ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

SIMULACIÓN DE UN MECANISMO DE CONTROL DE POTENCIA EN AODV EN FUNCIÓN DEL NÚMERO DE VECINOS USANDO CROSS LAYERING EN UNA MANET

TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN ELECTRÓNICA Y TELECOMUNICACIONES

LESLY YESENIA MAYGUA MARCILLO

lesye-20@hotmail.com

DIRECTOR: Dr. LUIS FELIPE URQUIZA AGUIAR

luis.urquiza@epn.edu.ec

CODIRECTOR: Dra. MARTHA CECILIA PAREDES PAREDES

cecilia.paredes@epn.edu.ec

Quito, diciembre 2017

AVAL

Certificamos que el presente trabajo fue desarrollado por Lesly Yesenia Maygua Marcillo, bajo nuestra supervisión.

Dr. Luis Felipe Urquiza Aguiar
DIRECTOR DEL TRABAJO DE TITULACIÓN

Dra. Martha Cecilia Paredes Paredes
CODIRECTOR DEL TRABAJO DE TITULACIÓN

DECLARACIÓN DE AUTORÍA

Yo Lesly Yesenia Maygua Marcillo, declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentada para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración cedo mis derechos de propiedad intelectual correspondientes a este trabajo, a la Escuela Politécnica Nacional, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normatividad institucional vigente.

LESLY YESENIA MAYGUA MARCILLO

DEDICATORIA

Este trabajo está dedicado, a mis padres Marty y Manuel, quienes han sido mi mayor inspiración y fortaleza en este arduo camino.

AGRADECIMIENTO

A mi Dios, por todas sus bendiciones. Por brindarme sabiduría y fortaleza cuando más lo necesité.

Queridos papá y mamá, quiero agradecerles por su enorme sacrificio para brindarme todas las facilidades en mi vida, por su paciencia, cariño y sobre todo por creer en mí, por ayudarme a progresar y hacer de mí una mejor persona. Este trabajo es de Uds. Gracias papi por esa motivación continua y por sus sabias palabras. Gracias mamita por ser incondicional. Los amo.

Un sentido agradecimiento para mi director de tesis, Dr. Luis Felipe, quien con mucho afán siempre estuvo dispuesto a guiarme y transmitirme sus conocimientos a lo largo de este proyecto. Por su tiempo y su paciencia inigualable, gracias Dr.

A la Escuela Politécnica Nacional, por darme la oportunidad de tener unos excelentes maestros y abrirme las puertas al conocimiento.

A mis amigos, gracias por su apoyo. Y a ti Dino, por siempre creer en mí y motivarme a ser mejor.

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

El presente proyecto de titulación se enfoca en el estudio de las redes inalámbricas, específicamente las MANETs (*Mobile Ad-hoc NETwork*) con la finalidad de diseñar un mecanismo de control de potencia en el protocolo de enrutamiento AODV en base a la detección de vecinos usando una técnica de Cross Layering. Con este método se prevé incrementar el número de transmisiones exitosas, reducir colisiones, usar solo la energía necesaria, así como mantener o mejorar el rendimiento de AODV y con ello el de la red. El mecanismo de control de potencia se desarrolla en el simulador de red NS-3.

Para detallar el proceso del presente proyecto se consideran varias etapas dentro de la metodología expuesta en la sección 2. La primera etapa hace referencia al funcionamiento del simulador NS-3, su proceso de instalación, librerías, clases y módulos necesarios para el desarrollo del presente trabajo. La segunda etapa detalla el arduo trabajo que se realizó para implementar el control de potencia; incluyendo el acceso a capa física desde AODV, uso de punteros, generación de funciones y modificación del protocolo de enrutamiento. Se detalla además el mecanismo que controla la potencia, sus consideraciones, parámetros y dependencias.

Como tercera etapa de la metodología se muestran los escenarios usados para las simulaciones, su desarrollo; modelos de movilidad implementados y generación de trazas tanto en NS-3 como con el software Bonnmotion. La última etapa incluye la generación de dos archivos, los cuales contienen los resultados de las simulaciones. Estos archivos incluyen información de los paquetes y de la potencia de transmisión para determinar el número de paquetes transmitidos, recibidos, desechados y reenviados y analizar la variación de potencia respectivamente.

En la tercera sección se muestra los resultados obtenidos de cuatro tipos de pruebas realizadas para validar la acción del mecanismo de control de potencia. Se incluye también, un análisis comparativo para evaluar las variaciones que se tiene del uso o no del mecanismo de control de potencia donde se puede determinar que el control de potencia es capaz de mejorar el rendimiento y sobrecarga de la red; así como reducir el consumo de energía. Finalmente se presentan las conclusiones y anexos.

PALABRAS CLAVE: MANET, potencia, traza, fichero, mecanismo.

ABSTRACT

The present Project is focused in the study of the wireless networks, specifically in the MANET (*Mobile Ad-hoc NETwork*) with the purpose of designing a power control mechanism in the AODV routing protocol by means of a Cross Layering technique. With this method we aim to increase the number of successful transmissions, reduce collisions, use only the necessary energy, as well as to maintain or improve the performance of AODV and therefore the network performance. This project is developed in the NS-3 network simulator.

In order to describe the process followed in this final project we consider several stages in the methodology, described in section two. The first stage refers to the operation of the NS-3 simulator, its installation process, libraries, classes and modules useful for this work. The second stage provides details about the hard work done to achieve power control implementation. This including access to physical layer, use of pointers, generation of functions and modification of the routing protocol in the network simulator. Also this step describe the logic of the power control mechanism.

The third stage of methodology shows the scenarios for the simulations, mobility models considered for the tests and generation of traces in NS-3 as well as in Bonnmotion. The last stage includes the generation of logs, which contain the results of the simulations. These files allow to analyze the transmitted packets and the power variation.

In the third section we show the results obtained from four types of tests to validate the action of the power control mechanism. It also includes a comparative analysis to assess the performance and overhead differences when the power control mechanism is activated or not. Finally, the conclusions and annexes are presented.

KEYWORDS: MANET, power, trace, log, mechanism.

1. INTRODUCCIÓN

Las redes inalámbricas son tecnologías con creciente interés y amplias expectativas en el área de las telecomunicaciones, tal es el caso de las MANETs (*Mobile Ad-hoc NETwork*) que gracias a sus cualidades han tenido un amplio desarrollo en áreas como: ciudades inteligentes, entornos de recuperación de desastres, redes vehiculares, redes de sensores inalámbricos, entre otros.

Pese a sus ventajas, el hecho de actuar sobre un canal inalámbrico compartido con condiciones inestables, ancho de banda limitado, altas tasas de error, mayor latencia, restricción de consumo de energía, topologías variables debido a la movilidad en los nodos hace que se presenten varios desafíos en la transmisión de la información. La potencia de transmisión es uno de ellos, parámetro importante ya que influye al momento de determinar si una transmisión es exitosa o no, por lo que un control de potencia sería necesario. Si la potencia es muy alta, la detección de vecinos será muy amplia y se darían saturaciones de canal, fuerte contención y por ende colisiones. Por el contrario, si se actúa con una potencia muy baja podrían detectarse muy pocos vecinos o ninguno lo cual llevaría a una transmisión fallida.

Al controlar potencia se obtienen ventajas como: incrementar el número de transmisiones exitosas, usar solo la energía necesaria en los nodos, evitaremos saturaciones de canal y colisiones.

El control de potencia en una MANET debería relacionarse a la operación de las capas superiores, ya que al interactuar entre capas se tendría mayor eficiencia en la red, sin embargo, este enfoque no ha sido muy considerado para el diseño de las redes ad-hoc.

El presente estudio propone agregar un mecanismo para controlar la potencia de transmisión en base a la detección de vecinos del protocolo AODV (*Ad-hoc On-Demand Distance Vector*) usando un criterio de Cross Layering (Capas cruzadas) para permitir coordinación, interacción e intercambio de información entre capa física y capa de red.

Para esto se usará el simulador de red NS-3, el cual permite simular los diferentes escenarios de redes móviles ad-hoc, modificar el protocolo de enrutamiento, diseñar el mecanismo que controla la potencia de transmisión y generar los respectivos archivos de traza. Se usa conjuntamente GAWK para el análisis de una parte de los resultados.

Este trabajo determina qué tanto puede variar el rendimiento del protocolo AODV y por ende de la red, si opera con un mecanismo de control de potencia considerando una colaboración entre capa física y de red.

1.1 Objetivos

El objetivo general de este estudio técnico es simular un mecanismo de control de potencia en función del número de vecinos en AODV usando una técnica de cross layering en una MANET con NS-3.

Los objetivos específicos de este estudio técnico son:

- Acceder a capa física desde el protocolo AODV (capa de red) para modificar la potencia.
- Diseñar el mecanismo de control de potencia.
- Generar una función en el protocolo AODV que determine el número de vecinos de cada nodo para usarla en el mecanismo de control de potencia.
- Realizar un análisis comparativo del rendimiento de AODV con y sin control de potencia.

1.2 Alcance

Se implementará un mecanismo que permita controlar potencia en una red Ad-hoc móvil (MANET) con una técnica de colaboración entre capas. El mecanismo controlará potencia en base al número de vecinos que proporciona el protocolo de enrutamiento AODV. Se realizará el control de potencia sobre capa física desde el protocolo de enrutamiento. El trabajo se lo realizará en el simulador de red NS-3 v.3.25 en la distribución para Linux Ubuntu 14.04.1.

Será necesario familiarizarse a fondo con el simulador de red y analizar los escenarios inalámbricos incluidos en la distribución del simulador, así se podrá determinar cómo se crea y funciona una red ad-hoc, como actúan las funciones de capa física, mac, red, y del protocolo AODV, con el fin de acceder a capa física desde el protocolo de enrutamiento y controlar potencia.

Se harán pruebas para verificar que efectivamente se modificó la potencia. Una vez que logremos esto, iniciaremos el análisis en el protocolo de enrutamiento AODV que define NS-3 para obtener la información más relevante y ligada a los vecinos que detecta. Con esto se procederá al diseño del mecanismo que va a controlar potencia.

Una vez definido el mecanismo se lo probará en un escenario de una red ad-hoc móvil. Se empleará el software Bonnmotion para generar una traza de movilidad en el escenario.

Finalmente se realizarán varias pruebas en diferentes escenarios para validar el mecanismo de control de potencia y se analizarán los resultados.

1.3 Marco Teórico

Con la proliferación de terminales inalámbricos portátiles, las redes ad-hoc han logrado un amplio interés en los últimos años como un método para proporcionar comunicaciones de datos sin el apoyo de una infraestructura gracias a que sus nodos pueden operar como hosts y como enrutadores, dando lugar a numerosas aplicaciones en el área de las comunicaciones inalámbricas. Pese a las ventajas que poseen estas redes, es claro que se enfrentan a varios desafíos en la transmisión debido al medio inalámbrico en el cual se manejan, por ejemplo, la movilidad en los nodos y la potencia de transmisión que juegan un papel importante en el desempeño de los protocolos de enrutamiento[1]. Más adelante se abordará el tema de la potencia de transmisión.

1.3.1 MANET (*Mobile Ad-hoc NETwork*),

Una MANET [2] es una red que define un conjunto de nodos móviles que pueden organizarse esporádicamente en topologías de red arbitrarias permitiendo a personas y dispositivos interconectarse sin ningún tipo de infraestructura preexistente formando una red independiente (Figura 1.1), o para conectarse con algún escenario de red exterior. Los nodos móviles participantes tienen un rango de transmisión limitado por lo que para alcanzar su destino puede tener múltiples saltos. Se comunican mediante tecnologías inalámbricas como: Zigbee, Bluetooth, WiMAX, Wi-Fi [3].

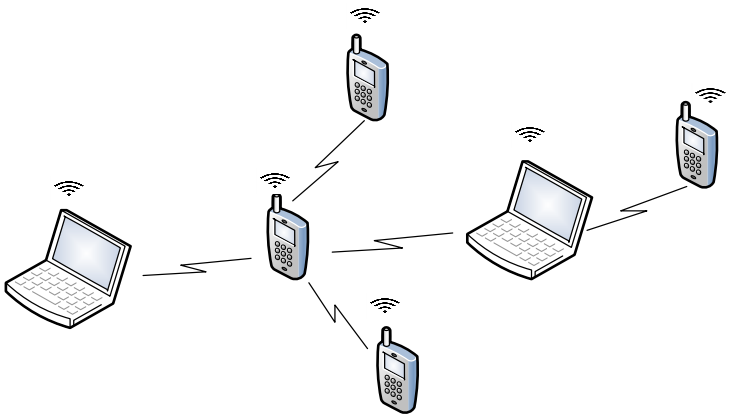

Figura 1.1. Escenario de una MANET. Figura adaptada de [3]

Las tecnologías que se emplean a nivel físico para estas redes principalmente son [4]:

- Spread Spectrum
- OFDM
- Infrarrojos

Entre sus características más notables están: bajo costo, flexibilidad, redes descentralizadas, facilidad de despliegue, auto configurables, fácil administración; gracias a esto han sido útiles en entornos de recuperación de desastres, internet de las cosas, ciudades inteligentes, redes militares, entre otros.

A pesar de ser redes muy versátiles, las MANETs difícilmente logran calidad de servicio [3] ya que se ven afectadas por la naturaleza variable e impredecible del canal inalámbrico y entornos de red como: canales compartidos, ancho de banda limitado, altas tasas de error, alta latencia, restricción de consumo de energía, topología dinámica. Todos estos factores incrementan además la vulnerabilidad de la red frente a ataques.

1.3.2 Tipos de MANETs

Para el presente trabajo se considerará una MANET clásica; sin embargo, existen algunos tipos de redes ad-hoc móviles [5][6]:

- iMANET (Internet Based Mobile Ad-hoc Network), [7] combina una MANET con el internet en la cual los nodos móviles pueden conectarse a internet directamente a través de un AP (Access Point) Figura 1.2, o indirectamente a través de otros terminales móviles. Estas redes son adecuadas para muchas aplicaciones de Internet en términos de fácil acceso y disponibilidad de información, sin embargo, se puede evidenciar frecuentes desconexiones del enlace debido a los cambios en la topología, por la movilidad en los nodos.

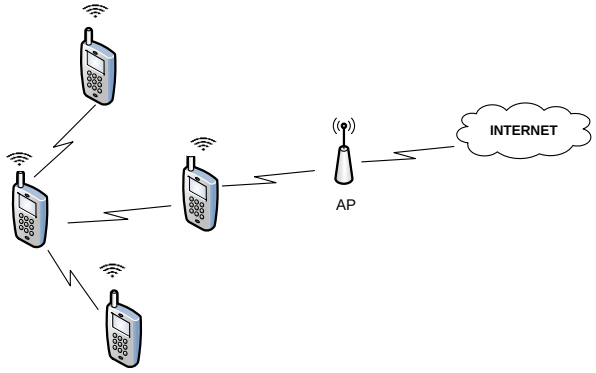

Figura 1.2. Escenario IMANET. Figura adaptada de [3].

Un ejemplo de su aplicación [8] es en juegos multijugador en el cual cada usuario puede conectarse mediante Bluetooth, wifi o 3G y actuar como diferentes fuentes de datos.

- VANET (Vehicular Ad-hoc Network), [9] son redes que permiten una comunicación ad-hoc entre vehículos e infraestructuras de transmisión ubicadas a lo largo de las carreteras denominadas RSUs (Roads Side Units) [10] para intercambiar información de forma autónoma, véase Figura 1.3.
  Una VANET establece dos tipos de comunicación, V2V (Vehicle to Vehicle) y V2I (Vehicle to Infraestructure), para esto cada vehículo se equipa con un OBU (On-Board Unit) el cual permite uno de estos tipos de comunicación [11].

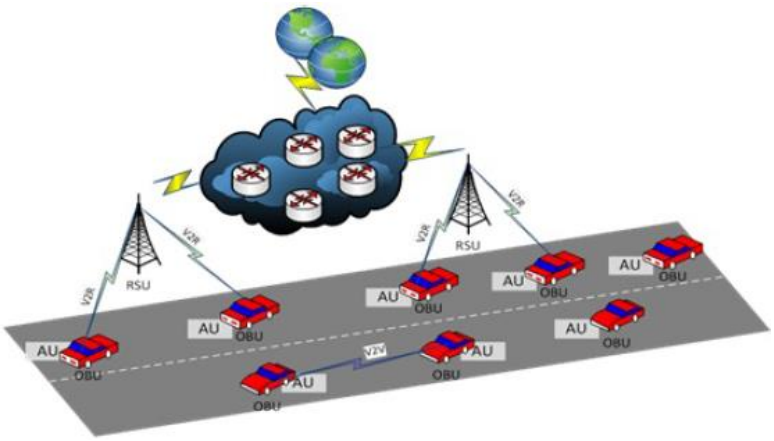

Figura 1.3. Arquitectura de una VANET. [12].

Una extensión de este tipo de redes es la InVANET (Intelligent Vehicular Ad-hoc Network) [5], que incorporan inteligencia artificial para que los vehículos se comporten de una manera inteligente frente a colisiones o accidentes.

- FANET (Flying Ad-hoc Network), es un grupo de UAVs (Unmanned Aereal Vehicle) que se comunican entre sí sin necesidad de un punto de acceso; sin embargo, al menos uno de ellos debe estar conectado a un satélite o base en tierra. [13] Los vehículos aéreos no tripulados operan de forma autónoma o bajo control remoto. Estas redes son usadas en aplicaciones civiles y militares, por ejemplo, para monitorear desastres [14] o incendios forestales [15].

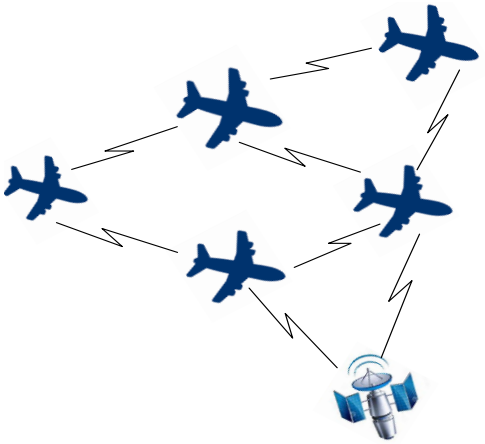

Figura 1.4. Escenario FANET. Figura adaptada de [13]

1.3.3 Potencia de transmisión en redes inalámbricas Ad-hoc y su impacto

Como se indicó en un inicio, la potencia de transmisión es uno de los desafíos a los que se enfrenta una red ad-hoc, ya que influye en aspectos como [16]:

- *Rango de transmisión de los nodos; intensidad de señal en el receptor:* Incrementar el nivel de potencia de transmisión puede aumentar el rango de transmisión de nodos, lo que reduce el número promedio de saltos en cada ruta, Figura 1.5. En este caso, el retardo de transmisión total a lo largo de cada ruta disminuye ya que, al no tener saltos intermedios, no se tiene tiempo de reenvío. Un mayor nivel de potencia además mejora la calidad de señal en el receptor reduciendo la tasa de error de bits de los paquetes (BER) en el caso de no considerar interferencias y, por tanto, evita el retardo adicional generado por las retransmisiones de la capa de enlace.

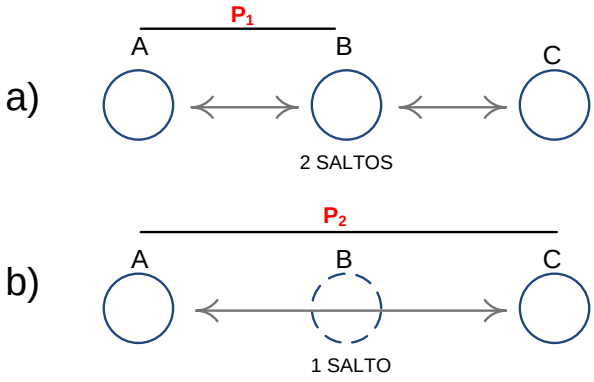

Figura 1.5. Rango de transmisión, P_1 y P_2 son potencias $P_2>P_1$.

En la parte a) podemos apreciar como el nodo A solo puede alcanzar hasta el nodo B con una potencia de transmisión P_1 y por tal, si A quiere llegar hasta C necesita de dos saltos. En b) se muestra como al usar una mayor potencia "P_2" el nodo A puede alcanzar hasta el nodo C sin necesidad de usar a B como intermediario teniendo así un solo salto.

- *Interferencia originada para otros nodos receptores*. Una potencia muy elevada crearía altos niveles de interferencia en los nodos.

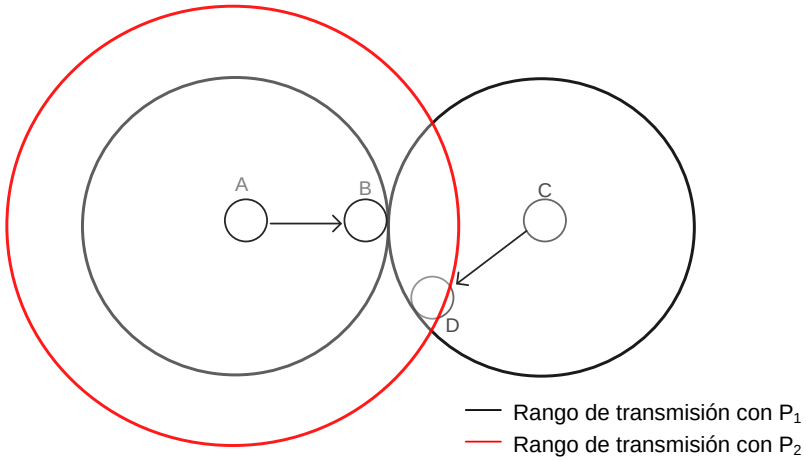

Figura 1.6. Interferencia originada para nodos receptores vecinos.

En la Figura 1.6 se puede observar como A y C están transmitiendo con una potencia P_1, en el caso que A incremente su potencia a P_2 estará creando interferencia a la transmisión que C tiene en curso.

- *Rendimiento y eficiencia energética de la red* [17] [18] [19]; un menor nivel de potencia puede reducir el consumo de energía en la comunicación lo cual incrementa el reuso del medio inalámbrico y mejoraría el throughput de la red.

- Determina factores que influyen notablemente en la operación de las capas: MAC, red y transporte [16] [20].

  - Capa MAC: En capa MAC puede determinar la región de contención y por ende el nivel de congestión del medio inalámbrico. La potencia determina la transmisión y el rango de censado de portadora del nodo emisor. Algún nodo dentro del rango de transmisión puede decodificar la trama exitosamente, y algún nodo puede detectar esta transmisión; por lo tanto, para los protocolos MAC de acceso

aleatorio basados en contención, cualquier nodo vecino debe ser silenciado cuando se está recibiendo paquetes desde otro nodo. Afecta además el nivel de interferencia que los nodos crean para sus vecinos, es decir influye en el grado de transmisiones simultáneas permitidas. La topología de la red inalámbrica también se vería afectada.

- Capa de red: El protocolo de enrutamiento se vería afectado ya que éste determina el conjunto de nodos opcionados para la selección del siguiente salto en el descubrimiento de ruta.

- Capa de transporte: Los mecanismos de control de congestión, como los del TCP, regulan las tasas de emisión permitidas. El control de potencia de transmisión determina las capacidades de enlace disponibles, por lo que afectan a la operación del control de congestión.

Se han realizado varios estudios [20] para mejorar el rendimiento de la red en cuanto a throughput, retardos y eficiencia energética basándose en un control de la potencia de transmisión, algunos de ellos se refieren a un control de potencia con protocolos MAC (esquemas PCM [21], [22], PCMA [23], etc.), sin embargo, éstos sólo consideran un control de potencia en una sola capa; lo cual no puede conducir necesariamente a la mejora del rendimiento de la red.

1.3.4 Control de potencia enfocado a una sola capa

El control de potencia de transmisión que generalmente se ha usado en las redes inalámbricas ad-hoc ha seguido un enfoque a una sola capa [16]; es decir, sólo se considera las interacciones entre el control de potencia y una capa única de la pila de protocolos. A continuación, se muestra el problema de control de potencia de trasmisión en capa MAC y red.

- Problema en capa MAC

Se enfoca en enviar paquetes en la mínima potencia necesaria tal que el SINR en el receptor esté apenas por encima del umbral predefinido para una transmisión exitosa, esto conduce a la mejora del reúso del medio inalámbrico y ahorro de energía en los nodos, razón por la cual los protocolos MAC han sido intensamente estudiados para control de potencia. La idea básica es que los nodos intercambien sus paquetes de

control, por ejemplo: paquetes RTS / CTS en el máximo nivel de potencia permisible y enviar sus paquetes de datos y ACK al mínimo nivel de potencia necesario para una comunicación fiable.

Esto significa que el enfoque a capa MAC para hacer frente al problema de control de potencia de transmisión sólo nos lleva a una optimización de esa capa, y por lo tanto para conseguir la optimización global de la red es preciso añadir la funcionalidad del control de potencia de transmisión en capa de red o superiores.

- Problema en capa de red

Los nodos receptores previstos en la transmisión son determinados por la capa de red mediante protocolos de enrutamiento.

El trabajo de capas más bajas (MAC y física) se centra en enviar el paquete al destino especificado por la capa de red. Por lo tanto, implementar un control de potencia en capa MAC no permite a los protocolos de enrutamiento elegir el nodo óptimo para el siguiente salto.

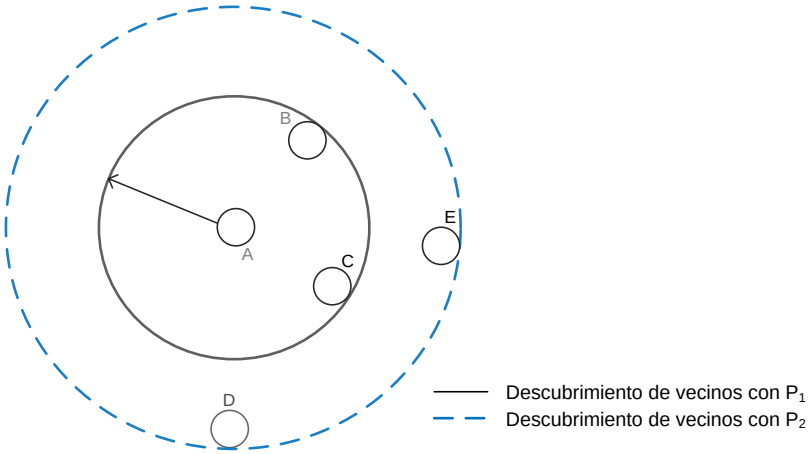

Figura 1.7. Descubrimiento de vecinos según el nivel de potencia. Donde $P_2 > P_1$.

La potencia de transmisión determina los nodos vecinos con los que un nodo puede comunicarse, véase Figura 1.7, y por lo tanto afecta a la selección del siguiente salto en los protocolos de enrutamiento.

La propuesta inicial de los protocolos de enrutamiento para una MANET (AODV [24] [25], DSR [26]) sólo se encargaba de descubrir las rutas disponibles entre origen y destino; no consideraron el consumo de energía de los nodos en la red, lo cual es crítico para nodos que operan con batería. Esto promueve el diseño de protocolos de enrutamiento

conscientes del consumo de energía en redes inalámbricas ad-hoc. Hasta ahora, la mayoría de los sistemas propuestos utilizan la mínima potencia de transmisión necesaria para una comunicación confiable en cada salto, y usan ese nivel de energía como métrica del enlace para la selección del siguiente salto.

1.3.5 Consideraciones para el diseño de control de potencia

Debe abordarse considerando problemas a largo plazo sobre la arquitectura de red. Kawadia y Kumar [27] enumeran varios principios que guían el diseño de algoritmos de control de potencia de transmisión.

- *Efecto sobre la capacidad y el rendimiento*. Debido a la naturaleza del medio, la potencia de transmisión causa interferencia en la región circundante. Si se reduce la potencia de transmisión puede reducirse el área de interferencia. Sin embargo, una potencia más baja resultaría en enlaces más cortos, por ende, más saltos para una ruta y esto generaría más carga de retransmisión a los nodos.

  Se debe tener claro que para una transmisión de un rango R, el área de la interferencia es proporcional a R^2, mientras que la carga de retransmisión (número de saltos) es inversamente proporcional a R, lo que implica que reducir el nivel de potencia de transmisión puede aumentar la capacidad de la red [28] .

- *Efecto de la carga de retransmisión en el control de potencia* [29]: El tiempo de retardo extremo a extremo es la suma del retraso de transmisión, propagación del retardo, retardo de procesamiento y retraso en cola. El retardo por propagación es proporcional a la distancia entre origen y destino, generalmente insignificante. El tiempo de procesamiento y transmisión es proporcional al número de saltos existentes entre origen y destino. El tiempo que un paquete espera para la transmisión cuando el canal está ocupado, es el tiempo de cola, lo cual depende de la congestión que maneje el canal inalámbrico.

  Cuando la carga en la red es baja, el retardo de cola es muy pequeño ya que los nodos que compiten para el acceso de canal son pocos Figura 1.8 a. En este caso el rendimiento de la red inalámbrica se puede mejorar aumentando el nivel de potencia de transmisión el cual reduce el número de saltos a una ruta.

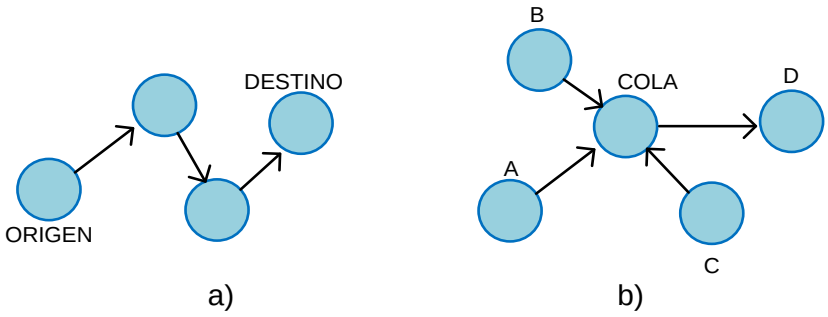

Figura 1.8. Transmisión con y sin competencia de medio.

Si la red está muy cargada, varios serán los nodos que estén compitiendo por un canal y el retraso de cola será muy influyente en el retraso extremo a extremo, Figura 1.8 b. En este caso reducir el nivel de potencia de transmisión aliviaría la congestión. El retardo de extremo a extremo es menor al disminuir el retardo de cola. Por tanto, para un mejor rendimiento de la red, el nivel de potencia de transmisión en cada nodo debe adaptarse según la carga existente en la red.

Sin embargo, estos aspectos no se consideraron del todo en el diseño de redes ad-hoc, por lo que el problema del control de potencia de transmisión puede tomarse como un error en el diseño de estas redes al no incluir tampoco un esquema de capas cruzadas, tema del que se habla a continuación.

1.3.6 Cross layering, diseño de capas cruzadas

El desarrollo de las redes inalámbricas ad-hoc implica nuevos enfoques en el diseño de los protocolos de comunicación, los cuales deberán adaptarse a las características del entorno de red inalámbrico como: canales compartidos e inestables, ancho de banda limitado, altas tasas de error, topología dinámica debido a la movilidad, etc. [30], [31]. El estricto diseño en capas de las arquitecturas de red que típicamente se emplean no es lo suficientemente flexible como para hacer frente a la naturaleza inalámbrica en los enlaces de transmisión; la falta de coordinación entre capas limita el rendimiento de las arquitecturas; y es claro que optimizar cada capa del modelo ISO/OSI por separado no da los mejores resultados [32].

Para superar dichas limitaciones, se ha propuesto el diseño de capas cruzadas donde el objetivo es mantener las funcionalidades originales de cada capa y además permitir coordinación, interacción e intercambio de información inherente a cada capa para ajustar

los parámetros según el estado de la red [33] y así tener una optimización conjunta de los protocolos.

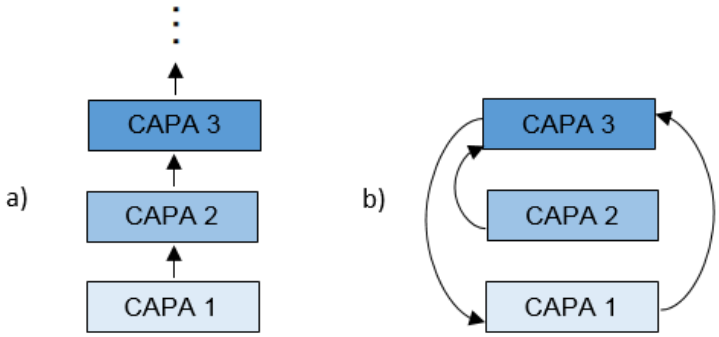

Figura 1.9. a) Arquitectura con diseño de capas plano b) Cross Layering

En la Figura 1.9 se puede observar como en una arquitectura de diseño convencional cada capa ofrece servicios únicamente a su capa superior adyacente, mientras que en la parte b podemos apreciar como las capas interactúan entre sí para tomar decisiones (Cross layering).

En base a los trabajos relacionados se pueden definir dos tipos de cross layering [31]:

- *Cross layering débil:* Permite la interacción entre parámetros de diferentes capas de la pila de protocolos, pueden ser interacciones no adyacentes.

- *Cross-layering fuerte:* Permite el diseño conjunto de los algoritmos implementados dentro de cualquier entidad en cualquier capa de la pila de protocolos; las características individuales relacionadas con las diferentes capas pueden perderse debido a la optimización. Este diseño puede proporcionar un mayor rendimiento reduciendo las posibilidades de despliegue e incrementando costo y complejidad.

Varios son los enfoques que Cross Layering tiene para optimizar el rendimiento de una red, algunos de ellos han sido propuestos en [34]; sin embargo, en este trabajo nos enfocaremos en conseguir un control de la potencia de transmisión en base a cross layering débil considerando una modificación en el protocolo de enrutamiento AODV para interactuar con capa física y así obtener el balance de potencia requerido. A continuación, se describen los protocolos de enrutamiento para redes ad-hoc y con ello el protocolo de interés, AODV.

1.3.7 Protocolos de enrutamiento para redes Ad-hoc

Debido a la movilidad que presentan los nodos en una MANET, los enlaces entre ellos pueden formarse y terminar con mucha frecuencia, lo cual demanda que la red sea auto-organizada y auto-configurable [35]. El camino entre origen y destino puede modificarse repentinamente, y, cuando esto sucede, la red debe ser capaz de encontrar una nueva ruta en el menor tiempo posible y todos los nodos participantes deben cooperar para lograrlo. [4] Esta funcionalidad requiere que cada nodo sea capaz de reenviar datos en nombre de otros nodos, es decir, cada nodo debe tener la capacidad de un router. Para lograr tal comunicación es necesario un protocolo de enrutamiento con algoritmos adecuados que consideren la característica dinámica de los nodos [36], [37]. Sin embargo, los recursos limitados de las MANETs han hecho que el diseño eficiente y confiable de un protocolo de enrutamiento sea todo un reto ya que estos deben incluir una estrategia inteligente de enrutamiento para usar de forma eficaz los recursos limitados y al mismo tiempo, adaptarse a los cambios arbitrarios de red.

Antes que crezca el interés por las redes inalámbricas, las redes cableadas utilizaron dos algoritmos principales:

- *Vector distancia:* [38] Las redes que usan vector distancia publican sus rutas como vectores de dirección y distancia; la primera es el router del siguiente salto o la interfaz de salida y la segunda es definida en base a una métrica que puede ser el número de saltos. Usan el algoritmo Bellman-Ford [39] para determinar la mejor ruta. Algunos protocolos por vector de distancia envían en forma periódica tablas de enrutamiento completas a todos los vecinos conectados por lo que, en redes extensas, estas actualizaciones pueden generar un tráfico considerable en los enlaces.

  El algoritmo no permite que un router conozca la topología exacta de una red; solo conoce la información de enrutamiento notificada por sus vecinos, es decir distancia o métrica para llegar a la red destino y qué ruta o interfaz usar para alcanzarla. Los protocolos basados en vector distancia funcionan mejor en situaciones donde: la red es simple y plana y no se requiere de un diseño jerárquico.

- *Estado de enlace* [40], [41] : En este caso, cada nodo mantiene información actualizada de la red mediante la difusión periódica del costo del enlace a sus nodos vecinos y a todos los otros nodos usando una estrategia de inundación. Cuando

cada nodo recibe un paquete de actualización, este actualiza su vista de la red y su información de estado de enlace mediante la aplicación de un algoritmo de ruta más corta para elegir el siguiente salto para cada destino.

Estos algoritmos tradicionales no son útiles en MANETs [37], puesto que: las actualizaciones periódicas de la red pueden consumir una parte significativa del ancho de banda disponible, aumentaría la contención de canales y además requeriría que cada nodo recargue su fuente de alimentación.

Para superar los inconvenientes asociados con los algoritmos mencionados, se han propuesto una serie de protocolos de enrutamiento para MANETs, los cuales pueden clasificarse según la manera en cómo fue obtenida y mantenida la información de ruteo por los nodos, de esta forma los protocolos de enrutamiento para una red ad-hoc móvil pueden dividirse en tres grupos diferentes, véase Figura 1.10. Y pueden ser vistos como una variación de los anteriores.

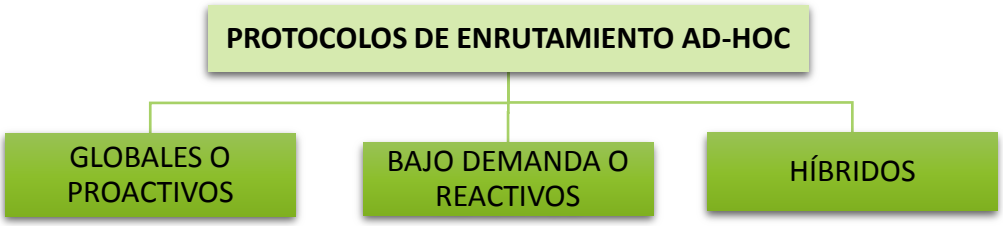

Figura 1.10. Clasificación de los protocolos de enrutamiento en redes ad-hoc [35].

- **Protocolos proactivos:** Son una variación de los protocolos por estado de enlace; conocidos también como protocolos accionados por tabla [41] determinan sus rutas hacia todos los destinos al inicio y mantienen la información de direccionamiento constantemente actualizada a través de un proceso periódico de intercambio de paquetes a intervalos de tiempo fijo, lo cual permite disponibilidad de direccionamiento a cada petición con el inconveniente de generar tráfico de señalización incluso cuando no se transmita ningún paquete de datos, llevando así a una sobrecarga de la red. Cada nodo posee una tabla de enrutamiento individual. Ejemplos de este tipo de protocolo son: OLSR [42] y DSDV [43].

- **Protocolos reactivos:** Derivan de los protocolos por vector distancia, son protocolos bajo demanda; [44] manejan un proceso de descubrimiento de rutas para determinar el correcto direccionamiento sólo en el momento en el que un paquete deba efectivamente transmitirse, es decir cuando sea requerido por el nodo origen [37]. A

esto se debe que se los conoce como protocolos bajo demanda. De esto modo, se reduce el tráfico de señalización innecesario lo cual evitará que aumenten los tiempos de entrega. AODV [24] [25] y DSR [45] son parte de este grupo de protocolos.

- **Protocolos híbridos:** Los de tipo híbrido combinan las características básicas de los dos tipos de protocolos precedentes en uno. [4] Limitan la aplicación de algoritmos proactivos sólo a los nodos adyacentes del que requiere transmitir. Es decir, al inicio del encaminamiento se determina la ruta de forma proactiva y a lo largo de la red se usan los protocolos de tipo reactivo. Un ejemplo de este tipo es el protocolo TORA [46].

Este trabajo tiene como interés el protocolo AODV, mismo que se detalla a continuación.

1.3.8 AODV (Ad-hoc On-Demand Distance Vector)

AODV es el protocolo de enrutamiento más usado para MANETs. Está definido en el RFC 3561 [47]. Es un protocolo de tipo reactivo que considera una variación del algoritmo de Vector Distancia Bellman-Ford [39] adaptado para ambientes móviles.
Determina la ruta correcta hacia el destino solo en caso de que un nodo quiera efectivamente transmitir un paquete; de este modo, se reduce el tráfico de señalización y disminuye el tiempo de entrega [25].
Las rutas se mantienen el tiempo que sea requerido por la fuente, siempre que haya conectividad entre los nodos del trayecto. Para tener rutas actualizadas y libres de lazos (incluso mientras se reparan enlaces rotos); el nodo destino genera un número de secuencia (*Destination Sequence Number*) antes de proporcionar información de direccionamiento*, el cual permite a los nodos* evaluar cuanto se ha actualizado cierto recorrido evitando así los lazos en la ruta. El destino seleccionado será aquel que contenga el número de secuencia más alto, mismo que corresponderá a una información de enrutamiento más actualizada [4].
En AODV cada nodo posee una tabla de enrutamiento, un registro por destino. Cada registro contiene: dirección destino, siguiente salto al destino, número de secuencia generado por el destino y el tiempo de vida [48].
Este protocolo usa enlaces simétricos entre nodos vecinos, es decir, no intenta seguir caminos entre nodos cuando uno de ellos no puede escuchar el otro; además la demanda sobre el ancho de banda total disponible para los nodos móviles es

sustancialmente menor que en aquellos protocolos que requieren anuncios periódicos de las rutas [27].

- **Operación del protocolo**

AODV maneja dos procesos para su funcionamiento; primero realiza un descubrimiento de ruta y luego trabaja para mantener la misma.

a) **Descubrimiento de ruta:** Toma lugar cuando no existe una ruta para cierto destino, esto puede ser cuando un nodo nunca antes envió información a cierto destino o porque la ruta expiró (la actualización no es garantía). Para este proceso AODV usa dos mensajes: RREQ (Route Request) y RREP (Route Replay) [4], [25].

- **RREQ:** Es un mensaje de broadcast enviado por el origen, el cual busca establecer un enlace con un destino. El RREQ es recibido por los nodos cercanos a la fuente y éstos a su vez retransmiten el mensaje hasta alcanzar el nodo destino o responder con la nueva ruta, véase Figura 1.11.

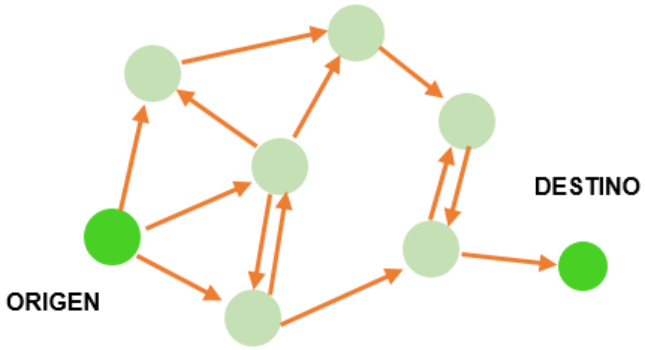

Figura 1.11. Mensaje de broadcast RREQ. Figura adaptada de [4].

Este mensaje incluye los campos: dirección IP del nodo origen y destino, números de secuencia y ID de difusión [25].

Dependiendo de la información transportada por el mensaje RREQ, los nodos que reciben este mensaje actualizan su información del origen en las tablas de enrutamiento. Los nodos analizan la dirección IP de origen y el ID de difusión de los RREQ, si ellos reciben un RREQ que ya han procesado, descartan el RREQ y no lo retransmiten para evitar difusiones innecesarias en la red. El nodo origen puede configurar un número máximo de saltos para construir una ruta.

- **RREP:** Es un mensaje de unicast enviado desde el destino al origen mediante el cual el origen recibe una respuesta ya sea del destinatario o de un nodo que posee una ruta reciente hacia aquel destino, confirmando que el camino está disponible. Figura 1.12.

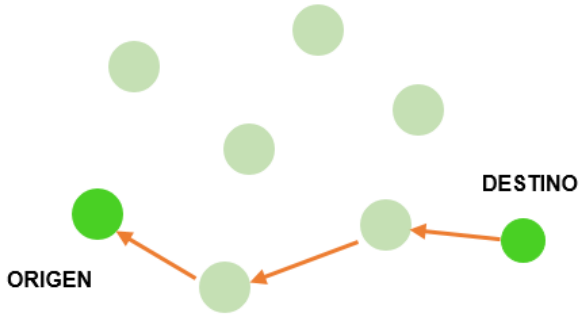

Figura 1.12. Mensaje de unicast RREP. Figura adaptada de [4].

La fuente actualiza su información de enrutamiento en la tabla para ese destino en caso de recibir un mensaje RREP con un número de secuencia mayor, con la misma secuencia, o con el mismo número de secuencia, pero con una cuenta de saltos menor [25].

b) Mantenimiento de la ruta: Una ruta se mantiene mientras la comunicación entre nodo origen y destino está activa una vez que la transmisión de paquetes de datos se haya detenido se activa un temporizador para eliminar la ruta de las tablas de enrutamiento. Este proceso usa los mensajes de Hello y RERR (Route Error) [25], [41].

- **Mensaje Hello:** Los nodos de la red que forman parte de rutas activas pueden transmitir periódicamente mensajes Hello, su objetivo es anunciar periódicamente su presencia en el rango de cobertura de sus vecinos para preservar una ruta. Si un nodo intermedio no recibe un mensaje Hello de su siguiente salto, se asume pérdida de conexión y que un fallo de enlace local se produjo.
  No es necesario que un nodo envíe mensajes Hello si su tabla de enrutamiento está vacía. Esto evita una inundación innecesaria de tráfico en la red.

- **RERR:** Cuando un nodo ya no es alcanzable envía este mensaje, ya sea porque detectó una ruptura de enlace local o por la falla de un mecanismo local de reparación. El nodo propaga el mensaje RERR hasta el origen para informarle del nuevo destino inalcanzable, véase figura 1.13. Después de recibir el RERR, si el

nodo origen aún requiere la ruta, este puede reiniciar un descubrimiento de ruta [4-37].

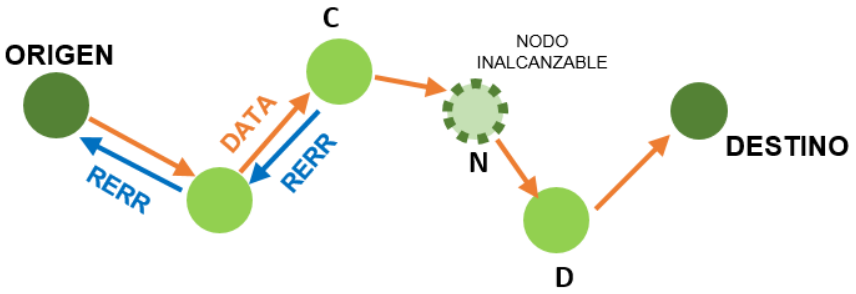

Figura 1.13. Emisión de mensajes RERR. Figura adaptada de [37].

En la Figura 1.13 se muestra como el nodo N se volvió inalcanzable por una ruptura de enlace entre C y D, por tal C envía mensajes de error RERR hasta alcanzar el origen y notificar lo sucedido.

2. METODOLOGÍA

Esta sección describe las etapas que se desarrollaron para conseguir el objetivo de este proyecto.
De inicio se analiza el simulador de red NS-3; su enfoque, lenguajes de programación, constructores, librerías, estructura de módulos y clases. Lo cual nos permitirá desarrollar las simulaciones de una MANET, modificar el protocolo de enrutamiento AODV, definir el mecanismo de control de la potencia de transmisión, analizar los resultados, entre otros.

La segunda parte describe el diseño y simulación del mecanismo de control de potencia, se indica todas las modificaciones e implementaciones que fueron necesarias en AODV para conseguir el control de potencia requerido.

En la tercera sección se muestran los escenarios en los cuales se llevó a cabo las simulaciones y se describen los modelos de movilidad empleados.

Finalmente se detalla el proceso para generar los ficheros (archivos) que fueron necesarios para imprimir los resultados de las simulaciones y su análisis correspondiente.

2.1. ESTUDIO DE NS-3

2.1.1 Generalidades

NS-3 [49] es un simulador de redes de eventos discretos para sistemas de internet que surge en 2006, fue desarrollado para proporcionar una plataforma de simulación de red abierta y extensible. Licenciado bajo GNU [50] GPLv2 [51], y está disponible para la investigación, desarrollo y uso. Proporciona modelos de redes de paquetes de datos y permite llevar a cabo experimentos de simulación en escenarios reales con tecnologías como: Wi-Fi, WiMAX, LTE; redes de sensores, redes cableadas TCP/IP, etc.
NS-3 está diseñado como un conjunto de bibliotecas que se pueden combinar entre sí o con bibliotecas de softwares externos. Mientras que algunas plataformas de simulación proporcionan un único interfaz gráfico de usuario integrado en el que se llevan a cabo todas las tareas, NS-3 es más modular en este aspecto; maneja varios interfaces externos y de análisis y visualización de datos.

El simulador es principalmente usado en sistemas Linux, aunque existe soporte para

FreeBSD, Cygwin (Windows) con un entorno de máquina virtual Linux mediante el servidor VMware.

Sus programas pueden escribirse en c++, phyton, o combinación de las mismas; permite el interfaz con lenguajes de alto nivel (por ahora sólo Python). Es posible además la extracción de resultados de simulación en ficheros para su análisis posterior. Debe aclararse que NS-3 no es una extensión compatible de NS-2, se trata de un simulador diferente.

2.1.2 Instalación del simulador

NS-3 requiere de Linux para su funcionamiento. En este trabajo se utiliza una máquina virtual con Ubuntu 14.04.1. en VMware.

Previo a la instalación del simulador, es necesario una serie de prerrequisitos [52], todos éstos se describen en el ANEXO I, una vez instalados se debe realizar lo siguiente:

1. Crear un nuevo directorio:
   mkdir NS-3
2. Descargar NS-3 de la página oficial del simulador, o descargar directamente el contenido del servidor web en la nueva carpeta NS-3:
   wget http://www.nsnam.org/release/ns-allinone-3.25.tar.bz2
3. Descomprimir el archivo
   tar xvf ns-allinone-3.25.tar.bz2
4. Instalar la version GUI en ns-allinone-3.25

   sudo apt-get install synaptic

   sudo synaptic
5. Construcción de NS-3 con build
   ./build.py --enable-examples --enable-tests
6. Construcción de NS-3 con waf (opcional)
   ./waf -d debug --enable-examples --enable-tests configure

Constructor Waf

NS-3 usa "Waf" [53] como un sistema constructor basado en Phyton que permite configurar, compilar, limpiar, instalar y desinstalar aplicaciones. Se describen a continuación las características más importantes de Waf [54]:

- *Orden de construcción automática:* El orden de construcción se calcula a partir de archivos de entrada y salida, entre otros.
- *Rendimiento:* Las tareas se ejecutan en paralelo de forma automática, el tiempo de inicio debe ser rápido (separación entre configuración y construcción).
- *Soporte IDE:* Generadores de proyectos Eclipse, Visual Studio y Xcode.

Waf es utilizado en particular por compañías innovadoras como Avalanche Studios y por proyectos open-source como NS-3.

Ejecución de un programa

Para generar un programa utilizable es necesario compilar el código fuente, para esto NS-3 usa Waf. Normalmente los scripts se ejecutan [55] bajo el control de Waf, lo cual permite que el sistema de compilación se asegure de que las rutas de biblioteca compartida se establecen correctamente y que las bibliotecas están disponibles en tiempo de ejecución. Waf verifica primero para asegurarse que el programa se construye correctamente y si es necesario ejecuta una compilación. Para ejecutar un programa NS-3, se usa la opción --run en Waf. Por ejemplo:

```
./waf --run nombre_de_programa
./waf --run nombre_de_programa -vis
```

La última es para ejecutar un programa con un paquete de phyton.

2.1.3 Estructura y terminología

La biblioteca de NS-3 está distribuida en módulos, los cuáles hacen uso de numerosas clases para implementar los diferentes modelos que incluye el simulador [56]. *E*l núcleo de simulación y los modelos de NS-3 están implementados en C ++.

Estructura

El software NS-3 está compuesto por diferentes módulos (capas), véase Figura 2.1. Los cuales se construyen como una biblioteca de software independiente que únicamente pueden verse influenciados de los módulos inferiores a ellos.
Es importante distinguir entre módulos y modelos. Los modelos NS-3 son simplemente representaciones abstractas de objetos del mundo real tales como dispositivos, canales, protocolos, etc. Mientras que un módulo NS-3 puede incluir más de un modelo; por ejemplo, el módulo de Internet contiene modelos para TCP y UDP.

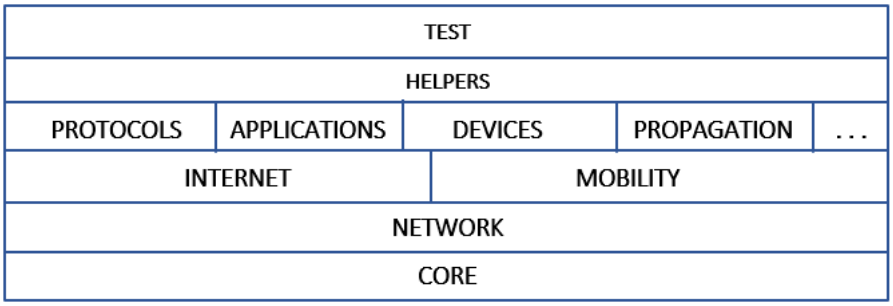

Figura 2.1. Organización de NS-3 en módulos. Figura adaptada de [56]

Los programas NS-3 individuales pueden hacer uso de los módulos o también denominados bibliotecas para llevar a cabo las simulaciones.

En el directorio src de NS-3 se ubica todo el código fuente del simulador, allí se pueden apreciar los diferentes módulos y modelos del mismo. A continuación, se describen los módulos del simulador:

- **Core:** [56] Es el núcleo del simulador y permite el uso de clases para la ejecución de las simulaciones. Este módulo abarca las clases y funciones comunes para todos los modelos de protocolo, hardware y medio. El núcleo de simulación se implementa en src / core. Sus componentes permiten el manejo de variables aleatorias como número de secuencia inicial, uso de punteros, manejo de eventos en tiempo de simulación, vectores, entre otros. Los principales elementos se muestran en la Figura 2.2.

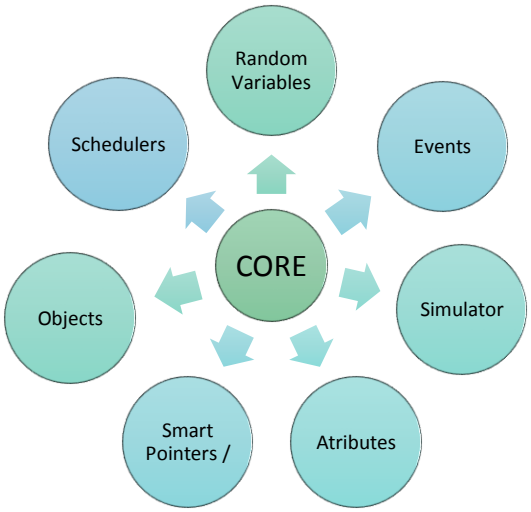

Figura 2.2. Componentes del núcleo del simulador.

- **Random Variables:** NS-3 soporta una serie de objetos de variables aleatorias definidos por la clase RandomVariableStream. Se derivan de ns3::Object y son manejados por punteros inteligentes. La forma correcta de crearlos es usando CreateObject <> como, por ejemplo:

```
Ptr<UniformRandomVariable> x =CreateObject<UniformRandomVariable>();
```

Se puede acceder a los valores llamando métodos en el objeto como:

```
Variable = x-> GetInteger ();
```

- **Events:** Ns-3 es un simulador de red de eventos discretos. Los eventos habilitan un objeto o clase para notificar la ocurrencia de algo a otras clases u objetos. La clase que notifica o genera el evento se denomina *publicador* y a las clases que controlan o reciben el evento se las conoce como *suscriptores*. Conceptualmente, el simulador mantiene un seguimiento de una serie de eventos que están programados para ejecutarse en un momento de simulación específico.

- **Simulator:** La clase *Simulator* Controla el tiempo virtual y la ejecución de los eventos de simulación. El trabajo del simulador es ejecutar los eventos en orden secuencial. Una vez que se complete un evento, el simulador saltará al siguiente (o terminara si no hay más eventos en la cola de eventos). Por ejemplo, si se ejecuta un evento programado para el tiempo de simulación “20 segundos", y el siguiente evento está programado para "80 segundos", el simulador saltará inmediatamente de 20 s a 80 s (del tiempo de simulación) para ejecutar el próximo evento. Para ejecutar un par de eventos programados e iniciar la simulación se debe hacer un llamado a Simulador::Run. Una vez iniciado, se ejecutarán secuencialmente todos los eventos programados desde el más antiguo hasta el más reciente hasta que no haya más eventos en la cola o hasta que el Simulator::Stop haya sido llamado.

- **Object:** NS-3 es un sistema de objetos C ++. [57] Un objeto es una variable; es un lugar que puede almacenar datos y operaciones que pueden actuar sobre esos datos durante el tiempo de ejecución. Un objeto es el resultado de una serie de procesos definidos en una clase, es decir, es la de la instanciación de una clase.
  Los objetos NS-3 son memoria administrada por una referencia de puntero inteligente y usan las propiedades comunes del diseño clásico orientado a objetos de c++, tales como: abstracción, encapsulación, herencia y polimorfismo.

Hay tres clases bases que se usan en NS-3. Las clases que heredan de estas clases base pueden instanciar objetos con propiedades especiales. Las clases base son: *class Object*, *class ObjectBase* y class *SimpleRefCount*. No es necesario que los objetos NS-3 hereden de estas clases base, pero estas permiten obtener propiedades especiales. Una de esas propiedades son los punteros inteligentes (*class Ptr*). En la práctica, *Class Object* es la clase que comúnmente usa NS-3.

- **Smart Pointers:** [57] Un puntero inteligente es un tipo abstracto de datos que simula un puntero y al mismo tiempo habilita funciones adicionales, tales como la gestión automática de la memoria ; estas características reducen los errores causados por el uso indebido de punteros. Suelen además realizar un seguimiento de la memoria a la que apuntan y permiten administrar recursos como: manejo de archivos y conexiones de red.

  El mal uso de un puntero puede ser una fuente importante de errores, por lo que un puntero inteligente evita en su mayoría la pérdida de memoria, esto lo consigue con la liberación de memoria automática, manejando también destrucción automática de objetos; un objeto controlado por un puntero inteligente se destruye automáticamente (finaliza y luego es liberado).

  Este tipo de punteros se utilizan ampliamente en las API NS-3 para evitar pasar referencias a una gran cantidad de objetos, lo cual puede causar pérdidas de memoria. Están representados por la clase *Ptr*.

  Los objetos NS-3 por lo general se almacenan en punteros inteligentes, estos objetos pueden crearse al momento del almacenamiento en el puntero o pueden ya ser existentes.

  Para crear un objeto y almacenarlo en un puntero inteligente NS-3, se agrega:

```
Ptr<Objeto_nuevo> p = CreateObject<Objeto_nuevo> ();
```

  Con este almacenamiento se evita pérdidas de memoria y se puede llamar a los métodos o funciones del objeto usado.

  El concepto de puntero inteligente es muy importante para el presente trabajo ya que como parte del mecanismo de control de potencia que se diseñará es necesario acceder a capa física desde el protocolo de enrutamiento y para esto se usará una serie de punteros que permitirán obtener los objetos de las diferentes capas, en la sección 2.2.1 se detallará su uso.

- **Network:** Incluye todos los componentes vinculados con los paquetes de información, etiquetas, cabeceras, nodos, puertos y direcciones IP, es un módulo imprescindible para simular cualquier tipo de red. En NS-3 este módulo se implementa en el directorio src /network. Sus principales clases se describen en la Figura 2.3.

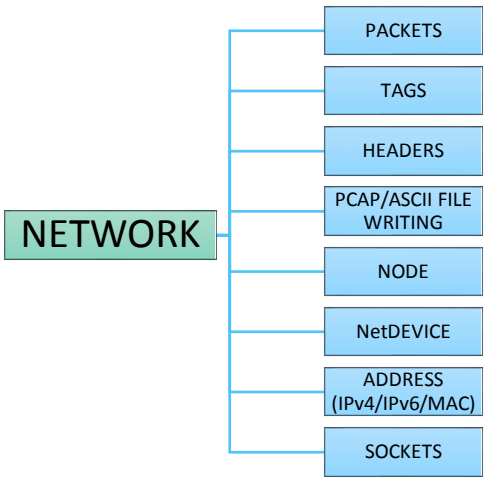

Figura 2.3. Componentes del módulo Network

Los paquetes son objetos fundamentales en el simulador y su diseño es importante a nivel de rendimiento y de gestión de recursos. En la sección 2.1.4 se describen las clases de Network que se usaron en el presente trabajo.

- **Mobility:** La movilidad en NS-3 pude obtenerse de manera interna o externa. Internamente este módulo incluye un conjunto de modelos que se utilizan para rastrear y mantener la posición cartesiana actual y la velocidad de un objeto, una fuente de rastreo y una serie de clases auxiliares que permiten la movilidad en los nodos. El código fuente lo podemos encontrar en: src/mobility. Algunos modelos que se incluyen en este módulo son [58]:

  - *ConstantPosition:* En este modelo de movilidad, la posición actual de un nodo no cambia hasta que se vuelve configurar en un nuevo valor por parte del usuario.

  - *GaussMarkov:* Es una versión del modelo de movilidad Gauss-Markov pero en 3D. Tiene memoria y aleatoriedad, los cuales son configurados mediante el parámetro "alfa" para su modelamiento. Cada objeto inicia con una velocidad específica, dirección y ángulo de paso equivalentes a la velocidad promedio, dirección y medio.

- *ConstantVelocity:* Este modelo mantiene la velocidad constante una vez que se ha configurado en un nuevo valor.

Algunas de las funciones básicas de los modelos son:

- *GetPosition:* Devuelve la posición actual de un objeto.
- *DoGetVelocity:* Permite obtener la velocidad actual.
- *GetDistanceFrom:* Retorna la distancia entre dos objetos usando como unidad de medida los metros.

Existen además herramientas externas que permiten generar modelos de movilidad, tales como [59]:

- *SUMO (Simulation of Urban Mobility),* es un paquete de simulación de tráfico de carreteras, portable, microscópico y de código abierto, diseñado para manejar grandes redes de carreteras. Está licenciado bajo la GPL.

- *TraNS (Traffic and Network Simulation Environment)*, herramienta de interfaz gráfica de usuario que integra simuladores de tráfico (SUMO) y de red (ns2) para generar simulaciones realistas de VANETs. TraNS permite que la información intercambiada en una VANET influya en el comportamiento de los vehículos en el modelo de movilidad. Por ejemplo, si un vehículo transmite información que advierte de un accidente, algunos de los vehículos vecinos pueden disminuir su velocidad.

- *BonnMotion:* Es un software que genera modelos de movilidad generalmente para redes ad-hoc, mismo que se usará para generar movilidad en los escenarios de este trabajo; en la sección 2.1.4 en Mobility module se describe con mayor detalle este software.

- **Helpers:** Permiten realizar de una manera sencilla tareas repetitivas y comunes, facilitan la asociación entre nodos, dispositivos de red y sus canales. Por ejemplo, para crear un dispositivo de red es necesario realizar una serie de acciones similares como: agregar una dirección MAC o IP, instalar el dispositivo de red en un nodo, configurar la pila de protocolos, conectar el dispositivo de red a un canal, etc. El "helper" permite combinar esas muchas operaciones distintas para evitar programas largos y tediosos de codificar.

Terminología

En NS-3 algunos términos tienen un significado en específico [60]. Los de mayor interés en este proyecto son los descritos a continuación:

- **Nodo:** Es un dispositivo informático básico al cual se le agregará funcionalidad; protocolos, tarjetas de red y aplicaciones. En NS-3 se representa por la clase *Node*, la cual proporciona métodos y funciones para la operación de los dispositivos en una simulación. La clase NodeContainer nos permite el manejo de un conjunto de nodos.

- **Canal:** Es una representación del medio físico por el cual se comunican los nodos. En NS-3, conectar un nodo a un objeto de canal, representa un canal de comunicación. El simulador puede modelar algo tan simple como un alambre, o tan complicado como el ambiente impredecible y con obstáculos de un canal inalámbrico. En NS-3 se representa por la clase *Channel*. Algunas clases de canales son:

- *CsmaChannel,* equivalente a un cable Ethernet. Modela un medio de comunicación de acceso múltiple con censado de portadora.
- *PointToPointChannel*, representa un canal punto a punto muy simple. Por ejemplo: un cable RS-232.
- *WifiChannel,* modela un canal inalámbrico.

- **Aplicación:** NS-3 no maneja un concepto real de sistema operativo tal como las aplicaciones de software que se ejecutan en los ordenadores para realizar tareas en el “mundo real”. En NS-3 las aplicaciones son funciones que se ejecutan en los nodos para realizar tareas en el mundo simulado y se representa por la clase *Application*. Por ejemplo, para incluir una aplicación del tipo cliente/servidor con envío de tráfico UDP, se usa: UdpEchoClientApplication, UdpEchoServerApplication.

- **Dispositivo de red**: Un dispositivo de red debe ser instalado en un nodo con el fin que éste pueda comunicarse con otros nodos en la simulación mediante los canales. Un nodo puede conectarse a más de un canal a través de múltiples interfaces de red tal como en la realidad. En NS-3 se representa por la clase *NetDevice* la cual proporciona métodos para gestionar las conexiones a los nodos y canales. Por ejemplo: CsmaNetDevice, WifiNetDevice, este último es de interés para el presente trabajo y se explica en la siguiente sección.

La Figura 2.4 es una representación de los elementos fundamentales en NS-3, tales como el nodo, tarjeta de red, canal, aplicaciones y su interrelación. Se puede apreciar además como los nodos A y B se conectan a un canal a través de sus objetos.

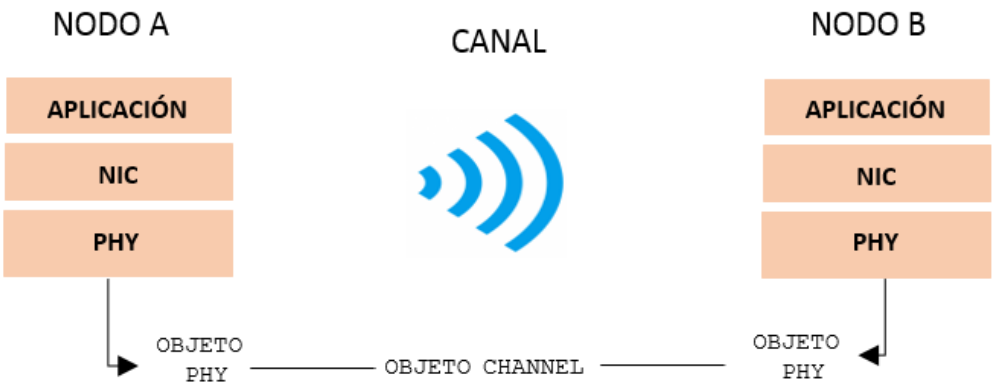

Figura 2.4. Representación de un nodo, tarjeta de red y canal en NS-3.

2.1.4 Módulos y clases usados

A continuación, se detalla los módulos también conocidos como bibliotecas que fueron necesarios para el desarrollo del presente trabajo, la Figura 2.5 muestra un resumen.

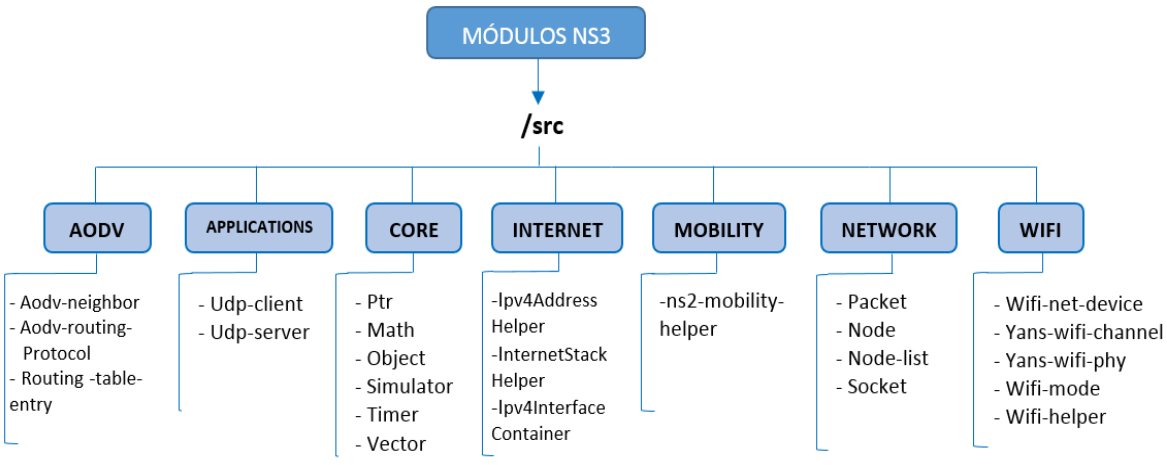

Figura 2.5. Módulos y clases empleados**.**

Como se observa en la Figura 2.5, cada módulo usa varias clases para modelar los diferentes protocolos, canales, dispositivos, etc. Las clases de mayor importancia para este trabajo se describen a continuación.

Core Module

Es el núcleo de NS-3, el cual permite la ejecución de las simulaciones mediante el uso de clases. Las clases y funciones más importantes que han sido empleadas son [61]:

- *SmartPointer::Ptr:* Permite el uso de punteros inteligentes.
- *Object:* Es una clase base que permite administrar memoria y generación de objetos.
- *Simulator*::Run(): Función global que ejecuta la simulación en NS-3.
- Simulator::Stop(): Es una función global que detiene la simulación luego que han transcurrido los segundos indicados en el argumento de la función.
- Simulator::Destroy(): Función global que libera los recursos comprometidos con la simulación en curso.
- *Simulator::ScheduleDestroy:* Programa la expiración de un evento cuando se llame a Simulator::Destroy
- *Simulator::Schedule:* Programa un evento para que después del tiempo indicado.
- *Simulator::Now():* Esta función crea una instancia de tiempo en NS-3 la cual contiene el tiempo de simulación actual.
- *Time:* Esta clase muestra los valores de simulación virtual y la resolución del valor del tiempo de simulación. La resolución es el intervalo de tiempo más pequeño representado en NS-3. La resolución predeterminada es en nanosegundos
- *CommandLine:* Clase que permite modificar valores por defecto mediante la línea de comandos, durante el inicio de la ejecución de la simulación en curso.
- *Math:* Esta clase incluye varias funciones que facilitan cálculos matemáticos.

Wifi module

Este módulo es uno de los más importantes en este proyecto ya que hace referencia a la tecnología de transmisión que considera este trabajo. Esta sección describe todos los modelos wifi que se emplearon para crear los escenarios base que permitirán simular una red ad-hoc. Además, se hace una descripción de la arquitectura que maneja un *device* en NS-3.

- **Arquitectura de un dispositivo NS-3 con WifiNetDevice**

Un dispositivo “nodo” en NS-3 está formado por diferentes capas, similar a lo que define el modelo de referencia ISO/OSI. En este caso como se requiere nodos que mantengan comunicación inalámbrica se hará referencia al módulo wifi. Véase Figura 2.6.

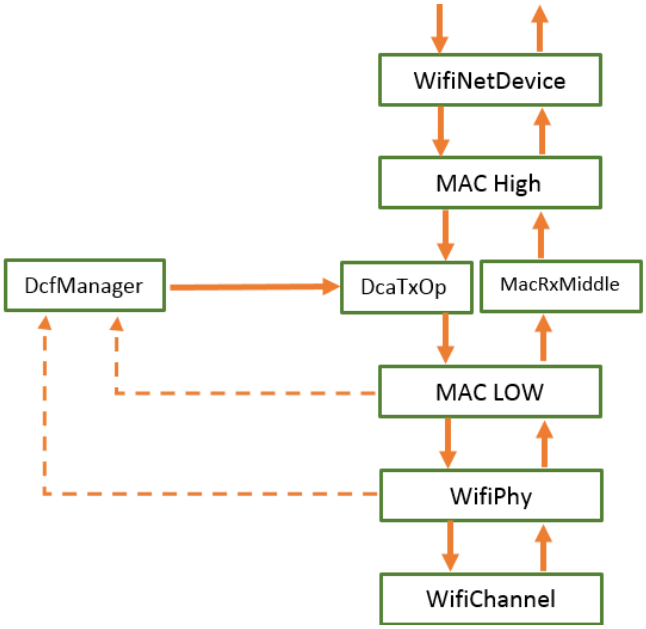

Figura 2.6. Arquitectura de un dispositivo inalámbrico con Wifi en NS-3

La Figura 2.6 muestra las capas por las que está formado un nodo inalámbrico en NS-3, a continuación de describe cada una de ellas:

WifiNetDevice

El WifiNetDevice [59] modela una tarjeta de red inalámbrica basada en el Estándar IEEE 802.11 [62]. Los nodos en NS-3 pueden contener un conjunto de objetos NetDevice al igual que un ordenador real contiene diferentes interfaces de red para Ethernet, Wifi, Bluetooth, etc. Los nodos pueden tener varios WifiNetDevices en diferentes canales, y el WifiNetDevice puede coexistir con otros tipos de dispositivos.

Se pueden crear modelos de infraestructura o ad hoc basados en IEEE 802.11 como:

- 802.11 Básico DCF [63] para infraestructura y redes Ad-hoc.
- 802.11 a, b, g, n, ac, ax en bandas de 2.4 GHz y 5GHz respectivamente.
- QoS basado en EDCA [64] y las extensiones de cola de 802.11e.
- 802.11 “s” para redes malladas, “p” para entornos vehiculares.

El código fuente de WifiNetDevice y sus clases se encuentra en el directorio src /wifi. La implementación del WifiNetDevice es modular como se muestra en la Fig.2.6.

Modelos MAC High

Estos modelos definen el tipo de comunicación inalámbrica; agregan procesos sobre la generación de beacons (mensajes de anuncio) a nivel MAC, sondeo, y estados de asociación. Se definen tres tipos:

- *Acces Point (Ns3::ApWifiMac):* implementa un AP que genera balizas periódicas, y que acepta cada intento de asociación.
- *STA Station (ns3::StaWifiMac):* implementa un sondeo activo y un estado de asociación que se encarga de la re asociación automática cuando se pierden demasiadas beacons.
- *Red ad-hoc (ns3::AdhocWifiMac):* implementa una MAC Wi-Fi que no genera beacons ni realiza sondeo o asociación.

Modelos MAC Low

Modelan funciones de acceso al medio como (DCF [63] y EDCA [64]), RTS/CTS y ACK. En NS-3 esta capa se divide en: MAC Low y MAC Middle. Tiene tres componentes principales:

- *ns3::MacLow*, se encarga de las transacciones RTS / CTS / DATA / ACK.
- *ns3::DcfManager y ns3::DcfState*, implementan funciones DCF y EDCAF.
- *ns3::DcaTxop y ns3::EdcaTxopN*, manejan la cola, fragmentación y retransmisión de paquetes si son necesarias.

Existen además múltiples algoritmos de control de velocidad que pueden ser usados por la capa MAC baja; algunos de ellos son:

- *ConstantRateWifiManager:* Siempre usa la misma tasa de transmisión para cada paquete enviado. Los usuarios pueden establecer un 'DataMode' deseado para todos los paquetes 'unicast' y 'ControlMode' para los paquetes de control.

- *MinstrelWifiManager:* Actualmente es el algoritmo de control de velocidad por defecto del kernel de Linux. Minstrel mantiene un seguimiento de la probabilidad de enviar con éxito una trama de cada tasa disponible; luego calcula el rendimiento esperado multiplicando la probabilidad por la tasa. Este enfoque se elige para asegurarse de que las tasas más bajas no se seleccionan en favor de las tasas más altas ya que es posible que las tasas más bajas tengan mayor probabilidad.

- *IdealWifiManager:* Selecciona el mejor modo de acuerdo con el SNR del último paquete enviado. Considere el nodo A que envía un paquete unicast al nodo B. Cuando B recibe correctamente el paquete, registra el SNR del paquete recibido en un ns3 :: SnrTag y agrega la etiqueta a un ACK de regreso a A. Al hacer esto, A es capaz de aprender la SNR del paquete enviado a B. A, a continuación, utiliza la SNR para seleccionar un modo de transmisión basado en un conjunto de umbrales SNR.

Modelos de capa física (PHY)

Se encargan de enviar y recibir la señal inalámbrica del WifiChannel. Deciden si cada trama será decodificada correctamente o no dependiendo de la modulación, relación señal a ruido (e interferencia) para el paquete, y del estado de la capa física (por ejemplo, la recepción no es posible mientras se está transmitiendo o en estado dormido). Están pendientes además del consumo de energía.

NS-3 maneja dos modelos de capa física con una interfaz base definida en la clase ns3::WifiPhy, estos son:

- *SpectrumWifiPhy:* Esta clase es una implementación basada en el espectro, utilizada para otros modelos inalámbricos NS-3. El espectro permite una descomposición de frecuencias de fino grano de la señal, y permite escenarios para incluir múltiples tecnologías coexistentes en el mismo canal.
- *YansWifiPhy,* esta clase fue usada para este proyecto por lo que se detalla más adelante.

WifiChannel

Ns3::WifiChannel es una clase base abstracta que permite implementar diferentes canales; al momento el único canal disponible “wifi” es el ns3::YansWifiChannel el cual trabaja en conjunto con la clase ns3::WifiPhy.
En la realidad, un canal permite que dos dispositivos se comuniquen mediante una conexión entre sus capas físicas respectivas. En NS-3, cada capa de la arquitectura de red está representada por un objeto; por tal el modelo WifiChannel interconecta objetos WifiPhy (representantes de capa física) para que los paquetes enviados por un Phy sean recibidos por algunos o todos los otros Phys en el canal. Véase Figura 2.7.

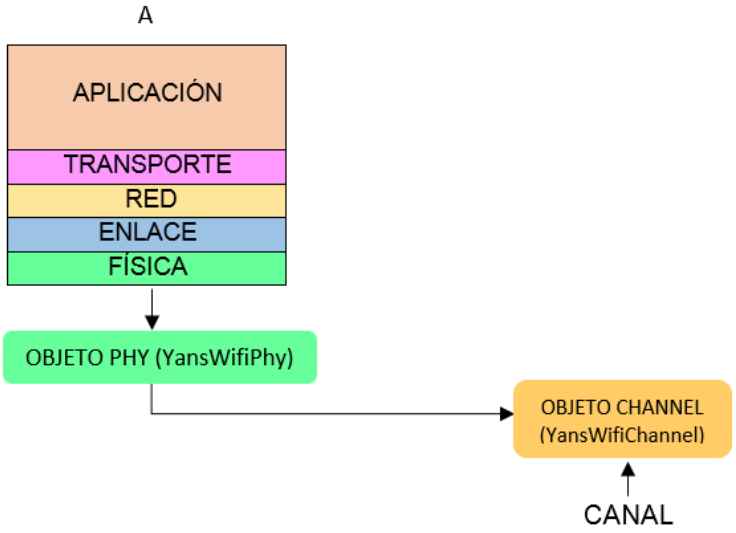

Figura 2.7. Conexión de canal en NS-3.

La Fig.2.7. Indica la conexión entre un canal y capa física, los cuales están representados por objetos de NS-3. Conectar los objetos representa el medio de comunicación.

YansWifiChannel

Como ya se indicó, el YansWifiChannel es actualmente el único modelo de canal en el módulo wifi y soporta modelos de pérdidas y retardo por propagación; ns3::PropagationLossModel y ns3::PropagationDelayModel respectivamente [65]. Se pueden añadir varios modelos de pérdida por propagación y un modelo de propagación del retardo al canal. La potencia de recepción final tiene en cuenta todos los modelos encadenados; de esta manera se puede utilizar un modelo de desvanecimiento lento y rápido juntos [66].

Los paquetes enviados desde un objeto ns3::YansWifiPhy al canal con una potencia de señal particular, se copian a todos los objetos ns3::YansWifiPhy receptores; después la potencia de la señal se reduce debido al modelo de pérdida de propagación.

Modelos de pérdidas por Propagación

Los modelos de pérdidas por propagación calculan la potencia de la señal recibida considerando la potencia de la señal de transmisión y las posiciones de las antenas transmisoras y receptoras. Varios de los modelos disponibles en NS-3 son: Cost231 [67], OkumuraHata [68] , Jakes [69] , LogDistance [70] , ThreeLogDistance [71], Nakagami TwoRayGround [72], entre otros. A continuación, se describen algunos de ellos [65]:

- *FixedRssLossModel:* Este modelo establece un nivel de potencia de recepción constante (configurable) independiente de la potencia de transmisión. Si se considera varios modelos de pérdidas, éste debería ser el primer modelo para evitar que se excluyan las pérdidas calculadas por los demás modelos incluidos.

- *MatrixPropagationLossModel:* La pérdida de propagación se fija para cada par de nodos y no depende de sus posiciones reales. Útil para pruebas sintéticas. Por defecto se supone que la pérdida de propagación es simétrica.

- *RandomPropagationLossModel:* Considera una pérdida de propagación aleatoria; cambia cada vez que se llama al modelo. Como consecuencia, todos los paquetes (incluso aquellos que se envíen entre dos nodos fijos) experimentan una pérdida de propagación aleatoria.
- *RangePropagationLossModel:* La pérdida depende solamente de la distancia en metros (rango) entre el transmisor y el receptor. El único atributo MaxRange determina la pérdida de trayecto. Los receptores en o dentro de los medidores MaxRange reciben la transmisión al nivel de potencia de transmisión; los que estén más allá de MaxRange reciben una potencia en -1000 dBm.

- *FriisPropagationLossModel:* Implementa una pérdida de propagación de Friis, permite predecir el nivel de potencia que se recibirá considerando cierta distancia en condiciones ideales, es decir, sin obstáculos de ninguna naturaleza cercanos al enlace que puedan afectar la propagación electromagnética. La fórmula original considerando el caso de una antena isotrópica sin pérdida de calor fue descrita como [73]:

$$\frac{P_r}{P_t} = \frac{\lambda^2}{(4\pi d)^2}$$

Ecuación 2.1. Propagación de Friis sin pérdida de calor

Con las actualizaciones a la fórmula original se tiene:

$$P_r = \frac{P_t G_t G_r \lambda^2}{(4\pi d)^2 L}$$

Ecuación 2.2. Ecuación original de Friis

Donde:
P_t: Potencia de transmisión (w)
P_r: Potencia de recepción (w)

G_t: Ganancia de transmisión (adimensional)
G_r: Ganancia de recepción (adimensional)
λ : Longitud de onda (m)
d: Distancia (m)
L: Pérdida del sistema (adimensional)

En la implementación, $\lambda = {}^{C}/_{f}$, donde C es la velocidad de la luz en vacío, y f es la frecuencia en Hz. Este modelo es válido sólo para propagación en el espacio libre.

Modelos de retardo de Propagación
Para este caso NS-3 puede implementar los siguientes modelos:

- *RandomPropagationDelayModel:* El retardo de propagación es completamente aleatorio, y cambia cada vez que se hace un llamado al modelo. Todos los paquetes, incluso aquellos que se envían entre dos nodos fijos experimentan un retraso aleatorio. Como consecuencia, el orden de los paquetes transmitidos no se mantiene.
- *ConstantSpeedPropagationDelayModel:* De este modelo se hablará más adelante en detalle ya que es el modelo que se utilizará para los escenarios.

Luego de haber definido la arquitectura de un nodo con interfaz inalámbrico en NS-3 se detallan los modelos de canal inalámbrico, capa física y mac empleados. Las clases necesarias para configurar cada uno de los parámetros determinantes de cada capa, tales como: tipo de comunicación, estándar 802.11, modulación, modelos de propagación, modelos de retardo, tasas de error, etc. Asimismo, se describe el dispositivo de red y demás afines que han sido empleados para los escenarios de simulación de una red ad-hoc.

- **WifiMacHelper**

Este ayudante permite configurar el modelo de capa MAC y define el tipo de comunicación inalámbrica. En este trabajo para los escenarios se emplea el ayudante de capa MAC en su configuración por defecto y el tipo de comunicación será Ad-hoc. Las líneas de código necesarias para definir lo indicado son:

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

- **YansWifiChannelHelper**

Para definir la parte lógica de los escenarios de este trabajo se usó una configuración de canal inalámbrico mediante el ayudante YansWifiChannelHelper. Esta clase permite [74] crear y manejar un canal inalámbrico WifiChannel con un modelo de pérdida por propagación y retardo predefinidos. Como modelo de pérdida por propagación usa el ns3::LogDistancePropagationLossModel y para modelar el retardo de propagación usa el ns3::ConstantSpeedPropagationDelayModel.

- *LogDistancePropagationLossModel:* Es un modelo que calcula la pérdida por propagación en base a la distancia. La potencia de recepción se calcula con el denominado modelo de propagación de distancia logarítmica, la Ecuación 2.3. define la pérdida de trayectoria.

$$L = L_o + 10nlog\left(\frac{d}{d_o}\right)$$

Ecuación 2.3. Pérdida de trayectoria para el modelo Log Distance

Donde:

n: Exponente de distancia de pérdida de trayecto
d_o: Distancia de referencia (m)
L_o: Pérdida de trayecto a la distancia de referencia (dB)
d: Distancia (m)
L: Pérdida de trayectoria (dB)

Cuando se requiere la pérdida de trayecto a una distancia menor que la distancia de referencia, se devuelve la potencia de transmisión.

Para este modelo, NS-3 usa por defecto una pérdida de referencia de 46.677 dB con 1m de distancia. La pérdida de referencia fue calculada en base al modelo de pérdida por propagación de Friis en la banda 5.15 GHz. Esta pérdida de referencia debe cambiarse si se usa 802.11b, g o n en 2.4GHz.

- *ConstantSpeedPropagationDelayModel:* En este modelo, la señal viaja con velocidad constante, igual a la velocidad de la luz. El retardo se calcula de acuerdo con las posiciones del transmisor y del receptor. Se utiliza la distancia euclidiana entre las antenas Tx y Rx. Este modelo considera que la Tierra es plana.

Para obtener la configuración de canal por default se emplea las siguientes líneas de código:

```
YansWifiChannelHelper wifiChannel = YansWifiChannelHelper::Default ();
wifiPhy.SetChannel (wifiChannel.Create ());
```

- **YansWifiPhy**

La clase YansWifiPhy implementa un modelo de capa física de 802.11a basado en el paper “Yet Another Network Simulator” [75]; depende de los modelos de pérdida de canal y retardo proporcionados por las clases NS-3: *PropagationLossModel* y *PropagationDelayModel*. Es responsable de tomar las tramas de capa MAC y enviarlas al canal que esté conectado o viceversa. Puede estar en uno de los siguientes modos:

- *Transmisión (TX):* Cuando el objeto de capa física “PHY” Transmite una señal en nombre de su MAC asociado.
- *Recepción (RX):* Cuando el objeto PHY está recibiendo una señal.
- *Ocupado CCA*: Cuando el objeto de capa física no está en modo Tx ni Rx, pero la energía medida es mayor que el umbral detectado.
- *IDLE:* No está en los estados TX, RX o CCA BUS.
- *SWITCHING:* Cuando el objeto PHY está cambiando de canal.
- *SLEEP:* PHY está en modo de ahorro de energía y no puede enviar ni recibir tramas.

Para usar esta clase y sus funciones asociadas es necesaria la librería #include "yans-wifi-phy.h". Existen una serie de funciones y variables en esta clase, pero para interés de este trabajo solo se estudiará las que se vinculan a la potencia de transmisión, tales como:

- *SetTxPowerStart:* Define el mínimo nivel de potencia de transmisión, por defecto se encuentra configurado en 16.026 dBm.
- *SetTxPowerEnd:* Define el máximo nivel de potencia de transmisión, configurado por defecto en 16.026 dBm. Para este proyecto se lo ha modificado a 20 que es lo que define el estándar.
- *GetTxPowerStart:* Permite obtener el mínimo valor de la potencia de transmisión del dispositivo.
- *SetTxLevel:* Configura los niveles de potencia para las transmisiones.
- *EnergyDetectionThreshold:* Define un límite en -96 dBm para que la capa física detecte una señal; la energía de la señal recibida debe ser mayor a este valor.

- **YansWifiPhyHelper**

El ayudante YansWifiPhyHelper se configura para incluir de manera sencilla el modelo de capa física en los escenarios base. Los objetos de capa física creados deben conectarse a los modelos de canal inalámbrico; los objetos WifiPhy deben ser apropiados para el

YansWifiChannel. Para esto el YansWifiPhyHelper crea una instancia entre ellos y agrega posiblemente modelos de tasas de error y un puntero hacia un modelo de movilidad. Véase Figura 2.8.

Debe estar claro que el YansWifiPhyHelper no crea el objeto de capa física WifiPhy en sí, únicamente permite saber a qué canal está conectado el nodo.

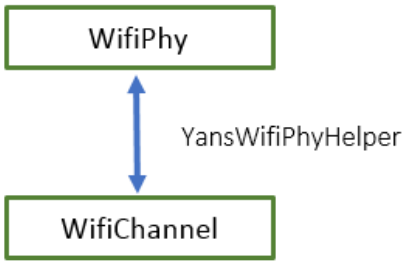

Figura 2.8. Conexión entre objetos de capa física y canal.

En los escenarios base de este trabajo se usó el YansWifiPhyHelper por defecto, el cual es configurado con el modelo de tase de error NistErrorRateModel. Para cambiar el modelo se puede llamar al método: YansWifiPhyHelper::SetErrorRateModel. Para configurar este ayudante en los escenarios se agrega:

```
YansWifiPhyHelper wifiPhy = YansWifiPhyHelper::Default ();
```

Este ayudante también permite imprimir los archivos de traza mediante el código:

```
AsciiTraceHelper ascii;
wifiPhy.EnableAsciiAll (ascii.CreateFileStream ("aodv2_result.tr"));
```

- **WifiHelper**

Esta clase permite [74] crear el dispositivo de red en sí. Para lo cual primero se crea el ayudante con las configuraciones por defecto, como sigue:

```
WifiHelper nombre_del_ayudante;
```

Esta configuración agrega el estándar wifi predeterminado que es 802.11a y la función *SetRemoteStationManager* la cual permite configurar parámetros de las estaciones remotas. Para cambiar el estándar wifi, se hace un llamado al ayudante WifiHelper:: SetStandard con el estándar deseado, así por ejemplo:

```
SetStandard (WIFI_PHY_STANDARD_80211b);
```

Para cambiar el tipo de RemoteStationManager se debe llamar al método WifiHelper:: SetRemoteStationManager.

```
wifi.SetRemoteStationManager ("ns3::ConstantRateWifiManager", "DataMode",
StringValue ("OfdmRate6Mbps"), "RtsCtsThreshold", UintegerValue (0));
```

Finalmente, para crear el dispositivo de red se debe instalar todas las capas creadas en el nodo o conjunto de nodos.

```
wifi.Install (wifiPhy, wifiMac, nodes);
```

Network Module

Este módulo permite incluir a las simulaciones la funcionalidad de la capa de red, maneja información referente a los paquetes, direcciones, etiquetas, sockets, nodos, dispositivos, entre otros. Las clases más útiles para este proyecto se describen a continuación [59]:

- **Clase Node, NodeContainer**

Esta clase permite crear el dispositivo básico “nodo” al cual se le agregará funcionalidad (aplicaciones, protocolos, tarjetas periféricas, movilidad, etc) para su desarrollo en el escenario de uso. Estos dispositivos se crean de forma individual o colectiva.

Para crear un nodo de forma individual se agrega:

```
Ptr <Node> n0 = CreateObject <Node> ();
```

Para crear una lista o conjunto de nodos:

```
NodeContainer nodes;
nodes.Create (10);
```

- **Packet**

Esta clase hace referencia a los paquetes, que son parte fundamental del simulador ya que representan los mensajes de información.

Cada paquete de red en NS-3 contiene un Buffer de bytes, una lista de etiquetas de byte, una lista de etiquetas de paquetes y un objeto PacketMetadata:

- *El búfer de bytes* almacena el contenido serializado de los fragmentos agregados a un paquete, con esto se espera que el contenido de un búfer de paquetes sea el de un paquete real. Los paquetes también se pueden crear con una carga útil cero arbitraria para la que no se asigna memoria real.

- *Cada lista de etiquetas* almacena un conjunto grande de estructuras de datos arbitrarias proporcionadas por el usuario en el paquete. Cada etiqueta se identifica únicamente por su tipo. Estas etiquetas normalmente contienen información de capa cruzada por paquete o identificadores de flujo.

La Fragmentación y desfragmentación son bastante naturales, se puede dividir en múltiples fragmentos y volver a ensamblar estos fragmentos. La gestión de memoria de los objetos Packet es automática y muy eficiente. Para este trabajo fue muy útil esta clase, pero nos limitamos a conocer su funcionamiento e influencias ya que esta clase estuvo implícitamente agregada al módulo del protocolo de enrutamiento y no fue objeto de modificación de este trabajo.

Sin embargo, varias de las funciones que se derivan de esta clase que fueron de utilidad, por ejemplo, para generar los archivos de resultados se emplearon:

- *GetUid:* Permite obtener el identificador (número) de paquete.
- *GetSize:* Devuelve el tamaño del paquete en bytes incluyendo la carga útil inicial.

- **NetDevice**

Esta clase simula la interfaz de red para el dispositivo. Define la API a la cual las capas IP y ARP necesitan tener acceso para administrar una instancia a capa de red de un dispositivo. Todavía no admite multidifusión a nivel MAC. La clase fue diseñada para ocultar muchos detalles de nivel MAC como sea posible desde la perspectiva de capa tres para permitir que una sola capa tres que funcione con cualquier tipo de capa MAC. Específicamente, *NetDevice* encapsula el formato de las direcciones MAC utilizadas por un dispositivo de tal manera que la capa tres no requiera modificación alguna para manejar nuevos formatos de dirección. La API actual ha sido optimizada para facilitar la adición de nuevos protocolos MAC y no añadir nuevos protocolos de la capa de red.

Internet Module

Este módulo [59] provee clases para que los nodos trabajen con la pila de protocolos de la arquitectura TCP/IPv4, incluyen protocolos como: IPv4, TCP, UDP, ARP, IPv6, entre otros. Se encuentra en el directorio src/Internet. Las clases usadas en este trabajo son:

- **InternetStackHelper**: Permite agregar la pila de protocolos TCP/IP sobre los nodos.

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

- **Ipv4AddressHelper:** Esta clase permite establecer el direccionamiento IPv4 sobre los nodos. Usa tres parámetros: red, máscara y primera dirección útil, ésta última es opcional. Para agregar direccionamiento es necesario las siguientes líneas de código.

```
Ipv4AddressHelper address;
address.SetBase ("10.0.0.0", "255.0.0.0");
```

Las funciones de esta clase empleadas para generar los archivos de resultados son:

- *GetSource*, retorna la dirección IPv4 de origen.
- *GetDestination,* devuelve la dirección IPv4 destino de la ruta.
- *GetGateway,* retorna la dirección IPv4 de la puerta de enlace (siguiente salto).

- **Ipv4InterfaceContainer:** Crea un contenedor entre las interfaces de los nodos conectados y sus respectivas direcciones IPv4.

```
Ipv4InterfaceContainer interfaces = address.Assign (devices);
```

Aodv Module

Este módulo es el de mayor interés en este trabajo ya que es aquí donde se realizarán varias modificaciones para incluir el mecanismo de control de potencia, específicamente en la clase aodv-routing-protocol.
El protocolo de enrutamiento AODV que modela NS-3 está basado en el RFC 3561 [47] por lo que todos los valores configurados son referidos de este. Se define en el directorio src/AODV del simulador.
Este modelo implementa toda la funcionalidad del servicio de intercambio de paquetes del protocolo para el descubrimiento de ruta y mantenimiento de la misma. Está formado por siete clases, de las cuales nos enfocaremos en las siguientes:

- **Aodv-routing-protocol**

Esta clase es la que define el funcionamiento del protocolo en sí, vincula seis clases adicionales del módulo AODV para llevar a cabo la operación del protocolo.

Incluye una serie de funciones por lo que solo se hará referencia a las funciones que fueron creadas en esta clase para este proyecto, mismas que se describen más adelante en la sección 2.2.1.

- **Aodv-neighbor**

Clase enfocada a los vecinos que detecta el protocolo AODV, posee funciones vinculadas a la búsqueda, actualización y eliminación de vecinos. En esta clase también se creó una función, la cual se detalla en la sección 2.2.1 *número de vecinos de AODV*.

- **Routing Table Entry**

Clase que modela el funcionamiento de la tabla de enrutamiento. Las funciones empleadas de esta clase son [76]:

- *GetHop*, devuelve el número de saltos, y lo hace en base a la variable "m_hops", que es la que almacena el número de saltos para llegar al destino de acuerdo a la ruta descubierta.
- *GetNextHop*, Devuelve la dirección IPv4 del siguiente salto.

Applications Module

Este módulo permite agregar varias aplicaciones (funcionalidades) a los nodos de una red. En este caso se ha usado una aplicación para definir el tipo de tráfico que se enviará entre los nodos participantes. A continuación, se describen las clases empleadas para generar tráfico UDP entre los nodos a través de una aplicación cliente/servidor:

- **UdpServerHelper**

Crea una aplicación de servidor la cual espera paquetes UDP de entrada y utiliza la información transportada en su carga útil para calcular retrasos y así determinar si algún paquete se perdió en la transmisión.

```
1  UdpServerHelper Server (9);
2  ApplicationContainer serverApps = Server.Install (nodes.Get(100));
3  serverApps.Start (Seconds (1.0));
4  serverApps.Stop (Seconds (200));
```

En la primera línea se muestra la activación del servidor UDP y la definición del puerto que usará; se lo almacena en la variable *Server*, la segunda línea instala la aplicación de servidor en el nodo 100; la tercera y cuarta línea muestran el tiempo inicial y final de simulación para el servidor.

- **UdpClientHelper**

Crea una aplicación cliente que envía paquetes UDP con un número de secuencia de 32 bits y un sello de tiempo de 64 bits.

```
1   UdpClientHelper Client (interfaces.GetAddress (100), 9);
2   Client.SetAttribute ("MaxPackets", UintegerValue (100));
3   Client.SetAttribute ("Interval", TimeValue (Seconds (0.09)));
4   Client.SetAttribute ("PacketSize", UintegerValue (1024));
5   ApplicationContainer clientApps = Client.Install (nodes);
6   clientApps.Start (Seconds (2.0));
7   clientApps.Stop (Seconds (200));
```

La primera línea muestra la configuración del cliente, mismo que hará peticiones al puerto 9 de la IP del nodo servidor; de la segunda a la cuarta línea se definen parámetros como el número de paquetes, tamaño de paquete e intervalo de envío. La quinta línea indica la instalación de la aplicación cliente en los nodos y finalmente se configura el tiempo inicial y final de simulación del cliente.

Mobility Module

Este módulo permite activar diferentes modelos de movilidad para los nodos, para este trabajo se ha implementado un modelo propio de NS-3 [58] y dos modelos se han generado con una herramienta externa llamada Bonnmotion. Las clases empleadas se muestran a continuación:

- **MobilityHelper**

Es una clase ayudante que permite asignar posiciones y modelos de movilidad a los nodos NS-3. Para activar este ayudante se usa:

```
MobilityHelper nombre_del_ayudante;
```

Las funciones necesarias de esta clase para este trabajo son:

- *MobilityHelper::SetMobilityModel,* configura el modelo de movilidad seleccionado. En este caso para el escenario preliminar de una red ad-hoc se usa el modelo ConstantPosition descrito en la sección 2.1.3.

```
SetMobilityModel ("ns3::ConstantPositionMobilityModel");
```

- *MobilityHelper::SetPositionAllocator,* configura la asignación de la posición de cada nodo durante el MobilityModel :: Install.

- *MobilityHelper::Install,* Instala el modelo de movilidad en cada nodo de acuerdo con el asignador de posición actual.

```
Install (nodes);
```

- **ListPositionAllocator**

Esta clase permite asignar posiciones a una lista de nodos definidos por el usuario; para el tercer escenario de simulación se ha usado esta clase para definir diferentes distancias entre los nodos de una forma lineal. Una muestra para configurar una posición es:

```
positionAlloc->Add (Vector (50.0, 0.0, 0.0));
```

- **Ns2-mobility-helper**

Esta clase es un ayudante que permite leer archivos de traza ns-2 y los convierte en eventos de movilidad de NS-3 para así configurar la movilidad en los nodos, por ende, permite importar a NS-3 la traza generada desde Bonnmotion. La clase se comporta de la siguiente manera:

1. Primero crea un objeto Ns2MobilityHelper con el archivo de traza especificado ".ns_movements".
2. Crea un archivo de registro utilizando el argumento de nombre de archivo de registro.
3. Se crea un grupo de nodos con el número de nodos especificados en la línea de comandos.
4. Se establece la movilidad en nodos mediante Install de Ns2MobilityHelper. En este momento, el archivo se lee línea por línea, y el movimiento es programado en el simulador.

Las siguientes líneas de código son necesarias para usar correctamente la clase Ns2MobilityHelper:

```
Ns2MobilityHelper ns2 = Ns2MobilityHelper (traceFile);
ns2.Install ();
```

- **Bonnmotion**

Es un software libre basado en Java [77] que permite generar y analizar escenarios móviles, es una herramienta de investigación muy útil para redes móviles ad-hoc. Sus

escenarios pueden ser importados por varios simuladores de red como: NS-2, NS-3, QualNet [78], ONE [79], entre otros.

BonnMotion está siendo desarrollado conjuntamente por algunas universidades como la Universidad de Bonn en Alemania, la Escuela de Minas de Colorado, USA, entre otras. A continuación, se muestran las principales consideraciones para generar una traza de movilidad con Bonnmotion.

Instalación

Para instalar Bonnmotion se ha hecho uso de la misma máquina virtual usada para correr NS-3, es decir se instala sobre una distribución de Ubuntu 14.04.1; el proceso es el siguiente:

1. Descargar Bonnmotion v3.0.0 de [80]
2. Instalar Java v.1.8
3. Descomprimir el archivo de Bonnomtion descargado

   unzip bonnmotion-2.0.zip

4. Ir a Bonnmotion e instalar
   ./install

Parámetros de una traza Bonnmotion

Para la generación de la traza de movilidad los parámetros se pueden ingresar por dos métodos; introducir los parámetros directamente en la línea de comandos o a su vez agregarlos en un archivo.

Los parámetros comunes para todos los modelos de movilidad son los siguientes:

- Número de nodos: “n”
- Duración del escenario [s]: “d”
- Tiempo de inicialización [s]: “i”
- Ancho del área de simulación [m]: “x”
- Largo del área de simulación [m]: “y”
- Profundidad del área de simulación [m]: “z”
- Controla la dimensión del escenario: -J <2D|3D>

Todos los parámetros específicos para cada escenario se los puede conocer mediante la línea de comando:

```
./bm –hm “Nombre del modelo”
```

Generación de trazas con Bonnmotion

Para este estudio, se generan varias trazas de movilidad con diferentes modelos. A continuación, se muestra el proceso necesario para generar la traza directamente desde la línea de comandos en el directorio /bin como sigue:

1. Definir todos los parámetros y el modelo con los cuales se va a crear la traza. Por ejemplo:

   ```
   ./bm -f traza Modelo -i  -n  -r  -x  -y  -a  - p  - c
   ```

2. Convertir a formato NS2 la traza generada

   ```
   ./bm NSFile -f traza
   ```

Con esta conversión se generan tres archivos con diferentes extensiones: .params, .ns_params, .ns_movements, de los cuales este último es el de interés por ser el que contiene la traza.

Una vez generada la traza se la puede importar desde el escenario definido en NS-3. Para esto el simulador de red posee una clase denominada *Ns2MobilityHelper*, la cual permite analizar archivos NS2 y convertir las declaraciones en eventos de movilidad NS-3, esta clase fue descrita en la sección 2.1.4 *Mobility Module.*

Modelos disponibles

Este software ofrece varios modelos de movilidad referidos a topologías malladas, área de desastres, redes estáticas y aleatorias, algunos de ellos son:

- *Column Mobility Model:* Es un modelo que considera un conjunto de nodos [81] distribuidos de forma lineal los cuales se desplazan uniformemente hacia adelante en una dirección en particular. El número de nodos y de grupos es configurable. Todos los grupos de nodos deben estar completos, sin nodos independientes. La columna de puntos de referencia selecciona un ángulo de orientación aleatorio y un vector de movimiento aleatorio. Los nodos siguen su punto de referencia individual a través de un mapa.

- *Original Gauss-Markov model:* Este modelo se basa en [82]. En esta implementación, la velocidad media no se especifica directamente, a diferencia de la

velocidad máxima que si puede ser definida. El modelo ha sido adaptado para hacer frente a los escenarios de borde en el caso de que un nodo se mueva sobre el borde, su vector de velocidad, así como su vector de velocidad esperado son reflejados.

- *ManhattanGrid:* Este es uno de los modelos de movilidad implementados en este proyecto [83]. El modelo representa una red de calles que modelan una parte de una ciudad; se considera para modelar movimientos realistas en áreas urbanas ya que su topología es en forma de malla, donde las calles son organizadas a través de una cuadrícula en dirección horizontal y vertical asemejándose a un área urbana. En la intersección de las calles (filas y columnas) el nodo selecciona entre la probabilidad de moverse en la misma calle es decir en la misma dirección y la probabilidad de cambiar de dirección a la izquierda o derecha. Los nodos únicamente se mueven por caminos predefinidos.

  Bonnmotion tiene una serie de parámetros para generar una traza de movilidad con este modelo, entre ellos: -u -v configuran el número de bloques entre los caminos, -u el número de columnas y -v el número de filas. La velocidad del nodo es configurable a través de -e -m -o para definir la velocidad mínima promedio o máxima respectivamente.

  El modelo que emplea este software contiene dos modificaciones:
  1. Incluye la posibilidad de configurar la velocidad mínima para un nodo.
  2. Permite configurar pausas en los nodos

- *Modelo Random Waypoint:* [84] Este modelo aleatorio es ampliamente usado para el estudio de redes ad-hoc; el movimiento de los nodos se rige de la siguiente manera: de inicio cada nodo está pausando durante un número fijo de segundos, luego el nodo determina un destino aleatorio dentro del área de simulación y una velocidad aleatoria entre 0 y una velocidad máxima. El nodo se mueve hacia el destino seleccionado y de nuevo hace una pausa por un período fijo antes de tomar otra ubicación aleatoria y seleccionar la velocidad. Cada nodo se mueve en línea recta tomando una dirección en forma de zigzag tomando pausas en cada punto de referencia, este comportamiento se mantiene hasta que la simulación termine.
  En Bonnmotion este modelo permite crear escenarios 2D y 3D con solo agregar la componente *-z*. Incluye una característica para restringir los movimientos móviles, por ejemplo: Con -d 1, los nodos se mueven sólo a lo largo del eje x, con -d 2, los

nodos se mueven a lo largo de la x o a lo largo del eje y (con una probabilidad de 0,5 cada uno) y con -d 3 que es la configuración por defecto se define el modelo clásico 2D de Random Waypoint. Para escenarios 3D, existen dos opciones más: Con -d 4, los nodos se mueven en x o y o z y con -d 5 se obtiene el comportamiento clásico de Random waypoint en 3D. Cada uno de los parámetros configurables para este modelo en Bonnmotion los podemos observar con la línea:

```
./bm -hm RandomWayPoint
```

Configuración de una red Ad-hoc en NS-3

En esta sección se describe el proceso para crear el escenario de la red ad-hoc que se maneja en este trabajo. Se muestra la creación de nodos, y la definición de la arquitectura de red de los dispositivos. Las capas deben ser creadas desde capa inferior (canal inalámbrico) hasta el dispositivo de red (capa superior); a continuación, se detalla el proceso.

1. **Seleccionar el modelo de capa física,** de esto dependerá el canal y el medio físico. Existen dos opciones: SpectrumWifiPhy o YansWifiPhy. Para el presente trabajo se usa:

```
YansWifiPhyHelper wifiPhy = YansWifiPhyHelper::Default ();
```

   Este ayudante implementa un modelo de capa física de 802.11a. por defecto, lo cual indica que usa OFDM y se maneja en la banda de 5GHz.

2. **Configurar el canal inalámbrico (WifiChannel)**, para este proyecto se usa el YansWifiChannelHelper que permite crear y manejar un canal inalámbrico con el LogDistancePropagationLossModel como modelo de pérdida por propagación y el ConstantSpeedPropagationDelayModel como un modelo de retardo de propagación. Las líneas de código configuradas son:

```
YansWifiChannelHelper wifiChannel = YansWifiChannelHelper::Default();
wifiPhy.SetChannel (wifiChannel.Create ());
```

3. **Configurar capa MAC (WifiMac),** al configurar capa mac se define el tipo de arquitectura y el nivel del dispositivo. En este caso se ha usado el tipo de red ad-hoc, para lo cual se configura el ayudante de capa MAC como sigue:

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

4. **Crear el dispositivo inalámbrico (WifiNetDevice),** para esto se configura el wifi Helper como sigue:

```
WifiHelper wifi;
wifi.SetRemoteStationManager ("parametros”);

devices = wifi.Install (wifiPhy, wifiMac, nodes);
```

5. **Habilitar la pila de protocolos TCP/IPv4 y direccionamiento IPv4.**

```
Se habilita la pila de protocolos TCP/IPv4
InternetStackHelper
Se habilita direccionamiento IPv4
Ipv4AddressHelper
```

6. **Habilitar enrutamiento con AODV**

```
Se activa el ayudante de AODV
AodvHelper aodv;
Habilita el enrutamiento
stack.SetRoutingHelper (aodv);
```

7. **Instalar aplicaciones,** en este caso solo se usará una aplicación para generar tráfico UDP entre cliente y servidor.

```
Se habilita los ayudantes Cliente y Servidor UDP
UdpServerHelper
UdpClientHelper
```

* Adicional se puede configurar algún tipo de movilidad en los nodos, lo cual se detalla en la sección 2.1.4 *Mobility Module.*

El código de los escenarios generados para este proyecto se adjunta en el *ANEXO IV*.

2.2 CONTROL DE POTENCIA

Esta sección es la más compleja e interesante del presente estudio, pues es la muestra del trabajo realizado para lograr la modificación del protocolo de enrutamiento AODV en el simulador de red y con ello cumplir el objetivo del proyecto. La sección detalla los procesos, acciones, e implementaciones que fueron necesarios para lograr el control de la potencia de transmisión a nivel de capa de red.

Se propone adicionar un mecanismo para controlar la potencia de transmisión en base a la detección de vecinos del protocolo AODV usando un criterio de Cross Layering para permitir coordinación, interacción e intercambio de información entre capa física y capa de red.

En la primera parte se analiza el protocolo de enrutamiento para buscar la manera de establecer un vínculo con capa física y así tener coordinación entre las dos capas y poder manejar la potencia de transmisión. Luego será necesario definir algunas funciones para obtener información de AODV y una función para conseguir la actualización periódica del mecanismo con la finalidad de obtener información del protocolo de enrutamiento actualizada.

La segunda parte de esta sección analiza el diseño del mecanismo de control de potencia, muestran las consideraciones para el cálculo de los parámetros establecidos en el mecanismo y como actuará en la red.

A continuación, se muestra cada una de las etapas que se desarrollaron.

2.2.1 Modificación - AODV Routing Protocol

Aodv-routing-protocol es una clase de NS-3 que modela el funcionamiento del protocolo de enrutamiento AODV, desde este se controlará la potencia de transmisión en la MANET a implementar; es necesaria su modificación para incluir el mecanismo de control de potencia y algunas funciones que permitan la operación del mismo.

Esta modificación considera tres etapas principales; acceso a capa física, obtención del número de vecinos del protocolo de enrutamiento, y actualización periódica del mecanismo para la detección de vecinos.

En la primera etapa se analiza el protocolo de enrutamiento para buscar la manera de establecer un vínculo entre este y capa física (que maneja la potencia de transmisión) considerado la idea del Cross Layering con el fin que la potencia de transmisión se configure en base a cierta información que emita el protocolo de enrutamiento.

Una vez que se acceda a capa física, ya es posible manipular la potencia de transmisión y el mecanismo de control de potencia ya puede ser diseñado. Tomando en cuenta que la base del mecanismo será el número de vecinos que detecte AODV, la segunda etapa estará dedicada al estudio de la clase *aodv-neighbor*, para a través de esta obtener toda la información de vecinos necesaria para incluirla en el mecanismo.

Finalmente, es necesario definir una función que ejecute el mecanismo de forma periódica con el fin de obtener la información de vecinos actualizada.

Acceso a capa física

El control de potencia es una actividad propia de capa física. Sin embargo, en este trabajo lo controlaremos desde capa de red con el objetivo de mejorar el rendimiento de AODV y de la red. Para esto es necesario establecer un vínculo entre capa física y capa de red a manera de poder modificar la potencia y tener el control requerido desde AODV.

NS-3 nos permite activar punteros inteligentes entre objetos, y, considerando que las capas de la arquitectura de red están representadas por objetos, es posible conseguir un puntero a capa física desde capa de red creando una especia de "cadena" para que estas capas se comuniquen.

- **Punteros implementados**

Para acceder a la capa física del simulador desde capa de red es necesario generar una secuencia de punteros inteligentes entre los objetos que representan las diferentes capas, tal como lo muestra la Figura 2.9. Para esto es necesario identificar en el *aodv-routing-protocol* los punteros en los cuales están almacenados los objetos que representan la interfaz de red del dispositivo, la tarjeta inalámbrica y capa MAC.

```
1  Ptr <NetDevice> dev = m_ipv4 -> GetNetDevice(m_ipv4->
2  GetInterfaceForAddress(iface.GetLocal()));
3  Ptr <WifiNetDevice> wifi = dev-> GetObject<WifiNetDevice> ();
4  Ptr <WifiMac> mac = wifi-> GetMac ();
```

Figura 2.9. Punteros existentes en aodv-routing-protocol.

Como se puede ver en la Figura 2.9, la línea 1 muestra un puntero a la interfaz de red del dispositivo denominado *dev*, la línea 3 contiene el puntero *wifi* a la tarjeta de red inalámbrica con WiFi; la última línea muestra el puntero a capa MAC denominado *mac.* Estos tres punteros que se muestran forman parte del código original de *aodv-routing-protocol.*

Una vez identificados estos punteros, es importante conocer la clase que se quiere alcanzar en capa física con la cadena de punteros. El objetivo en este estudio es llegar a la clase *YansWifiPhy*.

YansWifiPhy es una clase que representa un modelo de capa física de 802.11 a; incluye varias funciones que permiten configurar parámetros de capa física; en este caso las funciones de interés son aquellas que se relacionan con la potencia de transmisión, específicamente: *SetTxPowerStart, SetTxPowerEnd, GetTxPowerStart, SetTxLevel* y *EnergyDetectionThreshold* las cuales se definen en sección 2.1.4 *YansWifiPhy.*

Para alcanzar las capas inferiores que son el objetivo de esta etapa se han creado dos punteros adicionales, los cuales deben vincularse con los punteros antes mencionados. Estos son:

```
1   Ptr<WifiPhy> phy = mac->GetWifiPhy();
2   Ptr<YansWifiPhy>phyless= phy ->GetObject<YansWifiPhy> ();
```

Figura 2.10. Punteros a capa física.

En la Figura 2.10, la línea 1 muestra un puntero a capa física denominado *phy*¸ el cual se vincula a capa MAC mediante *GetWifiPhy*; la segunda línea define el puntero *phyless*, el cual apunta a la clase *YansWifiPhy;* se vincula a capa física a través de *GetObject<YansWifiPhy>*.

Con estos cinco punteros se define la secuencia lógica entre las diferentes capas y con ellos se obtiene el acceso a capa física desde AODV.

La Figura 2.11 es una representación de los punteros que se implementó en aodv-routing-protocol para alcanzar capa física y así acceder a la clase ns3::YasWifiPhy para modificar la potencia.

El código usado para lograr el acceso a capa física desde el protocolo de enrutamiento AODV se encuentra adjunto en el ANEXO III, sección *ACCESO A CAPA FÍSICA*.

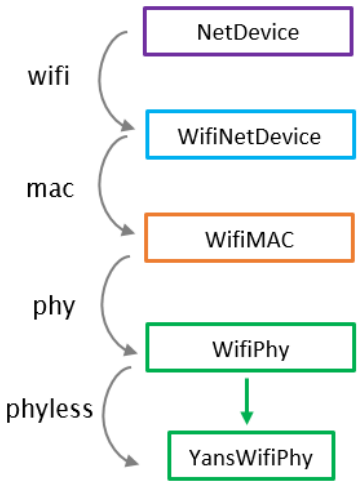

Figura 2.11. Representación de la cadena de punteros para capa física**.**

Número de vecinos de AODV

AODV detecta los posibles receptores de la información llamados “vecinos” o “neighbors” (en inglés) en base al rango de transmisión de los nodos, movilidad y además de la potencia de transmisión. Si la potencia se incrementa, la detección de vecinos será más

amplia y se darían saturaciones de canal, contención y con ello colisiones. Por el contrario, si se actúa con una potencia menor podrían detectarse muy pocos vecinos o ninguno lo cual llevaría a una transmisión fallida. Con estas consideraciones se propone un control de potencia en base al número de vecinos que detecte el protocolo de enrutamiento AODV. Con esto se podría mejorar el rendimiento del protocolo y consecuentemente de la red.

El modelo de AODV que maneja NS-3 refiere toda la información de vecinos a la clase *aodv-neighbor*. Por tal, es necesario analizar dicha clase para en ella agregar una función que permita obtener el número de vecinos que tiene cada uno de los nodos participantes en la MANET. En base a esta información variará el mecanismo de control de potencia.

La nueva función que se agregará guarda relación con una estructura que incluye datos importantes sobre los vecinos de AODV, a continuación, se describe.

- **Almacenamiento de vecinos**

La clase *aodv-neighbor* contiene una estructura denominada *Neighbor*, la cual incluye cierta información de vecinos, tales como: direcciones MAC, IP y tiempo de vida de un vecino lo cual sería útil en caso de que se requiera conocer información específica sobre algún vecino de los nodos participantes. Esta estructura, está asignada a un vector definido como "*m_nb*", el cual almacena todos los vecinos que detecte AODV, en NS-3 se lo conoce como vector de entradas.

```
1  struct Neighbor
2    {
3      Ipv4Address m_neighborAddress;
4      Mac48Address m_hardwareAddress;
5      Time m_expireTime;
6      bool close;
7      Neighbor (Ipv4Address ip, Mac48Address mac, Time t) :
8      m_neighborAddress(ip),m_hardwareAddress(mac),m_expireTime(t),
9      close (false)
10     {
11     }
12   };
13   std::vector<Neighbor> m_nb;
```

Figura 2.12. Código de la estructura Neighbor y vector de vecinos

En las líneas 1-12 de la Figura 2.12 se muestra la estructura Neighbor. En la primera línea se define de la estructura; en las líneas 3 y 4 se declara las variables que almacenan la dirección IPv4 y MAC de los vecinos respectivamente. La línea 14, que es

la de mayor interés muestra la definición del vector *m_nb*, en el cual se almacenarán todos los vecinos.

- **Función NumVecinos**

La nueva función que se agregará en la clase *aodv-neighbor* para obtener el número de vecinos de cada uno de los nodos de la MANET se denominará *NumVecinos*.
Es posible obtener el número de vecinos gracias a que *m_nb* (véase sección anterior) está definido como un vector a la estructura *Neighbor*, por tal se hace uso de la función *size* propia de *vector* [85] para obtener el tamaño del vector *m_nb* lo cual resultaría en el número de vecinos almacenados en el mismo.
La función *NumVecinos* se configura para que retorne el size del vector *m_nb* y se define en la clase *aodv-neighbor* como sigue:

```
int NumVecinos () { return m_nb.size(); }
```

Para llamar a la función en el *aodv-routing-protocol*, se usa:

```
m_nb.NumVecinos();
```

Este dato es de mucha importancia para este estudio ya que en base a este parámetro el mecanismo de control de potencia tomará decisiones sobre la variación en la potencia de transmisión.

Actualización periódica del mecanismo

Dado que en una MANET la topología de red es variable a cada instante por el movimiento continuo de los nodos, es necesario que el mecanismo de control de potencia se ejecute periódicamente para que tome la información actualizada de AODV (nodo origen, nodo destino, número de vecinos, siguiente salto, tiempos, etc), principalmente es importante obtener el número de vecinos actualizado.
Para esto se crea una función en *aodv-roting-protocol* denominada *UpdateMControl* dentro de la cual se definirá el mecanismo de control de potencia y se ejecutará de forma periódica cada 1 segundo. Arrancará su operación en el segundo 1.1 inmediatamente después de que actúen los paquetes de Hello en la red (El paquete de Hello inicia al 1 segundo).
El proceso para crear la función mencionada se describe a continuación.

1. Declarar la nueva función y la variable de temporizador que será el pivote de la actualización periódica.

```
1  void UpdateMControl ();
2  Timer m_upmcontrol;
```

Figura 2.13. Código para declarar la función UpdateMControl y su variable de timer

La Figura 2.13 en la línea 1 muestra la declaración de la función bajo el nombre de *UpdateMControl* y la segunda indica la declaración de una variable tipo *Timer* que servirá para configurar el tiempo de actualización periódica de la función.

2. Definir el argumento de la variable de temporizador

```
Argumento para la variable de temporizador m_upmcontrol
m_upmcontrol (Timer::CANCEL_ON_DESTROY),
```

3. Definir la función; esta función alojará al mecanismo de control de potencia y algunas de las configuraciones que fueron necesarias para la operación del mismo.

```
1  void
2  RoutingProtocol:: UpdateMControl ()
3  {
4    NS_LOG_FUNCTION (this);
5
6     // DIRECCIONES IP
7     // ACCESO A CAPA FISICA
8     // MECANISMOS DE CONTROL DE POTENCIA
9     // LOGS
10   m_upmcontrol.Schedule (Seconds (1));
11  }
```

Figura 2.14. Definición de la función UpdateMControl.

La Figura 2.14 muestra en la línea 2 la definición de la función *UpdateMControl* en el aodv-routing-protocol. Las líneas 6 – 9 representan las diferentes configuraciones del mecanismo de control de potencia. La línea 10 indica el tiempo de actualización de la función, es decir la función se ejecutará cada 1 segundo.

4. Llamar a la nueva función, este llamado se lo realiza dentro de la función Start que es la que inicia la operación del protocolo de enrutamiento.

```
1  Void
2  RoutingProtocol::Start ()
3  {
4    //Llamado a la nueva función UpdateMControl
5     m_upmcontrol.SetFunction(&RoutingProtocol::UpdateMControl,this);
6    //La función inicia al segundo 1.1
7     m_upmcontrol.Schedule (Seconds (1.1));
8  }
```

Figura 2.15. Llamado a la función UpdateMControl desde aodv-routing-protocol.

La línea 7 de la Figura 2.15 define el tiempo en el cual iniciará la ejecución de la función, en este caso el tiempo de inicio se ha configurado en el segundo 1.1 ya que se quiere que el mecanismo de control de potencia opere luego de que los paquetes de hello hayan actuado en la red. Los paquetes de hello inician al segundo 1.

2.2.2 Definición del mecanismo de control de potencia

Luego de realizar varias modificaciones en el modelo del protocolo de enrutamiento AODV (incluir cadenas de punteros, acceso a capa física, conteo de vecinos, función de actualización, entre otros), es posible adicionar el mecanismo de control de potencia. Esta sección se enfoca en detallar las consideraciones tomadas para definir el mecanismo de control de potencia haciendo referencia a varios factores como, por ejemplo: número de vecinos, área de simulación, rango de transmisión (el cual depende del tipo de propagación del canal), etc.

El mecanismo que controlará la potencia de transmisión depende de aspectos fundamentales que se definieron en la sección anterior; como: acceso a capa física, obtención del número de vecinos de AODV y la función de actualización periódica como se ve en la Figura 2.16.

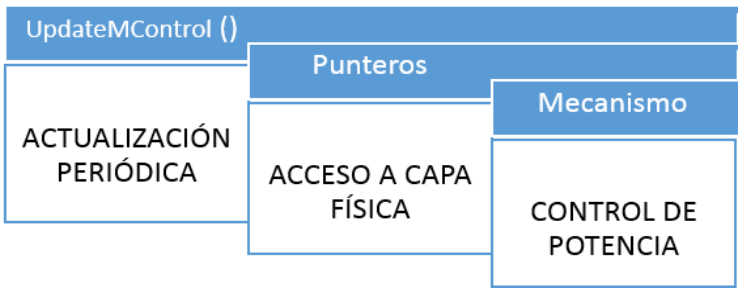

Figura 2.16. Dependencias del mecanismo

En la Figura 2.16, se puede ver que la función de actualización periódica "*UpdateMControl*" será la base del mecanismo de control de potencia ya que dentro de esta función se insertará la cadena de punteros que alcanzan capa física y el mecanismo propiamente dicho.

Operación del mecanismo

El mecanismo de control de potencia se ha diseñado para ser aplicado en MANETs que tengan como protocolo de enrutamiento AODV y que manejen distribuciones de nodos

simétricas o muy semejantes a ello dentro de un área limitada como se verá más adelante.

Como ya se indicó en secciones anteriores, el mecanismo opera según el número de vecinos que tenga cada nodo; es decir la potencia de transmisión se ajusta en base a los resultados de detección de vecinos que muestre el protocolo de enrutamiento AODV. Para conocer el valor de potencia que el mecanismo deberá incrementar o disminuir según los vecinos que detecte es necesario considerar parámetros como: área de simulación, rango de transmisión, alcance, potencia mínima, potencia máxima, entre otros.
Las operaciones matemáticas que realiza el mecanismo se harán en milivatios [mw] y una vez que se haya calculado la nueva potencia de transmisión para el nodo se realizará la conversión a dBm dado que la potencia configurable que admite NS-3 se mide en dBm. Finalmente, la potencia debe ser configurada en la capa física del nodo.

- **Lógica del mecanismo**

El mecanismo tiene una lógica condicional en base al número de vecinos; así según este parámetro se toman decisiones para elevar o reducir la potencia de transmisión. A continuación, la Figura 2.17 muestra un diagrama de flujo del mecanismo de control de potencia, en la cual se puede apreciar que, como primer paso, previo a realizar algún cálculo con la potencia, es necesario acceder a capa física. Para esto se configuran los punteros que se definieron en la sección anterior.

Como segundo paso se definen variables para almacenar los valores de potencia, número de vecinos, y el factor de adición para potencia. Las variables definidas son:

- *pot_act_d:* Esta variable es de tipo *doble* ya que almacenará la potencia de transmisión inicial que se tiene por defecto configurado en NS-3. El valor de la potencia se lo obtiene mediante la función *GetTxPowerStart.*

```
double pot_act_d=phyles->GetTxPowerStart ();
```

- *num_n:* Esta variable de tipo *entero* se la define para almacenar el número de vecinos de cada nodo.

```
int num_n=m_nb.NumVecinos();
```

- *pot_act_mw:* Esta variable de tipo *doble* permite almacenar la potencia inicial en [mw], esto ya que para realizar las operaciones entre potencias se manejará [mw].

```
double pot_act_mw;
```

- *p_mw:* Variable de tipo *doble* que permite almacenar la suma total de potencias en [mw]. Es útil al momento de aumentar o disminuir potencia del valor inicial.

```
double p_mw;
```

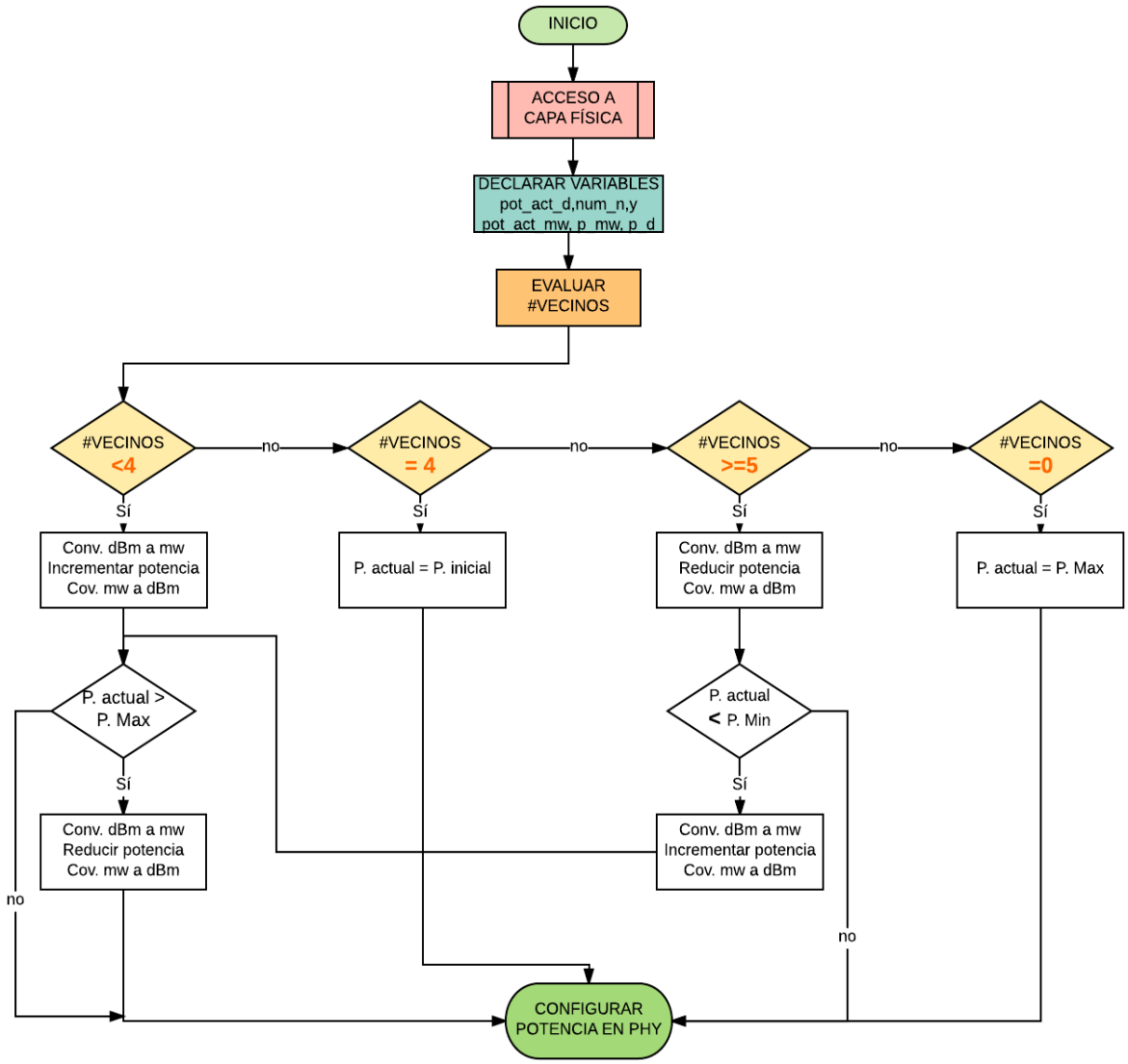

Figura 2.17. Flujograma del mecanismo de control de potencia

- *p_d:* Esta variable de tipo *doble* almacena el valor de la nueva potencia actual en [dBm]. Esta es la potencia que se configurará al final del mecanismo.

```
double p_d;
```

- *delta_pow:* La última variable que se define permite almacenar un valor fijo en [mw] el cual se sumará o restará de la potencia actual para así generar las variaciones de potencia.

```
double delta_pow=2;
```

Luego de definir las variables, el mecanismo inicia la evaluación del número de vecinos. Como se puede ver en la Figura 2.17 existen cuatro casos posibles. Para este trabajo se ha considerado que el número "ideal" de vecinos para un nodo sea de 4. Se ha elegido este número puesto que los escenarios de prueba manejan un área que asume una distribución simétrica de nodos, por tanto, se asume 1 nodo vecino por cada punto cardinal.

- *CASO 1:* El número de nodos vecinos detectados es menor que 4. Si un nodo presenta este caso el mecanismo incrementará un valor predefinido "y" *(*del cual se habla más adelante*)* a la potencia de transmisión actual del nodo.

- *CASO 2:* Los nodos vecinos detectados son en total 4. Ante este caso, se considera que el nodo tiene la cantidad de vecinos "ideal", por tanto, el mecanismo no hará más que configurar como valor de potencia actual el valor de potencia que presente el nodo en ese instante.

- *CASO 3:* Los vecinos detectados son mayores o igual a 5. En este caso el mecanismo disminuirá la potencia actual del nodo. Esto lo hará restando un valor predefinido "*y"* a la potencia.

- *CASO 4:* En este último caso se considera la posibilidad de que un nodo no haya podido detectar ningún vecino; de suceder, el mecanismo configurará la potencia actual del nodo en el valor asignado para la máxima potencia de transmisión.

En los casos 1 y 3 se considera tanto la mínima potencia de transmisión como la máxima potencia de transmisión. De estos valores y del valor de *delta_pow*, se habla en la siguiente sección.

- **Parámetros del mecanismo**

Existen algunos valores que el mecanismo de control de potencia toma para su operación. Estos valores hacen referencia a los siguientes parámetros:

- *Factor de adición:* Este factor indica cuánto se debe agregar o sustraer a la potencia actual de un nodo según sea el caso; se lo representa con la variable "*delta_pow*". Este valor se calcula en base al rango de transmisión (que depende del modelo de pérdidas por propagación del canal), throughput y sensibilidad. Por este concepto "y" tendrá el mismo valor en todos los escenarios. A continuación de detalla el proceso realizado para determinar el valor de la variable "*delta_pow".*

En base a pruebas realizadas se pudo determinar que la distancia máxima a la cual se puede transmitir sin pérdida de paquetes bajo la arquitectura de red considerada para este proyecto es de 113 m. A los 114 m ya se pierde un paquete, a los 115 m son dos y así depende del modelo de pérdidas por propagación. Para el mecanismo de este proyecto se considerará como referencia de pérdida la distancia de 115 m.

Para determinar la potencia necesaria para transmitir a una distancia de 115 m sin pérdida de paquetes primero se debe conocer la mínima potencia requerida para transmitir sin pérdida de paquetes (potencia a la distancia máxima de 113m), la cual se obtiene como sigue:

Por definición de potencia de recepción se tiene:

$$P_{Tx} - L_d = P_{Rx}$$

Ecuación 2.4. Potencia de recepción

Donde:

P_{Tx}: $Potencia\ de\ transmisión$

P_i: $Potencia$ de transmisión $por\ defecto$ de $NS3 = 16.026$ dBm

L_d: $Pérdidas\ por\ propagación\ a\ un\ distancia\ d$

P_{Rx}: $Potencia\ de\ recepción$

P_{Rmin}: $Mínima\ potencia\ de\ recepción\ necesaria\ para\ una\ transmisión$ sin $pérdidas.$

Adaptando la Ecuación 2.4 a este análisis se tiene:

$P_i - L_{d=113} = P_{Rx}$ (1)

Recordando la Ecuación 2.3, se calcula la pérdida por propagación a una distancia de 113m como sigue:

$L_{d=113} = L_o + 10nlog\left(\frac{d}{d_o}\right)$ (2)

$L_{d=113} = 46.677 + 10\ (3)log\left(\frac{113m}{1m}\right) = 108.269\ dB$

Reemplazando 2 en 1 se tiene:

$16.026\ dBm - 108.269\ dB = -92.243\ dBm$

Por tanto, la potencia de recepción de un nodo considerando las pérdidas generadas a una distancia de 113m es, y tomando en cuenta que a esta distancia no se tiene pérdida alguna de paquetes, se toma esta potencia como referencia para

establecer la *mínima potencia de recepción necesaria para tener una transmisión sin pérdidas*.

$P_{Rmin} = -92.243\ dBm$ (3)

La sensibilidad que maneja el simulador NS-3 es de -96 dBm; sin embargo, por las pruebas experimentales realizadas en topologías arbitrarias se pudo identificar que a -96dBm la información puede recibirse, pero con muchas pérdidas, es por esto que para el diseño del mecanismo de control de potencia se tomará como la mínima potencia de recepción para un nodo al valor calculado en 3.

Ahora, es necesario calcular la pérdida por propagación esperada a una distancia de 115m. Considerando la Ecuación 2.3 se tiene:

$L_{115} = L_o + 10nlog\left(\frac{d}{d_o}\right) = 108.498\ dB$ (4)

Una vez conocidos los valores de 3 y 4 ya se puede establecer una ecuación para determinar cuál sería la mínima potencia requerida para transmitir sin pérdidas a una distancia de 115 m.

$P_{Tx} - L_{115} = P_{Rmin}$
$P_{Tx} - 108.498\ dB = -92.243\ dBm$
$P_{Tx} = 16.255\ dBm = 42.218\ mw$ (5)

Al encontrar la potencia de transmisión necesaria para transmitir sin pérdidas a 115 m (5), ya se puede establecer una diferencia con la potencia de transmisión inicial que define el simulador de red y con ello se determina el valor del factor de adición *delta_pow.*

$delta_{pow} = P_{Tx} - P_i$
$delta_{pow} = 42.218\ mw - 40.049\ mw$
$delta_{pow} = 2.168\ mw$

Este valor se determinó en base al rango de transmisión de un nodo por lo que se tomará como un valor constante para el mecanismo de control de potencia en todos sus casos y se aproximará a:

$$\boldsymbol{delta_{pow} = 2\ mw}$$

- *Potencia mínima:* Es necesario establecer un límite inferior de potencia de transmisión ya que existen casos en los que el mecanismo disminuirá de forma gradual, y, en caso de no existir un límite es posible que la potencia tome valores

nulos o incluso negativos. Para esto el mecanismo incluye un parámetro que define la potencia mínima; la cual se calcula en base a la densidad de nodos de cada escenario y al área de simulación.

En este proyecto, el valor aproximado de la mínima potencia de transmisión de un nodo se calcula con una regla de tres simple, a continuación, se muestra un ejemplo para determinar la mínima potencia de transmisión con una densidad de 25 nodos.

Donde:

A: $Área\ de\ simulación$

δ: $Densidad\ de\ nodos$

P_i: $Potencia$ de transmisión $por\ defecto$ de $NS3$

$A = 500mx500m = 250000\ m^2$

$\delta = 25\ nodos$

$P_i = 16.026\ dBm = 40.049\ mw$

Debe considerarse que con una potencia de 40.049 mw se puede transmitir sin pérdida de paquetes a una distancia de hasta 113 m; por tanto, el área que se cubre con dicha potencia es de:

$A_{113m} = \pi R^2 = \pi * 113^2 = 40114.997\ m^2$

Ahora, el área total de simulación es de 250000 m^2 y tomando en cuenta que se tiene 25 nodos, a cada nodo le correspondería un área de aproximadamente de:

$A_{nodo} = \frac{250000m^2}{25\ nodos} = 10000\ {}^{m^2}/_{nodo}$

La regla de tres se establece como se muestra a continuación:

Potencia [mw]		Área [m^2]
40.049	→	40114.997
x	→	10000

Si una potencia de 40.049 mw cubre un área de 40114.997, la potencia requerida para cubrir el área de 10000m^2 que le correspondería a cada nodo es de:

$P_{mín} = \frac{40.049*10000}{40114.997} = 9.98\ mw = 9.99\ dBm$

De esta manera queda definida la mínima potencia de transmisión de un nodo considerando una densidad de 25 nodos. La Tabla 2.1 muestra la potencia mínima que se requerirá para cada densidad.

Tabla 2.1. Potencia mínima de transmisión conforme al número de nodos

Densidad [nodos]	Área aprox. [m^2/nodo]	Mín. Potencia Tx [dBm]
25	10000	9.99
36	6944.44	8.40
100	2500	3.97

- *Potencia máxima:* Es necesario considerar también un límite superior de la potencia de trasmisión. La máxima potencia de transmisión para el estándar IEEE 802.11a según varios artículos como [86] [87] es de 23 dBm. El libro de redes vehiculares ad-hoc [88] indica que el límite de la potencia de transmisión en redes ad-hoc debería ser de 20 dBm. Por tanto, para el mecanismo diseñado en este trabajo se considera como límite superior el valor de 20 dBm.

- **Conversión de potencia**

Todos los cálculos de potencias que se realicen se manejarán en mw ya que son sumas y diferencias. Cuando se haya determinado la potencia de transmisión actual se realizará una conversión de mw a dBm puesto que la potencia configurable en NS-3 solo admite dBm. La conversión realizada se muestra en las siguientes líneas:

```
pow(10,pot_act_d/10);

10*log10(p_mw);
```

En la primera línea se puede apreciar el uso de la función *pow* para obtener una potencia y así realizar la conversión de dBm a mw. La segunda línea muestra el proceso inverso; la conversión de mw a dBm mediante el uso de la función *log.*

- **Configuración de la potencia**

Finalmente la potencia calculada debe configurarse en la capa física del nodo; caso contrario, el nodo actuará con la potencia por defecto que tiene el módulo *Wifi* de NS-3.

```
phyless->SetTxPowerStart(p_d);
```

El puntero *phyless* accede a la capa física del nodo, por tanto debe llamarse en el la función *SetTxPowerStart* para configurar la potencia.

En el *ANEXO III-A*, se encuentra adjunto el código del mecanismo de control de potencia usado para cada densidad de nodos.

Variación del mecanismo para escenarios preliminares

Debido a que se realizará pruebas en escenarios preliminares, los cuales consideran una topología lineal, es necesario realizar variaciones en el mecanismo, mismas que se detallan a continuación.

- El número ideal de vecinos esperados es al menos dos nodos, ya que al ser un escenario lineal se espera que un nodo vea el nodo anterior y posterior. El condicional lógico se hará en base a este número.
- No es necesario establecer una potencia mínima ya que la transmisión se hará de extremo a extremo, es decir, el primer nodo transmite al último por lo que el mecanismo únicamente realizará sumas progresivas de potencia en pasos de *delta_pow* hasta el límite de la potencia máxima establecida.

El código de este mecanismo de control de potencia se lo encuentra en la sección de anexos, en el *ANEXO III-B.*

2.3 ESCENARIOS

Para analizar el desempeño de AODV con y sin un mecanismo de control de potencia se han definido varios escenarios ad-hoc sobre los cuales se han ejecutado las simulaciones.

Previo a los escenarios de prueba se consideran dos escenarios preliminares para verificar en ellos la validez del mecanismo de control de potencia.

Todos los escenarios creados definen el mismo diseño a nivel lógico, es decir manejan la misma pila de protocolos; la diferenciación de cada escenario está dado a nivel físico ya que el número de nodos, topología y patrón de movilidad varía para cada uno. A continuación, se describe cada uno de ellos.

2.3.1 Estructura de los escenarios a nivel lógico

Para generar un escenario que simule una red ad-hoc es necesario agregar una pila de protocolos para permitir la comunicación; así se crea un canal de comunicaciones, capa física, capa mac, capa de red, aplicaciones y el dispositivo en sí. Los parámetros

considerados en cada una de las capas mencionadas se resumen en la Tabla 2.2; la cual muestra todas las capas que se crearon en el simulador para dar funcionalidad a los nodos participantes. Asimismo, se muestran las clases y funciones de NS-3 que permiten la operación de cada una de ellas. En la sección 2.1.4 se puede apreciar una descripción más detallada de las clases necesarias para la creación del escenario.

Tabla 2.2. Consideraciones a nivel lógico de los escenarios.

ESTRUCTURA DE LOS ESCENARIOS			
EQUIVALENTE	**TIPO**	**CLASE NS-3**	**FUNCIÓN NS-3**
Canal	Inalámbrico	`YansWifiChannelHelper`	`Default`
Física	802.11 a	`YansWifiPhyHelper`	`Default`
Enlace	Ad-hoc	`WifiMacHelper`	`AdhocWifiMac`
Dispositivo	NIC: 802.11a	`WifiHelper`	`Default`
	Nodos	`NodeContainer`	`Create ()`
Red	TCP/IPv4	`InternetStackHelper`	`Install ()`
		`Ipv4AddressHelper`	`SetBase("IP","MK")`
	Enrutamiento	`AodvHelper`	`SetRoutingHelper ()`
Aplicación	Tráfico UDP	`UdpServerHelper`	`Server.Install ()`
		`UdpClientHelper`	`Client.Install ()`

Luego de definirse la configuración lógica de un escenario, es necesario agregar movilidad a los nodos participantes de la red ad-hoc para así trabajar en una MANET propiamente dicha; a continuación, se detallan los modelos de movilidad usados para los diferentes escenarios.

2.3.2 Diferenciación de escenarios a nivel físico

Los escenarios de simulación para este proyecto presentan diversas topologías, densidad de nodos y modelos de movilidad. Se ha considerado escenarios preliminares y escenarios de prueba; a continuación, se describen:

Escenarios preliminares

Se definen escenarios preliminares con el fin de probar el funcionamiento y validez del mecanismo del control de potencia; es claro que el mecanismo variará mínimamente para estos escenarios preliminares por el hecho de tener menor densidad de nodos y topología diferente, pero la idea que maneja el mecanismo es la misma. Se tienen dos escenarios preliminares:

- **Fijo**

Este escenario define una red ad-hoc con un grupo de 5 nodos fijos alineados de forma horizontal distanciados por diferente longitud entre ellos. La posición de los nodos se define con el *mobility module* de NS-3 y el modelo de movilidad que se usa para este escenario es el ns3::*ConstantPositionMobilityModel*, definido previamente en la sección 2.1.3. La Figura 2.18 es una muestra del escenario generado.

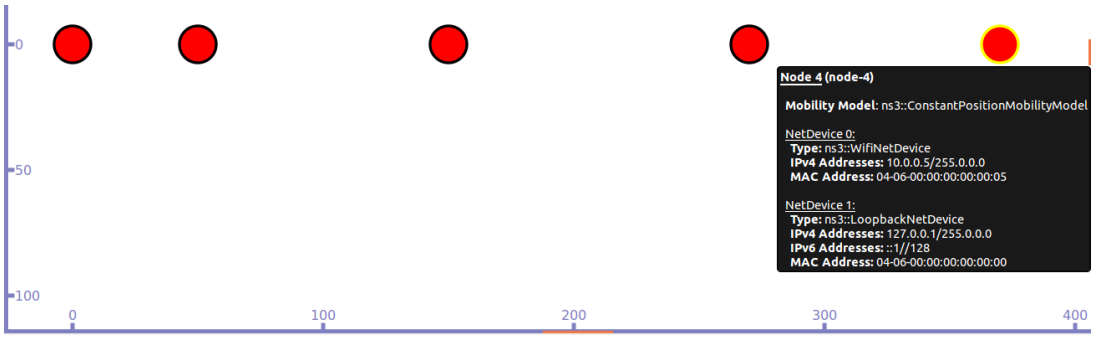

Figura 2.18. Escenario preliminar con cinco nodos estáticos

La Tabla 2.3 muestra los parámetros considerados a nivel de topología física para la distribucion de los nodos.

Tabla 2.3. Parámetros del escenario preliminar fijo

ESCENARIO FIJO	
Número de nodos	5
Long. Escenario [m]	370
Distancia max. Nodos[m]	50 – 150 – 270 – 370
Topología	Lineal
Tiempo de simulación [s]	20

En el ANEXO IV-A se encuentra adjunto el código del escenario incluyendo la traza generada por NS-3.

- **Column**

Este escenario define una MANET con un grupo de 5 nodos distribuidos de forma asimétrica; tras iniciar la simulación los nodos se distancian entre ellos; extendiéndose hasta distancias de 120 m. El modelo de movilidad que se usa para este escenario es

Column y se genera con Bonnmotion, mismo que se describe en la sección 2.1.4. Véase Figura 2.19.

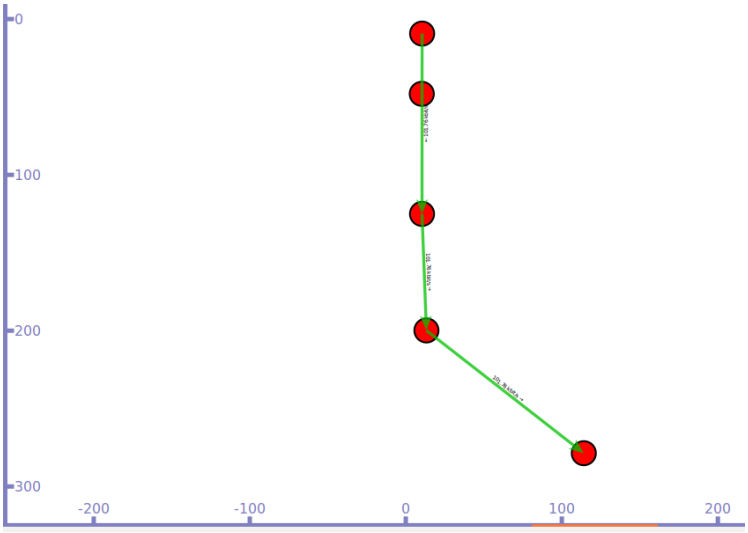

Figura 2.19. Escenario preliminar con el modelo Column de bonnmotion.

La Tabla 2.4 muestra los parámetros considerados para generar la traza de movilidad con el modelo Column de bonnmotion.

Tabla 2.4. Parámetros de traza - escenario preliminar Column.

COLUMN	
Número de nodos	5
Long. Escenario [m]	500
Distancia max. Nodos[m]	120
Topología	Lineal asimétrica
Tiempo de simulación [s]	600
Velocidad mínima [km/h]	1.8
Velocidad máxima [km/h]	5.4
Velocidad promedio [km/h]	3.6

En *el ANEXO IV*-B se encuentra el código usado para generar el escenario con la traza *Column*.

Escenarios ManhattanGrid

Este escenario aplica el modelo de movilidad ManhattanGrid a todos sus nodos participantes, el modelo se describe en la sección 2.1.4 en *Mobility Module*. Para las

pruebas realizadas se considera un área de 500 x 500 m y se manejan diferentes densidades de nodos: 25, 36 y 100 nodos; así mismo la velocidad de los dispositivos variará para cada prueba. A continuación, la Figura 2.20 muestra un ejemplo de un escenario ManhattanGrid con 36 nodos.

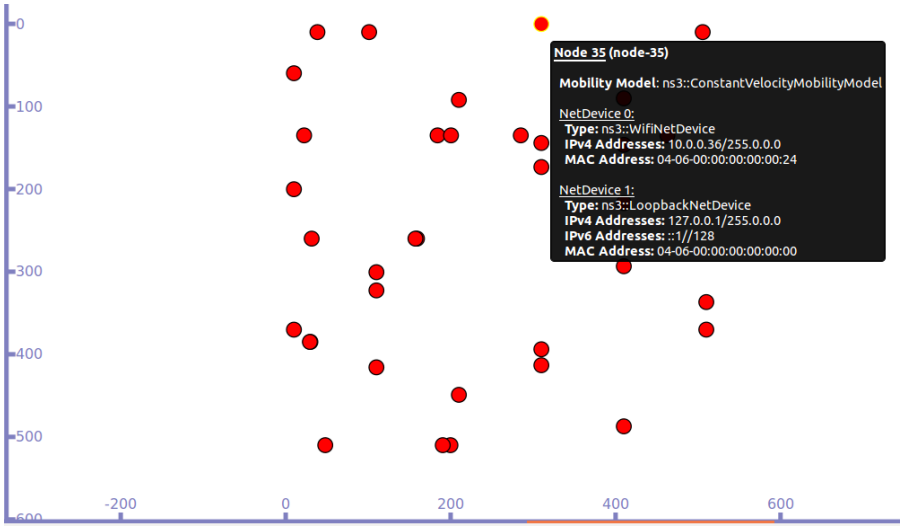

Figura 2.20. Escenario de prueba ManhattanGrid con 36 nodos

La Tabla 2.5 indica los parámetros considerados para generar la traza de movilidad con el modelo ManhattanGrid para 36 nodos con una velocidad promedio de 5km/h en bonnmotion.

Tabla 2.5. Parámetros de la traza ManhattanGrid con 36 nodos

MANHATTAN GRID 36 NODOS	
Número de nodos	36
Long. Escenario [m]	x=500, y=500
Distancia max. Nodos[m]	120
Modelo de movilidad	Manhattangrid
Tiempo de simulación [s]	600
Velocidad mínima [km/h]	1.8
Velocidad promedio [km/h]	5.04

Para generar la traza con 25 y 100 nodos se deberán variar únicamente el número de nodos y la velocidad promedio según sea el caso de simulación.

Escenarios RandomWaypoint

Este modelo de movilidad se describe en la sección 2.1.4 y al igual que con ManhattanGrid se realizará pruebas con este modelo en un área de 500 x 500 m considerando diferentes densidades de nodos: 25, 36 y 100 nodos. A continuación, la Tabla 2.6 muestra un ejemplo de los parámetros considerados para generar la traza de movilidad con 36 nodos con una velocidad de 5km/h. Para generar las demás trazas, los parámetros que deben variarse son: velocidad promedio y densidad de nodos.

Tabla 2.6. Parámetros de la traza RandonWayPoint con 36 nodos

RANDOMWAYPOINT 36 NODOS	
Número de nodos	36
Long. Escenario [m]	x=500, y=500
Distancia max. Nodos[m]	120
Modelo de movilidad	RandomWayPoint
Tiempo de simulación [s]	600
Velocidad mínima [km/h]	5.04
Velocidad máxima [km/h]	5.04

En el *ANEXO IV-C* se encuentra adjunto el código que se desarrolló para dar origen a los escenarios MANET, ya sea usando el modelo ManhattanGrid o RandomWaypoint.

2.4 FICHEROS DE RESULTADOS

Con el fin de analizar los resultados de las simulaciones se generó dos archivos en aodv-routing-protocol que incluyen la información más relevante del comportamiento de AODV y del funcionamiento del mecanismo de control de potencia; es decir permitirán analizar el rastreo de paquetes y la variación de la potencia de transmisión. Estos archivos se leerán con la ayuda de AWK.

2.4.1 Archivos

Desde el protocolo de enrutamiento AODV en NS-3 se generan dos archivos con las características que se detallan a continuación.

Archivo 1: Packet Trace

Este archivo contiene información que permite rastrear los paquetes transmitidos para así conocer la tasa de pérdidas, recepción, saltos, etc.; el archivo incluye parámetros como:

- Tiempo de simulación actual, se lo obtiene con la función *Now().As (Time::S)*.
- ID de paquete, para identificar el número de paquete meidante la función *GetUid*.
- Longitud del paquete, a través de la función *GetSize*.
- Nodo origen (IP source), usando la función *GetSource*.
- Nodo destino (IP destino), mediante la función *GetDestination*.
- Siguiente salto, muestra la dirección IPv4 del siguiente salto mediante la función *GetNextHop*
- Número de saltos, mediante la función *GetHop* se obtiene el número de saltos para llegar al destino.

Además, se ha incluido etiquetas de “T”, “R”, “F”, “D”, para indicar si un paquete fue transmitido, recibido, reenviado, o desechado respectivamente. Para conseguir esto es necesario analizar a detalle el código del protocolo de enrutamiento.
Todas las funciones usadas para elaborar el archivo *Packet Trace* han sido descritas en la sección 2.1.4 en *Network Module.*
El código de este archivo se lo puede ver en el *ANEXO V-A*.

Archivo 2: Power Level

El segundo archivo contiene información sobre la variación de la potencia de transmisión aplicando el mecanismo de control de potencia, los parámetros incluidos en este archivo son:

- Tiempo actual de simulación, usando la función *Now().As (Time::S)*.
- ID de nodo actual,
- Potencias, este parámetro se lo obtiene mediante *GetTxPowerStart*. Para poder diferenciar entre potencia inicial, actual y anterior se declaran variables globales y y unas líneas de código que permitan diferenciar las potencias.
- Vecinos, incluye la información tanto para vecinos actuales y vecinos anteriores de un nodo.

El *ANEXO V-B*, muestra el código de que se usó para generar el archivo *Power Level*.

Impresión de archivos

Para imprimir todos los parámetros mencionados en los archivos correspondientes es necesario el uso de la clase *ofstream* y algunas funciones:

- *Ofstream,* permite operar el flujo de datos de salida en archivos.
- *Open,* da apertura a un archivo.
- *Out,* admite operaciones de salida de datos;
- *App,* permite que todas las operaciones de salida ocurran al final del archivo, si el archivo ya existe incluye lo nuevo a su contenido existente.

Para generar el archivo, es necesario configurar las siguientes líneas de código:

```
1   std::ofstream ofs;
2   ofs.open ("LOG.txt", std::ofstream::out | std::ofstream::app);
3   ofs<< variable a imprimir << "\t" ;
4   …
5   ofs.close();
```

La línea 1 el llamado a la clase *ofstream*, se asigna a la variable *ofs*, en la segunda línea se abre el archivo a generar y se define el nombre del mismo; la tercera línea indica la impresión de las variables requeridas en el archivo. Finalmente, en la última línea se cierra el archivo.

2.4.2 Lectura de archivos con AWK

Los dos archivos generados con la información de las simulaciones incluyen datos muy extensos puesto que los parámetros a evaluar son varios; por lo que para analizar parte de la información es necesario el uso de una herramienta adicional; para esto se acude a AWK.

AWK

Es un lenguaje de programación muy potente que permite la fácil manipulación de datos estructurados [89] y es diseñado especialmente para el procesamiento de texto y la generación de reportes en base a ello, es decir selecciona registros particulares de un archivo y realiza operaciones sobre ellos. Su nombre hace referencia a los apellidos de sus autores: Alfred Aho, Peter Weinberger y Brian Kernighan. Algunas de las características de Awk son [90]:

- Permite administrar pequeñas bases de datos.
- Genera informes.
- Útil para validar datos y ordenarlos.

- Genera índices y otras tareas de preparación de documentos
- Permite experimentar con algoritmos para posterior ser adaptados a otros lenguajes.
- Extrae bits y fragmentos de datos para su procesamiento.
- Facilita extender el lenguaje con funciones escritas en C o C ++

Existen tres variantes de AWK, estas son:

- *AWK*, versión original de AWK escrita en 1977 por Laboratorios AT&T Bell.
- *NAWK*, es la nueva versión mejorada de AWK del Laboratorio AT &T.
- *GAWK*, es la implementación de AWK por GNU totalmente compatible con AWK y NAWK. Esta es la versión que se usa para el análisis de los archivos de este proyecto.

- **Instalación de GAWK**

Awk, por lo general está disponible por defecto en la mayoría de las distribuciones GNU/Linux, en caso de que este programa no se encuentre disponible el proceso de instalación es bastante sencillo y se muestra a continuación:

1. Descargar el código fuente de la plataforma web, para esto se usa:
   ```
   wget http://ftp.gnu.org/gnu/gawk/gawk-4.1.1.tar.xz
   ```
2. Descomprimir y extraer el código fuente descargado
   ```
   tar xvf gawk-4.1.1.tar.xz
   ```
3. Correr la configuración
   ```
   ./configure
   ```
4. Compilar el código fuente mediante *make.*
   ```
   make
   ```
5. Ejecutar un grupo de pruebas para verificar que la compilación tuvo éxito, opcional.
   ```
   make check
   ```
6. Instalar awk
   ```
   sudo make install
   ```

Para la ejecución de un script gawk se debe tener este archivo más el archivo que contiene la información a ser filtrada con gawk en una misma carpeta y correr la siguiente línea:

```
gawk –f filtro.awk archivo_info.log
```

- **Filtros con GAWK**

Con el fin de facilitar el análisis de los resultados obtenidos de las simulaciones se ha diseñado dos filtros en GAWK para analizar los paquetes y la potencia de transmisión respectivamente.

Filtro_pack:

Este filtro analiza la información del archivo *Packet Trace* (generado en la sección anterior) y devuelve la cantidad de paquetes totales transmitidos, recibidos, desechados y reenviados; muestra además el throughput y las tasas de pérdidas y de reenvío obtenidas. En base a la información filtrada que entrega este filtro es posible tabular los datos para el análisis de las transmisiones.

En el *ANEXO VI-A* se encuentra el código empleado para generar este filtro.

Filtro_power:

Este filtro permite determinar la potencia de transmisión promedio en [dBm] y [mw] de un nodo en específico en base al análisis del archivo generado en la sección anterior *Power Level.* De esta forma, se puede evidenciar la variación de la potencia de transmisión gracias al uso del mecanismo de control de potencia.

El código empleado para generar este filtro se encuentra adjunto en el *ANEXO VI-B.*

3. RESULTADOS Y DISCUSIÓN

En esta sección se presenta los resultados del proyecto de titulación. Para esto se realizan pruebas en diferentes escenarios ad-hoc y MANET con la finalidad de evidenciar la variación en el rendimiento de AODV implementando un control de potencia. Se incluye también, un análisis comparativo para identificar la diferencia en la operación de una MANET que incluye control de potencia y no.

3.1 PRUEBAS

Como ya se indicó en la sección 2.3.2, se realizarán dos tipos de pruebas: preliminares y pruebas finales. Éstas últimas consideran un escenario más realista y complejo por lo que variarán parámetros, tales como: número de nodos, velocidades, modelo de movilidad. Todas las pruebas de tipo final consideran un nodo fijo que será el servidor y los nodos restantes actuarán como clientes UDP que enviarán el tráfico y actuarán bajo el modelo de movilidad establecido para cada escenario. La posición del nodo servidor para todos los casos está en x=250m, y=0.001m para considerar una ubicación céntrica. Todos los nodos clientes enviarán 100 paquetes de datos con un tamaño de 1024 bytes en un tiempo total de simulación de 600s.

A continuación, se muestran los resultados promedio obtenidos de las simulaciones. Para las pruebas finales los valores promedio se obtienen de un total de 10 simulaciones para cada caso.

3.1.1 PRUEBAS PRELIMINARES

Estas pruebas se llevan a cabo sobre los escenarios preliminares que se definieron en la sección 2.3.2 y permiten validar que el mecanismo de control de potencia que se incluyó en el protocolo de enrutamiento AODV opera de una manera adecuada. A continuación, se muestran los resultados obtenidos de las simulaciones.

Escenario fijo

Para la prueba en esta red ad-hoc se considera como servidor al nodo 0, y el nodo 4 es el cliente, véase Figura 3.1. El tiempo de simulación es de 20s.

La Tabla 3.1 muestra los resultados obtenidos de las simulaciones con control de potencia y sin él.

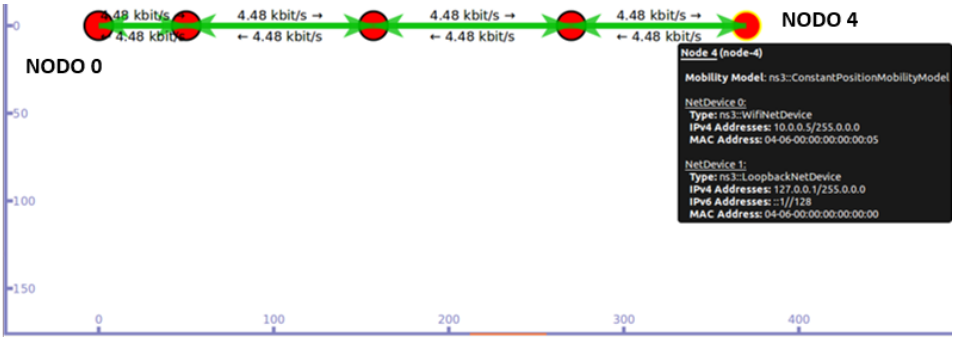

Figura 3.1. Prueba de transmisión en escenario fijo

Los parámetros incluidos en las tablas de resultados consideran las siguientes abreviaturas:

N_i *:* Identificador de un nodo.
RX: Número de paquetes recibidos.
DR: Número de paquetes desechados.
FW: Número de paquetes reenviados.
TH: Throughput.

Tabla 3.1. Resultados de las pruebas en una red ad-hoc

PRUEBA N°	DISTANCIA [m]					SIN MECANISMO				CON MECANISMO			
	N_0	N_1	N_2	N_3	N_4	RX	DR	FW	TH[%]	RX	DR	FW	TH[%]
1	0	50	150	270	370	0	0	3	0,0	90	0	271	90,0
2	0	50	150	270	370	0	1	4	0,0	147	0	464	73,5
3	0	50	145	270	370	0	0	0	0,0	6	0	18	6,0
4	0	50	155	270	370	99	0	298	99,0	100	0	300	100,0

Como se puede observar en la Tabla 3.1, el throughput es notablemente mayor al usar el mecanismo de control de potencia en las tres primeras pruebas. Con lo cual se puede validar el funcionamiento del mecanismo de control de potencia diseñado.

En la última prueba se puede ver que el throughput obtenido sin usar el mecanismo de control de potencia casi alcanza un valor que definiría una transmisión ideal. Esto sucede puesto que la distancia entre nodo N2 y el nodo N3 se redujo a 115m, distancia a la cual es posible una transmisión con mínimas pérdidas. Debe recordarse que a los 114m se pierde el primer paquete.

Escenario Columna

Esta es la primera prueba sobre una MANET, *Column* es un escenario muy sencillo como ya se indicó; de forma similar al caso anterior; el nodo 0 actuará como servidor y el nodo 4 es el cliente UDP, véase la Figura 3.2.

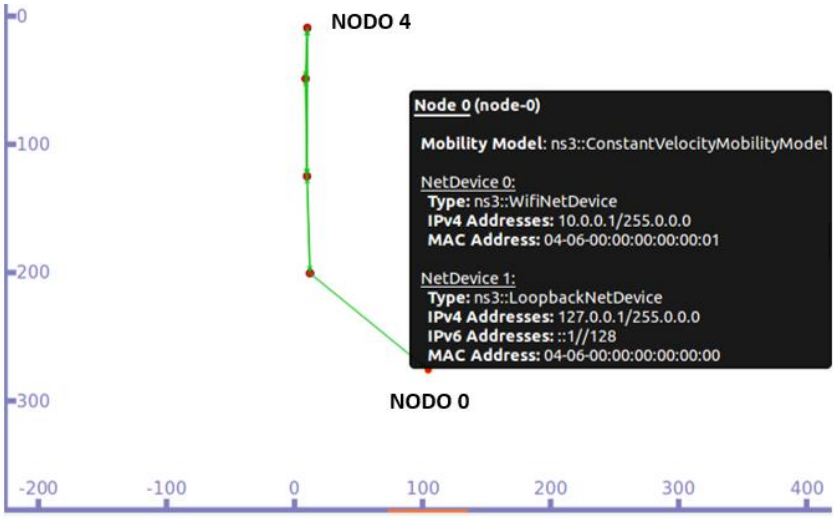

Figura 3.2. Prueba de transmisión en una MANET con el modelo Column

Un resumen de los resultados obtenidos de las simulaciones con control de potencia y sin él se puede observar en la Tabla 3.2. En la cual se verifica que el uso del mecanismo de control de potencia es favorable para la transmisión en la MANET. En las tres pruebas realizadas con diferentes trazas incluyendo el control de potencia, el throughput obtenido es muy elevado lo cual nos indica que las transmisiones son exitosas.

Con el uso del mecanismo el número de saltos incrementa puesto que el nodo actual siempre está buscando obtener un camino para llegar al destino. Los paquetes desechados conservan un nivel bajo.

Tabla 3.2. Resultados de las pruebas en una MANET con Column

PRUEBA N°	PAQ. TX	TRAZA	SIN MECANISMO				CON MECANISMO			
			RX	DR	FW	TH [%]	RX	DR	FW	TH [%]
1	100	c2	0	0	2	0,0	95	2	198	95,0
2	100	c2a	5	0	12	5,0	99	0	200	99,0
3	100	c2b	4	9	103	4,0	100	0	200	100,0

Con estos resultados, se puede determinar que el mecanismo de control de potencia que se implementó en el protocolo de enrutamiento AODV funciona de una manera adecuada y porque no decir beneficiosa para la red. A continuación, se muestran las pruebas finales que toman lugar en varios escenarios MANET más complejos.

3.1.2 PRUEBA 1: THROUGHPUT vs DENSIDAD – MANHATTANGRID

Esta prueba contempla simulaciones de una MANET que trabaja con el escenario ManhattanGrid. El objetivo de esta prueba es analizar el nivel de throughput que se

puede obtener en diferentes densidades de nodos trabajando con una velocidad promedio constante. Las simulaciones se realizarán con control de potencia y sin él en densidades de: 25, 36, y 100 nodos. La velocidad promedio en los nodos será de 5km/h para todos los casos.

Tabla 3.3 Resultados promedio Prueba 1

P. N°	DENSIDAD	SIN MECANISMO				CON MECANISMO			
		RX	DR	FW	TH [%]	RX	DR	FW	TH [%]
1a	25	420,5	29,30	541,20	17,52	595,00	30,90	919,60	24,79
1b	36	594	161,3	2802,2	16,97	710,1	170,9	2332,1	20,29
1c	100	1.339,1	583,7	13.260,9	13,53	1.212,2	828,3	12.844,60	12,24

La Tabla 3.3 muestra los resultados promedio de las simulaciones realizadas para cada caso con 10 muestras; incluye datos sobre el rastreo de paquetes.

- La prueba 1a considera 25 nodos, el nodo servidor con posición fija será el 24 y los nodos restantes serán los clientes UDP. El mecanismo de control de potencia usará el valor de potencia mínima determinado para la densidad de 25 nodos; $P_{min} = 9.99\ dBm$.
- La prueba 1b tiene 36 nodos en el escenario MANET, el nodo servidor UDP es el nodo 35. El mecanismo de control de potencia usará como potencia mínima $8.4\ dBm$.
- La última prueba del tipo 1 considera un escenario MANET con 100 nodos, donde el nodo 99 será el servidor UDP. El mecanismo de control de potencia usará. $P_{min} = 3.97\ dBm$.

Para analizar la variación de la potencia de transmisión en esta prueba, se considera la Tabla 3.4 en la cual se muestra el valor de la potencia promedio con y sin uso del mecanismo de control de potencia para los tres casos: prueba 1 a (25 nodos), prueba 1 b (36 nodos) y prueba 1c (100 nodos). Los datos considerados para la Tabla 3.4 son tomados de un nodo en particular seleccionado al azar de aquellas simulaciones en las cuales el throughput incrementa al usar el mecanismo de control de potencia.

Tabla 3.4 Variación de la potencia promedio de transmisión – Prueba 1

PRUEBA	TRAZA	NODO	SIN MECANISMO	CON MECANISMO
			Pot. Media [dBm]	Pot. Media [dBm]
Prueba 1a	m25b	15	16.026	13.547
Prueba 1b	m36f	23	16.026	13.233
Prueba 1c	m100e	88	16.026	13.033

El análisis comparativo de los datos de la Tabla 3.3 y Tabla 3.4 se realiza en la siguiente sección.

3.1.3 PRUEBA 2: THROUGHPUT vs DENSIDAD - RANDOMWAYPOINT

Esta prueba, similar a la prueba 1, intenta analizar el nivel de throughput que se puede obtener trabajando con una velocidad promedio constante en los nodos y con diferentes densidades, pero considerando el escenario RandomWaypoint en las MANETs. Se realizarán 10 simulaciones con y sin control de potencia para establecer un resultado promedio. Las densidades son: 25, 36, y 100 nodos. La velocidad promedio en los nodos es de 5km/h para todos los casos.

Al igual que en la prueba 1 el mecanismo de control de potencia variará su parámetro de potencia mínima conforme sea la densidad de nodos. La Tabla 3.5 es un resumen de los resultados obtenidos para los tres casos de la prueba 2.

Tabla 3.5. Resultados promedio Prueba 2

P. N°	DENSIDAD	SIN MECANISMO				CON MECANISMO			
		RX	DR	FW	TH [%]	RX	DR	FW	TH [%]
2a	25	401,3	7,7	754,8	16,72%	477,7	57,4	986,5	19,90%
2b	36	933,4	305,7	5368,4	26,67%	1065,9	310,5	4419,9	30,45%
2c	100	1.535,43	420,00	12324,57	15,51%	1.309,57	649,00	11516,57	13,23%

La Tabla 3.6 muestra la variación de la potencia de transmisión promedio para todos los casos de la prueba 2: prueba 2a (25 nodos), prueba 2b (36 nodos) y prueba 2c (100 nodos) con y sin control de potencia. Los datos se han tomado de simulaciones en las cuales al usar el mecanismo de control de potencia se ha obtenido una mejora en el throughput, y se considera diferentes nodos.

Tabla 3.6. Variación de la potencia promedio de transmisión – Prueba 2

PRUEBA	TRAZA	NODO	SIN MECANISMO	CON MECANISMO
			Pot. Media [dBm]	Pot. Media [dBm]
Prueba 2a	r25i	20	16.026	13.952
Prueba 2b	r36i	5	16.026	13.340
		20	16.026	17.782
Prueba 2c	r100b	70	16.026	13.276

En el caso de la prueba 2b, puede observarse que se han considerado dos nodos para

analizar el nivel de potencia. Esto con el fin de mostrar que el nivel de potencia puede reducir o incrementar conforme varíe la condición física del nodo transmisor.

El análisis de los datos tabulados se realizará más adelante.

3.1.4 PRUEBA 3: THROUGHPUT vs VELOCIDAD - MANHATTANGRID

Esta prueba considera una variación de velocidades frente a una densidad fija de nodos en un escenario MANET con ManhattanGrid. Las simulaciones se realizan con y sin control de potencia con velocidades de: 5, 6, 10 y 20 km/h que muestran el valor promedio de la velocidad a la que camina, trota y corre una persona respectivamente. Se considera un escenario con 36 nodos. El tiempo de simulación y el tamaño de paquetes se mantienen. El mecanismo de control de potencia para los casos a,b,c y d no cambia ya que se considera una misma densidad de nodos; en este caso al ser 36 nodos el mecanismo usa los parámetros definidos para esta densidad.

La Tabla 3.7 muestra un resumen de los resultados promedio obtenidos de 10 simulaciones para cada caso.

Tabla 3.7 Resultados promedio prueba 3

P. N°	V.PROM [km/h]	SIN MECANISMO				CON MECANISMO			
		RX	DR	FW	TH [%]	RX	DR	FW	TH [%]
3a	5	594	161,3	2802,2	16,97	710,1	170,9	2332,1	20,29
3b	6	603,5	170,2	2002,8	17,24	666,9	162	2369,2	19,05
3c	10	503,9	118	1685,6	14,40	527,6	51,9	1228,7	15,07
3d	20	562,5	32,4	1141	16,07	549	143,1	1876	15,69

La variación de la potencia de transmisión promedio para todos los casos de la prueba 3: prueba 3a (5 km/h), prueba 3b (6 km/h), prueba 3c (10 km/h), y prueba 3d (20 km/h) con y sin control de potencia se la puede apreciar en la Tabla 3.8. Los datos son tomados de las simulaciones en las cuales al usar el mecanismo de control de potencia el throughput mejora.

Tabla 3.8. Variación de la potencia promedio de transmisión – Prueba 3

PRUEBA	TRAZA	NODO	SIN MECANISMO	CON MECANISMO
			Pot. Media [dBm]	Pot. Media [dBm]
Prueba 3a	m36f2	22	16.026	13.145
Prueba 3b	m6i	1	16.026	13.607

Prueba 3c	m10c	7	16.026	13.706
		21	16.026	16.414
Prueba 3d	m20h	0	16.026	16.122
		33	16.026	13.276

En la siguiente sección se realiza el análisis comparativo de las tabulaciones presentadas.

3.1.5 PRUEBA 4: THROUGHPUT vs VELOCIDAD - RANDOMWAYPOINT

Esta última prueba al igual que la prueba 3 considera una variación de velocidades frente a una densidad fija de 36 nodos. Se simula en un escenario MANET RandomWaypoint con y sin control de potencia a velocidades de: 5, 6, 10 y 20 km/h; por lo cual se tiene cuatro variantes de esta prueba. El tiempo de simulación y el tamaño de paquetes son los mismos que para las pruebas previas.

La posición del nodo servidor está en x=250m, y=0.001m; los 35 nodos clientes obtendrán su posición según la traza de movilidad.

Se usa un único mecanismo de control de potencia ya que la densidad de nodos es constante para todos los casos. Al ser 36 nodos, el mecanismo usa los parámetros definidos para esta densidad.

A continuación, en la Tabla 3.9 se muestra los resultados promedio obtenidos de 10 simulaciones para cada caso.

Tabla 3.9 Resultados promedio Prueba 4

P. N°	V. PROM [km/h	SIN MECANISMO				CON MECANISMO			
		RX	DR	FW	TH [%]	RX	DR	FW	TH [%]
4a	5	933,4	305,7	5368,4	26,67	1065,9	310,5	4419,9	30,45
4b	6	1141,8	183,9	5135,3	32,62	1064,1	294,2	4825,9	30,40
4c	10	562,3	166,7	5454,6	16,07	587,3	245,3	3619	16,78
4d	20	714,8	469	6413,7	20,42	749,4	439,2	5461,2	21,41

La evaluación de la variación de la potencia de transmisión se la hace en base a la potencia media. Los datos se toman de las simulaciones que registraron el mayor throughput al usar el mecanismo de control de potencia. Véase Tabla 3.10.

Tabla 3.10. Variación de la potencia promedio de transmisión – Prueba 4

PRUEBA	TRAZA	NODO	SIN MECANISMO	CON MECANISMO
			Pot. Media [dBm]	Pot. Media [dBm]
Prueba 4a	r5i	17	16.026	13.102
Prueba 4b	r6b	31	16.026	13.627
Prueba 4c	r10g	21	16.026	13.167
Prueba 4d	r20j	16	16.026	13.145

En la siguiente sección se realiza un análisis comparativo de todos los datos tabulados para las diferentes pruebas.

3.2 DISCUSIÓN, ANÁLISIS COMPARATIVO

En esta sección se realiza un análisis comparativo del comportamiento de una MANET con y sin control de potencia enfocado a los diferentes escenarios MANET que se han manejado en este proyecto. Este análisis incluye el estudio de tres parámetros importantes como son: throughput, tasa de reenvío y tasa de paquetes perdidos.

El throughput permite analizar el rendimiento de la transmisión, es decir permite conocer la tasa efectiva de paquetes que se recibieron; se calcula en base a la razón de los paquetes recibidos sobre el total de paquetes transmitidos $\eta = Paq.Rx/Paq.Tx$.

La tasa de reenvío es importante analizarla ya que si se tiene un mayor nivel de reenvío significa que los paquetes atravesaron más saltos para poder llegar al destino lo cual genera mayor carga de retransmisión y esto puede llevar a que varios nodos compitan por un canal generando así problemas de contención y retrasos de cola. En este caso, la tasa de reenvío se analiza en base a la relación entre paquetes reenviados sobre los paquetes recibidos $r_f = Paq.Fw/Paq.Rx$ para de este modo evaluar el esfuerzo que se realizó entre los nodos para recibir cierta cantidad de paquetes. Donde r_f: $Tasa\ de\ reenvío$.

Finalmente, la tasa de paquetes perdidos se la analiza en base a la información de paquetes desechados (*drops)* lo cual es importante analizar puesto que estos paquetes pudieron haber sido transmitidos, pero por varias razones como: movilidad, obstáculos en el canal inalámbrico, contención no pudieron alcanzar su destino. De forma similar se analizará la tasa de pérdidas en base a la relación entre paquetes perdidos y paquetes recibidos para evaluar el nivel de pérdida que se tuvo que admitir para recibir cierto número de paquetes: $r_l = Paq.Drop/Paq.Rx$ .Donde r_l: $Tasa\ de\ pérdidas$.

En base a estos tres parámetros se determina que tan bueno puede ser incluir el mecanismo diseñado para controlar potencia.

3.2.1 PRUEBA 1: THROUGHPUT vs DENSIDAD - MANHATTANGRID

El análisis comparativo se hará en base a los parámetros antes mencionados, a continuación, se los detalla.

- **Throughput**

Los resultados promedios del throughput para esta prueba son variables, véase Figura 3.3. En base a esta figura se puede determinar que:

- Si se aplica el mecanismo de control de potencia en una MANET con una densidad ligera de nodos, el throughput aumentará, mientras que, si se tiene una MANET con un número de nodos bastante grande, como este caso 100 nodos (en 0.25 km^2), el throughput se reducirá. La principal razón para que esto suceda es que, mientras mayor es el número de nodos, menor será la mínima potencia de transmisión que considera el mecanismo para cada nodo. Debe recordarse que el mecanismo está diseñado para escenarios que manejan una distribución simétrica y por tal razón, mientras mayor sea el número de nodos, el área a cubrir para cada nodo será menor y por ende la potencia mínima necesaria para iniciar una transmisión será más pequeña.

  En este este caso, con los resultados de la prueba 1c (100 nodos), se pude afirmar que la cota mínima que se calculó en este trabajo es muy baja (3.97 dBm) y por tal motivo no se logra una tasa de throughput como se esperaría. Sin embargo, es claro que la diferencia no es muy amplia con lo cual se puede decir que pese a que la potencia mínima de transmisión es muy baja se logró un número importante de transmisiones reduciendo el consumo de energía tal como se muestra en la Figura 3.4.
- Se puede verificar que ante una densidad no muy alta de nodos, como es el caso 25 y 36 nodos (en 0.25 km^2) el mecanismo de control de potencia opera de manera favorable para la red arrojando un mejor nivel de throughput. Esto nos indica que la cota mínima de potencia de transmisión calculada para 25 (prueba 1a) y 36 (prueba 1b) nodos es adecuada, pues permite incrementar el número de transmisiones exitosas al mismo tiempo que redujo la potencia de transmisión requerida.

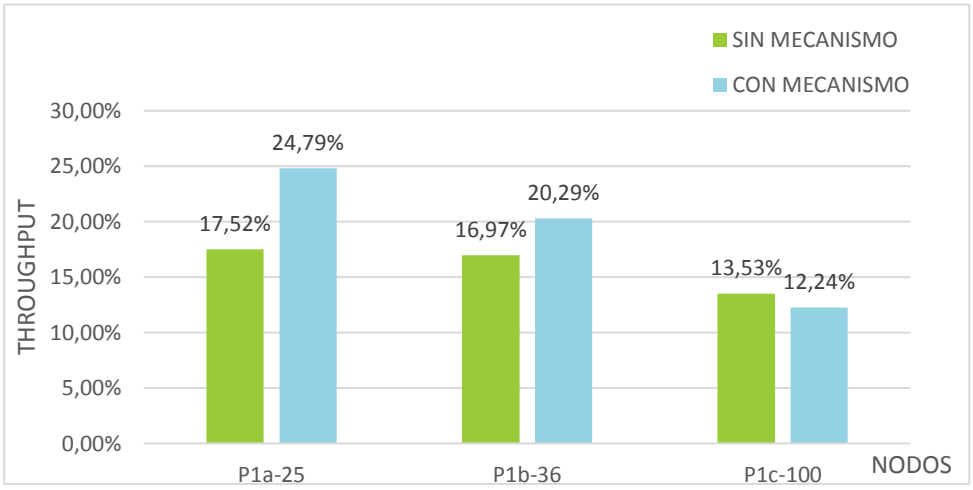

Figura 3.3. Resultados prueba 1- Throughput vs Densidad de nodos

- Otro factor que puede incidir para que en una densidad de 100 nodos al usar el mecanismo de control de potencia el throughput disminuya es el área de interferencia existente; pues este factor va a ser más notorio mientras que la cantidad de nodos incremente. Al existir una mayor cantidad nodos en el área designada, la probabilidad de que estén más cercanos uno de otros va a aumentar. Por tanto, al tener mayor interferencia entre los nodos, existe una mayor posibilidad de colisiones y pérdida de paquetes, lo cual afecta directamente al número de transmisiones exitosas.

- **Tasa de reenvío**

El nivel de reenvío que se puede observar en la Figura 3.4, indica que al menos en uno de los tres casos de la prueba 1 el mecanismo de control de potencia resultó ser útil (36 nodos) ya que en este caso el nivel reenvío se reduce en un 144% a la vez que la eficiencia en la transmisión mejoró. Por lo que podemos decir que el mecanismo de control de potencia es útil y que requiere un ajuste de límites y de pasos (*delta_power)* para operar mejor con las otras dos densidades.

Para el caso de prueba con 100 nodos el nivel de reenvío usando el mecanismo de control de potencia se ve muy elevado ya que como se explicó previamente, la potencia mínima para esta densidad es muy baja; por tanto, el mecanismo va a tratar de insistir las veces que sean necesarias para lograr transmitir, reenviando así un elevado número de

paquetes. Sin embargo, la transmisión no se puede dar por exitosa ya que la potencia con la que se opera es muy baja y el nivel de interferencia puede ser más consistente.

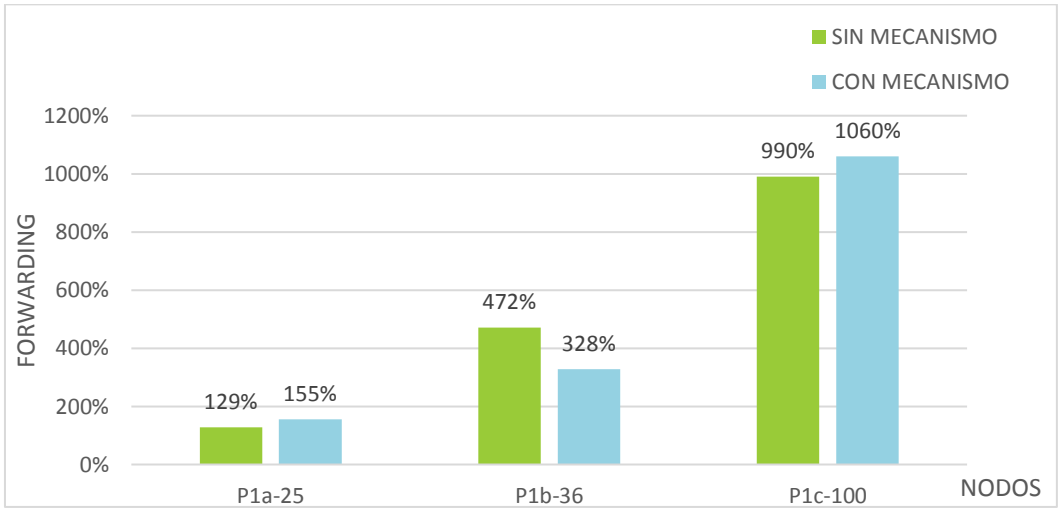

Figura 3.4. Resultados prueba 1- Reenvío vs Densidad de nodos

- **Tasa de pérdidas**

Al igual que con el reenvío; para el caso de la prueba 1c (100 nodos) se puede ver que el nivel de pérdidas usando el mecanismo de control de potencia incrementa en su mayoría, véase Figura 3.5. Esto es consecuencia indirecta del bajo nivel de potencia, pues, al tener mayor número de reenvíos estamos generando mayor carga en la red y esto a su vez provoca colisiones y fuertes contenciones por el canal llevando así a que la probabilidad de pérdida de paquetes se eleve.

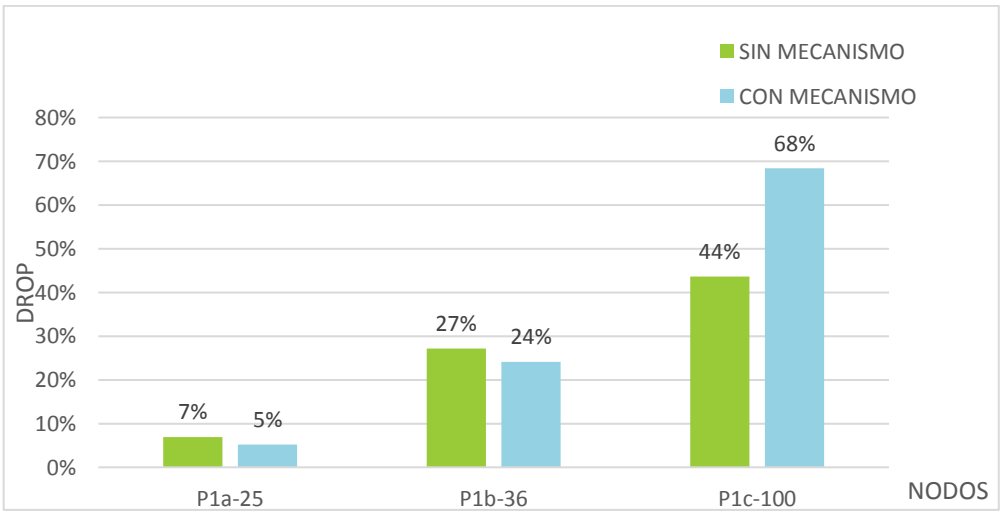

Figura 3.5. Resultados prueba 1- Pérdidas vs Densidad de nodos

Para los casos de prueba con 25 y 36 nodos (prueba 1a y prueba 1b respectivamente), se puede evidenciar que al usar el mecanismo de control de potencia la tasa de pérdida de paquetes tuvo una ligera baja; y es justamente en estos dos casos en los cuales el throughput es mayor usando el mecanismo de control de potencia. Esto se debe a que varios paquetes que antes no pudieron alcanzar su destino por su limitación en la potencia de transmisión ahora al usar el mecanismo lograron una transmisión exitosa gracias a que la potencia con la que fueron transmitidos debió ser mayor. Debe considerarse que la potencia no siempre va a ser incrementada, pues existen muchos casos de simulaciones en los cuales se ha logrado transmisiones exitosas con un nivel de potencia menor. Véase la Figura 3.6.

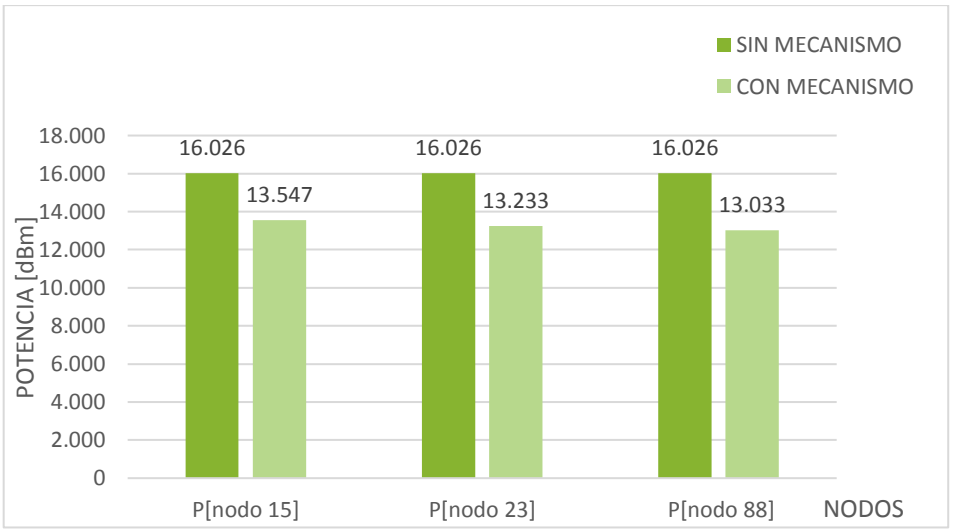

Figura 3.6. Potencia media de transmisión – Pruebas 1

En la Figura 3.6 se muestra la variación en la potencia de transmisión promedio luego de usar el mecanismo de control de potencia. Como se puede observar en los tres casos de la prueba 1 la potencia promedio de transmisión es menor con al menos 3 dBm con lo cual se está reduciendo el consumo de energía y por ende de baterías (punto crítico en los dispositivos de una red Ad-hoc).

Por los resultados obtenidos para esta prueba se puede inferir que si se requiere incluir un control de potencia en escenarios que incluyan densidades de nodos muy altas no debe considerarse una potencia de transmisión muy baja (proporcional a la densidad), puesto que esto limitará el número de transmisiones exitosas, tal como en el caso de la prueba 1c (100 nodos).

El mecanismo de control de potencia en general está bien, puesto que su diseño se basa en una distribución simétrica y la potencia mínima está relacionada directamente con la

densidad de nodos del escenario. Pero sin duda, para este caso, si se desea incrementar el número de transmisiones exitosas, se debe modificar la cota mínima de potencia considerada.

3.2.2 PRUEBA 2: THROUGHPUT vs DENSIDAD - RANDOMWAYPOINT

Esta prueba analiza el throughput al igual que la prueba 1, la diferencia es que se maneja en el escenario que usa RandomWayPoint.

- **Throughput**

En la Figura 3.7 se puede determinar que el mecanismo de control de potencia va a tener resultado favorables siempre y cuando se aplique a una densidad no muy alta de nodos. Es claro que al llegar a los 100 nodos (prueba 2c) no existe una mejora en el throughput *promedio*. Sin embargo, hubo casos en los que al usar el mecanismo de control de potencia para esta prueba el throughput mejoró.

De manera similar a lo que ocurrió para la prueba 1 con ManhattanGrid, va a suceder en esta prueba 2 con RandomWaypoint ya que la mínima potencia de transmisión calculada para 100 nodos es la misma y es muy baja, lo cual limita las transmisiones exitosas, incrementa el número de saltos (reenvíos) y por ende la probabilidad de contención, colisiones y retardos va a incrementar.

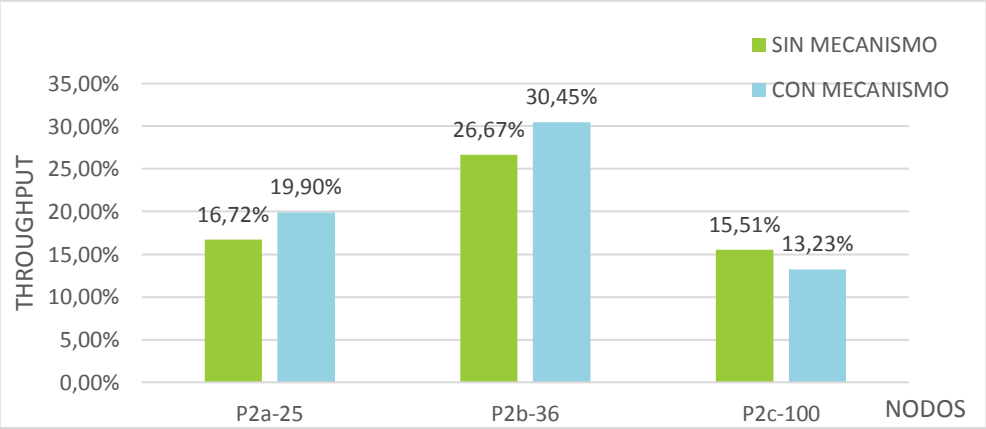

Figura 3.7. Resultados prueba 2- Throughput vs Densidad de nodos

A continuación, la Figura 3.8 y 3.9 muestran un resumen de la tasa de reenvío y la tasa de pérdidas respectivamente.

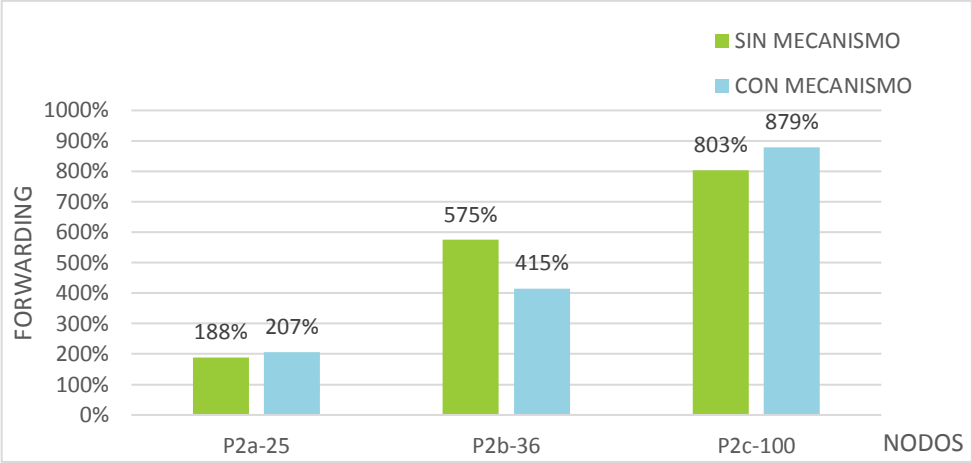

Figura 3.8. Resultados prueba 2- Reenvíos vs Densidad de nodos

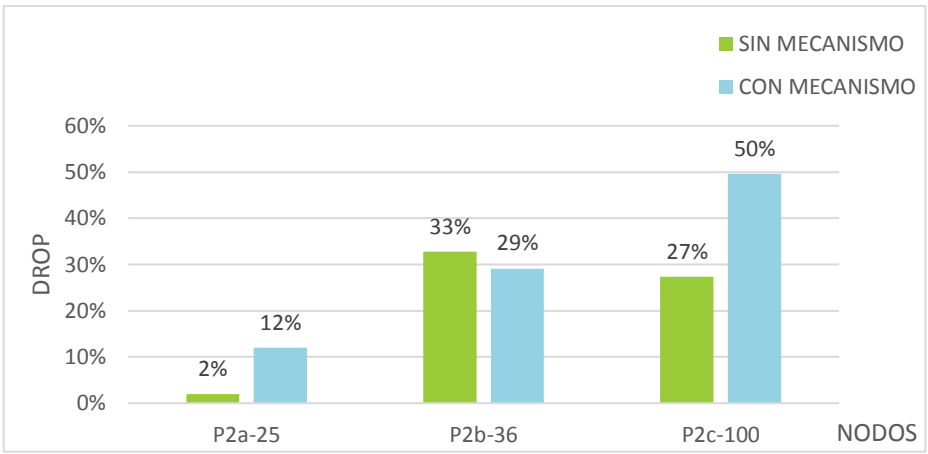

Figura 3.9. Resultados prueba 2- Pérdidas vs Densidad de nodos

Para analizar la potencia de transmisión promedio se ha considerado solo aquellos casos en los cuales el throughput es mayor si se usa el mecanismo de control de potencia. Véase Figura 3.10. Esto, ya que es interesante conocer cuál fue la potencia necesaria para tener transmisiones exitosas usando el mecanismo de control de potencia. Como se muestra en la Figura 3.10, se han conseguido transmisiones exitosas aun cuando el nivel de potencia es menor lo cual es favorable en cuanto a consumo de energía.

Para el caso de la prueba 2b, analizando el nodo N° 20 se puede observar que la potencia de transmisión promedio para este nodo usando el mecanismo es mayor que la potencia predefinida por el simulador. Este valor es justificable, puesto que gracias a un incremento mínimo de potencia se consiguió mejorar el throughput.

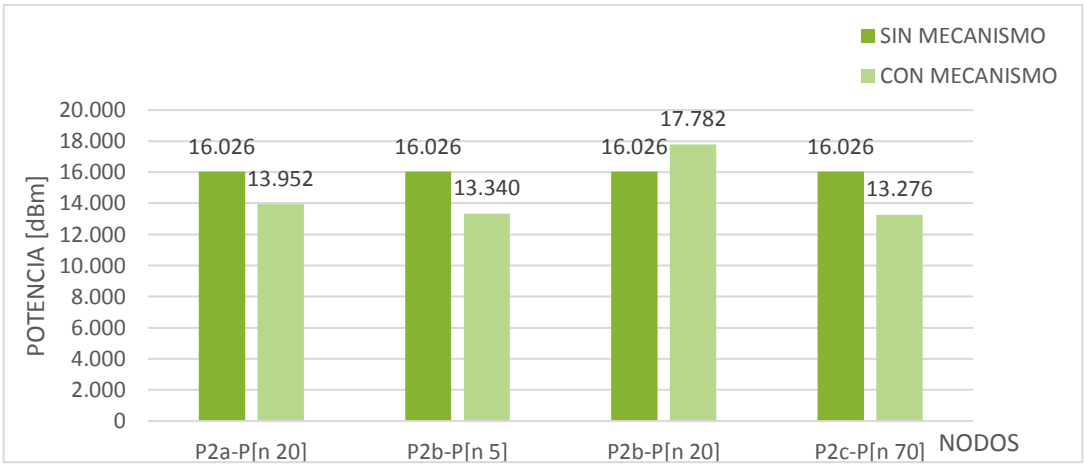

Figura 3.10. Potencia media de transmisión – Pruebas 2

3.2.3 PRUEBA 3: THROUGHPUT vs VELOCIDAD - MANHATTANGRID

Los resultados de la Tabla 3.7 permiten observar que en la mayoría de los casos para esta prueba es favorable el uso del mecanismo de control de potencia puesto que permite un mayor nivel de throughput y menor tasa de reenvío. A continuación, se describe el análisis de estos parámetros.

- **Throughput**

En la Figura 3.11 se puede verificar que el throughput promedio obtenido para las pruebas que incluyen control de potencia es mayor que para los casos que no lo consideran; excepto el último caso (prueba 3d – 20km/h) en el cual el throughput es ligeramente menor.

Este comportamiento nos permite decir que un nodo en movimiento con velocidades bajas de: 5 ,6 y 10 km/h (que son velocidades promedio a las cuales las personas suelen caminar o trotar), puede adoptar un menor nivel de potencia sin que esto influya de forma negativa en la recepción de paquetes. Al contrario, como se muestra en la Figura 3.11, al incluir un control de potencia en nodos móviles a las velocidades mencionadas se ve una mayor eficiencia a la vez que se reduce el consumo de energía. (Véase Figura

Por la prueba 3d (20km/h), se puede decir que, a medida que la velocidad con la que se mueve un nodo aumenta, la probabilidad de que se pueda reducir potencia, sin que esto afecte a la recepción de paquetes disminuye. Lo cual nos indica que mientras más rápido se mueva un nodo, es mejor que la potencia sea más alta para evitar una pérdida de paquetes.

Pese a que el valor de throughput *promedio* para la prueba 3d no sea favorable al usar el mecanismo de control de potencia, no quiere decir que no se dieron casos en los cuáles al usar el mecanismo se obtuvo un mejor throughput. Cinco de las 10 pruebas realizadas para promediar el throughput de la prueba 3d resultaron tener un mejor rendimiento usando el mecanismo de control de potencia. Sin embargo, hubo dos pruebas que hicieron que el throughput promedio sea mejor para el caso en el que no se usa el mecanismo de control de potencia. Esto quiere decir que el mecanismo diseñado puede ser poco útil a velocidades de 20km/h por lo que debería realizarse un ajuste en los parámetros si se quiere probar en velocidades mayores a 10km/h.

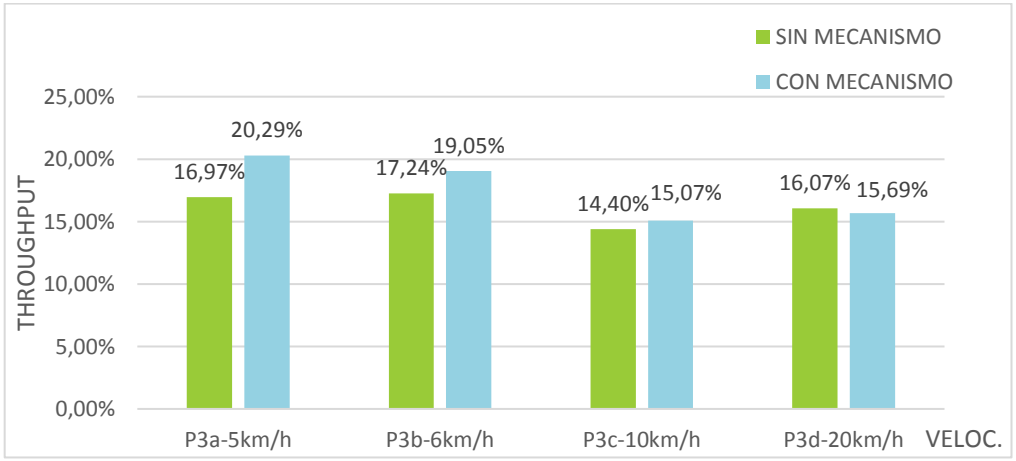

Figura 3.11. Resultados prueba 3- Throughput vs Velocidad

- **Tasa de reenvío**

La tasa de reenvío para esta prueba muestra resultados divididos. Usando el mecanismo de control de potencia se tiene menores tasas de reenvío en los dos casos siguientes:

- En la prueba 3a simulando con una velocidad promedio de 5km/h se puede determinar que el nivel promedio de reenvío se redujo en al menos 470 paquetes y en porcentajes en base a la Figura 3.12, se puede determinar que gracias al mecanismo de control de potencia el esfuerzo de los nodos se redujo en un 144%, a la vez que permite conseguir una mayor transmisión de paquetes.
- En la prueba 3c, considerando una velocidad de 10km/h el nivel de reenvío en los nodos se redujo en al menos 450 paquetes, es decir el esfuerzo realizado en la transmisión se redujo en un 102%.

Por tal motivo, podemos decir que el mecanismo de control de potencia aplicado en las pruebas 3 a y c resulta ser beneficioso para la red pues disminuye la carga de retransmisión e incrementa la eficiencia en la misma.

En las pruebas 3b y 3d al usar el mecanismo de control de potencia el nivel de reenvío de paquetes incrementa; en la prueba 3b el esfuerzo realizado por los nodos incrementa en un 23% lo cual es razonable ya que pudo darse el caso de que en algunos nodos la potencia de transmisión se redujo de forma considerable provocando así que tengan más saltos para llegar al destino. En la prueba 3d la tasa de reenvío de paquetes incrementa en el 139%, esto se justifica considerando que la velocidad de movimiento de los nodos es de 20 km/h, lo cual genera un cambio de topología arbitrario, obligando así a los nodos a buscar nuevas rutas para alcanzar el destino y con esto provocando reenvíos adicionales. La Figura 3.12 muestra un resumen de lo detallado.

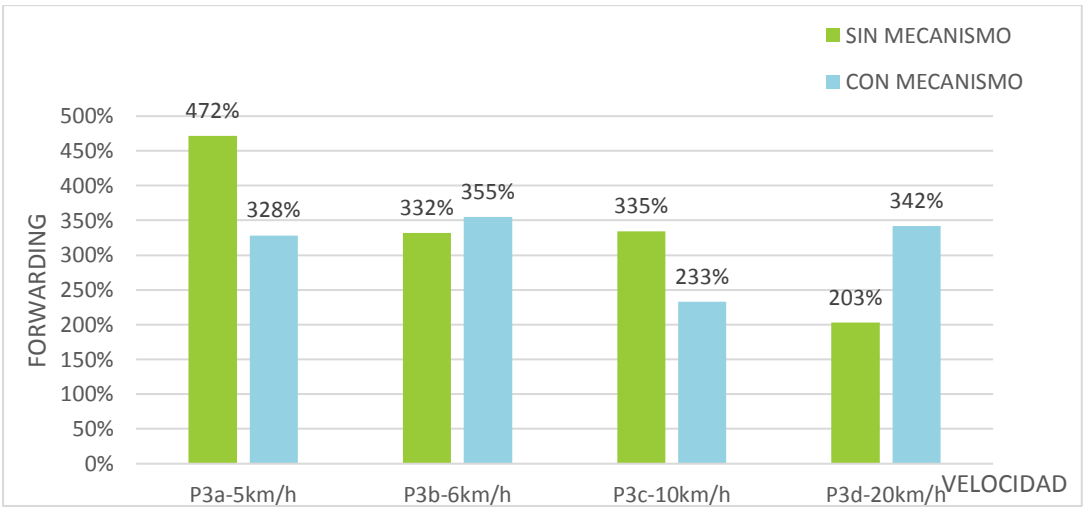

Figura 3.12. Resultados prueba 3- Reenvíos vs Velocidad

- **Tasa de pérdidas**

Conforme a la Figura 3.13; puede ser beneficioso usar el mecanismo de control de potencia para reducir el nivel de paquetes perdidos en una transmisión.

Las pruebas 3a, 3b, y 3c son casos favorables del uso del mecanismo de control de potencia para conseguir una menor tasa de pérdida de paquetes en la transmisión. Esto indica que, al usar el mecanismo, se evitó: numerosas colisiones, contención de canal o problemas a nivel físico que generaban la pérdida de paquetes. Sin embargo, para la prueba 3d, puede verse que tampoco resulta beneficioso el uso del mecanismo de control de potencia para intentar reducir el número de pérdida de paquetes. Se entiende que en esta prueba el mecanismo está reduciendo potencia en base a los vecinos que detecta, tal como lo hace en las pruebas 3a, b y c. El inconveniente es que los nodos se mueven

con mayor velocidad, por lo que en este caso no sería recomendable reducir la potencia al nivel que el mecanismo lo está haciendo.

Si se quiere implementar un control de potencia en una MANET donde los nodos manejen velocidades mayores o iguales a los 20km/h debe hacerse un ajuste en el cálculo de la mínima potencia de transmisión o en el paso de disminución de potencia para que no se reduzca mucho y pueda así tenerse transmisiones exitosas pese al movimiento que están presenciando los nodos.

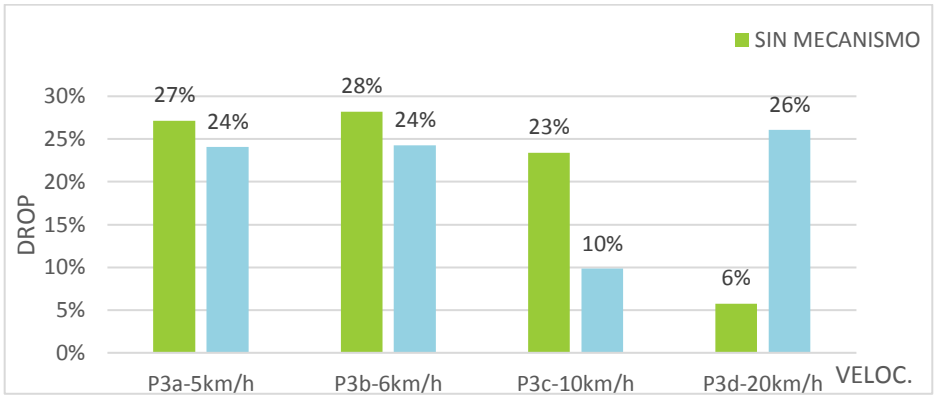

Figura 3.13. Resultados prueba 3- Pérdidas vs Velocidad

La Figura 3.14 muestra la variación de la potencia de transmisión promedio en algunos nodos en particular; seleccionados de las transmisiones en las cuales el throughput mejora al usar el mecanismo de control de potencia. Como se puede observar, en la mayoría de los casos el nivel de potencia que usa cada nodo se redujo al usar el mecanismo de control de potencia. En los casos en los cuales el nivel de potencia es mayor que la potencia por defecto que considera el simulador debe tomarse en cuenta que el incremento es mínimo y que gracias al mismo se tuvo un mayor número de transmisiones exitosas. Por lo tanto, el mecanismo es beneficioso para la red en cuanto a rendimiento, reducción de sobrecarga en la red y optimización del consumo de potencia.

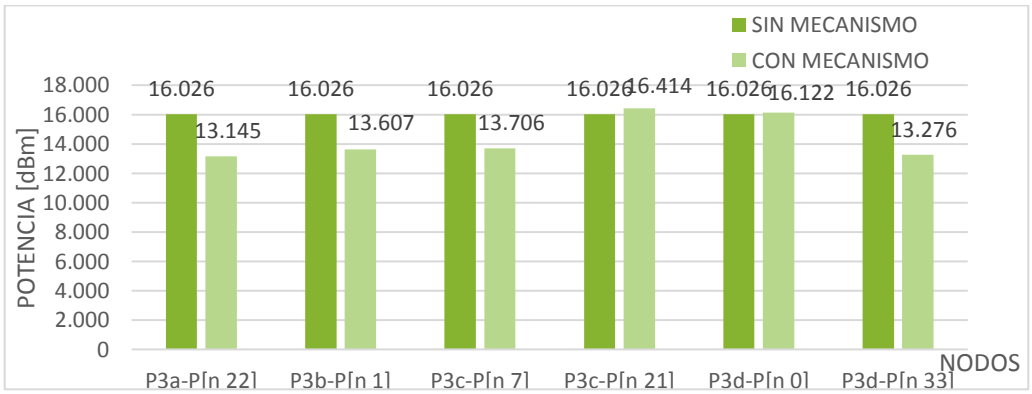

Figura 3.14. Potencia media de transmisión – Pruebas 3

3.2.4 PRUEBA 4: THROUGHPUT vs VELOCIDAD - RANDOMWAYPOINT

Los resultados en base a la Tabla 3.6 muestran que en tres de los cuatro casos que considera esta prueba es favorable el uso del mecanismo de control de potencia ya que permite tener un mayor nivel transmisiones exitosas. A continuación, se analiza los resultados.

- **Throughput**

Conforme a la Figura 3.15, el throughput promedio obtenido para las transmisiones de las pruebas 4a, 4c y 4d que consideran control de potencia es mayor que para los casos que no se considera control de potencia.

Se puede apreciar que el nivel de throughput es mayor en cuanto la velocidad a la que se mueven los nodos es menor. Así mismo, el mayor incremento de throughput en las pruebas que incorporan el mecanismo de control de potencia se da para el caso en el que los nodos se mueven con menor velocidad, es decir 5km/h. Para 10 y 20 km/h el nivel de throughput disminuye notablemente.

La prueba 4b presenta un mayor rendimiento en la transmisión sin considerar control de potencia, de las 10 pruebas realizadas 6 resultaron favorables para este caso. Esto se debe a que con el modelo RandomWaypoint los nodos que se desplazaban con 6km/h tendían a terminar en una distribución muy cerrada y no simétrica como se prevé deberían estar los nodos. Al tener una distribución diferente a la que se consideró para el diseño del mecanismo se espera que el mismo no funcione como lo ideado.

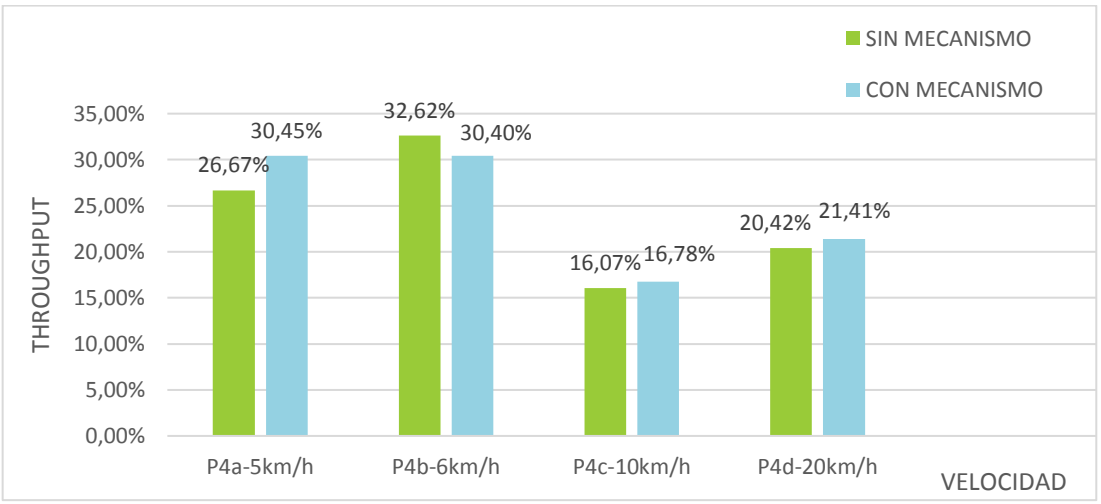

Figura 3.15. Resultados prueba 4- Throughput vs Velocidad

- **Tasa de reenvío**

  En esta prueba sin duda el mecanismo de control de potencia permite mejorar las tasas de reenvío para casi todos los casos; la Figura 3.16 muestra se reduce el esfuerzo de los nodos para lograr transmisiones exitosas.

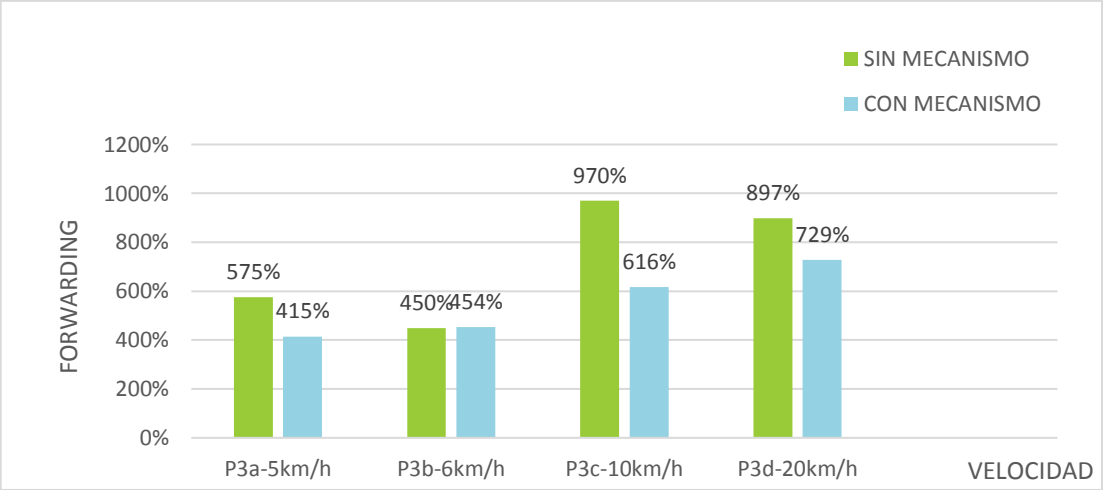

Figura 3.16. Resultados prueba 4- Reenvíos vs Velocidad

Un menor número de retransmisiones es muestra de que el número de saltos se redujo. Lo cual indica que el mecanismo diseñado incrementó la potencia a tal punto de tener enlaces más largos y por ende reducir el número de saltos. Esto beneficioso para la red ya que reduce la carga de retransmisión, colisiones, tiempos de cola y con ello tiempo de transmisión.

- **Tasa de pérdidas**

El nivel de paquetes perdidos tanto para las transmisiones que incluyen control de potencia como en las que no incluyen es semejante; tal como se puede ver en la Figura 3.17.

Hasta ahora en todos los casos que se aplicó control de potencia a nodos que se desplazaban a 20km/h no resultó muy favorable. Sin embargo, al momento de analizar la tasa de pérdida de paquetes a la mayor velocidad en los nodos (20km/h) usando control de potencia se obtiene un resultado favorable. Con esto podemos decir que el mecanismo puede ser útil incluso en escenarios de altas velocidades.

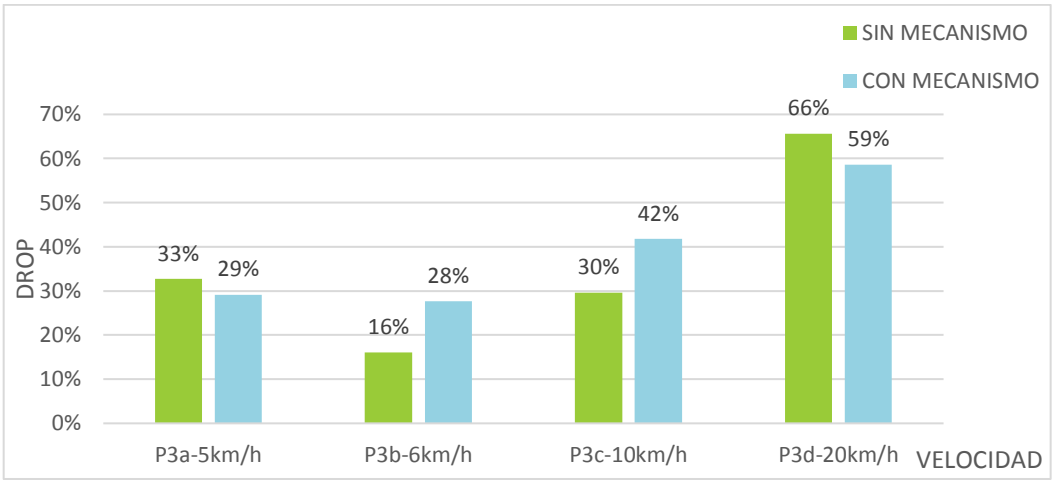

Figura 3.17. Resultados prueba 4- Pérdidas vs Velocidad

La potencia media de transmisión, al igual que en las pruebas anteriores, es menor que la potencia pre definida por el simulador para redes ad-hoc con IEEE 802.11. Véase Figura 3.18.

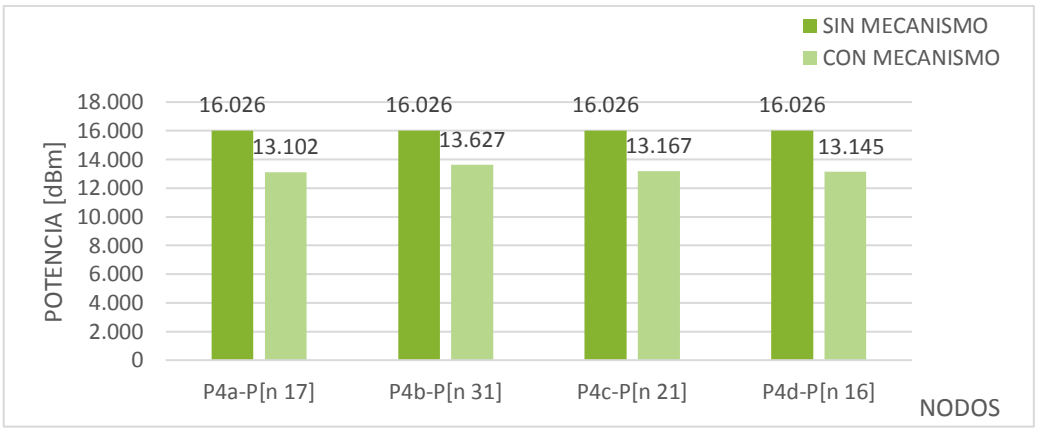

Figura 3.18. Potencia media de transmisión – Pruebas 4

Luego de todas las pruebas realizadas se puede determinar que el mecanismo diseñado y simulado para controlar la potencia de transmisión en una MANET resulta beneficioso en la mayoría de los casos. Esto se verifica en las simulaciones donde el mecanismo permite incrementar el rendimiento, reducir la carga de retransmisión, y en algunos casos reduce el nivel de paquetes perdidos.

Se pueden ver casos ideales como los de las pruebas: 3a (5km/h), 3c (10km/h), 4a (5km/h), 4d (20km/h) en los cuales al usar el mecanismo se incrementa el rendimiento en la transmisión, se reduce el nivel de paquetes retransmitidos y se obtiene una menor pérdida de paquetes.

Está claro que el mecanismo es perfectible ya que también puede obtenerse una reducción en el nivel de los parámetros que se han analizado (throughput, reenvío, pérdidas); así como pueden obtenerse resultados combinados; es decir, uno de los parámetros incrementa mientras que los dos reducen o viceversa.

Adicional a la mejora de los parámetros descritos, el mecanismo de control de potencia diseñado también permite reducir el consumo de potencia, pues en base a la potencia media se verificó que es posible que los nodos tengan transmisiones exitosas aun cuando actúan con un menor nivel de potencia, lo cual nos lleva a un menor gasto de energía.

Por el contrario, si la potencia de transmisión que determina el mecanismo es mayor a la potencia que define NS-3 vamos a tener como resultado enlaces más largos (menor tasa de reenvío), menor número de colisiones y de seguro una mayor tasa de recepción de paquetes.

Finalmente, se puede indicar que el mecanismo de control de potencia que se ha implementado para una MANET es de utilidad siempre y cuando se tenga un área de aplicación que considere una distribución uniforme de los nodos.

4. CONCLUSIONES

El mecanismo de control de potencia diseñado e implementado en NS-3 en este proyecto de titulación cumple con los objetivos trazados para este proyecto pues en base a las pruebas realizadas se determina que en la mayoría de los casos el uso del mecanismo resulta favorable para la red en cuanto a eficiencia, carga de reenvío, pérdida de paquetes, y menor consumo de potencia.

NS-3 es un simulador de red muy completo, modular y con mucha información de guía. Es una de las mejores opciones a la hora de simular escenarios realistas ya que permite agregar a las simulaciones una serie de características como, por ejemplo, diferentes tecnologías de transmisión, protocolos de comunicación, canales, modelos de movilidad, modificación de sus clases e importación de trazas de movilidad externas con ciertos softwares compatibles como es el caso de Bonnmotion. Herramienta muy útil para agregar modelos de movilidad en los nodos, muy flexible y de fácil vinculación con NS-3.

El mecanismo de control de potencia diseñado para este proyecto opera mejor en densidades ligeras de nodos, pues mientras mayor sea la densidad considerada, la cota mínima de la potencia de transmisión será más pequeña y por ende no se podrá alcanzar tantas transmisiones exitosas.

A través de las pruebas que consideran variación de velocidad se pudo determinar que la velocidad con la que se mueve un nodo tiene una relación directamente proporcional con la potencia de transmisión, es decir para los escenarios desarrollados en este proyecto, mientras mayor velocidad tenga un nodo, mayor debe ser la potencia de transmisión para mantener un enlace estable ante la topología de la red.

Con este trabajo se demuestra que la colaboración entre capas resulta útil para las redes inalámbricas Ah-hoc ya que permite la interacción de diferentes parámetros, con lo cual se logra tomar mejores decisiones para una transmisión. Tal es el caso de este proyecto, en el que se coordinó capa física con capa de red para así lograr una mayor eficiencia.

El mecanismo funciona de forma adecuada en nodos que manejen velocidades de hasta 10km/h. Si se quiere agregar un control de potencia en MANETs que tengan nodos desplazándose con velocidades mayores, tal es el caso de 20 km/h, se debe realizar un ajuste en la mínima potencia de transmisión, así como en el paso de potencia considerados. De este modo se podrá garantizar un mayor número de recepciones exitosas.

La potencia de transmisión promedio para la mayoría de las pruebas se redujo en al menos 20mw, con una tasa de recepción muy buena. Lo cual es muy importante para este tipo de redes inalámbricas considerando que su aspecto crítico es el consumo de baterías. Esto nos muestra que aún hay mucho por hacer en cuanto al control de potencia para redes ad-hoc.

El mecanismo de control de potencia que se ha presentado en este trabajo se enfoca a MANETs que asumen una distribución de nodos uniforme. Sería interesante plantear un mecanismo que controle la potencia de transmisión en escenarios no uniformes.

El número mínimo de vecinos que se ha considerado para el mecanismo del presente proyecto ha sido 4, teniendo en cuenta que debe existir al menos un vecino por cada punto cardinal. Esto considerando que se trabaja en un área que asume una distribución simétrica. Un trabajo futuro podría centrarse en definir el número mínimo de vecinos en base al área de simulación y a la densidad de nodos del escenario.

5. REFERENCIAS BIBLIOGRÁFICAS

[1] K. Vyas and A. Chaturvedi, “Comparative Analysis of Routing Protocols in Ad-hoc Networks,” pp. 692–697, 2014.

[2] B. K. Panda, J. Swain, D. P. Mishra, and B. Sahu, “Analysis of effect of mobility and transmission power on AODV and DSR in mobile Adhoc network,” *IFIP Int. Conf. Wirel. Opt. Commun. Networks, WOCN*, 2014.

[3] L. F. Urquiza, “Escuela Politécnica Nacional Maestría en Conectividad y Redes de Telecomunicaciones Mobile Ad hoc Networks,” 2016.

[4] S. Navarro, “Redes Wireless Ad-Hoc,” Capítulo 3, Universidad de Sevilla, 2006.

[5] Z. Hussain and R. Balakrishna, “A Survey on Manets – Types , Characteristics , Applications and Protocols used”, pp. 40-43, 2014.

[6] C. R. Das, “Performance comparison of cache invalidation strategies for Internet-based mobile ad hoc networks,” *2004 IEEE Int. Conf. Mob. Ad-hoc Sens. Syst. (IEEE Cat. No.04EX975)*, pp. 104–113, 2004.

[7] X. Fan, J. Cao, H. Mao, W. Wu, Y. Zhao, and C. Xu, “Web access patterns enhancing data access performance of cooperative caching in IMANETs,” *Proc. - IEEE Int. Conf. Mob. Data Manag.*, vol. 2016–July, pp. 50–59, 2016.

[8] X. Fan, J. Cao, Y. Liu, and Y. He, “Gossip-based cooperative caching for mobile phone games in IMANETs,” *Proc. Int. Conf. Parallel Distrib. Syst. - ICPADS*, no. 863, pp. 465–472, 2011.

[9] K. Emara, “Safety-aware Location Privacy in VANET: Evaluation and Comparison,” *IEEE Trans. Veh. Technol.*, vol. XX, no. XX, pp. 1–1, 2017.

[10] S. Zaidi, S. Bitam, and A. Mellouk, “Enhanced User Datagram Protocol for Video Streaming in VANET,” 2017.

[11] Y. Qian and N. Moayeri, “Design of secure and application-oriented vanets,” *IEEE Veh. Technol. Conf.*, pp. 2794–2799, 2008.

[12] L. Felipe and U. Aguiar, “Contribution to the design of VANET routing protocols for realistic urban environments ,” February, 2016.

[13] I. Bekmezci, O. K. Sahingoz, and S. Temel, “Flying Ad-Hoc Networks (FANETs): A

survey," *Ad Hoc Networks*, vol. 11, no. 3, pp. 1254–1270, 2013.

[14] M. B. Yassein, "Flying Ad-Hoc Networks: Routing Protocols, Mobility Models, Issues," *Int. J. Adv. Comput. Sci. Appl.*, vol. 7, no. 6, pp. 162–168, 2016.

[15] C. Barrado, R. Meseguer, J. López, E. Pastor, E. Santamaria, and P. Royo, "Wildfire monitoring using a mixed air-ground mobile network," *IEEE Pervasive Comput.*, vol. 9, no. 4, pp. 24–32, 2010.

[16] J. Zhu and B. Bensaou, "Power Control Protocols for Wireless," in Algorithms and Protocols for Wireless and Mobile Ad Hoc Networks, Hoboken: John Wiley & Sons, Inc., 2009, pp. 315–352.

[17] P. Reddy, "A Hybrid Genetic Fuzzy approach for power control cross layer MAC protocol in Wireless Network," pp. 181–186, 2015.

[18] J. Park and S. Sahni, "Maximum Lifetime Routing In Wireless Sensor Networks," vol. 12, no. 4, pp. 609–619, 2005.

[19] L. Li and J. Y. Halpern, "Minimum-Energy Mobile Wireless Networks Revisited," *IEEE Int. Conf. Commun. (ICC 2001)*, vol. 1, pp. 278–283, 2002.

[20] S. Agarwal, R. H. Katz, S. V. Krishnamurthy, and S. K. Dao, "Distributed power control in ad-hoc wireless networks," *12th IEEE Int. Symp. Pers. Indoor Mob. Radio Commun. PIMRC 2001. Proc. (Cat. No.01TH8598)*, p. F-59-F-66, 2001.

[21] S. Harold and V. . Vijayalakshmi.A, "Enhanced Power Control MAC Protocol for Single Cell, Dense Wireless Ad Hoc Networks," *Int. J. Comput. Appl.*, vol. 47, no. 20, pp. 27–32, 2012.

[22] V. S. Mansouri, M. Mohammad, and B. H. Khalaj, "Dynamic Scheduling MAC Protocol for Large Scale Sensor Networks," *pp.* 121-125, 2005.

[23] J. P. Monks, V. Bharghavan, and W.-M. W. Hwu, "A power controlled multiple access protocol for wireless packet\nnetworks," *Proc. IEEE INFOCOM 2001. Conf. Comput. Commun. Twent. Annu. Jt. Conf. IEEE Comput. Commun. Soc. (Cat. No.01CH37213)*, vol. 1, pp. 219–228, 2001.

[24] M. A. M. Al-Tahrawi, M. Ismail, R. Nordin, and T. Yuwono, "Performance of AODV and OLSR routing protocol in a hybrid sensor and vehicular network 802.11p," *Proc. - ICWT 2016 2nd Int. Conf. Wirel. Telemat. 2016*, pp. 140–145, 2017.

[25] L. Felipe, U. Aguiar, A. I. Co-directora, C. Tripp, and B. Barcelona, "Proyecto final

de máster: Design and implementation of routing protocols with anonymity for vehicular ad-hoc networks in urban environments," 2012.

[26] M. A. Rahman, F. Anwar, J. Naeem, and M. S. M. Abedin, "A simulation based performance comparison of routing protocol on Mobile Ad-hoc Network (proactive, reactive and hybrid)," *Int. Conf. Comput. Commun. Eng.*, no. May, pp. 1–5, 2010.

[27] V. Kawadia and P. R. Kumar, "Principles and protocols for power control in wireless ad hoc networks," *IEEE J. Sel. Areas Commun.*, vol. 23, no. 1, pp. 76–88, 2005.

[28] P. Gupta and P. R. Kumar, "The capacity of wireless networks," *IEEE Trans. Inf. Theory*, vol. 46, no. 2, pp. 388–404, 2000.

[29] S. Park, R. Sivakumar, "Load-Sensitive Transmission Power Control in Wireless Ad-hoc Networks," pp. 42–46, 2000.

[30] S. Sharma and R. Mishra, "A Cross Layer Approach for Intrusion Detection in MANETs," vol. 93, no. 9, pp. 34–41, 2014.

[31] S. Shakkottai, T. S. Rappaport, and P. C. Karlsson, "Cross-layer design for wireless networks," *Commun. Mag. IEEE*, vol. 41, no. 10, pp. 74–80, 2003.

[32] X. Ren and J. Zhang, "A novel cross-layer architecture for wireless protocol stacks," *2010 Int. Conf. Multimed. Technol. ICMT 2010*, no. 2007, 2010.

[33] K. Nahm, A. Helmy, and C. C. Jay Kuo, "Cross-layer interaction of TCP and ad hoc routing protocols in multihop IEEE 802.11 networks," *IEEE Trans. Mob. Comput.*, vol. 7, no. 4, pp. 458–469, 2008.

[34] D. Kliazovich and F. Granelli, "Why cross-layer? Its advantages and disadvantage," in *Cross layer designs in wlan systems, Edited*: P. Zorba, C. Skianis, and C. Verikoukis, 2015.

[35] K. Sharma, N. Mittal, and P. Rathi, "Comparative Analysis of Routing Protocols in Ad-hoc Networks," *Int. J. Adv. Sci. Technol.*, vol. 69, pp. 1–12, 2014.

[36] Y. Mai, Y. Bai, and N. Wang, "Performance Comparison and Evaluation of the Routing Protocols for MANETs Using NS3," vol. 5, pp. 187–195, 2017.

[37] B. Soujanya, T. Sitamahalakshmi, and C. Divakar, "Study of Routing Protocols in Mobile Ad Hoc Networks," *Int. J. Eng. Sci. Technol.*, vol. 3, no. 4, pp. 2622–2631, 2011.

[38] C. K. Jha, P. Kumar, P. D. Parihar, and L. Garg, "Realisation of link state routing

protocol and advance distance vector in different IP schema," *Proc. - 2014 6th Int. Conf. Comput. Intell. Commun. Networks, CICN 2014*, pp. 486–491, 2014.

[39] M. Baharloo, R. Hajisheykhi, M. Arjomand, and A. H. Jahangir, "An analytical performance evaluation for WSNs using loop-free bellman ford protocol," *Proc. - Int. Conf. Adv. Inf. Netw. Appl. AINA*, pp. 568–571, 2009.

[40] S. Waleed, M. Faizan, M. Iqbal, and M. I. Anis, "Demonstration of Single Link Failure Recovery using Bellman Ford and Dijkstra Algorithm in SDN," pp. 0–3.

[41] V. Rishiwal, M. Yadav, O. Singh, and M. Yadav, "Analysis of adhoc routing protocols: A retrospective view," *Proc. - 2016 Int. Conf. Adv. Comput. Commun. Autom. ICACCA 2016*, 2016.

[42] L. Rivoirard, M. Wahl, P. Sondi, M. Berbineau, and D. Gruyer, "Performance evaluation of AODV , DSR , GRP and OLSR for VANET with real-world trajectories," pp. 1–7.

[43] M. Imran and M. A. Qadeer, "Evaluation Study of Performance Comparison of Topology based Routing Protocol ; AODV and DSDV in MANET," pp. 208–212, 2016.

[44] L. Mejaele and E. Oketch Ochola, "Effect of varying node mobility in the analysis of black hole attack on MANET reactive routing protocols," *2016 Inf. Secur. South Africa - Proc. 2016 ISSA Conf.*, pp. 62–68, 2016.

[45] Istikmal, "Analysis And Evaluation Optimization Dynamic Source Routing (DSR) Protocol in Mobile Adhoc Network Based on Ant Algorithm," *Int. Conf. Inf. Commun. Technol.*, pp. 400–404, 2013.

[46] M. Yadav, "Multi-Hop Wireless Ad-Hoc Network Routing Protocols- A Comparative Study of DSDV, TORA, DSR And AODV," 2015.

[47] D. S. Perkins C., "RFC 3561," Santa Barbara, 2003.

[48] E. M. Royer and C.-K. Toh, "A review of current routing protocols for ad hoc mobile wireless networks," *Pers. Commun. IEEE*, vol. 6, no. 2, pp. 46–55, 1999.

[49] R. Ammon, "NS3- Introduction." [Online]. Disponible: https://www.nsnam.org/docs/tutorial/html/introduction.html. [Accedido: 11-Nov-2017].

[50] R. M. Stallman, *Software libre para una sociedad libre*, Traf. de S. Madrid, 2004.

[51] “GNU GENERAL PUBLIC LICENSE,” 1991. [Online]. Disponible: https://www.gnu.org/licenses/old-licenses/gpl-2.0.txt. [Accedido: 11-Nov-2017].

[52] M. Solera and F. Ruiz, “Simulación de Protocolos de Enrutamiento para Redes Móviles Ad-Hoc Mediante la Herramienta de Simulación NS-3,” 2014. [Online]. Disponible: http://repositorio.cedia.org.ec/handle/123456789/960. [Accedido: 11-Nov-2017].

[53] “Waf - project.” [Online]. Disponible: https://github.com/waf-project/waf. [Accedido: 11-Nov-2017].

[54] T. Nagy, “The Waf Book.” [Online]. Disponible: https://waf.io/book/. [Accedido: 11-Nov-2017].

[55] “NS3- Tutorial, Getting Started.” [Online]. Disponible: https://www.nsnam.org/docs/tutorial/html/getting-started.html. [Accedido: 11-Nov-2017].

[56] Tom Henderson, “NS3- Manual Release 3.26,” 2016. [Online]. Disponible: https://www.nsnam.org/docs/release/3.26/manual/html/index.html. [Accedido: 11-Nov-2017].

[57] R. Ammon, “NS3- Configuration and Attributes.” [Online]. Disponible: https://www.nsnam.org/docs/manual/html/attributes.html#. [Accedido: 11-Nov-2017].

[58] R. Ammon, “NS3- Doxygen Mobility,” 2017. [Online]. Disponible: https://www.nsnam.org/doxygen/group__mobility.html. [Accedido: 11-Nov-2017].

[59] T. Henderson, “NS3- Model Library, Release 3.26,” 2016. [Online]. Disponible: https://www.nsnam.org/docs/release/3.26/models/html/index.html. [Accedido: 11-Nov-2017].

[60] R. Ammon, “NS3- Tutorial, Conceptual Overview,” 2017. [Online]. Disponible: https://www.nsnam.org/docs/tutorial/html/conceptual-overview.html. [Accedido: 11-Nov-2017].

[61] R. Ammon, “NS3- Doxygen Core,” 2015. [Online]. Disponible: https://www.nsnam.org/doxygen/group__core.html. [Accedido: 11-Nov-2017].

[62] IEEE 802.11, “Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications,” vol. 2016, 2012.

[63] S. Kafaie, M. Hossam Ahmed, Y. Chen, and O. A. Dobre, “Performance Analysis of Network Coding with IEEE 802.11 DCF in Multi-Hop Wireless Networks,” *IEEE Trans. Mob. Comput.*, vol. 1233, no. c, pp. 1–1, 2017.

[64] N. Boulila, “Mobility impact on EDCA,” pp. 524–529, 2017.

[65] Tom Henderson, “NS3- Propagation Models, Release 3.26,” 2016. [Online]. Disponible: https://www.nsnam.org/docs/release/3.26/models/html/propagation.html#propagation. [Accedido: 11-Nov-2017].

[66] K. Amjad and A. J. Stocker, “Impact of node density and mobility on the performance of AODV and DSR in MANETS,” *Ieee*, no. November, pp. 61–65, 2010.

[67] M. J. B. Lee, “Comparison between path-loss prediction models for wireless system design ,” *Antennas Propag. Soc. Int. Symp. 2001. IEEE* , vol. 2, no. 8, p. 3, 2001.

[68] A. Medeisis and A. Kajackas, “On the use of the universal Okumura-Hata propagation prediction model in rural areas,” *Veh. Technol. Conf. Proceedings, 2000. VTC 2000-Spring Tokyo. 2000 IEEE 51st*, vol. 3, pp. 1815–1818 vol.3, 2000.

[69] “Jakes Propagation Loss Model.” [Online]. Disponible: https://www.nsnam.org/doxygen/classns3_1_1_jakes_propagation_loss_model.html#details. [Accedido: 11-Nov-2017].

[70] F. Comeau, S. Sivakumar, W. J. Phillips, and W. Robertson, “A Clustered Wireless Sensor Network Model Based on Log–Distance Path Loss,” *6th Annu. Commun. Networks Serv. Res. Conf. (cnsr 2008)*, pp. 366–372, 2008.

[71] “Three Log Distance Propagation Loss Model.” [Online]. Disponible: https://www.nsnam.org/doxygen/classns3_1_1_three_log_distance_propagation_loss_model.html. [Accedido: 11-Nov-2017].

[72] B. M. Mughal, A. A. Wagan, and H. Hasbullah, “Analyzing safety beacon loss rate in VANETs with two-ray ground and Nakagami propagation models,” *2011 Natl. Postgrad. Conf. - Energy Sustain. Explor. Innov. Minds, NPC 2011*, vol. 178, 2011.

[73] H. T. Friis, “A Note on a Simple Transmission Formula,” *Proc. IRE*, vol. 34, no. 5, pp. 254–256, 1946.

[74] Tom Henderson, “NS3- Models, Wifi-user Release 3.26,” 2016. [Online].

Disponible: https://www.nsnam.org/docs/release/3.26/models/html/wifi-user.html#yanswifichannelhelper. [Accedido: 11-Nov-2017].

[75] M. Lacage and T. R. Henderson, “Yet another network simulator,” *Proceeding from 2006 Work. ns-2 IP Netw. simulator - WNS2 ’06*, p. 12, 2006.

[76] R. Ammon, “NS3- Doxygen AODV Routing Module.” [Online]. Disponible: https://www.nsnam.org/doxygen/group__aodv.html. [Accedido: 11-Nov-2017].

[77] N. Aschenbruck, “README BonnMotion,” *BonnMotion*, p. 29, 2013.

[78] “QualNet Network Simulator Software.” [Online]. Disponible: http://web.scalable-networks.com/qualnet-network-simulator-software. [Accedido: 11-Nov-2017].

[79] Ari Keränen, “The ONE- The Opportunistic Network Environment simulator,” 2008. [Online]. Disponible: https://akeranen.github.io/the-one/. [Accedido: 11-Nov-2017].

[80] “BonnMotion - Download.” [Online]. Disponible: http://sys.cs.uos.de/bonnmotion/download.shtml. [Accedido: 11-Nov-2017]

[81] T. Camp, J. Boleng, and V. Davies, “A Survey of Mobility Models for Ad Hoc Network Research,” 5, 2002.

[82] B. Liang and Z. J. Haas, “Predictive distance-based mobility management for PCS networks,” in *IEEE INFOCOM ’99. Conference on Computer Communications. Proceedings. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. The Future is Now (Cat. No.99CH36320)*, 1999, pp. 1377–1384 vol.3.

[83] M. Z. Oo and M. Othman, “Analysis of single-path and multi-path AODVs over Manhattan Grid mobility model for mobile Ad hoc networks,” *ICEIE 2010 - 2010 Int. Conf. Electron. Inf. Eng. Proc.*, vol. 1, no. Iceie, pp. 214–218, 2010.

[84] J. Broch, D. A. Maltz, D. B. Johnson, Y.-C. Hu, and J. Jetcheva, “A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols,” *Proc. {ACM/IEEE} Int. Conf. Mob. Comput. Netw.*, pp. 85–97, 1998.

[85] “std::Vector.” [Online]. Disponible: http://www.cplusplus.com/reference/vector/vector/. [Accedido: 12-Oct-2017].

[86] K. Capt, “Maximum power for 802.11,” 2011. [Online]. Disponible: http://www.dslreports.com/faq/2998. [Accedido: 11-Nov-2017].

[87] M. Troi, “WLAN ENGINEERING,” 2014. [Online]. Disponible:

https://wlan1nde.wordpress.com/2014/11/26/wlan-maximum-transmission-power-etsi/. [Accedido: 11-Nov-2017].

[88] C. Campolo, R. Scopigno, and A. Molinaro, *Vehicular ad hoc Networks*. Cham: Springer International Publishing, 2015.

[89] “Learn AWK.” [Online]. Disponible: https://www.tutorialspoint.com/awk/awk_overview.htm. [Accedido: 11-Nov-2017].

[90] “GNU Awk - Guía de usuario.” [Online]. Disponible: https://www.gnu.org/software/gawk/manual/gawk.html#Foreword3. [Accedido: 11-Nov-2017].

6. ANEXOS

ANEXO I. Prerrequisitos para la instalación de NS-3

ANEXO II. Modificación en capa física de NS-3

ANEXO III. Código del mecanismo de control de potencia

ANEXO IV. Código de escenarios

ANEXO V. Código para la generación de LOGS

ANEXO VI. Código de los filtros en GAWK

ANEXO VII. Resultados de todas las pruebas realizadas

ANEXO I

Prerrequisitos para la instalación de NS-3

Instalar C++

```
sudo apt-get install gcc g++ python
```

Instalar Mercurial

```
sudo apt-get install mercurial
```

Instalar Bazaar

```
sudo apt-get install bzr
```

Instalar Python

```
sudo apt-get install gcc g++ python python-dev
```

Depurador

```
sudo apt-get install gdb valgrind
```

Librería GSL (GNU Scientific Library)

```
sudo apt-get install gsl-bin libgsl0-dev libgsl0ldbl
```

Analizador flex y generador bison

```
sudo apt-get install flex bison libfl-dev
```

Instalar tcpdump (lectura de salidas pcap)

```
sudo apt-get install tcpdump
```

Instalar librerías XML

```
sudo apt-get install libxml2 libxml2-dev
```

Instalar librerías estadísticas

```
sudo apt-get install sqlite sqlite3 libsqlite3-dev
```

Sistema GTK (toolkit basado en objetos)

```
$sudo apt-get install libgtk2.0-0 libgtk2.0-dev
```

Máquinas virtuales en ns-3

```
sudo apt-get install vtun lxc
```

Chequeo de código python

```
sudo apt-get install uncrustify
```

Documentación (doxygen)

```
sudo apt-get install doxygen graphviz imagemagick
sudo apt-get install texlive texlive-extra-utils texlive-latex-extra
```

Visualización de manuales y tutorial (con sphinx y dia)

```
sudo apt-get install python-sphinx dia
```

Instalación de visualizador pyviz de ns-3

```
sudo apt-get install python-pygraphviz python-kiwi python-pygoocanvas
libgoocanvas-dev
```

Módulo openflow

```
sudo apt-get install libboost-signals-dev libboost-filesystem-dev
```

Soporte de emulador MPI (Message Passing Interface)

```
sudo apt-get install openmpi-bin openmpi-common openmpi-doc libopenmpi-
dev
```

Enlaces modificados con python

```
sudo apt-get install gcc-multilib
```

ANEXO II

Modificaciones en Capa Física NS-3

- ANEXO II-A: Limitaciones en capa física

Este anexo muestra las modificaciones realizadas a la capa física *YansWifiPhy* para evitar limitaciones referentes a la potencia de transmisión.

```
//Configuracion de los niveles de potencia, inicialmente era 1
  .AddAttribute ("TxPowerLevels",
                 "Number of transmission power levels Disponible
between "
                 "TxPowerStart and TxPowerEnd included.",
                 UintegerValue (16),
                 MakeUintegerAccessor (&YansWifiPhy::m_nTxPower),
                 MakeUintegerChecker<uint32_t> ())
   //Definicion de la maxima potencia de transmision, inicialmente era
16dBm
    .AddAttribute ("TxPowerEnd",
                 "Maximum Disponible transmission level (dbm).",
                 DoubleValue (27.0206),
                 MakeDoubleAccessor (&YansWifiPhy::SetTxPowerEnd,
                                     &YansWifiPhy::GetTxPowerEnd),
                 MakeDoubleChecker<double> ())
```

ANEXO III

Código del Mecanismo de Control de Potencia

- ANEXO III-A: Código general para MANETs

Este anexo incluye el mecanismo de control de potencia y la función de actualización periódica del mismo.

```
  void
RoutingProtocol::UpdateMControl ()
{
  NS_LOG_FUNCTION (this);

  //DIRECCIONES IP
   Ptr<Ipv4> ipv4 = m_ipv4 -> GetObject<Ipv4>();
   Ipv4InterfaceAddress iface = m_ipv4->GetAddress (1,0);

  //ACCESO A CAPA FISICA
  Ptr<NetDevice> dev = m_ipv4->GetNetDevice (m_ipv4-
>GetInterfaceForAddress (iface.GetLocal ()));
  Ptr<WifiNetDevice> wifi = dev->GetObject<WifiNetDevice> ();
  Ptr<WifiMac> mac = wifi->GetMac ();
  Ptr<WifiPhy> phy = mac->GetWifiPhy();
  Ptr<YansWifiPhy> phyles = phy ->GetObject<YansWifiPhy> ();

  //MECANISMO
  double pot_act_d=phyles->GetTxPowerStart ();
  int num_n=m_nb.NumVecinos();
  double pot_act_mw;
  double p_mw;
  double p_d;//Potencia actual en dBm
  double delta_pow=2;  //(mw)

  //MECANISMO PARA ESCENARIOS MANET

    if (m_nb.NumVecinos()<4)
    {
           pot_act_mw=pow(10,pot_act_d/10);// dbm a mw
           p_mw=pot_act_mw+delta_pow;//potencia total en mw
           p_d=10*log10(p_mw); //potencia total del mecanismo en dbm

             if (p_d>20)
             {
             pot_act_mw=pow(10,p_d/10);// dbm a mw
             p_mw=pot_act_mw-delta_pow;// mw
             p_d=10*log10(p_mw);
             }
      }

    else if (m_nb.NumVecinos()==4)
   {

    p_d=pot_act_d;
    }
```

```
  else if (m_nb.NumVecinos()>=5)
  {
            pot_act_mw=pow(10,pot_act_d/10);// dbm a mw
            p_mw=pot_act_mw-delta_pow;//potencia total en mw
            p_d=10*log10(p_mw);

            if(p_d<Mín. power tx) // Según sea el caso
             {
              pot_act_mw=pow(10,p_d/10);// dbm a mw
              p_mw=pot_act_mw+delta_pow;// mw
              p_d=10*log10(p_mw);

              if (p_d>20)
              {
              pot_act_mw=pow(10,p_d/10);// dbm a mw
              p_mw=pot_act_mw-delta_pow;// mw
              p_d=10*log10(p_mw);
              }
             }
   }

   else if (m_nb.NumVecinos()==0)
  {
    p_d=20.00;
  }

   phyles->SetTxPowerStart(p_d);

  m_upmcontrol.Schedule (Seconds (1));
 }
```

- ANEXO III-B: Variación para escenarios preliminares

```
void
RoutingProtocol::UpdateMControl ()
{
  NS_LOG_FUNCTION (this);

  //DIRECCIONES IP
   Ptr<Ipv4> ipv4 = m_ipv4 -> GetObject<Ipv4>();
   Ipv4InterfaceAddress iface = m_ipv4->GetAddress (1,0);
  //ACCESO A CAPA FISICA
  Ptr<NetDevice> dev = m_ipv4->GetNetDevice (m_ipv4-
>GetInterfaceForAddress (iface.GetLocal ()));
  Ptr<WifiNetDevice> wifi = dev->GetObject<WifiNetDevice> ();
  Ptr<WifiMac> mac = wifi->GetMac ();
  Ptr<WifiPhy> phy = mac->GetWifiPhy();
  Ptr<YansWifiPhy> phyles = phy ->GetObject<YansWifiPhy> ();
   //MECANISMO
  double pot_act_d=phyles->GetTxPowerStart ();
  int num_n=m_nb.NumVecinos();
  double pot_act_mw;
  double p_mw;
  double p_d;//Potencia actual en dBm
  double delta_pow=2;  //(mw)
    //MECANISMO PARA 5 NODOS
```

```
    if (m_nb.NumVecinos()<2)
 {
          pot_act_mw=pow(10,pot_act_d/10);// dbm a mw
          p_mw=pot_act_mw+delta_pow;//adicion
          p_d=10*log10(p_mw); //potencia total del mecanismo en dbm
            if (p_d>20)
            {
            pot_act_mw=pow(10,p_d/10);// dbm a mw
            p_mw=pot_act_mw-delta_pow;// resta
            p_d=10*log10(p_mw); //potencia total del mecanismo en dbm
            }
  }

  phyles->SetTxPowerStart(p_d);

 m_upmcontrol.Schedule (Seconds (1));
}
```

ANEXO IV

Códigos de Escenarios

- ANEXO IV-A: Escenario Fijo

```
* This is an example script for ad-hoc network with AODV routing prot.
* Authors: Lesly Maygua <lesye-20@hotmail.com> EPN, Octubre 2017

#include "ns3/aodv-module.h"
#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/internet-module.h"
#include "ns3/mobility-module.h"
#include "ns3/point-to-point-module.h"
#include "ns3/wifi-module.h"
#include "ns3/v4ping-helper.h"
#include <iostream>
#include "ns3/constant-velocity-helper.h"
#include "ns3/applications-module.h"
#include "ns3/netanim-module.h"

using namespace ns3;

/**
 * El script crea 5 nodos en 2D,con distancias diferentes y envia traf
UDP
 * [10.0.0.1]<50>[10.0.0.2]<100>[10.0.0.3]<120>[10.0.0.4]<100>[10.0.0.5]
 */
class AodvExample
{
public:
  AodvExample ();
  // Configura los parametros del script, retorna V si la conf es exitosa
  bool Configure (int argc, char **argv);
  // Corre la simulacion
  void Run ();
  // Muestra resultados
  void Report (std::ostream & os);

private:
  // Variable del numero de nodos
  uint32_t size;
  // Tiempo de simulacion [s]
  double totalTime;
  /// Imprime las rutas si es V
  bool printRoutes;

  // Elementos de red
  NodeContainer nodes;
  NetDeviceContainer devices;
  Ipv4InterfaceContainer interfaces;

private:
  // Funcion para crear los nodos
  void CreateNodes ();
 // Funcion para crear el dispositivo
  void CreateDevices ();
```

```
 // Funcion que Instala el stack de protocolos IPV4
  void InstallInternetStack ();
 // Funcion que instala aplicaciones
  void InstallApplications ();
};

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

  test.Run ();
  test.Report (std::cout);
  return 0;
}
//------------------------------------------------------------------------
------
AodvExample::AodvExample () :
  size (5),
  totalTime (20),
  printRoutes (true)
{
}

bool
AodvExample::Configure (int argc, char **argv)
{
  SeedManager::SetSeed (12345);
  CommandLine cmd;
  cmd.AddValue ("printRoutes", "Print routing table dumps.",
printRoutes);
  cmd.AddValue ("size", "Number of nodes.", size);
  cmd.AddValue ("time", "Simulation time, s.", totalTime);
  cmd.Parse (argc, argv);
  return true;
}

void
AodvExample::Run ()
{
//Llamado a todas las funciones que se va a ejecutar
  CreateNodes ();
  CreateDevices ();
  InstallInternetStack ();
  InstallApplications ();

  std::cout << "Iniciando simulacion para " << totalTime << " s ...\n";
  Simulator::Stop (Seconds (totalTime));
  Simulator::Run ();
  Simulator::Destroy ();
}

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

void
AodvExample::CreateNodes ()
```

```
{
  std::cout << "Creando " << (unsigned)size << " nodos\n";
  nodes.Create (size);
     // Nombre de nodos
  for (uint32_t i = 0; i < size; ++i)
    {
      std::ostringstream os;
      os << "node-" << i;
      Names::Add (os.str (), nodes.Get (i));
    }

//MOVILIDAD con NS3 para 5 nodos fijos
   MobilityHelper mobility;

//Posición inicial
Ptr<ListPositionAllocator>positionAlloc=CreateObject<ListPositionAllocato
r> ();
positionAlloc->Add (Vector (0.0, 0.0, 0.0));
positionAlloc->Add (Vector (50.0, 0.0, 0.0));
positionAlloc->Add (Vector (157.0, 0.0, 0.0));
positionAlloc->Add (Vector (270.0, 0.0, 0.0));
positionAlloc->Add (Vector (370.0, 0.0, 0.0));
mobility.SetPositionAllocator (positionAlloc);
 //Elección del modelo de movilidad
 mobility.SetMobilityModel ("ns3::ConstantPositionMobilityModel");
 //Instalacion de modelo en nodos
 mobility.Install (nodes);
}

  //CREANDO EL DISPOSITIVO
void
AodvExample::CreateDevices ()
{
  //Se define capa MAC con el Helper y el tipo de red
  WifiMacHelper wifiMac;
  wifiMac.SetType ("ns3::AdhocWifiMac");
  //Se define capa fisica PHY;por Default el modelo es NistErrorRateModel
  YansWifiPhyHelper wifiPhy = YansWifiPhyHelper::Default ();
  //Configuracion de canal,por defecto incluye los modelos:
//Default:LogDistancePropagationLossModel,ConstantSpeedPropagationDelayMo
del.
  YansWifiChannelHelper wifiChannel = YansWifiChannelHelper::Default ();
  //Coneccion del phy al channel
  wifiPhy.SetChannel (wifiChannel.Create ());
  //Se crea el dispositivo con el estadanr 802.11a
  WifiHelper wifi;
  wifi.SetRemoteStationManager ("ns3::ConstantRateWifiManager",
"DataMode", StringValue ("OfdmRate6Mbps"), "RtsCtsThreshold",
UintegerValue (0));
  //Todas las capas creadas se instalan en los nodos y asi se crea el
disposit.
  devices = wifi.Install (wifiPhy, wifiMac, nodes);
}

void
AodvExample::InstallInternetStack ()
{
  AodvHelper aodv;
  InternetStackHelper stack;
  stack.SetRoutingHelper (aodv);
```

```
  stack.Install (nodes);
  Ipv4AddressHelper address;
  address.SetBase ("10.0.0.0", "255.0.0.0");
  interfaces = address.Assign (devices);
  if (printRoutes)
    {
      Ptr<OutputStreamWrapper> routingStream =
Create<OutputStreamWrapper> ("aodv.routes", std::ios::out);
      aodv.PrintRoutingTableAllAt (Seconds (8), routingStream);
    }
}

void
AodvExample::InstallApplications ()
{
  /*TRAFICO:Se enviara trafico UDP a travez de una app cliente-servidor*/

   //Definicion del puerto del servidor
   UdpServerHelper Server (9);
   //La aplicacion del servidor se instala en el nodo 0
   ApplicationContainer serverApps = Server.Install (nodes.Get(0));
   //Tiempo de inicio y fin de simulacion del servidor
   serverApps.Start (Seconds (1.0));
   serverApps.Stop (Seconds (200));

  UdpClientHelper Client (interfaces.GetAddress (0), 9);
  Client.SetAttribute ("MaxPackets", UintegerValue (100));
  Client.SetAttribute ("Interval", TimeValue (Seconds (0.09)));
  Client.SetAttribute ("PacketSize", UintegerValue (1024));
  ApplicationContainer clientApps = Client.Install (nodes.Get(4));
  clientApps.Start (Seconds (2.0));
  clientApps.Stop (Seconds (200));
}
```

- ANEXO IV-B: Escenario Columna

```
/* -*- Mode:C++; c-file-style:"gnu"; indent-tabs-mode:nil; -*- */
/*
 * This is an example script for MANET with AODV routing protocol usinga
 * column mobility model
 * Authors: Lesly Maygua <lesye-20@hotmail.com> EPN, Octubre 2017
 */

#include "ns3/aodv-module.h"
#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/internet-module.h"
#include "ns3/mobility-module.h"
#include "ns3/point-to-point-module.h"
#include "ns3/wifi-module.h"
#include "ns3/v4ping-helper.h"
#include <iostream>
#include "ns3/constant-velocity-helper.h"
#include "ns3/applications-module.h"

using namespace ns3;
/**
```

```
 * Este script crea 5 nodos usando el modelo de movilidad Column de
bonnmotion
 */
class AodvExample
{
public:
  AodvExample ();
  /// Configure script parameters, \return true on successful
configuration
  bool Configure (int argc, char **argv);
  /// Run simulation
  void Run ();
  /// Report results
  void Report (std::ostream & os);

private:

  // Variable del numero de nodos
  uint32_t size;
  // Tiempo de simulacion [s]
  double totalTime;
  /// Imprime las rutas si es V
  bool printRoutes;

  // Elementos de red
  NodeContainer nodes;
  NetDeviceContainer devices;
  Ipv4InterfaceContainer interfaces;

private:
  // Funcion para crear los nodos
  void CreateNodes ();
 // Funcion para crear el dispositivo
  void CreateDevices ();
 // Funcion que Instala el stack de protocolos IPV4
  void InstallInternetStack ();
 // Funcion que instala aplicaciones
  void InstallApplications ();
};

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

  test.Run ();
  test.Report (std::cout);
  return 0;
}
//------------------------------------------------------------------------
------
AodvExample::AodvExample () :
  size (5),
  totalTime (20),
  printRoutes (true)
{
}

bool
```

```
AodvExample::Configure (int argc, char **argv)
{
  SeedManager::SetSeed (12345);
  CommandLine cmd;
  cmd.AddValue ("printRoutes", "Print routing table dumps.",
printRoutes);
  cmd.AddValue ("size", "Number of nodes.", size);
  cmd.AddValue ("time", "Simulation time, s.", totalTime);
  cmd.Parse (argc, argv);
  return true;
}

void
AodvExample::Run ()
{
  //Llamado a todas las funciones que se va a ejecutar
  CreateNodes ();
  CreateDevices ();
  InstallInternetStack ();
  InstallApplications ();

  std::cout << "Iniciando simulacion para " << totalTime << " s ...\n";
  Simulator::Stop (Seconds (totalTime));
  Simulator::Run ();
  Simulator::Destroy ();
}

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

void
AodvExample::CreateNodes ()
{
  std::cout << "Creando " << (unsigned)size << " nodos\n";
  nodes.Create (size);
     // Nombre de nodos
  for (uint32_t i = 0; i < size; ++i)
    {
      std::ostringstream os;
      os << "node-" << i;
      Names::Add (os.str (), nodes.Get (i));
    }

//MOVILIDAD CON BONNMOTION

//Tomando el archivo de traza generado en bonnmotion
std::string traceFile = "/home/ns325/ns3/ns-allinone-3.25/ns-
3.25/scratch/c2.ns_movements";
//Helper ns2 leyendo el archivo de traza
  Ns2MobilityHelper ns2 = Ns2MobilityHelper (traceFile);
  ns2.Install ();
}

  //CREANDO EL DISPOSITIVO
void
AodvExample::CreateDevices ()
{
  //Se define capa MAC con el Helper y el tipo de red
```

```
  WifiMacHelper wifiMac;
  wifiMac.SetType ("ns3::AdhocWifiMac");
  //Se define capa fisica PHY;por Default el modelo es NistErrorRateModel
  YansWifiPhyHelper wifiPhy = YansWifiPhyHelper::Default ();
  //Configuracion de canal,por defecto incluye los modelos:

//Default:LogDistancePropagationLossModel,ConstantSpeedPropagationDelayMo
del.
  YansWifiChannelHelper wifiChannel = YansWifiChannelHelper::Default ();
  //Coneccion del phy al channel
  wifiPhy.SetChannel (wifiChannel.Create ());
  //Se crea el dispositivo con el estadanr 802.11a
  WifiHelper wifi;
  wifi.SetRemoteStationManager ("ns3::ConstantRateWifiManager",
"DataMode", StringValue ("OfdmRate6Mbps"), "RtsCtsThreshold",
UintegerValue (0));
  //Todas las capas creadas se instalan en los nodos y asi se crea el
disposit.
  devices = wifi.Install (wifiPhy, wifiMac, nodes);
}

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

  if (printRoutes)
    {
      Ptr<OutputStreamWrapper> routingStream =
Create<OutputStreamWrapper> ("aodv.routes", std::ios::out);
      aodv.PrintRoutingTableAllAt (Seconds (8), routingStream);
    }
}

void
AodvExample::InstallApplications ()
{
  /*TRAFICO:Se enviara trafico UDP a travez de una app cliente-servidor*/
   //Configuracion del servidor:
   //Definicion del puerto del servidor
   UdpServerHelper Server (9);
   //La aplicacion del servidor se instala en el nodo 0
   ApplicationContainer serverApps = Server.Install (nodes.Get(0));
   //Tiempo de inicio y fin de simulacion del servidor
   serverApps.Start (Seconds (1.0));
   serverApps.Stop (Seconds (200));
  //Configuracion de cliente
  UdpClientHelper Client (interfaces.GetAddress (0), 9);
  Client.SetAttribute ("MaxPackets", UintegerValue (100));
  Client.SetAttribute ("Interval", TimeValue (Seconds (0.09)));
  Client.SetAttribute ("PacketSize", UintegerValue (1024));
  //La aplicacion del cliente se define en todos los nodos
  ApplicationContainer clientApps = Client.Install (nodes.Get(4));
```

```
  clientApps.Start (Seconds (2.0));
  clientApps.Stop (Seconds (200));
 }
```

- ANEXO IV-C: Escenario MANET

Este código es útil para escenarios ManhattanGrid y RandomWaypoint, debe variarse únicamente la traza de movilidad y la densidad de nodos.

```
  /* -*- Mode:C++; c-file-style:"gnu"; indent-tabs-mode:nil; -*- */
/*
 * This is an example script for MANET with AODV routing protocol using a
 * mobility model from bonnmotion
 *
 * Authors: Lesly Maygua <lesye-20@hotmail.com> EPN, Octubre 2017
 */

#include "ns3/aodv-module.h"
#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/internet-module.h"
#include "ns3/mobility-module.h"
#include "ns3/point-to-point-module.h"
#include "ns3/wifi-module.h"
#include "ns3/v4ping-helper.h"
#include <iostream>
#include "ns3/applications-module.h"

using namespace ns3;

class AodvExample
{
public:
  AodvExample ();
  // Configura los parametros del script, retorna V si la conf es exitosa
  bool Configure (int argc, char **argv);
  // Corre la simulacion
  void Run ();
  // Muestra resultados
  void Report (std::ostream & os);

private:
  // Variable del numero de nodos
  uint32_t size;
  // Tiempo de simulacion [s]
  double totalTime;
  /// Imprime las rutas si es V
  bool printRoutes;

  // Elementos de red
  NodeContainer nodes;
  NetDeviceContainer devices;
  Ipv4InterfaceContainer interfaces;

private:
 // Funcion para crear los nodos
  void CreateNodes ();
```

```
 // Funcion para crear el dispositivo
  void CreateDevices ();
 // Funcion que Instala el stack de protocolos IPV4
  void InstallInternetStack ();
 // Funcion que instala aplicaciones
  void InstallApplications ();
};

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

  test.Run ();
  test.Report (std::cout);
  return 0;
}
//-----------------------------------------------------------------------
------
AodvExample::AodvExample () :
  size ( ), //Número de nodos
  totalTime (20),
  printRoutes (true)
{
}
///Con el cmd es posible que --vis funcione sino no entonces tratarde ver
q hace esto de aca abajo
bool
AodvExample::Configure (int argc, char **argv)
{
  // Enable AODV logs by default. Comment this if too noisy
  // LogComponentEnable("AodvRoutingProtocol", LOG_LEVEL_ALL);

  SeedManager::SetSeed (12345);
  CommandLine cmd;
  cmd.AddValue ("printRoutes", "Print routing table dumps.",
printRoutes);
  cmd.AddValue ("size", "Numbero de nodos.", size);
  cmd.AddValue ("time", "Simulation time, s.", totalTime);
  cmd.Parse (argc, argv);
  return true;
}

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

  std::cout << "Iniciando simulacion por " << totalTime << " s ...\n";
  Simulator::Stop (Seconds (totalTime));
  Simulator::Run ();
  Simulator::Destroy ();
}
```

```
void
AodvExample::Report (std::ostream &)
{
}
// NODOS
void
AodvExample::CreateNodes ()
{
  //Mensaje para mostrar cuantos nodos se crean
  std::cout << "Creando " << (unsigned)size << " nodos\n";
  nodes.Create (size);
  //Nombre de los nodos
  for (uint32_t i = 0; i < size; ++i)
    {
      std::ostringstream os;
      os << "node-" << i;
      Names::Add (os.str (), nodes.Get (i));
    }
  //MOVILIDAD
//Tomando el archivo de traza generado en bonnmotion con ManhatanGrid
std::string traceFile = "/home/ns325/ns3/ns-allinone-3.25/ns-
3.25/scratch/traza.ns_movements";

//Helper ns2 leyendo el archivo de traza generado
  Ns2MobilityHelper ns2 = Ns2MobilityHelper (traceFile);
  ns2.Install ();
}
//CREANDO EL DISPOSITIVO
void
AodvExample::CreateDevices ()
{
  //Se define capa MAC con el Helper y el tipo de red
  WifiMacHelper wifiMac;
  wifiMac.SetType ("ns3::AdhocWifiMac");
  //Se define capa fisica PHY;por Default el modelo es NistErrorRateModel
  YansWifiPhyHelper wifiPhy = YansWifiPhyHelper::Default ();
  //Configuracion de canal,por defecto incluye los modelos:

//Default:LogDistancePropagationLossModel,ConstantSpeedPropagationDelayMo
del.
  YansWifiChannelHelper wifiChannel = YansWifiChannelHelper::Default ();
  //Coneccion del phy al channel
  wifiPhy.SetChannel (wifiChannel.Create ());
  //Se crea el dispositivo con el estadanr 802.11a
  WifiHelper wifi;
  wifi.SetRemoteStationManager ("ns3::ConstantRateWifiManager",
"DataMode", StringValue ("OfdmRate6Mbps"), "RtsCtsThreshold",
UintegerValue (0));
  //Todas las capas creadas se instalan en los nodos y asi se crea el
disposit.
  devices = wifi.Install (wifiPhy, wifiMac, nodes);
 }

void
AodvExample::InstallInternetStack ()
{
  //Se inicia la config de AODV
  AodvHelper aodv;
  //Activamos el stack de protocolos IPv4
  InternetStackHelper stack;
```

```
  //Se activa enrutamiento con AODV
  stack.SetRoutingHelper (aodv);
  //Se instala el stack en los nodos
  stack.Install (nodes);
  //Se activa direccionamiento IPv4 con el ayudante
  Ipv4AddressHelper address;
  //Se determina la direccion IP inicial y la MSK
  address.SetBase ("10.0.0.0", "255.0.0.0");
  interfaces = address.Assign (devices);

  if (printRoutes)
    {
      Ptr<OutputStreamWrapper> routingStream =
Create<OutputStreamWrapper> ("aodv.routes", std::ios::out);
      aodv.PrintRoutingTableAllAt (Seconds (8), routingStream);
    }
}

//APLICACIONES
void
AodvExample::InstallApplications ()
{
  /*TRAFICO:Se enviara trafico UDP a travez de una app cliente-servidor*/

   //Configuracion del servidor:
   //Definicion del puerto del servidor
   UdpServerHelper Server (9);
   //La aplicacion del servidor se instala en el nodo 100
   ApplicationContainer serverApps = Server.Install (nodes.Get(99));
   //Tiempo de inicio y fin de simulacion del servidor
   serverApps.Start (Seconds (1.0));
   serverApps.Stop (Seconds (200));

  for (uint32_t i = 0; i<size-1; ++i)
   {
  //Configuracion del cliente
  //El cliente hara peticiones al puerto 9 de la ip del nodo servidor
  UdpClientHelper Client (interfaces.GetAddress (99), 9);
  Client.SetAttribute ("MaxPackets", UintegerValue (100));
  Client.SetAttribute ("Interval", TimeValue (Seconds (0.09)));
  Client.SetAttribute ("PacketSize", UintegerValue (1024));
  //La aplicacion del cliente se define en todos los nodos
  ApplicationContainer clientApps = Client.Install (nodes.Get(i));
  clientApps.Start (Seconds (2.0));
  clientApps.Stop (Seconds (200));
 }
}
```

ANEXO V

Código para la generación de LOGS

El código para la generación de los dos archivos se agrega en diferentes funciones del *aodv-routing-protocol* según se requiera la información.

- ANEXO V-A: Archivo 1 – Packet Trace

```
//Fichero Packet Trace

std::ofstream ofs;
ofs.open ("PacketT.txt",std::ofstream::out |std::ofstream::app);
//Impresion, tiempo de simulacion actual
ofs<< Now().As (Time::S)<< " " ;
//Impresion, ID paquete
ofs<< p->GetUid () <<" ";
//Nodo origen
ofs<< header.GetSource () <<" ";
//Nodo destino
ofs<< header.GetDestination () <<" ";
//Nodo actual
ofs<< m_ipv4->GetObject<Node> ()->GetId () <<" ";
//Numero de saltos al destino
ofs<< toDst.GetHop ()<<" ";
//Tamaño del paquete
ofs<< p->GetSize () << " ";
//Siguiente salto
ofs<< toDst.GetNextHop () <<" ";
ofs.close();

//En la funcion RouteInput se agrega las iniciales T, T1,y R para
//identificar paquetes transmitidos y recibidos
RoutingProtocol::RouteInput
ofs<< "T1" <<" ";
ofs<< "T"  <<" ";
ofs<< "R"  <<" ";
//En la funcion Forwarding, se agrega al fichero las iniciales F, DR
//para identificar paquetes reenviados y desechados respectivamente
RoutingProtocol::Forwarding
ofs<< "F" <<" ";
ofs<< "DR"<<" ";
```

- ANEXO V-B: Archivo 2 – Power Level

```
//Fichero Power Level
std::ofstream ofs;
ofs.open ("Power_Level.txt", std::ofstream::out | std::ofstream::app);
//Impresion, tiempo de simulacion actual
ofs<< Now().As (Time::S)<<" ";
//Impresion ID nodo
ofs<< m_ipv4->GetObject<Node> ()->GetId () <<" ";
//Numero de Vecinos actuales
ofs<< num_n<<" ";
```

```
//Numero de Vecinos anteriores
ofs<< vecinos_anterior<<" ";
//P.actual "P[dBm]"
ofs<< phyles->GetTxPowerStart ()<<" " ;
//P.anterior "P[dBm]"
ofs<< pot_act_d<< "\n" ;
//Actualizacion de vecinos
vecinos_anterior=num_n;
//Se cierra el archivo
ofs.close();
```

ANEXO VI

Filtros en GAWK

- ANEXO VI-A: Filtro de paquetes

```
#Este script genera un filtro para identificar la cantidad de paquetes
#transmitidos, recibidos, desechados y reenviados en una transmision
#Lesly Maygua Octubre, 2017
#T1 - 2.0s - 209 - 102.102.102.102 - 10.0.0.100 - 7 - 1024

BEGIN{
    salida1="resultados.txt"       #Archivo de salida

       ORS="\n"
        ctx=0
        cf=0
        cr=0
        cDR=0
        th=0    }
{

    paquete=$1              #Tipo de paquete
    id=$5                   #id paquete
    nactual=$11             #Id del nodo actual
    source=$7               #Origen del paquete
    dst=$9                  #Destino
    size=$13                #Tamaño del paquete
    time=$3                 #tiempo del evento

#Diferenciacion de paquetes

if (paquete = = "T" || paquete = = "T1")
        {
            ctx++
        }
if(paquete = = "F")
        {
            cf++
        }
if(paquete = = "R")
        {
           cr++
        }
if(paquete = = "DR")
        {
            cDR++
        }
  th=(cr/ctx)*100
  tf=(cf/ctx)*100
  td=(cDR/ctx)*100
}
END{
    printf("\nRESULTADOS - DIFERENCIANCION DE PAQUETES\n") >> salida1
    printf("\nPaquetes totales TX:\t%i",ctx) >> salida1
    printf("\nPaquetes totales RX:\t%i",cr) >> salida1
    printf("\nPaquetes reenviados:\t%i",cf) >> salida1
```

```
    printf("\nPaquetes desechados:\t%i",cDR) >> salida1
    printf("\n\nThroughput     [%] :\t%.3f",th) >> salida1
    printf("\nTasa de reenvio[%] :\t%.3f",tf) >> salida1
    printf("\nTasa de drop   [%] :\t%.3f",td) >> salida1

close(salida1)
}
```

- ANEXO VI-B: Filtro de potencia

```
#Este script genera un filtro para identificar la potencia promedio de
#transmision de un nodo en especifico
#Lesly Maygua Octubre, 2017
#T - ID nodo - V.act - V.ant - P.act - P. ant

BEGIN{
    salida1="power_variation.txt"       #Archivo de salida

       ORS="\n"
        c=0
        p=0
        pm=0
       }

{
    tiempo=$1               #Tiempo del evento
    id=$3                   #Id del nodo actual 3
    vactual=$5              #vecinos actuales 5
    vant=$7                 #vecinos anteriores7
    pact=$9                 #potencia actual 9
    pant=$11                #potencia anterior 11

#power
if(id = = "5")
        {
            pact_mw=10^(pact/10)
            p=p+pact_mw
            c++
            pm=(p/c)
        }
  pm_d=10*(log(pm)/log(10))  #Pot. promedio dBm
}

END{
    printf("\nPOTENCIA DE TRANSMISION PROMEDIO\n") >> salida1
    printf("\nPot. Media [ mw]:\t%.3f",pm) >> salida1
    printf("\nPot. Media [dBm]:\t%.3f",pm_d) >> salida1

close(salida1)
}
```

ANEXO VII

Resultados de todas las pruebas realizadas

Las filas resaltadas en color amarillo hacen referencia a las pruebas en la cuales el uso del mecanismo de control de potencia fue muy favorable.

- ANEXO VII-A: Prueba 1 - 25 nodos

P.N°	PAQ. TX	TRAZA	SIN MECANISMO				CON MECANISMO			
			RX	DR	FW	TH	RX	DR	FW	TH
1	100	m25a	402	0	102	16,75%	784	23	1419	32,67%
2	100	m25b	187	0	55	7,79%	510	50	685	21,25%
3	100	m25c	520	276	1647	21,67%	572	72	1151	23,83%
4	100	m25d	520	0	354	21,67%	581	3	396	24,21%
5	100	m25e	5	0	4	0,21%	195	0	93	8,13%
6	100	m25f	317	7	52	13,21%	406	0	191	16,92%
7	100	m25g	220	10	287	9,17%	365	0	465	15,21%
8	100	m25h	300	0	203	12,50%	505	6	566	21,04%
9	100	m25i	1056	0	1698	44,00%	1108	0	1811	46,17%
10	100	m25j	678	0	1010	28,25%	924	155	2419	38,50%
PROMEDIO			**420,50**	**29,30**	**541,20**	**17,52%**	**595,00**	**30,90**	**919,60**	**24,79%**

- ANEXO VII-B: Prueba 1 - 36 nodos

P.N°	PAQ. TX	TRAZA	SIN MECANISMO				CON MECANISMO			
			RX	DR	FW	TH	RX	DR	FW	TH
1	100	m36a	1084	209	3394	30,97%	1072	510	3431	30,63%
2	100	m36b	837	106	2100	23,91%	978	8	2646	27,94%
3	100	m36c	551	0	523	15,74%	525	1	608	15,00%
4	100	m36d	0	175	1110	0,00%	156	243	1084	4,46%
5	100	m36e	1173	205	4696	33,51%	1132	167	2716	32,34%
6	100	m36f	680	424	9567	19,43%	1085	591	5607	31,00%
7	100	m36g	200	0	133	5,71%	287	1	244	8,20%
8	100	m36h	946	51	3068	27,03%	1025	24	3101	29,29%
9	100	m36i	466	443	3431	13,31%	673	164	3814	19,23%
10	100	m36j	3	0	0	0,09%	168	0	70	4,80%
PROMEDIO			**594**	**161,3**	**2802,2**	**16,97%**	**710,1**	**170,9**	**2332,1**	**20,29%**

- ANEXO VII-C: Prueba 1 - 100 nodos

P.N	TRAZA	SIN MECANISMO				CON MECANISMO			
		RX	DR	FW	TH	RX	DR	FW	TH
1	m100a	1424	629	14448	14,38%	1291	833	13403	13,04%
2	m100b	782	872	12220	7,90%	747	927	12781	7,55%
3	m100c	1639	435	11934	16,56%	1498	482	12061	15,13%
4	m100d	387	863	18407	3,91%	212	1271	13259	2,14%
5	m100e	1330	562	12265	13,43%	1564	436	12344	15,80%
6	m100f	1212	939	12889	12,24%	1012	831	13436	10,22%
7	m100g	1493	405	12454	15,08%	1175	1147	12238	11,87%
8	m100h	2001	583	12682	20,21%	1782	497	11700	18,00%
9	m100i	2001	361	11056	20,21%	1828	757	13623	18,46%
10	m100j	1122	188	14254	11,33%	1013	1102	13601	10,23%
PROMEDIO		**1.339,10**	**583,7**	**13.260,90**	**13,53%**	**1.212,20**	**828,3**	**12.844,60**	**12,24%**

- ANEXO VII-D: Prueba 2 - 25 nodos

P.N°	PAQ. TX	TRAZA	SIN MECANISMO				CON MECANISMO			
			RX	DR	FW	TH	RX	DR	FW	TH
1	100	r25a	501	0	404	20,88%	556	0	651	23,17%
2	100	r25b	400	0	6	16,67%	436	1	115	18,17%
3	100	r25c	1124	28	4076	46,83%	958	525	5108	39,92%
4	100	r25d	289	7	326	12,04%	353	9	304	14,71%
5	100	r25e	872	0	1860	36,33%	736	1	1887	30,67%
6	100	r25f	335	6	144	13,96%	365	0	95	15,21%
7	100	r25g	290	0	292	12,08%	373	9	214	15,54%
8	100	r25h	100	0	333	4,17%	331	21	831	13,79%
9	100	r25i	0	22	17	0,00%	286	5	283	11,92%
10	100	r25j	102	14	90	4,25%	383	3	377	15,96%
PROMEDIO			**401,30**	**7,70**	**754,80**	**16,72%**	**477,70**	**57,40**	**986,50**	**19,90%**

- ANEXO VII-E: Prueba 2 - 36 nodos

P.N°	PAQ. TX	TRAZA	SIN MECANISMO				CON MECANISMO			
			RX	DR	FW	TH	RX	DR	FW	TH
1	100	r36a	1513	46	7066	43,23%	1722	328	6163	49,20%
2	100	r36b	1429	694	7524	40,83%	1089	490	6714	31,11%
3	100	r36c	1503	68	3763	42,94%	1456	416	4085	41,60%
4	100	r36d	1172	12	1229	33,49%	1197	91	2385	34,20%
5	100	r36e	1260	120	10405	36,00%	1206	128	1334	34,46%

6	100	r36f	105	1130	6059	3,00%	153	797	8831	4,37%
7	100	r36g	513	539	1340	14,66%	1138	43	1819	32,51%
8	100	r36h	1329	231	6547	37,97%	1259	538	5962	35,97%
9	100	r36i	444	16	4400	12,69%	1154	7	3893	32,97%
10	100	r36j	66	201	5351	1,89%	285	267	3013	8,14%
PROMEDIO			**933,4**	**305,7**	**5368,4**	**26,67%**	**1065,9**	**310,5**	**4419,9**	**30,45%**

- ANEXO VII-F: Prueba 2 - 100 nodos

P.N°	TRAZA	SIN MECANISMO				CON MECANISMO			
		RX	DR	FW	TH	RX	DR	FW	TH
1	r100a	1262	579	11497	12,75%	1111	704	11696	11,22%
2	r100b	1658	649	10758	16,75%	1207	522	12278	12,19%
3	r100c	1804	258	12216	18,22%	1438	980	11800	14,53%
4	r100d	2115	516	10460	21,36%	1758	727	10388	17,76%
5	r100e	1724	282	11848	17,41%	1745	755	12013	17,63%
6	r100f	1076	146	12807	10,87%	1202	379	12461	12,14%
7	r100g	1109	510	16686	11,20%	706	476	9980	7,13%
PROMEDIO		**1.535,4**	**420**	**12324,57**	**15,51%**	**1.309,6**	**649**	**11516,57**	**13,23%**

- ANEXO VII-G: Prueba 3 – 5 km/h

P.N°	PAQ. TX	TRAZA	SIN MECANISMO				CON MECANISMO			
			RX	DR	FW	TH	RX	DR	FW	TH
1	100	m36a2	1084	209	3394	30,97%	1072	510	3431	30,63%
2	100	m36b2	837	106	2100	23,91%	978	8	2646	27,94%
3	100	m36c2	551	0	523	15,74%	525	1	608	15,00%
4	100	m36d2	0	175	1110	0,00%	156	243	1084	4,46%
5	100	m36e2	1173	205	4696	33,51%	1132	167	2716	32,34%
6	100	m36f2	680	424	9567	19,43%	1085	591	5607	31,00%
7	100	m36g2	200	0	133	5,71%	287	1	244	8,20%
8	100	m36h2	946	51	3068	27,03%	1025	24	3101	29,29%
9	100	m36i2	466	443	3431	13,31%	673	164	3814	19,23%
10	100	m36j2	3	0	0	0,09%	168	0	70	4,80%
PROMEDIO			**594**	**161,3**	**2802,2**	**16,97%**	**710,1**	**170,9**	**2332,1**	**20,29%**

- ANEXO VII-H: Prueba 3 – 6 km/h

P.N°	PAQ. TX	TRAZA	SIN MECANISMO				CON MECANISMO			
			RX	DR	FW	TH	RX	DR	FW	TH
1	100	m6a	390	0	190	11,14%	389	0	197	11,11%
2	100	m6b	427	564	5588	12,20%	599	810	7951	17,11%
3	100	m6c	290	20	277	8,29%	352	7	268	10,06%
4	100	m6d	1195	308	6379	34,14%	1274	408	6563	36,40%
5	100	m6e	822	0	1168	23,49%	822	1	1055	23,49%
6	100	m6f	278	0	57	7,94%	298	0	309	8,51%
7	100	m6g	846	3	591	24,17%	829	11	1026	23,69%
8	100	m6h	791	526	4280	22,60%	790	233	2415	22,57%
9	100	m6i	69	201	250	1,97%	293	137	1814	8,37%
10	100	m6j	927	80	1248	26,49%	1023	13	2094	29,23%
PROMEDIO			**603,5**	**170,2**	**2002,8**	**17,24%**	**666,9**	**162**	**2369,2**	**19,05%**

- ANEXO VII-I: Prueba 3 – 10 km/h

P.N°	PAQ. TX	TRAZA	SIN MECANISMO				CON MECANISMO			
			RX	DR	FW	TH	RX	DR	FW	TH
1	100	m10a	759	0	721	21,69%	759	0	620	21,69%
2	100	m10b	300	0	0	8,57%	305	3	30	8,71%
3	100	m10c	773	387	3853	22,09%	916	236	3846	26,17%
4	100	m10d	1258	123	5580	35,94%	1082	216	5350	30,91%
5	100	m10f	200	0	462	5,71%	282	16	193	8,06%
6	100	m10e	334	26	184	9,54%	430	25	322	12,29%
7	100	m10g	200	0	370	5,71%	200	0	370	5,71%
8	100	m10h	84	6	168	2,40%	141	3	44	4,03%
9	100	m10i	1065	103	1788	30,43%	960	10	1066	27,43%
10	100	m10j	825	535	4451	23,57%	960	10	1066	27,43%
PROMEDIO			**503,9**	**118**	**1685,6**	**14,40%**	**527,6**	**51,9**	**1228,7**	**15,07%**

- ANEXO VII-J: Prueba 3 – 20 km/h

P.N°	PAQ. TX	TRAZA	SIN MECANISMO				CON MECANISMO			
			RX	DR	FW	TH	RX	DR	FW	TH
1	100	m20a	731	197	1700	20,89%	611	81	1512	17,46%
2	100	m20b	115	0	39	3,29%	117	0	65	3,34%
3	100	m20c	41	0	4	1,17%	136	528	3684	3,89%
4	100	m20d	1522	73	2248	43,49%	1384	177	2778	39,54%

5	100	m20e	114	3	98	3,26%	134	6	264	3,83%
6	100	m20f	120	0	2617	3,43%	178	289	4255	5,09%
7	100	m20g	202	0	2	5,77%	212	0	37	6,06%
8	100	m20h	1423	48	3023	40,66%	1461	237	3827	41,74%
9	100	m20i	1257	3	1679	35,91%	1152	107	2094	32,91%
10	100	m20j	100	0	0	2,86%	105	6	244	3,00%
PROMEDIO			**562,5**	**32,4**	**1141**	**16,07%**	**549**	**143,1**	**1876**	**15,69%**

- ANEXO VII-K: Prueba 4 – 5 km/h

P.N°	PAQ. TX	TRAZA	SIN MECANISMO				CON MECANISMO			
			RX	DR	FW	TH	RX	DR	FW	TH
1	100	r5a	1513	46	7066	43,23%	1722	328	6163	49,20%
2	100	r5b	1429	694	7524	40,83%	1089	490	6714	31,11%
3	100	r5c	1503	68	3763	42,94%	1456	416	4085	41,60%
4	100	r5d	1172	12	1229	33,49%	1197	91	2385	34,20%
5	100	r5e	1260	120	10405	36,00%	1206	128	1334	34,46%
6	100	r5f	105	1130	6059	3,00%	153	797	8831	4,37%
7	100	r5g	513	539	1340	14,66%	1138	43	1819	32,51%
8	100	r5h	1329	231	6547	37,97%	1259	538	5962	35,97%
9	100	r5i	444	16	4400	12,69%	1154	7	3893	32,97%
10	100	r5j	66	201	5351	1,89%	285	267	3013	8,14%
PROMEDIO			**933,4**	**305,7**	**5368,4**	**26,67%**	**1065,9**	**310,5**	**4419,9**	**30,45%**

- ANEXO VII-L: Prueba 4 – 6 km/h

P.N°	PAQ. TX	TRAZA	SIN MECANISMO				CON MECANISMO			
			RX	DR	FW	TH	RX	DR	FW	TH
1	100	r6a	1868	29	5225	53,37%	1692	32	6258	48,34%
2	100	r6b	474	396	8158	13,54%	1265	74	5595	36,14%
3	100	r6c	1467	429	5492	41,91%	1469	267	5319	41,97%
4	100	r6d	1223	183	7046	34,94%	935	924	5405	26,71%
5	100	r6e	615	191	5658	17,57%	672	248	5404	19,20%
6	100	r6f	1331	200	5121	38,03%	881	963	4564	25,17%
7	100	r6g	1052	130	4367	30,06%	832	80	5549	23,77%
8	100	r6h	1440	208	4199	41,14%	1157	132	4122	33,06%
9	100	r6i	1469	73	5529	41,97%	1266	222	5498	36,17%
10	100	r6j	479	0	558	13,69%	472	0	545	13,49%
PROMEDIO			**1141,8**	**183,9**	**5135,3**	**32,62%**	**1064,1**	**294,2**	**4825,9**	**30,40%**

- ANEXO VII-M: Prueba 4 – 10 km/h

P.N°	PAQ. TX	TRAZA	SIN MECANISMO				CON MECANISMO			
			RX	DR	FW	TH	RX	DR	FW	TH
1	100	r10a	273	73	6018	7,80%	173	234	4679	4,94%
2	100	r10b	0	0	0	0,00%	0	0	0	0,00%
3	100	r10c	1033	67	4946	29,51%	781	80	5672	22,31%
4	100	r10d	155	1	43	4,43%	304	327	1137	8,69%
5	100	r10e	666	572	15193	19,03%	519	459	1450	14,83%
6	100	r10f	847	231	8137	24,20%	901	223	7981	25,74%
7	100	r10g	300	4	6970	8,57%	1061	509	6244	30,31%
8	100	r10h	100	0	0	2,86%	100	0	0	2,86%
9	100	r10i	1640	215	3555	46,86%	1556	128	4520	44,46%
10	100	r10j	609	504	9684	17,40%	478	493	4507	13,66%
PROMEDIO			**562,3**	**166,7**	**5454,6**	**16,07%**	**587,3**	**245,3**	**3619**	**16,78%**

- ANEXO VII-N: Prueba 4 – 20 km/h

P.N°	PAQ. TX	TRAZA	SIN MECANISMO				CON MECANISMO			
			RX	DR	FW	TH	RX	DR	FW	TH
1	100	r20a	1522	360	7208	43,49%	1324	63	6559	37,83%
2	100	r20b	6	1027	6654	0,17%	462	712	2952	13,20%
3	100	r20c	344	173	6862	9,83%	149	227	6624	4,26%
4	100	r20d	651	434	5977	18,60%	439	1101	5158	12,54%
5	100	r20e	375	477	4650	10,71%	844	515	4960	24,11%
6	100	r20f	1257	676	6398	35,91%	993	434	6095	28,37%
7	100	r20g	1307	623	4809	37,34%	1274	778	4887	36,40%
8	100	r20h	530	402	5806	15,14%	873	278	6971	24,94%
9	100	r20i	1056	399	9241	30,17%	331	70	4501	9,46%
10	100	r20j	100	119	6532	2,86%	805	214	5905	23,00%
PROMEDIO			**714,8**	**469**	**6413,7**	**20,42%**	**749,4**	**439,2**	**5461,2**	**21,41%**

ORDEN DE EMPASTADO