Construcción de un módulo de control virtualizado para realizar balanceo de carga en MANETs

Óscar David Rueda Dimaté

Universidad Nacional de Colombia
Facultad de Ingeniería
Departamento de Ingeniería de Sistemas e Industríal
Bogotá, Colombia
2021

Construcción de un módulo de control virtualizado para realizar balanceo de carga en MANETs

Óscar David Rueda Dimaté

Tesis presentada como requisito parcial para optar al título de:
Magister en Ingeniería - Telecomunicaciones

Director:
Jorge Eduardo Ortiz Triviño Ph.D.

Línea de Investigación:
Computación aplicada - Telecomunicaciones
Grupo de Investigación en Redes de Telecomunicaciones Dinámicas y Lenguajes de Programación Distribuidos

Universidad Nacional de Colombia
Facultad de Ingeniería
Departamento de Ingeniería de Sistemas e Industíal
Bogotá, Colombia
2021

A mi abuelita, padres, hermanas y sobrinas por acompañarme en este momento de muchos otros.

Agradecimientos

Agradezco al Grupo de Investigación en Redes de Telecomunicaciones Dinámicas y Lenguajes de Programación Distribuidos Tlön del Departamento de Ingeniería de Sistemas e Industrial de la Universidad Nacional de Colombia y especialmente a su director Jorge Eduardo Ortiz Triviño y a mis asesores de tesis Henry Zárate Ceballos y Óscar Agudelo Rojas, por su apoyo y valiosos aportes en la realización de este trabajo.

Declaración de obra original

Yo declaro lo siguiente:

He leído el Acuerdo 035 de 2003 del Consejo Académico de la Universidad Nacional. Reglamento sobre propiedad intelectual y la Normatividad Nacional relacionada al respeto de los derechos de autor. Esta disertación representa mi trabajo original, excepto donde he reconocido las ideas, las palabras, o materiales de otros autores.

Cuando se han presentado ideas o palabras de otros autores en esta disertación, he realizado su respectivo reconocimiento aplicando correctamente los esquemas de citas y referencias bibliográficas en el estilo requerido.

He obtenido el permiso del autor o editor para incluir cualquier material con derechos de autor (por ejemplo, tablas, figuras, instrumentos de encuesta o grandes porciones de texto).

Por último, he sometido esta disertación a la herramienta de integridad académica, definida por la universidad.

Óscar David Rueda Dimaté
Nombre

Fecha: 15 de octubre de 2021

Resumen

Título en español: Construcción de un módulo de control virtualizado para realizar balanceo de carga en MANETs

El paradigma de las Redes Definidas por Software (SDN) permite el uso de técnicas de virtualización para la administración y despliegue de servicios en las Redes Ad-hoc Móviles (MANET). Esto representa un desafío debido a las propiedades de auto-organización, necesidad de conservación de la energía, infraestructura descentralizada y escalabilidad. Una arquitectura de virtualización apropiada habilita funcionalidades como el balanceo de carga en sistemas distribuidos, como el propuesto en el proyecto Tlön. Este trabajo, presenta un modelo que usa un agente local para crear un módulo de control virtualizado para realizar balanceo de carga en MANET, trabajando con la tecnología IEEE 802.11.

Palabras clave: Redes Ad-hoc Móviles, MANET, Redes Definidas por Software, SDN, controlador, virtualización.

Abstract

Título en inglés: Construction of a virtualized control module to perform load balancing in MANETs

Software Defined Networks (SDN) and Software Defined Wireless Networks (SDWN)- allow the use of virtualization techniques for managing and deploying services over Mobile Ad-hoc Networks (MANET). This can be a challenge due to the properties of self-organization, energy awareness, decentralized infrastructure and scalability of MANET. An appropriate virtualization architecture enables the load balancing over MANET on a distributed computer system, such as the one proposed in the Tlön project. This paper presents a model that uses a local agent to create a load balancer into the MANET, working with the Ad-hoc mode of IEEE 802.11 technology.

Keywords: Mobile Ad-hoc Networks, MANET, Software Defined Networks, SDN, controller, virtualization.

Esta tesis de maestría se sustentó el 1 de octubre de 2021 a las 9:00 am
y fue evaluada por los siguientes jurados:

Ingrid Patricia Páez Parra Ph.D.
Universidad Nacional de Colombia

Édgar Miguel Vargas Chaparro Ph.D.
Universidad Nacional de Colombia

Contenido

Lista de figuras

Lista de tablas

Abreviaturas

Abreviatura	Término
AL	Agente Local
ANS	Acuerdo de Nivel de Servicio
AP	Access Point
ARP	Address Resolution Protocol
AODV	Ad-hoc On-demand Distance Vector
B.A.T.M.A.N.	Better Approach to Mobile Ad-hoc Networking
CTS	Clear To Send
CRC	Cyclic Redundancy Check
DRM	Design Research Methodology
DSR	Dynamic Source Routing
eNodeB	enhanced NodeB
GSA	Gateway Selection Algorithm
HNA	Host Neighbor Announcements
HWMP	Hybrid Wireless Mesh Protocol
IEEE	Institute of Electrical and Electronics Engineers
InB	In-Band
IoT	Internet of Things
ITU	International Telecommunication Union
JLB	Jorge Luis Borges
JPS	Jean Paul Sartre
LAN	Local Area Network
LLDP	Link Layer Discovery Protocol
LTE	Long Term Evolution
MAC	Media Access Control
MANET	Mobile Ad-hoc Network
MTU	Maximum Transmission Unit
NAV	Network Allocation Vector
NFI	Network Fairness Index
NFV	Network Function Virtualization
OGM	Originator Message B.A.T.M.A.N.
OLSR	Open Link State Routing
OoB	Out-of-Band

Abreviatura	**Término**
OV	Operador Virtual
OVS	Open Virtual Switch
QoS	Quality-of-Service
RTS	Request To Send
RTT	Round Trip Time
RTP	Real-time Transport Protocol
SCME	Short Control Message Exchange
SDN	Software Define Networking
SDWN	Software Define Wireless Networking
SD-WAN	Software Define - Wide Area Network
SNR	Signal-to-Noise Ratio
SOHO	Small Office/Home Office
SOVORA	Sistema Operativo Virtualizado Orientado a Redes Ad-hoc
SSH	Secure Shell Protocol
SSID	Service Set ID
STP	Spanning Tree Protocol
TCP	Transmission Control Protocol
TQ	Transmission Quality
UDP	User Datagram Protocol
UD	User Device
VCS	Virtual Carrier Sensing
VLAN	Virtual LAN
VPN	Virtual Private Network
WHN	Wireless Home Network
WMN	Wireless Mesh Network
WNV	Wireless Network Virtualization
WSN	Wireless Sensor Network
ZRP	Zone Routing Protocol

Definiciones generales

- **Abstracción:** Según Liskov (2010), la abstracción es el centro de muchos de los trabajos en ciencias de la computación, que abarca desde la búsqueda de la interfaz adecuada para un sistema, así como la búsqueda de un diseño eficaz para la implementación del sistema. Además de ser la base para la construcción de programas, permitiendo que los programas se puedan construir de forma modular.
- **Hipervisor:** En computación se refiere a un monitor de máquina virtual. En *networking*, es la entidad de virtualización, encargada del control y la administración de los recursos de la red. Es la capa responsable del aislamiento entre diferentes particiones de una infraestructura virtualizada (Wen y cols., 2013).
- **Espectro radioeléctrico:** Las frecuencias del espectro electromagnético usadas para los servicios de difusión y servicios móviles, de policía, bomberos, radio astronomía, meteorología y fijos (Unión Internacional de Telecomunicaciones, 2005).

1. Introducción

La inclusión de las comunicaciones en los sistemas informáticos ha permitido la aparición de nuevas dinámicas entre los dispositivos y los usuarios, el despliegue de nuevos servicios, y una mayor demanda de nuevas características en los sistemas informáticos. La evolución del hardware y las comunicaciones ha dado lugar a un nuevo conjunto de tecnologías para utilizar sistemas tradicionales como Internet, educación, redes de sistemas de vigilancia y monitoreo, entre otros.

Hoy en día existen más tecnologías inalámbricas integradas en un dispositivo móvil que son compatibles con diferentes redes celulares como 2G (GSM CSD y GPRS), 3G (UMTS y HSDPA) y 4G (WiMAX, LTE y LTE-A). Estas tecnologías dependen de una estación base central y un conjunto de celdas para brindar servicios y cobertura. Sin embargo, también vienen equipadas para admitir comunicaciones de corto alcance que han evolucionado rápidamente gracias a la reducción de tamaño y al aumento de la capacidad informática de los microchips, lo que permite crear redes superpuestas dentro del mismo dispositivo. En esta evolución, dos tecnologías sobresalen: Bluetooth e IEEE 802.11.

Mediante el uso de dispositivos móviles inalámbricos, es posible construir redes dinámicas llamadas *Mobile Ad-hoc NETworks* (MANET), que pueden configurarse de forma autónoma, no necesitan control centralizado y pueden recuperarse automáticamente en caso de falla. Es posible agrupar nodos de una MANET en diferentes *clusters* para formar nubes móviles, las cuales son plataformas flexibles que permiten utilizar los recursos distribuidos (Fitzek y Katz, 2014). Este agrupamiento y elección de un *cluster-head* puede basarse en diferentes criterios como el agrupamiento del conjunto dominante, mantenimiento de la red, movilidad, eficiencia energética, balanceo de carga y métricas combinadas (J. Y. Yu y Chong, 2005).

Esta clase de redes permiten generar comportamientos autónomos y desplegar sistemas inteligentes en entornos informáticos, como la computación ubicua. Las infraestructuras y redes para interconectar un gran número de dispositivos de sensado y de cómputo en entornos urbanos, haciendo cada día más inteligente a los objetos, son el núcleo de las tecnologías del *Internet of Things* (IoT), junto con los mecanismos de almacenamiento, integración, procesamiento, análisis y manejo de la información generada del *big data* (Bibri, 2017).

El concepto del IoT apareció alrededor del 2010, refiriéndose a la integración del mundo

físico con el mundo informático (Y. Huang y Li, 2010). En este contexto las ”cosas“ son sensores, dispositivos embebidos, objetos físicos y virtuales, y sistemas inteligentes conectados a los seres humanos a través de Internet. Uno de los principales en este campo es el uso de las *Wireless Sensor Networks* (WSN) para crear ambientes robustos, tal como las ciudades inteligentes y sustentables, edificios inteligentes y agricultura inteligente, mejorando el monitoreo y el proceso de toma de decisiones.

Por otra parte, la Virtualización de Redes Inalámbricas (WNV) ha sido considerada una de las tecnologías más prometedoras para el desarrollo del Internet del futuro (Liang y Yu, 2015). La virtualización de los recursos de red inalámbricos, tales como el espectro radioeléctrico, es un desafió complejo. La virtualización y compartición de las redes inalámbricas traerá una utilización más eficiente de sus escasos recursos. Sin embargo, se necesita que esta compartición sea justa (Zaki, Zhao, Goerg, y Timm-Giel, 2010).

El incremento del número de servicios introducidos por las tecnologías de IoT, *Ubiquitous Sensor Networks* (USN), *Wireless Home Networks* (WHN), redes celulares 5G y otras, requiere arquitecturas de red que garanticen propiedades como la seguridad, escalabilidad, optimización energética, tolerancia a fallas y balanceo de carga (Haque y Abu-Ghazaleh, 2016). *Software Defined Networking* (SDN) es el paradigma que introduce la programabilidad en las arquitecturas de red y que promete facilitar su administración y operación.

SDN permite la virtualización de red con la abstracción de una interfaz que puede ser utilizada por un controlador para aislar y manipular los flujos del tráfico de datos. Más aún, SDN emplea un controlador para optimizar la eficiencia de múltiples redes, como por ejemplo, redes ópticas, redes móviles inalámbricas, redes de datacenter y de la computación en la nube (Turner y cols., 2008; W. Wang, Chen, Zhang, y Jiang, 2016; Wen y cols., 2013; Zhou, Wang, McLaughlin, y Zhou, 2015).

1.1. Identificación del problema

En el diseño de esquemas para MANET, se deben tener en cuenta varios aspectos de las tecnologías y servicios a prestar. Un desafío especial es el balanceo de carga, el cual se ha estudiado desde diferentes perspectivas (Haque y Abu-Ghazaleh, 2016). El creciente número de servicios que se prestan o se reciben en ambientes como el de Internet de las Cosas (IoT), redes de sensores inalámbricos, redes inalámbricas hogareñas y otras redes de tipo móvil con servicios en la nube, requieren esquemas de red seguros, escalables y que garanticen la continuidad de los servicios (Jo y Kim, 2016), así como el diseño de nuevos esquemas para implementar un control lógicamente centralizado, pero físicamente distribuido, que tenga en cuenta la inestabilidad de los enlaces de las redes inalámbricas.

Según Liang y Yu (2015), el espectro radioeléctrico es uno de los recursos más importantes en las comunicaciones inalámbricas.

Desafío

El desafío detrás de este trabajo de maestría es implementar un mecanismo de balanceo de carga simple que brinde una gestión eficiente del recurso del espectro radioeléctrico desde una virtualización de MANET.

Preguntas de investigación

- ¿Cómo implementar una virtualización de red para realizar balanceo de carga en MANET?
- ¿Cómo se puede definir la equidad en redes de telecomunicaciones?
- ¿Cómo se relaciona la equidad con el balanceo de carga en MANET?
- ¿Para qué puede servir una virtualización que permita el control del tráfico en MANET?

Aporte de la investigación

El principal aporte de este trabajo es el diseño altamente detallado de una arquitectura implementable que integra tecnologías clave para los sistemas inalámbricos distribuidos, como es el proyecto Tlön. Con esta arquitectura de virtualización de MANET pretendemos lograr una red bien balanceada que permita la implementación de los sistemas multiagentes y de las aplicaciones distribuidas en las capas superiores del sistema Tlön.

Este proyecto se enfoca en desarrollar una implementación sencilla de balanceo de carga y una evaluación experimental. El sistema de cómputo Tlön es el proyecto central del grupo de investigación con el mismo nombre. El proyecto Tlön es fundamentalmente un desarrollo experimental que busca dirigir el desarrollo de proyectos de valor que aporten a la construcción del sistema, generando nuevos conocimientos y adelantos en tecnologías que fortalezcan el crecimiento científico y económico del país.

1.2. Objetivos

Objetivo general

Construir un sub-sistema de control virtualizado para realizar balanceo de carga sobre el tráfico en MANETs para el sistema Tlön.

Para contruir un sub-sistema como el propuesto, seguimos varios pasos propuestos en los siguientes objetivos específicos:

Objetivos específicos

1. Identificar las estrategias de control virtualizado usadas para realizar balanceo de carga en MANETs.
2. Diseñar una estrategia de control virtualizado para realizar balanceo de carga sobre el tráfico de una MANET.
3. Implementar un módulo de control virtualizado para realizar balanceo de carga sobre el tráfico en una MANET.
4. Validar el funcionamiento del sub-sistema de control virtualizado en un escenario real.

Seguimiento al cumplimiento de los objetivos

Tabla 1-1.: Seguimiento al cumplimiento de los objetivos. Elaboración propia.

Objetivo	Capítulo	Secciones	Comentarios
1	4	4.2	Este objetivo se cumple específicamente con la Figura **4-3** y la Tabla **4-3**.
2	5	5.2	Este objetivo se cumple específicamente con las Figuras **5-4** y **5-5**.
3	6	6.1	La implementación se realizó en cada nodo de las Figuras **6-2** y **6-3**.
4	6	6.2	El funcionamiento se puede observar en las Figuras de la **6-8** a la **6-17**.

2. Tlön

Tlön es el mundo socialmente inspirado en que se desarrollan los minuciosos planes, diseños y arquitecturas de un sistema de computo distribuido basado en los principios aplicables de inmanencia de Spinoza (1980), existencia y esencia de Sartre (1943), estado de Hobbes (1996), justicia y equidad de Rawls (1995); Rawls y Kelly (2001) y paradigma de Kuhn (1962).

Figura 2-1.: Grupo de Investigación en Redes de Telecomunicaciones Dinámicas y Lenguajes de Programación Distribuidos (Tlön) en salida al Jardín Botánico de Bogotá el 12 de diciembre de 2020. Tomado de Tlön (2021).

Tlön también es una comunidad de personas que aporta en pequeñas partes a la construcción del proyecto desarrollado al interior del grupo de investigación con el mismo nombre.

Principios

Inmanencia: Es el modo en el que pueden darse todas las interacciones *causa-efecto* en el interior de un sistema (Spinoza, 1980).

Existencia y esencia: La existencia se refiere al propio soporte y no causalidad de todas las interacciones que son en sí en el interior de un sistema, que es la esencia (Sartre, 1943).

Estado: Es el sistema artificial instituido por pactos entre los hombre que lo componen y gobiernan y que ocurren cuando se da el paso del estado de la naturaleza al estado civil (Hobbes, 1996).

Justicia: Se puede definir como la primera virtud de las instituciones sociales. La justicia tomada como imparcialidad indica una distribución de recursos acorde con las capacidades naturales de cada individuo por lo cual todos los miembros de una sociedad, a pesar de su cantidad de recursos, perciben la misma posición social o estatus (Rawls, 1995).

Equidad: Según Rawls y Kelly (2001), se puede entender como el *debido* trato entre personas mutuamente auto interesadas, libres y que carecen de autoridad las unas sobre las otras, que están cooperando o compitiendo unas con otras en las *prácticas* en que se embarcan, y establecen o reconocen entre ellas las reglas que definen esa actividad y determinan las respectivas cuotas en beneficios y cargas. Se puede extraer del análisis sobre estas prácticas que la equidad es el concepto fundamental para la justicia (Rawls y Kelly, 2001, p. 143).

Paradigma: Se refiere a un conjunto de prácticas y saberes que definen una disciplina durante un periodo específico y que son ampliamente aceptados por las comunidades científicas (Kuhn, 1962).

Modelo de capas del sistema Tlön

La red ad-hoc del proyecto Tlön está conformada por una colección de dispositivos portables e inalámbricos, que se pueden mover libremente. Las funciones de control y administración propias de la red son asumidas por la entidad de virtualización de la capa superior, distribuida entre todos los nodos.

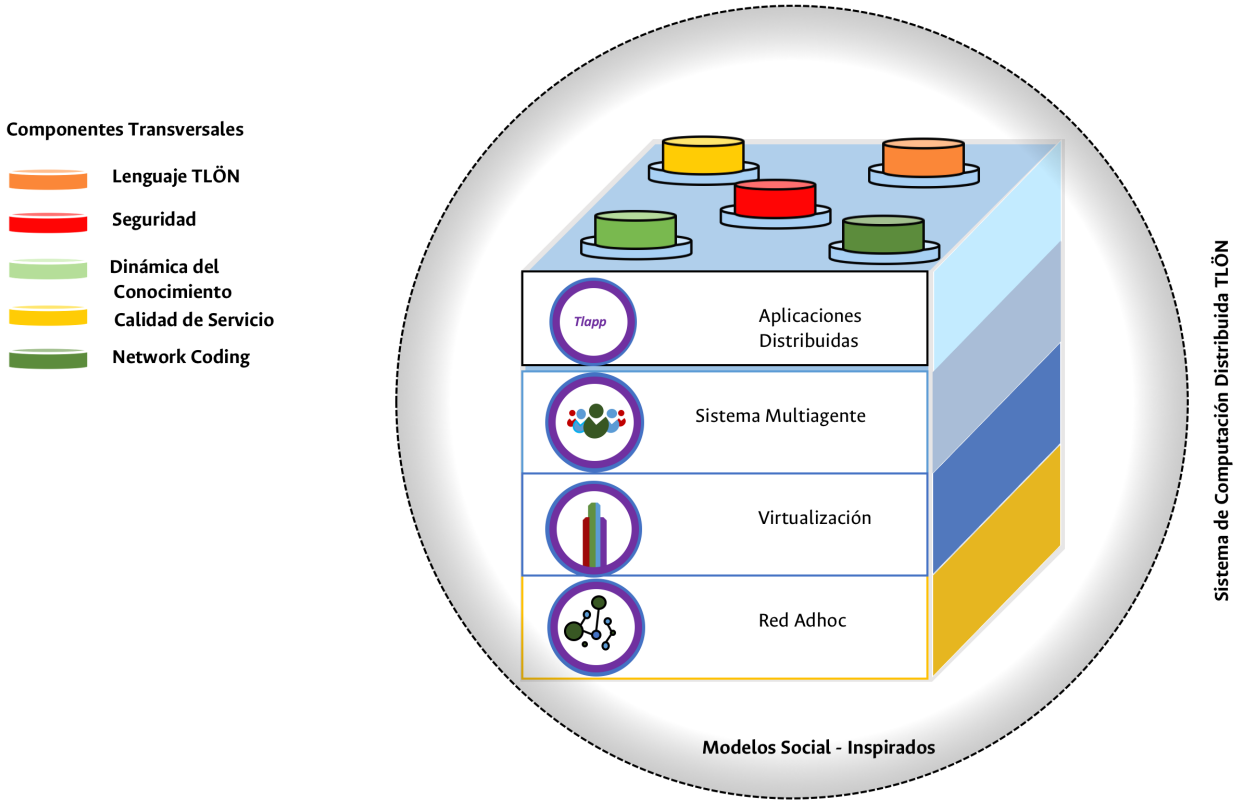


Figura 2-2.: Modelo de capas del sistema Tlön. Tomado de Tlön (2021).

Cada capa tiene las siguientes funcionalidades:

- **Red ad-hoc:** En esta capa se desarrollan las actividades de formación, mantenimiento y finalización de las conexiones entre los nodos que participan en la red ad-hoc de forma autónoma, escalable y de infraestructura descentralizada.

- **Virtualización:** En esta capa se desacoplan las funcionalidades del sistema entre las ofrecidas por los recursos distribuidos entre los nodos y las ofrecidas por las aplicaciones de las capas superiores.

- **Sistema multi-agente:** Esta capa permite que los agentes y las comunidades de agentes interactúen en sus propios ambientes y que realizan las acciones que requieran para cumplir sus objetivos de forma autónoma, dinámica y estocástica.

- **Aplicaciones:** En esta capa se provee al sistema con todos los medios para ofrecer a los usuarios los servicios que permitan utilizar los recursos.

Analogías del modelo social inspirado

Con base en el modelo social de pseudo estado de Hobbes (1996), en la Figura **2-2** se presenta una abstracción por capas del sistema de cómputo Tlön, donde cada capa de abstracción tiene su entidad análoga en el modelo de estado, como se describe en la Figura **2-3**.

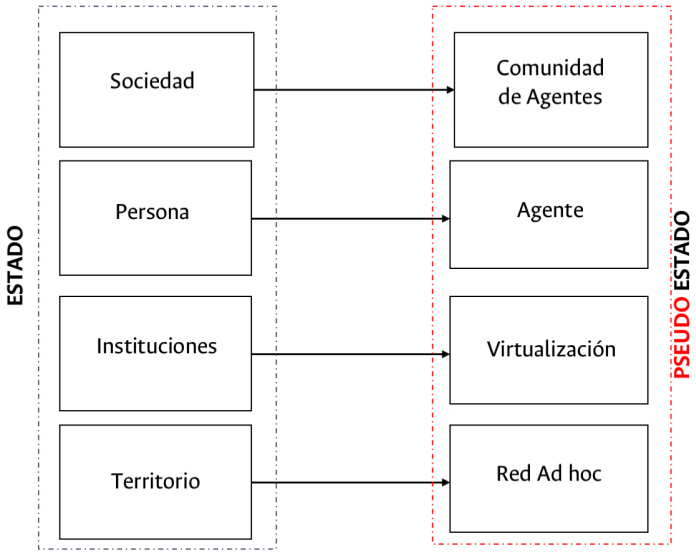
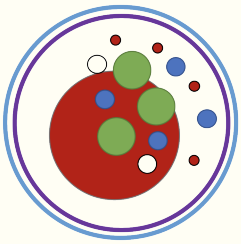


Figura 2-3.: Analogías entre estado y *pseudo* estado. Tomado de Tlön (2021).

Analogía entre territorio y MANET

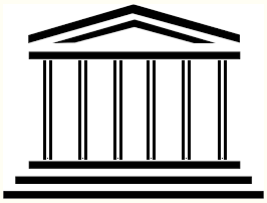
Las MANET son la primera capa de abstracción del modelo social en donde funciona el sistema de cómputo Tlön y son análogas al territorio en donde funciona un estado. Se utilizan en este modelo, dadas sus propiedades de topología dinámica y estocástica, cooperación entre nodos heterogéneos, infraestructura descentralizada y escalabilidad.



Territorio: Es el referente político-administrativo de los contenedores ajustables, a los cuales se les dan límites formales. Estos "contenedores" o "recipientes" ajustables o maleables, corresponden a las unidades concretas, pero transitorias, de ocupación humana en las que toma forma el espacio/tiempo (Fals Borda, 2000).

Analogía entre institución y virtualización

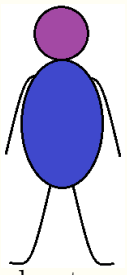
La virtualización es una abstracción del modelo social que administra los recursos disponibles en la red como la capacidad de procesamiento, el almacenamiento de información, almacenamiento de energía, y capacidad de interacción con el entorno. Es análoga a las instituciones que administran un estado.



Instituciones: Son organismos públicos o privados creados para desempeñar una determinada labor cultural, científica, política o social. Las instituciones son sistemas de índole social y cooperativa, creadas bajo imposiciones legales, que procuran ordenar y normalizar el comportamiento de un grupo de individuos.

Analogía entre personas y agentes inteligentes

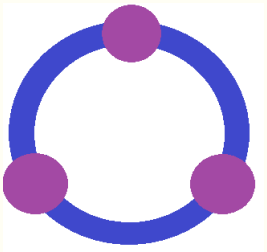
El desarrollo de la inteligencia artificial se ha aplicado el concepto de racionalidad a una variedad de agentes que se comportan, análogamente a las personas, dependiendo de la naturaleza del medio en el que existen.



Agentes inteligentes: Son entidades que perciben y actúan sobre el entorno con la ayuda de sensores y actuadores y que se pueden clasificar de acuerdo a los atributos que posean y que van a definir su comportamiento (Russell y Norvig, 2020).

Analogía entre sociedad y comunidad de agentes

Las comunidades de agentes son sistemas que, con base en conocimientos, tienen la habilidad de proveer soluciones a problemas complejos y son capaces de aprender de sus propios comportamientos en la solución de problemas, ampliando así su base de conocimientos y construyendo su historia (Nossum, 1987).



Sociedad: Es una asociación, más o menos autosuficiente, de personas que en sus relaciones generan comportamientos de colaboración y cooperación con el fin de obtener bienestar y/o felicidad. Sus miembros reconocen ciertas reglas de conducta como obligatorias y en su mayoría están de acuerdo con ellas (Rawls, 1995).

Política de balanceo de carga en Tlön

La presión sobre el territorio está aumentando debido al aumento de la población cada vez más móvil, la necesidad de espacio para vivir y trabajar, así como de infraestructuras de transporte. De manera análoga, cada vez más se restringen las comunicaciones inalámbricas en las MANET debido a problemas de interferencia entre las redes operando en el mimo territorio y a las limitaciones en el número de bandas de frecuencia no licenciadas.

Por último, pero no menos importante, esta la urgente necesidad de establecer políticas sobre el uso del espectro radioeléctrico y presentar esquemas y soluciones que técnicamente permitan hacer un uso equilibrado de la red construida y mantenida en un territorio por una comunidad.

Es por esto que se propone el siguiente artículo como política de balanceo de carga desde el sistema Tlön:

Artículo 100: Sobre el balanceo de carga del sistema Tlön.
(1) El sistema proveerá los mecanismos de control para la utilización de los recursos de la red como el espectro radioeléctrico y limitará las molestias causadas por el uso de dicho recurso a un nivel que no sea perjudicial para las personas, animales y plantas ni para su medio ambiente.
(2) El tráfico de control en el medio inalámbrico de extremo a extremo se transportará en una red separada del tráfico de datos. En la virtualización de red inalámbrica se tomará las medidas necesarias para implementar esta separación utilizando un controlador u entidad que implemente esta medida.
(3) La capacidad de la red de datos estará limitada por la tecnología inalámbrica que se disponga. Los desvíos para aliviar la presión del tráfico de datos en los enlaces de la MANET serán aplicados por un módulo de control virtualizado que implemente esta disposición.

Contribución de esta investigación al sistema Tlön

La contribución de esta investigación al sistema Tlön corresponde al módulo de balanceo ubicado en la capa de virtualización del modelo de capas de la Figura **2-2**, que será el encargado de implementar la política de balanceo de carga definida anteriormente. También, se observa en la Figura **2-4**, la integración de este módulo de balanceo con otros componentes del Sistema Operativo de Tlön.

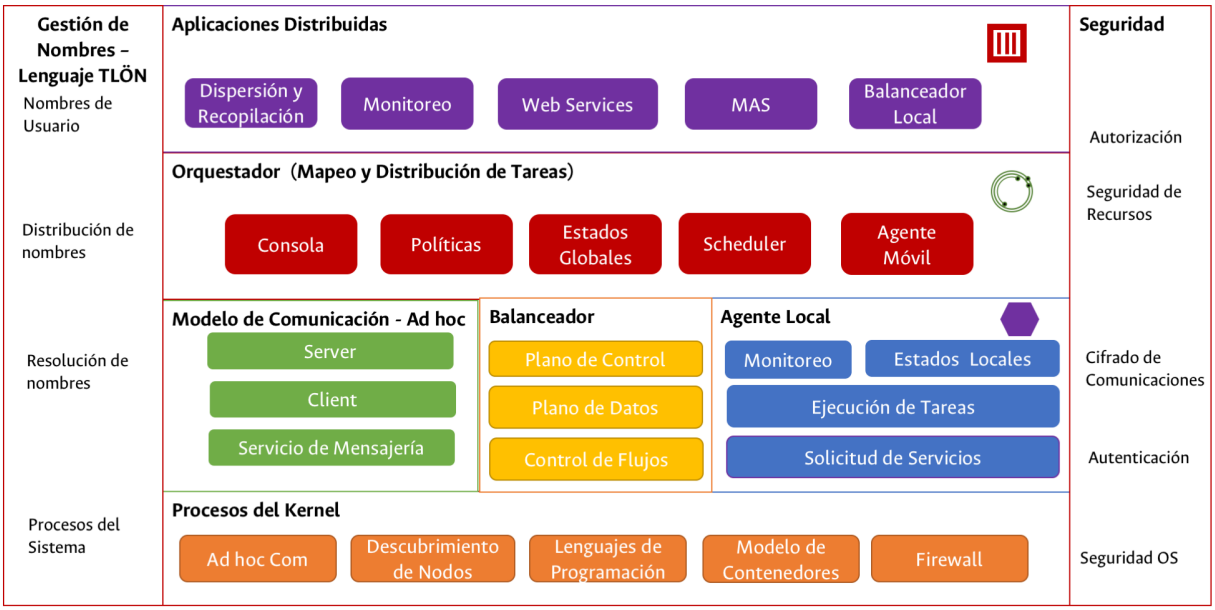


Figura 2-4.: Balanceador de carga dentro del Sistema Operativo Virtualizado Orientado a Redes ad-hoc (SOVORA). Tomado de Zárate-Ceballos y Ortiz-Triviño (2020).

3. Generalidades

3.1. Redes ad-hoc móviles

Las MANET son redes de dispositivos móviles conectados por interfaces inalámbricas, con un nivel de recursos dinámico, capaces de proveer servicios sin importar las condiciones estocásticas de los nodos al transcurrir el tiempo. Por esta razón, las MANET son apropiadas para montar sobre ellas un sistema de cómputo distribuido.

Este tipo de sistemas pueden ser capaces de generar comportamientos pseudosociales desde el instante de su conformación hasta el fin de su operación. La concepción de estas redes que se auto organizan y regulan permite problematizar en tendencias y necesidades de la sociedad, tales como las redes de telecomunicaciones de emergencia, IoT y ciudades inteligentes.

Según Ajmal, Jabeen, Rasheed, y Hasan (2015), la investigación en MANET continua siendo impulsada por el hecho de que estas redes representan el tipo más general de redes inalámbricas. Casi todos los otros tipos de redes inalámbricas, incluyendo la redes celulares, de sensores, vehiculares, de retransmisión, etc., pueden ser tratadas como subtipos de MANET. La ausencia de una autoridad central de control e infraestructura es una propiedad característica de estas redes. Por tanto, las funciones de control y administración son distribuidas entre los nodos en toda la red.

Algunas ventajas de estas redes inalámbricas incluyen la conectividad y los servicios de banda ancha que pueden prestar a muy bajo costo, en lugares donde el despliegue de una red cableada sería notablemente costosa como en las veredas o comunidades rurales (Chung y cols., 2012).

Formalmente, las redes ad-hoc son un grafo aleatorio con un conjunto de vértices, comúnmente llamados nodos. En este caso, los nodos son móviles y están unidos por un conjunto de enlaces denominados aristas, que cambian de forma dinámica en función del tiempo y las condiciones del ambiente, por ejemplo, las peticiones de los usuarios (Newman, 2003).

MANET

Son sistemas computacionales auto-organizados, formados por un conjunto de nodos que se comunican entre sí a través de enlaces inalámbricos y que no dependen de una infraestructura preexistente para funcionar (Ospina-López y Ortiz-Triviño, 2015).

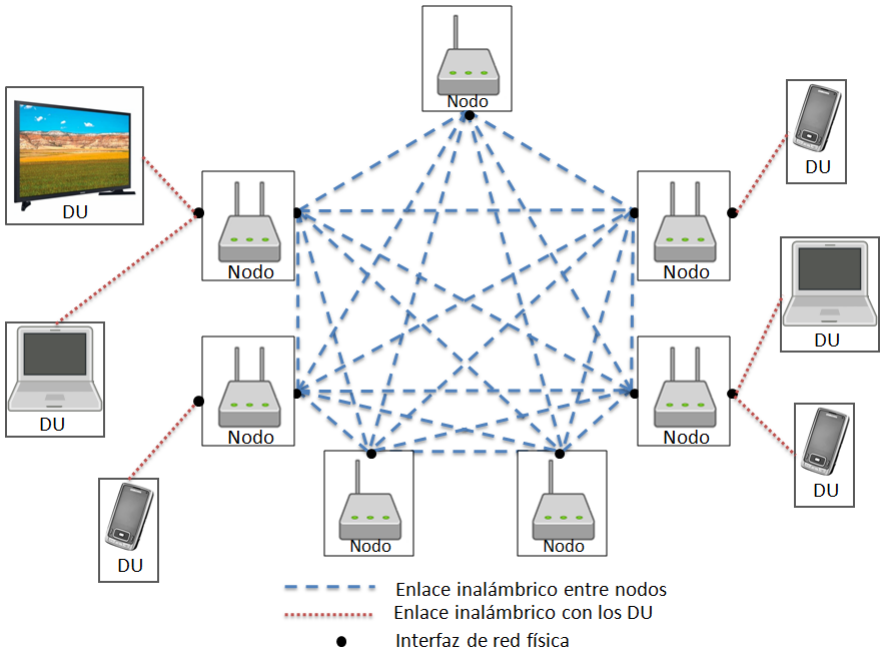


Figura 3-1.: Los nodos y los Dispositivos de Usuario (DU) de una MANET se comunican entre sí a través de enlaces inalámbricos. Elaboración propia.

Estándar IEEE 802.11

El estándar IEEE 802.11 fue creado en 1997 por el Instituto de Ingenieros Eléctricos y Electrónicos (IEEE) para las especificaciones de las capas física (PHY) y de *Medium Access Control* (MAC) para *Wireless Local Area Networks* (WLAN). El estándar original ha sido extendido en varias enmiendas para soportar mayores velocidades de datos, modos *mesh* y ad-hoc, movilidad, comunicación vehicular, diseño *cross-layer*, comunicación multimedia, mejor rendimiento y QoS (Raychaudhuri y Gerla, 2011).

IEEE 802.11 hace parte de la familia de especificaciones IEEE 802 que cuenta con las especificaciones para las dos capas inferiores del modelo de Interconexión de Sistemas Abiertos ISO/OSI: física y de enlace, como se observa en la Figura **3-2**. La subcapa MAC determina cómo acceder al medio inalámbrico y enviar datos, pero los detalles de la transmisión y recepción se dejan para la capa física. La capa física especifica las tecnologías de radio como

las incluidas inicialmente en 802.11 que son: *Frequency Hopping Spread Spectrum* (FHSS) y *Direct Sequence Spread Spectrum* (DSSS). Con las revisiones se incluyeron capas físicas como 802.11b que especifica una alta velocidad para DSSS (HR-DSSS) (Gast, 2005).

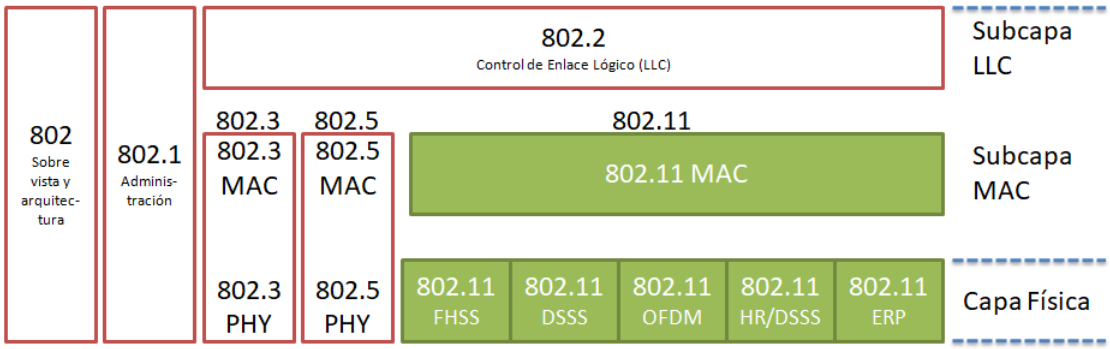


Figura 3-2.: Familia IEEE 802 y su relación con las capas física y de enlace del modelo OSI. Elaboración propia con base en Gast (2005).

En MANET generalmente se utiliza IEEE 802.11 como el protocolo de la subcapa MAC, que funciona con base en el mecanismo de *Carrier Sense Multiple Access with Collition Avoidance* (CSMA/CA) (Abbas y Kure, 2009). La Función de Coordinación Distribuida (FCD), está basada en CSMA/CA, especifica que cada nodo debe sensar el medio por una duración aleatoria antes de iniciar una transmisión y envía los paquetes solo cuando encuentra el canal libre durante todo el tiempo. De otro modo, el nodo espera un tiempo de *back-off* y vuelve a sensar el medio cuando expira este tiempo, y así continua intentando hasta encontrar el canal libre o dejar el sensado al superar un umbral de intentos.

Las redes IEEE 802.11 b, g y n utilizan el espectro radioeléctrico en las bandas de frecuencia ISM de los 2.4GHz, como se observa en la Figura **3-3**. Esta banda de 2.4 GHz se divide en 14 canales con un ancho de banda de 22MHz. Generalmente, se asignan los canales 1, 6 y 11 a diferentes redes ubicadas físicamente cerca para evitar la interferencia.

El problema del nodo oculto

El mecanismo CSMA/CA con el *back-off* aleatorio, puede reducir la probabilidad de colisiones pero no las elimina completamente. Algunas veces, los nodos no pueden comunicarse directamente entre ellos, como se observa en la Figura **3-4**. El nodo B puede comunicarse con ambos nodos A y C. El nodo A no está en el rango de recepción del nodo C y viceversa. Si A y C inician transmisiones simultáneamente, los datos no podrán ser recibidos correctamente por el nodo B.

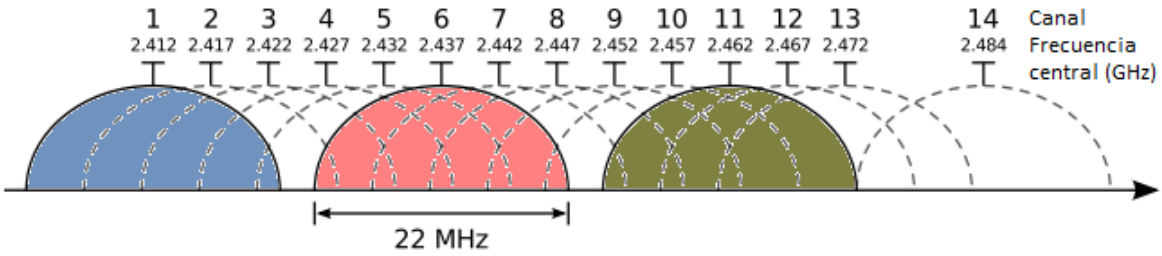


Figura 3-3.: Los canales 1, 6 y 11 del estándar IEEE802.11 en la banda ISM de 2.4GHz no se solapan en todo su ancho de banda de 22MHz. Elaboración propia.

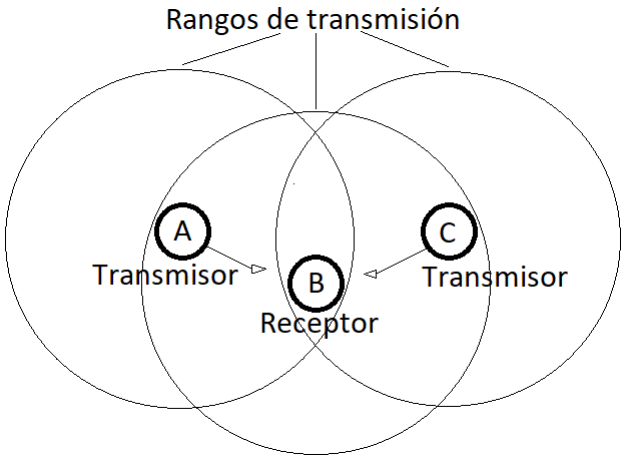


Figura 3-4.: A es un nodo oculto para el nodo C y viceversa. Al transmitir A y C simultáneamente hacia el nodo B se genera colisión. Elaboración propia con base en Raychaudhuri y Gerla (2011).

Para mitigar este problema, el estándar IEEE 802.11 introduce los conceptos de *Virtual Carrier Sensing* (VCS) y *Short Control Message Exchange* (SCME). Cada paquete transmitido contiene un valor que especifica el tiempo que el medio debe estar ocupado por el paquete y por todos los paquetes de control subsecuentes requeridos para completar satisfactoriamente la transmisión. Todos los nodos que escucharon la transmisión, fijan el parámetro *Network Allocation Vector* (NAV) y se abstienen de realizar cualquier transmisión durante esa duración, inclusive si el medio está desocupado (Raychaudhuri y Gerla, 2011).

En la Figura **3-5**, se ilustra la secuencia completa de una transmisión exitosa. El nodo transmisor comienza enviando un mensaje *Request To Send* (RTS) que contiene la dirección de origen y destino, y la duración de la secuencia completa de la transmisión. El RTS sirve para reservar el enlace de radio para la transmisión y silenciar cualquier otro nodo durante el tiempo de NAV-RTS. El nodo receptor confirma la transmisión enviando un mensaje *Clear*

To Send (CTS), con su propia dirección y otro campo de duración NAV-CTS. Solo entonces el nodo transmisor puede enviar los datos sin preocuparse por la presencia de cualquier nodo oculto (Gast, 2005).

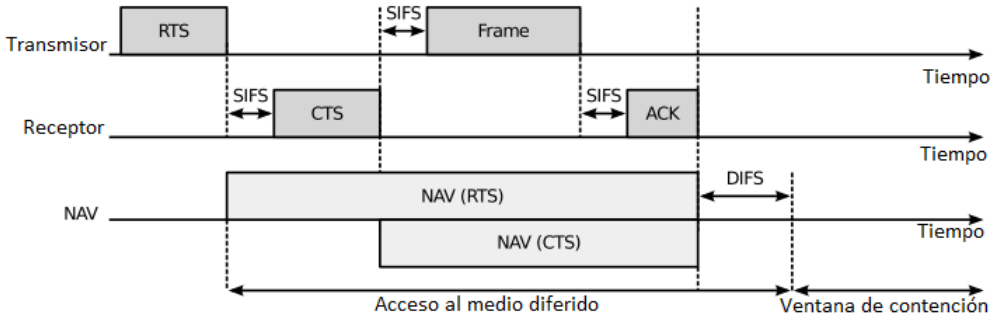


Figura 3-5.: Solución del problema del nodo oculto con el mecanismo RTS/CTS y la utilización del NAV para la detección de portadora virtual. Elaboración propia con base en Gast (2005).

Protocolos de enrutamiento para MANET

El enrutamiento es un componente principal de cualquier red ad-hoc, dado que es responsable de balancear la carga entre la fuente y el destino (Tashtoush y cols., 2014). Los protocolos de enrutamiento para MANET pueden ser divididos en reactivos, proactivos e híbridos, como se observa en la Figura **3-6**.

Los protocolos reactivos operan "bajo demanda" y no tienen un conocimiento previo de las rutas en la red hasta cuando un nodo fuente requiere una ruta para transmitir datos a un destino específico. Los protocolos reactivos más notables son el *Ad-hoc On-demand Distance Vector* (AODV) (Perkins, Belding-Royer, y Das, 2003) y el *Dynamic Source Routing* (DSR), que implementan los mecanismos de enrutamiento *hop-by-hop* y desde la fuente, respectivamente. La mayor desventaja de los protocolos reactivos es su alta latencia (Tashtoush y cols., 2014).

Los protocolos proactivos, como el protocolo *Optimized Link State Routing* (OLSR) (Clausen y Jacquet, 2003), necesitan información completa de las rutas entre cada par de nodos, sin importar si estos nodos requieren transmitir datos o no. La red también debe estar atenta a cada cambio que ocurra en la topología para actualizar las tablas de enrutamiento. La información sobre las rutas en la red es obtenida intercambiando mensajes de control periódicamente (Tashtoush y cols., 2014).

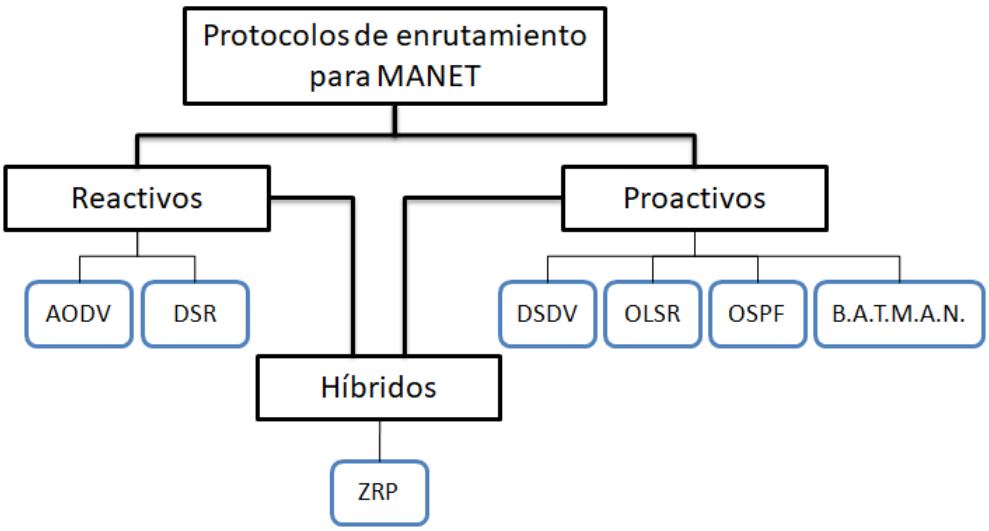


Figura 3-6.: Protocolos de enrutamiento para MANET. Elaboración propia con base en Tashtoush y cols. (2014).

Los protocolos híbridos mezclan características de ambos protocolos reactivos y proactivos para aprovechar sus ventajas. *Zone Routing Protocol* (ZRP) es uno de los protocolos híbridos más populares, donde los nodos de la red son agrupados dentro de zonas. Los nodos de la misma zona utilizan el protocolo proactivo, mientras que los nodos de diferentes zonas usan el protocolo reactivo. La zona se identifica por el número de saltos con respecto al nodo central (Tashtoush y cols., 2014).

Protocolo B.A.T.M.A.N.

Better Approach To Mobile Ad-hoc Networking (B.A.T.M.A.N.) inició en 2006 como un *daemon* en el espacio de usuario (batmand), y sirvió como alternativa al protocolo OLSR. En 2007, se inició el desarrollo del protocolo B.A.T.M.A.N. Advanced como un módulo de enrutamiento de capa de enlace que opera en el espacio de memoria del kernel, evitando así el copiado de paquetes desde y hacia el espacio de usuario, con lo que se mejoró el rendimiento. En el 2011, B.A.T.M.A.N. Adv. fue incluido en el kernel de Linux desde la versión 2.6.38, como resultado de la experimentación y estabilidad en el sistema (Seither, König, y Hollick, 2011).

B.A.T.M.A.N. pertenece a la familia de protocolos proactivos y se puede categorizar como un protocolo de vector-distancia con el mecanismo de enrutamiento *hop-by-hop*. El principio de operación de B.A.T.M.A.N. es similar de alguna forma al protocolo *Destination-Sequenced Distance-Vector* (DSDV). B.A.T.M.A.N. también utiliza mensajes de anuncio para el enrutamiento con Números de Secuencia (NS). La diferencia esta en que B.A.T.M.A.N. utiliza NS

para cada registro de la tabla de enrutamiento mientras que DSDV tiene un NS para toda la tabla. También, se diferencian en que B.A.T.M.A.N. envía periódicamente mensajes de anuncio que a la vez utiliza para medir la calidad de los enlaces, mientras que DSDV solo envía mensajes en respuesta a cambios en la tabla de enrutamiento o en la topología (Furlan, 2011).

Aunque B.A.T.M.A.N. presenta un rendimiento similar al de otros protocolos ya establecidos, tiene unas características especiales que surgen de su naturaleza bio-inspirada. Por ejemplo, B.A.T.M.A.N. sigue estrictamente un enfoque de enrutamiento descentralizado y da un manejo a los mensajes e información sobre la red que puede ser aprovechada por otros métodos. Esto lo convierte en una plataforma de investigación prometedora para la optimización y el desarrollo de nuevos métodos de evaluación de la calidad de los enlaces (Sliwa, Falten, y Wietfeld, 2019).

El protocolo detecta la presencia de "originadores" B.A.T.M.A.N., sin importar que el camino de comunicación desde/hacia un originador sea de un solo salto o de múltiples saltos. B.A.T.M.A.N. no intenta encontrar la totalidad de caminos de enrutamiento, sino que solamente aprende cual de los vecinos con quienes tiene un enlace local es el mejor *gateway* hacia cada originador. También, realiza un seguimiento a los nuevos originadores e informa a los vecinos de su existencia. El protocolo se asegura que cada ruta consista únicamente de enlaces bidireccionales (Neumann y cols., 2008).

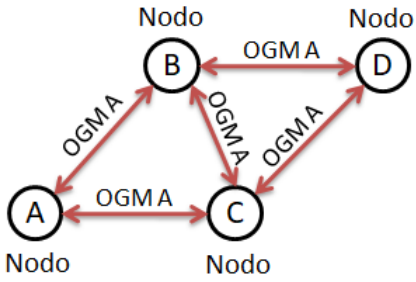


Figura 3-7.: Proceso de *broadcast* del OGM de B.A.T.M.A.N. Elaboración propia con base en Hundeboll y Ledet-Pedersen (2011).

Regularmente, cada nodo envía mediante *broadcast* UDP un Mensaje Originador (OGM) a sus vecinos de enlace local, informándoles su existencia. Estos vecinos de enlace local reenvían los OGM mediante *broadcast*, inundando la red con estos mensajes, como se observa en la Figura **3-7**. El número de OGM recibidos desde un originador dado, pasando por cada vecino de enlace local, es usado para estimar la calidad de una ruta de un solo salto o de múltiples saltos. Mediante el NS en cada OGM, B.A.T.M.A.N. puede distinguir entre OGM nuevos y OGM duplicados, asegurando que cada OGM se contado solo una vez (Neumann y cols., 2008).

El paquete B.A.T.M.A.N.

El paquete B.A.T.M.A.N. consta de un OGM y cero o más mensajes adicionados de *Host Neighbor Announcements* (HNA), transmitidos mediante *User Datagram Protocol* (UDP) por el puerto 4305 asignado al protocolo B.A.T.M.A.N. por la *Internet Assigned Numbers Authority* (IANA). En la Figura **3-8**, se observa un paquete B.A.T.M.A.N. con el OGM y un mensaje HNA.

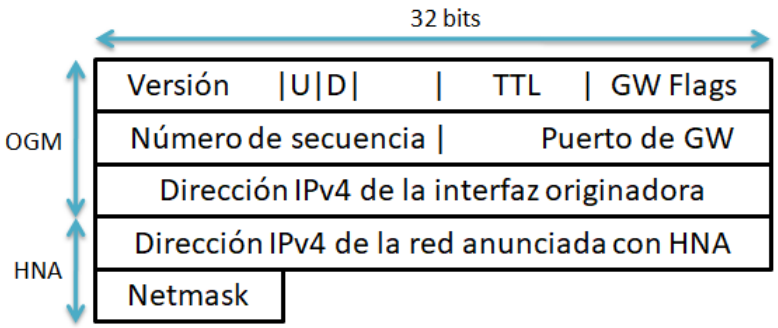


Figura 3-8.: Formato general del paquete B.A.T.M.A.N. Elaboración propia con base en Neumann y cols. (2008).

- Versión: Cada paquete recibido con este campo diferente a la versión del protocolo, debe ser ignorado.
- U: Esta bandera indica si un nodo es un vecino directo o no.
- D: Esta bandera indica si un vecino es bidireccional o no.
- Time To Live (TTL): El TTL puede ser usado para definir un límite superior del número de saltos en que un OGM puede ser transmitido.
- Bandera de gateway (GWFlags): Debe indicar si el originador ofrece un Gateway o no.
- Número de secuencia (NS): El originador numera consecutivamente cada OGM nuevo con un número de secuencia que incrementa uno por uno.
- Dirección del originador: Es la dirección IPv4 de la interfaz B.A.T.M.A.N. en la cual fue generado el OGM.
- Dirección de red HNA: Es la dirección IPv4 de la red anunciada por el originador.
- Máscara de red (netmask): Es el número de bits que presenta el tamaño de la red anunciada con HNA.

Anunciamiento de vecinos en B.A.T.M.A.N.

El mensaje OGM de B.A.T.M.A.N. puede incluir mensajes HNA sobre los *host* vecinos alcanzables a través del originador. Como estos *host* vecinos desconocen la existencia de la red *mesh* que utiliza el protocolo B.A.T.M.A.N., el originador que los conecta debe anunciarlos a los nodos vecinos. Esto se lleva a cabo adicionando una lista de las direcciones de los vecinos a cada OGM transmitido por el originador. Entonces, los demás nodos en la red pueden conservar una lista de los *host* vecinos a la red del originador al que están conectados. En la Figura **3-9**, se observa un escenario donde A está conectado al vecino X y C esta conectado al vecino Y. Cuando A recibe un paquete del vecino X dirigido al vecino Y, A sabe que el paquete debe ser transmitido a C a través de B.

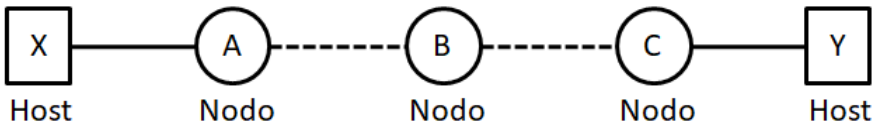


Figura 3-9.: Los nodos pueden proveer acceso a los *host* que no están en la red *mesh* con HNA. Elaboración propia con base en Hundeboll y Ledet-Pedersen (2011).

Estructura de datos del protocolo B.A.T.M.A.N.

El protocolo B.A.T.M.A.N. depende del mantenimiento, en cada nodo, de un conjunto de estructuras de datos que son diferentes para los originadores y para los vecinos de enlace local (Furlan, 2011).

Para cada originador O se mantiene un número de secuencia NS_O del último OGM recibido y una lista de los nodos candidatos desde los cuales se ha recibido al menos un OGM de O. Para cada nodo candidato $N_i(O)$, se mantiene una ventana deslizante que contiene el valor de *Transmission Quality* (TQ) leída en el OGM proveniente de O y recibida vía R_i. La calidad de un camino se obtiene calculando el promedio de los valores de TQ contenidos dentro de la ventana global.

Para cada vecino, se mantienen dos ventanas deslizantes de dimensión fija: la primera es usada para contar los OGM recibidos correctamente del vecino, mientras la segunda es necesaria para contar los OGM de *echo* recibidos del vecino. Estas dos ventanas son usadas para estimar la calidad de los enlaces hacia los vecinos.

Estimación de la calidad del enlace local en B.A.T.M.A.N.

La calidad de un enlace local es estimada indirectamente usando los OGM recibidos desde un nodo vecino. *Receive Quality* (RQ) se define como el porcentaje de OGM recibidos. Este

cálculo se realiza teniendo en cuenta el NS del último OGM recibido dentro de una ventana deslizante de tamaño N (128 por defecto). De manera similar, utilizando otra ventana deslizante, se calcula el porcentaje de los OGM de *echo*, que fueron iniciados por el mismo nodo y retransmitidos hacia el mismo nodo por un vecino, como se observa en la Figura **3-10**. A este valor se le conoce como *Echo Quality* (EQ). Finalmente, la TQ de A a B, se puede calcular con la siguiente ecuación:

$$EQ_{AB} = TQ_{AB} * RQ_{AB}$$
$$TQ_{AB} = \frac{EQ_{AB}}{RQ_{AB}} \tag{3-1}$$

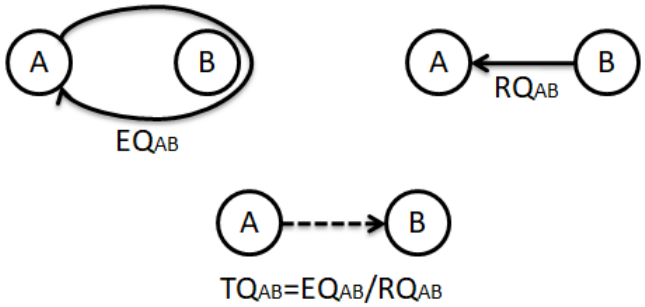


Figura 3-10.: Estimación de la calidad de transmisión TQ con la cuenta de los OGM de *echo* y de los OGM recibidos. Elaboración propia con base en Open-Mesh (2020).

Estructura del módulo B.A.T.M.A.N. Advanced

B.A.T.M.A.N. Adv. es el módulo del kernel de Linux que implementa el protocolo de enrutamiento en la capa 2 del modelo ISO/OSI. El módulo B.A.T.M.A.N. Adv. está integrado como una capa entre las interfaces de red y la interfaz de red virtual, como se ilustra en la Figura **3-11**. Las capas superiores que utilizan la interfaz de red virtual, no necesitan saber nada acerca del módulo B.A.T.M.A.N. Adv. o de la red *mesh* por la cual se comunican. Esta estructura hace que la red completa se comporte como un switch virtual que conmuta paquetes de una interfaz virtual a otra (Hundeboll y Ledet-Pedersen, 2011).

3.2. Redes definidas por software

Es ampliamente aceptado que las redes del futuro van a ser definidas por software (Zhou y cols., 2015).

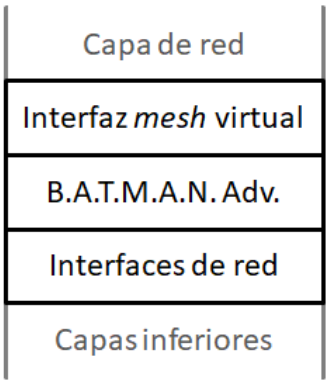


Figura 3-11.: Estructura de encapsulación de B.A.T.M.A.N. Adv. Elaboración propia con base en Hundeboll y Ledet-Pedersen (2011)

El paradigma de SDN introduce la programación a la administración de las redes inalámbricas y promete facilitar la gestión del espectro radioeléctrico, lo cual permitiría maximizar tanto la eficiencia espectral como la calidad de servicio (W. Wang y cols., 2016). SDN se puede entender como una tecnología complementaria a los protocolos de enrutamiento *mesh* tradicionales que permite una eficiente implementación de servicios de red, como por ejemplo el balanceo de carga (F. Yang y cols., 2014).

El fuerte despliegue de SDN y sus exitosos casos de uso en las redes tradicionales cableadas (p.e. *Open Virtual Switch* (OVS) (Linux Foundation, 2021) y *Software Defined - Wide Area Network* (SD-WAN) B4 de Google (S. Jain y cols., 2013)), incita a aprovechar las mismas posibilidades de desacoplamiento de las funciones de red y las funciones de control sobre las redes inalámbricas. SDN ha evolucionado rápidamente como la solución para satisfacer la creciente demanda de servicios inalámbricos y dinámicos de la computación ubicua.

En las arquitecturas de SDN, se tiene en cuenta los siguientes tres pilares:

- Desacoplamiento de los planos de control y de datos: La lógica de control es removida completamente de los dispositivos de red.
- Abstracción lógica de la red: La lógica de las redes tradicionales es abstraída de la implementación en hardware a niveles superiores definidos por software.
- Presencia de una entidad de control de red programable: El controlador interactúa y coordina a los dispositivos de red.

SDN básicamente virtualiza la arquitectura de la red aislando los tráficos de control y de datos, como se ilustra en las Figuras **3-12** y **3-13**. El plano de control esta lógicamente centralizado y físicamente distribuido y es capaz de obtener, actualizar e incluso predecir la

información global del estado de la red (Zhou y cols., 2015). Otra arquitectura dividida por planos es demostrada en (Zhou y cols., 2015).

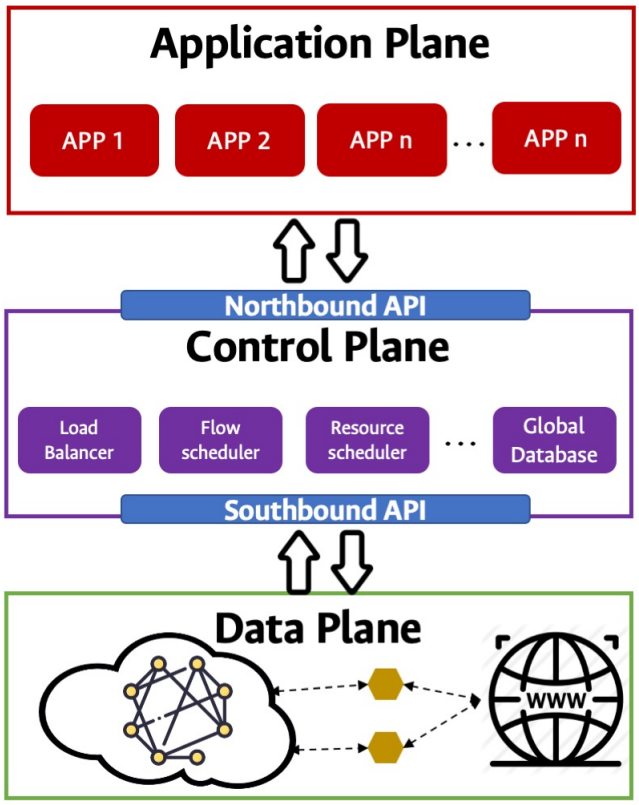


Figura 3-12.: Arquitectura básica de una SDN. Elaboración propia con base en H. Huang y cols. (2015); Jagadeesan y Krishnamachari (2015); M. Yang y cols. (2015).

Protocolo OpenFlow

OpenFlow es un protocolo abierto basado en el concepto de SDN. OpenFlow se implementó por primera vez en la Universidad de Standford por el equipo de Turner y cols. (2008), como una forma de experimentar con nuevos protocolos en redes reales como las ya existentes en el campus. OpenFlow se ha convertido rápidamente en la interfaz estándar entre los planos de control y de datos, permitiendo la comunicación entre un controlador y los nodos habilitados con este protocolo.

OpenFlow traslada la inteligencia del reenvío de los paquetes en la red a un controlador, permitiendo la programación de las tablas de reenvío de paquetes y manteniendo los nodos simples. En la estructura de un nodo tradicional, existe el dominio de control para la toma

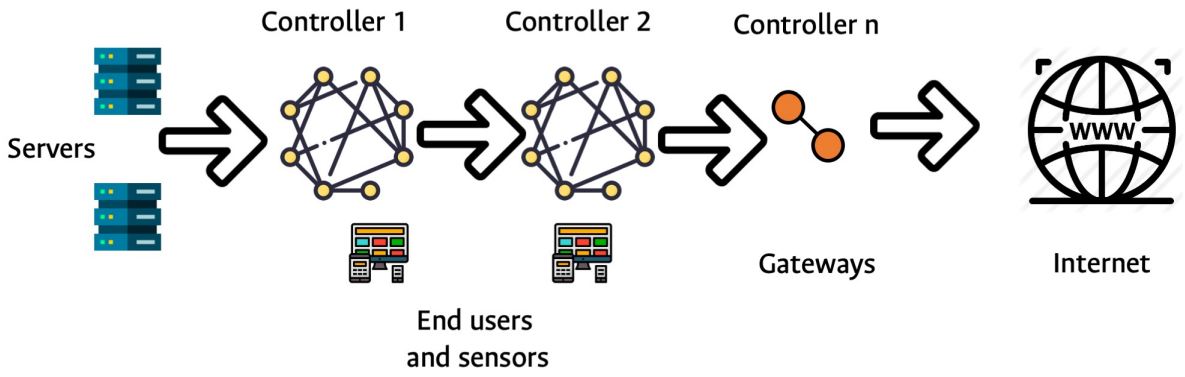


Figura 3-13.: Arquitectura escalable de una SDN. Elaboración propia con base en Wu y cols. (2015); M. Yang y cols. (2015).

de decisiones de reenvío de paquetes hacia diferentes puertos, y un dominio de reenvío donde ocurre la conmutación de los paquetes con base en las tablas de enrutamiento. Como ya no existe un dominio de control residiendo en los nodos OpenFlow, el dominio de reenvío puede mantenerse simple con una tabla con un conjunto de reglas y acciones correspondientes para el procesamiento de los flujos de tráfico, como se observa en la Figura **3-14** (F. Yang y cols., 2014).

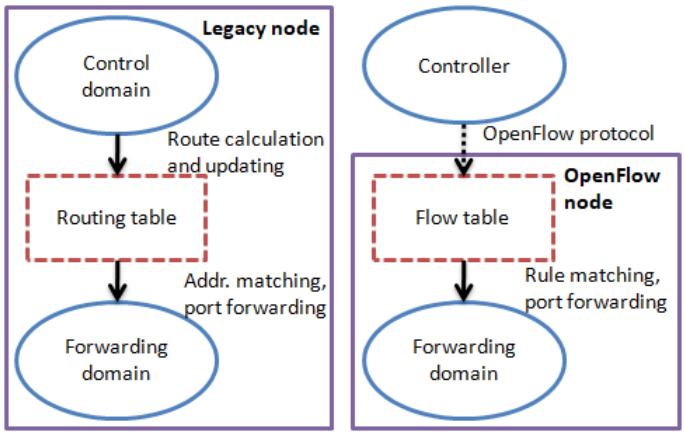


Figura 3-14.: Comparativo entre un nodo tradicional y un nodo habilitado con OpenFlow. Elaboración propia con base en F. Yang y cols. (2014).

Un nodo OpenFlow debe establecer y mantener una conexión de red hacia su controlador mediante *Transmission Control Protocol* (TCP). Existen dos formas básicas de categorizar la red que esta conexión atraviesa: *in-band* y *out-of-band*. El control in-band se refiere a la red en la que el camino hacia el controlador solapa la red de datos. Su principio fundamental

es que el nodo OpenFlow debe reconocer y reenviar el tráfico de control sin involucrar al controlador. El control *out-of-band* es aquel donde la conexión de red hacia el controlador está completamente separada de la red de datos.

El control *in-band* presenta las siguientes ventajas:

- No se requiere un puerto dedicado para el control. Esto es importante en nodos con pocos puertos de red como en los nodos inalámbricos y las plataformas embebidas.
- No se requiere construir y mantener una red separada y dedicada para el control.

Por su parte, el control *out-of-band* presenta las siguientes ventajas:

- Sencillez: Se simplifica ligeramente la implementación del protocolo OpenFlow en el nodo.
- Confiabilidad: Cuando se presenta tráfico de datos excesivo, no se interfiere el tráfico de control.
- Integridad: Los dispositivos que no pertenezcan a la red de control no pueden suplantar a un nodo o al controlador.
- Confidencialidad: Las dispositivos que no pertenezcan a la red de control no pueden espiar el tráfico de control.

4. Virtualización y balanceo de carga

La virtualización puede interpretarse en general como la abstracción y la distribución de los recursos por diferentes partes. Con la virtualización, se puede reducir significativamente el costo de equipos y administración, dado el incremento en la utilización del hardware, el desacoplamiento de funcionalidades asociadas a la infraestructura, la fácil migración a nuevos productos y servicios y su flexibilidad en la administración (Liang y Yu, 2015).

Por otra parte, el balanceo de carga es un problema básico de la optimización de redes en el enrutamiento eficiente del tráfico entre un par de nodos que desean comunicarse (Kleinberg, Rabani, y Tardos, 2001). Allí, como también sucede en otros aspectos de la virtualización, surge la noción de equidad. De hecho, el problema del balanceo de carga equitativo se refiere a la distribución del trabajo de enrutamiento a los nodos participantes. Su objetivo es optimizar las rutas que minimicen la carga de tráfico máxima de la red (Hyytiä y Virtamo, 2007).

Así, dadas estas dos perspectivas, buscaremos identificar las estrategias de virtualización que permitan realizar balanceo de carga para las MANET que soportan el sistema Tlön. SDN, como veremos mas adelante en los trabajos relacionados, ofrece una solución viable a ambos paradigmas.

4.1. Virtualización de redes inalámbricas

Las redes de comunicación inalámbricas se han estudiado desde diferentes perspectivas en su evolución (Qadir, Ahmed, y Ahad, 2014). Las recientes tendencias en propuestas para su arquitectura se resumen en cuatro categorías: Virtualización de Redes Inalámbricas (WNV), Redes Definidas por Software (SDN), Redes Cognitivas (CN), y Virtualización de Redes basadas en la Nube (CNV). Estas tendencias son disciplinas hermanas en el estudio de las redes inalámbricas, las cuales se aproximan desde diferentes perspectivas a la programación de redes, como se ve en la Figura **4-1**.

En el cruce de estas tecnologías, se encuentra la orquestación de redes que en general se considera como un mecanismo de automatización de tareas y de control de los recursos y elementos que soportan un servicio o solución dentro de un sistema distribuido (de Sousa y cols., 2018).

Virtualización

Es el proceso compartimento de recursos de la red a través de la abstracción, partición y aislamiento de las funciones de red del hardware físico (Wen y cols., 2013).

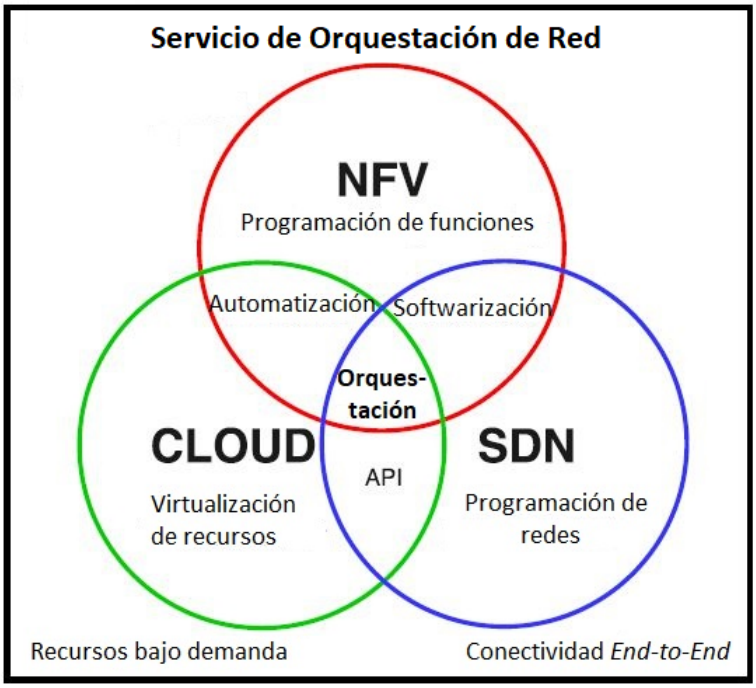


Figura 4-1.: Relaciones entre NFV, SDN y Cloud en un Servicio de Orquestación de Red (NSO). Elaboración propia con base en de Sousa y cols. (2018).

La virtualización en redes inalámbricas puede darse desde diferentes perspectivas como son: basadas en flujos, en protocolos y en el espectro. Una infraestructura totalmente virtualizada está formada por la coexistencia y cooperación de diferentes dominios de virtualización. Estos dominios pueden ser, a pesar de sus diferencias, inter-compatibles y pueden complementarse en sus debilidades (Wen y cols., 2013). Tanto la tecnología de la red inalámbrica como la granularidad deseada de las funciones de red, influyen en la selección de una perspectiva de virtualización. En la Tabla **4-1** se muestra un resumen de estas tres perspectivas de virtualización en redes de tipo inalámbrico.

De la perspectiva *protocol-based*, es destacable el enfoque que se ha venido dando hacia el diseño *cross-layer*, que ha sido ampliamente utilizado para sobrepasar las limitaciones en cuanto a inestabilidad de los canales inalámbricos en las MANET, ya que permite que múltiples capas de red cooperen para cumplir los requisitos de una aplicación, es decir, cada capa de red puede adaptar sus parámetros y comportamientos para cada aplicación (Han,

Shin, y Lee, 2014).

Una de las propuestas de virtualización basada en protocolo implementa un esquema parcial en la capa MAC con "granularidad fina", para soportar el diseño de aplicaciones *cross-layer*, dividiendo la capa MAC en una "máquina de control" y un "ejecutor", y siguiendo las recomendaciones encontradas en la patente de Rajamani y Jeyaseelan (2005) sobre la partición eficiente de las funciones MAC. Allí se resalta la importancia de las tecnologías de virtualización MAC para proveer la controlabilidad y el aislamiento requeridos por una aplicación, o por un agente de una aplicación.

En el ámbito académico, la virtualización inalámbrica de redes se ha enfocado en la experimentación, por lo que ha surgido gran interés en los *testbeds* y en el desarrollo de proyectos con tecnologías de virtualización de redes inalámbricas, las cuales eliminan muchas tareas engorrosas relacionadas con el hardware. En los últimos, años se han realizado investigaciones y desarrollos en estas tecnologías de virtualización de redes inalámbricas, como es el caso de una plataforma para el diseño, evaluación y comparación de protocolos *cross-layer* para redes ad-hoc multi-salto, en el cual se destaca el diseño realizado para la virtualización y sincronización del tiempo en la red (Noubir, Qian, Thapa, y Wang, 2009).

Dominios de virtualización de redes inalámbricas

En la Tabla **4-2**, se pueden observar diferentes aspectos de las tecnologías o arquitecturas de virtualización con sus alcances y dominios, tales como *testbed, host, network y wireless*, de los cuales una arquitectura puede abarcar varios.

Global Environment for Network Innovations (GENI) (Wen y cols., 2013) es una agrupación (federación) de varios testbeds como PlanetLab y ORBIT, con diferentes aplicaciones bajo el mismo marco para proveer una plataforma de desarrollo unificada, flexible, abierta y rica en funciones, la cual está dividida en múltiples *clusters* y planos de control y administración.

La meta-arquitectura de GENI se puede dividir en tres secciones principales: la agregación federativa de componentes virtualizados, la administración de los registros de usuarios (clearinghouse management registries), y las herramientas de experimentación de servicios.

PlanetLab puede ser visto como un sistema operativo distribuido, ejecutándose a través de una red donde los elementos físicos son nodos de computadores. Cada nodo en PlantLab está basado en una virtualización de servidor usando Linux Vservers, donde cada nodo está ejecutando un hipervisor basado en Linux que soporta diferentes Vserver VMs.

Tabla 4-1.: Perspectivas de la virtualización inalámbrica. Elaboración propia con base en Wen y cols. (2013)

Perspectiva de virtualización inalámbrica	Descripción	Ejemplos de tecnologías
Flow-based	Esta inspirado en tecnologías de *Software-defined networking* SDN. Se enfoca en proveer aislamiento, planificación, administración de servicios diferenciados entre *uplink* y *downlink* para diferentes *slices*. Solo con virtualización *flow-based*, todos los *slices* virtuales comparten la misma pila de protocolos inalámbricos.	OpenRoads y vBTS son ejemplos de *overlay flow-base virtualization*. NVS y la virtualización eNodeB, son ejemplos de arquitecturas que usan un *scheduler* modificado para aislar la programación de los recursos de cada *slice*
Protocol-based	Se enfoca en el aislamiento, adaptación y administración de múltiples instancias de protocolos inalámbricos que funcionan en el mismo hardware de radio (desacopla el protocolo inalámbrico del hardware). En la capa MAC puede haber descomposición del protocolo y virtualización del planificador, mientras que en la capa física se puede incluir asignación de recurso de un hardware DSP.	Una virtualización parcial a la NIC IEEE 802.11 basada en PSM y PCF realiza mejoras en la capa MAC de nodos en topología *mesh*. Tecnologías de *Software-defined radio* SDR como SORA y OpenRadio. Una implementación completa de una virtualización basada en protocolo, potencialmente permitiría a diferentes pilas de protocolos operar en el mismo hardware de radio.
Spectrum-based	Busca el mecanismo para compartir el espectro radioeléctrico. Envuelve la abstracción y la asignación dinámica de frecuencias a cada protocolo, mediante técnicas de reorganización de espectro y *radio slicing*. Desacopla completamente el hardware RF de los protocolos, permitiendo a un solo hardware RF, ser usado por múltiples nodos inalámbricos virtuales, o que múltiples hardware de RF sean usados por un solo nodo virtual.	Se puede llevar a la implementación mediante tecnologías de "Radio cognitiva" y técnicas de *dynamic spectrum allocation/access* (DSA). En el dominio digital, *Spectrum Virtual Layer* (SVL), virtualiza el espectro usando técnicas de procesamiento de señal intermedia, re-mapeo de espectro, organización y DSA. En el dominio analógico, Picasso, presenta modificaciones en los circuitos RF para permitir la transmisión *full-duplex* en la misma antena.

En Europa, existe también un conjunto de proyectos agrupados en el proyecto *Future Internet Research and Experimentation* (FIRE) que, de modo similar a GENI, está basado en la federación de diferentes redes de experimentación a lo largo de Europa, incluyendo PANLAB, OneLab, FEDERICA y otras.

Smart Application on Virtual Infrastructure (SAVI), anteriormente *Virtualiced Application*

Networking Infrastructure (VANI), es la contra-parte canadiense de GENI. SAVI fue introducido por la Universidad de Toronto como una plataforma de virtualización orientada a servicios para experimentación en redes. Está dividida en el plano de administración y control (CMP) y el plano de aplicaciones. El CMP está implementado en el lenguaje *Business Processes Execution Language* (BPEL) y realiza las funciones de agendamiento y administración de recursos de alto nivel. El plano de aplicación es simplemente el conjunto de aplicaciones de usuarios ejecutándose en la cima de los recursos de virtualización.

Los tipos de recursos soportados por SAVI son los de procesamiento, los cuales son programables como las *Field Programable Gateway Arrays* (FPGA), por lo que los usuarios deben incluir un módulo de interfaz en un lenguaje de descripción de hardware; además de los recursos de almacenamiento, que son una gran cantidad de servidores ejecutando servicios HTTP para una comunicación directa entre los usuarios y los servidores una vez establecida la relación atreves del CMP; y los recursos de fábrica que incluyen los switches, gateways y bridges.

Para el proyecto AKARI de Japón, la implementación de la capa de virtualización es un proceso difícil que requiere la inmersión en la tecnología particular que está siendo virtualizada. En este proceso, se debe determinar la arquitectura del *framework* y de la capa de control en la cual el hipervisor de virtualización residirá.

También se dice que la dificultad de optimización en la administración de recursos, varía dependiendo de la granularidad del control y del nivel de autonomía requerida. El escalamiento de la meta-arquitectura puede ser limitada por el ancho de banda del sistema.

Las tecnologías como la de *Spectrum Virtualization Layer* (SVL) y la de *Spectrum Slicing* del proyecto Picasso, realizan virtualización de las funciones asociadas a los dispositivos de radio frecuencia RF y de detección/utilización del espectro radioeléctrico, mediante tecnologías como son *Spectrum Slicing* y dispositivos de hardware especializados como las FPGA. En Picasso, se observa cómo esta solución maneja un *stack* virtualizado para cada servicio al cual asocia una banda base, que es procesada mediante un algoritmo de *spectrum slicing* en una FPGA y transmitida mediante un circuito de radio frecuencia RF que realiza una cancelación digital de interferencia en la señal.

Virtualización del eNodeB

El *evolved Node B* (eNodeB) es el elemento de hardware que proporciona los protocolos de radio necesarios para la comunicación inalámbrica con los Dispositivos de Usuario (DU) (Zaki, 2013). En la Figura **4-2**, se puede observar la arquitectura de virtualización del eNodeB con el hipervisor como mediador entre los recursos de hardware físico y los eNodeB virtuales para cada Operador Virtual (OV).

Tabla 4-2.: Dominios abarcados por diferentes tecnologías de virtualización. Elaboración propia con base en Wen y cols. (2013)

Dominio	Específico	GENI	SAVI	Planet Lab	Cloud MAC	Open Flow	eNodeB	CAP WAP	Open Radio	Picasso
Testbed	Meta arquitectura	X	X							
	Aplicación específica	X	X	X						
Host	Infraestructura *Cloud*	//	X	X	X					
	Sistema Operativo	//	X	X						
Network	NIC	//	X	X						
	switching / routing	//	X		X	X				
	Funciones *Flow-based*	//	X		X	X	X	X		
Wireless	Arquitectura general	//	//	X						
	Servicio/Aplicación	//	X		X					
	BS / AP Virtual	//	X		X		X	X		
	Recursos programados	//	//		X		X	X		
	Funcionalidades MAC	//	//		X			X	X	
	Funcionalidades PHY	//	//						X	
	Espectro	//	//							X
	Circuito RF	//	//							X

El eNodeB también es responsable de programar los recursos de tiempos y frecuencias del espectro radioeléctrico entre los diferentes usuarios, garantizándoles calidades de servicio diferenciadas y algunas funcionalidades de administración de la movilidad de los usuarios como las señales de transferencia a otros eNodeB y las mediciones de los enlaces de radio. La arquitectura de virtualización del e investigación en sistemas LTE, ha demostrado las ventajas de llevar a la practica la virtualización red inalámbrica (Zaki y cols., 2010).

4.2. Balanceo de carga

Balanceo de carga

Es la mejor distribución de la *carga de tráfico*[a] de la red entre un número definido de enlaces tal que los eventos de congestión sean minimizados.

[a]La carga de tráfico, en términos generales, es la taza a la que se transmiten los paquetes en la proximidad de un nodo dado.

Generalmente, el balanceo de carga hace referencia a cualquier método para distribuir el procesamiento o peticiones de servicio a través de los dispositivos de una red (Mcheick, Karawash, y Baba, 2011). El problema del balanceo de carga se da cuando llega al sistema de manera aleatoria las peticiones de transferencia de tráfico a una dirección específica en la red.

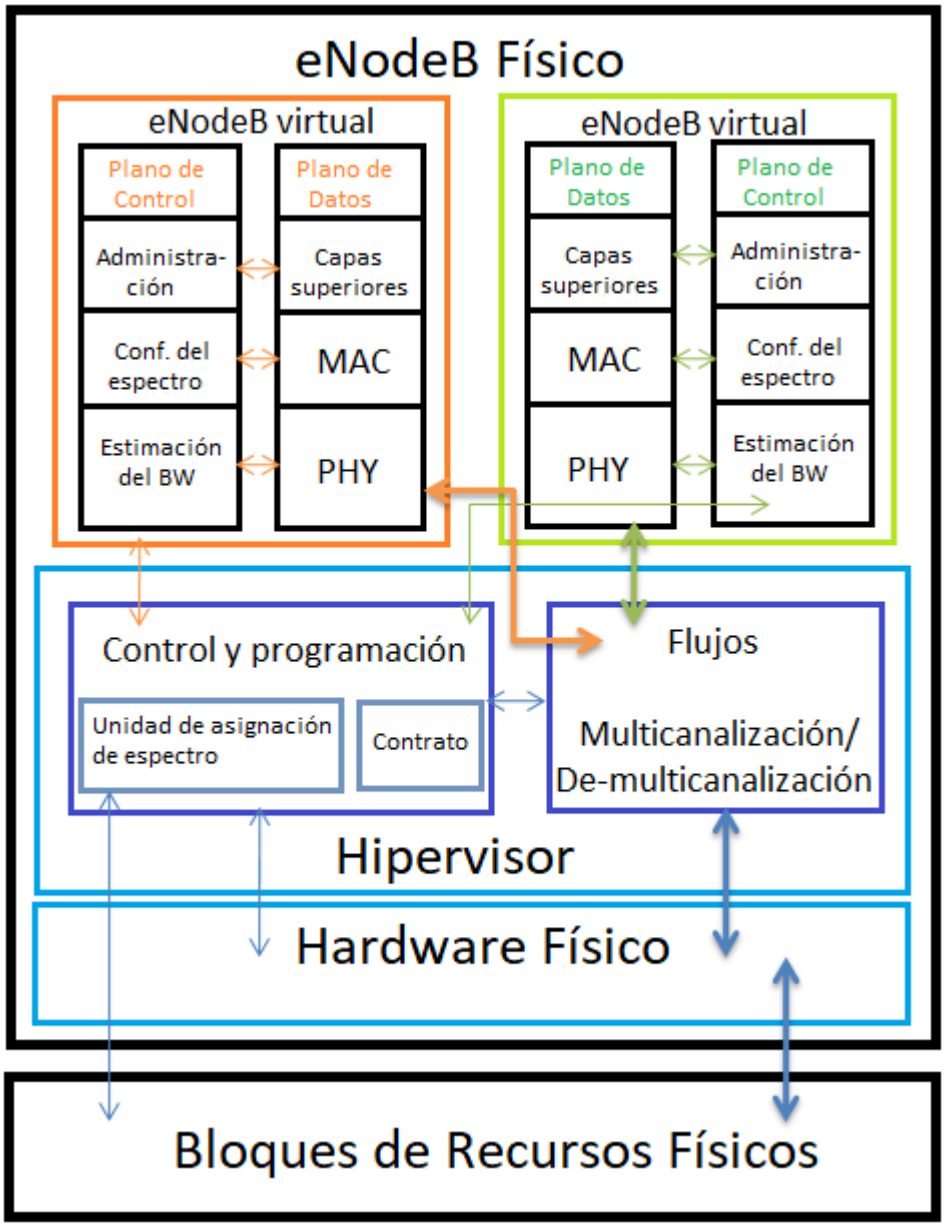


Figura 4-2.: Arquitectura LTE con el eNodeB virtualizado mediante el hypervisor. Elaboración propia con base en Zaki (2013).

Maheshwari y Nedunchezhian (2012) encontraron que las principales investigaciones han abordado el problema de balanceo de carga a través de la estimación de la congestión de red y el control de tráfico. Algunos enfoques han utilizado métricas de energía y potencia para tomar decisiones de enrutamiento para el balanceo de carga, y también existen algunos que se han basado en clustering.

Según Zaki y cols. (2010), la equidad en redes inalámbricas puede definirse en términos del uso y compartición del espectro, la energía, o un producto de ambos, inclusive en términos de la QoS entregada a los usuarios finales. Además, se debe tener en cuenta la interferencia causada al compartir estos recursos en las métricas de desempeño de la red. Por ejemplo, el *delay* depende de los flujos en el enlace propio y en los enlaces adyacentes por la interferencia entre los canales inalámbricos (Hyytiä y Virtamo, 2007; Kleinberg y cols., 2001; Mcheick y cols., 2011; Toumpis y Gitzenis, 2009; Zaki y cols., 2010).

Identificación de las estrategias de virtualización y balanceo de carga utilizadas en redes inalámbricas

Aunque en la literatura se encuentran limitados trabajos en SDN para redes inalámbricas (Arun, Chakraborty, y Manoj, 2014), debido a su reciente aplicación en este tipo de redes, existe un creciente interés por el estudio e integración de estas aproximaciones de redes definidas por software con redes inalámbricas tipo *mesh* y MANET (Haque y Abu-Ghazaleh, 2016). En la Tabla **4-3**, se resumen algunas características de estos trabajos.

Según Pham y Perreau (2003), los nodos situados en el centro de una MANET están mucho más cargados en comparación con los que están posicionados en los límites de la red. Estos autores introdujeron un modelo básico para balancear la carga en una MANET con rutas de múltiples caminos. En este modelo, se consideró una red de N nodos distribuidos uniformemente sobre un disco D de radio R y con densidad δ, donde N está relacionado con la densidad.

Zaman, Khan, y Reddy (2009) presentan un problema crítico de balanceo de carga cuando los nodos de una MANET intentan acceder a Internet usando múltiples gateways. La técnica de control de cluster, provista por el controlador, consiste en seleccionar el gateway con la menor carga.

Toumpis y Gitzenis (2009) presentan un algoritmo de balanceo de carga que, de manera análoga a un circuito eléctrico con la ley de Kirchoff, busca satisfacer la condición de equilibrio en la que se minimiza la función del costo total de los flujos de tráfico en una red de sensores inalámbrica (se maximiza la utilidad de la red). La red es modelada como un grafo dirigido con un conjunto de nodos y de arcos. Cada enlace inalámbrico tiene asociado un costo (que puede corresponder a la energía disipada, el delay, etc.), que depende de los flujos en ese y otros enlaces cercanos debido a la interferencia. Los nodos (fuentes de tráfico hacia la red) son análogos a las fuentes de corriente en un circuito eléctrico y los arcos son análogos a elementos eléctricos que exhiben una caída de voltaje cuando la interferencia los

acopla (interferistores).

Un modelo matemático práctico para el problema del balanceo de carga en una red similar a una MANET fue propuesto por Mcheick y cols. (2011) con la utilización de una matriz de taza del tráfico de llegada T que contiene los flujos transportados entre un conjunto de rutas entre nodos dinámicos.

El primer diseño de red inalámbrica usando OpenFlow se implementó por Dely, Kassler, y Bayer (2011) para brindar una solución de movilidad en un *testbed*, donde un controlador centralizado realiza el proceso de "mover" las reglas del flujo de datos para la estación cliente entre dos puntos de acceso inalámbrico. Esta arquitectura logró desacoplar las funciones de red usando un controlador OpenFlow y creando dos redes inalámbricas virtuales que usan diferentes SSIDs (señalización *out-of-band*): una para el flujo de control y la segunda para la transferencia de datos entre los nodos. Este tipo de "movimiento" habilita casos de uso en redes inalámbricas como el balanceo de carga y la movilidad de los usuarios. En este diseño se balancea la carga en la red al transferir la conexión de una estación cliente con un AP hacia otro AP.

La virtualización del eNodeB busca distribuir los recursos del espectro radioeléctrico a diferentes Operadores Virtuales (OV). En contraste, el balanceo de carga apunta a equilibrar la utilización del espectro total entre múltiples eNodeB pertenecientes al mismo OV. Cada OV ejecuta el balanceo de carga independientemente de los demás OV.(Li y cols., 2012). Se puede brindar una ganancia significativa en el rendimiento de la red descargando los eNodeB virtuales con tráfico excesivo a través del mecanismo de balanceo de carga dinámico. El mecanismo ejecuta periódicamente y transfiere (de ser necesario y posible) a usuarios con baja prioridad y rendimiento de eNodeBs sobrecargados a otros que tengan suficientes recursos de reserva.

Amokrane, Langar, Boutabayz, y Pujolle (2013) propusieron el algoritmo bio-inspirado *Ant Colony Online Flow-based Energy efficient* (AC-OFER) como un protocolo de enrutamiento que tiene en cuenta el costo de la conmutación entre los modos de sueño y actividad de los *Mesh Accecss Points* (MAP) en el consumo de energía de la red. El algoritmo tiene en cuenta parámetros como el número de flujos, enlaces y canales, entre otros. SDN se presenta en este trabajo como una solución de acceso a la parte cableada de la red. Los escenarios evaluados fueron cuatro: el esquema de balanceo de carga, el cual se utilizo para ilustrar el peor escenario de consumo de energía, seguido por los algoritmos *Shortest Path* (SP), *Minimum link Residual Capacity* (MRC) y AC-OFER. Se evidencio que el esquema de balanceo de carga no esta adaptado para la eficiencia energética, dado que si el tráfico esta balanceado entre los AP de la red, se utilizan al máximo todos los AP y por tanto se utiliza la mayor cantidad de energía. También, fue evidente que con el balanceo de carga se aceptan el menor número

de flujos por enlace.

Detti, Pisa, Salsano, y Blefari-Melazzi (2013) mostraron la habilidad de un controlador de red centralizado para establecer arbitrariamente caminos para los flujos de datos y con ello la posibilidad de implementar algoritmos especializados de teletráfico en las redes inalámbricas. Se propuso una arquitectura híbrida que comprendía un módulo OLSR para la configuración *in-band* de la red de control y un controlador SDN centralizado para configurar el plano de datos, demostrando su aplicabilidad en el balanceo de carga entre un par de gateways de Internet con Border Gateway Protocol (BGP) y con una política basada en Round Robin implementada en el controlador mediante el Gateway Selection Algorithm (GSA). Aquí fue necesario el uso tanto del controlador SDN como del módulo OLSR, dado que un solo controlador centralizado introduce un único punto de falla al presentarse eventos como un daño del controlador o fragmentación de la red de control.

F. Yang y cols. (2014) también implementaron un prototipo de red inalámbrica con múltiples caminos entre origen y destino, para mostrar la aplicabilidad y sencillez en el manejo y redirección de flujos de datos que permite un controlador OpenFlow, junto con el protocolo de enrutamiento B.A.T.M.A.N. que provee la topología e información de calidad de los enlaces inalámbricos. En este prototipo se presentaron cuatro componentes típicos en una infraestructura *mesh* inalámbrica: clientes, enrutadores, gateways y servidores remotos. El controlador OpenFlow realizó un balanceo de carga básico redirigiendo los flujos de datos en los dos casos de prueba: entre nodos y entre gateways, demostrando así la mejora en el rendimiento de la red.

El problema de optimización abordado por H. Huang y cols. (2015) es cómo satisfacer preferentemente el flujo de control en redes inalámbricas, ya que una eficiente atención a los mensajes de control permite que los paquetes en el plano de datos fluyan sin congestión. Mediante la simulación de tres algoritmos de asignación de y programación de tráfico: *Fixed-Bands Non-Sharing* (FB-NS), *Non-Fixed-Bands Non-Sharing* (NFB-NS) y *Non-Fixed-Bands Sharing* (NFB-S), se demostró que en general el tráfico de control es más importante que el tráfico de datos en Redes *Mesh* Inalámbricas Definidas por Software (SD-WMN).

Wu y cols. (2015) introdujeron una arquitectura de controladores distribuidos en diferentes particiones geográficas de una red inalámbrica a escala urbana. Se empieza a notar la necesidad de esquemas de red escalables en el plano de control, especialmente en redes donde el creciente número de nodos puede sobrepasar las capacidades de un solo controlador centralizado, o en soluciones que demandan movilidad y una cobertura geográfica de la red de forma descentralizada.

El enfoque inalámbrico de SDN se ha basado en el uso de OpenFlow para interconectar redes

públicas *mesh* virtuales y descargar el procesamiento de los nodos hacia la nube (Hakiri, Sellami, Patil, Berthou, y Gokhale, 2017). De manera similar, las tecnologías de *fog computing* son usadas para ampliar las capacidades de los dispositivos móviles al permitir el análisis en tiempo real y realizar funciones computacionales en el borde de una red. Una tendencia adicional revela que los dispositivos *fog* inalámbricos se usan cada vez más localmente, por ejemplo, para comunicaciones intra-vehiculares y entre sensores, y para el manejo inteligente de energía en los edificios.

H. H. C. Yu, Quer, y Rao (2017) presentan una implementación práctica de una SDN MANET utilizada para el intercambio de tráfico local con tecnologías *device-to-device* (D2D), con el estándar IEEE802.11g, en modo ad-hoc, y con *Independent Basic Service Set* (IBSS). Con el diseño del plano de control en una topología estrella que utiliza un SSID separado, se da una solución al retardo adicional presentado con el enfoque in-band propuesto por Detti y cols. (2013).

An y Lim (2019) implementaron el esquema *In-band Software Defined Wireless Networks* (IB-SDWN) junto con el protocolo OLSR, para reducir la interferencia entre los planos de datos y de control, ajustando las tazas de transmisión de los tráficos datos.

Finalmente, en la Figura **4-3**, podemos observar una clasificación de las estrategias de balanceo de carga utilizadas en redes inalámbricas realizada a partir de los trabajos relacionados en la Tabla **4-3**. En todas estas estrategias nos encontramos algún tipo de control virtualizado, como el de las SDN, y de las cuales seleccionamos la redirección del tráfico para aplicar en esta investigación.

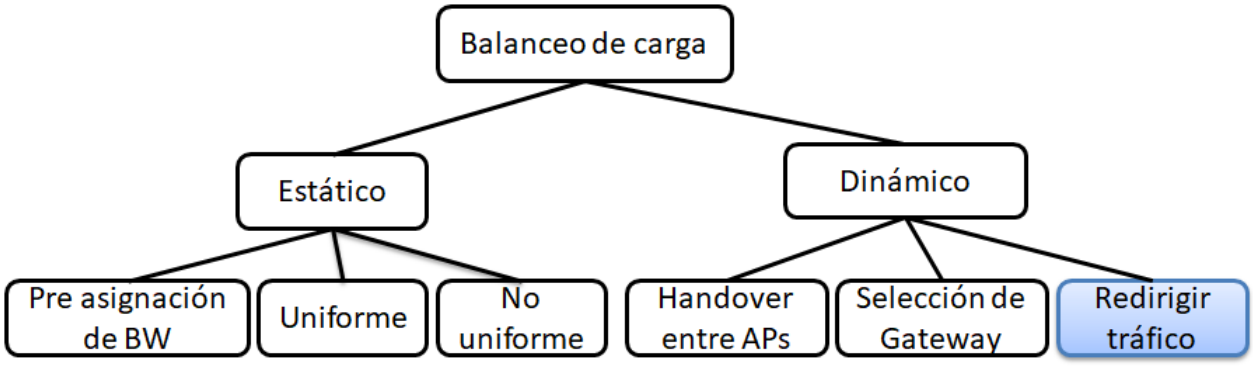


Figura 4-3.: Caracterización de las estrategias de balanceo de carga. Elaboración propia con base en los trabajos relacionados de la Tabla **4-3**.

Tabla 4-3.: Identificación de las estrategias de control virtualizado para realizar balanceo de carga en redes inalámbricas. Elaboración propia con base en Haque y Abu-Ghazaleh (2016); H. Huang y cols. (2015)

Ref	Año	Proyecto	Implem.	Flujo control	Estrategia de balanceo / parámetros
Zhao, Yang, y Liu	2005	Multi-gateway	No	N/A	GSA / Número de nodos
Triviño Cabrera, Casilari, Bartolomé, y Ariza	2006	Mobile gateway	No	N/A	Descubrimiento de múltiples gateways / Ancho de banda
Le-Trung, Engelstad, Skeie, y Taherkordi	2008	Intra/inter MANET	No	N/A	Selección de gateway / Distancia y carga de tráfico
Dely y cols.	2011	KAUMesh	Si	OoB-OLSR	Handovers entre APs
Li y cols.	2012	eNodeB	No	N/A	Dinámico / Total eNodeBs y disponibilidad de sus recursos
Amokrane y cols.	2013	AC-OFER	No	InB-Estático	Eficiencia energética: nodos, BW por flujo, flujos aceptados por enlace
Detti y cols.	2013	wmSDN	Emu	InB-OLSR	GSA con Round Robin
F. Yang y cols.	2014	OpenFlow LB	Si	InB-BATMAN	Redirección de flujos
H. Huang y cols.	2015	Traffic Orch.	No No No	OoB-FB-NS OoB-NFB-NS InB-NFB-S	Algoritmos de asignación de bandas para maximizar el throughput
Wu y cols.	2015	UbiFlow	Si	OoB-Estático	Doble hash / delay, jitter, packet loss y throughput
Akyildiz, Wang, y Lin	2015	SoftAir	Si	InB-Estático	Alternating Direction Method of Multipliers
Y. Wang, Zhang, y Chen	2016	SDNPS	Si	OoB-Estático	Inter cluster - Costo mínimo / Link residual BW
Betzler, Quer, Camps-Mur, Demirkol, y Garcia-Villegas	2016	SDN Const. Edge Devices	Si	InB-Extended OF	Minimiza el máximo (min-max) de la congestión de los enlaces individuales
Hakiri y cols.	2017	SDN Fog network	Emu	InB-OLSR	Redirección de tráfico / BW, SNR
H. H. C. Yu y cols.	2017	D2D SDN MANET	Si	OoB-Estático	Redirección de tráfico / Throughput
An y Lim	2019	IB-SDWN	Si	InB-OLSR	Ajuste de throughput de datos

5. Metodología y diseño

5.1. Metodología

Al tratarse Tlön de un proyecto de investigación aplicada y experimental, como se observa en la Figura **5-1**, debe guiarse de una metodología como *Design Research Methodology* (DRM) introducida por T.M. Blessing y cols. (2009).

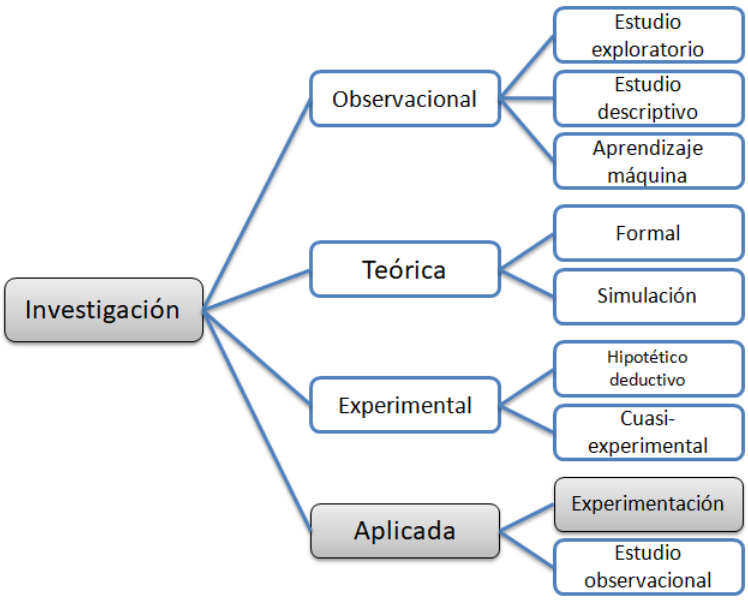


Figura 5-1.: Clasificación de la investigación en Tlön. Elaboración propia.

DRM tiene en cuenta dos componentes de la investigación como son el desarrollo del entendimiento y el desarrollo del soporte, como puede observarse en la Figura **5-2**. Estos dos componentes van de la mano en el cumplimiento de los objetivos del diseño experimental que son:

- Formular y validar los modelos y teorías sobre el fenómeno y el diseño de todas sus facetas (personas, productos, conocimiento/métodos/herramientas, organización, micro y macro economía).

- Desarrollar y validar el soporte fundado en estos modelos y teorías, para mejorar la práctica del diseño, incluir la educación y sus resultados.

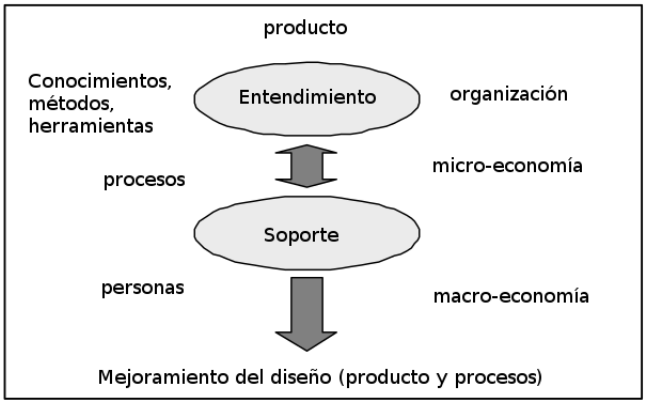


Figura 5-2.: Diseño experimental: fin, objetivos y faces del diseño. Elaboración propia con base en T.M. Blessing y cols. (2009).

Cada una de estas facetas se concentra en una disciplina particular con sus propias metodologías y métodos como en ingeniería, sociología, psicología y ciencias de la computación, entre otras. Así que, se deben emplear otros métodos junto con DRM dependiendo de la pregunta de investigación e hipótesis particulares.

Teniendo en cuenta esto y que el investigador puede no estar familiarizado con los métodos de otras disciplinas, se pueden presentar casos donde la metodología de investigación se escoge sin conocimiento de los paradigmas subyacentes, o no es la más adecuada para el tipo de proyecto, llevando a un uso incorrecto de la metodología.

No solo se busca entender sino mejorar el diseño lo que requiere primero de un entendimiento del modelo o teoría actual de la situación, segundo, una visión clara (modelo o teoría) de la situación deseada, y finalmente, una visión del soporte que probablemente va a cambiar de la situación actual a la situación deseada.

DRM surge como una metodología más rigurosa para abordar la investigación en diseños y para guiar la selección y aplicación de un enfoque adecuado y un método apropiado para diseñar.

El *framework* DRM consiste de cuatro etapas, que son: aclaración de la investigación, estudio descriptivo I, estudio de prescripción y estudio descriptivo II. Cada una de estas etapas forma un proceso con sus medios básicos como entradas y los resultados principales como salidas y tienen las principales características:

- En la etapa de aclaración de la investigación se trata de encontrar evidencia que soporte lo asumido por el investigador para formular un objetivo realista y que valga la pena. Basándose en la búsqueda de literatura se realiza una descripción inicial de la situación existente y de la situación deseada.

- El estudio descriptivo I realiza una descripción detallada de la situación existente para determinar los factores a tener en cuenta para mejorar la aclaración de la investigación.

- En la etapa de estudio de prescripción se utiliza todo el entendimiento de la situación actual para corregir y elaborar una descripción inicial de la situación deseada teniendo en cuenta la aproximación escogida por el investigador para tratar los factores que la mejoran.

- Se procede al estudio descriptivo II para investigar el impacto del soporte diseñado y evaluar su habilidad para llevar a cabo la situación deseada.

- Finalmente, el *framework* DRM está diseñado para ser un proceso iterativo como se muestra en la Figura **5-3**.

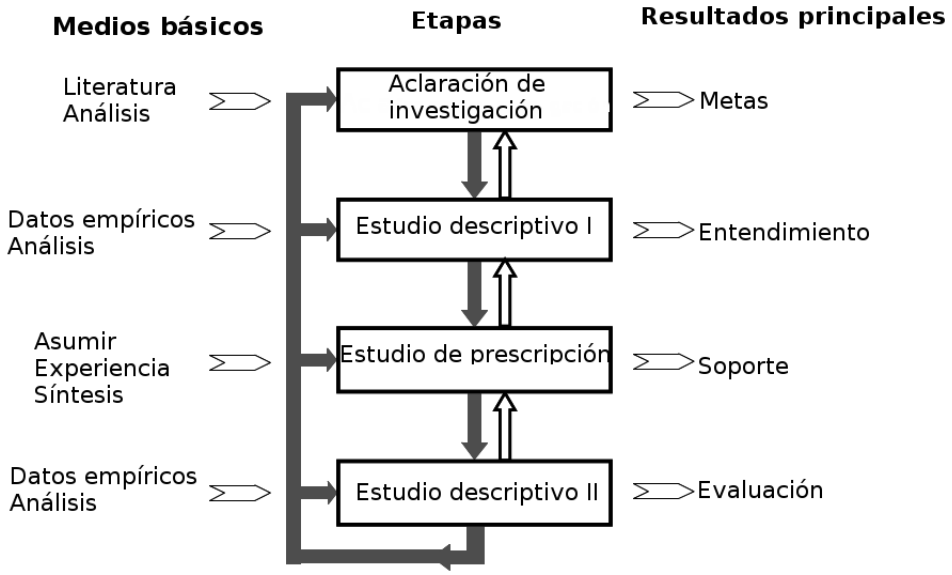


Figura 5-3.: *Framework* DRM. Elaboración propia con base en T.M. Blessing y cols. (2009).

Para el cumplimiento de los objetivos de este proyecto se han seguido las siguientes etapas del *framework* DRM:
El diseño de las pruebas y análisis de los resultados se presentan en el capítulo 6.

Tabla 5-1.: Seguimiento de las etapas del framework DRM para el cumplimiento de los objetivos. Elaboración propia con base en T.M. Blessing y cols. (2009).

No.	Pasos	Capítulos
1	Revisión de la literatura para comprender las metodologías utilizadas en el diseño del control para balanceo de carga.	4
2	Describir los procesos de control, sus características y atributos, comparando los unos con otros y analizando las posibles relaciones entre ellos.	5.2
3	Explicar los paradigmas que pueden brindar soporte al diseño del control virtualizado para balanceo de carga y a la predicción de resultados basado en las teorías asumidos por el investigador.	4
4	Presentar los resultados y su evaluación de una manera analítica.	6.2

5.2. Diseño

El modelo propuesto ofrece la posibilidad de tener un ambiente IoT para monitorear diferentes nodos y aplicaciones, permitiendo el control de las variables del sistema, como son el consumo de energía, distribución de las aplicaciones y de los recursos computacionales. En este contexto, se hace imprescindible contar con un conjunto de herramientas para ejecutar las funciones de red necesarias para satisfacer estas aplicaciones de gestión del sistema.

Este modelo contiene tres artefactos para su operación, que son: orquestador, controlador y agente local con su módulo de balanceo de carga, como se observa en la Figura **5-4**. El orquestador tiene una sobrevista del sistema y la información del estado global y los estados locales de cada nodo. Esta información es enviada desde los nodos hacia el orquestador de forma periódica para mantener un estado actualizado del sistema. El orquestador toma las decisiones de actualizar una política de de QoS y de enviar una aplicación para su ejecución en los nodos de la red.

El agente local actúa como un agente reactivo que ejecuta las políticas de control de recursos en cada nodo de la red. Las aplicaciones se ejecutan dentro de contenedores virtuales, uno para cada aplicación, y el agente local controla la cantidad de recursos que se asignan para cada contenedor y en consecuencia para cada aplicación. El modelo permite que el control de los recursos y monitoreo de las aplicaciones se realice por uno o más agentes locales ejecutándose en un mismo nodo.

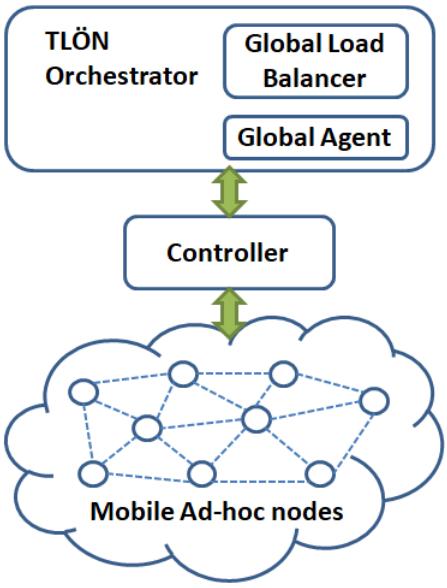


Figura 5-4.: Arquitectura del sistema de balanceo de carga. Elaboración propia.

Las siguientes son las funcionalidades de cada elemento del sistema:

- **Orquestador:** Se encarga de administrar todas las interacciones al interior del sistema con el fin de implementar un sistema multiagente.

- **Controlador:** Se encarga de insertar las reglas de los flujos correspondientes para cada servicio en cada uno de los nodos de la red. Como cada flujo que pasa a través de un nodo tiene su propio socket que esta relacionado con alguna aplicación, el controlador puede aplicar políticas para aplicaciones especificas que vengan desde el orquestador o que sean establecidas directamente por los usuarios del sistema.

- **Agente local:** Se encarga de recibir las políticas del sistema Tlön desde el agente global y ejecutar las acciones apropiadas en la red mediante los módulos especializados en cada función de red, como el balanceador de carga local.

- **Balanceador de carga local:** Se encarga de garantizar el direccionamiento hacia los caminos que recorren los flujos de tráfico por los enlaces inalámbricos con mejor calidad de transmisión.

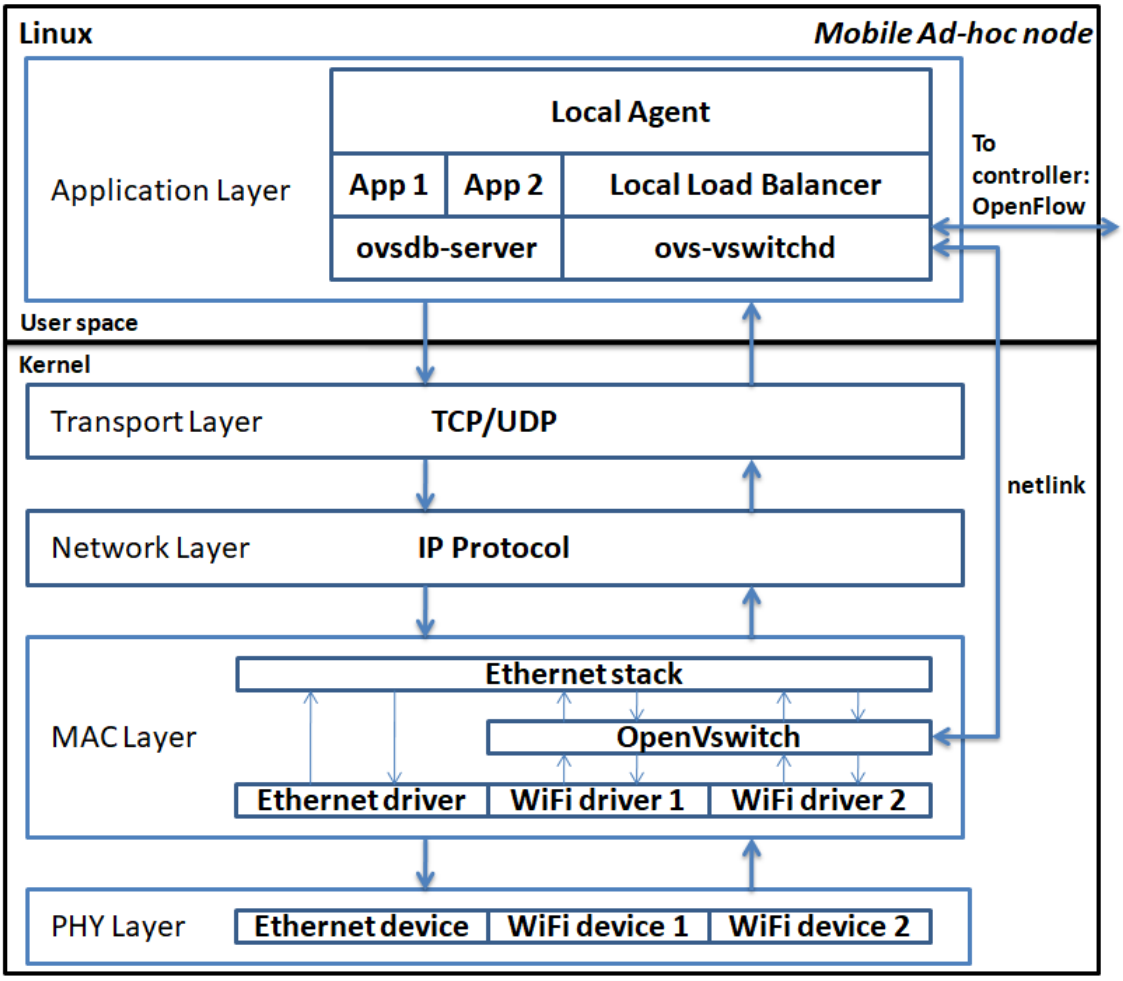


Figura 5-5.: Arquitectura del nodo MANET. Elaboración propia.

Métricas cuantitativas

La experimentación con funcionalidades de balanceo de carga se ha dado principalmente en simulaciones y emulaciones, en las que se puede simplificar la selección de las variables a tener en cuenta. Sin embargo, en la práctica con redes reales debemos tener en cuenta la aplicación para la cual se realiza el diseño, y así mismo, seleccionar la métrica más adecuada. A continuación, se definen algunas variables relevantes en esta investigación:

- Throughput: Las tareas que completaron su ejecución dentro de un periodo de tiempo estipulado.
- Utilización del espectro radioeléctrico: Provee información sobre si el espectro radioeléctrico esta siendo utilizado. Para un balanceo de carga eficiente, la utilización del espectro debe ser maximizada.

Índice de Equidad de Red

R. Jain, Chiu, y WR (1984) definió el *Network Fairnes Index* (NFI) que se utiliza en Tlön para la medición de la equidad en las MANET, dado que este índice se adapta a los sistemas distribuidos donde un conjunto de recursos debe ser asignados a un número de usuarios.

Si un sistema distribuido asigna flujos (ventanas para la transmisión de información entre nodos) a n usuarios, tal que al *i-ésimo* usuario se le asigne el flujo f_i, entonces usaremos la siguiente ecuación para calcular el NFI:

$$f(X) = \frac{\left[\sum_{i=1}^{n} x_i\right]^2}{n \sum_{i=1}^{n} x_i^2} \tag{5-1}$$

donde x_i corresponde al throughput del i-ésimo flujo.

Modelo de colas para asignación de flujos

Desde la Teoría de colas, con los procesos de *nacimiento - muerte*, podemos hacer una analogía con la ley de conservación de la corriente de Kirchoff, donde los flujos en un diagrama de transición de estados son análogos a las corrientes en un circuito eléctrico (Robertazzi, 2000). Los flujos de probabilidades *fluyen* de un estado a otro a lo largo de las transiciones. Numéricamente, los flujos de probabilidades son el producto de las probabilidades de los estados en que se originó la transición y las tasas de transición asociadas a la transición.

La analogía con los circuitos eléctricos es sencilla. Las probabilidades de estado corresponden a las tensiones (o voltajes) de cada nodo y las tasas de transición corresponden a las conductancias (valores inversos a las resistencias). Sin embargo, existen algunas diferencias. No existe una fuente de voltaje o batería en el mundo de las colas. Más bien, se tiene la ecuación de normalización que sostiene que, en un sistema de colas con número finito de estados m, en cualquier instante del tiempo, la suma de las probabilidades de los estados es igual a uno.

$$p_0 + p_1 + p_2 + ... + p_m = 1 \tag{5-2}$$

El concepto más útil que se traslada desde el mundo de los circuitos eléctricos es la idea de observar qué *fluye* a través de la frontera, como se muestra en la Figura **5-6**, se dibujo una frontera alrededor del estado n. Los flujos de probabilidad entrantes y salientes, y la diferencia entre estas dos cantidades es simplemente el cambio en la probabilidad del estado n. Para el estado 0 o inicial solamente se tienen dos flujos: uno hacia y otro desde el estado 1.

$$\frac{dP_n(t)}{dt} = -(\lambda + \mu)P_n(t) + \lambda P_{n-1}(t) + \mu P_{n+1}(t),$$

$$\frac{dP_0(t)}{dt} = -\lambda P_0(t) + \mu P_1(t). \tag{5-3}$$

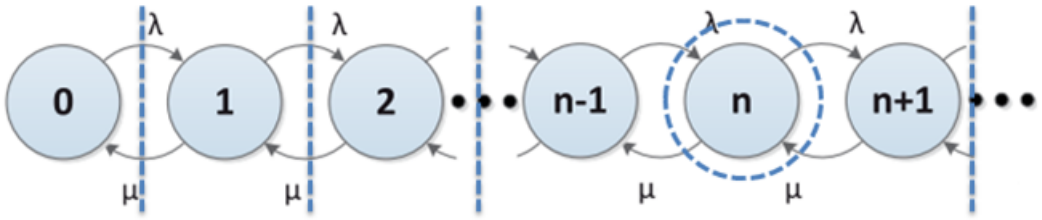


Figura 5-6.: Balance de frontera entre estados y alrededor de un estado n. Elaboración propia con base en Robertazzi (2000).

Como en los circuitos eléctricos, es útil estudiar un sistema de colas (tal como el M/M/1) en un periodo de tiempo limitado con lo que se pueden resolver las ecuaciones diferenciales. Cuando $dP_n(t)/dt = 0$, el sistema de encolamiento esta en equilibrio. Dado que las probabilidades no cambian con el tiempo, se pueden reescribir las ecuaciones como sigue:

$$0 = -(\lambda + \mu)P_n + \lambda P_{n-1} + \mu P_{n+1},$$

$$0 = -\lambda P_0 + \mu P_1. \tag{5-4}$$

Un resultado simple de esta teoría que se mantiene bajo una sorprendente variedad de situaciones es la Ley de Little. Esta relaciona el número promedio de clientes en un sistema de colas, $\bar{n}$; la tasa promedio de llegada, λ; y el tiempo promedio de espera para pasar por el sistema de colas, $\bar{\tau}$:

$$\bar{n} = \lambda\bar{\tau}. \tag{5-5}$$

Hasta el momento, las tazas de llegada y de servicio han sido independientes del número de clientes en el sistema de colas. Muchos sistemas de colas específicos se pueden desarrollar si las tazas de llegada y/o servicio dependen del número de clientes.

Al dibujar fronteras verticales entre cada par de estados adyacentes como las mostradas en la Figura **5-6**, e igualando el flujo de las probabilidades de flujos, resulta una serie de ecuaciones de balanceo local:

$\lambda(0)p_0 = \mu(1)p_1,$
$\lambda(1)p_1 = \mu(2)p_2,$
$\lambda(2)p_2 = \mu(3)p_3,$

$$\lambda(n-1)p_{n-1} = \mu(n)p_n. \tag{5-6}$$

Sustituyendo las recursiones en cada una de las otras, lleva a la siguiente ecuación:

$$p_n = \left[\prod_{i=1}^{n} \frac{\lambda(i-1)}{\mu(i)}\right] p_0. \tag{5-7}$$

Modelo analítico del sistema LTE

Un *scheduler* del sistema de radio LTE distribuye los recursos de frecuencia entre los dispositivos de usuario activos. Existen numerosos parámetros de entrada que son utilizados por el *scheduler* para tomar las decisiones de distribución, por ejemplo, las condiciones del canal, las tasa de transmisión de datos acumuladas y la QoS (Zaki, 2013). Las condiciones del canal pueden cambiar estocásticamente y el programador no tiene control sobre estas.

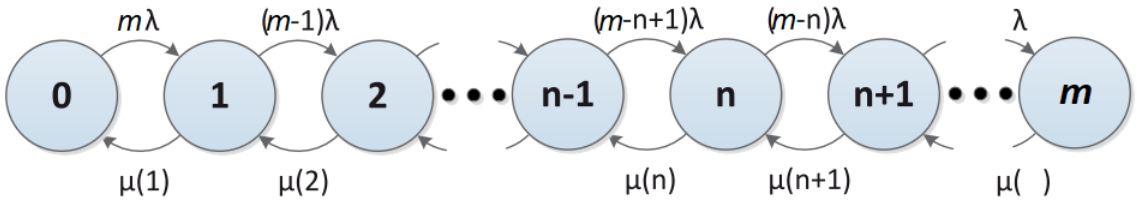


Figura 5-7.: Cadena de Markov de Tiempo Continuo (CMTC). Elaboración propia con base en Zaki (2013).

Este modelo se presenta como una Cadena de Markov de Tiempo Continuo (CMTC) compuesta por $m+1$ estados como se observa en la Figura **5-7**. Cada estado representa el número de usuarios activos del sistema n, tal que, una transición del estado n al $n+1$ significaría que uno de los usuarios inactivos ha terminado su tiempo OFF y está listo para transmitir. Por otra parte, una transición del estado n al $n-1$, significaría que un usuario activo ha terminado su transmisión y pasa a estado inactivo.

Con este modelo de tráfico ON/OFF de duración infinita, se puede modelar un esquema de tráfico de mejor esfuerzo, por ejemplo, la navegación WEB o la descarga de un archivo. La parte ON del modelo representa la descarga del archivo, donde la duración depende de parámetros como la carga, el ancho de banda, el número de usuarios y la calidad del canal. La parte OFF del modelo representa el tiempo de lectura del archivo descargado o el tiempo

en que el usuario lee la pagina WEB antes de solicitar otra.

Al usar el número de usuarios activos n como variable del sistema, no se está especificando cual de los clientes termina su estado ON y abandona el sistema de colas en un instante determinado, como se muestra en la Figura **5-8**. Así, la disciplina de servicio debe ser *First In First Out* (FIFO) o, por ejemplo, *Processor Sharing* (PS). En PS, cada uno de los usuarios activos en la cola recibe μ/n del esfuerzo de servicio. Esto modela como un sistema como el LTE puede atender de las tareas de los usuarios activos de un poquito a la vez por turno. Para calcular las probabilidades de este sistema tenemos

$$\lambda_n = \lambda(m-n),\ 0 \leq n \leq N. \tag{5-8}$$

Las probabilidades de estado se calculan sustituyendo las recursiones como en la ecuación 5-7 y quedan como sigue:

$$p_n = \left[\prod_{i=1}^{n} \frac{\lambda(m-i+1)}{\mu(i)}\right] p_0. \tag{5-9}$$

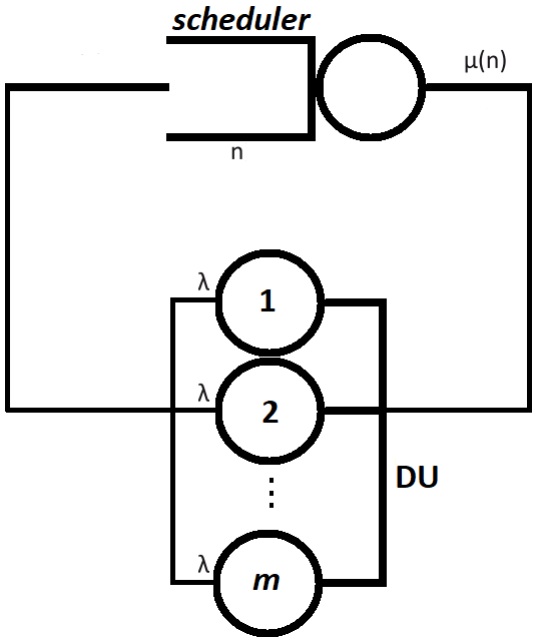


Figura 5-8.: Modelo de colas del sistema LTE. Elaboración propia con base en Robertazzi (2000).

Colas para asignación de canales

En contraste a las redes cableadas donde se da prioridad al retardo, los problemas más críticos que afectan la QoS en redes inalámbricas son la probabilidad de que se bloquee una transmisión y que una transmisión en proceso se termine forzadamente.

Se asume que existen m canales para asignar a cada transmisión. Existen dos tipos de transmisiones que tienen la misma prioridad: originarias y de *handoff*, y se denotan las tazas de llegada promedio λ_o y λ_h, respectivamente. Para $i = 0, 2, ..., m$, sea p_i la probabilidad de que i canales estén ocupados. Todas las llegadas son procesos de Poisson con taza de servicio μ y tiempos de servicio exponencial y estas suposiciones son igualmente aplicables para todas las transmisiones del sistema. El diagrama de transición de estados para este sistema inalámbrico se observa en la Figura **5-9** y su modelo de colas se observa en la Figura **5-10**.

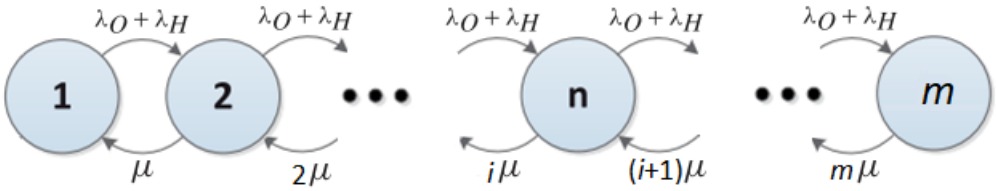


Figura 5-9.: Transición de estados de un sistema con transmisiones de *handoff*. Elaboración propia con base en Barbeau y Kranakis (2007).

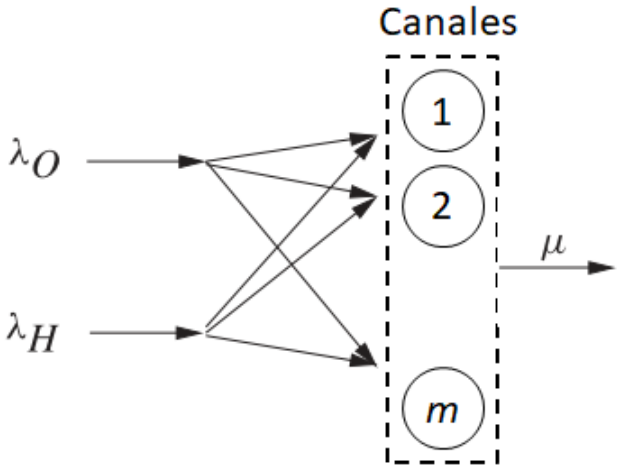


Figura 5-10.: Modelo de colas de un sistema con transmisiones originarias y de *handoff*. Elaboración propia con base en Barbeau y Kranakis (2007).

La taza total de llegadas es $\lambda_o + \lambda_h$ y las ecuaciones de equilibrio son:

$$i\mu p_i = (\lambda_o + \lambda_h)p_{i-1},$$

$$\sum_{i=0}^{m} p_i = 1 \quad \text{para} \quad i = 0, ..., m. \tag{5-10}$$

Usando recursivamente la ecuación 5-10, se obtiene:

$$p_i = \frac{(\lambda_o+\lambda_h)^i}{i!\mu^i} p_0,$$
$$p_0 = \left(\sum_{i=0}^{m} \frac{(\lambda_o+\lambda_h)^i}{i!\mu^i}\right)^{-1},$$

$$B_O = B_H = p_m = \frac{(\lambda_o + \lambda_h)^m}{i!\mu^i} \left(\sum_{i=0}^{m} \frac{(\lambda_o + \lambda_h)^i}{i!\mu^i}\right)^{-1} \tag{5-11}$$

La ecuación 5-11 describe el funcionamiento del sistema con transmisiones originarias y de *handoff* y es conocida como la fórmula de Erlang B.

Especificación mínima del uso del balanceador de carga

A continuación se muestran las especificaciones técnicas para la solución de balanceo de carga propuesta para el sistema que se deben implementar tomando como marco de referencia las mejores prácticas para la prestación de servicios de telecomunicaciones con Acuerdos de Nivel de Servicio (ANS). El balanceo de carga es una aplicación básica para la prestación de estos servicios, por lo que se deben incluir los ANS dentro de la especificación mínima de la solución propuesta para el sistema Tlön.

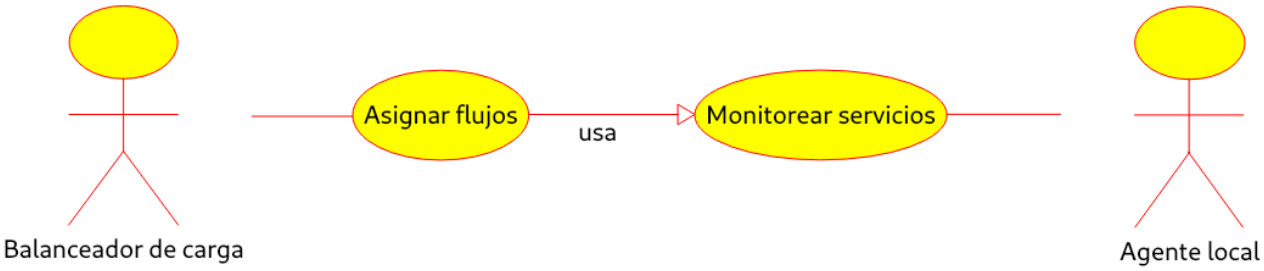


Figura 5-11.: Diagrama de casos de uso del agente local con balanceador de carga. Elaboración propia.

Nombre del servicio: Balanceo de carga simple

Descripción: El servicio permite ejecutar una asignación de los flujos de tráfico para la prestación de servicios de comunicación por diferentes caminos entre nodos de una MANET con el balanceo de carga simple tipo *round robin*.

Actores: Balanceador de carga y agente local.

Flujo principal:

1. En un inicio, el agente local monitorea los recursos/aplicaciones del sistema y recibe las solicitudes de nuevos servicios.
2. El agente local aprovisiona la infraestructura necesaria y recursos de atención para prestar el servicio y actualiza los estados del nodo [**A1**].
3. Según la topología de la MANET, el balanceador de carga inicializa los flujos de tráfico entre los nodos de origen y destino para prestar el servicio.
4. El agente local monitorea y verifica el cumplimiento de los indicadores a través de rutinas de chequeo.
5. El balanceador local inicializa los flujos de tráfico entre los nodos de origen y destino para prestar el servicio.

Flujo alterno:
[**A1**]**:** El agente local aprovisiona la infraestructura y recursos redundantes e independientes para brindar alta disponibilidad, en caso de que el sistema cuente con ellos.

Algoritmo del balanceador de carga y el agente local

El algoritmo 1 muestra como se selecciona un camino de datos óptimo entre dos host conectados a la red *mesh* a través de los nodos N disponibles y los enlaces inalámbricos L que los conectan (Hakiri y cols., 2017).

Algoritmo 1: Balanceo de carga

```
Data: Th, N, L;
Result: Rerouting data flows to the optimal path
initializeDefaultFlowRules(N);
while Listening to LLDP packets do
    if TrafficCongestion(Th) then
        calculateNewOFRules(N, L);
        FloodPackets(N);
        CalculateOptimalPath(source,destination,N, L);
        if isBestPATH then
            InstallNewOFRules(N);
        else
            goto:;
            calculateNewOFRules(N, L);
        end
    else
        monitoring();
    end
end
```

Th: Umbral de congestión de tráfico

N: Nodos disponibles en la red

L: Enlaces inalámbricos entre los nodos

6. Pruebas y resultados

6.1. Pruebas

Para la implementación del módulo de control virtualizado y la validación de su funcionamiento se tuvieron en cuenta tanto los recursos disponibles como los diferentes escenarios de utilidad identificados en otros tipos de redes como las redes comunitarias, redes Small Office/Home Office (SOHO) y otras.

Los nodos se ubicaron en diferentes cuartos de una casa de 3 pisos, como se observa en la Figura **6-1**, donde ningún nodo tiene línea de vista con otro, dada la existencia de placas de concreto entre los pisos y muros de ladrillo que dividen los cuartos.

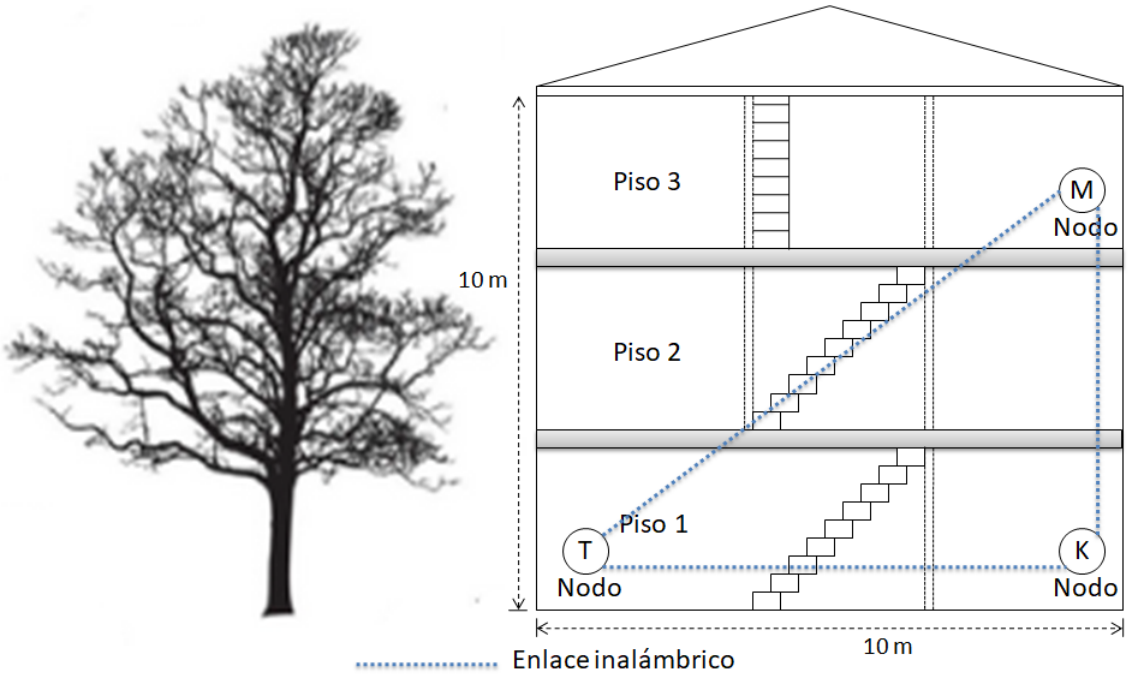


Figura 6-1.: Distribución de los nodos en una casa/oficina de 3 pisos donde se llevaron a cabo las pruebas. Elaboración propia.

Caracterización de los dispositivos

Se diseñaron las pruebas para trabajar con dispositivos embebidos que permitieron crear las redes de control y datos en el modo ad-hoc de la tecnología IEEE 802.11, por ejemplo, las Raspberry Pi modelos Zero W, 2B y 3B (Raspberry Pi Foundation, 2020), o las Odroid modelo XU4 (Hardkernel co Ltd, 2020). Para la captura de paquetes de los canales inalámbricos, se utilizó el adaptador USB de captura de paquetes WiFi AirPcap y el analizador de protocolos de red Wireshark, con el que se obtuvieron las gráficas de las trazas. En la implementación de un escenario real para validar el funcionamiento del sistema control virtualizado, se utilizaron los dispositivos del grupo Tlön, presentados en la Tabla **6-1**.

Tabla 6-1.: Dispositivos de hardware utilizados en las pruebas. Elaboración propia.

No.	Hostname	Función	Marca y modelo	SO	Procesador	SDCard	RAM
1	Admin	Gestión	Dell Vostro 3470	Win. 10	3.60GHz	N/A	4GB
2	Muisca	Nodo	Acer Aspire 4315	Deb. 10	533MHz	N/A	512MB
3	Tikuna	Nodo	Acer Aspire A114	Deb. 10	1.10GHz	N/A	4GB
4	Quimbaya	Nodo/control	Raspberry Pi 3B v1.2	Raspbian	1200MHz	32GB	1GB
5	Kogui	Nodo	Raspberry Pi 3B v1.2	Raspbian	1200MHz	16GB	1GB
6	N/A	WiFi USB	TP-LINK WN721N				
7	N/A	WiFi USB	Encore N150				
8	N/A	Capturador	AirPcap 802.11				
9	N/A	Switch	ZTE ZXV10 W300				

El plano de control utiliza el controlador de red SDN Faucet (Faucet Developers, 2021) que está desarrollado en Python con base en el controlador RYU (RYU project team, 2020) para redes en producción. En la implementación del modelo de virtualización inalámbrica con SDN de este proyecto, se utilizaron los componentes de software presentados en la Tabla **6-2**.

Tabla 6-2.: Componentes de software utilizados en las pruebas. Elaboración propia.

No.	Componente	Nombre
1	Controlador SDN	Faucet 1.1
2	Módulo de kernel de B.A.T.M.A.N.	Batman-adv 2018.3
3	Batctl	Batctl debian-2019.0-1
4	Open Virtual Switch	Ovs-vswitchd
5	Módulo switch virtual	Ovs-vsctl v2.10.1
6	Analizador de protocolo de red	Wireshark 3.2.5
7	Generador de tráfico	IPerf e iPerf3
8	Módulo A.L.F.R.E.D.	Alfred 2018.2

Nodo - Controlador

Los nodos son dispositivos de red que participan activamente en las tareas de enrutamiento y demás funciones propias de un cluster. Estas funciones se realizan sobre un sistema operativo instalado en cada nodo, el cual permite asumir un rol de host/servidor, controlador, gateway, etc. En el rol de controlador, un nodo se conecta a todos los otros nodos de la red y desde allí se toman todas las decisiones de balanceo de carga. Como gateway, un nodo debe realizar las funciones propias en el borde de la red que permitan la comunicación con otras redes.

Escenarios de pruebas

Se diseñaron 2 escenarios de pruebas: Jorge Luis Borges (JLB) y Jean Paul Sartre (JPS). Estos permitieron mostrar el funcionamiento del balanceo de carga en una red con 3 nodos actuando como switch y un nodo actuando como controlador, como se observa en las Figuras **6-2** y **6-3**.

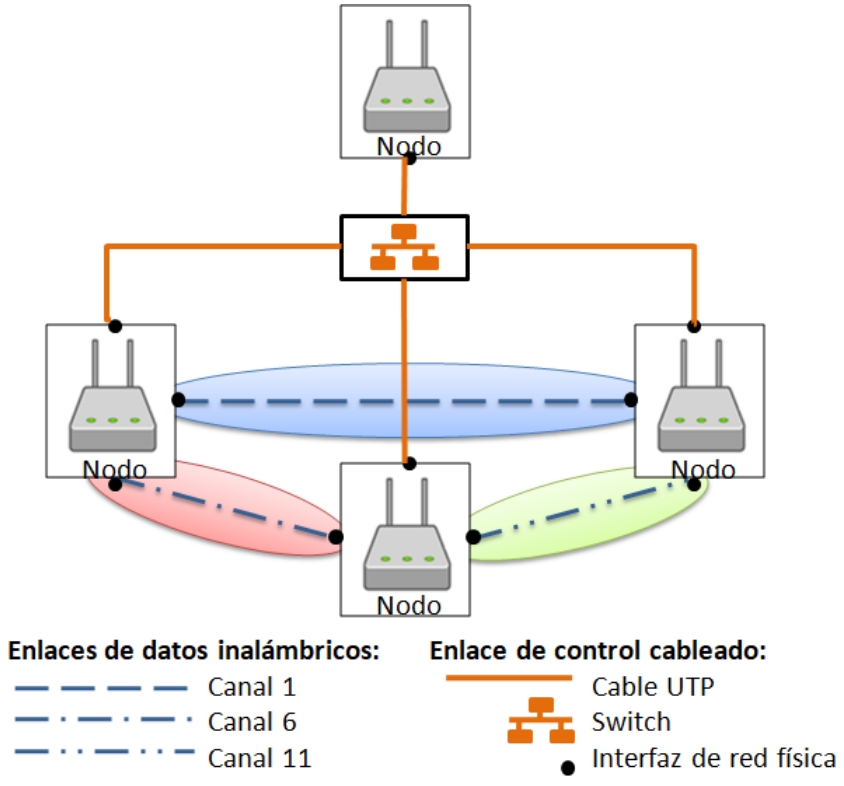


Figura 6-2.: Escenario JLB. Topología con el plano de control conectado con las interfaces de red Ethernet y el plano de datos conectado con 2 interfaces inalámbricas en cada nodo, configuradas para usar los canales 1, 6 y 11. Elaboración propia.

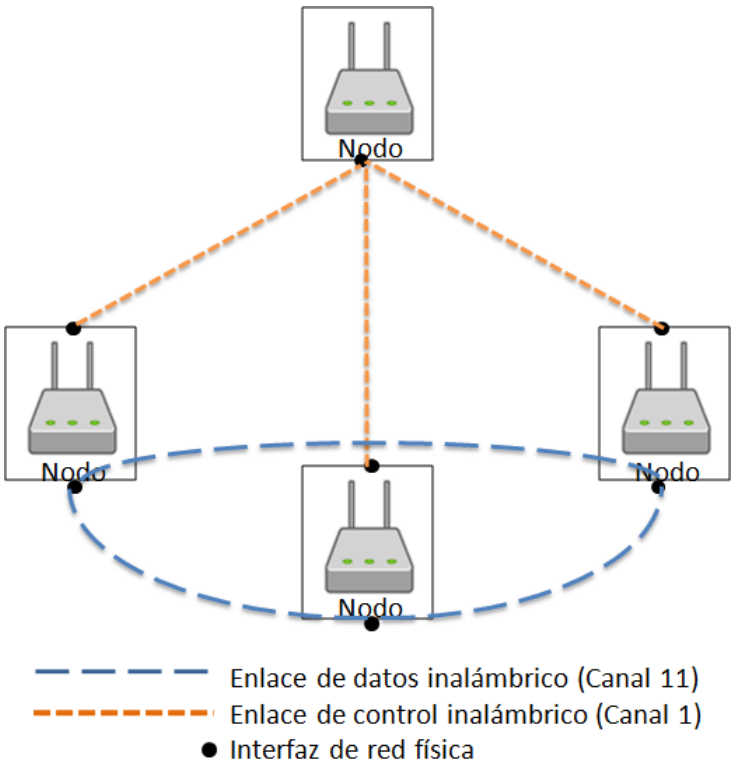


Figura 6-3.: Escenario JPS. Topología con el plano de control conectado con una interfaz inalámbrica en el canal 1 y el plano de datos conectado con una interfaz inalámbrica en el canal 11 configurada en modo mesh. Elaboración propia.

En el escenario JLB los nodos fueron provistos de 2 interfaces de radio para implementar el plano de datos con enlaces IEEE 802.11 operando en 2 frecuencias de la banda de 2.4 GHz (canales 1, 6 y 11, teniendo en cuenta que estos no se solapan en todo su ancho de banda). También, los nodos contaban con una interfaz cableada, la cual permitió establecer la red en bus Ethernet al nodo controlador y realizar la gestión administrativa de los nodos y de las interfaces inalámbricas, conectándose a ellos mediante SSH.

En el escenario JPS los nodos utilizaron únicamente 2 interfaces inalámbricas: una para el plano de control y una para el plano de datos. Vale la pena caracterizar el plano de control dado que este se implementó sobre una red inalámbrica donde se logró el paso de información entre los nodos y el controlador, utilizando una MANET con el protocolo B.A.T.M.A.N. transportando los paquetes de control OpenFlow. Se recopilaron de estadísticas de parámetros característicos del como el TQ de B.A.T.M.A.N., el *throughput* y el Round Trip Time (RTT) entre cada nodo y el controlador. En el plano de datos se logro la transmisión a diferentes tazas de paquetes UDP y de un archivo de gran tamaño con FTP.

El plano de control se configuró en modo *out-of-band* para ambos escenarios así: conectando los nodos con el controlador mediante sus puertos Ethernet y un switch físico en JLB y el otro mediante una interfaz inalámbrica utilizando el canal 1 en JPS. Aunque los nodos tenían la posibilidad de movimiento, esto no afectó en gran medida la calidad de los enlaces inalámbricos, cambiando así la topología establecida.

El plano de datos se configuró con los puertos WiFi operando en modo mesh con IEEE 802.11s y con OVS. Estos puertos están configurados en modo troncal con las VLAN 100 y 200. Aunque el etiquetado de paquetes con una VLAN en el medio inalámbrico no es útil para implementar una separación física de los tráficos, OVS las utiliza para realizar una separación lógica y un manejo más eficiente de los flujos. La separación física del tráfico se puede lograr disponiendo de otra interfaz inalámbrica que opere en otra banda de frecuencias.

Este puerto está configurado en modo acceso, donde los paquetes que ingresan a este puerto no tienen etiquetada una VLAN. Sin embargo, el módulo OVS adiciona una VLAN a todos los paquetes que ingresan por un puerto en modo acceso para permitir el procesamiento de reenvío de paquetes, con base en VLANs en las tablas de flujos.

Así mismo, este puerto está configurado para operar como interfaz física del puerto bat0 con características como el modo ad-hoc, la *Maximum Transmission Unit* (MTU) de 1500 bytes y la autoconfiguración de su direccionamiento IP utilizando el protocolo Avahi.

Solución de problemas de redes inalámbricas

Antes de comenzar las pruebas, debemos solucionar 3 problemas comunes a las redes inalámbricas: problema del nodo oculto, interferencia co-canal y loops.

- **Interferencia co-canal:** Cuando se establece un flujo que atraviesa varios nodos, se deben asignar canales a los enlaces de manera que no se solapen en todo su ancho de banda entre un salto y el siguiente. Así por ejemplo, se podrían asignar los canales 1 y 6 (2,412GHz y 2,437GHz, respectivamente), o los canales 2 y 7 (2,417GHz y 2,442GHz, respectivamente) y así sucesivamente, con lo cual se soluciona el problema de interferencia co-canal. En JPS tampoco se dará la interferencia entre los planos de control y de datos porque estos utilizan canales diferentes.
- **Nodo oculto:** Con esta configuración que usa 2 canales inalámbricos diferentes, se soluciona el problema del nodo oculto, al menos entre estos tres nodos. Por esto, se podría prescindir de los mensajes RTS/CTS en la configuración de las interfaces WiFi; pero, dada la ocupación de estos canales por otras redes vecinas que no tienen relación con este experimento, se mantuvieron los mensajes RTS/CTS.

- **Loops:** Finalmente, el problema de la formación de loops se soluciona aplicando en cada nodo *Spanning Tree Protocol* (STP).

En la Figura **6-4**, se observa la ocupación de los canales 1, 6 y 11 por los enlaces Tlön, Uqbar y Orbis, con potencias de las señales en promedio de -28dBm, -32dBm y -25dBm. También, se puede observar que mientras los canales 1 y 6 presentaron un constante tráfico con señal constante y débil de otras redes en la banda ISM no licenciada en el espacio de pruebas, el canal 11 estuvo libre durante la mayor parte del tiempo de pruebas.

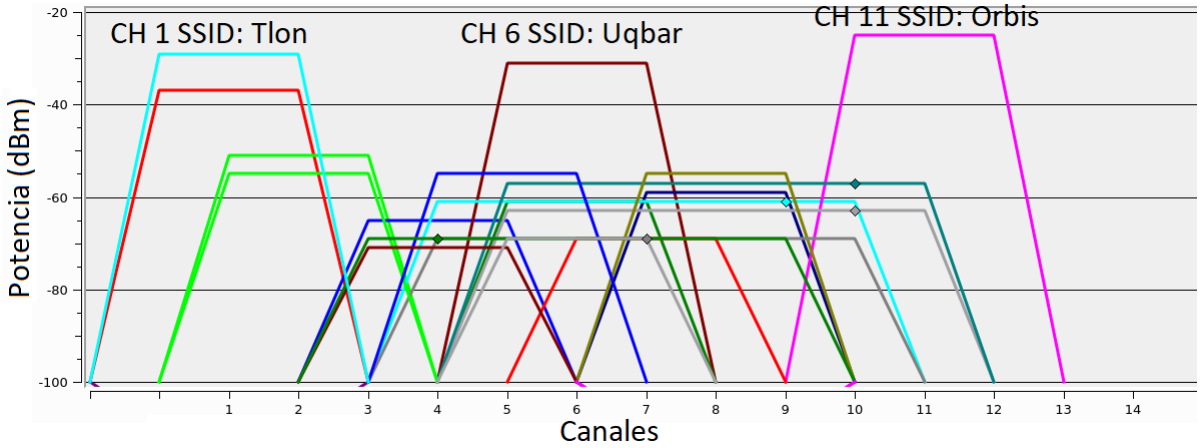


Figura 6-4.: Ocupación de los canales IEEE 802.11 en el espacio de pruebas y asignación de los canales 1, 6 y 11 a los enlaces inalámbricos del plano de datos. Imagen generada con el software gráfico de escaneo inalámbrico LinSSID de Linux.

Pruebas con B.A.T.M.A.N. y OVS

Se desplegó el escenario de pruebas del plano de datos utilizando los mismos nodos con sus mismos puertos para realizar el balanceo de carga mediante la redirección de flujos, tanto con B.A.T.M.A.N. como con OVS. En la Tabla **6-3**, se observa la asignación de los puertos de cada nodo a los canales.

Tabla 6-3.: Asignación de canales y puertos de los nodos. Elaboración propia.

SSID	Canal	AP	Nodo	Puerto	Nodo	Puerto
Tlön	1	02:12:34:56:78:9A	Tikuna	Port 1	Muisca	Port 2
Uqbar	6	02:12:34:56:78:9B	Muisca	Port 1	Kogui	Port 2
Orbis	11	02:12:34:56:78:9C	Kogui	Port 1	Tikuna	Port 2

Se realizó la transmisión del video entre los nodos Muisca y Tikuna. Los caminos por donde puede transitar este tráfico son: el enlace directo entre Muisca y Tikuna mediante el Flujo 1

y el enlace alterno entre Muisca Y Tikuna pasando por Kogui mediante los Flujos 2 y 3.

En la Figura **6-5**, se observa que al bajar administrativamente el puerto del enlace Tlön en Tikuna, se cae el Flujo 1 y el Bridge br0 en el nodo Muisca continúa sacando el tráfico por el puerto del enlace Uqbar (Flujo 2) hacia Kogui. Este a su vez lo retransmite hacia Tikuna por el enlace Orbis (Flujo 3). Al volver a subir administrativamente el puerto del enlace Tlön en Tikuna, se restablece la transmisión por el Flujo 1.

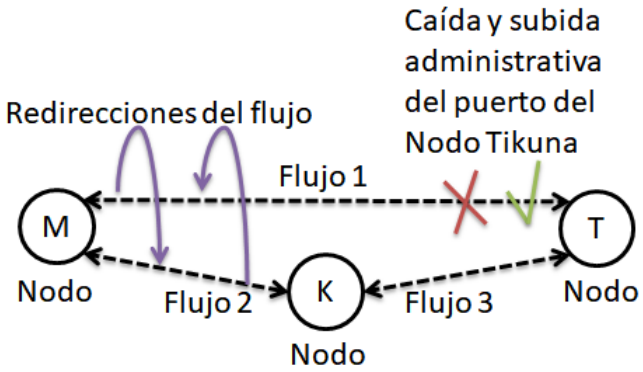
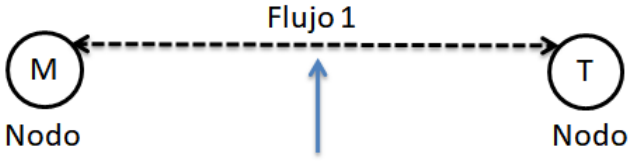


Figura 6-5.: La redirección entre los Flujos 1 y 2 ocurre al bajar o subir administrativamente el puerto inalámbrico asociado al Flujo 1 en el nodo Tikuna. Elaboración propia.

En la Figura **6-6**, se observa que el tráfico del Flujo 1 tiene como fuente y destino las direcciones MAC e IP de los host asociados a Muisca y Tikuna. En la Tabla **6-4**, se presentan las reglas en Muisca y Tikuna asociadas a la acción de inserción del Flujo 1.



	Fuente	Destino
IP	Nodo M	Nodo T
MAC	Nodo M	Nodo T

Figura 6-6.: Establecimiento de la transmisión de un flujo de video entre Muisca y Tikuna. Elaboración propia.

Tabla 6-4.: Reglas y acciones para el establecimiento del flujo 1. Elaboración propia.

Reglas del flujo	Nodo: Puerto de entrada:	**Muisca**	**Tikuna** 1
Acciones: **Insertar Flujo 1**	IP destino: MAC destino: Puerto de salida:	IP Tikuna MAC Tikuna 2	IP Muisca MAC Muisca 1

En la Figura **6-7**, se observa que el tráfico de video fue redirigido hacia los Flujos 2 y 3, manteniendo las direcciones IP de origen y destino y modificando la dirección MAC al pasar por Muisca - Kogui - Tikuna. En la Tabla **6-5**, se presentan las reglas en Muisca, Kogui y Tikuna asociadas a las acciones de remover el Flujo 1 e insertar los Flujos 2 y 3.

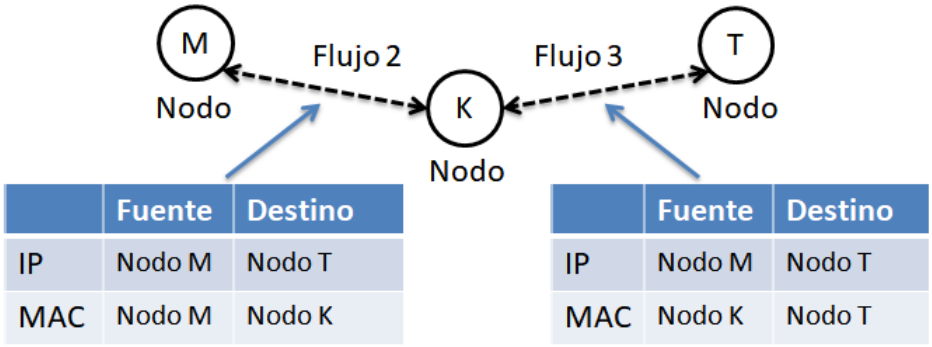


Figura 6-7.: Redirección del flujo de video con la modificación de la cabecera MAC de los paquetes generados desde Muisca y retransmitidos por Kogui hacia Tikuna. Elaboración propia.

Procedimiento de recuperación de la dirección MAC origen

Los nodos en la MANET se comportan como un switch de capa 2 del modelo OSI. Por tanto, las direcciones IP de los *hosts* origen y destino no debe variar al pasar por los diferentes flujos entre ellos.

Por otra parte, las direcciones MAC de origen y destino en la cabecera de los paquetes se debe ir modificando a medida que estos transcurren por los nodos así: host origen - nodo origen - nodo intermedio - nodo destino - host destino. Para imitar el funcionamiento de un switch de capa 2 del modelo OSI se debe realizar un procedimiento llamado *recuperación de la MAC*, que consiste en reinsertar la dirección MAC de origen aprendida por el controlador del host origen a los paquetes que llegan al nodo destino y finalmente hacia el host destino.

Tabla 6-5.: Reglas y acciones para la redirección entre los flujos. Elaboración propia.

Reglas del flujo	Nodo:	**Muisca**	**Kogui**	**Tikuna**
	Puerto de entrada:		2	2
Acciones:	IP destino:	IP Tikuna		IP Muisca
Remover Flujo 1	MAC destino:	MAC Tikuna		MAC Muisca
	Puerto de salida:	2		1
Acciones:	IP destino:	IP Tikuna	IP Tikuna	
Insertar Flujo 2	MAC destino:	MAC Kogui	MAC Muisca	
	Puerto de salida:	1	2	
Acciones:	IP destino:		IP Tikuna	IP Muisca
Insertar Flujo 3	MAC destino:		MAC Tikuna	MAC Kogui
	Puerto de salida:		1	2

6.2. Resultados

A continuación, se presentan los resultados de las pruebas de balanceo de carga en los escenarios JLB y JPS y de la medición de las características de las red en los planos de control y de datos, como el troughput y el RTT.

Resultados JLB

La MANET con el protocolo B.A.T.M.A.N., presentó un buen rendimiento en la transmisión en streaming del video de prueba tanto por el enlace directo entre Muisca y Tikuna como por el enlace alterno Muisca - Kogui - Tikuna. Al darse la redirección del tráfico hacia el enlace alterno, se presentó una perdida de los paquetes recibidos por Tikuna de aproximadamente 7 segundos como se observa a los 129 y 68 segundos en las Figuras **6-8** y **6-9**, respectivamente. En la Figura **6-9**, se presenta una redirección de vuelta hacia el enlace directo entre Muisca y Tikuna a los 210 segundos sin que se pierdan paquetes, dado que este enlace es más eficiente al ser de un solo salto.

En la Figura **6-10**, se observa como al operar los nodos con OVS se presentó una pérdida de los paquetes recibidos por Tikuna de aproximadamente 14 segundos, tanto al darse la redirección hacia los enlaces alternos (Uqbar y Orbis) a los 84 y 238 segundos, como al retornar al enlace directo (Tlön) a los 154 segundos. A pesar del congelamiento de la imagen recibida en Tikuna al redirigir el tráfico, el rendimiento de la red con OVS fue apropiado para la transmisión de un video.

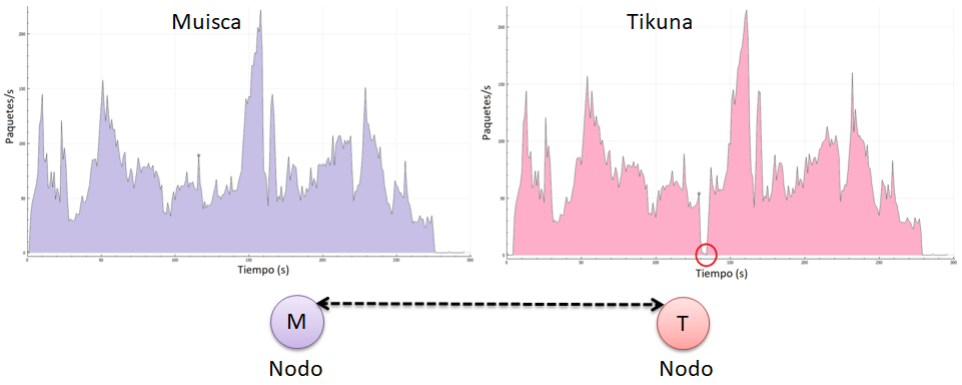


Figura 6-8.: Tráfico de video con el protocolo B.A.T.M.A.N. entre los nodos Muisca y Tikuna con una redirección hacia el nodo Kogui a los 129 segundos, cuando se observa una caída en el tráfico recibido por Tikuna. Elaboración propia.

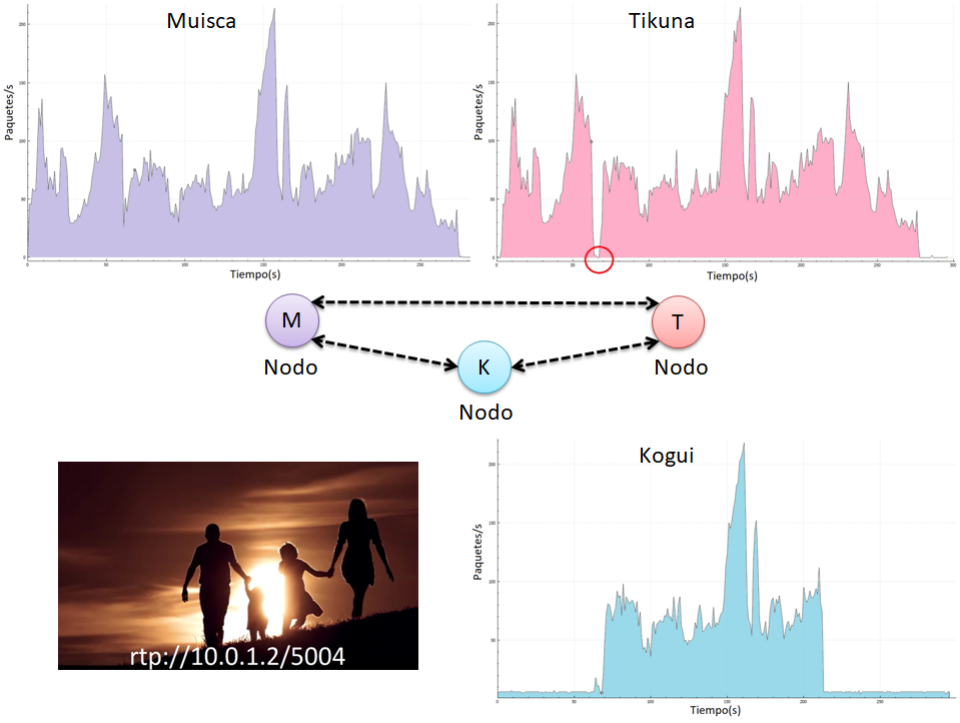


Figura 6-9.: Tráfico de video con el protocolo B.A.T.M.A.N. entre los nodos Muisca y Tikuna con 2 redirecciones: una hacia el nodo Kogui a los 68 segundos, cuando se observa una caída en el tráfico recibido por Tikuna, y otra de vuelta al enlace inicial directo con Tikuna a los 210 segundos. Elaboración propia.

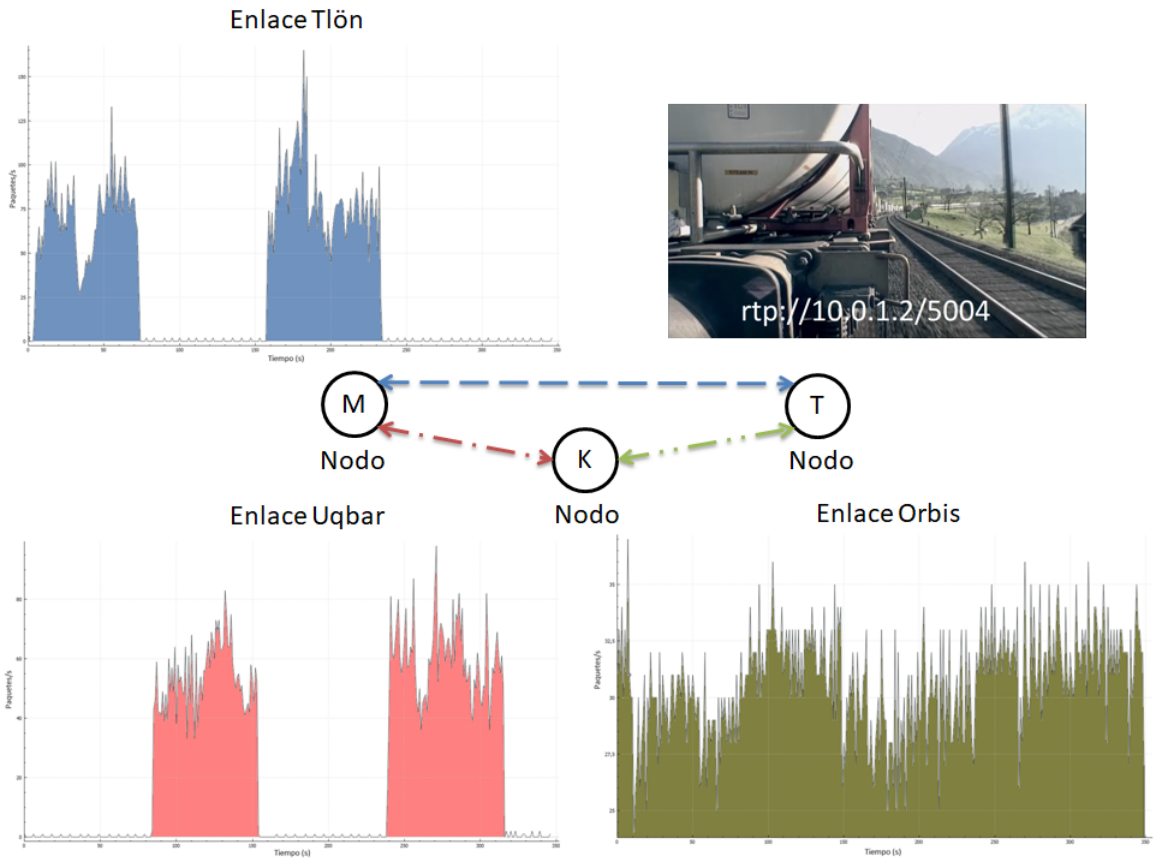


Figura 6-10.: Tráfico de video con OVS entre Muisca y Tikuna con 3 redirecciones entre los enlaces: de Tlön hacia Uqbar y Orbis a los 84 y 238 segundos y de Uqbar y Orbis hacia Tlön a los 154 segundos. El tráfico de Orbis (canal 11) se capturo desde un laptop con el AirPcap cerca a Tikuna. Elaboración propia.

JLB es más equitativo con B.A.T.M.A.N. que con OVS

Complementario a las anteriores verificaciones de la transmición de video, calculamos el NFI utilizando la ecuación 5-1 y los throughput medidos con iperf en la red tanto con B.A.T.M.A.N., como con OVS:

NFI con B.A.T.M.A.N.:
$x1 = 9{,}125 Mbps$ Muisca - Tikuna
$x2 = 2{,}308 Mbps$ Muisca-Kogui-Tikuna
$NFI = 0{,}738$ –> 73.8 % equitativo

NFI con OVS:
$x1 = 15{,}98 Mbps$ Muisca - Tikuna
$x2 = 3{,}029 Mbps$ Muisca-Kogui-Tikuna
$NFI = 0{,}683$ –> 68.3 % equitativo

Resultados JPS

En el plano de control la conexión entre el controlador y los nodos con el protocolo B.A.T.M.A.N. fue de un solo salto. En otras palabras, no se requirió que un nodo reenviara mensajes de control desde y hacia otro nodo de la MANET. A continuación, se observa el TQ de cada enlace desde el controlador hacia los nodos:

```
Originator      (#/255)      Nexthop
Nodo T             213       Nodo T
Nodo M             238       Nodo M
Nodo K             255       Nodo K
```

Se capturaron los paquetes generados con iPerf desde el controlador hacia los nodos, en intervalos de 10 segundos, con el adaptador WiFi AirPcap ubicado en la mitad del espacio de pruebas. En la Figura **6-11**, se observa el resultado de la captura de los paquetes TCP, BATADV y los paquetes TCP que se detectaron en Wireshark con errores de transmisión Ethernet, el cual utiliza el algoritmo Cyclic Redundancy Check (CRC). En la Figura **6-12**, se observa la medición del *throughput* de cada uno de estos enlaces, siendo el mayor el del nodo más cercano al controlador (Kogui) con 8.77 Mbits/s.

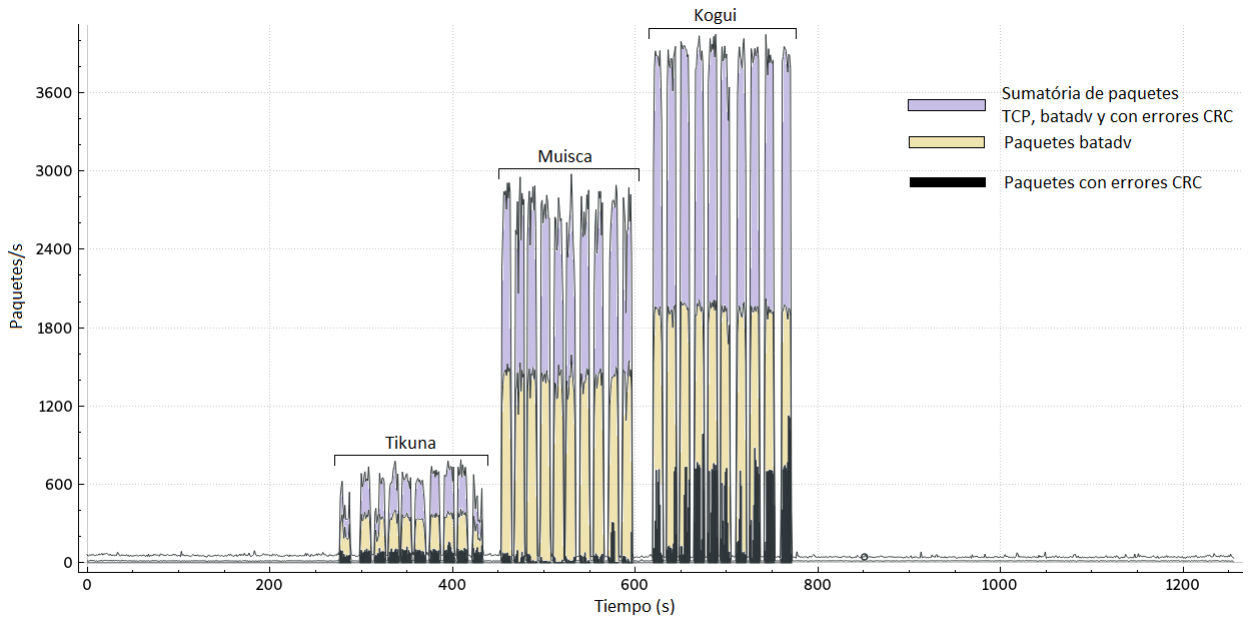


Figura 6-11.: Transmisión de paquetes entre los nodos y el controlador a través de la red *mesh* de control. Elaboración propia.

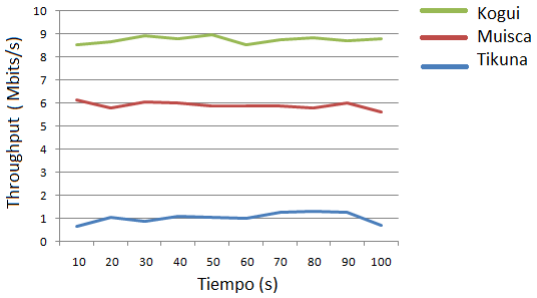


Figura 6-12.: *Throughput* entre los nodos y el controlador. Elaboración propia.

En la Figura **6-13**, se observa la captura de los mensajes iniciales al establecer dirección IP del controlador en cada uno de los nodos. También, se muestran los paquetes TCP que fueron retransmitidos luego de iniciar las comunicaciones controlador/nodos.

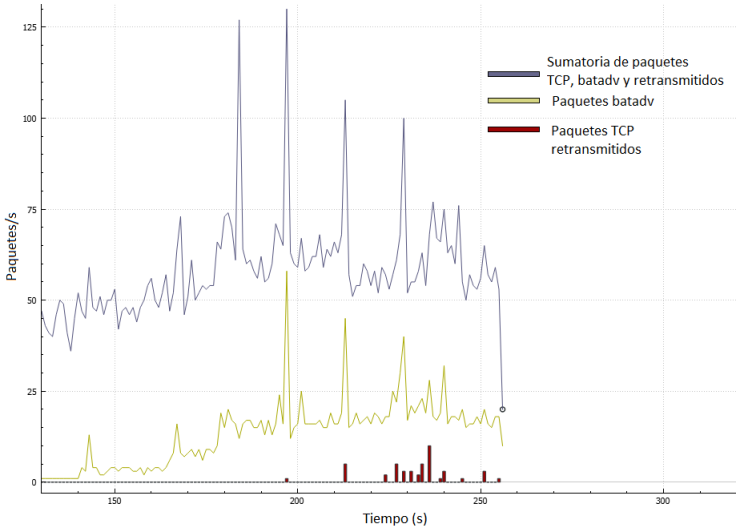


Figura 6-13.: Paquetes TCP de conexión inicial entre los nodos y el controlador mediante el protocolo OpenFlow transportado sobre B.A.T.M.A.N. Elaboración propia.

El plano de control presentó un buen rendimiento siendo implementado sobre una red ad-hoc con un número pequeño de nodos para controlar. Los mensajes de control más recurrentes tienen tamaños que no superan los 200 bytes sumando las cabeceras de los protocolos B.A.T.M.A.N. y OpenFlow. La prueba de ping desde el nodo más alejado del controlador (Muisca), presentó un RTT promedio de 15 ms con desviación de 7.6 ms para tramas con tamaño de 1500 bytes y un RTT promedio de 3ms con desviación de 1.7 ms para trama con tamaño de 15 bytes, como se observa en las Figuras **6-14** y **6-15**. Los mensajes iniciales

de saludo HELLO, capturados desde el controlador hacia el Muisca, tuvieron una latencia apropiada en la red inalámbrica de 5 ms en promedio.

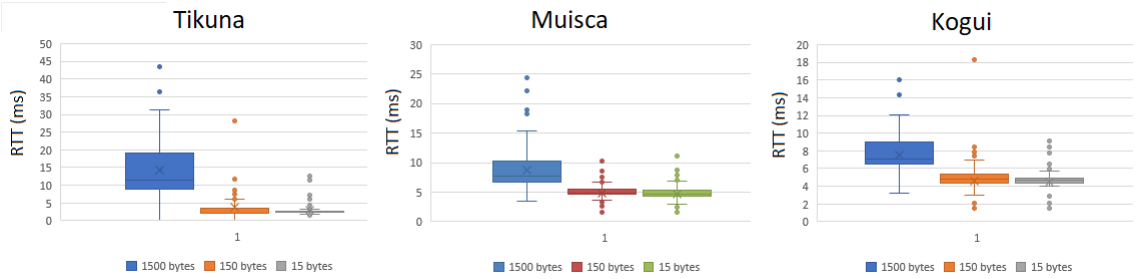


Figura 6-14.: RTT de la prueba de ping entre los nodos y el controlador con tamaños de las tramas de (a) 1500 bytes (b) 150 bytes y (c) 15 bytes. Elaboración propia.

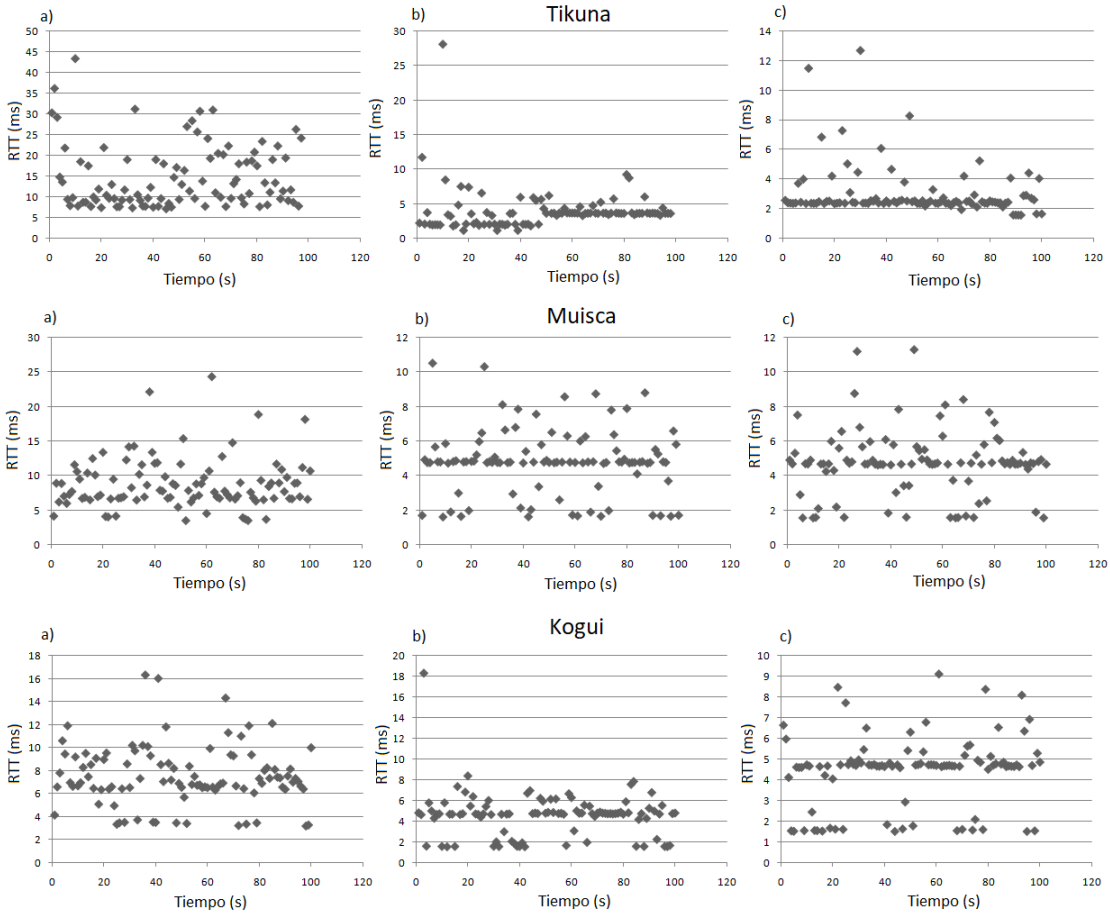


Figura 6-15.: RTT entre los nodos y el controlador con tamaños de las tramas de (a) 1500 bytes (b) 150 bytes y (c) 15 bytes. Elaboración propia.

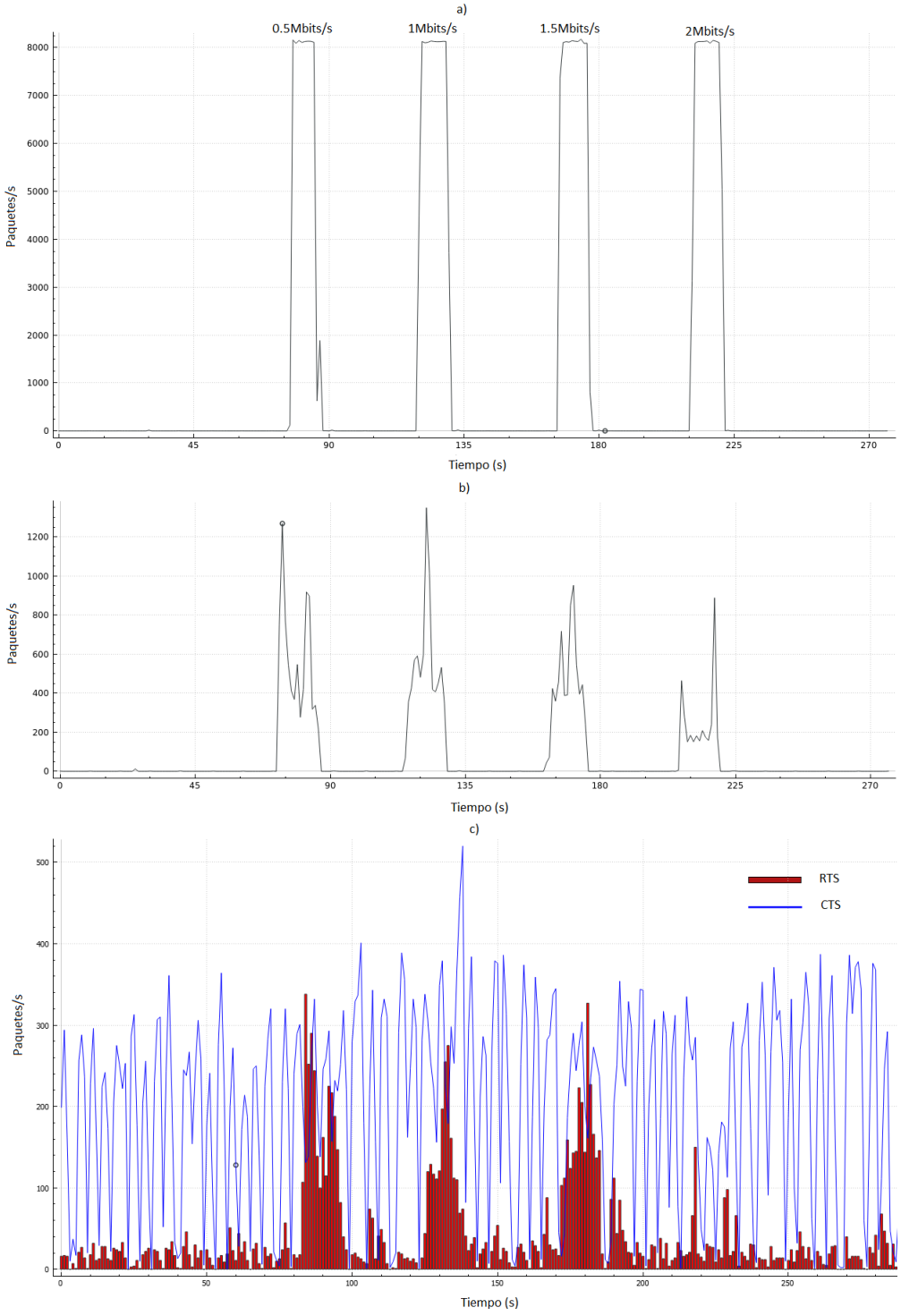


Figura 6-16.: Prueba de transmisión de paquetes UDP en el plano de datos (a) Paquetes transmitidos por Muisca con tazas de 0.5, 1, 1.5 y 2 Mbits/s. b) Paquetes recibidos por Tikuna. c) Paquetes RTS/CTS capturados en el medio inalámbrico. Elaboración propia.

A diferencia del plano de control donde se requiere una comunicación TCP entre los nodos y el controlador, el plano de datos puede transmitir paquetes tanto TCP como UDP que no requieren retransmisiones. En la Figura **6-16**, se observa el resultado de la prueba conexión UDP entre Muisca y Tikuna a diferentes tazas de transmisión (0.5, 1, 1.5 y 2 Mbits/s). La existencia de nodos de otras redes en el mismo canal hace necesario un mayor número de mensajes RTS/CTS durante los intervalos de las pruebas de 10 segundos cada uno, dando como resultado una reducción en el número de paquetes recibidos.

Finalmente, se transfirió un archivo de 663MB mediante FTP de Muisca a Tikuna en 812.67 segundos sin interrupciones, como se observa en la Figura **6-17**.

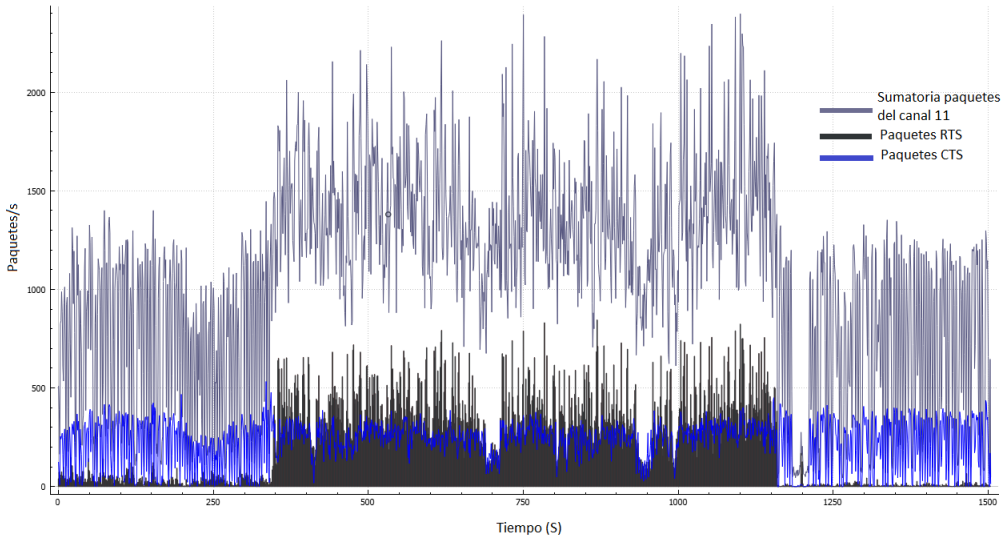


Figura 6-17.: Transferencia de archivo de 663MB en el plano de datos entre Muisca y Tikuna con FTP. Elaboración propia.

Análisis del resultado de la prueba de transferencia

Esta prueba representa el funcionamiento de un servicio básico de transferencia de archivos a través de la MANET. La medición de 812.67 segundos representa la tasa de atención del flujo 1 que se estableció entre Muisca y Tikuna y que permaneció activo y sin modificaciones durante ese tiempo.

Para este análisis debemos tener en cuenta que el tiempo de atención de una transferencia de datos en un medio inalámbrico depende no solo del número de flujos en la cola, sino también de otros fenómenos como lo es la interferencia de otras redes cercanas en el mismo canal. Según el modelo de tráfico ON/OFF de duración infinita, podemos entender que el tiempo de 812.67 segundos corresponde al estado ON del sistema y que hasta el final de este período no se atendieron otros flujos dada la disciplina de servicio: primero en entrar - primero en ser atendido.

7. Conclusiones, recomendaciones y trabajo futuro

Conclusiones

- SDN es una tecnología complementaria en la virtualización de MANET con la que se puede implementar un sistema de cómputo distribuido.
- El módulo de control virtualizado es apropiado para la gestión de los recursos del sistema como el espectro radioeléctrico. Esta virtualización permite la implementación de políticas de balanceo de carga como el desvió de los flujos del centro de una MANET hacia los nodos de la periferia menos congestionados.
- La red de control tipo ad-hoc es apropiada para escenarios que pueden tener cambios en sus topologías e incluir características de reenvío multi-salto de los mensajes de control, movilidad de los nodos y formación de clusters.
- El esquema de control *out-of-band* tiene ventajas de implementación en redes inalámbricas, como la separación de los canales de frecuencias, que permiten experimentar en ambientes con alta interferencia y probar servicios como el balanceo de carga.

  Sin embargo, este tipo de solución requiere que cada nodo participante cuente con dos interfaces de red inalámbricas, lo cual constituye una limitación en los costos de implementación.
- La transmisión de datos en redes inalámbricas está limitada por la tecnología de radio, que en el caso de IEEE 802.11b es de máximo 11Mbits/s, por lo que el balanceo de carga se hace necesario al diseñar aplicaciones que requieran grandes anchos de banda.
- Las implementaciones con B.A.T.M.A.N. son más equitativas que con OVS.

Recomendaciones

- Para la implementación del balanceador de carga es recomendable que la función de cada elemento de la red esté instalada en un dispositivo físico único y no compartido con otras funciones. En otras palabras, los nodos deben corresponder únicamente a los dispositivos incluidos en la MANET, la funcionalidad del controlador no debe estar

instalada en un servidor que atienda otras funciones, etc. Esto se hace con el fin de aislar, identificar y dar solución a los problemas que se puedan presentar en la red.

- Es recomendable desplegar la red en un espacio abierto con baja ocupación de los canales IEEE 802.11 para la realización de las pruebas y captura de paquetes del medio inalámbrico. La calidad de transmisión de los enlaces inalámbricos disminuye considerablemente en espacios cerrados donde se pueden interponer entre los nodos obstáculos como paredes de bloque o placas de concreto reforzado.

Trabajo futuro

En el trabajo hasta aquí realizado se implementó un esquema de balanceo de carga simple en el cual se realiza una transferencia del tráfico a otras rutas. Este esquema permitirá el desarrollo y prueba de algoritmos de balanceo de carga más sofisticados que incluyan variables que afectan las comunicaciones inalámbricas, como la interferencia entre canales adyacentes, el ancho de banda y la potencia. Para esto, se deberán tener en cuenta los siguientes aspectos:

- Extraer la información generada por el protocolo B.A.T.M.A.N. de la red de control *mesh* para tomar decisiones de distribución de los flujos de tráfico sobre el plano de datos. Esto permitirá redirigir eficientemente los flujos de datos al tiempo que se maneja la interferencia co-canal generada entre los enlaces inalámbricos de los dos planos.
- Aplicar políticas de balanceo de carga también para el tráfico en el plano de control, que permitan manejar eficientemente el creciente número de mensajes hacia un solo controlador en una MANET.

La implementación del esquema de control *in-band*, como trabajo futuro, permitirá analizar el rendimiento de la red en comparación con el esquema de control *out-off-band* implementado en esta tesis. Esto además permitirá proponer otros trabajos en el eje de calidad del servicio del sistema Tlön donde se analicen variables como la interferencia entre las redes de control y de datos. También, permitirá formular políticas de administración del uso del medio compartido, y aprovechar las diferentes interfaces inalámbricas con las que cuentan los dispositivos heterogéneos en una MANET.

Se debe realizar una caracterización de los nodos inalámbricos en cuanto a sus capacidades para actuar como switches OpenFlow. Con un bajo número de flujos, como los manejados en esta tesis, no se presentaron inconvenientes de operación de los dispositivos utilizados con sus sistemas operativos. Sin embargo, con el manejo de un número excesivo de flujos, un nodo puede fallar en su tarea de retransmisión en la MANET. Nos proponemos investigar y responder preguntas como ¿Cuantos flujos puede atender simultáneamente una MANET?

A. B.A.T.M.A.N.

Especificaciones técnicas de B.A.T.M.A.N.

Los OGM recibidos por un nodo contienen información acerca del nodo originador y del nodo que lo retransmitió. Cada nodo alberga información en una lista sobre los otros originadores conocidos en la red. Esta lista tiene un registro por cada originador desde el cual se ha enviado el OGM, con la siguiente información (Neumann y cols., 2008):

- Dirección IPv4 de origen.
- Un *timestamp* que debe ser actualizado con cada OGM recibido de cada originador.
- NS de enlace bidireccional: Se usa para almacenar el NS del último OGM retransmitido exitosamente al mismo nodo que lo inició.
- NS actualizado: Es el NS del último OGM aceptado de algún originador.
- Lista HNA: Son todas las redes anunciadas por un originador con su rango IP y máscara de red.
- Capacidades de gateway: Indica si el originador es un gateway y sus parámetros.

De cada vecino en enlace local, se guarda una lista con la siguiente información:

- Ventana deslizante: El NS con que se recibe un nuevo OGM puede estar dentro de la ventana *(In-window)*, en cuyo caso no se actualiza el NS o puede estar fuera de rango *(Out-of-range)*, en cuyo caso se actualiza el NS al nuevo valor.
- Cuenta de paquetes: La cantidad de NS registrados en una ventana deslizante.
- Último tiempo válido: Es el *timestamp* cuando el último OGM válido fue recibido por vía del vecino.
- Último TTL: Es el *Time To Live* (TTL) del último OGM recibido por vía del vecino.

Tipos de paquetes B.A.T.M.A.N

Dentro del protocolo B.A.T.M.A.N existe un esquema de señalización de las tramas y de los paquetes generados para el descubrimiento de rutas. Servicios como seguridad, network coding, mensajes unicast, multicast y banderas adicionales indican los estados de los paquetes sobre la red. Esta información puede ser utilizada para predecir el comportamiento de la red y diferenciar los tráficos de control y de datos. Los principales tipos de paquetes B.A.T.M.A.N. son los siguientes:

```
BATADV_IV_OGM:        Mensaje originador (OGM) para B.A.T.M.A.N.  Advance.
BATADV_BCAST:         Paquetes que transportan datos con destino broadcast.
BATADV_CODED:         Paquetes codificados con network coding.
BATADV_UNICAST:       Paquetes que transportan datos con destino unicast.
BATADV_UNICAST_FRAG:  Paquetes unicast que transportan una parte (fragmento)
                      de los datos del paquete original.
BATADV_UNICAST_4ADDR: Paquetes unicast que incluyen la dirección origen del
                      nodo emisor.
BATADV_ICMP:          Paquetes unicast similares a ICMP utilizados para realizar
                      ping o traceroute.
BATADV_UNICAST_TVLV:  Paquetes unicast que transportan contenedores TVLV.
```

Configuración de las pruebas con B.A.T.M.A.N

Inicialmente, se realizaron las pruebas con los nodos operando en con el protocolo B.A.T.M.A.N. La conectividad de las interfaces inalámbricas se logro establecer fácilmente con la configuración automática de batman-adv en el boot del sistema operativo del nodo y a una configuración consistente de las interfaces de red teniendo en cuenta la topología del escenario de pruebas.

Se agregó el módulo batman-adv para ser cargado por el SO de cada nodo en el momento del boot, como se muestra a continuación:

```
muisca:~$ sudo nano /etc/modules

# /etc/modules: kernel modules to load at boot time.
#
# This file contains the names of kernel modules that should be loaded
# at boot time, one per line. Lines begining with '#' are ignored.

batman-adv
```

A continuación, se observa la configuración de las interfaces inalámbricas de un nodo que serán agregadas mas adelante al bridge creado por batman-adv con la interfaz virtual bat0.

```
muisca:~$ sudo vi /etc/network/interfaces

auto wlan0                              auto wlan1
iface wlan0 inet manual                 iface wlan1 inet manual
    mtu 1500                                mtu 1500
    wireless-channel 6                      wireless-channel 1
    wireless-essid Uqbar                    wireless-essid Tlon
    wireless-mode ad-hoc                    wireless-mode ad-hoc
    wireless-ap 02:12:34:56:78:9B           wireless-ap 02:12:34:56:78:9A
```

```
muisca:~$ sudo batctl if add wlan0 wlan1
muisca:~$ sudo ip link set up dev bat0
```

Verificamos la conectividad de las interfaces inalámbricas de cada nodo:

Conectividad inalámbrica entre los nodos con B.A.T.M.A.N.

```
muisca:~$ sudo iw wlan0 link            muisca:~$ sudo iw wlan1 link
Joined IBSS 02:12:34:56:78:9b           Joined IBSS 02:12:34:56:78:9a
(on wlan0)                              (on wlan1)
        9SSID: Uqbar                            SSID: Tlon
        freq: 2437                              freq: 2412

tikuna:~ $ sudo iw wlan0 link           tikuna:~ $ sudo iw wlan1 link
Joined IBSS 02:12:34:56:78:9a           Joined IBSS 02:12:34:56:78:9c
(on wlan0)                              (on wlan1)
        SSID: Tlon                              SSID: Orbis
        freq: 2412                              freq: 2462

kogui:~ $ sudo iw wlan0 link            kogui:~ $ sudo iw wlan1 link
Joined IBSS 02:12:34:56:78:9c           Joined IBSS 02:12:34:56:78:9b
(on wlan0)                              (on wlan1)
        SSID: Orbis                             SSID: Uqbar
        freq: 2462                              freq: 2437
```

B. OpenFlow y OVS

Especificaciones técnicas de OpenFlow

La versión especifica el protocolo OpenFlow que se está utilizando. Durante la fase de borrador anterior del protocolo OpenFlow, el bit más significativo se estableció para indicar una versión experimental. Los bits más bajos indican el número de revisión del protocolo. La versión del protocolo descrita por la especificación actual es 1.4.0, y su versión ofp es 0x05. Cada paquete OpenFlow inicia con la siguiente cabecera:

```
uint8_t version:      Versión del protocolo OpenFlow.
uint8_t type:         Una de las constantes OFTP.
uint16_t length:      Longitud incluida la cabecera.
uint32_t xid:         Id de la transacción asociada con el paquete.
```

Los principales tipos de paquetes OpenFlow son los siguientes:

Mensajes Inmutables:

```
OFPT_HELLO                OFPT_ERROR
OFPT_ECHO_REQUEST         OFPT_ECHO_REPLY
OFPT_EXPERIMENTER
OFPT_TABLE_MOD
```

Mensajes de configuración del switch:

```
OFPT_FEATURES_REQUEST    OFPT_FEATURES_REPLY
OFPT_GET_CONFIG_REQUEST  OFPT_GET_CONFIG_REPLY
OFPT_SET_CONFIG
OFPT_TABLE_MOD
```

Mensajes asíncronos:

```
OFPT_PACKET_IN             OFPT_FLOW_REMOVED
OFPT_PORT_STATUS
OFPT_TABLE_MOD
```

Mensajes de comando del controlador:

```
OFPT_PACKET_OUT            OFPT_FLOW_MOD
OFPT_GROUP_MOD             OFPT_PORT_MOD
OFPT_TABLE_MOD

CONTROLADOR              NODO
OFPT_HELLO               OFPT_HELLO
                         OFPT_ECHO_REQUEST
OFTP_FEATURES_REPLY
OFTP_ECHO_REPLY
OFTP_MULTIPART_REQUEST
OFTP_FLOW_MOD
                         OFTP_MULTIPART_REPLY
```

Como ejemplo, se muestra la traza de un mensaje `OFTP_HELLO` enviado por un nodo al controlador transportado sobre el protocolo B.A.T.M.A.N.

```
B.A.T.M.A.N. Unicast
Packet Type: BATADV UNICAST (64)
Version: 15
Time to Live: 50
TT Version: 5
Destination: Chongqin 8c:c0:29 (ec:5c:68:8c:c0:29)

OpenFlow 1.4
Version: 1.4 (0x05)
Type: OFPT HELLO (0)
Length: 8
Transaction ID: 1
```

Configuración del controlador

Existen numerosas formas, adicionales a la que se utilizo en esta tesis, como los contenedores de Docker u otras maquinas virtuales, para instalar y conectar el controlador Faucet a la red de control. Dependiendo del hardware y sistema operativo donde se instale Faucet, se requieren algunos paquetes para su compilación, pero en general se requiere que el sistema cuente con la versión de Python 3.4 o superior (Open Virtual Switch, 2016).

La configuración del controlador Faucet contiene 2 vlans: 100 para la comunicación entre los host virtuales y 200 para la comunicación entre los servidores virtuales. Adicionalmente, se especifica y el resultado en su log de eventos. En esta configuración se establecieron 4

datapaths con 2 puertos para cada nodo Raspberry Pi que se comunican utilizando la VLAN nativa 100.

En la Figura **B-1**, se observa la representación gráfica de configuración realizada mediante el archivo YAML presentado a continuación.

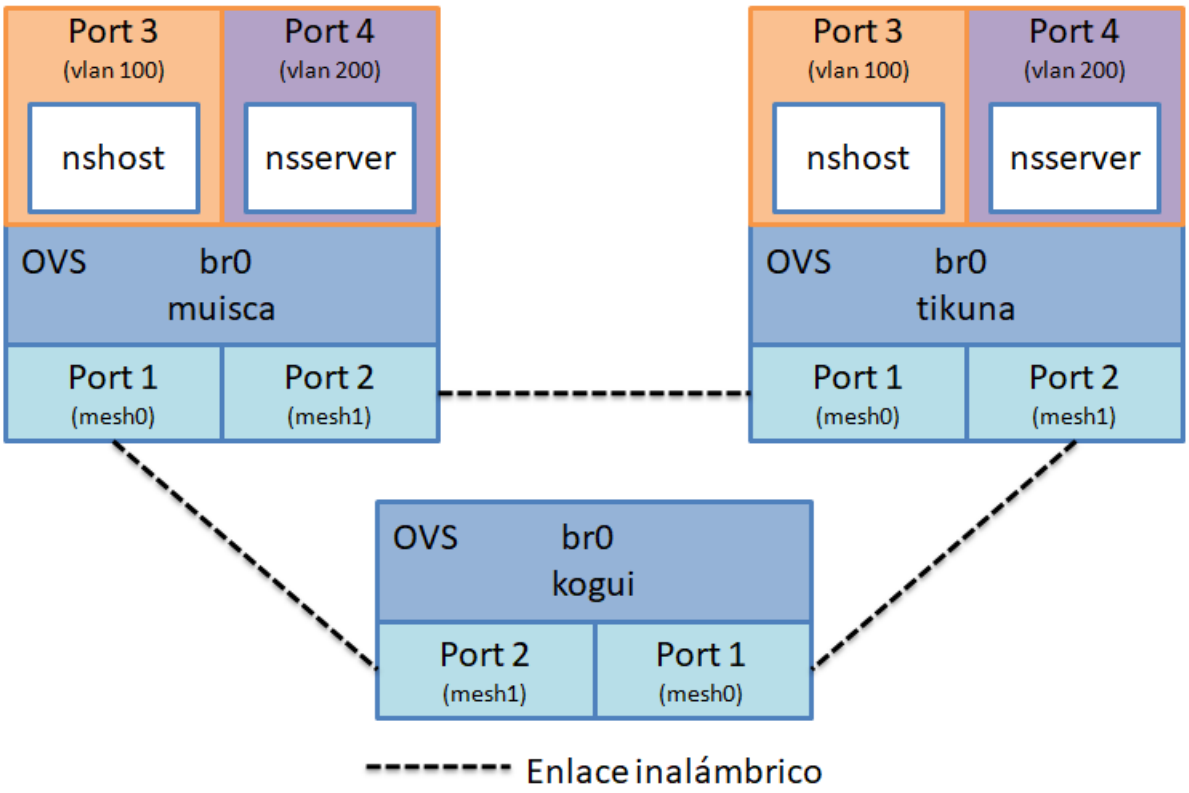


Figura B-1.: Topología en anillo de la red tolerante a fallos de un switch o puerto configurada en el controlador Faucet con las VLAN 100 y 200. Elaboración propia.

Configuración del controlador Faucet ubicada en /etc/faucet/faucet.yaml

```
vlans:
 vlan100:
  vid: 100
 vlan200:
  vid: 200

dps:
 muisca:
  dp_id: 0x1
   timeout: 3600
   arp_neighbor_timeout: 1700
   hardware: 'Open vSwitch'
   stack:
    priority: 1
   interfaces:
    1:
     description: "enlace Uqbar"
     stack:
      dp: kogui
      port: 2
    2:
     description: "enlace Tlon"
     stack:
      dp: tikuna
      port: 1
    3:
     description: "hostmuisca"
     native_vlan: 100
    4:
     description: "servermuisca"
     native_vlan: 200

 tikuna:
  dp_id: 0x2
   timeout: 3600
   arp_neighbor_timeout: 1700
   hardware: 'Open vSwitch'
   interfaces:
    1:
     descrition: "enlace Tlon"
     stack:
      dp: muisca
      port: 2
    2:
     description: "enlace Orbis"
     stack:
      dp: kogui
      port: 1
    3:
     description: "hosttikuna"
     native_vlan: 100
    4:
     description: "servertikuna"
     native_vlan: 200

 kogui:
  dp_id: 0x3
   timeout: 3600
   arp_neighbor_timeout: 1700
   hardware: 'Open vSwitch'
   interfaces:
    1:
     description: "enlace Orbis"
     stack:
      dp: tikuna
      port: 2
    2:
     description: "enlace Uqbar"
     stack:
      dp: muisca
      port: 1
```

Se Verifica que el controlador Faucet haya creado los datapads tal como los definimos en la configuración anterior, revisando el log de eventos ubicado en /var/log/faucet/faucet.log:

```
$ sudo vi /var/log/faucet/faucet.log

Add new datapath DPID 1 (0x1)
DPID 1 (0x1) muisca Configuring VLAN vlan100 vid:100
untagged: Port 1, Port 2
Add new datapath DPID 2 (0x2)
DPID 2 (0x2) tikuna Configuring VLAN vlan100 vid:100
untagged: Port 1, Port 2
Add new datapath DPID 3 (0x3)
DPID 3 (0x3) kogui Configuring VLAN vlan100 vid:100
untagged: Port 1, Port 2
```

También, se verifica que el controlador este aprendiendo las direcciones MAC de los paquetes iniciales recibidos por los host virtuales conectados a los nodos. En el primer datapad (DPID 1), se observa que muisca recibió un paquete tipo *multicast all nodes* (MAC destino 33:33:00:00:00:02) e IPv6 (IP destino ff02::2) por el puerto 3 (vlan 100).

```
$ sudo vi /var/log/faucet/faucet.log

DPID 1 (0x1) muisca L2 learned on Port 3 f8:1a:67:07:f0:37 (L2 type
0x86dd, L2 dst 33:33:00:00:00:02, L3 src fe80::fa1a:67ff:fe07:f037,
L3 dst ff02::2) Port 2 VLAN 100 (1 hosts total)

DPID 2 (0x2) tikuna L2 learned on Port 3 e8:94:f6:1a:fa:64 (L2 type
0x86dd, L2 dst 33:33:00:00:00:fb, L3 src fe80::ea94:f6ff:fe1a:fa64,
L3 dst ff02::fb) Port 2 VLAN 100 (2 hosts total)

DPID 3 (0x3) kogui L2 learned on Port 3 00:17:c4:1f:64:41 (L2 type
0x86dd, L2 dst 33:33:00:00:00:fb, L3 src fe80::217:c4ff:fe1f:6441,
L3 dst ff02::fb) Port 2 VLAN 100 (1 hosts total)
```

Configuración de OVS

Con los siguientes comandos se crean el bridge en cada uno de los nodos y se definen sus puertos los cuales deben coincidir con los definidos en la configuración de los datapaths definidos en el controlador Faucet anteriormente:

```
muisca:~$ sudo ovs-vsctl add-br br0 \
         -- set bridge br0 other-config:datapath-id=0000000000000001 \
         -- add-port br0 mesh0 -- set interface mesh0 ofport_request=1 \
         -- add-port br0 mesh1 -- set interface mesh1 ofport_request=2 \
         -- add-port br0 veth0 -- set interface veth0 ofport_request=3 \
         -- add-port br0 veth2 -- set interface veth2 ofport_request=4 \
         -- set-controller br0 tcp:192.168.1.105:6653 \
         -- set controller br0 connection-mode=out-of-band

tikuna:~ $ sudo ovs-vsctl add-br br0 \
         -- set bridge br0 other-config:datapath-id=0000000000000002 \
         -- add-port br0 mesh0 -- set interface wlan0 ofport_request=1 \
         -- add-port br0 mesh1 -- set interface wlan1 ofport_request=2 \
         -- add-port br0 veth0 -- set interface veth0 ofport_request=3 \
         -- add-port br0 veth2 -- set interface veth2 ofport_request=4 \
         -- set-controller br0 tcp:192.168.1.105:6653 \
         -- set controller br0 connection-mode=out-of-band

kogui:~ $ sudo ovs-vsctl add-br br0 \
         -- set bridge br0 other-config:datapath-id=0000000000000003 \
         -- add-port br0 mesh0 -- set interface mesh0 ofport_request=1 \
         -- add-port br0 mesh1 -- set interface mesh1 ofport_request=2 \
         -- set-controller br0 tcp:192.168.1.103:6653 \
         -- set controller br0 connection-mode=out-of-band
```

Verificamos la configuración del uno de OVS:

```
sua@muisca:~$ sudo ovs-vsctl show
3d37833c-5fe9-4f49-b1a8-fea9a1c50e57
    Bridge "br0"
        Controller "tcp:192.168.1.105:6653"
            is_connected: true
        Port "mesh0"
            Interface "mesh0"
        Port "mesh1"
            Interface "mesh1"
        Port "veth0"
            Interface "veth0"
        Port "veth2"
            Interface "veth2"
        Port "br0"
            Interface "br0"
                type: internal
    ovs_version: "2.10.7"
```

Se realizó la configuración en modo mesh de las interfaces inalámbricas de muisca y de forma similar se configuran tikuna y kogui.

```
muisca:~$ sudo iw dev wlan0 interface add mesh0 type mp mesh_id Uqbar
muisca:~$ sudo iw dev mesh0 set channel 6
muisca:~$ sudo ifconfig wlan0 down
muisca:~$ sudo ifconfig mesh0 up
muisca:~$ sudo ip addr add 10.0.0.1/24 dev mesh0

muisca:~$ sudo iw dev wlan1 interface add mesh1 type mp mesh_id Tlon
muisca:~$ sudo iw dev mesh1 set channel 1
muisca:~$ sudo ifconfig wlan1 down
muisca:~$ sudo ifconfig mesh1 up
muisca:~$ sudo ip addr add 10.0.0.2/24 dev mesh1
```

A continuación, se observa el resultado de la configuración inalámbrica en muisca.

```
muisca:~$ sudo iwconfig
lo        no wireless extensions.
eth0      no wireless extensions.
veth0     no wireless extensions.
veth2     no wireless extensions.
br0       no wireless extensions.
ovs-system  no wireless extensions.

wlan0     IEEE 802.11  ESSID:off/any
          Mode:Managed  Access Point: Not-Associated   Tx-Power=20 dBm
          Retry short limit:7   RTS thr:off   Fragment thr:off
          Encryption key:off
          Power Management:off

wlan1     IEEE 802.11  ESSID:off/any
          Mode:Managed  Access Point: Not-Associated   Tx-Power=20 dBm
          Retry short limit:7   RTS thr:off   Fragment thr:off
          Encryption key:off
          Power Management:off

mesh0     IEEE 802.11  Mode:Auto  Tx-Power=20 dBm
          Retry short limit:7   RTS thr:off   Fragment thr:off
          Power Management:on

mesh1     IEEE 802.11  Mode:Auto  Tx-Power=20 dBm
          Retry short limit:7   RTS thr:off   Fragment thr:off
          Power Management:off
```

```
muisca:~$ sudo iw dev
phy#0                                   phy#1
 Interface mesh0                         Interface mesh1
  ifindex 7                               ifindex 8
  wdev 0x3                                wdev 0x100000003
  addr 00:17:c4:1f:64:41                  addr e8:94:f6:1a:fa:64
  type mesh point                         type mesh point
  channel 6 (2437 MHz), width: 20         channel 1 (2412 MHz), width: 20
  MHz (no HT), center1: 2437 MHz          MHz (no HT), center1: 2412 MHz
  txpower 20.00 dBm                       txpower 20.00 dBm
 Interface wlan0                         Interface wlan1
  ifindex 3                               ifindex 5
  wdev 0x1                                wdev 0x100000001
  addr 00:17:c4:1f:64:41                  addr e8:94:f6:1a:fa:64
  type managed                            type managed
  txpower 20.00 dBm                       txpower 20.00 dBm
```

Configuramos el host y servidor virtual en muisca utilizando namespaces y de forma similar se configura tikuna.

```
muisca:~$ sudo ip netns add nshost
muisca:~$ sudo ip link add veth0 type veth peer name veth1
muisca:~$ sudo ip link set veth1 netns nshost
muisca:~$ sudo ip netns exec nshost ip addr add 10.0.1.1/24 dev veth1
muisca:~$ sudo ip netns exec nshost ip link set dev veth1 up
muisca:~$ sudo ip netns exec nshost ip link set up dev lo
muisca:~$ sudo ip link set up dev veth0

muisca:~$ sudo ip netns add nsserver
muisca:~$ sudo ip link add veth2 type veth peer name veth3
muisca:~$ sudo ip link set veth3 netns nsserver
muisca:~$ sudo ip netns exec nsserver ip addr add 10.0.2.1/24 dev veth3
muisca:~$ sudo ip netns exec nsserver ip link set dev veth3 up
muisca:~$ sudo ip netns exec nsserver ip link set up dev lo
muisca:~$ sudo ip link set up dev veth2
```

Verificamos toda la configuración de direccionamiento IP de muisca.

```
muisca:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group
default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
state UP group default qlen 1000
    link/ether 00:1d:72:25:ca:61 brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.102/24 brd 192.168.1.255 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::21d:72ff:fe25:ca61/64 scope link
       valid_lft forever preferred_lft forever
3: wlan0: <BROADCAST,MULTICAST> mtu 1500 qdisc mq state DOWN group
default qlen 1000
    link/ether 00:17:c4:1f:64:41 brd ff:ff:ff:ff:ff:ff
4: wlan1: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group
default qlen 1000
    link/ether e8:94:f6:1a:fa:64 brd ff:ff:ff:ff:ff:ff
5: mesh0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq master
ovs-system state UP group default qlen 1000
    link/ether 00:17:c4:1f:64:41 brd ff:ff:ff:ff:ff:ff
    inet 10.0.0.1/24 scope global mesh0
       valid_lft forever preferred_lft forever
    inet6 fe80::217:c4ff:fe1f:6441/64 scope link
       valid_lft forever preferred_lft forever
6: mesh1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq master
ovs-system state UP group default qlen 1000
    link/ether e8:94:f6:1a:fa:64 brd ff:ff:ff:ff:ff:ff
    inet 10.0.0.2/24 scope global mesh1
       valid_lft forever preferred_lft forever
    inet6 fe80::ea94:f6ff:fe1a:fa64/64 scope link
       valid_lft forever preferred_lft forever
```

Continua en la página siguiente

```
                                    Continua de la página anterior
7: veth0@if9: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue
master ovs-system state UP group default qlen 1000
   link/ether fa:ec:58:73:f5:88 brd ff:ff:ff:ff:ff:ff link-netns nshost
   inet6 fe80::f8ec:58ff:fe73:f588/64 scope link
      valid_lft forever preferred_lft forever
8: veth2@if11: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue
master ovs-system state UP group default qlen 1000
   link/ether 76:4b:fd:f9:ca:33 brd ff:ff:ff:ff:ff:ff link-netns nsserver
   inet6 fe80::744b:fdff:fef9:ca33/64 scope link
      valid_lft forever preferred_lft forever
9: ovs-system: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue
state UNKNOWN group default qlen 1000
   link/ether 3e:89:7c:97:67:cc brd ff:ff:ff:ff:ff:ff
   inet6 fe80::3c89:7cff:fe97:67cc/64 scope link
      valid_lft forever preferred_lft forever
10: br0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state
UNKNOWN group default qlen 1000
   link/ether 00:17:c4:1f:64:41 brd ff:ff:ff:ff:ff:ff
   inet6 fe80::217:c4ff:fe1f:6441/64 scope link
      valid_lft forever preferred_lft forever
```

Finalmente, verificamos la configuración IP de uno del host virtual de muisca.

```
muisca:~$ sudo ip netns exec nsmuisca ip address
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
    group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
11: veth1@if6: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc
    noqueue state UP group default qlen 1000
    link/ether b6:96:83:f5:91:48 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 10.0.1.1/24 scope global veth1
       valid_lft forever preferred_lft forever
    inet6 fe80::b496:83ff:fef5:9148/64 scope link
       valid_lft forever preferred_lft forever
```

Reglas de acción del OVS

A continuación, se definen las reglas de acción en cada una de las siguientes tablas conmutación en los nodos :

Tabla 0: *Access Control List* (ACLs) basadas en puertos
Tabla 1: Procesamiento de VLANs de entrada
Tabla 2: ACLs basadas en VLANs
Tabla 3: Procesamiento L2 de entrada, aprendizaje de MAC
Tabla 4: Reenvío L3 para IPv4
Tabla 5: Reenvío L3 para IPv6
Tabla 6: Procesamiento IP virtual, e.g. para la IP del router implementada por Faucet
Tabla 7: Procesamiento L2 de salida
Tabla 8: Inundación

Para el control de admisión en la tabla 0, se define un flujo para descartar los paquetes son con fuente *multicast*, con el *Spanning Tree Protocol* y para pasar a la siguiente tabla OpenFlow los paquetes que no hagan *match* con estas condiciones:

```
$sudo ovs-ofctl add-flow br0 \
"table=0, dl_src=01:00:00:00:00:00/01:00:00:00:00:00, actions=drop" \
"table=0, dl_dst=01:80:c2:00:00:00/ff:ff:ff:ff:ff:f0, actions=drop" \
"table=0, priority=0, actions=resubmit(,1)"
```

Se comprueba la condición en la que un paquete es descartado:

```
$sudo ovs-appctl ofproto/trace br0 in_port=mesh1,dl_dst=01:80:c2:00:00:05
Flow: in_port=2,vlan_tci=0x0000,dl_src=00:00:00:00:00:00,
dl_dst=01:80:c2:00:00:05,dl_type=0x0000
bridge("br0")
-------------
 0. dl_dst=01:80:c2:00:00:00/ff:ff:ff:ff:ff:f0, priority 32768
    drop
Final flow: unchanged
Megaflow: recirc_id=0,eth,in_port=2,
dl_src=00:00:00:00:00:00/01:00:00:00:00:00,
dl_dst=01:80:c2:00:00:00/ff:ff:ff:ff:ff:f0,dl_type=0x0000
Datapath actions: drop
```

También, como ejemplo, se genera la traza de un paquete que ingresa por el puerto veth0 del nodo y pasa por las diferentes tablas de conmutación. El puerto veth0 está configurado como puerto de acceso por lo que en la tabla 0 se etiquetan los paquete que ingresan al bridge con la

VLAN 100 (acción *push vlan:0x8100*) y se reenvían a la tabla 1 donde el controlador aprende la dirección MAC de origen de los paquetes y se reenvían a la siguiente tabla. Finalmente, se les asigna una prioridad a los paquetes que ingresan en la tabla 3 y en la tabla 4 se reenvían al otro puerto del bridge omitiendo el reenvió hacia el mismo puerto de ingreso.

```
$sudo ovs-appctl ofproto/trace br0 in_port=veth0,
dl_src=00:19:d1:52:74:63, dl_dst=f8:1a:67:07:f0:37 -generate
Flow: in_port=3,vlan_tci=0x0000,
dl_src=00:19:d1:52:74:63, dl_dst=f8:1a:67:07:f0:37,dl_type=0x0000
bridge("br0")
-------------
 0. in_port=2,vlan_tci=0x0000/0x1fff, priority 4096, cookie 0x5adc15c0
    push_vlan:0x8100
    set_field:4196->vlan_vid
    goto_table:1
 1. dl_vlan=100, priority 4096, cookie 0x5adc15c0
    CONTROLLER:96
    goto_table:2
 2. priority 0, cookie 0x5adc15c0
    goto_table:3
 3. dl_vlan=100, priority 8192, cookie 0x5adc15c0
    output:1
    output:2
     >> skipping output to input port
Final flow: unchanged
Megaflow: recirc_id=0,eth,in_port=3,vlan_tci=0x0000/0x1fff,
dl_src=00:19:d1:52:74:63,dl_dst=f8:1a:67:07:f0:37,dl_type=0x0000
Datapath actions:
push_vlan(vid=100,pcp=0),userspace(pid=3743822145,controller(reason=1,
dont_send=0,continuation=0,recirc_id=118,rule_cookie=0x5adc15c0,
controller_id=0,max_len=96)),2
```

C. Software y programación

A continuación, se presentan algunas clases del software desarrollado para las pruebas: Balanceador, Controller y LocalAgent.

Clase Balanceador en balanceador.py

```
class Balanceador(app_manager.RyuApp):

    def __init__(self, *args, **kwargs):
        super(Redirigir, self).__init__(*args, **kwargs)
        self.mac_to_port = {}
        self.flow_config_service = InitialRuleService(self.logger)

    @set_ev_cls(ofp_event.EventOFPPortStatus, MAIN_DISPATCHER)
    def get_async_reply_handler(self, ev):
        msg = ev.msg
        datapath = msg.datapath
        reason = msg.reason
        if(reason == OFPPR_MODIFY):
            self.track_node_fsm.port_status_uppdate(datapath, msg.desc)

    @set_ev_cls(ofp_event.EventOFPSwitchFeatures, CONFIG_DISPATCHER)
    def switch_features_handler(self, ev):
        initial_mods = self.flow_config_service.initial_rules(ev)
        datapath = ev.msg.datapath
        ofproto = datapath.ofproto
        parser = datapath.ofproto_parser
```

Clase Controller en controller.py

```
class Controller():

    datpath_service = DataPathService()
    operational_plane_service = OperationalPlaneService()

    def __init__(self):
        self.logger = logging.getLogger("Controller")
        self.track_switch_datapath = None
        # Datapath for contacting the sender node
        self.node_switch_datapath = None
        # Datapath for contacting the receiver node
        self.timestamp_ns = None
        self.configure_throughput_thread = None
        self.inventory_service = InventoryService.get()
        self.receiver_connected = False
        self.sender_connected = False
        self.connected = False

    def port_status_update(self,datapath,port):
        is_sender = datapath.id ==
        self.inventory_service.nodeside_switch_dp_id()
        status = "up" if(port.state == 4) else "down"
        device = "receiver" if(is_receiver) else "sender"

    # Register a new datapath
    def register_datapath(self,dp):
        self.logger.info("Register Datapath %d",dp.id)
        if (dp.id == self.inventory_service.nodeside_switch_dp_id()):
            self.logger.info("Node connected")
            self.node_switch_datapath = dp
            self.receiver_connected = True
        elif (dp.id == self.inventory_service.trackside_switch_dp_id()):
            self.track_switch_datapath = dp
            self.logger.info("Track switch connected")
        elif (dp.id == self.inventory_service.sender_dp_id()):
            self.sender_switch_path = dp
            self.logger.info("Configure sender")
            self.sender_connected = True
```

Clase LocalAgent en localagent.py

```
class LocalAgent():
    def on_connect(self,client,userdata,flags,rc):
        try:
            if self.position == "receiver":
                client.subscribe("position")
            else:
                client.subscribe("control")
        except Exception:
            print("Exception in user code:")
            print("-"*60)
            traceback.print_exc(file=sys.stdout)
            print("-"*60)
    def handle_position(self,data):
        self.timestamp = time.time_ns()
        self.logger.info("Position-Data %s",data)
        self.switch_current_DP()
```

D. Análisis con Wireshark

A continuación, se observan las trazas de los datagramas en Wireshark recibidas con el capturador Air Pcap en los diferentes canales inalámbricos utilizados. Con estas trazas se logro verificar la correcta configuración de las interfaces WiFi en los nodos de los escenarios de prueba y detectar los posibles errores de la información en los paquetes IEEE 802.11.

A continuación, podemos observar una traza en la que capturamos un paquete ARP request:

```
>IEEE 802.11 Data, Flags: ........C
    Type/Subtype: Data (0x0020)
    >Frame Control Field: 0x0800
        .... ..00 = Version: 0
        .... 10.. = Type: Data frame (2)
        0000 .... = Subtype: 0
        >Flags: 0x00
            .... ..00 = DS status: Not leaving DS or network is operating
                        in AD-HOC mode (To DS: 0 From DS: 0) (0x0)
            .... .0.. = More Fragments: This is the last fragment
            .... 0... = Retry: Frame is not been retransmitted
            ...0 .... = PWR MGT: STA will stay up
            ..0. .... = More data: No data buffered
            .0.. .... = Protected flag: Data is not protected
            0... .... = +HTC/Order flag: Not strictly ordered
        .000 0001 0011 1010 = Duration: 314 microseconds
        Receiver address: Broadcast (ff:ff:ff:ff:ff:ff)
        Transmitter address: 00:00:00:00:00:11 (00:00:00:00:00:11)
        Destination adddress: Broadcast (ff:ff:ff:ff:ff:ff)
        Source address: 00:00:00:00:00:11 (00:00:00:00:00:11)
        BSS Id: MS-NLB-PhysServer-18_34:56:78:9a (02:12:34:56:78:9a)
        .... .... .... 0000 = Fragment number: 0
        0001 1100 0011 .... = Sequence number: 451
        Frame check sequence: 0x32679ffc [unverified]
        [FCS Status: Unverified]
```

```
>Adress Resolution Protocol (request)
    Hardware type: Ethernet (1)
    Protocol type: IPv4 (0x0800)
    Hardware size: 6
    Protocol size: 4
    Op code: request (1)
    Sender MAC address: 00:00:00:00:00:11 (00:00:00:00:00:11)
    Sender IP address: 10.0.1.254
    Target MAC address: 00:00:00:00:00:00 (00:00:00:00:00:00)
    Target IP address: 10.0.1.1
```

A continuación, podemos observar la traza en la que capturamos un paquete de broadcast en el cual reconocemos la configuración mesh en las interfaces inalámbricas de dos nodos que se comunican por el enlace Uqbar en el canal 6:

```
>802.11 radio information
    PHY type: 802.11b (HR/DSSS) (4)
    Short preamble: False
    Data rate: 1,0 Mb/s
    Channel: 6
    Frequency: 2437MHz
    Signal strength (dB): 81 dB
    Signal strength (dBm): -19 dBm
    Noise level (dBm): -100 dBm
    Signal/noise ratio (dB): 81 dB
  > [Duration: 856us]
       [Preamble: 192us]
```

```
> IEEE 802.11 Wireless Management
  > Tagged parameters (43 bytes)
    > Tag: Mesh ID: Uqbar
         Tag Number: Mesh ID (114)
         Tag length: 5
         Mesh ID: Uqbar
    > Tag: Mesh Configuration
         Tag Number: Mesh Configuration (113)
         Tag length: 7
         Path Selection Protocol: 0x01
         Path Selection Metric: 0x01
         Congestion control: 0x00
         Syncronization Method: 0x01
         Authentication Protocol: 0x00
       > Formation Info: 0x02
            .... ...0 = Connected to Mesh Gate: No
            .000 001. = Number of Peerings: 1
            0... .... = Connected to AS: No
       > Capability: 0x09
            .... ...1 = Accepting Additional Mesh Peerings: Yes
            .... ..0. = MCCA Support: No
            .... .0.. = MCCA Enabled: No
            .... 1... = Mesh Forwarding: Yes
            ...0 .... = MBCA Enables: No
            ..0. .... = TBTT Adjustment: No
            .0.. .... = Power save: None of the peer-specific mesh power
                        modes is deep sleep mode
            0... .... = Reserved: 0x0
```

Medición del throughput con Iperf

A continuación, se observan los comandos utilizados para generar tráfico entre Muisca y Tikuna con la herramienta Iperf y el resultado del throughput medido por el servidor en Tikuna.

```
sua@muisca:~$ sudo ip netns exec nshost iperf -c 10.0.1.2 -p 20000 -u
-t 10 -b 0.1Mbps
------------------------------------------------------------
Client connecting to 10.0.1.2, UDP port 20000
Sending 1470 byte datagrams, IPG target: 9.54 us (kalman adjust)
UDP buffer size:  160 KByte (default)
------------------------------------------------------------
local 10.0.1.1 port 44169 connected with 10.0.1.2 port 20000
Interval       Transfer     Bandwidth
0.0-10.0 sec  18.3 MBytes  15.4 Mbits/sec
Sent 13086 datagrams
Server Report:
0.0-10.0 sec  18.3 MBytes  15.3 Mbits/sec   2.447 ms    0/13086 (0%)
```

```
aikia@tikuna:~$ sudo ip netns exec nshost iperf -s -p 20000 -u
------------------------------------------------------------
Server listening on UDP port 20000
Receiving 1470 byte datagrams
UDP buffer size:  208 KByte (default)
------------------------------------------------------------
local 10.0.1.2 port 20000 connected with 10.0.1.1 port 50217
Interval       Transfer     Bandwidth    Jitter   Lost/Total Datagrams
0.0-10.1 sec  18.3 MBytes  15.4 Mbits/sec   2.447 ms    0/13086 (0%)
local 10.0.1.2 port 20000 connected with 10.0.1.1 port 46685
0.0-10.4 sec  3.57 MBytes  2.89 Mbits/sec  18.725 ms    0/ 2550 (0%)
local 10.0.1.2 port 20000 connected with 10.0.1.1 port 36744
0.0-10.0 sec  18.1 MBytes  15.2 Mbits/sec   2.683 ms    0/12937 (0%)
local 10.0.1.2 port 20000 connected with 10.0.1.1 port 48744
0.0-10.3 sec  3.63 MBytes  2.95 Mbits/sec  18.177 ms    0/ 2586 (0%)
local 10.0.1.2 port 20000 connected with 10.0.1.1 port 33463
0.0-10.0 sec  18.3 MBytes  15.3 Mbits/sec   3.575 ms    0/13081 (0%)
local 10.0.1.2 port 20000 connected with 10.0.1.1 port 60174
0.0-10.2 sec  3.63 MBytes  3.00 Mbits/sec  12.077 ms    0/ 2589 (0%)
```

Referencias

Abbas, A. M., y Kure, O. (2009). A probabilistic quality of service routing for mobile Ad hoc networks. En *2009 first international conference on networked digital technologies* (pp. 256–260). doi: 10.1109/NDT.2009.5272075

Ajmal, S., Jabeen, S., Rasheed, A., y Hasan, A. (2015). An intelligent hybrid spread spectrum MAC for interference management in mobile ad hoc networks. *Computer Communications*, *72*, 116–129. Descargado de `http://www.sciencedirect.com/science/article/pii/S0140366415001541` doi: http://dx.doi.org/10.1016/j.comcom.2015.04.006

Akyildiz, I. F., Wang, P., y Lin, S.-C. (2015). SoftAir: A software defined networking architecture for 5G wireless systems. *Computer Networks*, *85*, 1–18. Descargado de `http://www.sciencedirect.com/science/article/pii/S1389128615001632` doi: https://doi.org/10.1016/j.comnet.2015.05.007

Amokrane, A., Langar, R., Boutabayz, R., y Pujolle, G. (2013, 12). Online flow-based energy efficient management in Wireless Mesh Networks. En *2013 ieee global communications conference (globecom)* (pp. 329–335). doi: 10.1109/GLOCOM.2013.6831092

An, N., y Lim, H. (2019). Poster: Protecting Control Planes in In-Band Software-Defined Wireless Networks. En *The 25th annual international conference on mobile computing and networking.* New York, NY, USA: Association for Computing Machinery. Descargado de `https://doi.org/10.1145/3300061.3343396` doi: 10.1145/3300061.3343396

Arun, K. P., Chakraborty, A., y Manoj, B. S. (2014, 12). Communication overhead of an OpenFlow wireless mesh network. En *2014 ieee international conference on advanced networks and telecommuncations systems (ants)* (pp. 1–6). doi: 10.1109/ANTS.2014.7057232

Barbeau, M., y Kranakis, E. (2007). *Principles of Ad hoc networking* (First ed.; Wiley, Ed.). John Wiley {\&} Sons, Ltd.

Betzler, A., Quer, F., Camps-Mur, D., Demirkol, I., y Garcia-Villegas, E. (2016). On the benefits of wireless SDN in networks of constrained edge devices. En *2016 european conference on networks and communications (eucnc)* (pp. 37–41). doi: 10.1109/EuCNC.2016.7561000

Bibri, S. (2017). The IoT for Smart Sustainable Cities of the Future: An Analytical Framework for Sensor-Based Big Data Applications for Environmental Sustainability. *Sustainable Cities and Society*, *38*. doi: 10.1016/j.scs.2017.12.034

Chung, J., González, G., Armuelles, I., Robles, T., Alcarria, R., y Morales, A. (2012,

10). Characterizing the multimedia service capacity of wireless mesh networks for rural communities. En *2012 ieee 8th international conference on wireless and mobile computing, networking and communications (wimob)* (pp. 628–635). doi: 10.1109/WiMOB.2012.6379142

Clausen, T., y Jacquet, P. (2003). *RFC3626: Optimized Link State Routing Protocol (OLSR).* USA: RFC Editor.

Dely, P., Kassler, A., y Bayer, N. (2011, 7). OpenFlow for Wireless Mesh Networks. En *Computer communications and networks (icccn), 2011 proceedings of 20th international conference on* (pp. 1–6). doi: 10.1109/ICCCN.2011.6006100

de Sousa, N. F. S., Perez, D. A. L., Rosa, R. V., Santos, M. A. S., y Rothenberg, C. E. (2018). Network Service Orchestration: {A} Survey. *CoRR*, *abs/1803.0*. Descargado de `http://arxiv.org/abs/1803.06596`

Detti, A., Pisa, C., Salsano, S., y Blefari-Melazzi, N. (2013, 10). Wireless Mesh Software Defined Networks (wmSDN). En *2013 ieee 9th international conference on wireless and mobile computing, networking and communications (wimob)* (pp. 89–95). doi: 10.1109/WiMOB.2013.6673345

Fals Borda, O. (2000). *Acción y Espacio. autonomías en la nueva República*. IEPRI.

Faucet Developers. (2021). *Faucet Documentation*. Descargado de `https://faucet.nz/`

Fitzek, F. H. P., y Katz, M. D. (2014). *Mobile Clouds*. Chichester, UK: John Wiley & Sons, Ltd. Descargado de `http://doi.wiley.com/10.1002/9781118801338` doi: 10.1002/9781118801338

Furlan, D. (2011). *Improving BATMAN Routing Stability and Performance* (Tesis Doctoral no publicada). Universita Degli Studi di Trento.

Gast, M. (2005). *802.11 Wireless Networks The Definitive Guide*. O'Reilly.

Hakiri, A., Sellami, B., Patil, P., Berthou, P., y Gokhale, A. (2017, 10). Managing Wireless Fog Networks using Software-Defined Networking. En *2017 ieee/acs 14th international conference on computer systems and applications (aiccsa)* (pp. 1149–1156). doi: 10.1109/AICCSA.2017.9

Han, S. Y., Shin, B., y Lee, D. (2014, 9). A fine-grain partial MAC virtualization to support cross layer design in wireless ad hoc networks. En *Local computer networks (lcn), 2014 ieee 39th conference on* (pp. 506–509). doi: 10.1109/LCN.2014.6925828

Haque, I. T., y Abu-Ghazaleh, N. (2016). Wireless Software Defined Networking: a Survey and Taxonomy. *IEEE Communications Surveys Tutorials*, *PP*(99), 1. doi: 10.1109/COMST.2016.2571118

Hardkernel co Ltd. (2020). *ODROID XU4 Wiki.* Descargado de `https://wiki.odroid.com/odroid-xu4/odroid-xu4`

Hobbes, T. (1996). *Leviathan*. Oxford University Press.

Huang, H., Li, P., Guo, S., y Zhuang, W. (2015). Software-defined wireless mesh networks: architecture and traffic orchestration. *IEEE Network*, *29*(4), 24–30. doi: 10.1109/MNET.2015.7166187

Huang, Y., y Li, G. (2010). A Semantic Analysis for Internet of Things. *Intelligent Computation Technology and Automation, International Conference on*, *1*, 336–339. doi: 10.1109/ICICTA.2010.73

Hundeboll, M., y Ledet-Pedersen, J. (2011). *Inter-flow network coding for wireless mesh networks* (Tesis Doctoral no publicada). Aalborg University.

Hyytiä, E., y Virtamo, J. (2007). On Traffic Load Distribution and Load Balancing in Dense Wireless Multihop Networks. *EURASIP Journal on Wireless Communications and Networking*, *2007*(1), 16932. doi: 10.1155/2007/16932

Jagadeesan, N. A., y Krishnamachari, B. (2015). Software-Defined Networking Paradigms in Wireless Networks: A Survey. *ACM Computing Surveys*, *47*(2), 27:1 - 27:11.

Jain, R., Chiu, D. M., y WR, H. (1984). A Quantitative Measure Of Fairness And Discrimination For Resource Allocation In Shared Computer Systems. *CoRR*, *cs.NI/9809*.

Jain, S., Kumar, A., Mandal, S., Ong, J., Poutievski, L., Singh, A., ... Vahdat, A. (2013). B4: Experience with a Globally Deployed Software Defined WAN. En *Proceedings of the acm sigcomm conference.* Hong Kong, China. Descargado de `http://cseweb.ucsd.edu/ vahdat/papers/b4-sigcomm13.pdf`

Jo, M. Y., y Kim, K. (2016). A research on the regional routing scheme based mobile agent for SDN. En *International conference on information networking* (Vol. 2016-March, pp. 211–213). doi: 10.1109/ICOIN.2016.7427116

Kleinberg, J., Rabani, Y., y Tardos, E. (2001, 8). Fairness in Routing and Load Balancing. *Journal of Computer and System Sciences*, *63*(1), 2–20. Descargado de `http://linkinghub.elsevier.com/retrieve/pii/S0022000001917520` doi: 10.1006/jcss.2001.1752

Kuhn, T. S. (1962). *The Structure of Scientific Revolutions*. University of Chicago Press.

Le-Trung, Q., Engelstad, P. E., Skeie, T., y Taherkordi, A. (2008). Load-balance of Intra/inter-MANET Traffic over Multiple Internet Gateways. En *Proceedings of the 6th international conference on advances in mobile computing and multimedia* (pp. 50–57). New York, NY, USA: ACM. Descargado de `http://doi.acm.org/10.1145/1497185.1497200` doi: 10.1145/1497185.1497200

Li, M., Zhao, L., Li, X., Li, X., Zaki, Y., Timm-Giel, A., y Gorg, C. (2012, 5). Investigation of Network Virtualization and Load Balancing Techniques in LTE Networks. En *2012 ieee 75th vehicular technology conference (vtc spring)* (pp. 1–5). doi: 10.1109/VETECS.2012.6240347

Liang, C., y Yu, F. R. (2015). Wireless Network Virtualization: A Survey, Some Research Issues and Challenges. *IEEE Communications Surveys Tutorials*, *17*(1), 358–380. doi: 10.1109/COMST.2014.2352118

Linux Foundation. (2021). *What Is Open vSwitch?* Descargado de `https://docs.openvswitch.org/en/latest/intro/what-is-ovs/`

Liskov, B. (2010). *Distributed Computing* (N. A. Lynch y A. A. Shvartsman, Eds.). Berlin, Heidelberg: Springer Berlin Heidelberg. doi: 10.1007/978-3-642-15763-9{_}2

Maheshwari, D., y Nedunchezhian, R. (2012). Load Balancing in Mobile Ad Hoc Networks: A Survey. *International Journal of Computer Applications*, *59*(16), 44–49.

Mcheick, H., Karawash, A., y Baba, T. (2011). Load Balancing Mathematical Model. En *2011 developments in e-systems engineering* (pp. 581–586). doi: 10.1109/DeSE.2011.63

Neumann, A., Aichele, C., Lindner, M., y Wunderlich, S. (2008). Better approach to mobile ad-hoc networking (BATMAN). En *Ietf draft* (pp. 1–24). IETF. Descargado de `https://tools.ietf.org/html/draft-openmesh-b-a-t-m-a-n-00`

Newman, M. E. J. (2003). The Structure and Function of Complex Networks. *SIAM Review*, *45*(2), 167–256. Descargado de `http://dx.doi.org/10.1137/S003614450342480` doi: 10.1137/S003614450342480

Nossum, R. (1987). *Advanced topics in artificial intelligence* (J. Siekmann, Ed.). Springer Verlag.

Noubir, G., Qian, W., Thapa, B., y Wang, Y. (2009, 3). Experimentation-oriented platform for development and evaluation of MANET cross-layer protocols. *Ad Hoc Networks*, *7*(2), 443–459. Descargado de `http://www.sciencedirect.com/science/article/pii/S1570870508000619` doi: 10.1016/j.adhoc.2008.05.002

Open-Mesh. (2020). *B.A.T.M.A.N. Advanced Documentation Overview.* Descargado de `https://www.open-mesh.org/projects/open-mesh/wiki`

Open Virtual Switch. (2016). *OVS Faucet Tutorial.* Descargado de `https://docs.openvswitch.org/en/latest/tutorials/faucet/`

Ospina-López, J. P., y Ortiz-Triviño, J. E. (2015). Characterization of a cluster and its resources in an Ad-Hoc network starting from the truncated geometric distribution. *ITECKNE: Innovación e Investigación en Ingeniería*, *12*(1), 68–75.

Perkins, C., Belding-Royer, E., y Das, S. (2003). *RFC3561: Ad Hoc On-Demand Distance Vector (AODV) Routing.* USA: RFC Editor.

Pham, P. P., y Perreau, S. (2003). Performance analysis of reactive shortest path and multipath routing mechanism with load balance. En *Ieee infocom 2003. twenty-second annual joint conference of the ieee computer and communications societies (ieee cat. no.03ch37428)* (Vol. 1, pp. 251–259). doi: 10.1109/INFCOM.2003.1208677

Qadir, J., Ahmed, N., y Ahad, N. (2014). Building programmable wireless networks: an architectural survey. *EURASIP Journal on Wireless Communications and Networking*, *2014*(1), 1–31. Descargado de `http://dx.doi.org/10.1186/1687-1499-2014-172` doi: 10.1186/1687-1499-2014-172

Rajamani, K., y Jeyaseelan, J. (2005). *Efficient partitioning of MAC (media access control) functions.* Google Patents. Descargado de `http://www.google.com.na/patents/US20050270993`

Raspberry Pi Foundation. (2020). *FAQs.* Descargado de `https://www.raspberrypi.org/documentation/faqs/`

Rawls, J. (1995). *Teoría de la justicia.* Fondo de Cultura Económica.

Rawls, J., y Kelly, E. (2001). *Justice as Fairness: A Restatement*. Harvard University Press.

Raychaudhuri, D., y Gerla, M. (2011). *Emerging Wireless Technologies and the Future Mobile Internet*. Cambridge University Press. Descargado de `https://books.google.com.co/books?id=NwjniHbu8OUC`

Robertazzi, T. (2000). *Computer Networks and Systems* (Third Edit ed.). Springer Verlag. doi: 10.1007/978-1-4612-1164-8

Russell, S. J., y Norvig, P. (2020). *Artificial Intelligence: A Modern Approach* (4th ed.). Prentice Hall.

RYU project team. (2020). *RYU SDN Framework* (1.0 ed.). Descargado de `https://book.ryu-sdn.org/en/Ryubook.pdf`

Sartre, J.-P. (1943). *El ser y la nada*.

Seither, D., König, A., y Hollick, M. (2011, 10). Routing performance of Wireless Mesh Networks: A practical evaluation of BATMAN advanced. En *2011 ieee 36th conference on local computer networks* (pp. 897–904). doi: 10.1109/LCN.2011.6115569

Sliwa, B., Falten, S., y Wietfeld, C. (2019). Performance Evaluation and Optimization of B.A.T.M.A.N. V Routing for Aerial and Ground-Based Mobile Ad-Hoc Networks. En *2019 ieee 89th vehicular technology conference (vtc2019-spring)* (pp. 1–7). doi: 10.1109/VTCSpring.2019.8746361

Spinoza, B. (1980). *Ética demostrada según el orden geométrico*. Fondo de Cultura Económica.

Tashtoush, Y., Darwish, O., y Hayajneh, M. (2014). Fibonacci sequence based multipath load balancing approach for mobile ad hoc networks. *Ad Hoc Networks*, *16*, 237–246. Descargado de `http://www.sciencedirect.com/science/article/pii/S1570870514000031` doi: https://doi.org/10.1016/j.adhoc.2013.12.015

Tlön, G. (2021). *TLÖN - Grupo de Investigación en Redes de Telecomunicaciones Dinámicas y Lenguajes de Programación Distribuidos.* Descargado de `http://tlon.unal.edu.co/`

T.M. Blessing, L., Chakrabarti, A., T.M. Blessing, L., y Chakrabarti, A. (2009). *DRM, a Design Research Methodology*. Springer.

Toumpis, S., y Gitzenis, S. (2009, 4). Load Balancing in Wireless Sensor Networks using Kirchhoff's Voltage Law. En *Ieee infocom 2009* (pp. 1656–1664). doi: 10.1109/INFCOM.2009.5062084

Triviño Cabrera, A., Casilari, E., Bartolomé, D., y Ariza, A. (2006). Traffic Load Distribution in Ad Hoc Networks through Mobile Internet Gateways. *Proceedings of Fourth International Working Conference on Performance Modelling and Evaluation of Heterogeneous Networks*, *1*.

Turner, N. M., J., T. A., H, B., G, P., L, P., J, R., y S, S. (2008). OpenFlow: enabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review*, *32*(2), 69–74.

Unión Internacional de Telecomunicaciones. (2005). *Manual de gestión nacional del espectro*.

Wang, W., Chen, Y., Zhang, Q., y Jiang, T. (2016). A software-defined wireless networking enabled spectrum management architecture. *IEEE Communications Magazine*, *54*(1), 33–39. doi: 10.1109/MCOM.2016.7378423

Wang, Y., Zhang, Y., y Chen, J. (2016). SDNPS: A Load-Balanced Topic-Based Publish/-Subscribe System in Software-Defined Networking. *Applied Sciences*, *6*(91), 1–21. doi: 10.3390/app6040091

Wen, H., Kumar, P., y Le-Ngoc, T. (2013). *Wireless Virtualization*. Springer.

Wu, D., Arkhipov, D. I., Asmare, E., Qin, Z., y McCann, J. A. (2015, 4). UbiFlow: Mobility management in urban-scale software defined IoT. En *2015 ieee conference on computer communications (infocom)* (pp. 208–216). doi: 10.1109/INFOCOM.2015.7218384

Yang, F., Gondi, V., Hallstrom, J. O., Wang, K. C., y Eidson, G. (2014). OpenFlow-based load balancing for wireless mesh infrastructure. En *2014 ieee 11th consumer communications and networking conference (ccnc)* (pp. 444–449). doi: 10.1109/C-CNC.2014.6866608

Yang, M., Li, Y., Jin, D., Zeng, L., Wu, X., y Vasilakos, A. V. (2015). Software-Defined and Virtualized Future Mobile and Wireless Networks: A Survey. *Mobile Networks and Applications*, *20*(1), 4–18. Descargado de `http://dx.doi.org/10.1007/s11036-014-0533-8` doi: 10.1007/s11036-014-0533-8

Yu, H. H. C., Quer, G., y Rao, R. R. R. (2017, 5). Wireless SDN mobile ad hoc network: From theory to practice. En *2017 ieee international conference on communications (icc)* (pp. 1–7). IEEE. Descargado de `http://ieeexplore.ieee.org/document/7996340/` doi: 10.1109/ICC.2017.7996340

Yu, J. Y., y Chong, P. H. J. (2005). A survey of clustering schemes for mobile ad hoc networks. *IEEE Communications Surveys Tutorials*, *7*(1), 32–48. doi: 10.1109/COMST.2005.1423333

Zaki, Y. (2013). *Future Mobile Communications LTE Optimization and Mobile Network Virtualization*. Springer Vieweg, Wiesbaden. doi: 10.1007/978-3-658-00808-6

Zaki, Y., Zhao, L., Goerg, C., y Timm-Giel, A. (2010, 10). LTE wireless virtualization and spectrum management. En *Wireless and mobile networking conference (wmnc), 2010 third joint ifip* (pp. 1–6). doi: 10.1109/WMNC.2010.5678740

Zaman, R. U., Khan, K. U. R., y Reddy, A. V. (2009, 12). A review of gateway load balancing strategies in Integrated Internet-MANET. En *Internet multimedia services architecture and applications (imsaa), 2009 ieee international conference on* (pp. 1–6). doi: 10.1109/IMSAA.2009.5439450

Zárate-Ceballos, H., y Ortiz-Triviño, J. E. (2020). S.O.V.O.R.A.: A Distributed Wireless Operating System. *Information*, *11*(12). Descargado de `https://www.mdpi.com/2078-2489/11/12/581` doi: 10.3390/info11120581

Zhao, J. H., Yang, X. Z., y Liu, H. W. (2005, 4). Load-balancing strategy of multi-gateway for ad hoc Internet connectivity. En *International conference on information*

technology: Coding and computing (itcc'05) - volume ii (Vol. 2, pp. 592–596). doi: 10.1109/ITCC.2005.184

Zhou, Q., Wang, C. X., McLaughlin, S., y Zhou, X. (2015). Network virtualization and resource description in software-defined wireless networks. *IEEE Communications Magazine*, *53*(11), 110–117. doi: 10.1109/MCOM.2015.7321979