
el tiempo"<< Now() << ".xml";
```

20. Se ejecuta la simulación.

```
return t.Run();
```

5.2 Descripción del Hardware Utilizado.

La tabla 5, a continuación, muestra las características del equipo empleado.

Tabla 5

Características del equipo

Parámetro	Hardware
Equipo	Laptop Dell Intel I7 6ta Gen
Memoria Ram	12,0 GB
Disco Duro	1T
Memoria Dedicada	3 GB
Procesador	Intel® Core(TM) i7-6500U CPU @ 2,50GHz 2,60 GHz

Fuente: Autor

5.3 Descripción del Software Utilizado.

5.3.1 Sistema Operativo.

Una característica de NS-3 es que es compatible con diferentes plataformas entre ellas Linux, es por eso que se optó por usar el Sistema Operativo Ubuntu (véase la figura 39), que se basa en GNU/Linux, además de ser distribuido como software libre.

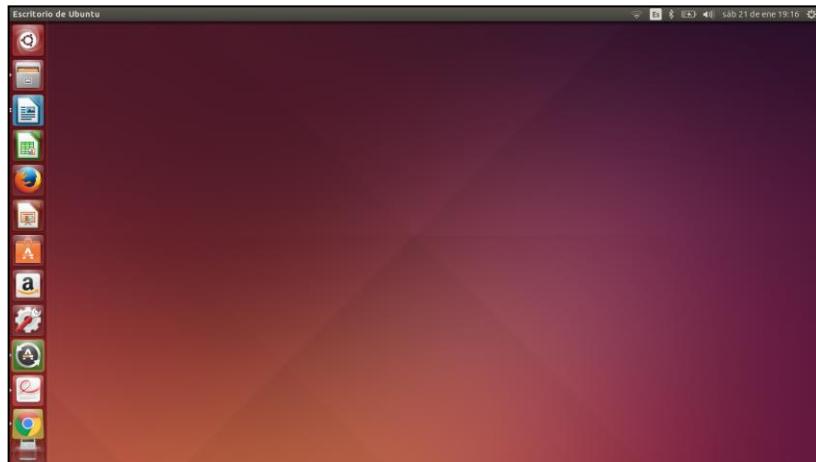
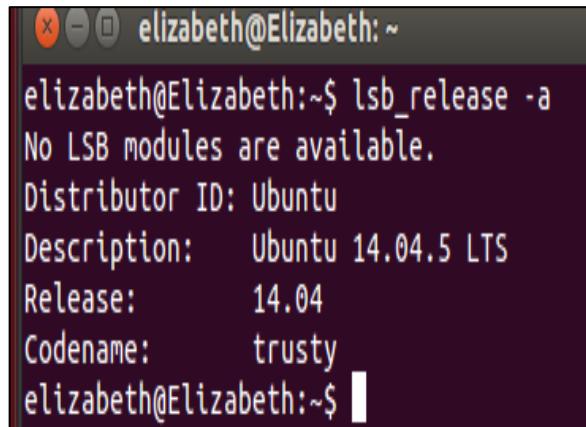


Figura 39 Sistema operativo Ubuntu

Fuente: Autor

La versión utilizada es la 14.04.5 LTS, ya que tienen soporte a largo plazo, este soporte es de 5 años, esta versión tiene soporte hasta finales del año 2019.

Para poder observar que versión de Ubuntu tenemos (véase figura40), se ejecuta el comando: lsb_release -a. Se puede usar cualquier versión de Ubuntu actual, pero es preferible usar una LTS para no tener que estar actualizando el sistema operativo cada año.



```
elizabeth@Elizabeth:~$ lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 14.04.5 LTS
Release:        14.04
Codename:       trusty
elizabeth@Elizabeth:~$
```

Figura 40 Versión de Ubuntu utilizada.

Fuente: Autor

5.3.2 Simulador de redes.

El simulador de redes empleado es NS-3, existen diferentes versiones que se han ido desarrollando a través de los años como se muestra en la figura 41, se escogió la versión 3.22 ya que siempre se recomienda no utilizar la última versión ya que está aún no tiene depurado todos sus posibles errores.

ns-3.24 (September 2015)	ns-3.23 (May 2015)	ns-3.25 (March 2016)
ns-3.21 (September 2014)	ns-3.20 (June 2014)	ns-3.22 (February 2015)
ns-3.18 (August 2013)	ns-3.17 (May 2013)	ns-3.19 (December 2013)
ns-3.15 (August 2012)	ns-3.14 (June 2012)	ns-3.16 (December 2012)
ns-3.12 (August 2011)	ns-3.11 (May 2011)	ns-3.13 (December 2011)
ns-3.9 (August 2010)	ns-3.8 (May 2010)	ns-3.10 (January 2011)
ns-3.6 (October 2009)	ns-3.5 (July 2009)	ns-3.7 (January 2010)
ns-3.3 (December 2008)	ns-3.2 (September 2008)	ns-3.4 (April 2009)
		ns-3.1 (July 2008)

Figura 41 Versiones de NS-3

Fuente: Autor

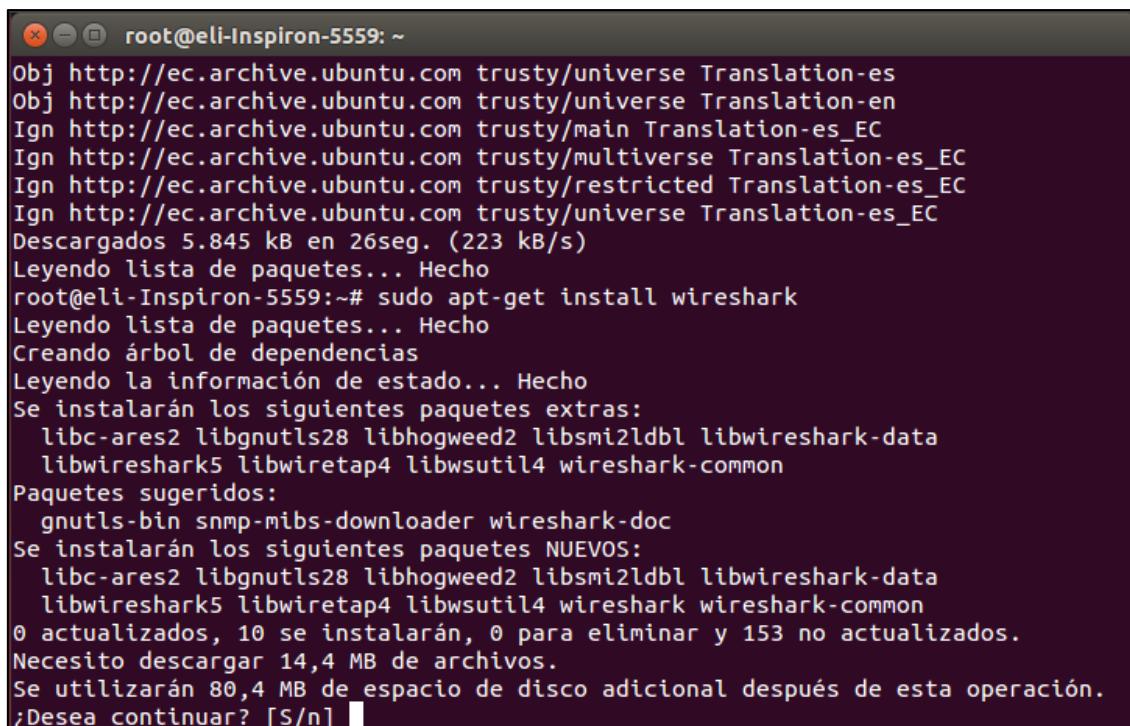
5.3.3 Herramientas de Análisis y Animación.

Para el análisis de los datos se emplea la siguiente herramienta:

5.3.3.1 Herramientas de Análisis de datos -Wireshark

Este permite analizar el tráfico de una red, en este caso se lo utilizará para analizar las redes Ethernet. Esta herramienta no viene instalada dentro del paquete del simulador, se la realiza ejecutando los siguientes comandos en la consola.

```
sudo -i  
sudo add-apt-repository ppa:pi-rho/security  
sudo apt-get update  
sudo apt-get install wireshark
```



The terminal window shows the root user executing commands to install Wireshark. It displays package dependencies, download progress, and the final confirmation prompt.

```
root@eli-Inspiron-5559: ~  
Obj http://ec.archive.ubuntu.com trusty/universe Translation-es  
Obj http://ec.archive.ubuntu.com trusty/universe Translation-en  
Ign http://ec.archive.ubuntu.com trusty/main Translation-es_ES  
Ign http://ec.archive.ubuntu.com trusty/multiverse Translation-es_ES  
Ign http://ec.archive.ubuntu.com trusty/restricted Translation-es_ES  
Ign http://ec.archive.ubuntu.com trusty/universe Translation-es_ES  
Descargados 5.845 kB en 26seg. (223 kB/s)  
Leyendo lista de paquetes... Hecho  
root@eli-Inspiron-5559:~# sudo apt-get install wireshark  
Leyendo lista de paquetes... Hecho  
Creando árbol de dependencias  
Leyendo la información de estado... Hecho  
Se instalarán los siguientes paquetes extras:  
  libc-ares2 libgnutls28 libhogweed2 libsmi2ldbl libwireshark-data  
    libwireshark5 libwiretap4 libwsutil4 wireshark-common  
Paquetes sugeridos:  
  gnutls-bin snmp-mibs-downloader wireshark-doc  
Se instalarán los siguientes paquetes NUEVOS:  
  libc-ares2 libgnutls28 libhogweed2 libsmi2ldbl libwireshark-data  
    libwireshark5 libwiretap4 libwsutil4 wireshark wireshark-common  
0 actualizados, 10 se instalarán, 0 para eliminar y 153 no actualizados.  
Necesito descargar 14,4 MB de archivos.  
Se utilizarán 80,4 MB de espacio de disco adicional después de esta operación.  
¿Desea continuar? [S/n] ■
```

Figura 42 Proceso de instalación de wireshark

Fuente: Autor

En el proceso de instalación aparecerá lo que se muestra en la figura 43 a continuación.

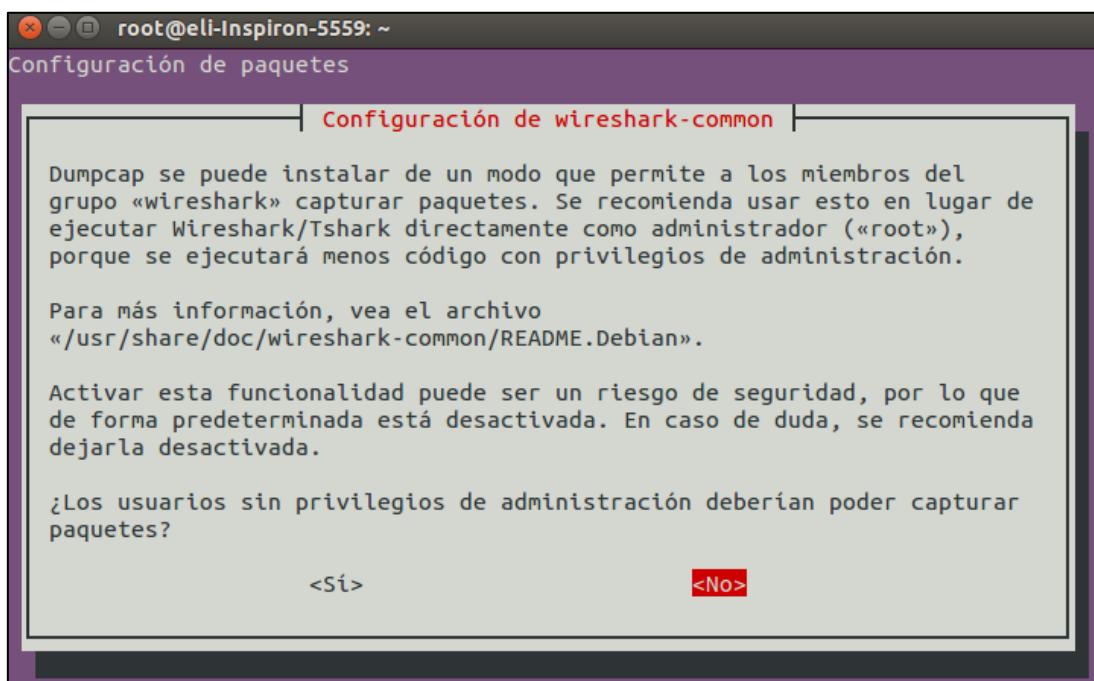


Figura 43 Configuración de wireshark

Fuente: Autor

Se escoge *SI* o *NO* dependiendo si se desea que el usuario sin privilegios de administrador pueda capturar paquetes.

Terminado el proceso anterior ya se tendrá instalado la herramienta (véase figura 44).

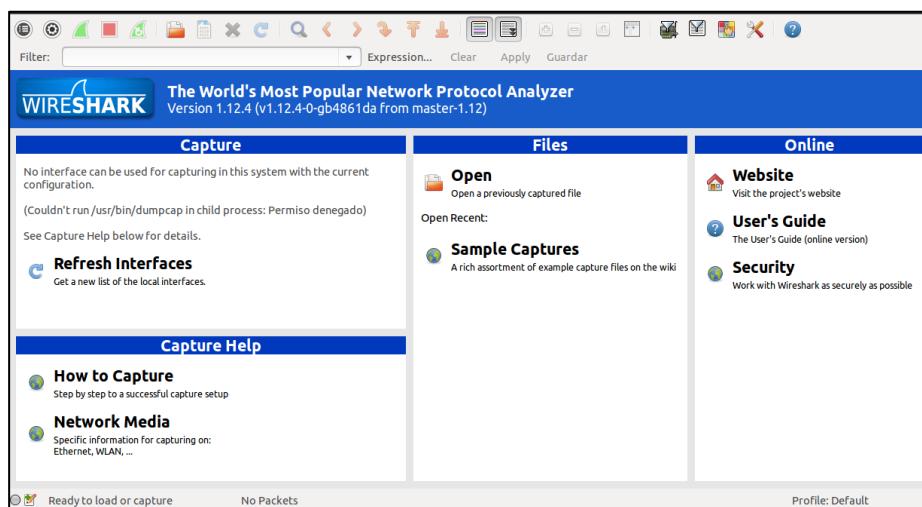


Figura 44 Programa Wireshark

Fuente: Autor

Se empleó dos herramientas para la animación:

5.3.3.2 Herramientas de Animación- PyViz

Es un visualizador de simulación en vivo, funciona tanto para Python como para C++.

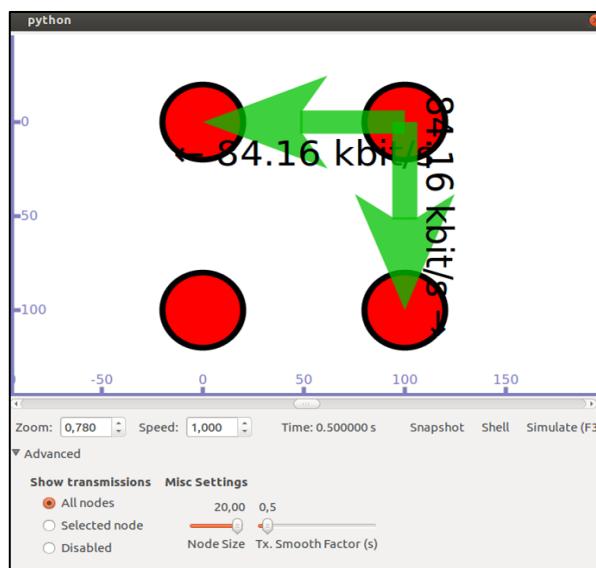


Figura 45 Herramienta PyViz

Fuente: Autor

Para ejecutar esta herramienta se escribe en la consola --vis al momento de hacer correr el programa deseado, en la figura 46 se muestra un ejemplo.

```
eli@eli-Inspiron-5559:~$ sudo -i  
[sudo] password for eli:  
root@eli-Inspiron-5559:~# cd ns3/ns-allinone-3.22/ns-3.22/  
root@eli-Inspiron-5559:~/ns3/ns-allinone-3.22/ns-3.22# ./waf --run malla -- vis
```

Figura 46 Sentencias para ejecutar PyViz

Fuente: Autor

5.3.3.3 Herramientas de Animación –NetAnim

Es un visualizador de simulación basado en un archivo de rastreo XML, que se obtiene durante la simulación. Se utilizó la versión 3.015 que viene por defecto.

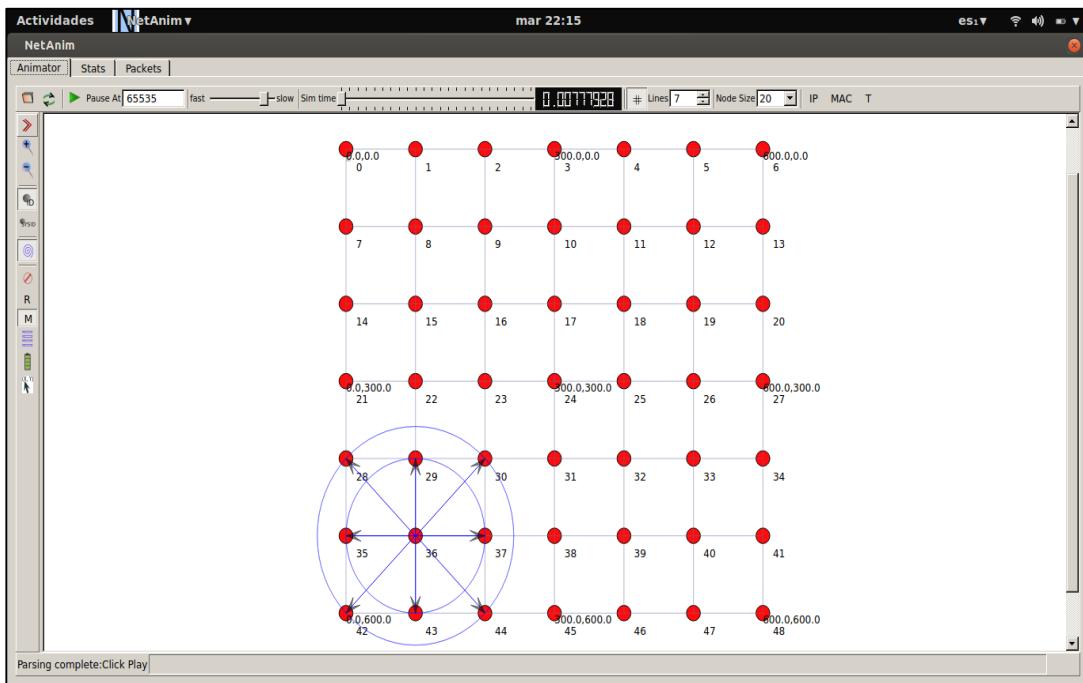


Figura 47 Herramienta de visualización NetAnim

Fuente: Autor

Para ejecutar esta herramienta se realiza lo siguiente (véase figura 48):

```
eli@eli-Inspiron-5559:~$ sudo -i
[sudo] password for eli:
root@eli-Inspiron-5559:~# cd ns3
root@eli-Inspiron-5559:~/ns3# cd ns-allinone-3.22/
root@eli-Inspiron-5559:~/ns3/ns-allinone-3.22# cd netanim-3.105/
root@eli-Inspiron-5559:~/ns3/ns-allinone-3.22/netanim-3.105# ./NetAnim
```

Figura 48 Proceso para ejecutar NetAnim

Fuente: Autor

5.4 Diagrama de flujo

5.4.1 Diagrama general de un script en NS-3 .



Figura 49. Diagrama de flujo general de un script en NS-3
Fuente: Autor

6. RESULTADOS

A continuación, se muestra los resultados obtenidos de las simulaciones con los diferentes estándares y en diferentes topologías. Así mismo se describen las pruebas realizadas para determinar la métrica, en una red tipo malla con el estándar IEEE 802.11s.

6.1 Estándar IEEE 802.3

6.1.1 Topología tipo bus

La figura 50 muestra el resultado de la herramienta PyViz, para la topología tipo bus.

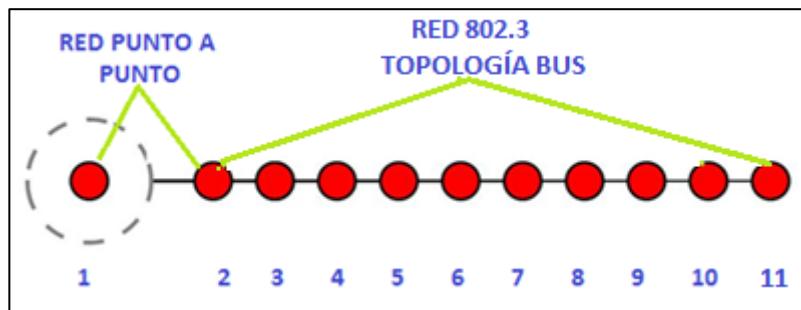


Figura 50 Topología tipo bus – Herramienta PyViz

Fuente: Autor

La figura 50 permite observar una red cableada compuesta, la misma que se encuentra formada por once nodos, dos configurados como una red punto a punto y nueve configurados como un red con topología tipo bus con un método de acceso CSMA, cabe mencionar que el nodo número 2 de la red punto a punto también esta configurado con el método de acceso CSMA .

En la figura 51 se emplea un visualizador de simulación *NetAnime*, el mismo que se encuentra basado en un archivo de rastreo XML.

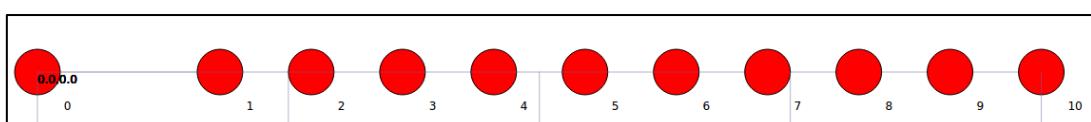


Figura 51 Topología tipo bus – Herramienta NetAnim

Fuente: Autor

La figura 52 muestra como es enviado el paquete por medio de los dos primeros nodos, que corresponden a la red punto a punto, en donde cada paquete es enviado uno a uno de nodo a nodo.

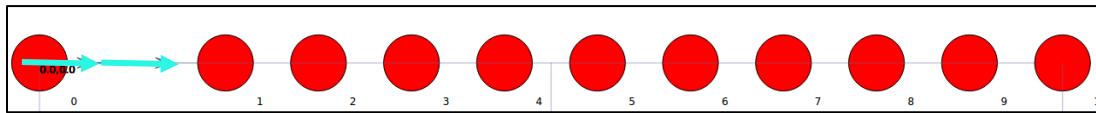


Figura 52 Paquete enviados del nodo 0 al nodo 1

Fuente: Autor

La figura 53 y figura 54 muestra cuando el paquete ingresa a la red tipo bus configurada con el método de acceso CSMA, en donde todos los paquetes son enviados hacia todos los nodos a la vez, tanto de ida como de regreso.

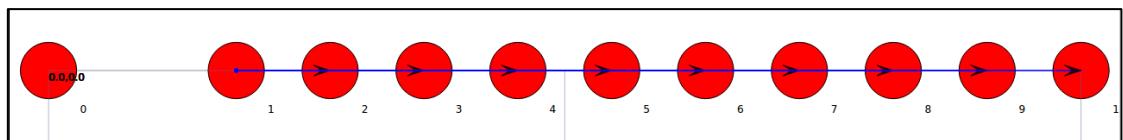


Figura 53 Envío de paquetes del nodo 1 al nodo 10

Fuente: Autor

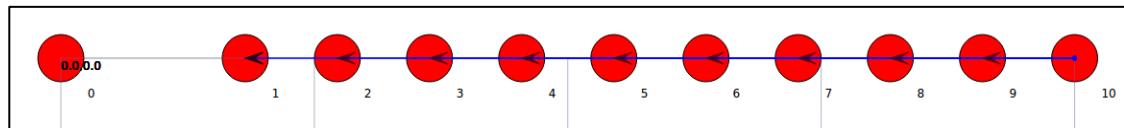


Figura 54 Envío de paquetes del nodo 10 al nodo 1

Fuente: Autor

La simulación de la red tipo bus arrojó como resultado un archivo de trazas .tr como el de la figura 55, cada línea mostrada corresponde a un evento.

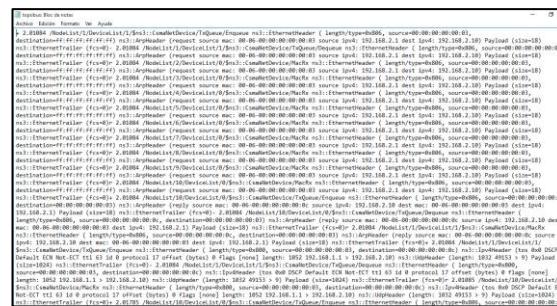


Figura 55 Resultado de archivo .tr – Topología tipo bus

Fuente: Autor

Como un archivo de trazas es complicado analizar, se puede analizar los mismos resultados con archivos .pcap, en la figura 56 se muestra los archivos .pcap generados, cabe mencionar que se crea uno por cada nodo. Este tipo de archivos son analizados mediante *Wireshark*.

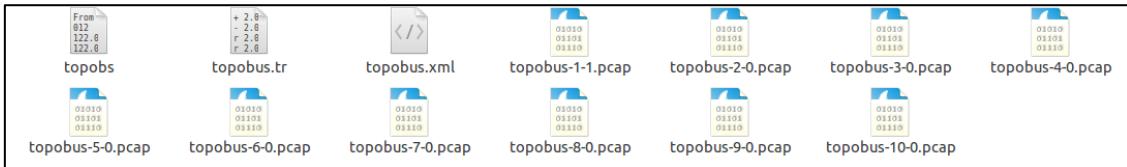


Figura 56 Archivos generados – Topología tipo bus

Fuente: Autor

La figura 57 lista los paquetes capturados y la interpretación del paquete seleccionado, como se observa se ha enviado 64 Bytes y se ha recibido 64 Bytes usando el estándar IEEE 802.3 Ethernet.

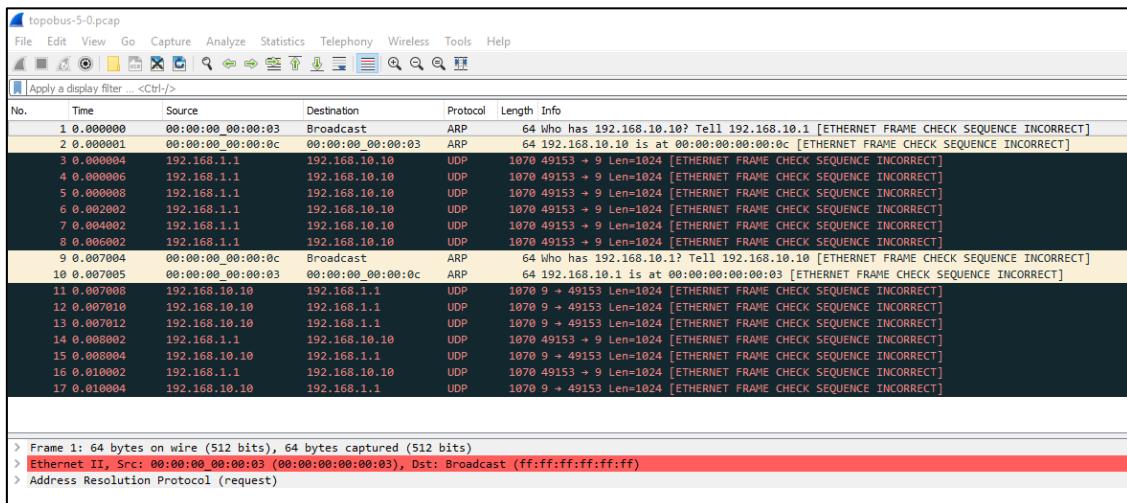


Figura 57 Resultado de archivo .pcap – Herramienta Wireshark – Nodo 5

Fuente: Autor

6.1.2 Topología tipo estrella

La figura 58 muestra el resultado de la herramienta PyViz, para la topología tipo estrella.

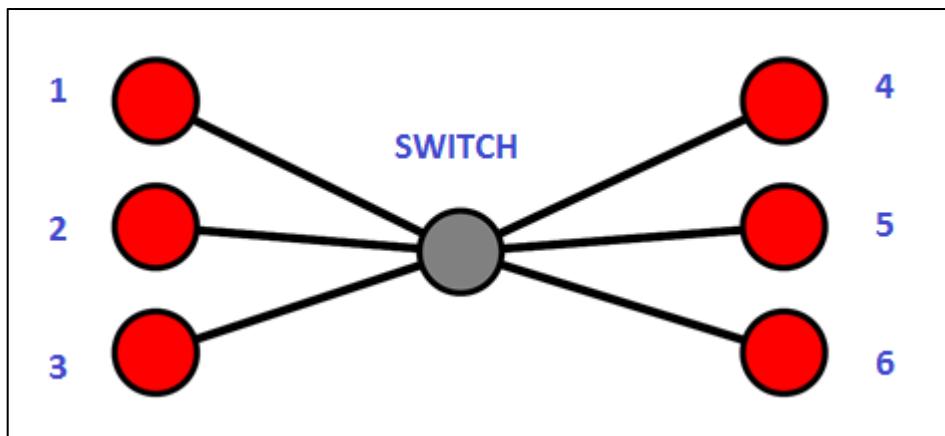


Figura 58 Topología tipo estrella mediante la herramienta Pviz

Fuente: Autor

La figura 58 permite observar la estructura de una topología tipo estrella en una red cableada, la misma que se encuentra compuesta por seis nodos, y un switch. Los nodos están configurados con un método de acceso CSMA.

Ya que se trata de una topología tipo estrella todos los nodos están conectados a un punto central en este caso el switch y cada paquete enviado por los nodos pasa a través del switch.

La figura 59 muestra la topología antes vista en donde se aprecia la posición de cada nodo dentro de una cuadricula.

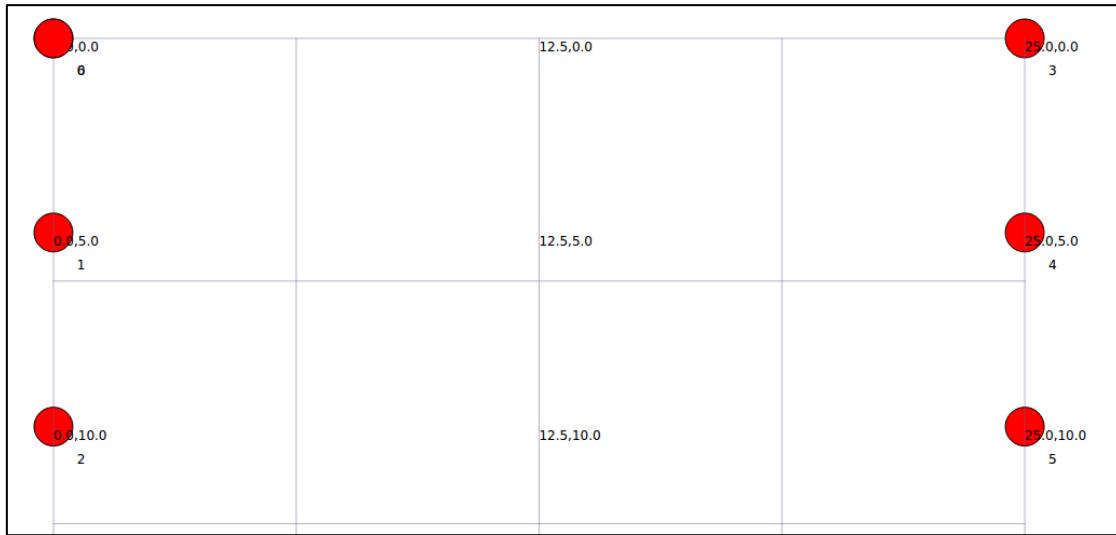


Figura 59 Topología tipo estrella – Herramienta NetAnim.

Fuente: Autor

La figura 60 muestra la dirección IP y MAC asignada a cada nodo.

Node:0 IP:192.168.1.1 MAC:00:00:00:00:00:01	Node:1 IP:192.168.1.2 MAC:00:00:00:00:00:03	Node:2 IP:192.168.1.3 MAC:00:00:00:00:00:05
Node:3 IP:192.168.1.4 MAC:00:00:00:00:00:07	Node:4 IP:192.168.1.5 MAC:00:00:00:00:00:09	Node:5 IP:192.168.1.6 MAC:00:00:00:00:00:0b
Node:6 IP:0.0.0.0 MAC:00:00:00:00:00:02		

Figura 60 Direcciones Asignadas

Fuente: Autor

La figura 61 muestra como viajan los paquetes desde el primer nodo hacia los demás nodos.

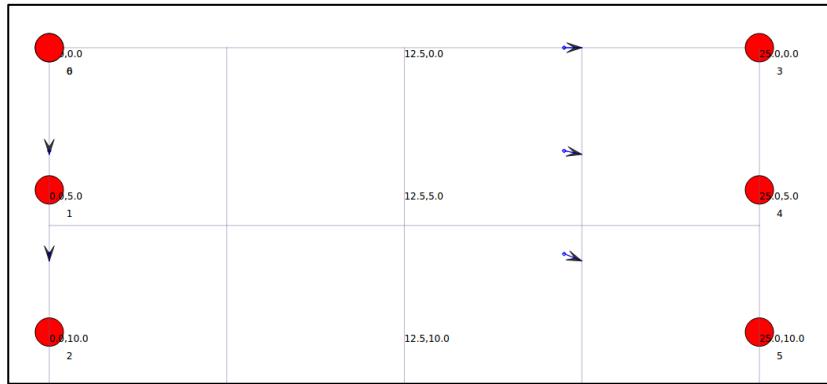


Figura 61 Envió de paquetes en topología estrella. –Herramienta NetAnim

Fuente: Autor

La figura 62 muestra los paquetes enviados desde el nodo seis hacia los cinco, cuatro, tres, dos y uno, se observa que cada paquete es enviado en un slot de tiempo distinto.

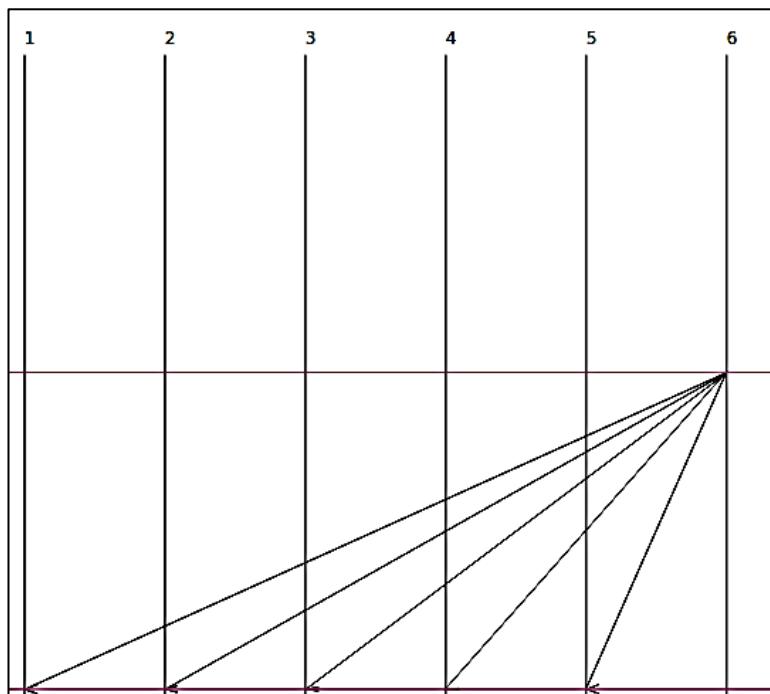


Figura 62 Diagrama de envío de paquetes desde el nodo 6 a los diferentes nodos.

Fuente: Autor

La simulación de la red tipo estrella arrojó como resultado un archivo de trazas .tr como el de la figura 63, cada línea mostrada corresponde a un evento.

```

topoestrella.tr x
+ 1.00441 /NodeList/0/DeviceList/0/$ns3::CsmaNetDevice/TxQueue/Enqueue ns3::EthernetHeader ( length/type=0x806, source=00:00:00:00:00:01,
destination=ff:ff:ff:ff:ff:ff) ns3::ArpHeader (request source mac: 00-06-00:00:00:00:01 source ipv4: 192.168.1.1 dest ipv4: 192.168.1.2) Payload
(size=18) ns3::EthernetTrailer (fcs=0)
- 1.00441 /NodeList/0/DeviceList/0/$ns3::CsmaNetDevice/TxQueue/Dequeue ns3::EthernetHeader ( length/type=0x806, source=00:00:00:00:00:01,
destination=ff:ff:ff:ff:ff:ff) ns3::ArpHeader (request source mac: 00-06-00:00:00:00:01 source ipv4: 192.168.1.1 dest ipv4: 192.168.1.2) Payload
(size=18) ns3::EthernetTrailer (fcs=0)
r 1.00646 /NodeList/0/DeviceList/0/$ns3::CsmaNetDevice/MacRx ns3::EthernetHeader ( length/type=0x806, source=00:00:00:00:00:01,
destination=ff:ff:ff:ff:ff:ff) ns3::ArpHeader (request source mac: 00-06-00:00:00:00:01 source ipv4: 192.168.1.1 dest ipv4: 192.168.1.2) Payload
(size=18) ns3::EthernetTrailer (fcs=0)
+ 1.00646 /NodeList/0/DeviceList/1/$ns3::CsmaNetDevice/TxQueue/Enqueue ns3::EthernetHeader ( length/type=0x806, source=00:00:00:00:00:01,
destination=ff:ff:ff:ff:ff:ff) ns3::ArpHeader (request source mac: 00-06-00:00:00:00:01 source ipv4: 192.168.1.1 dest ipv4: 192.168.1.2) Payload
(size=18) ns3::EthernetTrailer (fcs=0)
- 1.00646 /NodeList/0/DeviceList/1/$ns3::CsmaNetDevice/TxQueue/Dequeue ns3::EthernetHeader ( length/type=0x806, source=00:00:00:00:00:01,
destination=ff:ff:ff:ff:ff:ff) ns3::ArpHeader (request source mac: 00-06-00:00:00:00:01 source ipv4: 192.168.1.1 dest ipv4: 192.168.1.2) Payload
(size=18) ns3::EthernetTrailer (fcs=0)
+ 1.00646 /NodeList/0/DeviceList/0/$ns3::CsmaNetDevice/TxQueue/Enqueue ns3::EthernetHeader ( length/type=0x806, source=00:00:00:00:00:01,
destination=ff:ff:ff:ff:ff:ff) ns3::ArpHeader (request source mac: 00-06-00:00:00:00:01 source ipv4: 192.168.1.1 dest ipv4: 192.168.1.2) Payload
(size=18) ns3::EthernetTrailer (fcs=0)
- 1.00646 /NodeList/0/DeviceList/1/2/$ns3::CsmaNetDevice/TxQueue/Dequeue ns3::EthernetHeader ( length/type=0x806, source=00:00:00:00:00:01,
destination=ff:ff:ff:ff:ff:ff) ns3::ArpHeader (request source mac: 00-06-00:00:00:00:01 source ipv4: 192.168.1.1 dest ipv4: 192.168.1.2) Payload
(size=18) ns3::EthernetTrailer (fcs=0)
+ 1.00646 /NodeList/0/DeviceList/1/2/$ns3::CsmaNetDevice/TxQueue/Enqueue ns3::EthernetHeader ( length/type=0x806, source=00:00:00:00:00:01,
destination=ff:ff:ff:ff:ff:ff) ns3::ArpHeader (request source mac: 00-06-00:00:00:00:01 source ipv4: 192.168.1.1 dest ipv4: 192.168.1.2) Payload
(size=18) ns3::EthernetTrailer (fcs=0)
- 1.00646 /NodeList/0/DeviceList/3/$ns3::CsmaNetDevice/TxQueue/Dequeue ns3::EthernetHeader ( length/type=0x806, source=00:00:00:00:00:01,
destination=ff:ff:ff:ff:ff:ff) ns3::ArpHeader (request source mac: 00-06-00:00:00:00:01 source ipv4: 192.168.1.1 dest ipv4: 192.168.1.2) Payload
(size=18) ns3::EthernetTrailer (fcs=0)
+ 1.00646 /NodeList/0/DeviceList/4/$ns3::CsmaNetDevice/TxQueue/Enqueue ns3::EthernetHeader ( length/type=0x806, source=00:00:00:00:00:01,
destination=ff:ff:ff:ff:ff:ff) ns3::ArpHeader (request source mac: 00-06-00:00:00:00:01 source ipv4: 192.168.1.1 dest ipv4: 192.168.1.2) Payload
(size=18) ns3::EthernetTrailer (fcs=0)
- 1.00646 /NodeList/0/DeviceList/4/$ns3::CsmaNetDevice/TxQueue/Dequeue ns3::EthernetHeader ( length/type=0x806, source=00:00:00:00:00:01,
destination=ff:ff:ff:ff:ff:ff) ns3::ArpHeader (request source mac: 00-06-00:00:00:00:01 source ipv4: 192.168.1.1 dest ipv4: 192.168.1.2) Payload
(size=18) ns3::EthernetTrailer (fcs=0)

```

Figura 63 Resultado de archivo . tr – Topología estrella

Fuente: Autor

Se puede analizar los mismos resultados con archivos .pcap, en la figura 64 se muestra los archivos .pcap generados.

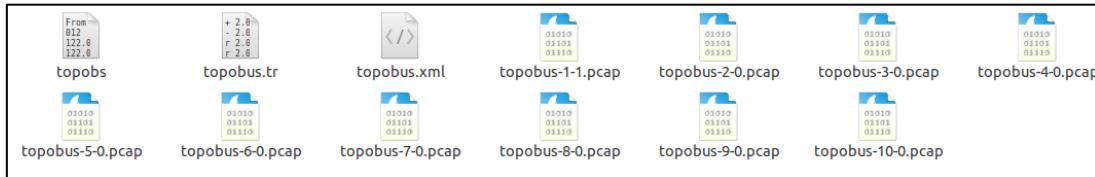


Figura 64 Archivos generados –Topología tipo estrella

Fuente: Autor

La figura 65 lista los paquetes capturados y la interpretación del paquete seleccionado, se observa que se ha enviado 64 Bytes y se ha recibido 64 Bytes usando el estándar IEEE 802.3 Ethernet con una topología en estrella.

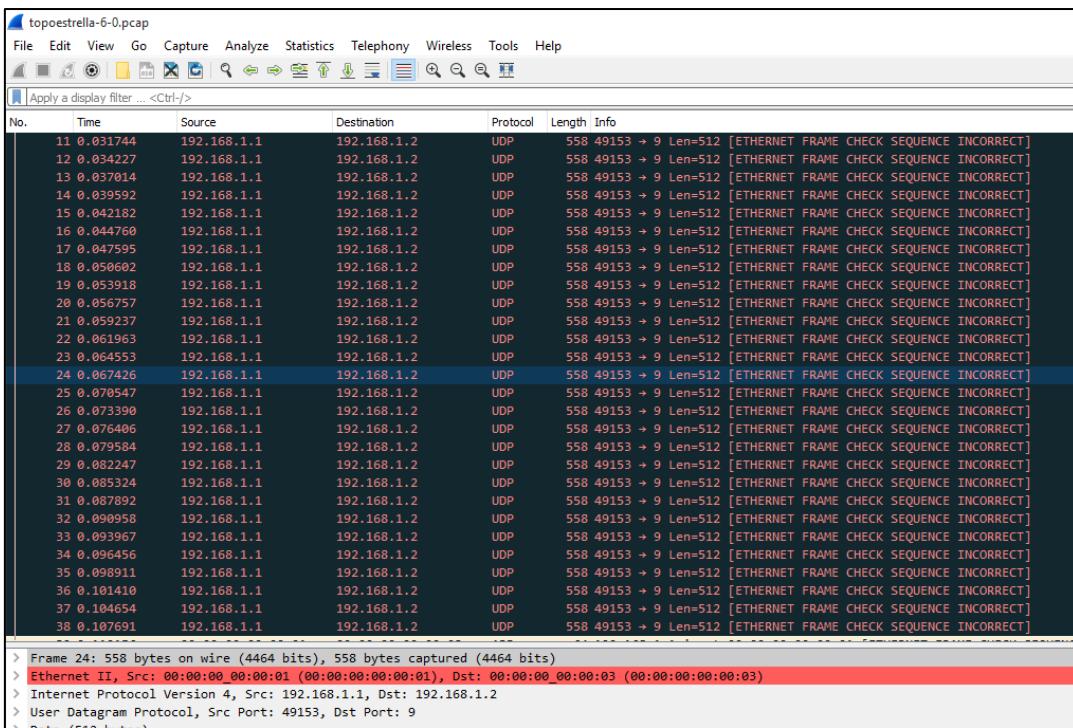


Figura 65 Resultado de archivo .pcap – Topología tipo estrella – Nodo 6

Fuente: Auto

6.2 Estándar IEEE 802.11 b/g

6.2.1 Modo Infraestructura

6.2.1.1 Modo Infraestructura – IEEE 802.11b

La figura 66 muestra el resultado de la herramienta NetAnim, de una red en modo infraestructura.

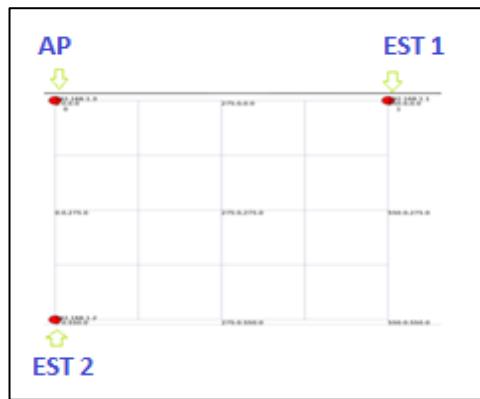


Figura 66 Modo Infraestructura – Herramienta NetAnm
Fuente: Autor

La figura 66 permite observar la estructura del modo infraestructura conformada por un nodo que representa un AP y dos nodos que representan las estaciones.

En la figura 67 se observa la asociación entre las estaciones y el AP.

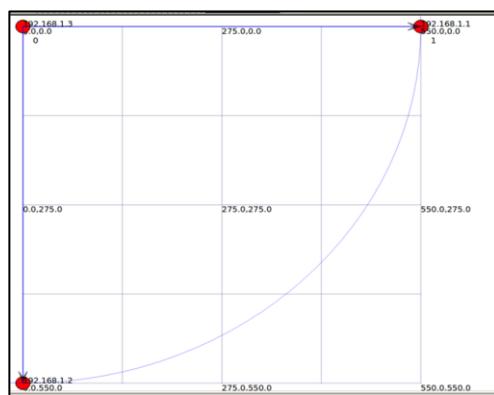


Figura 67 Asociación del AP con las estaciones
Fuente: Autor

La figura 68 muestra los archivos .pcap generados, para una red configurada en modo infraestructura.

anim1	21/3/2017 20:15	Documento XML	54 KB
infra_ap-0-0	21/3/2017 20:15	Wireshark capture...	43 KB
infra_estaciones-1-0	21/3/2017 20:15	Wireshark capture...	57 KB
infra_estaciones-2-0	21/3/2017 20:15	Wireshark capture...	26 KB

Figura 68 Archivos generados – Modo infraestructura

Fuente: Autor

Las figuras 69, 70 y 71 muestran los paquetes capturados y la interpretación del paquete seleccionado, como se puede observar el AP envía beacons para anunciar que la red está transmitiendo, además las estaciones envían solicitudes al AP para poder asociarse a él. Se realizan las consultas ARP y luego se envía los datos desde la estación al AP. También se puede observar tanto el estándar configurado como la velocidad de transmisión de datos.

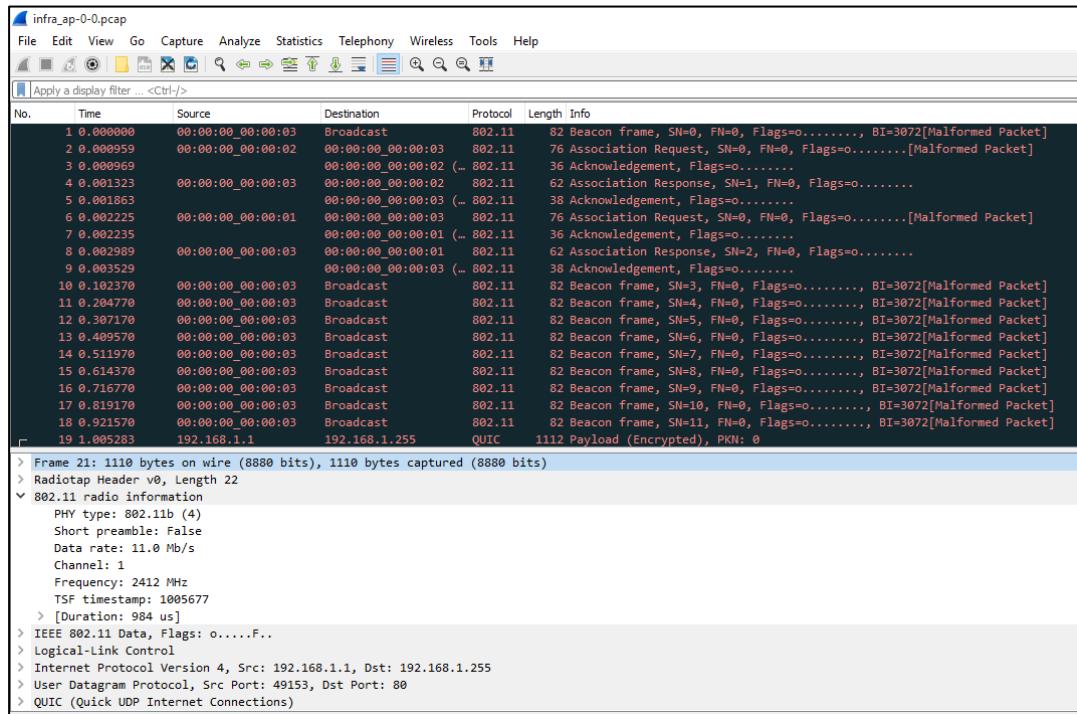


Figura 69 Resultado archivo .pcap – Modo infraestructura - AP

Fuente: Autor

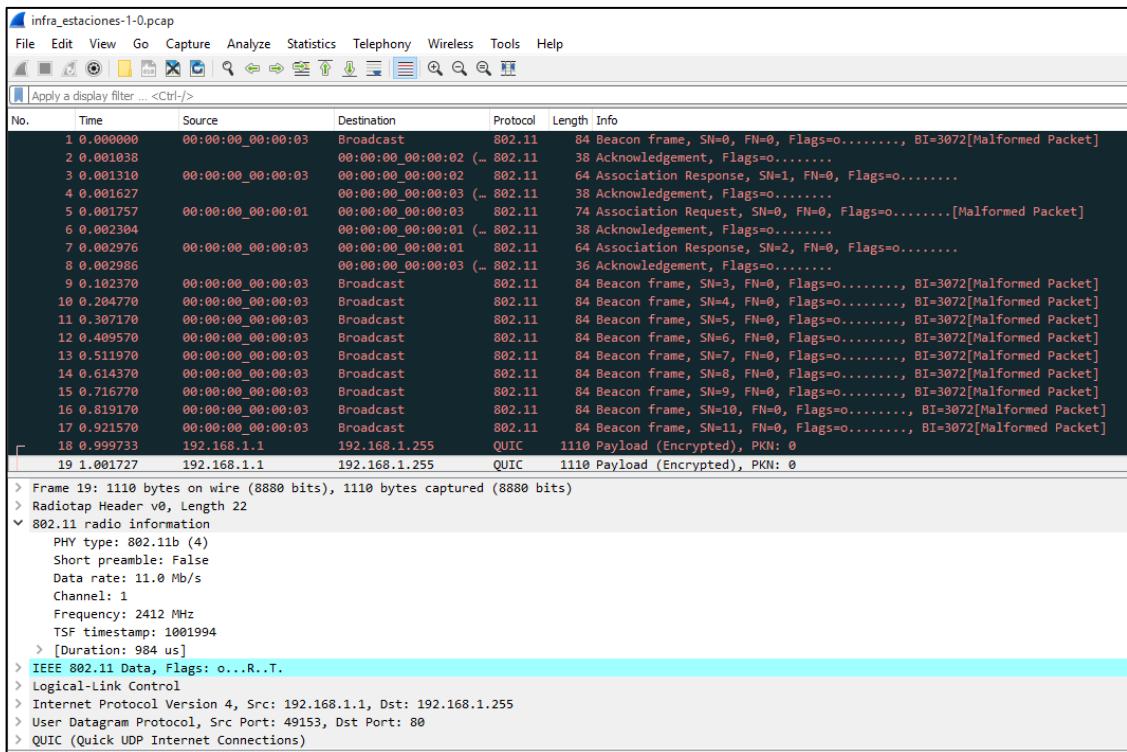


Figura 70 Resultado archivo .pcap – Modo infraestructura – Estación 1

Fuente: Autor

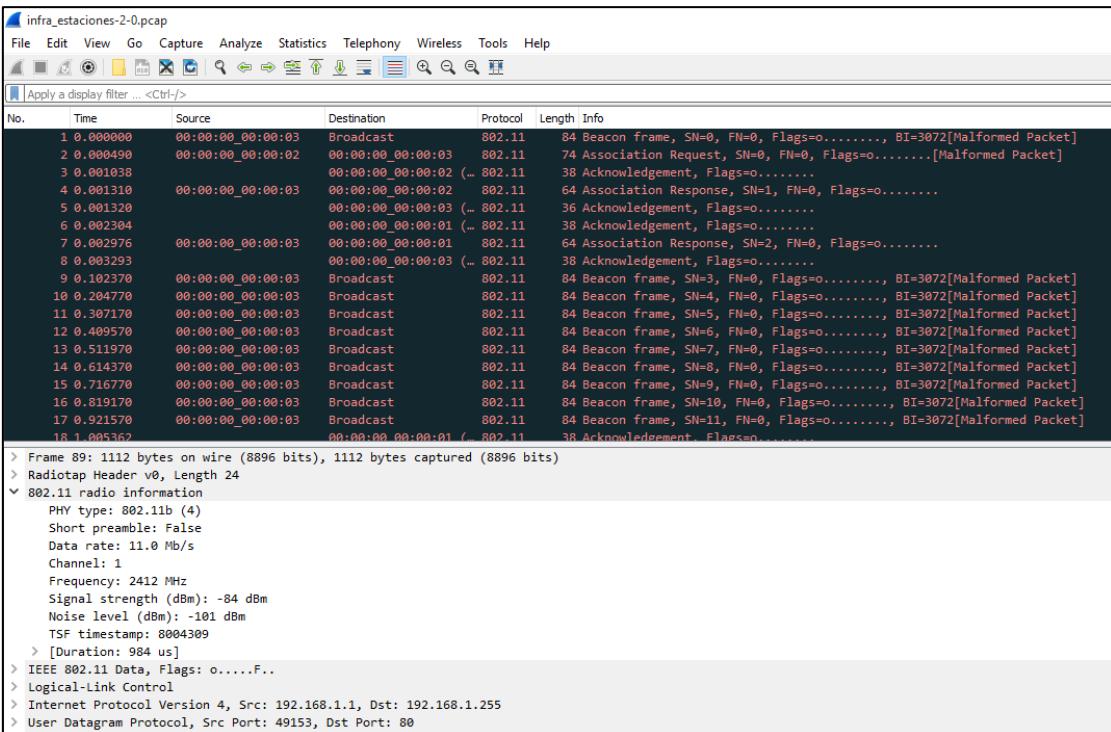


Figura 71 Resultado archivo .pcap – Modo infraestructura – Estación 2

Fuente: Autor

La figura 72 muestra el resultado al configurar una red en modo insfraestructura con el estándar IEEE 802.11b, las estaciones se encuentran ubicadas a 55 m del AP (véase figura 73). Se observa que a esta distancia de 10 paquetes enviados se reciben 10.

```
root@eli-Inspiron-5559:~/ns3/ns-allinone-3.22/ns-3.22# ./waf --run infraestructura-b
Waf: Entering directory '/root/ns3/ns-allinone-3.22/ns-3.22/build'
Waf: Leaving directory '/root/ns3/ns-allinone-3.22/ns-3.22/build'
'build' finished successfully (1.496s)
Prueba, envia 10 paquetes, distancia de 55m
PAQUETE RECIBIDO
root@eli-Inspiron-5559:~/ns3/ns-allinone-3.22/ns-3.22#
```

Figura 72 Resultado paquetes enviados

Fuente: Autor



Figura 73 Configuración modo infraestructura, estaciones separadas 55 m

Fuente: Autor

La figura 74 muestra el comportamiento de la red al variar la distancia entre la estación y el AP. Mientras mayor sea la distancia menor será el número de paquetes transmitidos (véase figura 75 y figura 76).

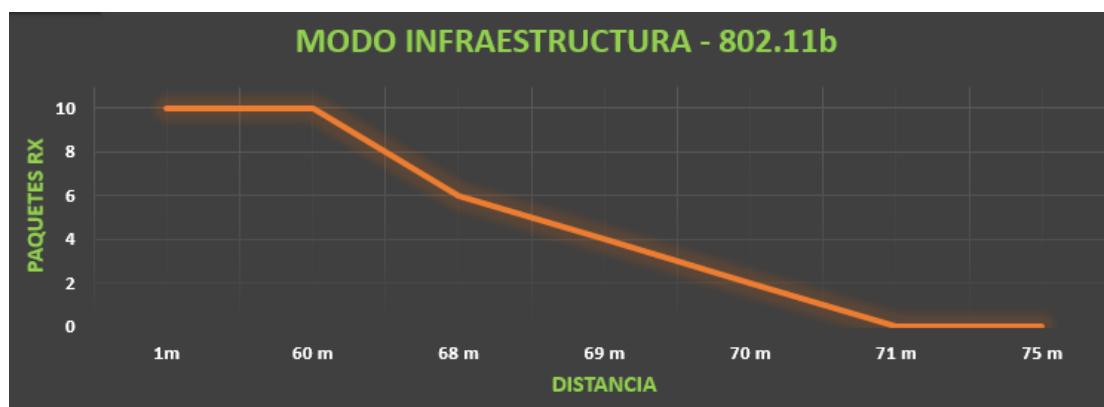


Figura 74 Comportamiento de la red en función de la variación de la distancia.

Fuente: Autor



Figura 75 Configuración modo infraestructura, estaciones separadas 68 m

Fuente: Autor



Figura 76 Configuración modo infraestructura, estaciones separadas 71

Fuente: Autor

6.2.1.2 Modo Infraestructura – IEEE 802.11g

La figura 77, muestra los archivos .pcap generados, para una red configurada en modo infraestructura.

anim1	21/3/2017 20:19	Documento XML	63 KB
infra_ap-0-0	21/3/2017 20:19	Wireshark capture...	52 KB
infra_estaciones-1-0	21/3/2017 20:19	Wireshark capture...	69 KB
infra_estaciones-2-0	21/3/2017 20:19	Wireshark capture...	29 KB

Figura 77 Archivos generados – Modo infraestructura

Fuente: Autor

Las figuras, 78 , 79 y 80 muestran, los paquetes capturados y la interpretación del paquete seleccionado, como se puede observar se ha enviado, se observa que el AP envía beacons para anunciar que la red está trasmitiendo, además las estaciones envían solicitudes al AP para poder asociarse a él. Se realizan las consultas ARP y luego se envía los datos desde la estación al AP. También se puede observar tanto el estándar configurado como la velocidad de transmisión de datos.

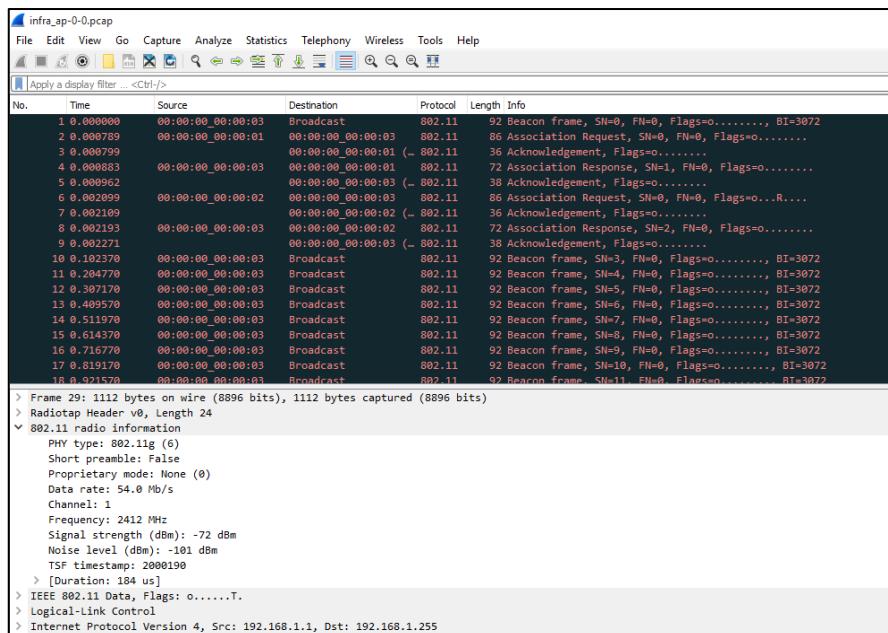


Figura 78 Resultado archivo .pcap – Modo Infraestructura - AP

Fuente: Autor

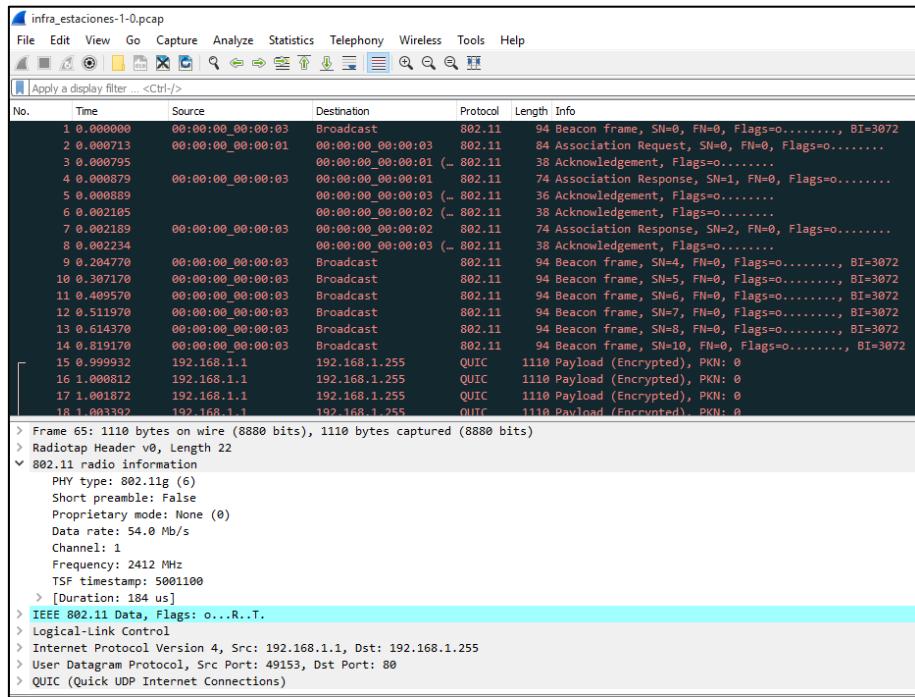


Figura 79 Resultado archivo .pcap – Modo infraestructura – Estación 1
Fuente: Autor

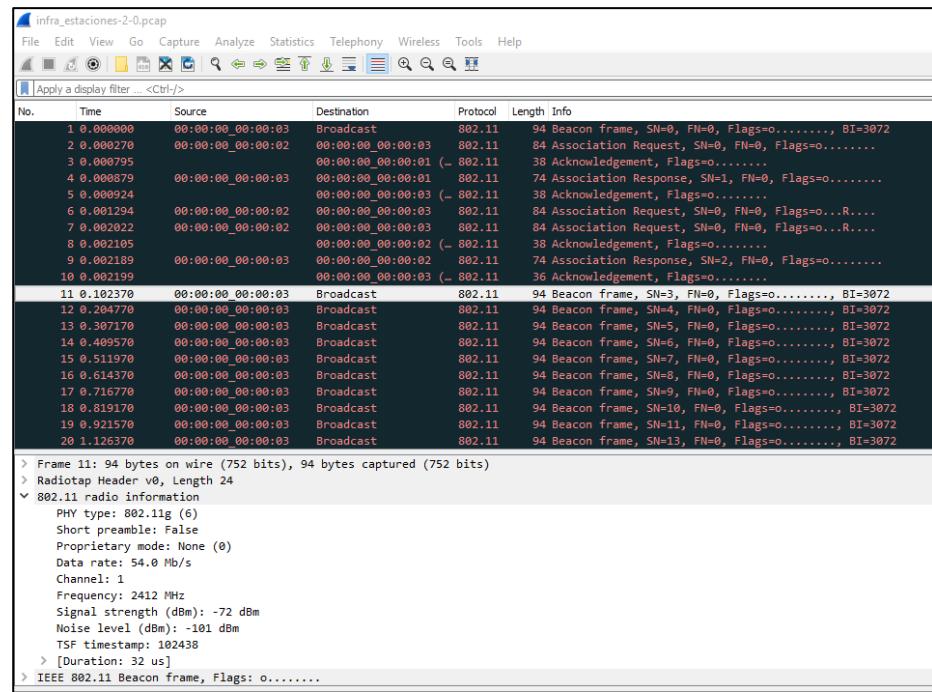


Figura 80 Resultado archivo .pcap – Modo infraestructura – Estación 2
Fuente: Autor

La figura 81 muestra el resultado al configurar una red en modo infraestructura con el estándar IEEE 802.11g, las estaciones se encuentran ubicadas a 20m del AP (véase figura 82). Se observa que a esta distancia de 10 paquetes enviados se reciben 10.

```
root@eli-Inspiron-5559:~/ns3/ns-allinone-3.22/ns-3.22# ./waf --run infraestructura-g
Waf: Entering directory '/root/ns3/ns-allinone-3.22/ns-3.22/build'
Waf: Leaving directory '/root/ns3/ns-allinone-3.22/ns-3.22/build'
'build' finished successfully (1.484s)
  Prueba, envia 10 paquetes, distancia de 20m
PAQUETE RECIBIDO
root@eli-Inspiron-5559:~/ns3/ns-allinone-3.22/ns-3.22#
```

Figura 81 Resultado paquetes enviados.

Fuente: Autor

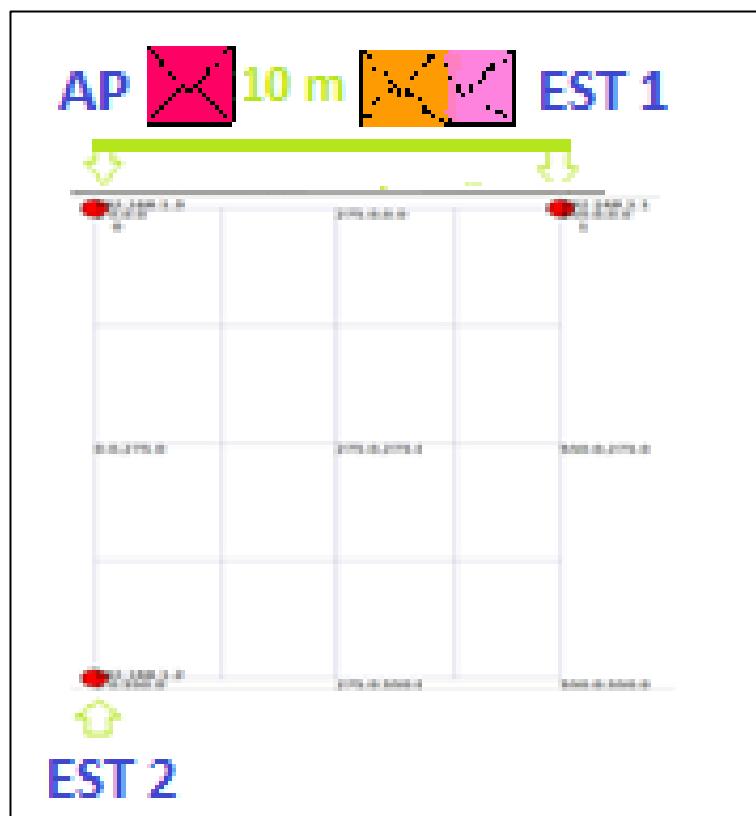


Figura 82 Configuración modo infraestructura, estaciones separadas 10 m
Fuente: Autor

La figura 83 muestra el comportamiento de la red al variar la distancia entre la estación y el AP. Mientras mayor sea la distancia menor será el número de paquetes transmitidos (véase figura 84 y figura 85).

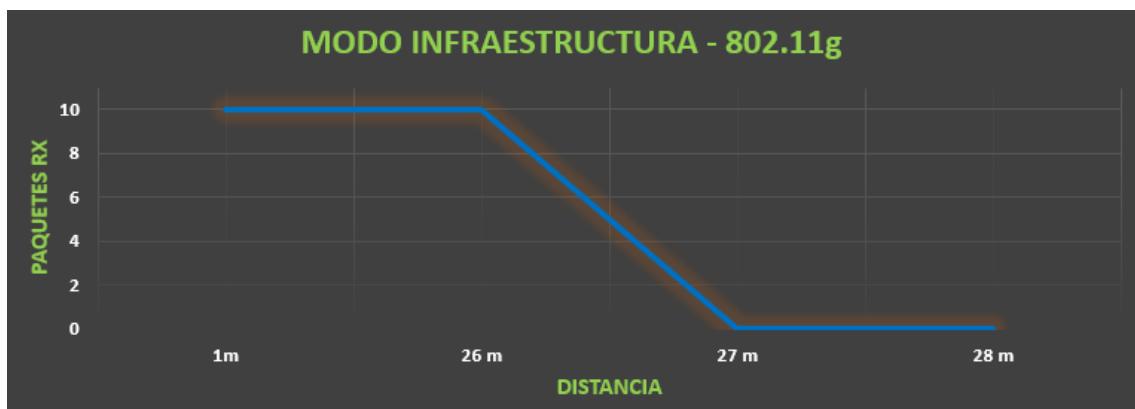


Figura 83 Comportamiento de la red en función de la distancia.
Fuente: Autor



Figura 84 Configuración modo infraestructura, estaciones separadas 26 m
Fuente: Autor



Figura 85 Configuración modo infraestructura, estaciones separadas 28 m

Fuente: Autor

6.2.2 Modo AD-HOC

6.2.2.1 Modo AD-HOC – IEEE 802.11b

La figura 86 muestra el resultado de la herramienta PyViz.

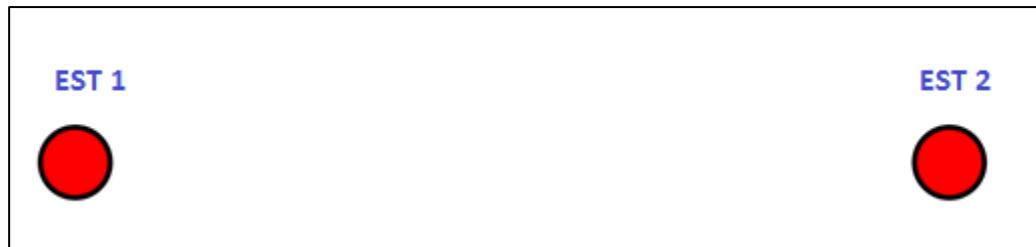


Figura 86 Modo Ad –Hoc – Herramienta PyViz

Fuente: Autor

La figura 86 se encuentra compuesta por dos nodos. Los nodos representan dos estaciones que están configurados con el estándar 802.11b modo Ad-Hoc.

La figura 87 muestra los archivos .pcap generados, para una red configurada en modo Ad-Hoc.

Nombre	Fecha de modificación	Tipo	Tamaño
adhoc-0-0	23/3/2017 19:28	Wireshark capture...	12 KB
adhoc-1-0	23/3/2017 19:28	Wireshark capture...	12 KB
adhoc-b	23/3/2017 19:28	Archivo TR	11 KB
anim1	23/3/2017 19:28	Documento XML	2 KB

Figura 87 Archivos generados – Modo Ad - Hoc

Fuente: Autor

Para las figuras siguientes se emplea un visualizador de simulación *NetAnime*.

La Figura 88 muestra como viajan los paquetes de la estación 1 a la estación 0.



Figura 88 Envió de paquete – Modo Ad-Hoc - IEEE 802.11b

Fuente: Autor

La figura 89 muestra la secuencia de los 10 paquetes enviados desde la estación 2 a la estación 1 en los diferentes slots de tiempos.

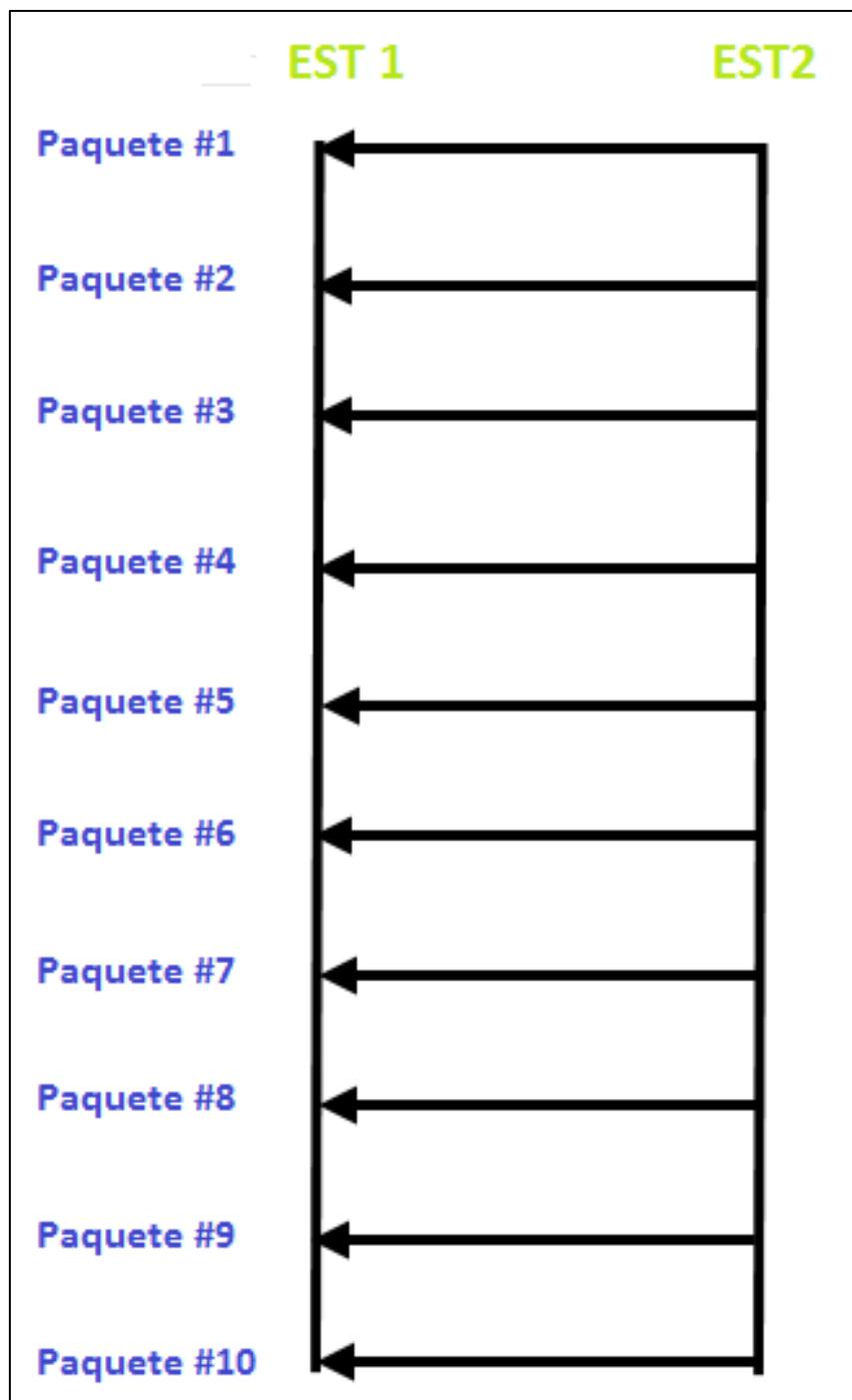


Figura 89 Diagrama de paquetes enviados – Modo Ad-Hoc - IEEE 802.11b.

Fuente: Autor

La simulación arrojó como resultado un archivo de trazas .tr como el de la figura 90, cada línea mostrada corresponde a un evento.

Figura 90 Resultado de archivo . tr – Modo Ad – Hoc –IEEE 802.11b

Fuente: Autor

La figura 91 y 92 lista los paquetes capturados y la interpretación del paquete seleccionado, se puede observar el estándar configurado como la velocidad de transmisión de datos.

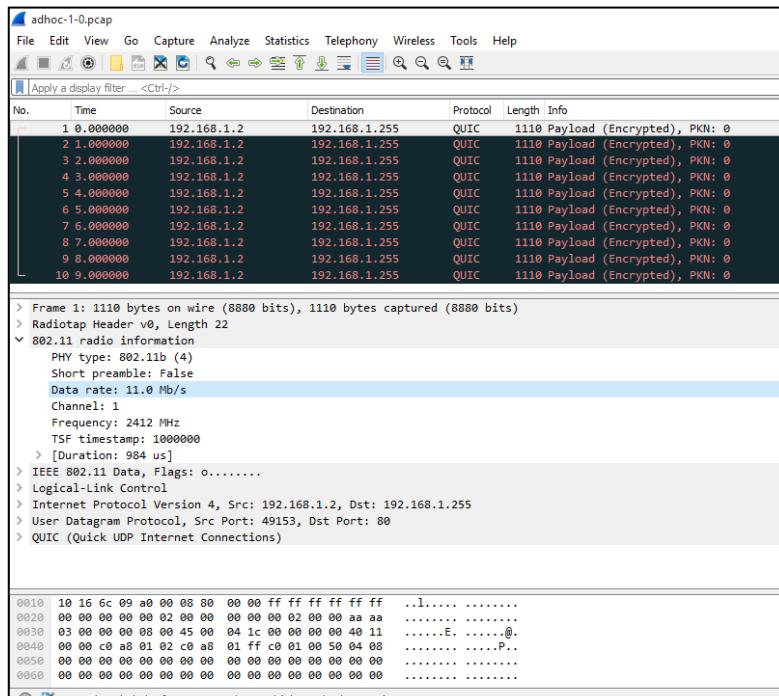


Figura 91 Resultados de archivo .pcap – Modo Ad – Hoc -Estación 1- IEEE 802.11b
Fuente: Autor

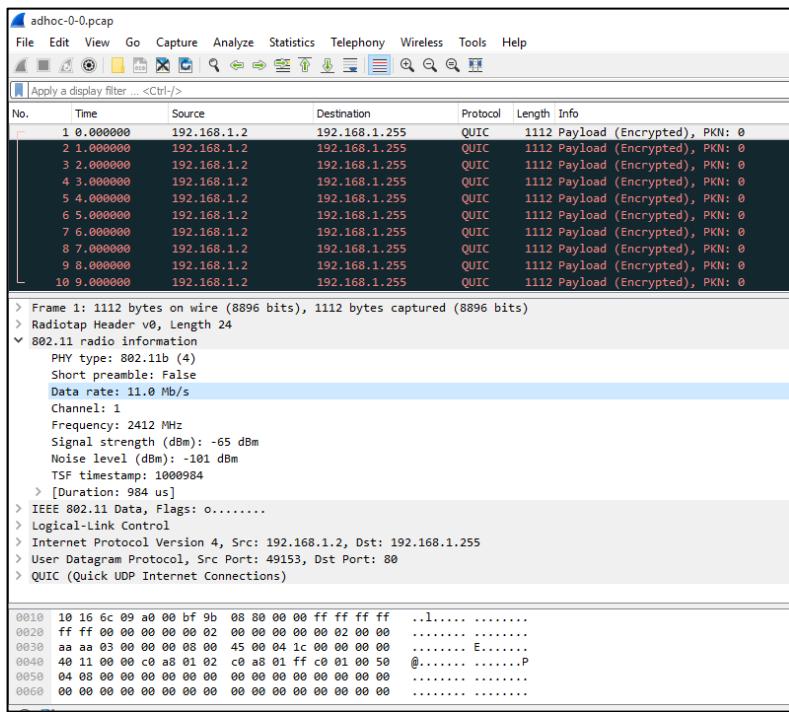


Figura 92 Resultado archivo .pcap – Modo Ad – Hoc - Estación 2-- IEEE 802.11b
Fuente: Autor

La figura 93 muestra el resultado al configurar una red en modo Ad-Hoc con el estándar IEEE 802.11b, las estaciones se encuentran ubicadas a 50m del AP (véase figura 94). Se observa que a esta distancia de 10 paquetes enviados se reciben 10.

```
root@eli-Inspiron-5559:~/ns3/ns-allinone-3.22/ns-3.22# ./waf --run adhoc-2nodos-b
Waf: Entering directory '/root/ns3/ns-allinone-3.22/ns-3.22/build'
[ 865/2465] cxx: scratch/infraestructura-g.cc -> build/scratch/infraestructura-g.cc.7.o
[2454/2465] cxxprogram: build/scratch/infraestructura-g.cc.7.o -> build/scratch/infraestructura-g
Waf: Leaving directory '/root/ns3/ns-allinone-3.22/ns-3.22/build'
'build' finished successfully (2.727s)
Prueba desde el nodo 1 hacia el nodo 0, n mero de paquetes enviados: 10 , distancia entre los nodos: 50m
PAQUETE RECIBIDO!
root@eli-Inspiron-5559:~/ns3/ns-allinone-3.22/ns-3.22# 
```

Figura 93 Resultado paquetes enviados – Modo Ad-Hoc - IEEE 802.11b.
Fuente: Autor

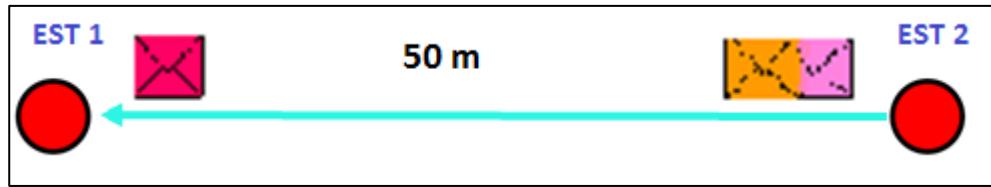


Figura 94 Configuración modo ad-hoc. Estaciones separadas 50 m

Fuente: Autor

La figura 95 muestra el comportamiento de la red al variar la distancia entre los nodos. Mientras mayor sea la distancia menor será el número de paquetes transmitidos (véase figura 96 y figura 97).

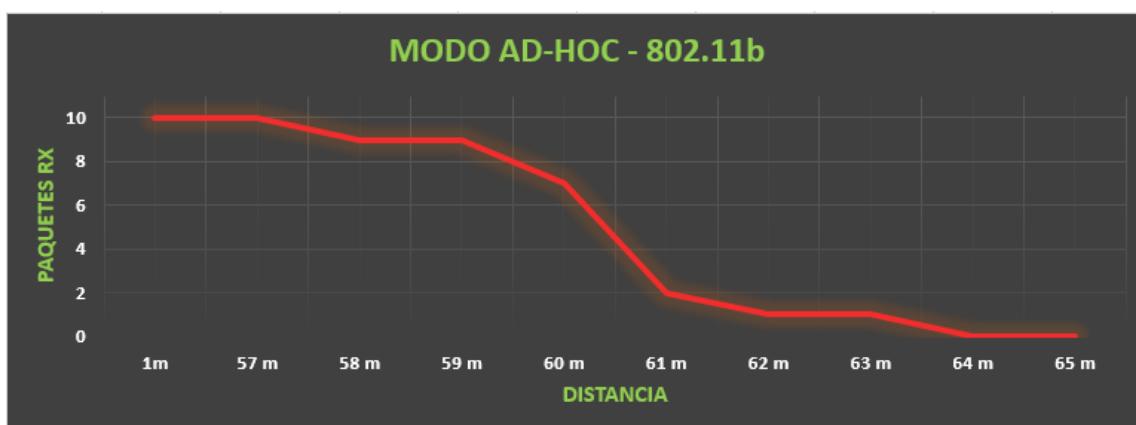


Figura 95 Comportamiento de la red en función de la distancia – Modo Ad-Hoc - IEEE 802.11b.

Fuente: Autor

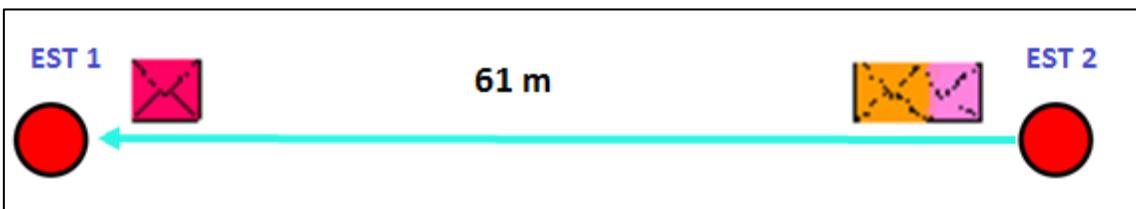


Figura 96 Configuración modo ad-hoc. Estaciones separadas 61 m – IEEE 802.11b

Fuente: Autor

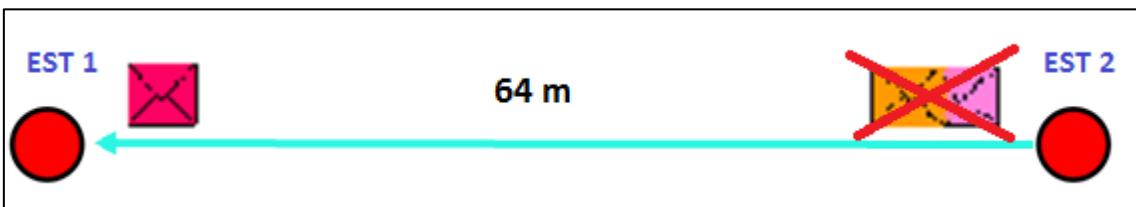


Figura 97 Configuración modo ad-hoc. Estaciones separadas 64 m – IEEE 802.11b.

Fuente: Autor

6.2.2.2 Modo AD-HOC – IEEE 802.11g

La figura 98, muestra los archivos .pcap generados, para una red configurada en modo Ad-Hoc

adhoc-0-0	23/3/2017 19:28	Wireshark capture...	12 KB	
adhoc-1-0	23/3/2017 19:28	Wireshark capture...	12 KB	
adhoc-g	23/3/2017 19:28	Archivo TR	11 KB	
anim1	23/3/2017 19:28	Documento XML	2 KB	

Figura 98 Archivos generados – Modo Ad –Hoc –IEEE 802.11g.

Fuente: Autor

La simulación de la red configurada como modo Ad – Hoc arrojó como resultado un archivo de trazas .tr como el de la figura 99, cada línea mostrada corresponde a un evento.

```
t 1 /NodeList/1/DeviceList/0/$ns3::WifiNetDevice/Phy/State/Tx ns3::WifiMacHeader (DATA ToDS=0, FromDS=0, MoreFrag=0, Retry=0, MoreData=0
Duration/ID=0usDA=ffff:ffff:ffff:ff, SA=00:00:00:00:00:02, BSSID=00:00:00:00:00:02, FragNumber=0, SeqNumber=0) ns3::LlcSnapHeader (type 0x800) ns3::Ipv4Header (tos 0x0
DSCP Default ECN Not-ECT ttl 64 id 0 protocol 17 offset (bytes) 0 flags [none] length: 1052 192.168.1.2 > 192.168.1.255) ns3::UdpHeader (length: 1032 49153 > 80)
Payload (size=1024) ns3::WifiMacTrailer ()r 1.00019 /NodeList/0/DeviceList/0/$ns3::WifiNetDevice/Phy/State/Rx0k ns3::WifiMacHeader (DATA ToDS=0, FromDS=0, MoreFrag=0,
Retry=0, MoreData=0 Duration/ID=0usDA=ffff:ffff:ffff:ff, BSSID=00:00:00:00:00:02, FragNumber=0, SeqNumber=0) ns3::LlcSnapHeader (type 0x800)
ns3::Ipv4Header (tos 0x0 DSCP Default ECN Not-ECT ttl 64 id 0 protocol 17 offset (bytes) 0 flags [none] length: 1052 192.168.1.255) ns3::UdpHeader (length: 1032 49153 > 80)
Payload (size=1024) ns3::WifiMacTrailer ()r 2.00019 /NodeList/0/DeviceList/0/$ns3::WifiNetDevice/Phy/State/Tx ns3::WifiMacHeader (DATA ToDS=0,
FromDS=0, MoreFrag=0, Retry=0, MoreData=0 Duration/ID=0usDA=ffff:ffff:ffff:ff, SA=00:00:00:00:00:02, BSSID=00:00:00:00:00:02, FragNumber=0, SeqNumber=1)
ns3::LlcSnapHeader (type 0x800) ns3::Ipv4Header (tos 0x0 DSCP Default ECN Not-ECT ttl 64 id 1 protocol 17 offset (bytes) 0 flags [none] length: 1052 192.168.1.2 >
192.168.1.255) ns3::UdpHeader (length: 1032 49153 > 80) Payload (size=1024) ns3::WifiMacTrailer ()r 2.00019 /NodeList/0/DeviceList/0/$ns3::WifiNetDevice/Phy/State/Rx0k
ns3::WifiMacHeader (DATA ToDS=0, FromDS=0, MoreFrag=0, Retry=0, MoreData=0 Duration/ID=0usDA=ffff:ffff:ffff:ff, SA=00:00:00:00:00:02, BSSID=00:00:00:00:00:02,
FragNumber=0, SeqNumber=1) ns3::LlcSnapHeader (type 0x800) ns3::Ipv4Header (tos 0x0 DSCP Default ECN Not-ECT ttl 64 id 1 protocol 17 offset (bytes) 0 flags [none]
length: 1052 192.168.1.2 > 192.168.1.255) ns3::UdpHeader (length: 1032 49153 > 80) Payload (size=1024) ns3::WifiMacTrailer ()t 3 /NodeList/1/DeviceList/0/
$ns3::WifiNetDevice/Phy/State/Tx ns3::WifiMacHeader (DATA ToDS=0, FromDS=0, MoreFrag=0, Retry=0, MoreData=0 Duration/ID=0usDA=ffff:ffff:ffff:ff, SA=00:00:00:00:00:02,
BSSID=00:00:00:00:00:02, FragNumber=2) ns3::LlcSnapHeader (type 0x800) ns3::Ipv4Header (tos 0x0 DSCP Default ECN Not-ECT ttl 64 id 2 protocol 17 offset
(bytes) 0 flags [none] length: 1052 192.168.1.2 > 192.168.1.255) ns3::UdpHeader (length: 1032 49153 > 80) Payload (size=1024) ns3::WifiMacTrailer ()r 3.00019
/NodeList/0/DeviceList/0/$ns3::WifiNetDevice/Phy/State/Rx0k ns3::WifiMacHeader (DATA ToDS=0, FromDS=0, MoreFrag=0, Retry=0, MoreData=0
Duration/ID=0usDA=ffff:ffff:ffff:ff, SA=00:00:00:00:00:02, FragNumber=2) ns3::LlcSnapHeader (type 0x800) ns3::Ipv4Header (tos 0x0 DSCP Default ECN Not-ECT ttl 64 id 2 protocol 17 offset
(bytes) 0 flags [none] length: 1052 192.168.1.2 > 192.168.1.255) ns3::UdpHeader (length: 1032 49153 > 80) Payload (size=1024) ns3::WifiMacTrailer ()r 4.00019 /NodeList/0/DeviceList/0/
$ns3::WifiNetDevice/Phy/State/Tx ns3::WifiMacHeader (DATA ToDS=0, FromDS=0, MoreFrag=0, Retry=0, MoreData=0 Duration/ID=0usDA=ffff:ffff:ffff:ff, SA=00:00:00:00:00:02,
BSSID=00:00:00:00:00:02, FragNumber=3) ns3::LlcSnapHeader (type 0x800) ns3::Ipv4Header (tos 0x0 DSCP Default ECN Not-ECT ttl 64 id 3 protocol 17 offset (bytes) 0 flags [none] length: 1052 192.168.1.2 >
192.168.1.255) ns3::UdpHeader (length: 1032 49153 > 80) Payload (size=1024) ns3::WifiMacTrailer ()r 5 /NodeList/0/$ns3::WifiNetDevice/Phy/State/Tx
ns3::WifiMacHeader (DATA ToDS=0, MoreFrag=0, Retry=0, MoreData=0 Duration/ID=0usDA=ffff:ffff:ffff:ff, SA=00:00:00:00:00:02, BSSID=00:00:00:00:00:02,
FragNumber=4) ns3::LlcSnapHeader (type 0x800) ns3::Ipv4Header (tos 0x0 DSCP Default ECN Not-ECT ttl 64 id 4 protocol 17 offset (bytes) 0 flags [none]
length: 1052 192.168.1.2 > 192.168.1.255) ns3::UdpHeader (length: 1032 49153 > 80) Payload (size=1024) ns3::WifiMacTrailer ()r 6 /NodeList/1/DeviceList/0/$ns3::WifiNetDevice/Phy/State/Tx
ns3::WifiMacHeader (DATA ToDS=0, MoreFrag=0, Retry=0, MoreData=0 Duration/ID=0usDA=ffff:ffff:ffff:ff, SA=00:00:00:00:00:02, BSSID=00:00:00:00:00:02,
FragNumber=5) ns3::LlcSnapHeader (type 0x800) ns3::Ipv4Header (tos 0x0 DSCP Default ECN Not-ECT ttl 64 id 5 protocol 17 offset (bytes) 0 flags [none] length: 1052 192.168.1.2 > 192.168.1.255) ns3::UdpHeader (length: 1032 49153 > 80)
Payload (size=1024) ns3::WifiMacTrailer ()r 6.00019 /NodeList/0/DeviceList/0/$ns3::WifiNetDevice/Phy/State/Rx0k ns3::WifiMacHeader (DATA ToDS=0, FromDS=0, MoreFrag=0,
Retry=0, MoreData=0 Duration/ID=0usDA=ffff:ffff:ffff:ff, SA=00:00:00:00:00:02, BSSID=00:00:00:00:00:02, FragNumber=5) ns3::LlcSnapHeader (type 0x800)
```

Figura 99 Resultado archivo .tr – Modo Ad-Hoc –IEEE 802.11g.

Fuente: Autor

Las figuras 100 y 101 lista los paquetes capturados y la interpretación del paquete seleccionado, se puede observar el estándar configurado como la velocidad de transmisión de datos.

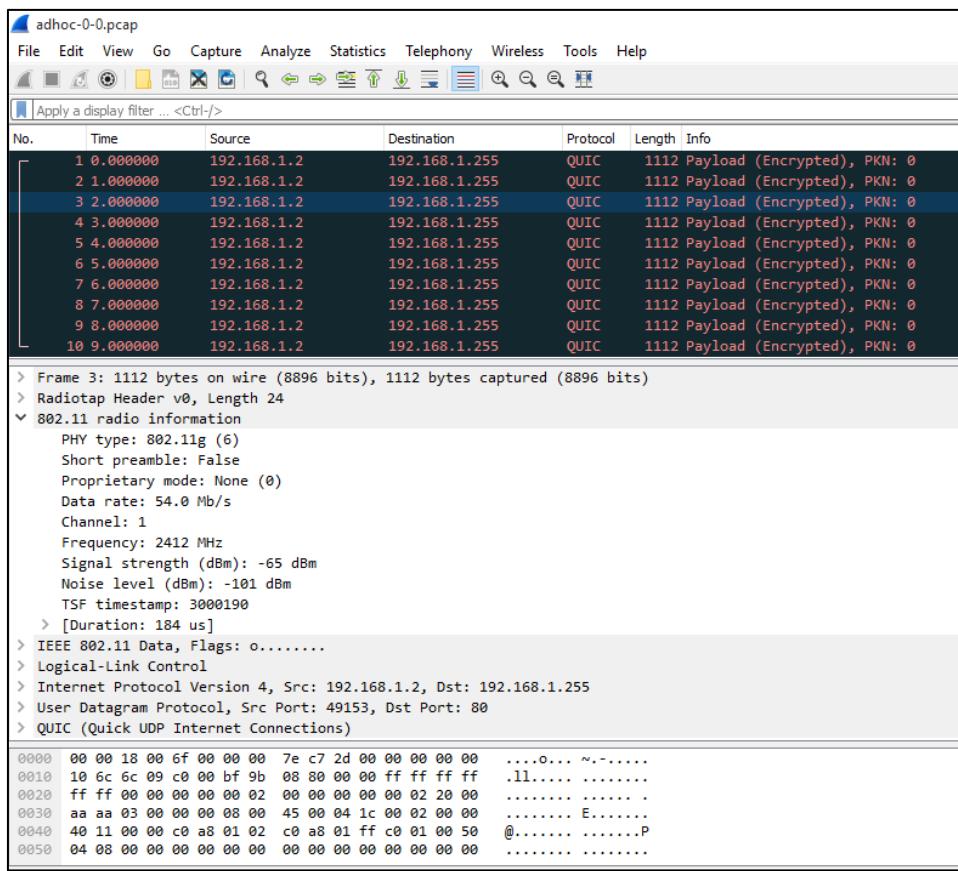


Figura 100 Resultado archivo .pcap – Modo Ad – Hoc – Estación 1 – IEEE 802.11g.

Fuente: Autor

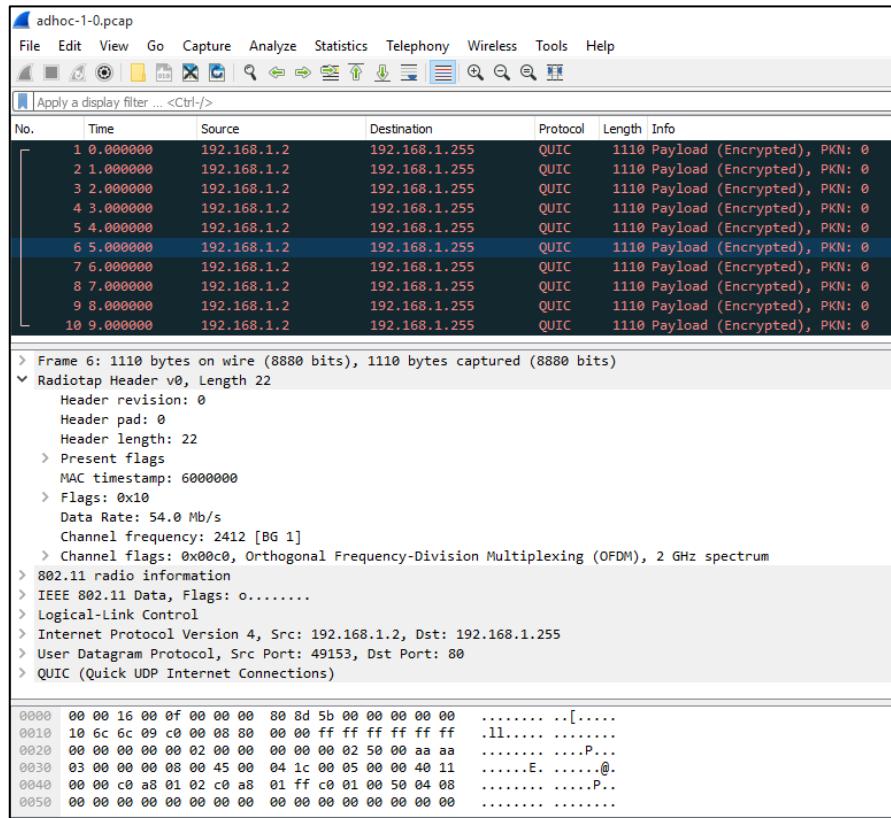


Figura 101 Resultado archivo .pcap – Modo Ad – Hoc – Estación 2 – IEEE 802.11g.

Fuente: Autor

La figura 102 muestra el resultado al configurar una red en modo Ad-Hoc con el estándar IEEE 802.11g, las estaciones se encuentran ubicadas a 22m del AP (véase figura 103). Se observa que a esta distancia de 10 paquetes enviados se reciben 10.

```

root@eli-Inspiron-5559:~/ns3/ns-allinone-3.22/ns-3.22# ./waf --run adhoc-2nodos-g
Waf: Entering directory '/root/ns3/ns-allinone-3.22/ns-3.22/build'
Waf: Leaving directory '/root/ns3/ns-allinone-3.22/ns-3.22/build'
'build' finished successfully (1.492s)
Prueba desde el nodo 1 hacia el nodo 0, número de paquetes enviados: 10 , distancia entre los nodos: 22m
PAQUETE RECIBIDO!
root@eli-Inspiron-5559:~/ns3/ns-allinone-3.22/ns-3.22# 

```

Figura 102 Resultado paquetes enviados – Modo Ad –Hoc –IEEE 802.11g.

Fuente: Autor

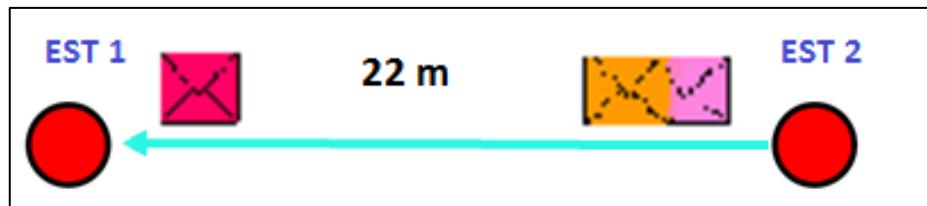


Figura 103 Configuración modo ad-hoc. Estaciones separadas 22 m – Modo Ad –Hoc –IEEE 802.11g.
Fuente: Autor

La figura 104 muestra el comportamiento de la red al variar la distancia entre los nodos. Mientras mayor sea la distancia menor será el número de paquetes transmitidos (véase figura 105 y figura 106).

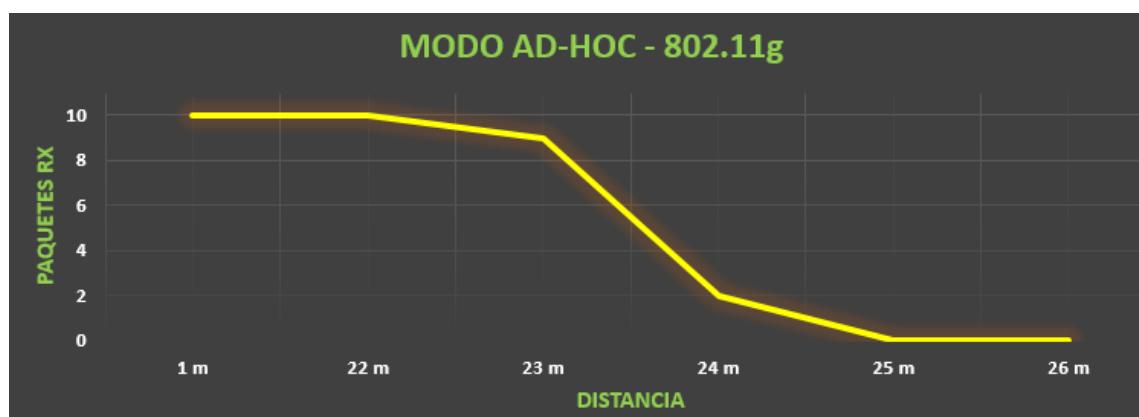


Figura 104 Comportamiento de la red en función de la distancia – Modo Ad –Hoc –IEEE 802.11g.
Fuente: Autor

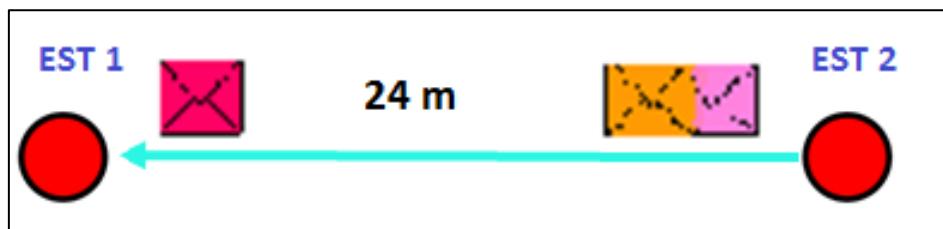


Figura 105 Configuración modo ad-hoc. Estaciones separadas 24 m – Modo Ad –Hoc –IEEE 802.11g.
Fuente: Autor

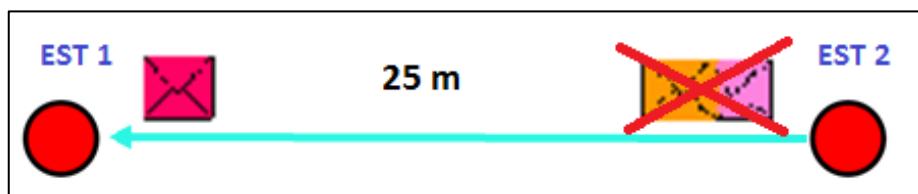


Figura 106 Configuración modo ad-hoc. Estaciones separadas 25 m – Modo Ad –Hoc –IEEE 802.11g.
Fuente: Autor

6.3 Estándar IEEE 802.11s

El estándar 802.11s ha sido configurado como una red mesh nxm, en la cual se puede variar el número de nodos, la distancia entre los nodos, tiempo total de la simulación, el tamaño de los paquetes, el intervalo en el que se envían los paquetes entre otras características. La figura 107 muestra todos los archivos que se generan al correr el script.

anim	21/3/2017 20:47	Documento XML	6.265 KB	
mesh	21/3/2017 20:47	Documento XML	3 KB	
REPOR--0-1	21/3/2017 20:47	Wireshark capture...	2.869 KB	
REPOR--1-1	21/3/2017 20:47	Wireshark capture...	2.946 KB	
REPOR--2-1	21/3/2017 20:47	Wireshark capture...	2.839 KB	
REPOR--3-1	21/3/2017 20:47	Wireshark capture...	2.873 KB	
reporte-Oen el tiempo+0.0ns	21/3/2017 20:47	Documento XML	2 KB	
reporte-Oen el tiempo+100000001.0ns	21/3/2017 20:47	Documento XML	3 KB	
reporte-Oen el tiempo+300000003.0ns	21/3/2017 20:47	Documento XML	3 KB	
reporte-Oen el tiempo+600000014.0ns	21/3/2017 20:47	Documento XML	3 KB	
reporte-Oen el tiempo+1000000019.0ns	21/3/2017 20:47	Documento XML	3 KB	
reporte-Oen el tiempo+1500000019.0ns	21/3/2017 20:47	Documento XML	3 KB	
reporte-Oen el tiempo+2100000042.0ns	21/3/2017 20:47	Documento XML	3 KB	
reporte-Oen el tiempo+2800000089.0ns	21/3/2017 20:47	Documento XML	3 KB	
reporte-Oen el tiempo+3600000160.0ns	21/3/2017 20:47	Documento XML	3 KB	
reporte-Oen el tiempo+4500000255.0ns	21/3/2017 20:47	Documento XML	3 KB	
reporte-Oen el tiempo+5500000374.0ns	21/3/2017 20:47	Documento XML	3 KB	
reporte-Oen el tiempo+6600000517.0ns	21/3/2017 20:47	Documento XML	3 KB	
reporte-Oen el tiempo+7800000683.0ns	21/3/2017 20:47	Documento XML	3 KB	
reporte-Oen el tiempo+9100000873.0ns	21/3/2017 20:47	Documento XML	3 KB	
reporte-Oen el tiempo+10500001087.0ns	21/3/2017 20:47	Documento XML	3 KB	
reporte-Oen el tiempo+12000001325.0ns	21/3/2017 20:47	Documento XML	3 KB	
reporte-Oen el tiempo+13600001587.0ns	21/3/2017 20:47	Documento XML	3 KB	
reporte-Oen el tiempo+15300001873.0ns	21/3/2017 20:47	Documento XML	3 KB	

Figura 107 Archivos generados – Red Mesh

Fuente: Autor

La figura 108, muestra uno de los archivos .pcap generados, los archivos generados va depender del número de nodos de la que está compuesta la malla. . Este tipo de archivos son analizados mediante *Wireshark*.

No.	Time	Source	Destination	Protocol	Length	Info
1	0:00:00:00:00:00:02	Broadcast		802.11	60	Beacon frame, SN=0, FN=0, Flags=o....., BI=0
2	0:00:00:00:00:01	00:00:00:00:00:02		802.11	82	Action, SN=0, FN=0, Flags=o....., SSID=Broadcast
3	0:00:00:00:00:03	00:00:00:00:00:02		802.11	82	Action, SN=0, FN=0, Flags=o...R..., SSID=Broadcast
4	0:00:00:00:00:04			802.11	14	Acknowledgement, Flags=o.....
5	0:00:00:00:00:02	00:00:00:00:00:03		802.11	80	Action, SN=1, FN=0, Flags=o....., SSID=Broadcast
6	0:00:00:00:00:02			802.11	14	Acknowledgement, Flags=o.....
7	0:00:00:00:00:02	00:00:00:00:00:03		802.11	82	Action, SN=2, FN=0, Flags=o....., SSID=Broadcast
8	0:00:00:00:00:02	00:00:00:00:00:02		802.11	14	Acknowledgement, Flags=o.....
9	0:00:00:00:00:03	00:00:00:00:00:02		802.11	80	Action, SN=1, FN=0, Flags=o....., SSID=Broadcast
10	0:00:00:00:00:01	00:00:00:00:00:02		802.11	82	Action, SN=0, FN=0, Flags=o...R..., SSID=Broadcast
11	0:00:00:00:00:04			802.11	14	Acknowledgement, Flags=o.....
12	0:00:00:00:00:02	00:00:00:00:00:01		802.11	80	Action, SN=3, FN=0, Flags=o....., SSID=Broadcast
13	0:00:00:00:00:02			802.11	14	Acknowledgement, Flags=o.....
14	0:00:00:00:00:02	00:00:00:00:00:01		802.11	82	Action, SN=4, FN=0, Flags=o....., SSID=Broadcast
15	0:00:00:00:00:02	00:00:00:00:00:02		802.11	14	Acknowledgement, Flags=o.....
16	0:00:00:00:00:01	00:00:00:00:00:02		802.11	80	Action, SN=1, FN=0, Flags=o....., SSID=Broadcast
17	0:00:00:00:00:01	00:00:00:00:00:01		802.11	82	Action, SN=0, FN=0, Flags=o...R..., SSID=Broadcast
18	0:00:00:00:00:04			802.11	14	Acknowledgement, Flags=o.....
19	0:00:00:00:00:02	00:00:00:00:00:04		802.11	80	Action, SN=5, FN=0, Flags=o....., SSID=Broadcast
20	0:00:00:00:00:02	00:00:00:00:00:02		802.11	14	Acknowledgement, Flags=o.....
21	0:00:00:00:00:02	00:00:00:00:00:04		802.11	82	Action, SN=6, FN=0, Flags=o....., SSID=Broadcast
22	0:00:00:00:00:02	00:00:00:00:00:02		802.11	14	Acknowledgement, Flags=o.....
23	0:00:00:00:00:04	00:00:00:00:00:02		802.11	80	Action, SN=1, FN=0, Flags=o....., SSID=Broadcast
24	0:00:00:00:00:04	00:00:00:00:00:04		802.11	14	Acknowledgement, Flags=o.....
25	0:00:00:00:00:03	00:00:00:00:00:02		802.11	80	Action, SN=1, FN=0, Flags=o...R..., SSID=Broadcast
26	0:00:00:00:00:03	00:00:00:00:00:03		802.11	14	Acknowledgement, Flags=o.....
27	0:00:00:00:00:04	Broadcast		LLC	76	S, func=RR, N(R)=0; DSAP NULL LSAP Individual, SSAP ISO Network Layer (unofficial?) Command
28	0:00:00:00:00:04	Broadcast		LLC	76	S, func=RR, N(R)=0; DSAP NULL LSAP Individual, SSAP 0x1e Response
29	0:00:00:00:00:04	Broadcast		LLC	76	S, func=RR, N(R)=0; DSAP NULL LSAP Individual, SSAP 0x1e Command
30	0:00:00:00:00:01	Broadcast		802.11	69	Action, SN=3, FN=0, Flags=o....., SSID=Broadcast

> Frame 1: 60 bytes on wire (480 bits), 60 bytes captured (480 bits)
> IEEE 802.11 Beacon frame, Flags: o.....
> IEEE 802.11 wireless LAN management frame

Figura 108 Resultado archivo .pcap – Red Mesh

Fuente: Autor

Para un mejor entendimiento de este estándar se imprimen varios reportes durante toda la simulación. La figura 109 muestra el esquema general del reporte que se imprime por nodo, y cada cierto intervalo de tiempo, en este se puede observar el número de vínculos hacia otros nodos, la métrica, entre otros campos.

```

<MeshPointDevice time="0:1" address="00:00:00:00:00:01">
  <Statistics txUnicastData="2" txBroadcastData="1080" txBroadcastDataBytes="28" rxUnicastData="2" rxBroadcastData="1080" rxBroadcastDataBytes="28" rxBroadcastDataBytes="28" fwdUnicastData="0" fwdBroadcastData="1" fwdBroadcastDataBytes="28" />
  -<Interface BeaconInterval="0.5" Channel="0" Address="00:00:00:00:00:01">
    <Statistics rxBeacons="3" txFrames="11" rxFrames="1551" rxBytes="1711"/>
  </Interface>
  -<Hwmp address="00:00:00:00:00:01" maxQueueSize="255" Dot11MeshHWMPmaxPREQtries="3" Dot11MeshHWMPnetDiameterTraversalTime="0.1024" Dot11MeshHWMPpreqMinInterval="0.1024" Dot11MeshHWMPperrMinInterval="0.1024" Dot11MeshHWMPactiveRootTimeout="5.12" Dot11MeshHWMPactivePathTimeout="5.12" Dot11MeshHWMPpathToRootInterval="2.048" Dot11MeshHWMPrrnInterval="5.12" isRoot="0" maxTd="32" unicastPerrThreshold="32" unicastPreqThreshold="1" unicastDataThreshold="1" doFlag="0" rfflag="1">
    <Statistics txUnicast="2" txBroadcast="1" txBytes="1144" droppedTd="0" totalQueued="1" totalDropped="0" initiatedPreq="1" initiatedPrep="0" initiatedPerr="0"/>
    -<HwmpProtocolMac address="00:00:00:00:00:01">
      <Statistics txPreq="1" txPrep="0" txPerr="0" rxPreq="1" rxPrep="1" rxPerr="0" txMgt="1" txMgtBytes="41" rxMgt="2" rxMgtBytes="76" txData="4" txDataBytes="1168" rxData="4" rxDataBytes="1168"/>
    </HwmpProtocolMac>
  </Hwmp>
  -<PeerManagementProtocol>
    <Statistics linksTotal="3" linksOpened="3" linksClosed="0"/>
    -<PeerManagementProtocolMac address="00:00:00:00:00:01">
      <Statistics txOpen="3" txConfirm="3" txClose="0" rxOpen="3" rxConfirm="3" rxClose="0" dropped="0" brokenMgt="0" txMgt="6" txMgtBytes="318" rxMgt="6" rxMgtBytes="306" beaconShift="0"/>
    </PeerManagementProtocolMac>
    <PeerLink localAddress="00:00:00:00:00:01" peerInterfaceAddress="00:00:00:00:00:02" peerMeshPointAddress="00:00:00:00:02" metrica="151" lastBeacon="0.00491789" localLinkId="1" peerLinkId="2" assocId="0"/>
    <PeerLink localAddress="00:00:00:00:00:01" peerInterfaceAddress="00:00:00:00:00:04" peerMeshPointAddress="00:00:00:00:04" metrica="153" lastBeacon="0.0330582" localLinkId="2" peerLinkId="3" assocId="1"/>
    <PeerLink localAddress="00:00:00:00:00:01" peerInterfaceAddress="00:00:00:00:00:03" peerMeshPointAddress="00:00:00:00:03" metrica="150" lastBeacon="0.0634322" localLinkId="3" peerLinkId="3" assocId="2"/>
  </PeerManagementProtocol>
</MeshPointDevice>
```

Figura 109 Reporte para el MP: 00:00:00:00:00:01

Fuente: Autor

La red se encuentra configurada en un modo reactivo, la tabla de vecinos se crea antes de la transmisión de los datos, para enviar los paquetes de un nodo fuente a un nodo destino, primero el nodo fuente envía un mensaje (PREQ) para solicitar la ruta de sus vecinos y guardar los valores de la métrica, los nodos que reciben el mensaje PREQ envían un mensaje PREP al nodo fuente, una vez que el nodo fuente recibe los PREP tendrá completa toda la información que requiere para realizar el envío, esta información contiene toda la trayectoria hacia su destino (métrica) y en base a dicha información envía el paquete por la trayectoria más eficiente.

Para una malla de 2x2, con una distancia entre nodos de 100m, la figura 110 muestra una parte del archivo .xml, donde se comprueba tanto el envío del mensaje como la recepción de los mensajes PREQ y PREP.

```
▼ <HwmpProtocolMac address="00:00:00:00:00:01">
  <Statistics txPreq="1" txPrep="0" txPerr="0" rxPreq="1" rxPrep="1">
```

Figura 110 Mensajes PREQ y PREP

Fuente: Autor

Una vez recibido el mensaje PREP se crea las posibles rutas desde el nodo fuente al nodo destino, la figura 111 muestra otra parte del archivo .xml , el mismo que establece que existen 2 asociaciones abiertas en total para el nodo 1, además de indicar la dirección IP del nodo del que se realiza el reporte, la dirección del siguiente nodo al que se vincula, la métrica existente entre ambos nodos, el valor generado por el nodo para identificar esa conexión, el valor generado por el nodo vecino para identificar esa conexión y el identificador numérico de la asociación. Estas asociaciones representan los caminos hacia el siguiente nodo y cada nodo tendrá vínculos hacia otros nodos de manera que permitan llegar el paquete del nodo fuente al nodo destino.

```

▼<PeerManagementProtocol>
  <Statistics linksTotal="2" linksOpened="2" linksClosed="0"/>
  ▶<PeerManagementProtocolMac address="00:00:00:00:00:01">...
    </PeerManagementProtocolMac>
    <PeerLink localAddress="00:00:00:00:00:01"
      peerInterfaceAddress="00:00:00:00:00:02"
      peerMeshPointAddress="00:00:00:00:00:02" metrica="151"
      lastBeacon="0.00491799" localLinkId="1" peerLinkId="2"
      assocId="0"/>
    <PeerLink localAddress="00:00:00:00:00:01"
      peerInterfaceAddress="00:00:00:00:00:03"
      peerMeshPointAddress="00:00:00:00:00:03" metrica="150"
      lastBeacon="0.0634243" localLinkId="2" peerLinkId="2"
      assocId="1"/>

```

Figura 111 Reporte del nodo 1

Fuente: Autor

Ya que cada uno de los nodos, va a asociarse con otros nodos que se encuentren dentro de la malla, la figura 112 muestra la asociación entre los nodos de la malla.

Para una malla de 2x2 con una distancia entre nodos de 100m, en total se tienen dos asociaciones desde el nodo 0; una hacia el nodo 1 y la otra hacia el nodo 2, lo cual comprueba los resultados obtenidos en el reporte mostrado en la figura 111.

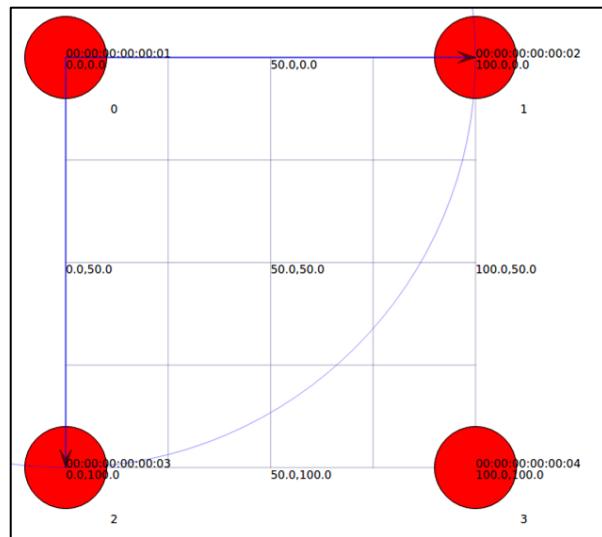


Figura 112 Asociación desde el nodo 1 - NetAnime

Fuente: Autor

En la figura 113 se muestra el reporte del nodo 3, en el mismo se informa que existen 3 enlaces abiertos, pero solo se lograron establecer 2 asociaciones una con el nodo 1 y otra con el nodo 4 ya que la distancia existente entre el nodo 3 y el nodo 2 supera al rango en el que se logra establecer una asociación.

```
▼ <PeerManagementProtocol>
  <Statistics linksTotal="2" linksOpened="3" linksClosed="1"/>
  ▶ <PeerManagementProtocolMac address="00:00:00:00:00:03">...
    </PeerManagementProtocolMac>
    <PeerLink localAddress="00:00:00:00:00:03"
      peerInterfaceAddress="00:00:00:00:00:04"
      peerMeshPointAddress="00:00:00:00:00:04" metrica="162"
      lastBeacon="4075.03" localLinkId="1" peerLinkId="2" assocId="0"/>
    <PeerLink localAddress="00:00:00:00:00:03"
      peerInterfaceAddress="00:00:00:00:00:01"
      peerMeshPointAddress="00:00:00:00:00:01" metrica="104"
      lastBeacon="4075.03" localLinkId="4" peerLinkId="4" assocId="3"/>
  ▷ ...
  ▷ ...
  ▷ ...
```

Figura 113 Reporte del nodo 2

Fuente: Autor

En la figura 114 se evidencia que se intentan establecer 3 asociaciones del nodo tres hacia los demás nodos.

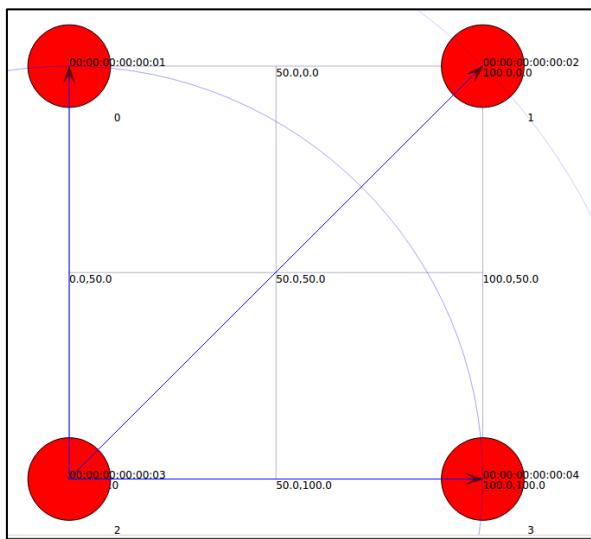


Figura 114 Asociaciones del nodo 3

Fuente: Autor

El cálculo de la métrica ALM existente entre estas asociaciones se realiza según se encuentra definido en 11B.10 de 802.11s Draft D3.0 como:

$$airtime = \frac{O}{1 - \text{frame error rate}} + \frac{Bt}{r} \quad Ec. 2$$

Donde

- ⊕ O es el acceso de canal dependiente de PHY que incluye las cabeceras de trama.
- ⊕ Bt es la longitud del paquete de prueba en bits (8192 por defecto).
- ⊕ r es el bitrate actual del paquete.

El resultado final se multiplica por 10.24 us (según lo requerido por el borrador 802.11s).

$O + \frac{Bt}{r}$ de la Ec. 2 representa $Overhead + payload + DIFS + SIFS + AckTxTime$

En el anexo 11 se muestra el código para calcular $m1 = Overhead + payload$ y $m2 = DIFS + SIFS + AckTxTime$, los mismos que al ser sumados deben ser equivalentes a la métrica obtenida en el registro.

La figura 101 se observa una hoja de cálculo en la cual se ha tabulado los datos obtenidos tanto para m1 como para m2, y se ha comparado el ALM obtenido con el ALM de los registros, en algunos casos existe la diferencia de una cifra y se debe al redondeo.

TIEMPO	m1=Overhead + payload	m2= DIFS + SIFS + AckTxTime	m1+m2 = ALM	ALM -Reporte
0,1	141	9	150	151
0,1	141	9	150	150
0,3	141	9	150	150
0,3	141	9	150	150
0,6	141	9	150	150
0,6	141	9	150	150
1	147	9	156	156
1	141	9	150	150
1,5	144	9	153	154
1,5	141	9	150	150
2,1	148	9	157	158
2,1	141	9	150	150
2,8	152	9	161	162
2,8	141	9	150	150
3,6	146	9	155	155
3,6	141	9	150	150
4,5	146	9	155	156
4,5	141	9	150	150
5,5	148	9	157	158
5,5	141	9	150	150

Figura 115 Cálculo de ALM

Fuente: Autor

Debido a que el número de reportes generados es muy elevado, se procedió a tabularlos mediante un archivo .excel. Las figuras 116 a 123 muestran las diferentes métricas de cada nodo hacia sus nodos vecinos en una malla de 2 x 2.



Figura 116 Métrica ALM del nodo 0 hacia el nodo 1

Fuente: Autor



Figura 117 Métrica ALM del nodo 0 hacia el nodo 2

Fuente: Autor



Figura 118 Métrica ALM del nodo 1 hacia el nodo 0

Fuente: Autor



Figura 119 Métrica ALM del nodo 1 hacia el nodo 3

Fuente: Autor



Figura 120 Métrica ALM del nodo 2 hacia el nodo 0

Fuente: Autor



Figura 121 Métrica ALM del nodo 2 hacia el nodo 3

Fuente: Autor



Figura 122 Métrica ALM del nodo 3 hacia el nodo 1

Figura: Autor



Figura 123 Métrica ALM del nodo 3 hacia el nodo 2

Figura: Autor

En la figura 124 se muestra que en una malla de 2x2, donde sus nodos se encuentran separados 100m entre sí. de 500 paquetes enviados solo se reciben 214 por lo tanto se abran perdido 286.

```
txBytes="526000" rxBytes="225128" txPackets="500" rxPackets="214" lostPackets="286"
```

Figura 124 Paquetes recibidos y perdidos en una malla de 2x2 con una separación de 100m entre nodos
Fuente: Autor

A medida que aumenta el número de nodos aumentará el número de asociaciones y mientras mayor sea la distancia entre los nodos menor será el número de asociaciones.

La figura 125 representa el reporte del nodo 37 de una malla 7 x 7, este nos muestra que se intentó establecer asociaciones con 8 nodos vecinos, pero solo se pudo asociar con 4 de ellos debido a la distancia en que se encontraban los otros 4 del nodo 37.

```
<PeerManagementProtocol>
  <Statistics linksTotal="4" linksOpened="8" linksClosed="4"/>
  > <PeerManagementProtocolMac address="00:00:00:00:00:26">...</PeerManagementProtocolMac>
    <PeerLink localAddress="00:00:00:00:00:26" peerInterfaceAddress="00:00:00:00:00:27"
      peerMeshPointAddress="00:00:00:00:00:27" metrica="150" lastBeacon="851.064" localLinkId="2"
      peerLinkId="1" assocId="1"/>
    <PeerLink localAddress="00:00:00:00:00:26" peerInterfaceAddress="00:00:00:00:00:2d"
      peerMeshPointAddress="00:00:00:00:00:2d" metrica="158" lastBeacon="851.097" localLinkId="4"
      peerLinkId="1" assocId="3"/>
    <PeerLink localAddress="00:00:00:00:00:26" peerInterfaceAddress="00:00:00:00:00:25"
      peerMeshPointAddress="00:00:00:00:00:25" metrica="150" lastBeacon="851.06" localLinkId="8"
      peerLinkId="10" assocId="7"/>
    <PeerLink localAddress="00:00:00:00:00:26" peerInterfaceAddress="00:00:00:00:00:1f"
      peerMeshPointAddress="00:00:00:00:00:1f" metrica="150" lastBeacon="851.059" localLinkId="9"
      peerLinkId="6" assocId="8"/>
```

Figura 125 Reporte del nodo 37

Fuente: Autor

Para una malla de 7x7 con una distancia entre nodos de 100m, en total se tienen cuatro asociaciones desde el nodo 37; una hacia el nodo 20, otra hacia el nodo 36, otra hacia el nodo 38 y la última hacia el nodo 44 (véase figura 126) lo cual comprueba los resultados obtenidos en el reporte mostrado en la figura 125.

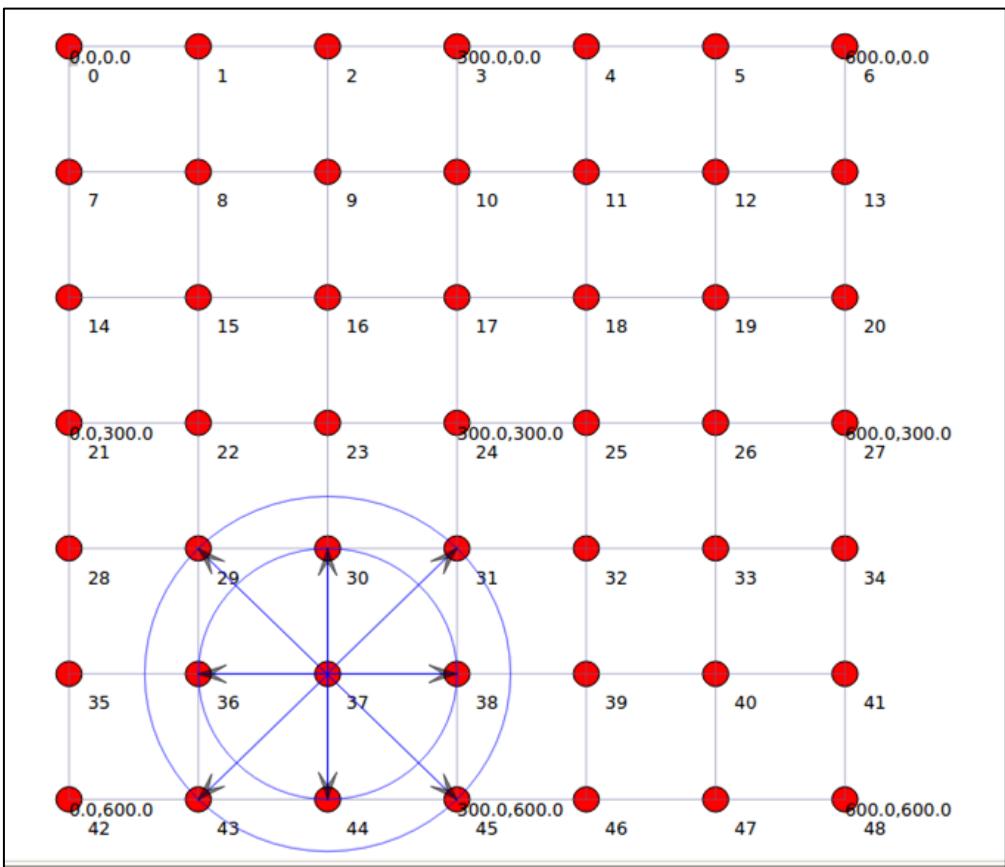


Figura 126 Asociaciones del nodo 3

Fuente: Autor

Después de haber llevado a cabo una serie de pruebas variando el número de nodos del que se encuentra compuesto la red, se obtuvieron los datos de la tabla 6.

Tabla 6
Resultados obtenidos al variar el número de nodos de la red

No Nodos	PAQUETES	PAQUETES	PAQUETES
	ENVIADOS	PERDIDOS	RECIBIDOS
4	500	286	214
10	500	8	492
15	500	8	492
30	500	83	417
36	500	91	409

49	500	21	479
64	500	21	479
72	500	30	470
81	500	105	395
100	500	74	426

Nota: Fuente Autor

La figura 127 muestra la comparación entre los paquetes recibidos y los paquetes perdidos, en función a los números de nodos estando estos separados 100m entre sí.

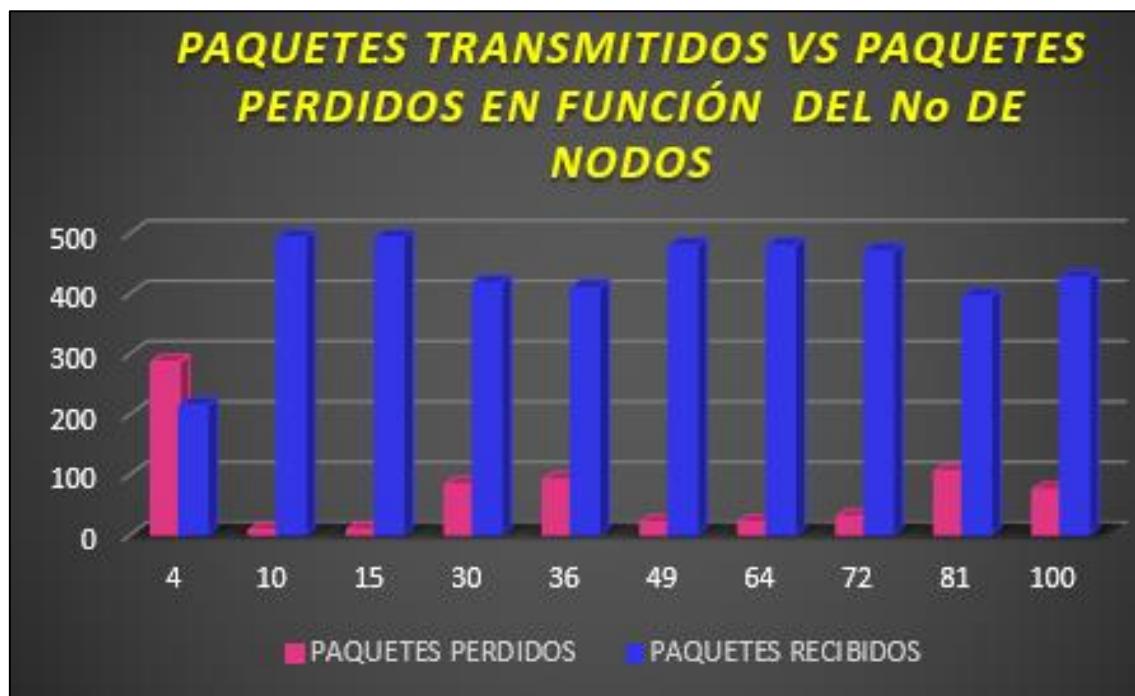


Figura 127 Paquetes recibidos vs paquetes perdidos.

Fuente: Autor

En base a los datos de la tabla 6 las figuras 128 y 129 muestran las figuras de mérito tanto para la tasa de pérdidas de paquetes, así como el Throughput (Kbps) obtenido.

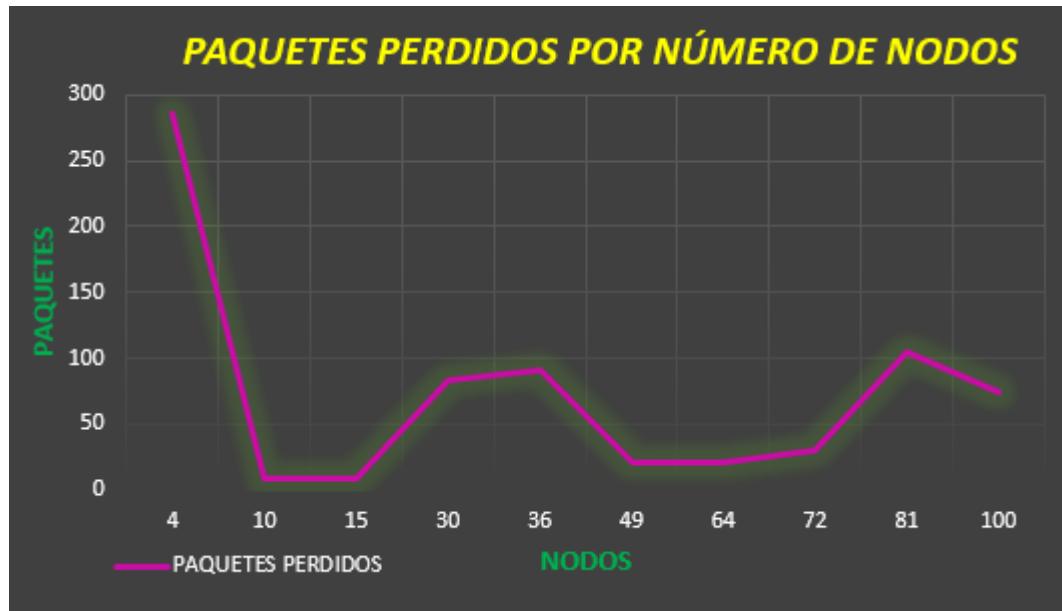


Figura 128 Tasa de paquetes perdidos – Nodos separados 100m

Fuente: Autor

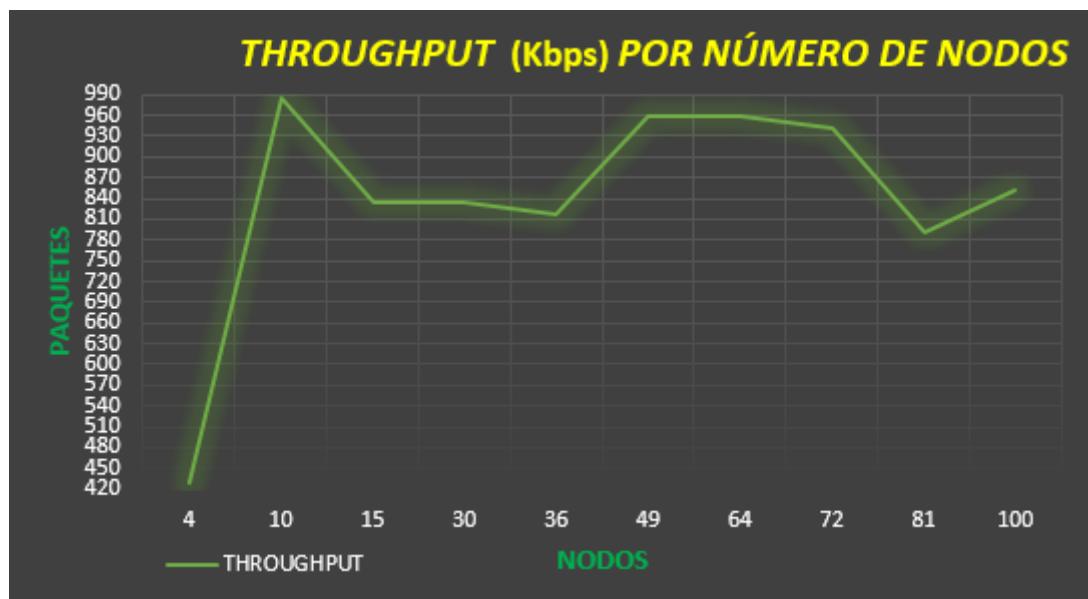


Figura 129 Throughput – nodos separados 100m

Fuente: Autor

6.4 Análisis de los resultados.

- El estándar IEEE 802.3 emplea CSMA / CD pero NS-3 solo modela la parte de CSMA. Como se puede observar en las figuras 57 y 65 no existe perdidas de paquetes, tanto para la topología tipo bus y estrellas, esto se debe a que NS-3 no existe colisiones en Ethernet, ya que los dispositivos NS-3-CSMA utiliza la naturaleza del canal para obtener información instantánea, y así evitar las colisiones basadas en prioridad.
- Como se puede evidenciar en la figura 74, que representa la configuración del estándar IEEE 802.11 b modo infraestructura, con una velocidad de transmisión de 11Mbps, técnica DSSS, modelo de perdida de propagación de larga distancia, con una pérdida de referencia de 46.67 dB, presenta una eficiencia de la red de 100% en una distancia de 60m en relación a los paquetes enviados frente a los paquetes recibidos, a partir de esta distancia se evidencia una eficiencia de 60% a una distancia de 68m, 40% a una distancia de 69m y 20% a una distancia de 70m, desde los 71m va existir una pérdida de datos totales.
- Como se puede observar en la figura 83, que representa la configuración del estándar IEEE 802.11 g modo infraestructura, con una velocidad de transmisión de 54 Mbps, técnica ERP-ODFM, modelo de perdida de propagación de larga distancia, con una pérdida de referencia de 46.67 dB, presenta una eficiencia de la red de 100% en una distancia de 26m en relación a los paquetes enviados frente a los paquetes recibidos, a partir de esta distancia se evidencia una eficiencia de 0% es decir va existir una pérdida de datos totales.
- Como se puede evidenciar en la figura 95, que representa la configuración del estándar IEEE 802.11 b modo Ad-Hoc, con una velocidad de transmisión de 11Mbps, técnica DSSS, modelo de perdida de propagación de larga distancia, con una pérdida de referencia de 46.67 dB, presenta una eficiencia de la red de 100% en una distancia de 57m en relación a los paquetes enviados frente a los paquetes recibidos, a partir de esta distancia se evidencia una eficiencia de 90% a una

distancia de 58m, 70% a una distancia de 60m, 20% a una distancia de 61m y 10% a una distancia de 62m, desde los 64m va existir una pérdida de datos totales.

- Como se puede observar en la figura 104, que representa la configuración del estándar IEEE 802.11 g modo Ad-Hoc, con una velocidad de transmisión de 54 Mbps, técnica ERP-ODFM, modelo de perdida de propagación de larga distancia, con una pérdida de referencia de 46.67 dB, presenta una eficiencia de la red de 100% en una distancia de 22m en relación a los paquetes enviados frente a los paquetes recibidos, a partir de esta distancia se evidencia una eficiencia de 90% a una distancia de 23m, 20% a una distancia de 24m, desde los 25m va existir una pérdida de datos totales.
- Se evidencia en las figuras 116, 117, 118, 119, 120, 121, 122 y 123 que las métricas de los diferentes nodos sufren una ligera oscilación al inicio de la transmisión, pero el mismo que no es de relevancia ya que los nodos van a escoger el camino para transmitir los datos que posea la mejor la mejor métrica. Además, se evidencia que la mayor pérdida de paquetes se da en la malla de 2x2 en donde los nodos se encuentran separados 100m entre sí.

7. DISCUSIÓN

El presente trabajo tiene como propósito general la implementación de un simulador de redes, específicamente NS-3, en el Laboratorio de Telecomunicaciones del AEIRNNR. Con los resultados obtenidos se confirmó la importancia de la implementación del simulador sobre todo para el estudio de redes inalámbricas ya que permite la modificación de diferentes parámetros y trabajar con un gran número de nodos, lo que representa una gran ventaja al momento de estudiar este tipo de redes, ya que se puede observar el comportamiento de estas redes a gran escala o a pequeña escala sin necesidad de tener que derrochar recursos y enormes cantidades de tiempo en su implementación.

El estándar IEEE 802.3, especifica las características de redes basadas en Ethernet, esta define como medio de transporte de los datos, el cable y como método de acceso al medio CSMA/CD, otra característica importante es su topología que puede ser tipo bus o estrella y que puede trabajar a velocidades de 10 o 100 Mbps, una de las desventajas de este tipo de redes son las colisiones lo que causará perdida de información y utilización de recursos en retransmisiones. Al realizar varias pruebas tanto en topología tipo bus como de estrella y en velocidades tanto de 10 como de 100 Mbps, no se pudo observar colisión alguna en ningún instante de tiempo, esto se debe a que internamente el simulador NS-3 no modela CSMA /CD, este modela solo la parte de CSMA, que permite el acceso múltiple a un medio compartido con función de detección de portadora, todo esto conlleva a tener un modelo ideal es decir este proporciona información de manera instantánea, por lo que jamás ocurrirá colisiones, ni abra retransmisiones. A pesar de que no se pueden observar colisiones ni retransmisiones, se puede trabajar con varios atributos en los dispositivos finales, se puede establecer parámetros como direcciones, tipo de encapsulación y el modelo de error. Además, que posee mecanismo de rastreo ASCII para el nivel superior y el rastreo PCAP para el nivel inferior.

El estándar 802.11 b/g que especifica las características para redes WI-FI, este estándar nos permite crear redes LAN con velocidades de transmisión altas, el estudio de este estándar es de relevancia ya que WI-FI se encuentra presente en todos nuestros espacios ya sean públicos como parques, o privados como oficinas. El problema con este tipo de

redes es el alcance de las mismas, ya que la señal no tiene la potencia para cubrir rangos muy extensos. El estándar 802.11b, ofrece una velocidad de transmisión hasta 11 Mbps y un rango de cobertura de 300 metros en espacios abiertos, en cambio el estándar 802.11g ofrece una velocidad de 54 Mbps y un rango de cobertura menor que el estándar 802.11b, es compatible con dispositivos 802.11b y ambos trabajan en la frecuencia de 2.4 Ghz. En el simulador se realizaron pruebas tanto en una red tipo infraestructura como en una red Ad-Hoc, con un modelo de propagación LogDistance, con una pérdida de referencia de 46.67 dB, con un control de tasa ConstantRate, el tamaño de los paquetes de 1024 bytes y cada uno de ellos es enviado en un intervalo de 1 segundo. Se pudo evidenciar que tiene mayor rango de cobertura el estándar 802.11b con 60m en modo infraestructura y 57m en modo Ad-Hoc, en comparación con el estándar 802.11g con 26 m en modo infraestructura y 22 m en modo Ad-Hoc, en distancias superiores a las mencionadas anteriormente se puede observar degradaciones en su rendimiento hasta llegar a un punto en el que ya no se transmitirá ningún paquete.

El estándar 802.11s emplea HWMP como esquema multi-salto y ALM como métrica predeterminada. Se realizó pruebas variando el número de nodos por el que está compuesta la red, siendo estos: 4, 10, 15, 30, 36, 49, 64, 72, 81 y 100 nodos. Durante estas pruebas se mantuvo constante ciertos parámetros como lo son: el modelo de retardo de propagación (ConstantSpeedPropagationDelayModel), el modelo de pérdida de propagación (LogDistancePropagationLossModel) en el mismo que se estableció una pérdida de referencia de 46.6777 dB, el protocolo de enlace peer-link. El modelo de pérdida de propagación escogido fue en función de la distancia, ya que las pruebas se realizaron tomando en cuenta una distancia en particular. En base a los resultados obtenidos el peor escenario es donde los nodos se encuentren separados 100m entre sí, es la malla de 2x2; debido a que solo tendrá dos asociaciones para llegar del nodo inicial al nodo final causando una gran pérdida de paquetes. En cambio, en las mallas conformadas por: 10, 15, 49 y 72 nodos son en las que se evidencia la tasa más baja de pérdidas de paquetes por lo tanto mayor throughput. Cabe enfatizar que el rendimiento de la red se debe a varios factores como: la métrica ALM, ya que en base a ella se escoge el camino para transmitir los datos; el TTL asignado, en este caso con el valor de 32, este factor influye sobre todo en las redes conformadas por un gran número de nodos ya que si el

camino elegido para transmitir los datos entre el nodo inicio y el nodo final supera al TTL asignado el paquete se descartará; el número de nodos y la distancia entre los mismos también son relevantes debido a que según la distancia entre los nodos, será el número de asociaciones posibles por nodo.

8. CONCLUSIONES

- ✚ La implementación de NS-3 es relevante ya que es empleado en numerosos proyectos para realizar una evaluación sobre el desempeño de una red determinada antes de la etapa de implementación, por otro lado, se escogió implementarlo en un sistema operativo con versión LTS por el soporte a largo plazo que ofrece.
- ✚ NS-3 no es el más adecuado para el estudio de las redes tipo Ethernet (estándar IEEE 802.3) ya que posee un modelo CSMA ideal en donde los datos son transmitidos a velocidades equivalentes a la de la luz, por lo que este modelo no permite que exista colisión alguna lo cual no refleja un ambiente real.
- ✚ Con NS-3 se puede reflejar un entorno de red muy semejante a uno real ya que permite simular cualquier tipo de topología, movilidad, tamaño, tráfico entre muchos otros parámetros que posee una red inalámbrica, el modelo de pérdida de propagación elegido es de gran relevancia ya que por medio de este se modela las irregularidades del medio real a representar, como paredes, personas, equipos eléctricos y otros sistemas inalámbricos los mismos que causan interferencias que disminuyen la intensidad de la señal; en el caso de las redes Wi-Fi configuradas en modo Infraestructura tanto para el estándar IEEE 802.11b como para el estándar IEEE 802.11g transmiten a una mayor distancia, debido a que poseen un AP y de este equipo va depender la distancia de transmisión de las misma.
- ✚ En las redes malladas, la pila de protocolos del estándar 802.11s (Protocol Peer Link Management y HWMP) son fundamentales para la implementación de los vínculos en la red y para el enrutamiento de la misma. Se evidencia que la mayor tasa de pérdidas de paquetes es en una malla de 2x2, ya que solo existirán dos posibles caminos para llegar del nodo inicio al nodo final y el mejor rendimiento de la red se evidencia en redes de conformadas por 10, 15, 49 y 72 nodos ya que estas tienen las tasas más altas de paquetes transmitidos.

9. RECOMENDACIONES

- Se recomienda que el simulador de redes 3 se encuentre instalado en un servidor, con el fin de que las computadoras que se conecten al servidor tengan acceso al mismo.
- Para obtener resultados reales en simulaciones de redes Ethernet (IEEE 802.3), se recomienda modificar el modelo de acceso al medio CSMA, en uno no ideal para que puedan existir colisiones, en otras palabras, implementar un modelo CSMA/CD.
- Durante la configuración de las redes Wi-Fi se recomienda configurar correctamente los modelos de pérdida de propagación para obtener resultados válidos por ejemplo si queremos evaluar el desempeño en función de rss se debe usar el método de Friss.
- Realizar un análisis comparativo de los paquetes transmitidos vs los paquetes perdidos y el throughput en las redes malladas, variando la distancia existente entre los nodos que conforman la misma.
- Se recomienda sacar un respaldo del simulador ns-3 instalado en las máquinas del laboratorio de telecomunicaciones si en un futuro es necesario formatearlas.

10. BIBLIOGRAFÍA

- Colegio de la UNLPAM. (2011). Redes de Computadoras. Recuperado de: <http://online2.exactas.unlpam.edu.ar/pe3/trabajos/previos/2011/sitiojuan/Teoricos/Redes%20-%20Cableadas%20e%20Inal%C3%A1mbricas.pdf>
- Castillo, G. G. (2005). Sistemas de Comunicaciones – Redes II. Recuperado de: <http://mixteco.utm.mx/~resdi/historial/materias/IPv4.pdf>
- Molero, L. G. (Sin Fecha). Ethernet e IEEE 802.3 y Arquitectura de TCP-IP. Recuperado de: <http://www.urbe.edu/info-consultas/web-profesor/12697883/archivos/Redes%20de%20Area%20Local%20y%20Metropolitana-cd2/Contenido/EtherneteIEEE802.3yArquitecturadeTCP-IP.pdf>
- Tanenbaum, A. S. (2003). Redes de computadoras. México: PEARSON EDUCACIÓN
- Universidad de Oviedo. (Sin Fecha). Tema 4: redes locales. Recuperado de: <http://www.isa.uniovi.es/docencia/redes/Apuntes/tema4.pdf>
- Universidad de Granada. (Sin Fecha). Redes de área local - Transmisión de datos y redes de ordenadores. Recuperado de: <http://elvex.ugr.es/decsai/internet/pdf/5%20LAN.pdf>
- Gralla, P. (2006), Como funcionan las redes inalámbricas. Madrid, España. Anaya multimedia.
- Vilcaguano, N. J., & Herrera, M. Y. (2007). Redes inalámbricas y la factibilidad de implementación en la Universidad Técnica de Cotopaxi. Latacunga. Ecuador. Recuperado de: <http://repositorio.utc.edu.ec/bitstream/27000/485/1/T-UTC-1041.pdf>

- ✚ Pellejero, I., & Andreu, F.; y Lesta, A. (2006). Fundamentos y aplicaciones de seguridad en redes WLAN: de la teoría a la práctica. Barcelona, España. MARCOMBO S.A.
- ✚ Roig, O., & Valenzuela, J.; Comes, R. (2003). Principios de comunicaciones móviles. Barcelona, España. UPC.
- ✚ Griera et al. (2008). Estructura de redes de computadores. Barcelona, España. UOC.
- ✚ Kruegle, H. (2007). CCTV Surveillance: Analog and Digital Video Practices and Technology. EE. UU. ELSEVIER
- ✚ Pellejero, I., Andreu, F., & Lesta, A. (2006). Fundamentos y aplicaciones de seguridad en redes WLAN. Barcelona, España. MARCOMBO S.A.
- ✚ Avila, C. M. (2015). Implementación de un Receptor OFDM IEEE 802.11 basado en SDR, Recuperado de <http://www.ptolomeo.unam.mx:8080/xmlui/bitstream/handle/132.248.52.100/8110/tesis.pdf?sequence=1>
- ✚ Conner, S., Krusy, J., Kim, K., Zuniga, J. (2006). IEEE 802.11s Tutorial: Overview of the Amendment for Wireless Local Area Mesh Networking. Recuperado de: http://www.ieee802.org/802_tutorials/06-November/802.11s_Tutorial_r5.pdf
- ✚ Espiga, A. (2012). Selección de portal en redes inalámbricas malladas utilizando aprendizaje estadístico. Recuperado de http://premat.fing.edu.uy/ingenieriamatematica/archivos/tesis_alejandro_espiga.PDF
- ✚ Municio, E., Quispe, M., Paucar, R., Chuchón, M., & Díaz, D. (Sin Fecha). Evaluación de protocolos de encaminamiento en una red inalámbrica mallada desplegada en una zona rural. Recuperado de http://oa.upm.es/19959/1/Articulo_Redes_Malladas_URSI2013_19_03_03.pdf

- ⊕ Espiga, A. (2012). Selección de portal en redes inalámbricas malladas utilizando aprendizaje estadístico. Recuperado de http://premat.fing.edu.uy/ingenieriamatematica/archivos/tesis_alejandro_espiga.PDF
- ⊕ Morales, M., Calle, M. A. Tovar, J. D. & Cuéllar, J. C. (2013). Simulando con OMNET: selección de la herramienta y su utilización. Santiago de Cali: Universidad Icesi.
- ⊕ ns-3 project. (2010-2017). Introduction-tutorial. Estados Unidos. Recuperado de <https://www.nsnam.org/docs/tutorial/html/introduction.html>
- ⊕ ns-3 project. (2010-2017). ns-3 Documentation. Estados Unidos. Recuperado de <https://www.nsnam.org/doxygen-release/#module-sec>
- ⊕ ns-3 project. (2010-2017). Conceptual Overview. Estados Unidos. Recuperado de <https://www.nsnam.org/docs/tutorial/html/conceptual-overview.html#a-first-ns-3-scriTpt>
- ⊕ ns-3 project. (2010-2017). Applications. Estados Unidos. Recuperado de <https://www.nsnam.org/docs/tutorial/html/conceptual-overview.html#a-first-ns-3-scriTpt>
- ⊕ ns-3 project. (2010-2017). FAQ. Estados Unidos. Recuperado de <https://www.nsnam.org/docs/tutorial/html/conceptual-overview.html#a-first-ns-3-scriTpt>
- ⊕ ns-3 project. (2010-2017).
- ⊕ Barbieri, S. (Sin Fecha). Ethernet / IEEE 802.3. Recuperado del sitio de internet de Universidad Nacional de Centro de la Prov. de Bs., Ingeniería en Sistemas – Facultad

<http://www.exa.unicen.edu.ar/catedras/comdat1/material/Ethernet2010.pdf>

■ Forouzan, B. A. (2001). *Transmisión de datos y redes de comunicación*. Madrid, España: McGraw-Hi

■ Mifsud, E.; Lerma-Blasco, R. (2013). *Servicios en red*. Madrid, España: McGraw-Hi

■ Caro, J. (Sin Fecha). *Redes*. SENA. Recuperado de:
<http://es.calameo.com/read/00446867775d62a38b2ca>

■ Novoa, A, & Reyes, R. (2007). *Análisis, estudio y site survey para investigar la factibilidad con respecto a la cobertura de señal wireless basada en el estándar 802.11 (wi-fi) en el campus sur de la Universidad Politécnica Salesiana*. (Título de grado). Universidad Politécnica Salesiana, Quito, Ecuador

■ Ruiz, F. (2007). *Redes de área metropolitana inalámbricas como una alternativa para enlaces de última milla según el estándar IEEE 8002.16*. Universidad de San Carlos de Guatemala, Guatemala.

■ Sosa, E. (2011). *Contribuciones al establecimiento de una red global de sensores inalambricos interconectados*. Universidad Nacional de La Plata. Buenos Aires, Argentina.

■ Escudero, A. (2007). *Estándares en Tecnologías Inalámbricas*. Recuperado de:
http://it46.se/courses/wireless/materials/es/02_Estandares/02_es_estandares-inalambricos_guia_v01.pdf.

■ Montes, E. (2008). *Estudio de la migración del estándar 802.11 al estándar 802.16 en zonas rurales*. (Título de grado). Pontificia Universidad Católica del Perú, Lima, Perú

- Andréu, J. (2011). *Redes locales*. Malaga, España. EDITEX
- Aguirre, T (2014). *Evaluación de VoIP en redes mesh parciales* (Título de maestría). Universidad Nacional Autónoma de México, México D.F.
- Blengio, R; Sosa, G, & Sosa, D (2008). *Monitoreo Redes Mesh*. Recuperado de: <https://iie.fing.edu.uy/publicaciones/2008/SSB08/SSB08.pdf>

11. ANEXOS

ANEXO 1: INSTALACIÓN DE NS-3

1. Abrir terminal

2. Ingresar modo root

↳ sudo -i

3. Instalar pre-requisitos:

- ↳ apt-get install gcc g++ Python
- ↳ apt-get install gcc g++ python python-dev
- ↳ apt-get install qt4-dev-tools libqt4-dev
- ↳ apt-get install mercurial
- ↳ apt-get install bzr
- ↳ apt-get install cmake libc6-dev libc6-dev-i386 g++-multilib
- ↳ apt-get install gdb valgrind
- ↳ apt-get install gsl-bin libgsl0-dev libgsl0ldbl
- ↳ apt-get install flex bison libfl-dev
- ↳ apt-get install tcpdump
- ↳ apt-get install sqlite sqlite3 libssqlite3-dev
- ↳ apt-get install libxml2 libxml2-dev
- ↳ apt-get install libgtk2.0-0 libgtk2.0-dev
- ↳ apt-get install vtun lxc
- ↳ apt-get install uncrustify
- ↳ apt-get install doxygen graphviz imagemagick
- ↳ apt-get install texlive texlive-extra-utils texlive-latex-extra texlive-font-utils texlive-lang-portuguese dvipng
- ↳ apt-get install python-sphinx dia
- ↳ apt-get install python-pygraphviz python-kiwi python-pygoocanvas libgoocanvas-dev ipython
- ↳ apt-get install libboost-signals-dev libboost-filesystem-dev
- ↳ apt-get install openmpi-bin openmpi-common openmpi-doc libopenmpi-dev

4. Crear carpeta “ns3”

✚ cd

✚ mkdir ns3

5. Abrir carpeta “ns3”

✚ cd ns3

6. Descargar el archivo

✚ wget http://www.nsnam.org/release/ns-allinone-3.22.tar.bz2

7. Instalar los paquetes

✚ tar xjf ns-allinone-3.22.tar.bz2

8. Abrir la carpeta “ns-allinone-3.22”

✚ cd ns-allinone-3.22/

9. Para construir su NS3, ejecute el comando siguiente:

✚ ./build.py --enable-examples --enable-tests

Si la construcción es exitosa, entonces se mostrará el siguiente mensaje:

“Build finished successfully”

10. Abrir la carpeta “ ns-3.22 ”

✚ cd ns-3.22/

11. Ejecute el siguiente comando para configurar la WAF (herramienta de construcción)

 ./waf -d debug --enable-examples --enable-tests configure

ANEXO 2: INSTALACIÓN DE OPENFLOW-SWITCH

1. Abrir terminal
 2. Ingresar en modo root
- ↳ sudo -i
3. Ingresar a la carpeta donde se encuentra instalado el simulador y a continuación copiar las siguientes líneas:

```
↳ cd /root/ns3/ns-allinone-3.22/ns-3.22
↳ export BOOSTDIR=/home/non-privileged-user/boost
↳ wget http://downloads.sourceforge.net/project/boost/boost/1.57.0/boost_1_57_0.tar.bz2
↳ tar jxf boost_1_57_0.tar.bz2
↳ cd boost_1_57_0
↳ ./bootstrap.sh
↳ ./b2 --prefix=$BOOSTDIR install
↳ cd /root/ns3/ns-allinone-3.22/ns-3.22
↳ hg clone http://code.nsnam.org/openflow
↳ cd openflow
↳ ./waf configure
↳ ./waf build
```

Si la construcción es exitosa, entonces se mostrará el siguiente mensaje:
"Build finished successfully"

```
↳ cd /root/ns3/ns-allinone-3.22/ns-3.22
↳ ./waf configure --enable-examples --enable-tests --with-
openflow=file:/root/ns3/ns-allinone-3.22/ns-3.22/openflow
```

Como se puede observar Openflow no se encuentra habilitado, a continuación se

procederá a habilitarlo, se debe abrir la carpeta: root/ns3/ns-allinone-3.22/ns-3.22/openflow/include y copiar la carpeta “openflow” que se encuentra dentro de la carpeta include abierta anteriormente.

Seguidamente abrir la carpeta: root/ns3/ns-allinone-3.22/ns-3.22/build y pegar la carpeta “openflow” dentro de la carpeta build abierta.

A continuación, copiar las siguientes líneas:

- + cd /root/ns3/ns-allinone-3.22/ns-3.22
- + ./waf configure --enable-examples --enable-tests --with-openflow=file:/root/ns3/ns-allinone-3.22/ns-3.22
- + cd openflow
- + ./waf configure
- + ./waf build
- + cd /root/ns3/ns-allinone-3.22/ns-3.22
- + ./waf configure --enable-examples --enable-tests --with-openflow=file:/root/ns3/ns-allinone-3.22/ns-3.22
- + ./waf build
- + ./waf configure --enable-examples --enable-tests --with-openflow=openflow

Una vez terminado, ya se debe encontrar habilitado el modulo y debe aparecer lo siguiente: Como se puede observar Openflow no se encuentra habilitado, a continuación se procederá a habilitarlo, se debe abrir la carpeta: root/ns3/ns-allinone-3.22/ns-3.22/openflow/include y copiar la carpeta “openflow” que se encuentra dentro de la carpeta include abierta anteriormente.

Seguidamente abrir la carpeta: root/ns3/ns-allinone-3.22/ns-3.22/build y pegar la carpeta “openflow” dentro de la carpeta build abierta.

A continuación, copiar las siguientes líneas:

```
↳ cd /root/ns3/ns-allinone-3.22/ns-3.22
↳ ./waf configure --enable-examples --enable-tests --with-
    openflow=file:/root/ns3/ns-allinone-3.22/ns-3.22
↳ cd openflow
↳ ./waf configure
↳ ./waf build
↳ cd /root/ns3/ns-allinone-3.22/ns-3.22
↳ ./waf configure --enable-examples --enable-tests --with-
    openflow=file:/root/ns3/ns-allinone-3.22/ns-3.22
↳ ./waf build
↳ ./waf configure --enable-examples --enable-tests --with-openflow=openflow
```

Una vez terminado, ya se debe encontrar habilitado el modulo.

ANEXO 3: MODO DE INGRESO

1. Abrir terminal

2. Ingresar en modo root

sudo -i

3. Ingresar a la carpeta donde se encuentra instalado el simulador:

cd /root/ns3/ns-allinone-3.22/ns-3.22

4. Hacer correr el script deseasdo con:

./waf -- run nombredelscript

**ANEXO 4: ESTÁNDAR IEEE 802.3 – ETHERNET - TOPOLOGÍA TIPO
BUS.**

```

#include "ns3/netanim-module.h"
#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/csma-module.h"
#include "ns3/internet-module.h"
#include "ns3/point-to-point-module.h"
#include "ns3/applications-module.h"
#include "ns3/ipv4-global-routing-helper.h"

using namespace ns3;

NS_LOG_COMPONENT_DEFINE ("TOPOLOGIA_BUS");

int
main (int argc, char *argv[])
{
{
    LogComponentEnable ("UdpEchoClientApplication", LOG_LEVEL_INFO);
    LogComponentEnable ("UdpEchoServerApplication", LOG_LEVEL_INFO);
}

uint32_t nodoCsma = 9;
CommandLine cmd;
cmd.AddValue ("nodoCsma", "Número de \"extra\" Nodos / dispositivos CSMA",
nodoCsma);
cmd.Parse (argc, argv);

NodeContainer Nodosp2p;
Nodosp2p.Create (2);

NodeContainer Nodoscsma;
Nodoscsma.Add (Nodosp2p.Get (1));

```

```

Nodoscsma.Create (nodoCsma);

PointToPointHelper pointToPoint;
pointToPoint.SetDeviceAttribute ("DataRate", StringValue ("10Mbps"));
pointToPoint.SetChannelAttribute ("Delay", StringValue ("2ms"));

NetDeviceContainer Dispositivosp2p;
Dispositivosp2p = pointToPoint.Install (Nodosp2p);

CsmaHelper csma;

NetDeviceContainer Dispositivoscsma;
Dispositivoscsma = csma.Install (Nodoscsma);

InternetStackHelper stack;
stack.Install (Nodosp2p.Get (0));
stack.Install (Nodoscsma);

Ipv4AddressHelper ipv4;
ipv4.SetBase ("192.168.1.0", "255.255.255.0");
Ipv4InterfaceContainer Interfacesp2p;
Interfacesp2p = ipv4.Assign (Dispositivosp2p);

ipv4.SetBase ("192.168.10.0", "255.255.255.0");
Ipv4InterfaceContainer Interfacescsma;
Interfacescsma = ipv4.Assign (Dispositivoscsma);

UdpEchoServerHelper echoServer (9);

ApplicationContainer serverApps = echoServer.Install (Nodoscsma.Get (nodoCsma));
serverApps.Start (Seconds (1.0));
serverApps.Stop (Seconds (10.0));

```

```
    UdpEchoClientHelper echoClient (Interfacescsma.GetAddress (nodoCsma), 9);
    echoClient.SetAttribute ("MaxPackets", UintegerValue (10));
    echoClient.SetAttribute ("Interval", StringValue ("2ms"));
    echoClient.SetAttribute ("PacketSize", UintegerValue (1024));
```

```
ApplicationContainer clientApps = echoClient.Install (Nodosp2p.Get (0));
clientApps.Start (Seconds (2.0));
clientApps.Stop (Seconds (10.0));
```

```
Ipv4GlobalRoutingHelper::PopulateRoutingTables ();
```

```
AnimationInterface anim ("/root/ns3/ns-allinone-3.22/ns-
3.22/src/resultados/topobus/topobus.xml");
anim.SetConstantPosition (Nodosp2p.Get(0), 0.0, 0.0);
anim.SetConstantPosition (Nodosp2p.Get(1), 20.0, 0.0);
anim.SetConstantPosition (Nodoscsma.Get(1), 30.0, 0.0);
anim.SetConstantPosition (Nodoscsma.Get(2), 40.0, 0.0);
anim.SetConstantPosition (Nodoscsma.Get(3), 50.0, 0.0);
anim.SetConstantPosition (Nodoscsma.Get(4), 60.0, 0.0);
anim.SetConstantPosition (Nodoscsma.Get(5), 70.0, 0.0);
anim.SetConstantPosition (Nodoscsma.Get(6), 80.0, 0.0);
anim.SetConstantPosition (Nodoscsma.Get(7), 90.0, 0.0);
anim.SetConstantPosition (Nodoscsma.Get(8), 100.0, 0.0);
anim.SetConstantPosition (Nodoscsma.Get(9), 110.0, 0.0);
```

```
AsciiTraceHelper ascii;
csma.EnableAsciiAll (ascii.CreateFileStream ("/root/ns3/ns-allinone-3.22/ns-
3.22/src/resultados/topobus/topobus.tr"));
csma.EnablePcapAll ("/root/ns3/ns-allinone-3.22/ns-
3.22/src/resultados/topobus/topobus", true);
pointToPoint.EnablePcapAll ("topobus",true);
```

```
    Simulator::Run();  
    Simulator::Destroy();  
    return 0;  
}
```

**ANEXO 5: ESTÁNDAR IEEE 802.3 – ETHERNET - TOPOLOGÍA TIPO
ESTRELLA.**

```

#include <iostream>
#include <fstream>

#include "ns3/netanim-module.h"
#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/csma-module.h"
#include "ns3/internet-module.h"
#include "ns3/applications-module.h"
#include "ns3/openflow-module.h"
#include "ns3/log.h"

using namespace ns3;

NS_LOG_COMPONENT_DEFINE ("OpenFlowCsmaSwitchExample");
bool use_drop = true;
ns3::Time timeout = ns3::Seconds (0);

bool
SetDrop (std::string value)
{
    use_drop = true;
    return true;
}

int
main (int argc, char *argv[])
{
    CommandLine cmd;
    cmd.AddValue ("d", "Use Drop Crontoler ", MakeCallback (&SetDrop));
    cmd.AddValue ("drop", " Use Drop Crontoler ", MakeCallback (&SetDrop));
}

```

```

cmd.Parse (argc, argv);

LogComponentEnable ("OpenFlowCsmaSwitchExample", LOG_LEVEL_INFO);
LogComponentEnable ("OpenFlowInterface", LOG_LEVEL_INFO);
LogComponentEnable ("OpenFlowSwitchNetDevice", LOG_LEVEL_INFO);

NS_LOG_INFO ("Creación de nodos.");
NodeContainer terminales;
terminales.Create (6);

NodeContainer Switchesma;
Switchesma.Create (1);

NS_LOG_INFO ("Construcción de la topología");
CsmaHelper csma;
csma.SetChannelAttribute ("DataRate", DataRateValue (10000000));
csma.SetChannelAttribute ("Delay", TimeValue (MilliSeconds (2)));

NetDeviceContainer terminalDevices;
NetDeviceContainer switchDevices;
for (int i = 0; i < 6; i++)

{
    NetDeviceContainer link = csma.Install (NodeContainer (terminales.Get (i),
Switchesma));
    terminalDevices.Add (link.Get (0));
    switchDevices.Add (link.Get (1));
}

// Se crea el netdevice del switch para poder enviar paquetes a cada nodo
Ptr<Node> switchNode = Switchesma.Get (0);
OpenFlowSwitchHelper swtch;

```

```

Ptr<ns3::ofi::LearningController> controller =
CreateObject<ns3::ofi::LearningController> ();
if (!timeout.IsZero ()) controller->SetAttribute ("ExpirationTime", TimeValue
(timeout));
swtch.Install (switchNode, switchDevices, controller);

InternetStackHelper internet;
internet.Install (terminales);

// Asignar direcciones ip
NS_LOG_INFO ("Asignación de direccioens IPs");
Ipv4AddressHelper ipv4;
ipv4.SetBase ("192.168.1.0", "255.255.255.0");
ipv4.Assign (terminalDevices); //se asigna una ip a cada nodo

// Enviar paquetes desde n0 a n1 en el segundo 1
NS_LOG_INFO ("Creación de nivel de aplicación.");
uint16_t port = 9; // Discard port (RFC 863)

OnOffHelper onoff ("ns3::UdpSocketFactory",
Address (InetSocketAddress (Ipv4Address ("192.168.1.2"), port)));
onoff.SetConstantRate (DataRate ("10Mb/s"));

ApplicationContainer app = onoff.Install (terminales.Get (0));

app.Start (Seconds (1.0));
app.Stop (Seconds (10.0));

// Enviar paquetes desde n0 a n2 en el segundo 1.1
onoff.SetAttribute ("Remote",

```

```

        AddressValue (InetSocketAddress (Ipv4Address ("192.168.1.3"), port)));
app = onoff.Install (terminales.Get (0));
app.Start (Seconds (1.1));
app.Stop (Seconds (10.0));

// Enviar paquetes desde n0 a n3 en el segundo 1.2
onoff.SetAttribute ("Remote",
        AddressValue (InetSocketAddress (Ipv4Address ("192.168.1.4"), port)));
app = onoff.Install (terminales.Get (0));
app.Start (Seconds (1.2));
app.Stop (Seconds (10.0));

// Enviar paquetes desde n0 a n4 en el segundo 1.3
onoff.SetAttribute ("Remote",
        AddressValue (InetSocketAddress (Ipv4Address ("192.168.1.5"), port)));
app = onoff.Install (terminales.Get (0));
app.Start (Seconds (1.3));
app.Stop (Seconds (10.0));

// Enviar paquetes desde n0 a n5 en el segundo 1.4
onoff.SetAttribute ("Remote",
        AddressValue (InetSocketAddress (Ipv4Address ("192.168.1.6"), port)));
app = onoff.Install (terminales.Get (0));
app.Start (Seconds (1.4));
app.Stop (Seconds (10.0));

// Enviar paquetes desde n1 a n0 en el segundo 1.5
onoff.SetAttribute ("Remote",
        AddressValue (InetSocketAddress (Ipv4Address ("192.168.1.1"), port)));
app = onoff.Install (terminales.Get (1));
app.Start (Seconds (1.5));

```

```

app.Stop (Seconds (10.0));

// Enviar paquetes desde n1 a n2 en el segundo 1.6
onoff.SetAttribute ("Remote",
    AddressValue (InetSocketAddress (Ipv4Address ("192.168.1.3"), port)));
app = onoff.Install (terminales.Get (1));
app.Start (Seconds (1.6));
app.Stop (Seconds (10.0));

// Enviar paquetes desde n1 a n3 en el segundo 1.7
onoff.SetAttribute ("Remote",
    AddressValue (InetSocketAddress (Ipv4Address ("192.168.1.4"), port)));
app = onoff.Install (terminales.Get (1));
app.Start (Seconds (1.7));
app.Stop (Seconds (10.0));

// Enviar paquetes desde n1 a n4 en el segundo 1.8
onoff.SetAttribute ("Remote",
    AddressValue (InetSocketAddress (Ipv4Address ("192.168.1.5"), port)));
app = onoff.Install (terminales.Get (1));
app.Start (Seconds (1.8));
app.Stop (Seconds (10.0));

// Enviar paquetes desde n1 a n5 en el segundo 1.9
onoff.SetAttribute ("Remote",
    AddressValue (InetSocketAddress (Ipv4Address ("192.168.1.6"), port)));
app = onoff.Install (terminales.Get (1));
app.Start (Seconds (1.9));
app.Stop (Seconds (10.0));

```

```

// Enviar paquetes desde n2 a n0 en el segundo 2.0
onoff.SetAttribute ("Remote",
    AddressValue (InetSocketAddress (Ipv4Address ("192.168.1.1"), port)));
app = onoff.Install (terminales.Get (2));
app.Start (Seconds (2.0));
app.Stop (Seconds (10.0));

// Enviar paquetes desde n2 a n1 en el segundo 2.1
onoff.SetAttribute ("Remote",
    AddressValue (InetSocketAddress (Ipv4Address ("192.168.1.2"), port)));
app = onoff.Install (terminales.Get (2));
app.Start (Seconds (2.1));
app.Stop (Seconds (10.0));

// Enviar paquetes desde n2 a n3 en el segundo 2.2
onoff.SetAttribute ("Remote",
    AddressValue (InetSocketAddress (Ipv4Address ("192.168.1.4"), port)));
app = onoff.Install (terminales.Get (2));
app.Start (Seconds (2.2));
app.Stop (Seconds (10.0));

// Enviar paquetes desde n2 a n4 en el segundo 2.3
onoff.SetAttribute ("Remote",
    AddressValue (InetSocketAddress (Ipv4Address ("192.168.1.5"), port)));
app = onoff.Install (terminales.Get (2));
app.Start (Seconds (2.3));
app.Stop (Seconds (10.0));

// Enviar paquetes desde n2 a n5 en el segundo 2.4
onoff.SetAttribute ("Remote",

```

```

        AddressValue (InetSocketAddress (Ipv4Address ("192.168.1.6"), port)));
app = onoff.Install (terminales.Get (2));
app.Start (Seconds (2.4));
app.Stop (Seconds (10.0));

// Enviar paquetes desde n2 a n5 en el segundo 2.4
onoff.SetAttribute ("Remote",
        AddressValue (InetSocketAddress (Ipv4Address ("192.168.1.6"), port)));
app = onoff.Install (terminales.Get (2));
app.Start (Seconds (2.4));
app.Stop (Seconds (10.0));

// Enviar paquetes desde n3 a n0 en el segundo 2.5
onoff.SetAttribute ("Remote",
        AddressValue (InetSocketAddress (Ipv4Address ("192.168.1.1"), port)));
app = onoff.Install (terminales.Get (3));
app.Start (Seconds (2.5));
app.Stop (Seconds (10.0));

// Enviar paquetes desde n3 a n1 en el segundo 2.6
onoff.SetAttribute ("Remote",
        AddressValue (InetSocketAddress (Ipv4Address ("192.168.1.2"), port)));
app = onoff.Install (terminales.Get (3));
app.Start (Seconds (2.6));
app.Stop (Seconds (10.0));

// Enviar paquetes desde n3 a n2 en el segundo 2.7
onoff.SetAttribute ("Remote",
        AddressValue (InetSocketAddress (Ipv4Address ("192.168.1.3"), port)));
app = onoff.Install (terminales.Get (3));
app.Start (Seconds (2.7));
app.Stop (Seconds (10.0));

```

```

// Enviar paquetes desde n3 a n4 en el segundo 2.8
onoff.SetAttribute ("Remote",
    AddressValue (InetSocketAddress (Ipv4Address ("192.168.1.5"), port)));
app = onoff.Install (terminales.Get (3));
app.Start (Seconds (2.8));
app.Stop (Seconds (10.0));

// Enviar paquetes desde n3 a n5 en el segundo 2.9
onoff.SetAttribute ("Remote",
    AddressValue (InetSocketAddress (Ipv4Address ("192.168.1.6"), port)));
app = onoff.Install (terminales.Get (3));
app.Start (Seconds (2.9));
app.Stop (Seconds (10.0));

// Enviar paquetes desde n4 a n0 en el segundo 3.0
onoff.SetAttribute ("Remote",
    AddressValue (InetSocketAddress (Ipv4Address ("192.168.1.1"), port)));
app = onoff.Install (terminales.Get (4));
app.Start (Seconds (3.0));
app.Stop (Seconds (10.0));

// Enviar paquetes desde n4 a n1 en el segundo 3.1
onoff.SetAttribute ("Remote",
    AddressValue (InetSocketAddress (Ipv4Address ("192.168.1.2"), port)));
app = onoff.Install (terminales.Get (4));
app.Start (Seconds (3.1));
app.Stop (Seconds (10.0));

// Enviar paquetes desde n4 a n2 en el segundo 3.2

```

```

onoff.SetAttribute ("Remote",
    AddressValue (InetSocketAddress (Ipv4Address ("192.168.1.3"), port)));
app = onoff.Install (terminales.Get (4));
app.Start (Seconds (3.2));
app.Stop (Seconds (10.0));

// Enviar paquetes desde n4 a n3 en el segundo 3.3
onoff.SetAttribute ("Remote",
    AddressValue (InetSocketAddress (Ipv4Address ("192.168.1.4"), port)));
app = onoff.Install (terminales.Get (4));
app.Start (Seconds (3.3));
app.Stop (Seconds (10.0));

// Enviar paquetes desde n4 a n5 en el segundo 3.4
onoff.SetAttribute ("Remote",
    AddressValue (InetSocketAddress (Ipv4Address ("192.168.1.6"), port)));
app = onoff.Install (terminales.Get (4));
app.Start (Seconds (3.4));
app.Stop (Seconds (10.0));

// Enviar paquetes desde n5 a n0 en el segundo 3.5
onoff.SetAttribute ("Remote",
    AddressValue (InetSocketAddress (Ipv4Address ("192.168.1.1"), port)));
app = onoff.Install (terminales.Get (5));
app.Start (Seconds (3.5));
app.Stop (Seconds (10.0));

// Enviar paquetes desde n5 a n1 en el segundo 3.6
onoff.SetAttribute ("Remote",
    AddressValue (InetSocketAddress (Ipv4Address ("10.1.1.2"), port)));
app = onoff.Install (terminales.Get (5));
app.Start (Seconds (3.6));

```

```

app.Stop (Seconds (10.0));

// Enviar paquetes desde n5 a n2 en el segundo 3.7
onoff.SetAttribute ("Remote",
    AddressValue (InetSocketAddress (Ipv4Address ("192.168.1.3"), port)));
app = onoff.Install (terminales.Get (5));
app.Start (Seconds (3.7));
app.Stop (Seconds (10.0));

// Enviar paquetes desde n5 a n3 en el segundo 3.8
onoff.SetAttribute ("Remote",
    AddressValue (InetSocketAddress (Ipv4Address ("192.168.1.4"), port)));
app = onoff.Install (terminales.Get (5));
app.Start (Seconds (3.8));
app.Stop (Seconds (10.0));

// Enviar paquetes desde n5 a n4 en el segundo 3.9
onoff.SetAttribute ("Remote",
    AddressValue (InetSocketAddress (Ipv4Address ("192.168.1.5"), port)));
app = onoff.Install (terminales.Get (5));
apupp.Stop (Seconds (10.0));

// Configuración de rastreo
AnimationInterface anim ("/root/ns3/ns-allinone-3.22/ns-
3.22/src/resultados/topoestrella/topoestrella.xml");
anim.SetConstantPosition (terminales.Get(0), 0.0, 0.0);
anim.SetConstantPosition (terminales.Get(1), 0.0, 5.0);
anim.SetConstantPosition (terminales.Get(2), 0.0, 10.0);
anim.SetConstantPosition (terminales.Get(3), 25.0, 0.0);
anim.SetConstantPosition (terminales.Get(4), 25.0, 5.0);
anim.SetConstantPosition (terminales.Get(5), 25.0, 10.0);

```

```
NS_LOG_INFO ("Configuración de rastreo.");

AsciiTraceHelper ascii;
csma.EnableAsciiAll (ascii.CreateFileStream ("/root/ns3/ns-allinone-3.22/ns-
3.22/src/resultados/topoestrella/topoestrella.tr"));
csma.EnablePcapAll ("/root/ns3/ns-allinone-3.22/ns-
3.22/src/resultados/topoestrella/topoestrella", true);

//Ejecución
NS_LOG_INFO ("Ejecución de la simulación.");
Simulator::Run ();
Simulator::Destroy ();

}
```

**ANEXO 6: ESTÁNDAR IEEE 802.11b – WIFI - MODO
INFRAESTRUCTURA**

```

#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/mobility-module.h"
#include "ns3/config-store-module.h"
#include "ns3/wifi-module.h"
#include "ns3/internet-module.h"
#include "ns3/yans-wifi-helper.h"
#include "ns3/nqos-wifi-mac-helper.h"
#include "ns3/ssid.h"
#include "ns3/internet-stack-helper.h"
#include "ns3/ipv4-address-helper.h"
#include "ns3/udp-echo-helper.h"
#include "ns3/ipv4-global-routing-helper.h"
#include "ns3/netanim-module.h"

#include <iostream>
#include <fstream>
#include <vector>
#include <string>

using namespace ns3;

NS_LOG_COMPONENT_DEFINE ("Infra");

void ReceivePacket (Ptr<Socket> socket)
{
    while (socket->Recv ())
    {
        NS_LOG_UNCOND ("PAQUETE RECIBIDO");
    }
}

```

```

static void GenerateTraffic (Ptr<Socket> socket, uint32_t pktSize,
                           uint32_t pktCount, Time pktInterval )
{
    if (pktCount > 0)
    {
        socket->Send (Create<Packet> (pktSize));
        Simulator::Schedule (pktInterval, &GenerateTraffic,
                             socket, pktSize,pktCount-1, pktInterval);
    }
    else
    {
        socket->Close ();
    }
}

```

```

int main (int argc, char *argv[])
{
    std::string phyMode ("DsssRate11Mbps");
    uint32_t distancia = 50; // metros
    uint32_t tamapaquete = 1024 ;// bytes
    uint32_t numpaquetes = 10;
    double intervalo = 1.0; // segundos
    bool verbose = false;

    CommandLine cmd;

    cmd.AddValue ("phyMode", "Wifi Phy mode", phyMode);
    cmd.AddValue ("tamapaquete", "tamaño del paquete enviado", tamapaquete);
    cmd.AddValue ("numpaquetes", "número de paquetes generado", numpaquetes);

```

```

cmd.AddValue ("intervalo", "intervalo entre el envio de paquetes", intervalo);
cmd.AddValue ("verbose", "turn on all WifiNetDevice log components", verbose);
cmd.AddValue ("distancia", "distancia entre los nodos", distancia);
cmd.Parse (argc, argv);
// Convertir en objeto de tiempo
Time interPacketInterval = Seconds (intervalo);

// Deshabilitar la fragmentación de los tramas inferiores a 2200 bytes
Config::SetDefault ("ns3::WifiRemoteStationManager::FragmentationThreshold",
StringValue ("2200"));

// Desactive RTS / CTS para las tramas inferiores a 2200 bytes
Config::SetDefault ("ns3::WifiRemoteStationManager::RtsCtsThreshold", StringValue
("2200"));

// Fijar la velocidad de transmisión de datos
Config::SetDefault ("ns3::WifiRemoteStationManager::NonUnicastMode",
StringValue (phyMode));

NodeContainer ap;
ap.Create (1);

NodeContainer nodo;
nodo.Create (2);

WifiHelper wifi;
if (verbose)
{
    wifi.EnableLogComponents ();
}

//Establece el estandár con el que va a trabajar
wifi.SetStandard (WIFI_PHY_STANDARD_80211b);

```

```

YansWifiPhyHelper wifiPhy = YansWifiPhyHelper::Default ();

wifiPhy.Set ("RxGain", DoubleValue (0));
wifiPhy.SetPcapDataLinkType (YansWifiPhyHelper::DLT_IEEE802_11_RADIO);

YansWifiChannelHelper wifiChannel;
wifiChannel.SetPropagationDelay ("ns3::ConstantSpeedPropagationDelayModel");
wifiChannel.AddPropagationLoss ("ns3::LogDistancePropagationLossModel",
                               "Exponent", DoubleValue (3.0),
                               "ReferenceLoss", DoubleValue (46.6777));
wifiPhy.SetChannel (wifiChannel.Create ());

NqosWifiMacHelper wifiMac = NqosWifiMacHelper::Default ();
wifi.SetRemoteStationManager ("ns3::ConstantRateWifiManager",
                             "DataMode", StringValue (phyMode),
                             "ControlMode", StringValue (phyMode));

// Configuración de la estación
Ssid ssid = Ssid ("wifi-default");

wifiMac.SetType ("ns3::StaWifiMac",
                 "Ssid", SsidValue (ssid),
                 "ActiveProbing", BooleanValue (false));
NetDeviceContainer staDevice = wifi.Install (wifiPhy, wifiMac, nodo);
NetDeviceContainer devices = staDevice;
// Configuración del AP
wifiMac.SetType ("ns3::ApWifiMac",
                 "Ssid", SsidValue (ssid));
NetDeviceContainer apDevice = wifi.Install (wifiPhy, wifiMac, ap);
devices.Add (apDevice);

//Configurar movilidad

```

```

MobilityHelper mobility;
mobility.SetPositionAllocator ("ns3::GridPositionAllocator",
                            "MinX", DoubleValue (0.0),
                            "MinY", DoubleValue (0.0),
                            "DeltaX", DoubleValue (distancia),
                            "DeltaY", DoubleValue (distancia),
                            "GridWidth", UIntegerValue (2),
                            "LayoutType", StringValue ("RowFirst"));

mobility.SetMobilityModel("ns3::ConstantPositionMobilityModel");
mobility.Install (ap);
mobility.Install (nodo);

```

```

InternetStackHelper internet;
internet.Install (ap);
internet.Install (nodo);

```

```

Ipv4AddressHelper ipv4;
NS_LOG_INFO ("Assign IP Addresses.");
ipv4.SetBase ("192.168.1.0", "255.255.255.0");
Ipv4InterfaceContainer i = ipv4.Assign (devices);

```

```

TypeId tid = TypeId::LookupByName ("ns3::UdpSocketFactory");
Ptr<Socket> recvSink = Socket::CreateSocket (ap.Get (0), tid);
InetSocketAddress local = InetSocketAddress (Ipv4Address::GetAny (), 80);
recvSink->Bind (local);
recvSink->SetRecvCallback (MakeCallback (&ReceivePacket));

```

```

Ptr<Socket> source = Socket::CreateSocket (nodo.Get (0), tid);
InetSocketAddress remote = InetSocketAddress (Ipv4Address ("255.255.255.255"),
80);
source->SetAllowBroadcast (true);
source->Connect (remote);

```

```

wifiPhy.EnablePcap ("/root/ns3/ns-allinone-3.22/ns-
3.22/src/resultados/infra_b/infra_ap", apDevice);
wifiPhy.EnablePcap ("/root/ns3/ns-allinone-3.22/ns-
3.22/src/resultados/infra_b/infra_estaciones", staDevice);
std::string animFile = "/root/ns3/ns-allinone-3.22/ns-
3.22/src/resultados/infra_b/anim1.xml" ;

cmd.AddValue ("animFile", "File Name for Animation Output", animFile);
AnimationInterface anim (animFile);

Simulator::ScheduleWithContext (source->GetNode ()->GetId (),
Seconds (1.0), &GenerateTraffic,
source, tamapaquete, numpaquetes, interPacketInterval);

NS_LOG_UNCOND (" Prueba, envia " << numpaquetes << " paquetes, distancia de "
<< distancia << "m" );

ns3::PacketMetadata::Enable ();
Simulator::Stop (Seconds (30.0));
Simulator::Run ();
Simulator::Destroy ();

return 0;
}

```

**ANEXO 7: ESTÁNDAR IEEE 802.11g – WIFI - MODO
INFRAESTRUCTURA**

```

#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/mobility-module.h"
#include "ns3/config-store-module.h"
#include "ns3/wifi-module.h"
#include "ns3/internet-module.h"
#include "ns3/yans-wifi-helper.h"
#include "ns3/nqos-wifi-mac-helper.h"
#include "ns3/ssid.h"
#include "ns3/internet-stack-helper.h"
#include "ns3/ipv4-address-helper.h"
#include "ns3/udp-echo-helper.h"
#include "ns3/ipv4-global-routing-helper.h"
#include "ns3/netanim-module.h"

#include <iostream>
#include <fstream>
#include <vector>
#include <string>

using namespace ns3;

NS_LOG_COMPONENT_DEFINE ("Infra");

void ReceivePacket (Ptr<Socket> socket)
{
    while (socket->Recv ())
    {
        NS_LOG_UNCOND ("PAQUETE RECIBIDO");
    }
}

```

```

}

static void GenerateTraffic (Ptr<Socket> socket, uint32_t pktSize,
                           uint32_t pktCount, Time pktInterval )
{
    if (pktCount > 0)
    {
        socket->Send (Create<Packet> (pktSize));
        Simulator::Schedule (pktInterval, &GenerateTraffic,
                             socket, pktSize,pktCount-1, pktInterval);
    }
    else
    {
        socket->Close ();
    }
}

```

```

int main (int argc, char *argv[])
{
    std::string phyMode ("ErpOfdmRate54Mbps");
    uint32_t distancia = 20; // metros
    uint32_t tamapaquete = 1024 ;// bytes
    uint32_t numpaquetes = 10;
    double intervalo = 1.0; // segundos
    bool verbose = false;

```

CommandLine cmd;

```

cmd.AddValue ("phyMode", "Wifi Phy mode", phyMode);
cmd.AddValue ("tamapaquete", "tamaño del paquete enviado", tamapaquete);

```

```

cmd.AddValue ("numpaquetes", "número de paquetes generado", numpaquetes);
cmd.AddValue ("intervalo", "intervalo entre el envio de paquetes", intervalo);
cmd.AddValue ("verbose", "turn on all WifiNetDevice log components", verbose);
cmd.AddValue ("distancia", "distancia entre los nodos", distancia);
cmd.Parse (argc, argv);

// Convertir en objeto de tiempo
Time interPacketInterval = Seconds (intervalo);

// Deshabilitar la fragmentación de los tramas inferiores a 2200 bytes
Config::SetDefault ("ns3::WifiRemoteStationManager::FragmentationThreshold",
StringValue ("2200"));

// Desactive RTS / CTS para las tramas inferiores a 2200 bytes
Config::SetDefault ("ns3::WifiRemoteStationManager::RtsCtsThreshold", StringValue
("2200"));

// Fijar la velocidad de transmisión de datos
Config::SetDefault ("ns3::WifiRemoteStationManager::NonUnicastMode",
StringValue (phyMode));

NodeContainer ap;
ap.Create (1);

NodeContainer nodos;
nodos.Create (2);

WifiHelper wifi;
if (verbose)
{
    wifi.EnableLogComponents ();
}

```

```

//Establece el estandár con el que va a trabajar
wifi.SetStandard (WIFI_PHY_STANDARD_80211g);

YansWifiPhyHelper wifiPhy = YansWifiPhyHelper::Default ();

wifiPhy.Set ("RxGain", DoubleValue (0));
wifiPhy.SetPcapDataLinkType (YansWifiPhyHelper::DLT_IEEE802_11_RADIO);

YansWifiChannelHelper wifiChannel;
wifiChannel.SetPropagationDelay ("ns3::ConstantSpeedPropagationDelayModel");
wifiChannel.AddPropagationLoss ("ns3::LogDistancePropagationLossModel",
                               "Exponent", DoubleValue (3.0),
                               "ReferenceLoss", DoubleValue (46.6777));
wifiPhy.SetChannel (wifiChannel.Create ());

NqosWifiMacHelper wifiMac = NqosWifiMacHelper::Default ();
wifi.SetRemoteStationManager ("ns3::ConstantRateWifiManager",
                             "DataMode", StringValue (phyMode),
                             "ControlMode", StringValue (phyMode));

// Configuración de la estación
Ssid ssid = Ssid ("wifi-default");

wifiMac.SetType ("ns3::StaWifiMac",
                 "Ssid", SsidValue (ssid),
                 "ActiveProbing", BooleanValue (false));
NetDeviceContainer staDevice = wifi.Install (wifiPhy, wifiMac, nodos);
NetDeviceContainer devices = staDevice;

// Configuración del AP

```

```

wifiMac.SetType ("ns3::ApWifiMac",
    "Ssid", SsidValue (ssid));
NetDeviceContainer apDevice = wifi.Install (wifiPhy, wifiMac, ap);
devices.Add (apDevice);

//Configurar movilidad
MobilityHelper mobility;
mobility.SetPositionAllocator ("ns3::GridPositionAllocator",
    "MinX", DoubleValue (0.0),
    "MinY", DoubleValue (0.0),
    "DeltaX", DoubleValue (distancia),
    "DeltaY", DoubleValue (distancia),
    "GridWidth", UintegerValue (2),
    "LayoutType", StringValue ("RowFirst"));
mobility.SetMobilityModel("ns3::ConstantPositionMobilityModel");
mobility.Install (ap);
mobility.Install (nodos);

InternetStackHelper internet;
internet.Install (ap);
internet.Install (nodos);

Ipv4AddressHelper ipv4;
NS_LOG_INFO ("Assign IP Addresses.");
ipv4.SetBase ("192.168.1.0", "255.255.255.0");
Ipv4InterfaceContainer i = ipv4.Assign (devices);

TypeId tid = TypeId::LookupByName ("ns3::UdpSocketFactory");
Ptr<Socket> recvSink = Socket::CreateSocket (ap.Get (0), tid);
InetSocketAddress local = InetSocketAddress (Ipv4Address::GetAny (), 80);
recvSink->Bind (local);

```

```

recvSink->SetRecvCallback (MakeCallback (&ReceivePacket));

Ptr<Socket> source = Socket::CreateSocket (nodos.Get (0), tid);
InetSocketAddress remote = InetSocketAddress (Ipv4Address ("255.255.255.255"),
80);
source->SetAllowBroadcast (true);
source->Connect (remote);

wifiPhy.EnablePcap ("/root/ns3/ns-allinone-3.22/ns-
3.22/src/resultados/infra_g/infra_ap", apDevice);
wifiPhy.EnablePcap ("/root/ns3/ns-allinone-3.22/ns-
3.22/src/resultados/infra_g/infra_estaciones", staDevice);
std::string animFile = "//root/ns3/ns-allinone-3.22/ns-
3.22/src/resultados/infra_g/anim1.xml" ;

cmd.AddValue ("animFile", "File Name for Animation Output", animFile);
AnimationInterface anim (animFile);

Simulator::ScheduleWithContext (source->GetNode ()->GetId (),
Seconds (1.0), &GenerateTraffic,
source, tamapaquete, numpaquetes, interPacketInterval);

NS_LOG_UNCOND (" Prueba, envia " << numpaquetes << " paquetes, distancia de "
<< distancia << "m" );

ns3::PacketMetadata::Enable ();
Simulator::Stop (Seconds (30.0));
Simulator::Run ();
Simulator::Destroy ();

return 0;
}

```

ANEXO 8: ESTÁNDAR IEEE 802.11b – WIFI - MODO AD-HOC

```

#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/mobility-module.h"
#include "ns3/config-store-module.h"
#include "ns3/wifi-module.h"
#include "ns3/internet-module.h"
#include "ns3/olsr-helper.h"
#include "ns3/ipv4-static-routing-helper.h"
#include "ns3/ipv4-list-routing-helper.h"
#include "ns3/netanim-module.h"

#include <iostream>
#include <fstream>
#include <vector>
#include <string>

using namespace ns3;

NS_LOG_COMPONENT_DEFINE ("WifiAdhoc");

void ReceivePacket (Ptr<Socket> socket)
{
    while (socket->Recv ())
    {
        NS_LOG_UNCOND ("PAQUETE RECIBIDO!");
    }
}

static void GenerateTraffic (Ptr<Socket> socket, uint32_t pktSize,
                           uint32_t pktCount, Time pktInterval )
{
    if (pktCount > 0)

```

```

{
    socket->Send (Create<Packet> (pktSize));
    Simulator::Schedule (pktInterval, &GenerateTraffic,
                         socket, pktSize,pktCount-1, pktInterval);
}
else
{
    socket->Close ();
}
}

```

```

int main (int argc, char *argv[])
{
    std::string phyMode ("DsssRate11Mbps");
    double distancia = 50; //metros
    uint32_t tamapaquete = 1024; // bytes
    uint32_t numpaquetes = 10;
    double intervalo = 1.0; // segundos
    bool verbose = false;

    CommandLine cmd;

    cmd.AddValue ("phyMode", "Wifi Phy mode", phyMode);
    cmd.AddValue ("tamapaquete", "tamaño del paquete enviado", tamapaquete);
    cmd.AddValue ("numpaquetes", "número de paquetes generado", numpaquetes);
    cmd.AddValue ("intervalo", "intervalo entre el envio de paquetes", intervalo);
    cmd.AddValue ("verbose", "turn on all WifiNetDevice log components", verbose);
    cmd.AddValue ("distancia", "distancia entre los nodos", distancia);
}

```

```

cmd.Parse (argc, argv);

// Convertir en objeto de tiempo
Time interPacketInterval = Seconds (intervalo);

// Deshabilitar la fragmentación de los tramas inferiores a 2200 bytes
Config::SetDefault ("ns3::WifiRemoteStationManager::FragmentationThreshold",
StringValue ("2200"));

// Desactive RTS / CTS para las tramas inferiores a 2200 bytes
Config::SetDefault ("ns3::WifiRemoteStationManager::RtsCtsThreshold", StringValue
("2200"));

// Fijar la velocidad de transmisión de datos
Config::SetDefault ("ns3::WifiRemoteStationManager::NonUnicastMode",
StringValue (phyMode));

NodeContainer nodos;
nodos.Create (2);

WifiHelper wifi;
if (verbose)
{
    wifi.EnableLogComponents ();
}
wifi.SetStandard (WIFI_PHY_STANDARD_80211b);

YansWifiPhyHelper wifiPhy = YansWifiPhyHelper::Default ();
// ponerlo a cero; De lo contrario, se agregará ganancia
wifiPhy.Set ("TxGain", DoubleValue(0)); // Ganancia de transmisión
wifiPhy.Set ("RxGain", DoubleValue (0) ); // Ganancia de recepción
wifiPhy.SetPcapDataLinkType (YansWifiPhyHelper::DLT_IEEE802_11_RADIO);

```

```

YansWifiChannelHelper wifiChannel;
wifiChannel.SetPropagationDelay ("ns3::ConstantSpeedPropagationDelayModel");
wifiChannel.AddPropagationLoss ("ns3::LogDistancePropagationLossModel",
                               "Exponent", DoubleValue (3.0),
                               "ReferenceLoss", DoubleValue (46.6777));
wifiPhy.SetChannel (wifiChannel.Create ());

```

```

NqosWifiMacHelper wifiMac = NqosWifiMacHelper::Default ();
wifi.SetRemoteStationManager ("ns3::ConstantRateWifiManager",
                             "DataMode", StringValue (phyMode),
//Establecer como modo ad-hoc
wifiMac.SetType ("ns3::AdhocWifiMac");
NetDeviceContainer devices = wifi.Install (wifiPhy, wifiMac, nodos);

```

```

//Configurar movilidad
MobilityHelper mobility;
mobility.SetPositionAllocator ("ns3::GridPositionAllocator",
                             "MinX", DoubleValue (0.0),
                             "MinY", DoubleValue (0.0),
                             "DeltaX", DoubleValue (distancia),
                             "DeltaY", DoubleValue (distancia),
                             "GridWidth", UIntegerValue (2),
                             "LayoutType", StringValue ("RowFirst"));
mobility.SetMobilityModel ("ns3::ConstantPositionMobilityModel");
mobility.Install (nodos);

```

```

InternetStackHelper internet;
internet.Install (nodos);

Ipv4AddressHelper inter;
NS_LOG_INFO ("Asignación de direcciones IPs.");

```

```

inter.SetBase ("192.168.1.0", "255.255.255.0");
Ipv4InterfaceContainer i = inter.Assign (devices);

TypeId tid = TypeId::LookupByName ("ns3::UdpSocketFactory");
Ptr<Socket> recvSink = Socket::CreateSocket (nodos.Get (0), tid);
InetSocketAddress local = InetSocketAddress (Ipv4Address::GetAny (), 80);
recvSink->Bind (local);
recvSink->SetRecvCallback (MakeCallback (&ReceivePacket));

Ptr<Socket> source = Socket::CreateSocket (nodos.Get (1), tid);
InetSocketAddress remote = InetSocketAddress (Ipv4Address ("255.255.255.255"),
80);
source->SetAllowBroadcast (true);
source->Connect (remote);

wifiPhy.EnablePcap ("/root/ns3/ns-allinone-3.22/ns-
3.22/src/resultados/adhoc_b/adhoc", devices);
AsciiTraceHelper ascii;
wifiPhy.EnableAsciiAll (ascii.CreateFileStream ("/root/ns3/ns-allinone-3.22/ns-
3.22/src/resultados/adhoc_b/adhoc-b.tr"));
std::string animFile = "/root/ns3/ns-allinone-3.22/ns-
3.22/src/resultados/adhoc_b/anim1.xml" ;

cmd.AddValue ("animFile", "File Name for Animation Output", animFile); // bagian
NetAnim
AnimationInterface anim (animFile);

Simulator::ScheduleWithContext (source->GetNode ()->GetId (),
Seconds (1.0), &GenerateTraffic,
source, tamapaquete, numpaquetes, interPacketInterval);

```

```
NS_LOG_UNCOND (" Prueba desde el " << "nodo 1" << " hacia el " << "nodo 0" <<
", número de paquetes enviados: "<< numpaquetes << " , distancia entre los nodos: "
<< distancia << "m");
```

```
ns3::PacketMetadata::Enable ();
Simulator::Run ();
Simulator::Destroy ();

return 0;
}
```

ANEXO 9: ESTÁNDAR IEEE 802.11g – WIFI - MODO AD-HOC

```

#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/mobility-module.h"
#include "ns3/config-store-module.h"
#include "ns3/wifi-module.h"
#include "ns3/internet-module.h"
#include "ns3/olsr-helper.h"
#include "ns3/ipv4-static-routing-helper.h"
#include "ns3/ipv4-list-routing-helper.h"
#include "ns3/netanim-module.h"

#include <iostream>
#include <fstream>
#include <vector>
#include <string>

using namespace ns3;

NS_LOG_COMPONENT_DEFINE ("WifiSimpleAdhoc");

void ReceivePacket (Ptr<Socket> socket)
{
    while (socket->Recv ())
    {
        NS_LOG_UNCOND ("PAQUETE RECIBIDO!");
    }
}

static void GenerateTraffic (Ptr<Socket> socket, uint32_t pktSize,
                           uint32_t pktCount, Time pktInterval )
{
    if (pktCount > 0)

```

```

{
    socket->Send (Create<Packet> (pktSize));
    Simulator::Schedule (pktInterval, &GenerateTraffic,
        socket, pktSize,pktCount-1, pktInterval);
}
else
{
    socket->Close ();
}
}

```

```

int main (int argc, char *argv[])
{
    std::string phyMode ("ErpOfdmRate54Mbps");
    double distancia = 20; // metros
    uint32_t tamapaquete = 1024; // bytes
    uint32_t numpaquetes = 10;
    double intervalo = 1.0; // segundos
    bool verbose = false;

    CommandLine cmd;

    cmd.AddValue ("phyMode", "Wifi Phy mode", phyMode);

    cmd.AddValue ("tamapaquete", "tamaño del paquete enviado", tamapaquete);
    cmd.AddValue ("numpaquetes", "número de paquetes generado", numpaquetes);
    cmd.AddValue ("intervalo", "intervalo entre el envio de paquetes", intervalo);
    cmd.AddValue ("verbose", "turn on all WifiNetDevice log components", verbose);
    cmd.AddValue ("distancia", "distancia entre los nodos", distancia);
}

```

```

cmd.Parse (argc, argv);
// Convertir en objeto de tiempo
Time interPacketInterval = Seconds (intervalo);

// Deshabilitar la fragmentación de los tramas inferiores a 2200 bytes
Config::SetDefault ("ns3::WifiRemoteStationManager::FragmentationThreshold",
StringValue ("2200"));

// Desactive RTS / CTS para las tramas inferiores a 2200 bytes
Config::SetDefault ("ns3::WifiRemoteStationManager::RtsCtsThreshold", StringValue
("2200"));

// Fijar la velocidad de transmisión de datos
Config::SetDefault ("ns3::WifiRemoteStationManager::NonUnicastMode",
StringValue (phyMode));

NodeContainer nodos;
nodos.Create (2);

WifiHelper wifi;
if (verbose)
{
    wifi.EnableLogComponents ();
}

//Establece el estandár con el que va a trabajar
wifi.SetStandard (WIFI_PHY_STANDARD_80211g);

YansWifiPhyHelper wifiPhy = YansWifiPhyHelper::Default ();
// ponerlo a cero; De lo contrario, se agregará ganancia
wifiPhy.Set ("TxGain", DoubleValue(0)); // Ganancia de transmisión
wifiPhy.Set ("RxGain", DoubleValue (0 )); // Ganancia de recepción

wifiPhy.SetPcapDataLinkType (YansWifiPhyHelper::DLT_IEEE802_11_RADIO);

```

```

YansWifiChannelHelper wifiChannel;
wifiChannel.SetPropagationDelay ("ns3::ConstantSpeedPropagationDelayModel");
wifiChannel.AddPropagationLoss ("ns3::LogDistancePropagationLossModel",
                               "Exponent", DoubleValue (3.0),
                               "ReferenceLoss", DoubleValue (46.6777));
wifiPhy.SetChannel (wifiChannel.Create ());

NqosWifiMacHelper wifiMac = NqosWifiMacHelper::Default ();
wifi.SetRemoteStationManager ("ns3::ConstantRateWifiManager",
                             "DataMode",StringValue (phyMode),
                             "ControlMode",StringValue (phyMode));

//Establecer como modo ad-hochoc mode
wifiMac.SetType ("ns3::AdhocWifiMac");
NetDeviceContainer devices = wifi.Install (wifiPhy, wifiMac, nodos);

//Configurar movilidad
MobilityHelper mobility;
mobility.SetPositionAllocator ("ns3::GridPositionAllocator",
                             "MinX", DoubleValue (0.0),
                             "MinY", DoubleValue (0.0),
                             "DeltaX", DoubleValue (distancia),
                             "DeltaY", DoubleValue (distancia),
                             "GridWidth", UintegerValue (2),
                             "LayoutType", StringValue ("RowFirst"));
mobility.SetMobilityModel ("ns3::ConstantPositionMobilityModel");
mobility.Install (nodos);

InternetStackHelper internet;
internet.Install (nodos);

Ipv4AddressHelper inter;

```

```

NS_LOG_INFO ("Asignación de direcciones IPs.");
inter.SetBase ("192.168.1.0", "255.255.255.0");
Ipv4InterfaceContainer i = inter.Assign (devices);

TypeId tid = TypeId::LookupByName ("ns3::UdpSocketFactory");
Ptr<Socket> recvSink = Socket::CreateSocket (nodos.Get (0), tid);
InetSocketAddress local = InetSocketAddress (Ipv4Address::GetAny (), 80);
recvSink->Bind (local);
recvSink->SetRecvCallback (MakeCallback (&ReceivePacket));

Ptr<Socket> source = Socket::CreateSocket (nodos.Get (1), tid);
InetSocketAddress remote = InetSocketAddress (Ipv4Address ("255.255.255.255"),
80);
source->SetAllowBroadcast (true);
source->Connect (remote);

wifiPhy.EnablePcap ("/root/ns3/ns-allinone-3.22/ns-
3.22/src/resultados/adhoc_g/adhoc", devices);
AsciiTraceHelper ascii;
wifiPhy.EnableAsciiAll (ascii.CreateFileStream ("/root/ns3/ns-allinone-3.22/ns-
3.22/src/resultados/adhoc_g/adhoc-g.tr"));
std::string animFile = "/root/ns3/ns-allinone-3.22/ns-
3.22/src/resultados/adhoc_g/anim1.xml" ;

cmd.AddValue ("animFile", "File Name for Animation Output", animFile);
AnimationInterface anim (animFile);

Simulator::ScheduleWithContext (source->GetNode ()->GetId (),
Seconds (1.0), &GenerateTraffic,
source, tamapaquete, numpaquetes, interPacketInterval);
NS_LOG_UNCOND (" Prueba desde el " << "nodo 1" << " hacia el " << "nodo 0" <<

```

", número de paquetes enviados: "<< numpaquetes << " , distancia entre los nodos: "
<< distancia << "m");

```
ns3::PacketMetadata::Enable ();
```

```
Simulator::Run ();
```

```
Simulator::Destroy ();
```

```
return 0;
```

```
}
```

ANEXO 10: ESTÁNDAR IEEE 802.11s

```

#include "ns3/core-module.h"
#include "ns3/internet-module.h"
#include "ns3/network-module.h"
#include "ns3/applications-module.h"
#include "ns3/wifi-module.h"
#include "ns3/mesh-module.h"
#include "ns3/mobility-module.h"
#include "ns3/mesh-helper.h"
#include "ns3/ipv4-global-routing-helper.h"
#include "ns3/flow-monitor-helper.h"
#include "ns3/netanim-module.h"

#include <iostream>
#include <sstream>
#include <fstream>

using namespace ns3;

NS_LOG_COMPONENT_DEFINE ("mesh-hwmp");

class MeshTest
{
public:
    /// Prueba de inicio
    MeshTest ();
    /// Configurar la prueba desde los argumentos de la línea de comandos
    void Configure (int argc, char ** argv);
    /// Prueba de ejecución
    int Run ();
private:
    int     m_xSize;
    int     m_ySize;

```

```

double m_step;
double m_randomStart;
double m_totalTime;
double m_packetInterval;
uint16_t m_packetSize;
uint32_t m_nIfaces;
bool m_chan;
bool m_pcap;
std::string m_stack;
std::string m_root;

/// Lista de nodos de red
NodeContainer nodes;
/// Lista de todos los dispositivos de punto de malla
NetDeviceContainer meshDevices;
// Direcciones de interfaces:
Ipv4InterfaceContainer interfaces;
// MeshHelper. El informe no es un método estático
MeshHelper mesh;
private:
    /// Create nodes and setup their mobility
    void CreateNodes ();
    /// Instalar m_stack de Internet en los nodos
    void InstallInternetStack ();
    /// Instalar aplicaciones
    void InstallApplication ();
    /// Imprimir el diagnóstico de dispositivos de malla
    void Report ();
};

MeshTest::MeshTest () :
    m_xSize (8),
    m_ySize (8),

```

```

m_step (100.0),
m_randomStart (0.1),
m_totalTime (50.0),
m_packetInterval (0.1),
m_packetSize (1024),
m_nIfaces (1),
m_chan (true),
m_stack ("ns3::Dot11sStack"),
m_root ("ff:ff:ff:ff:ff:ff")

{
}

void
MeshTest::Configure (int argc, char *argv[])
{
  CommandLine cmd;
  cmd.AddValue ("x-size", "Número de nodos en una cuadrícula de filas.", m_xSize);
  cmd.AddValue ("y-size", "Número de filas en una cuadrícula.", m_ySize);
  cmd.AddValue ("step", "Separación entre los nodos ", m_step);
  cmd.AddValue ("start", "Retardo de inicio aleatorio máximo, segundos",
m_randomStart);

  cmd.AddValue ("time", "Tiempo de simulación, segundos. [100 s]", m_totalTime);
  cmd.AddValue ("packet-interval", "Intervalo entre paquetes en ping UDP, segundos",
m_packetInterval);

  cmd.AddValue ("packet-size", "Tamaño de paquetes en ping UDP", m_packetSize);
  cmd.AddValue ("interfaces", "Número de interfaces de radio utilizadas por cada punto
de malla.", m_nIfaces);

  cmd.AddValue ("channels", "Utilice diferentes canales de frecuencia para diferentes
interfaces.", m_chan);

  cmd.AddValue ("stack", "Tipo de pila de protocolos. Ns3 :: Dot11sStack por defecto",
m_stack);

  cmd.AddValue ("root", "Dirección MAC del punto de malla raíz en HWMP", m_root);
}

```

```

cmd.Parse (argc, argv);
NS_LOG_DEBUG ("Grid:" << m_xSize << "*/*" << m_ySize);
NS_LOG_DEBUG ("Tiempo de simulación: " << m_totalTime << " s");
}

void
MeshTest::CreateNodes ()
{
//Crear estaciones m_ySize * m_xSize para formar una topología de cuadrícula
nodes.Create (m_ySize*m_xSize);
// Configuración YansWifiChannel
YansWifiPhyHelper wifiPhy = YansWifiPhyHelper::Default ();
YansWifiChannelHelper wifiChannel = YansWifiChannelHelper::Default ();
wifiPhy.SetChannel (wifiChannel.Create ());

//Crear ayudante de malla y establecer instalador de pila a ella
//El instalador de pilas crea todos los protocolos necesarios e instala en los dispositivos
de punto de malla
mesh = MeshHelper::Default ();
if (!Mac48Address (m_root.c_str ()).IsBroadcast ())
{
    mesh.SetStackInstaller (m_stack, "Root", Mac48AddressValue (Mac48Address
(m_root.c_str ())));
}
else
{
    // Si root no está establecido, no utilizamos el atributo "Root", porque se especifica
sólo para 11s
    mesh.SetStackInstaller (m_stack);
}
if (m_chan)
{.....
of.close ();

```

```
}

int
main (int argc, char *argv[])
{
    MeshTest t;
    t.Configure (argc, argv);
    return t.Run ();
}
```

ANEXO 11: ALM

ANEXO 12: MODELO DEL REPORTE DEL ESTÁNDAR IEEE 802.11s

```

- <MeshPointDevice time="300.3" address="00:00:00:00:00:01">
  <Statistics txUnicastData="417" txUnicastDataBytes="437660" txBroadcastData="2" txBroadcastDataBytes="56" rxUnicastData="426" rxUnicastDataBytes="447128" rxBroadcastData="1"
    rxBroadcastDataBytes="28" fwdUnicastData="0" fwdUnicastDataBytes="0" fwdBroadcastData="1" fwdBroadcastDataBytes="28"/>
- <Interface BeaconInterval="0.5" Channel="0" Address="00:00:00:00:00:01">
  <Statistics rxBeacons="1202" txFrames="476" txBytes="445690" rxFrames="546" rxBytes="457518"/>
</Interface>
- <Hwmp address="00:00:00:00:00:01" maxQueueSize="255" Dot11MeshHWMPmaxPREQretries="3" Dot11MeshHWMPnetDiameterTraversalTime="0.1024"
  Dot11MeshHWMPpreqMinInterval="0.1024" Dot11MeshHWMPperrMinInterval="0.1024" Dot11MeshHWMPactiveRootTimeout="5.12" Dot11MeshHWMPactivePathTimeout="5.12"
  Dot11MeshHWMPpathToRootInterval="2.048" Dot11MeshHWMPrannInterval="5.12" isRoot="0" maxTtl="32" unicastPerrThreshold="32" unicastPreqThreshold="1" unicastDataThreshold="1"
  doFlag="0" rfFlag="1">
  <Statistics txUnicast="417" txBroadcast="3" txBytes="441048" droppedTtl="0" totalQueued="4" totalDropped="0" initiatedPreq="4" initiatedPrep="44" initiatedPerr="0"/>
- <HwmpProtocolMac address="00:00:00:00:00:01">
  <Statistics txPreq="4" txPrep="44" txPerr="0" rxPreq="77" rxPrep="13" rxPerr="19" txMgt="48" txMgtBytes="1704" rxMgt="109" rxMgtBytes="3926" txData="420" txDataBytes="441104"
    rxData="428" rxDataBytes="450608"/>
</HwmpProtocolMac>
</Hwmp>
- <PeerManagementProtocol>
  <Statistics linksTotal="2" linksOpened="3" linksClosed="1"/>
- <PeerManagementProtocolMac address="00:00:00:00:00:01">
  <Statistics txOpen="3" txConfirm="3" txClose="2" rxOpen="4" rxConfirm="3" rxClose="2" dropped="0" brokenMgt="0" txMgt="8" txMgtBytes="362" rxMgt="9" rxMgtBytes="398"
    beaconShift="3"/>
</PeerManagementProtocolMac>
<PeerLink localAddress="00:00:00:00:00:01" peerInterfaceAddress="00:00:00:00:00:0a" peerMeshPointAddress="00:00:00:00:00:0a" metrica="154" lastBeacon="300.02" localLinkId="2"
  peerLinkId="3" assocId="1"/>
<PeerLink localAddress="00:00:00:00:00:01" peerInterfaceAddress="00:00:00:00:00:02" peerMeshPointAddress="00:00:00:00:00:02" metrica="150" lastBeacon="300.025" localLinkId="3"
  peerLinkId="6" assocId="2"/>

```

**ANEXO 13: TABLAS DE LOS RESULTADOS DE LAS PRUEBAS DEL
PROTOCOLO IEEE 802.11s EN DIFERENTES MALLAS, VARIANDO
LA DISTANCIA ENTRE NODOS**

Tabla 7 Resultados del protocolo IEEE 802.11s para una malla formada por 4 nodos.

Resultados del protocolo IEEE 802.11s para una malla formada por 4 nodos. s

DISTANCIA(m)	TX	RX	LS
25	500	500	0
50	500	500	0
75	500	500	0
110	500	311	189
125	500	0	500
130	500	0	500

Nota: Fuente Autor.

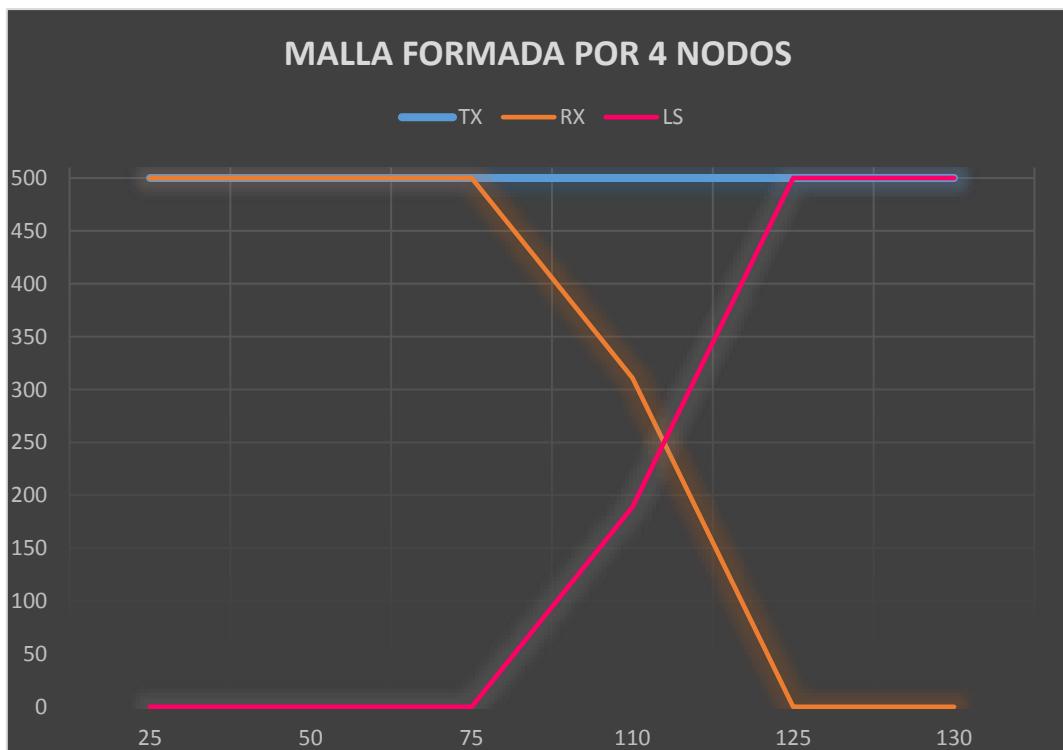


Figura 130: Malla formada por 4 nodos variando la distancia entre nodos.

Fuente: El autor.

Tabla 8

Resultados del protocolo IEEE 802.11s para una malla formada por 10 nodos. s

DISTANCIA(m)	TX	RX	LS
25	500	492	8
50	500	489	11
75	500	0	500
110	500	381	119
125	500	0	500
130	500	0	500

Nota: Fuente Autor.

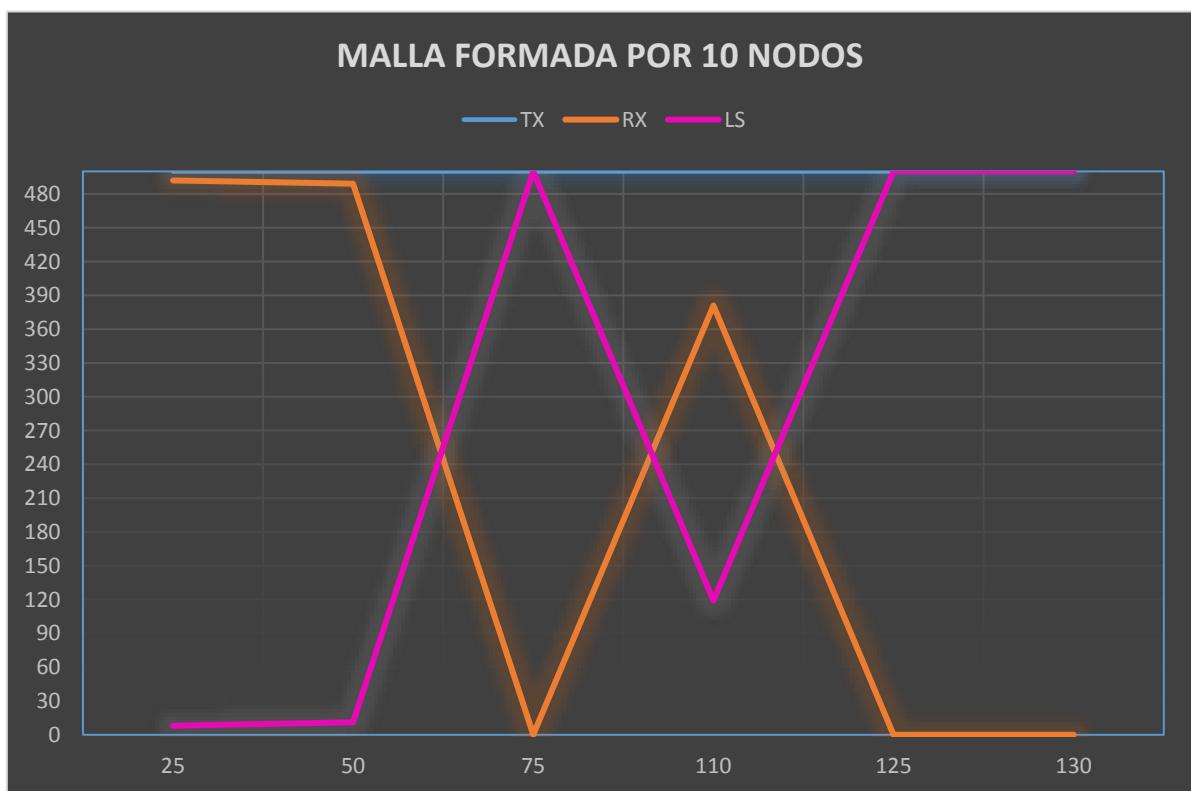


Figura 131 Malla formada por 10 nodos variando la distancia entre nodos.

Fuente: El autor.

Tabla 9

Resultados del protocolo IEEE 802.11s para una malla formada por 15 nodos

DISTANCIA	TX	RX	LS
25	500	495	5
50	500	491	9
75	500	0	500
110	500	457	43
125	500	0	500
130	500	0	500

Nota: Fuente Autor.

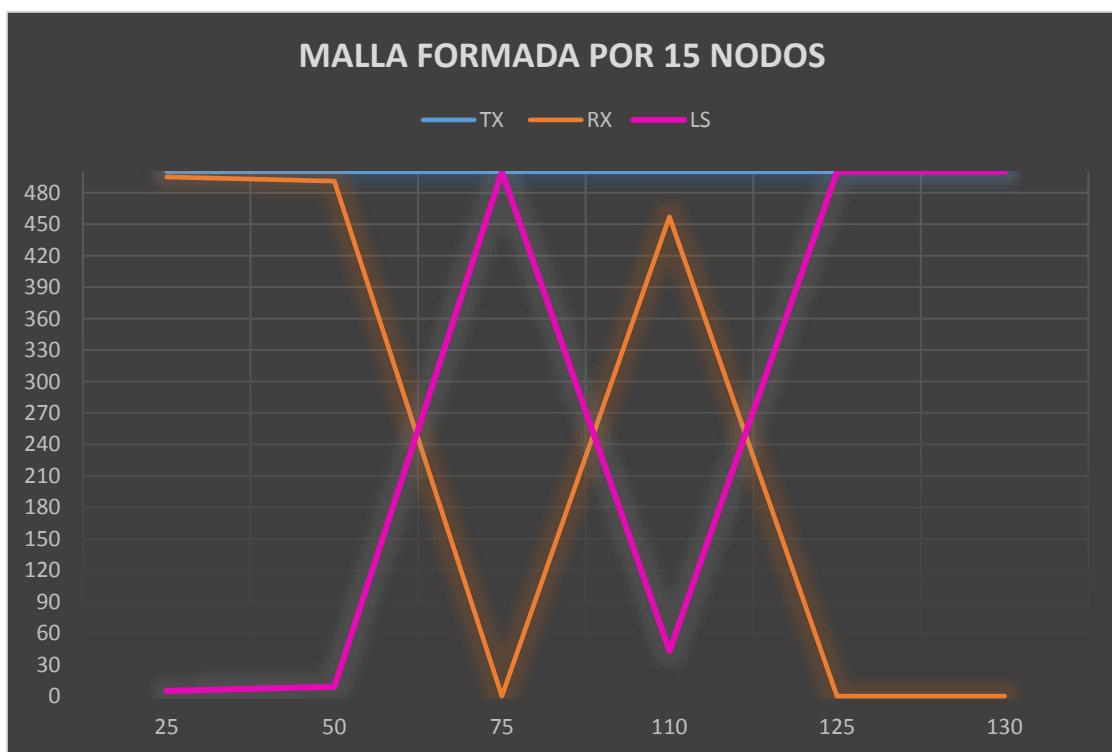


Figura 132 Malla formada por 15 nodos variando la distancia entre nodos.

Fuente: El autor.

Tabla 10

Resultados del protocolo IEEE 802.11s para una malla formada por 30 nodos

DISTANCIA	TX	RX	LS
75	500	0	500
110	500	392	108
125	500	0	500
130	500	0	500

Nota: Fuente Autor.

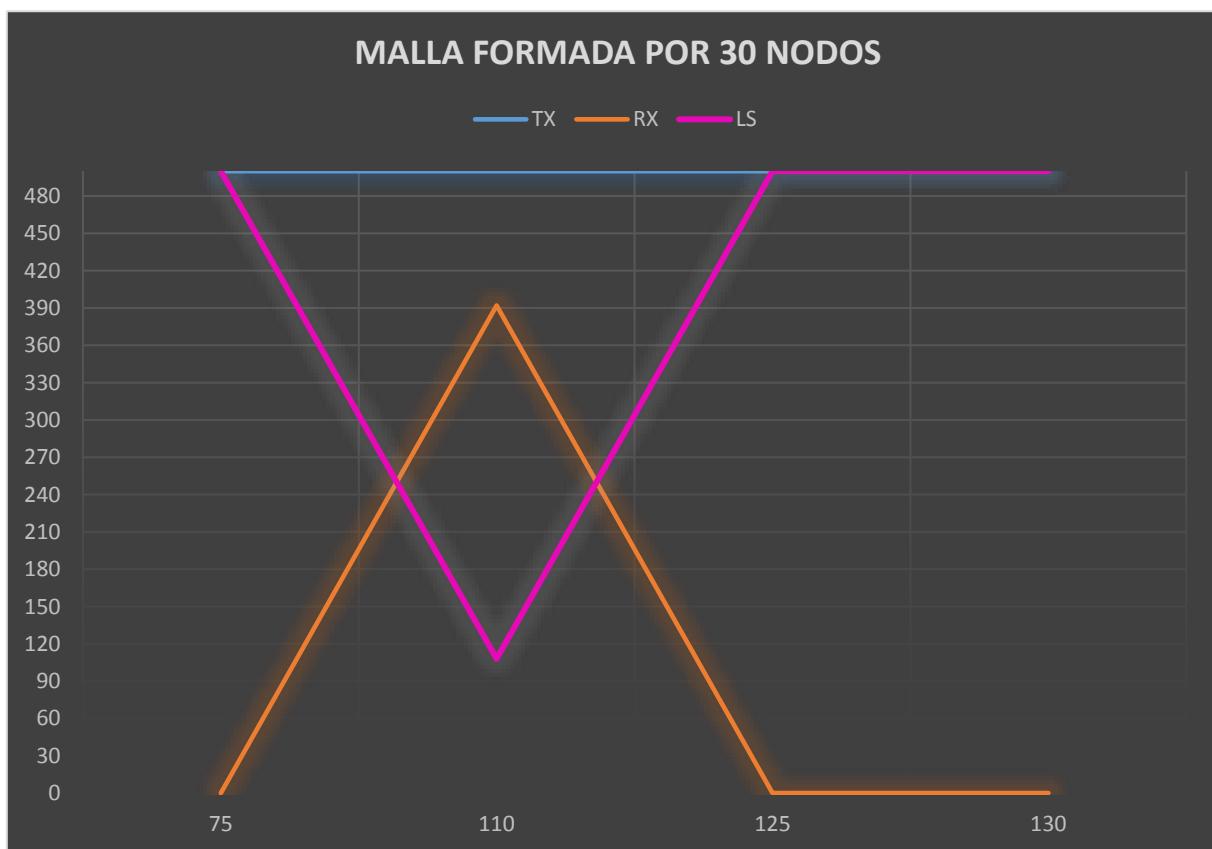


Figura 133 Malla formada por 30 nodos variando la distancia entre nodos.

Fuente: El autor.

Tabla 11

Resultados del protocolo IEEE 802.11s para una malla formada por 36 nodos

DISTANCIA	TX	RX	LS
75	500	0	500
110	500	306	194
125	500	0	500
130	500	0	500

Nota: Fuente Autor.

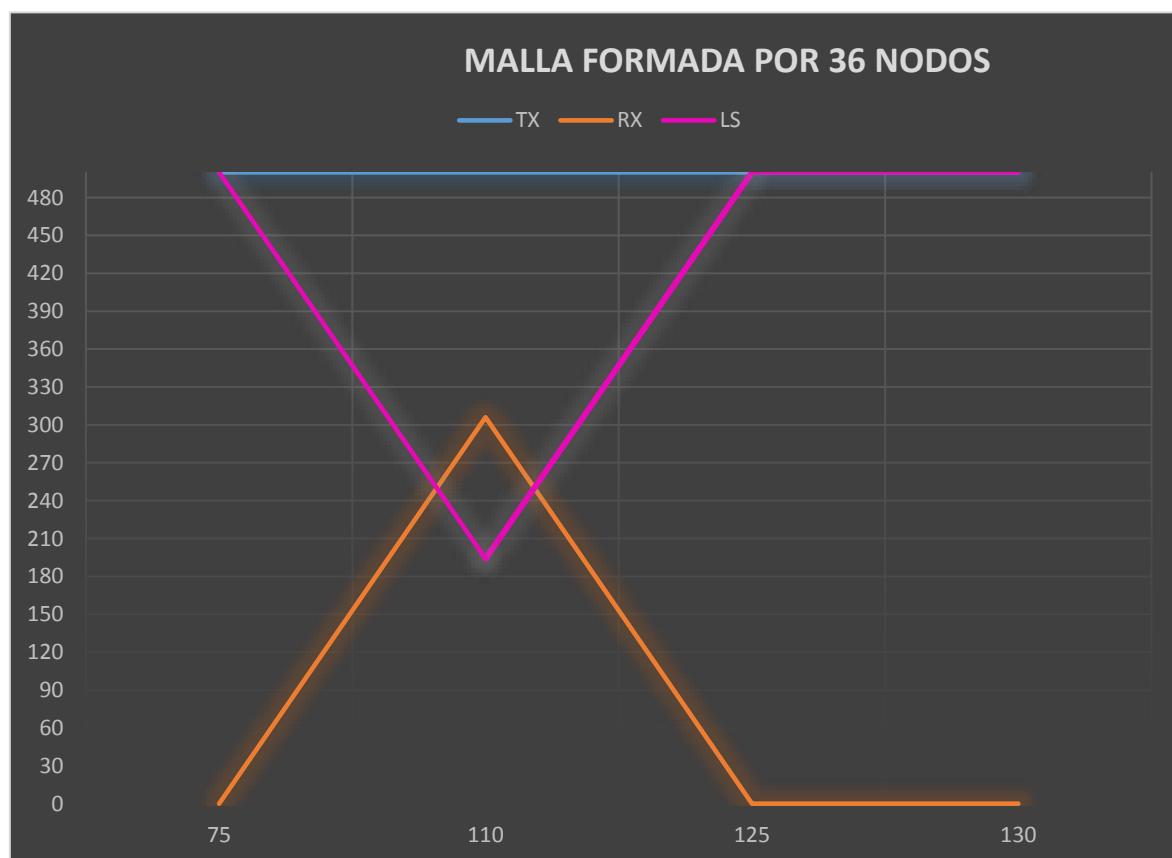


Figura 134 Malla formada por 36 nodos variando la distancia entre nodos.
Fuente: El autor.

Tabla 12

Resultados del protocolo IEEE 802.11s para una malla formada por 49 nodos.

DISTANCIA	TX	RX	LS
75	500	0	500
110	500	0	500
125	500	0	500
130	500	0	500

Nota: Fuente Autor.

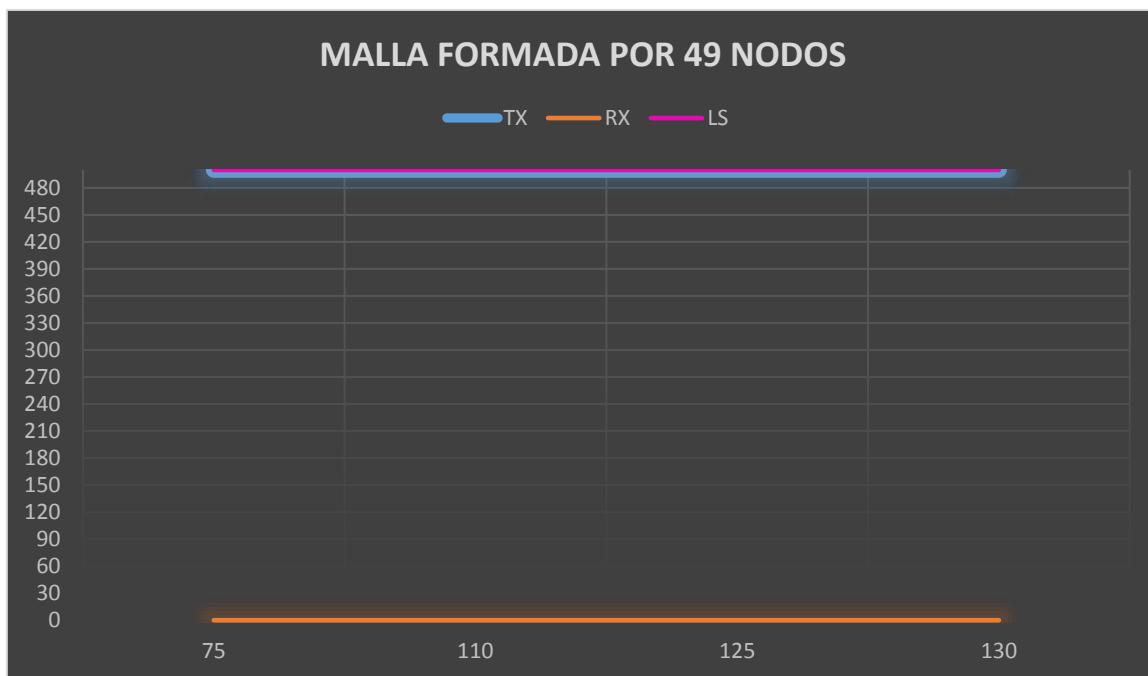


Figura 135 Malla formada por 49 nodos variando la distancia entre nodos.

Fuente: El autor.