PONTIFICIA UNIVERSIDAD CATÓLICA DEL ECUADOR

FACULTAD DE INGENIERÍA

TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DE TÍTULO DE MAGISTER EN REDES DE COMUNICACIONES

ANÁLISIS Y SIMULACIÓN DE UNA RED MÓVIL AD HOC UTILIZANDO NS2, PARA DETERMINAR EL COMPORTAMIENTO DE LAS VARIANTES DEL PROTOCOLO TCP EN RELACIÓN AL NÚMERO DE NODOS EN LA RED

CARLOS DARWIN AGUILAR MORA

DIRECTOR: MGTR. DAVID RAMIREZ

QUITO, SEPTIEMBRE 2017

Declaración

Yo, CARLOS DARWIN AGUILA MORA, declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentado 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 los derechos de propiedad intelectual correspondiente a este trabajo, a la Pontificia Universidad Católica del Ecuador, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normativa institucional vigente.

……..

CARLOS DARWIN AGUILAR MORA

Certificación

Certifico que el presente trabajo fue desarrollado por Carlos Darwin Aguilar Mora bajo mi supervisión.

……..

Ing. David Ramírez Mgs.

DIRECTOR DEL PROYECTO

Dedicatoria

A Yessy, mi amada esposa por su apoyo incondicional, a mis dos Daygoritos, Isaac y Joaquín. Los tres son el motor de mi vida.

Agradecimiento

En primer lugar, a Dios Todopoderoso por darme la vida y por permitirme llegar a estas instancias de mi vida profesional. A la Virgencita de El Cisne protectora de mi familia y a San Judas Tadeo, abogado de las causas imposibles.

De manera especial al Ing. David Ramirez por aceptar la dirección del presente proyecto y por su importante apoyo en la consecución de los objetivos planteados.

A mis docentes de la Maestría de Redes de Comunicaciones por compartir ampliamente sus conocimientos y experiencias.

A Silvio (+), mi padre, por sus sabios consejos que nunca olvidaré.

A mi mami Mercedes y a mis hermanas Silvia y Fernanda por su apoyo incondicional.

A mis tíos Jorge Patricio y María del Cisne por ser parte fundamental para alcanzar este sueño.

A mis amigos y compañeros de trabajo por el apoyo brindado, especialmente a Rommel.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

CAPÍTULO 1

1 INTRODUCCIÓN

1.1 **Introducción.**

Las redes móviles Ad Hoc son redes auto configurables en la cual los nodos no necesitan una estación base para comunicarse entre sí. En este tipo de redes los nodos se pueden cambiar su ubicación geográfica a discreción o dependiendo del escenario en el cual se desenvuelven. Las redes móviles Ad Hoc son adecuadas en entornos en los cuales es necesaria una comunicación que se configure de forma rápida, como por ejemplo escenarios de emergencia y rescate.

TCP ha sido ampliamente estudiado en redes donde los nodos por lo general permanecen fijos. No así en redes móviles Ad Hoc donde la movilidad de los nodos ha desencadenado investigaciones en la cual los protocolos convencionales han sido modificados o nuevos protocolos han sido propuestos. En la misma forma TCP se ve afectado por la movilidad presente en este tipo de redes.

El presente trabajo analizará el comportamiento de las variantes de TCP para control de congestión sobre redes móviles Ad Hoc. A través de simulación se obtendrá indicadores tales como RTT (Round Time Trip), jitter, retardo, y otros que nos permitirán concluir cuál de los mecanismos estudiados presenta una mejor adaptación a las redes móviles Ad Hoc.

La etapa de simulación de redes móviles Ad Hoc contemplará varios escenarios de prueba cada uno con diferente cantidad de nodos, de tal forma que faciliten efectuar las mediciones deseadas, las pruebas consistirán en generar conexiones TCP en los escenarios propuestos, mediante el simulador de redes NS2 sobre una plataforma Linux. NS2 es un simulador de eventos discretos muy utilizado para la simulación de redes, utilizado también para redes móviles Ah Hoc.

Al final del proyecto el resultado de las mediciones permitirá interpretar el comportamiento de TCP en lo relacionado a sus mecanismos de control de congestión, en base a las pruebas de simulación efectuadas.

La primera parte de este anteproyecto de tesis de Maestría en Redes de Comunicaciones muestra el contenido a ser tratado y desarrollado, a continuación, se realiza la justificación de realizar el proyecto, se presenta el objetivo principal y los objetivos específicos, un índice tentativo y finalmente se propone un cronograma de desarrollo del proyecto de tesis.

1.2 **Antecedentes.**

En la última década el campo de las redes inalámbricas ha experimentado un crecimiento exponencial, han existido grandes avances en cuanto a infraestructura de red, disponibilidad de aplicaciones móviles y la aparición de dispositivos inalámbricos como computadores portátiles, PDAs, teléfonos celulares, todos cada vez con mejores capacidades tanto en hardware como en software. Estos dispositivos están jugando un papel cada vez más importante en la vida de los usuarios, tal es así

que los usuarios pueden utilizar sus teléfonos celulares para navegar por internet y acceder a su correo electrónico, los viajeros con sus computadores portátiles puede navegar por internet desde los aeropuertos, estaciones de tren, cafés y otros lugares públicos; los turistas pueden utilizar terminales GPS en sus vehículos para acceder a mapas y localizar lugares de interés turístico; adicionalmente archivos u otra información se puede compartir mediante la conexión de computadores portátiles a través de redes LAN inalámbricas mientras se atiende una conferencia por ejemplo; en el hogar se pueden sincronizar los datos y transferir archivos entre los dispositivos portátiles y de escritorio (Liu y Chlamtac, 2004).

En general, las redes móviles Ad Hoc están formadas dinámicamente por un sistema autónomo de nodos móviles que están conectados mediante enlaces inalámbricos sin hacer uso de la infraestructura de red existente; los nodos son libres de moverse aleatoriamente y organizarse arbitrariamente, por lo tanto, la topología de red inalámbrica pude cambiar rápidamente y de manera impredecible (Macker y Corson, 2002).

Las redes Ah Hoc son adecuadas para su uso en situaciones en las que una infraestructura de red fija no es factible o rentable. Uno de los muchos usos posibles de estas redes es en entornos empresariales donde la colaboración podría ser más importante fuera del entorno de la oficina que en el interior, también pueden ser utilizadas para proveer aplicaciones de gestión de servicios críticos, como en la recuperación de desastres, donde toda la infraestructura de red ha sido destruida y el acceso a la comunicación es crucial. Mediante el uso de una red móvil Ad Hoc levantar una red podría tomar horas en lugar de semanas, como se requiere en el caso

de una red de infraestructura fija. Otro ejemplo de aplicación es el bluetooth que está diseñado para soportar redes de área personal (PAN), al eliminar la necesidad de cables entre los dispositivos. El famoso protocolo IEEE 802.11 o WiFi también es compatible con un sistema de red Ad Hoc en ausencia de un punto de acceso inalámbrico (Sarkar, Basavaraju y Puttamadappa, 2008).

Otras aplicaciones de las redes móviles Ad Hoc:

- Redes comunitarias
- Redes empresariales
- Redes domésticas
- Redes de respuesta a emergencias
- Redes vehiculares
- Redes de sensores

La arquitectura Ad Hoc brinda muchos beneficios tales como la autoconfiguración y la adaptabilidad a las características móviles altamente variables como la energía, las condiciones de transmisión, distribución de tráfico y balance de carga. Estos beneficios plantean nuevos retos relacionados principalmente a la imprevisibilidad de la red debido a la movilidad de los nodos (Sarkar, Basavaraju y Puttamadappa, 2008).

Haciendo un enfoque al presente trabajo hay que considerar que TCP fue diseñado originalmente para trabajar en redes fijas. TCP proporciona un protocolo de control de transporte eficaz orientado a la conexión que proporciona a su vez control de flujo y control de congestión requeridos para asegurar la entrega de paquetes fiable.

Debido a que las tasas de error de red por cable son bastante bajas, TCP utiliza la pérdida de paquetes como un indicador de congestión de la red, y se ocupa de manera eficaz al hacer el ajuste correspondiente a su ventana de congestión. En una red inalámbrica móvil ad hoc, las pérdidas de paquetes no son causadas únicamente por congestión de la red, sino por la alta tasa de error de medianas y frecuentes desconexiones inalámbricas de movilidad, lo que resulta en el uso de mecanismos de control que se invocan de manera inapropiada, reduciendo así la utilización del ancho de banda de la red e incrementando el tiempo para la restauración de la conexión. Además, la variación en la capacidad de enlace podría causar enlaces asimétricos y retardo en la entrega de acuses de recibo, que pueden afectar el ajuste de ventana de congestión. Como resultado de esto, los mecanismos de control de flujo y de control de congestión estándar podrían no funcionar adecuadamente en redes móviles ad hoc, afectando el rendimiento de TCP (Liu y Chlamtac 2004),

Referente al control de congestión, TCP utiliza varios mecanismos que contribuyen a dicho control, los mismos que están estandarizados en el documento RFC 2581 (Allman, Paxon, Stevens, 1999), en el cual se contemplan: arranque lento (slow start), evitación de la congestión (congestion avoidance) y recuperación rápida (fast recovery). Los dos primeros son obligatorios en TCP, diferenciándose en la forma en que aumentan el tamaño de la ventana de congestión en respuesta a los paquetes ACK (acknowledgement) recibidos, mientras que el tercero, recuperación rápida es recomendable, aunque no obligatorio (Kurose y Ross 2010). Estos mecanismos operan adecuadamente en redes cableadas, no así en redes inalámbricas cuyas características difieren significativamente con respecto a las redes cableadas, estas

características que afectan el rendimiento de TCP en entornos inalámbricos son (Ashish, Jagannadha, Mansoor y Vijay, 2001):

- Ancho de banda limitado: en redes cableadas comúnmente se dispone de anchos de banda de 100 Mbps, llegando al orden de los Gbps con el uso de medios ópticos; comparado con redes inalámbricas el ancho de banda es mucho menor, por ejemplo: citando el estándar IEEE 802.11b cuyo ancho de banda alcanza los 11 Mbps. Por lo tanto, el ancho de banda disponible es uno de los principales cuellos de botella que degrada el rendimiento de TCP en medios inalámbricos.
- RTTs (tiempos de ida y vuelta) grandes: en general, los RTT en redes inalámbricas son mayores que en redes cableadas, lo que ocasiona que la ventana de congestión aumente a un ritmo mucho menor en el caso de las redes inalámbricas, lo que limita el rendimiento de TCP en estas redes.
- Pérdidas aleatorias: las pérdidas de transmisión en redes inalámbricas son significativamente mayores que en redes cableadas, lo que resulta en retransmisiones excesivas y disminución de la ventana de congestión.
- Movilidad de los usuarios: en redes móviles ad hoc los nodos pueden moverse aleatoriamente provocando cambios frecuentes en la topología, lo cual conlleva a pérdida de paquetes y forza a los nodos a iniciar frecuentemente algoritmos de descubrimiento de ruta, lo que resulta en reducción significativa de rendimiento.
- Flujos cortos: cuando se transmiten pequeñas cantidades de datos, existe alta probabilidad de que la transferencia se complete antes de que el tamaño de ventana del emisor alcance su tamaño máximo, lo que se traduce en subutilización de la capacidad de la red

- Consumo de energía: el consumo de energía es un factor muy importante en el caso de dispositivos que operan con baterías como laptops, PDAs y teléfonos móviles.

Dadas estas características, varios esquemas de TCP se han propuesto para abordar el problema de rendimiento de TCP referente a congestión en redes inalámbricas, algunos esquemas planteados incluyen TCP New Reno, TCP Vegas, TCP Westwood, entre otros (Law, Krishnamurthy y Faloutsos, 2009).

Estos esquemas también conocidos como variantes de TCP se describen a continuación:

- TCP New Reno: esta variante se refiere al procedimiento de recuperación rápida (fast recovery) que comienza cuando se reciben tres ACKs duplicados, y termina ya sea cuando se presenta un timeout en la retransmisión o cuando un ACK confirma la llegada de todos los datos incluyendo los datos que estaban pendientes cuando se inició el proceso de recuperación rápida. Difiere de la implementación original de TCP por la inclusión de la variable "recover" en su algoritmo (Floyd, Henderson, y Gurtov, 2004).
- TCP Vegas: es una variante de TCP que alcanza entre un 40 y 70% mejor rendimiento que TCP Reno con menores pérdidas de paquetes, para lo cual utiliza tres técnicas basadas en mecanismo de retransmisión, evitación de la congestión y modificación del mecanismo de arranque lento (slow start) (Brakmo, O'Malley y Peterson, 1994).
- TCP Westwood: mejora el rendimiento de control de congestión usando como retroalimentación la medición de extremo a extremo del ancho de banda

disponible, esta medición es una estimación la cual se utiliza para configurar correctamente la ventana de congestión. (Mascolo, Casetti, Gerla, Lee y Sanadidi, 2000)

Varias investigaciones se han realizado referente al tema, las mismas que sirven de base para el desarrollo del proyecto, entre otras se puede mencionar:

- TCP for Wireless Network (Ashish, Jagannadha, Mansoor y Vijay, 2001): presenta un análisis de las redes inalámbricas frente a las redes cableadas. Con un enfoque en el desempeño de TCP.
- TCP Vegas: New Techniques for Congestion Detection and Avoidance (Brakmo, O'Malley y Peterson, 1994): describe las técnicas utilizadas por esta variante TCP y presenta los resultados de una simulación en l que evalúa rendimiento de TCP Vegas.
- TCP Westwood: congestion control with faster recovery (Mascolo, Casetti, Gerla, Lee y Sanadidi, 2000) realiza una simulación en NS-2 para demostrar el proceso de estimación de ancho de banda.
- Contribución a los modelos de gestión de las redes móviles Ad-hoc (Torres, 2015): si bien la investigación se refiere a modelos de gestión, también realiza simulación de redes móviles sobre NS-2 basado en un estudio previo de algunas características de la red para determinar el número de nodos necesarios para crear un escenario de simulación que cumpla con ciertos criterios para analizar métricas de las redes móviles ad hoc.
- Estudio y análisis de prestaciones de redes móviles Ad Hoc mediante simulaciones NS-2 para validar modelos analíticos (Torres, 2015]: este trabajo presenta la simulación de modelos matemáticos aplicados al diseño

de redes Ad Hoc, el cual aporta sobre todo en la simulación y posterior análisis de datos.

1.3 **Justificación.**

El desarrollo tecnológico y la necesidad del usuario de estar conectado a la red desde cualquier ubicación geográfica han sobrepasado los esquemas de conexión tradicionales, conllevando a las empresas de telecomunicaciones a desplegar grandes infraestructuras de redes inalámbricas, así como a los usuarios al uso de variedad de dispositivos móviles. Esto debido a la creciente demanda de acceso a la información sobre todo a través de internet, cuyo uso se ha vuelto tan necesario, no solo en el ámbito laboral, sino que también abarca otros ámbitos como el académico, de la salud, de seguridad, de entretenimiento, entre otros. En este sentido no sólo las empresas de telecomunicaciones sino las empresas en general según sea su necesidad han implementado redes inalámbricas que comúnmente son de infraestructura fija, sin embargo, existen escenarios en donde puede resultar muy difícil o imposible disponer de dicha infraestructura, para solventar esta limitante existen las redes móviles Ad Hoc en donde un terminal puede ser emisor, receptor y además llevar a cabo las funciones de enrutamiento.

Si bien las redes móviles Ad Hoc se han posicionado y han incrementado su uso, estas presentan algunas desventajas relacionadas con ancho de banda, calidad de servicio, seguridad, congestión, ruido e interferencia; lo que ha permitido abrir muchos campos de estudio para investigación con el fin de mejorar este tipo de redes.

Por otro lado, TCP incorpora mecanismos para el control de congestión que es uno de los temas de estudio actual de las redes móviles Ad Hoc.

Por consiguiente, este proyecto plantea abordar este campo de estudio para analizar detenidamente el comportamiento de diferentes variables que afectan el rendimiento de una red en diferentes escenarios, valiéndose del simulador de redes NS2 y de los mecanismos de control de congestión de TCP.

1.4 **Objetivo general.**

Analizar y comparar los mecanismos TCP para control de congestión en redes móviles Ad Hoc

1.5 **Objetivos específicos.**

- Analizar los conceptos, características, ventajas, desventajas de las redes móviles Ad Hoc.
- Estudiar los mecanismos TCP para el control de congestión, a fin de conocer de mejor manera su funcionamiento.
- Diseñar y simular redes móviles Ad Hoc en distintos escenarios mediante NS2.
- Analizar las mediciones obtenidas como resultado de la simulación.
- Comparar el comportamiento de los mecanismos TCP en los diferentes escenarios simulados.

1.6 **Estructura de la Tesis**

El presente proyecto, en su primer capítulo muestra de manera resumida el contenido y el enfoque general del trabajo de tesis. En el segundo capítulo se analizan los conceptos de las redes móviles Ad Hoc, mecanismos de control de congestión de TCP, parámetros de medición y el simulador NS2, que son objeto de estudio para el desarrollo de la presente Tesis. En el tercer capítulo se plantean los escenarios de simulación, así como la simulación efectuada. En el cuarto capítulo se realiza la discusión de los resultados obtenidos, para finalizar con las conclusiones y recomendaciones en el capítulo cinco.

CAPÍTULO 2

2 ESTADO DEL ARTE

2.1 **Redes móviles Ad Hoc.**

Una red inalámbrica Ad Hoc es una colección de nodos inalámbricos que se auto configuran para formar una red sin la necesidad de una infraestructura existente, cuando estas redes tienen nodos móviles se llaman redes móviles Ad Hoc (MANET) (ver Figura 1).

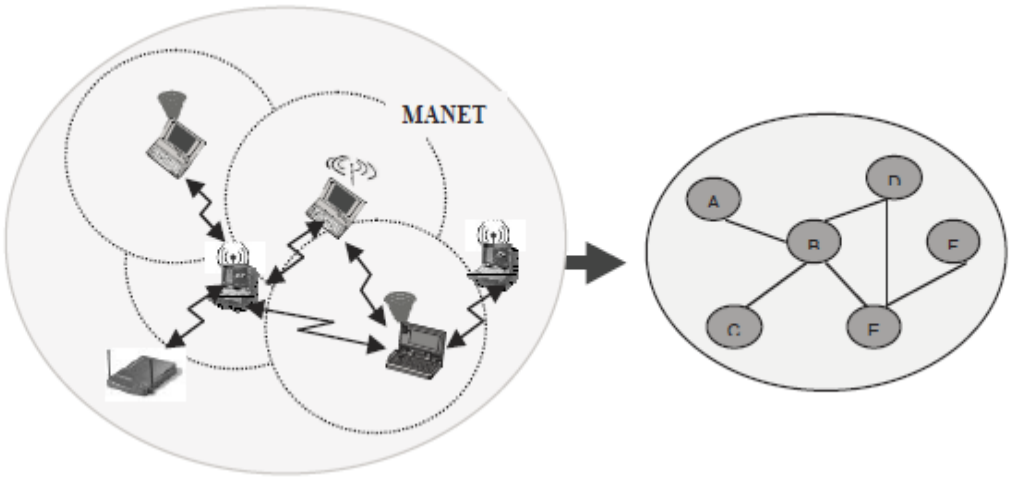

Figura 1 Red móvil Ad Hoc

Fuent: Basagni, Conti, Giordano, Stojmenovic, (2004)

2.1.1 Introducción

En los años 80 y 90 las investigaciones sobre redes Ad Hoc eran principalmente para propósitos militares, sin embargo, a partir de los años 90

la importancia de estas redes ha crecido sobre todo en el campo comercial y residencial debido a las siguientes razones: (Bangnan, Hischke, Walke, 2003).

- Incremento de dispositivos utilizados para el procesamiento de información tales como computadores portátiles, pocket PCs, PDAs, etc., y la necesidad de intercambiar información digital tales como videos, música y documentos entre personas dentro de áreas de corta cobertura.
- Aparecimiento de tecnologías inalámbricas tales como bluetooth, IEEE 802.11, hiperLAN 2 y W-CHAME3 que permiten establecer redes Ad Hoc entre una cantidad de nodos inalámbricos.
- Redes Ad Hoc como solución prometedora para aumentar la cobertura de los sistemas inalámbricos de banda ancha.

2.1.2 Características principales

Las redes móviles Ad Hoc heredan las características comunes de las redes inalámbricas y agregan características específicas de las redes Ad Hoc (Bakshi, Sharma y Mishra, 2013) tales como:

- Wireless: los nodos se comunican de forma inalámbrica y comparten el mismo medio (radio, infra rojo, etc.)
- Basadas en Ah Hoc: una red móvil Ah Hoc es una red temporal formada dinámicamente por un conjunto de nodos según la necesidad se presente.
- Autónomas y sin infraestructura: no dependen de ninguna infraestructura establecida o administración centralizada. Cada nodo

opera en modo distribuido (peer to peer) y actúa como un router independiente.

- Enrutamiento multihop: no requieren de routers dedicados, cada nodo actúa como un router y envía paquetes a los demás para permitir el intercambio de información entre los hosts móviles.
- Movimiento de nodos: cada nodo es libre de moverse mientras se comunica con otros nodos; por naturaleza la topología de una red Ad Hoc es dinámica debido al movimiento constante de los nodos participantes, esto hace que los patrones de intercomunicación entre nodos también cambien continuamente.
- Auto configurables: las actividades de descubrimiento de la topología y entrega de mensajes son ejecutadas por los propios nodos.
- Energía limitada: algunos o todos los nodos de una MANET pueden depender de baterías que limitan la operación de los nodos, por lo que lo que uno de los criterios relevantes en el diseño de estas redes es la conservación de energía.

2.1.3 Ventajas

- Movilidad: los nodos móviles inalámbricos pueden moverse al mismo tiempo en diferentes direcciones.
- Rapidez: la creación de una red Ad Hoc desde cero requiere algunos cambios de configuración, pero no requiere de hardware o software adicional, por lo que constituye una solución ideal si se desea conectar varios computadores de forma rápida y sencilla.

- Tolerancia a fallas: los protocolos de enrutamiento y transmisión de las redes MANET están diseñados para manejar situaciones de fallas de conexión, lo que las hace tolerante a fallas.
- Conectividad: el uso de puertas de enlace no es necesario para la comunicación dentro de una MANET gracias a la colaboración entre los nodos para la entrega de paquetes.
- Costo: las redes MANET resultan económicas debido a que se eliminan los costos asociados con infraestructura y se reduce el consumo de energía en los nodos móviles (Bakshi, Sharma y Mishra, 2013).

2.1.4 Limitantes

- Ancho de banda limitado: la capacidad de los enlaces inalámbricos es mucho menor que los enlaces cableados, lo que constituye una limitante para las redes MANET.
- Energía limitada: la duración de las baterías es limitada, lo que no permite tiempos de operación prolongados en los nodos móviles, es por ello que se han implementado algunos algoritmos para la conservación de energía.
- Latencia: debido a la necesidad de conservación de energía cuando los nodos no tienen datos que transmitir pasan a un estado inactivo o de espera, cuando el intercambio de datos entre dos nodos pasa a través de nodos que está en espera, puede añadirse mayor latencia si el algoritmo de enrutamiento decide que estos nodos deben activarse.
- Errores de transmisión: la atenuación y la interferencia son características heredadas de las redes inalámbricas, las cuales incrementan la tasa de error. (Bakshi, Sharma y Mishra, 2013).

2.1.5 Escenarios de aplicación

Las redes móviles Ah Hoc pueden ser utilizadas en cualquier lugar donde exista poco o ninguna infraestructura de red o ya sea que la infraestructura es costosa o inaccesible; las redes MANET son aplicables a una gran variedad de situaciones en las que las redes tradicionales no se pueden aplicar, a continuación, se menciona algunas áreas de aplicación:

- Militar: es muy adecuado su uso para situaciones militares en las que implementar una infraestructura de red no es factible, y es primordial la transmisión de información de la situación y posicionamiento.
- Sensores: las redes de sensores son otra aplicación de las redes MANET, las cuales están compuestas de un gran número de sensores y son utilizadas en distintas áreas de medición como temperatura, presión, contaminación, humedad, entre otras.
- Zonas de desastre: pueden ser utilizadas para operaciones de rescate y emergencia en situaciones de desastres como incendios inundaciones o terremotos; debido a que los desastres pueden ocurrir en lugares donde no existe una infraestructura de comunicaciones y el despliegue rápido de una red de comunicaciones es necesario para transmitir información importante.
- Redes de área personal (PAN, por sus siglas en inglés): son creadas por varios dispositivos móviles principalmente en modo Ad Hoc, y frecuentemente usadas en redes domésticas (Bakshi, Sharma y Mishra, 2013).

2.2 **Protocolo de control de transferencia TCP**

En el modelo de referencia TCP/IP se encuentra la capa de transporte (Figura 2), la cual desempeña el papel crítico de proporcionar directamente servicios de comunicación a los procesos de aplicación que se ejecutan en hosts diferentes, para cumplir su objetivo la capa de transporte se vale de protocolos y principalmente de los protocolos TCP y UDP.

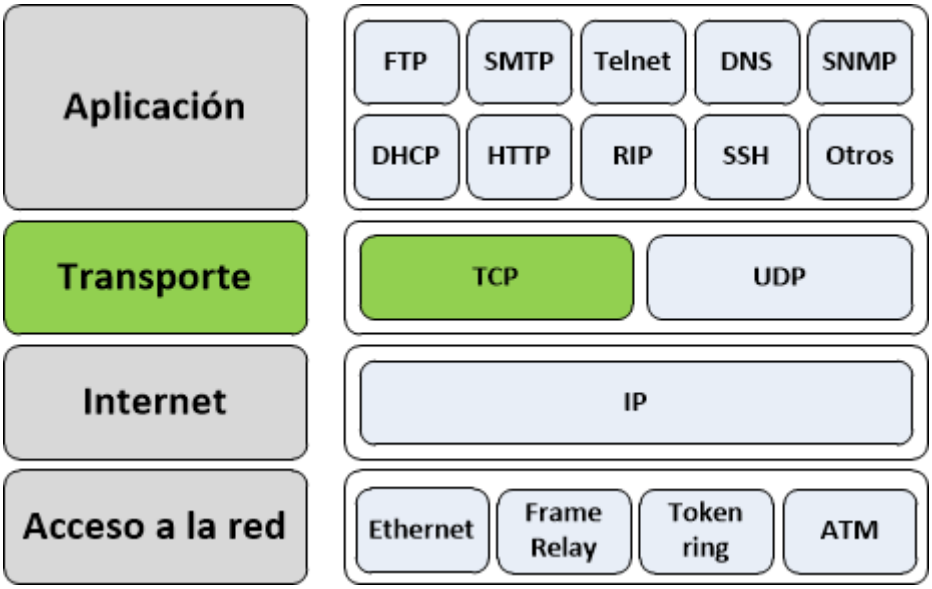

Figura 2 Modelo TCP/IP

Debido a que el presente proyecto se centra en el comportamiento de las variantes de TCP para el control de congestión, se hará énfasis en el protocolo TCP.

Uno de los protocolos más complejos de TCP/IP es el protocolo de control de transmisión (TCP), que se construyó para proporcionar servicios orientados a la conexión, transferencia de datos confiable y comunicación de extremo a extremo entre los hosts de una red de conmutación de paquetes (Houmkozlis y Rovithakis,

2012); Según (IETF, 1981) se define como proceso a los elementos activos de un host de acuerdo con la definición de que un proceso es un programa en ejecución, en este sentido y de acuerdo al modo de operación de TCP, en el origen los procesos transmiten datos llamando al módulo de TCP y pasando búferes de datos como argumentos de la llamada, el módulo TCP por su lado empaqueta en segmentos los datos provenientes de estos búferes y efectúa una llamada al módulo de internet para que transmita cada segmento al módulo de TCP en el destino. Cada flujo de datos debe ser segmentado en varios paquetes si supera un umbral predefinido, el mismo que es acordado durante el establecimiento de la conexión. Un segmento consta de una cabecera y el cuerpo que incluye los datos a transmitir, el encabezado posee varios campos creados por un emisor y utilizados por el receptor para gestionar adecuadamente los segmentos recibidos (Houmkozlis y Rovithakis, 2012).

Durante la transmisión TCP existe la posibilidad de que los enlaces se saturen, provocando congestión, pérdida de paquetes, reducción del rendimiento, incremento de latencia de extremo a extremo, lo que degrada el rendimiento global de la red. Por ello uno de los principales retos de TCP es solucionar el problema de la congestión, el cual al afectar el rendimiento de la red también afecta el de las aplicaciones que dependen de la red; para lo cual son necesarios mecanismos que regulen el flujo de los emisores en cuanto la congestión de red se produzca (Kurose y Ross, 2010).

Para afrontar el problema de la congestión se plantean algunos métodos para el control de la congestión, según se explica a continuación:

2.2.1 Métodos para control de congestión

El control de congestión es la adaptación de la tasa de transferencia en respuesta a las cambiantes condiciones de la red tales como pérdida de paquetes o retardos.

En un enfoque general según (Kurose y Ross, 2010), se puede diferenciar entre las técnicas de control de congestión basándose en si la capa de red proporciona alguna ayuda explícita a la capa de transporte con propósitos de controlar la congestión, en este sentido se habla de control terminal a terminal y control asistido por la red como se describe a continuación.

2.2.1.1 Control terminal a terminal

Muchos de los algoritmos de control de congestión están diseñados para funcionar en un esquema de terminal a terminal, esquema que puede subdividirse en esquemas basados en origen y basados en el destino (Houmkozlis y Rovithakis,2012).

En esquemas basados en el origen, el origen establece su tamaño de ventana en función de la congestión de red percibida, si el origen percibe que en la ruta entre origen y destino existe congestión mínima, entonces puede incrementar el tamaño de ventana; por otro lado, si percibe congestión a lo largo de la ruta, reducirá el tamaño de ventana. (Houmkozlis y Rovithakis, 2012) (Kurose y Ross, 2010).

En esquemas de control de congestión basados en el destino, es el destino el que controla el tamaño de ventana.

2.2.1.2 Control asistido por la red

En este método los equipos de capa de red es decir los routers, proporcionan al origen información del estado de la red referente a la congestión. Esta información puede ser proporcionada de dos formas, directa, desde un router de la red al origen (emisor) mediante un paquete de asfixia o bloqueo el cual notifica la congestión; y mediante la actualización de un campo de un paquete que se transmite del origen al destino para notificar que existe congestión.

Debido a que la capa IP no proporciona información de congestión de la red, TCP debe utilizar un control terminal a terminal en lugar de un control de congestión asistido por la red.

2.2.2 Mecanismos para control de congestión

Una de las funciones de TCP es realizar el control de congestión, los mecanismos utilizados para por TCP para controla la congestión están basados en los siguientes algoritmos: arranque lento, evitación la congestión y recuperación rápida. Los algoritmos arranque lento y evitación de la congestión deben ser utilizados por el emisor a fin de controlar la cantidad de

tráfico enviado, para lo cual se definen tres variables (Rodríguez, Vidal y Alves, 2004):

- cnwd (congestión window), controla en el emisor la cantidad de datos que se pueden enviar sin haber recibido un ACK.
- rwnd (receiver's advertised window), indica la cantidad de datos que puede recibir el destino.
- ssthresh (slow start treshold), indica en qué fase del control de congestión se encuentra el emisor.

A continuación, se estudian los algoritmos antes mencionados:

2.2.2.1 Arranque lento

Este algoritmo se utiliza para iniciar la transmisión de paquetes y para detectar el ancho de banda disponible.

- Se inicializa la ventana de congestión con un valor igual a 1 MSS (tamaño máximo de segmento)

  Cwnd=1

- Cuando se recibe un ACK la ventana de congestión aumenta en una unidad, permitiendo la transmisión de dos segmentos de tamaño máximo, y así sucesivamente generando un crecimiento exponencial.

  Cwnd++

- Si se detecta que existe congestión se establece el valor de la variable ssthresh o umbral a la mitad del tamaño de la ventana de congestión.

  Ssthresh = cwnd/2

El algoritmo de arranque lento finaliza cuando (Kurose y Ross, 2010).:

- Un ACK no llega al emisor, con lo que la ventana de congestión se inicializa en un 1 MSS e inicia el proceso de arranque lento nuevamente.
- El tamaño de la ventana de congestión alcanza el valor del umbral, en cuyo caso inicia la fase de evitación de la congestión.
- Se reciben tres ACK idénticos, con lo que inicia la fase de recuperación rápida.

Arranque lento no evita la congestión, pero evita la congestión inmediata asegurándose de que el remitente no envíe gran cantidad de información (Olasoji, Olanrewaju y Adebayo, 2016).

2.2.2.2 Evitación de la congestión

Cuando el tamaño de la ventana alcanza el umbral establecido la fase de arranque lento se detiene y comienza la fase de evitación de la congestión, en esta fase el mecanismo como su nombre lo indica pretende evitar llegar

a una congestión, para lo cual en lugar de incrementar de manera exponencial el valor de la ventana de congestión, incrementa solamente en un MSS es decir de manera lineal.

Evitación de la congestión y arranque lento son algoritmos independientes con objetivos diferentes, sin embargo, cuando ocurre la congestión se aplican conjuntamente de la siguiente manera (Stevens, 1997):

- Se inicializa la ventana de congestión en 1 MSS y la variable ssthresh en 65535 bytes.
- Cuando se produce la congestión, el valor de ssthresh se estable en la mitad del tamaño de ventana actual; si la congestión se indica mediante un tiempo de espera, el valor de la ventana de congestión será igual a 1 MSS, es decir inicia el algoritmo de arranque lento.
- Cuando los datos son recibidos por el receptor, el tamaño de la ventana de congestión aumenta, pero la forma en que aumenta depende de si TCP está ejecutando arranque lento o evitación de la congestión.

Si la ventana de congestión es menor o igual que ssthresh, TCP está ejecutando arranque lento, de lo contrario TCP está ejecutando evitación de la congestión.

2.2.2.3 Recuperación rápida

Recuperación rápida es una modificación del algoritmo de evitación de la congestión en el que la congestión es detectada a través de tres acuses de recibo (ACK) duplicados.

Cuando un segmento arriba a su destino en desorden, un ACK duplicado se emite inmediatamente, cuyo propósito es informar al origen o emisor e indicar el número de secuencia que se espera. Dentro de la red un ACK duplicado puede usarse por las siguientes razones:

- Debido a segmentos perdidos, en este caso todos los segmentos que llegan después de un segmento perdido activarán ACK duplicados.
- Debido a la reordenación de segmentos.
- Debido a la replicación de ACKs o segmentos de datos.

En recuperación rápida, la fase de arranque lento no se lleva a cabo después de que el paquete descartado ha sido reenviado, en lugar de esto, la fase de evitación de la congestión se lleva a cabo, con lo que el enlace de comunicación está libre y el proceso de arranque lento no se reinicia, esto se conoce como algoritmo de recuperación rápida (Olasoji, Olanrewaju y Adebayo, 2016).

2.2.3 Variantes TCP

TCP es el protocolo responsable de la entrega de datos de extremo a extremo, para lo cual emplea mecanismos que ayudan a que la entrega sea fiable, sin embargo la continua evolución de las aplicaciones y de internet, así como la demanda de servicios por parte de los usuarios han incrementado el tráfico en la red, y con ello el problema de la congestión; todo esto obliga a una mejora constante del protocolo, es por ello que TCP ha sufrido algunas variantes que tienen como objetivo superar los problemas de congestión que afectan el rendimiento de la red.

En esta sección se revisan las diferentes variantes de TCP:

2.2.3.1 TCP Tahoe

Esta variante de TCP también conocida como BSD Network Release 1.0 (BNR1), es la implementación original del mecanismo para control de congestión del sistema operativo BSD en el año 1988 (Liu 2001).

La versión inicial conocida como Tahoe antiguo implementó los algoritmos de arranque lento y evitación de la congestión, basado en el tiempo de espera para el control de congestión, pero debido a las deficiencias en la detección de congestión a través del tiempo de espera, se agregó un nuevo algoritmo llamado retransmisión rápida (Olasoji, Olanrewaju y Adebayo, 2016).

Tahoe funciona de la siguiente manera:

- Declara que existe congestión cuando el emisor recibe tres ACK duplicados.
- Retransmite el segmento perdido sin esperar la expiración del tiempo de espera.
- Reajusta la ventana de congestión a 1 MSS e inicia el algoritmo de arranque lento.

2.2.3.2 TCP Reno

También conocida como BSD Network Release 2.0 (BNR2) fue implementada en el año 1990, conserva las características de TCP Tahoe, e incorpora el algoritmo de recuperación rápida. A diferencia de Tahoe que reajusta la ventana de congestión a 1 MSS e inicia el algoritmo de arranque lento, Reno hace que la ventana de congestión sea igual al valor de ssthersh entrando inmediatamente en fase de evitación de la congestión y por ende se da el algoritmo de recuperación rápida (Rodríguez, Vidal y Alves, 2004).

Reno realiza recuperación rápida de la siguiente manera: (Ait-Hellal y Altman, 2000).

- La variable Ssthresh se establece en la mitad del tamaño de la venta de congestión (ssthresh = cwnd/2)
- La ventana de congestión toma el valor del umbral ssthresh mas tres veces MSS (cwnd = ssthresh +3 MSS).

- Por cada ACK duplicado recibido se incrementa el valor de la ventana de congestión en 1 y se envía un nuevo paquete.
- Cuando se recibe el primer ACK no duplicado, la ventana de congestión toma el valor del umbral (cwnd = ssthresh).

La principal ventaja de TCP Reno sobre TCP Tahoe es que Reno mantiene el registro de nuevos datos con ACKs duplicados y evita iniciar la fase de arranque lento (Liu, 2001).

2.2.3.3 TCP Vegas

Es una modificación de TCP RENO, Vegas lee y registra el reloj del sistema cada vez que se envía un paquete, cuando se recibe un ACK, lee nuevamente el reloj y realiza el cálculo del RTT (round-trip time o tiempo de ida y vuelta) promedio, la desviación media y el tiempo de espera (Ait-Hellal y Altman, 2000).

Vegas utiliza tres técnicas para mejorar el rendimiento de TCP y reducir la pérdida de paquetes, estas son (Olasoji, Olanrewaju y Adebayo, 2016).:

- Mecanismo de arranque lento modificado: introduce una fase de ventana de congestión (cwnd) constante en cada RTT, y aumenta exponencialmente en cada RTT de manera alternada.

  Durante la fase constante se evalúa el rendimiento estimado y alcanzado, si el rendimiento alcanzado es menor que el estimado

durante cierto período de tiempo, se mantiene un crecimiento lineal y no exponencial de la ventana de congestión (Liu, 2001).

Cuando el rendimiento alcanzado supera el estimado, TCP vegas pasa a la fase evitación de la congestión, con lo que se evita pérdidas en la fase de arranque lento (Ait-Hellal y Altman, 2000).

- Mecanismo mejorado de evitación de la congestión: monitorea los cambios en el rendimiento comparando el rendimiento promedio con el esperado, calculado usando el tamaño de la ventana de congestión. Este mecanismo utiliza la información del rendimiento como indicador para mantener la cantidad óptima de datos en la red (Liu, 2001).
- Uso de un nuevo mecanismo de retransmisión: vegas lleva un registro de cuando se envió cada paquete y calcula una estimación de RTT para cada transmisión, esto se hace monitoreando cuánto tiempo le tomó cada ACK volver al remitente. Cada vez que se recibe un ACK duplicado se realiza la siguiente comprobación:

  ¿RTT actual > RTT estimado?

  Si se cumple la condición anterior se retransmite el paquete sin esperar la llegada de tres ACK duplicados o un tiempo de espera, como lo hace reno (Oyeyinka, Oluwatope, Akinwale, Folorunso, Aderounmu y Abiona, 2011).

Según (Oyeyinka et al., 2011) existen otras variantes de TCP como: STCP, HSTCP, BIC-TCP, CUBIC, Fast TCP, TCP Africa, West Wood, TCP Illinois

entre otras. Sin embargo, en el presente proyecto se menciona y explica acerca de Tahoe, Reno y Vegas que son las variantes objeto de análisis.

2.3 **Parámetros de medición.**

Cuando un paquete es enviado desde el origen, para llegar al destino debe pasar por una serie de dispositivos intermedios de red como switches, routers, firewalls, etc., durante el trayecto del paquete por estos dispositivos éste puede verse afectado por algunos factores, que en el peor de los casos causan la pérdida del paquete.

Estos factores conocidos como parámetros de medición de red permiten evaluar el comportamiento de una red, así como identificar posibles problemas y tomar decisiones para la mejora de la red, aunque existen varios parámetros de red en el presente trabajo se estudiará la latencia, el RTT y el jitter.

2.3.1 Latencia

Según (Yáquez, n.d.), la latencia es el retardo o tiempo de viaje de un paquete medido de extremo a extremo. El retardo de extremo a extremo es la acumulación de varios retardos existentes en la red.

Los retardos más importantes son: retardo de procesamiento, retardo de cola, retardo de transmisión, y retardo de propagación (Kurose y Ross, 2010) como se muestra en la siguiente figura:

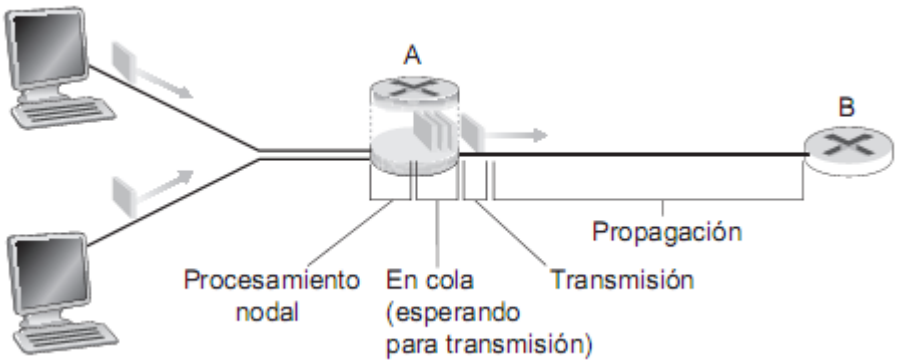

Figura 3 Tipos de retardo

Fuente: Kurose y Ross (2010)

De acuerdo con la Figura 3, *el retardo de procesamiento*, es el tiempo requerido para examinar la cabecera del paquete y determinar la ruta; después del procesamiento nodal, el router A dirige el paquete a la cola que precede al enlace hacia el router B.

Una vez en la cola, el paquete experimenta un *retardo de cola* mientras espera para ser transmitido en el enlace, este retardo dependerá de la cantidad de paquetes existente en la cola esperando su transmisión.

Generalmente en las redes de conmutación de paquetes el primer paquete en llegar es el primero en ser transmitido, por lo que un paquete será trasmitido después de que todos los paquetes en la cola hayan sido transmitidos. El *retardo de transmisión* es la cantidad de tiempo que se requiere para colocar todos los bits del paquete sobre el enlace, este tiempo está dado por L/R, donde L es la longitud de paquetes en bits y R la tasa de transmisión del enlace.

Finalmente, una vez que los bits son colocados sobre el enlace, necesitan propagarse hacia el destino, router B en la Figura 3. El tiempo necesario para propagarse desde el comienzo del enlace hasta el router B, se conoce como *retardo de propagación*. Este retardo está dado por *d/s*, donde *d* es la distancia entre dos nodos y *s* la velocidad de propagación del enlace, la misma que depende del medio (fibra óptica, cobre, etc.) y está en el rango de:

$$2*10^8 \text{ metros/segundos a } 3*10^8 \text{ metros/segundos}$$

2.3.2 RTT (Round Trip Time)

RTT es el tiempo que tarda un paquete en viajar desde el origen hasta el destino y volver nuevamente al origen, este tiempo incluye los retardos de propagación, retardos de cola y retardos de procesamiento de los paquetes.

La Figura 4 ilustra el concepto de RTT.

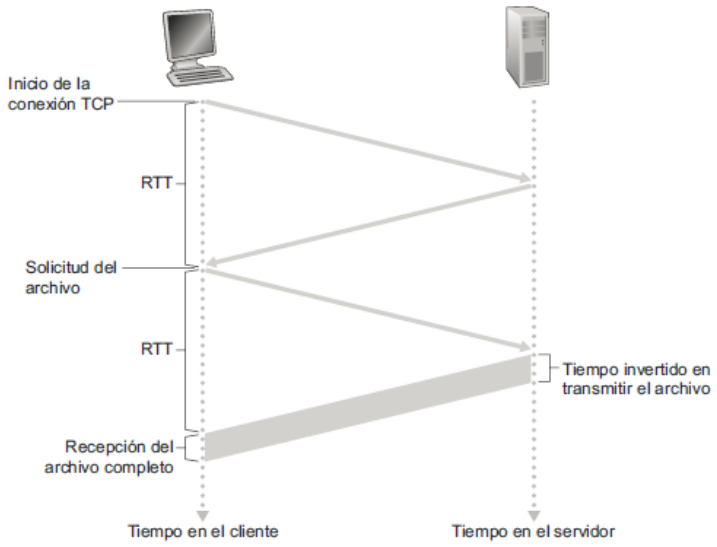

Figura 4 Definición de retardo

Fuente: Kurose y Ross, (2010)

2.3.3 Jitter

El jitter también conocido como latencia variable entre paquetes, es la diferencia de tiempo extremo a extremo en la red entre paquetes secuenciales de un mismo flujo, como se representa en la Figura 5.

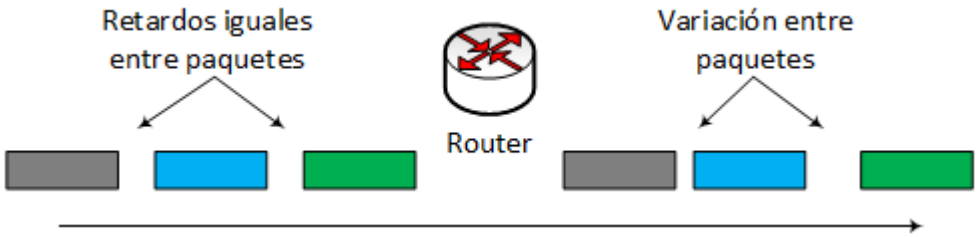

Figura 5 Representación de Jitte

Fuente: Yáquez, (n.d.)

El jitter es causado por la congestión en la red, tiempos de espera variables en la cola, pérdida de sincronización o por las diferentes rutas por las que un paquete es enviado para llegar a su destino.

Las aplicaciones en tiempo real tienen requerimientos estrictos de latencia y jitter

2.4 **Simulador NS2 (Network Simulator-2)**

Existen en el mercado varias herramientas de simulación que contribuyen al diseño e implementación de redes, su uso se ha expandido notablemente en el campo de la investigación debido a que permiten medir, comparar y evaluar parámetros de red

como rendimiento, retardo, jitter entre otros; y determinar el comportamiento de una determinada red en distintos escenarios sin la necesidad de invertir en infraestructura física.

Según Morales, Calle, Tovar y Cuéllar (2013), a la hora de seleccionar una herramienta de simulación es importante evaluar ciertos criterios que permitan argumentar la selección de la herramienta que más se adapte a las necesidades. Estos criterios son:

- Nivel de uso investigativo de las herramientas en áreas académicas e investigativas.
- Tipo de licencia, enfocándose en software comercial y software libre.
- Curva de aprendizaje, con respecto al nivel de exigencia para lograr un manejo adecuado de la herramienta.
- Plataformas soportadas, refiriéndose a los diversos sistemas operativos en los que la herramienta se puede ejecutar y utilizar sin problema.
- Presentación de resultados, la capacidad de la herramienta para mostrar los resultados de forma gráfica.
- Tecnologías y protocolos de capa 2 y capa 3 que soporta, refiriéndose a las capas del modelo de referencia OSI.
- Tráfico que permite modelar, con respecto a las aplicaciones y servicios que la herramienta está en capacidad de simular.

La Tabla 1, muestra la evaluación de seis herramientas de simulación de red, basada en los criterios antes mencionados.

Tabla 1 Evaluación de herramientas de simulación

HERRAMIENTA / CRITERIO	OPNET	OMNET	NS-3	GNS3	NS-2	NC-TUNS
Uso investigativo	Alto	Alto	Medio	Bajo	Alto	Alto
Licencia	Comercial	Libre	Libre	Libre / Comercial	Libre	Libre
Curva de aprendizaje	Alto	Alto	Alto	Bajo	Alto	Alto
Plataformas	Windows, Unix	Windows, Unix	Windows, Unix, Mac	Windows, Unix, Mac	Windows, Unix, Mac	Linux
Gráficos de resultados	Buena	Aceptable	Aceptable	Limitada	No tiene	Aceptable
Tecnologías de nivel 2 y nivel 3 que soporta	Alto	Alto	Medio	Bajo	Alto	Alto
Tráfico que permite modelar	Alto	Medio	Medio	Nulo	Alto	Alto

Nota. Fuente: Morales et al., 2013

Con base a los criterios evaluados se ha seleccionado NS2 como herramienta de simulación para el presente proyecto, considerando que, es ampliamente utilizado en el campo académico e investigativo, su licenciamiento de tipo libre permite utilizar sin problema la herramienta, aprovechando todas las funcionalidades que esta entrega, sin la necesidad de inversiones económicas mayores; es multiplataforma, si bien el aprendizaje de la herramienta puede resultar complejo, existe gran cantidad de información disponible que puede contribuir a una mejor comprensión y aprendizaje de la herramienta.

NS2, además soporta la simulación del protocolo TCP, enrutamiento, protocolos multicast sobre redes cableadas e inalámbricas (Sourceforge. 2011),

Si bien NS2 no tiene la capacidad de representar gráficamente los resultados, los datos arrojados en archivos planos pueden ser leídos y representados por herramientas como NAM, XGRAPH y AWK que se integran perfectamente con NS2.

2.4.1 Estructura de NS2

NS2 es un simulador de eventos discretos que nació como una variante del simulador de red denominado REAL en el año 1989 (Sourceforge. 2011)

Se basa en dos lenguajes de programación: OTcl, versión extendida de Tcl (Tool command language) que sirve como interfaz para el usuario y C++ que contiene la implementación de los protocolos. De esta forma cuando se realiza una simulación el código OTcl se vincula con las clases de C++ para poder ejecutar las funciones implementadas en C++ (Sourceforge. 2011),

En la Figura 6, se muestra la arquitectura de NS2.

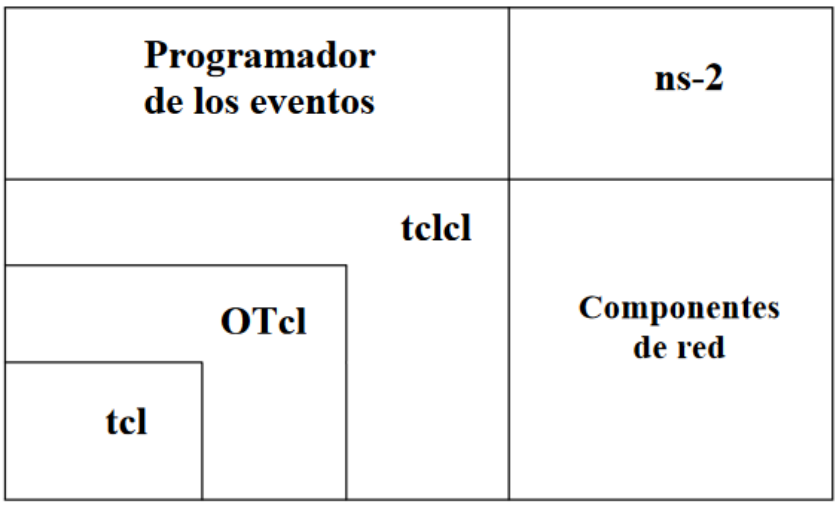

Figura 6 Estructura de NS2

Fuente: Chung y Claypool (n.d.)

En la Figura 7, se muestra la visión del simulador desde el punto de vista del usuario.

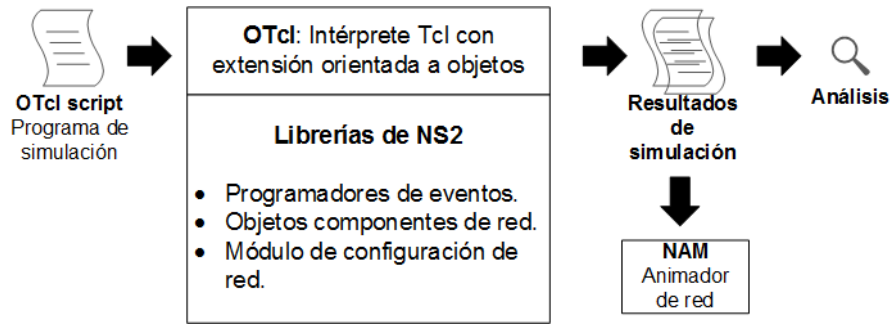

Figura 7 Secuencia de simulación con herramientas de NS2

Fuente: Chung y Claypool, (n.d.)

Como se observa en la Figura 7, se inicia con un script OTcl que es procesado por NS2, posteriormente los resultados de la simulación se almacenan en archivos de registro que genera NS2, cada paquete transmitido durante la simulación genera una línea en el archivo de registro donde se describen varios parámetros del paquete, teniendo la posibilidad de guardar datos específicos de cada nodo o de toda la red. NS2 permite generar un archivo de registro especial, el cual es interpretado por NAM (Network Animator), el mismo que ofrece una visualización básica de la simulación. (López y García, 2010)

2.4.2 TCL y OTCL

Tcl (Tool Command Language) es un lenguaje que posee una sintaxis muy simple y permite una sencilla integración con otros lenguajes (Altman y Jiménez, 2012), Tcl presenta las siguientes características principales:

- Permite un desarrollo rápido.
- Proporciona una interfaz gráfica.
- Compatible con varias plataformas.
- Flexible para la integración.
- Fácil de usar.
- Es libre.

En la Figura 8, se muestra un ejemplo de un script que incluye la creación de un procedimiento, asignación de valores a variables y la creación de un bucle (Chung y Claypool, n.d.)

```
# Writing a procedure called "test"
proc test {} {
    set a 43
    set b 27
    set c [expr $a + $b]
    set d [expr [expr $a - $b] * $c]
    for {set k 0} {$k < 10} {incr k} {
        if {$k < 5} {
            puts "k < 5, pow = [expr pow($d, $k)]"
        } else {
            puts "k >= 5, mod = [expr $d % $k]"
        }
    }
}

# Calling the "test" procedure created above
test
```

Figura 8 Ejemplo de código fuente para TCL

Fuente: Chung y Claypool, (n.d)

2.4.3 NAM (Network Animator)

NAM es una herramienta que permite visualizar trazas de simulación de red, así como del mundo real. NAM utiliza el archivo de resultados generado por

NS2, el cual contiene información de la topología como nodos, enlaces y trazas de paquetes (Henderson, 2011).

En la Figura 9, se muestra el editor de NAM y sus principales funciones:

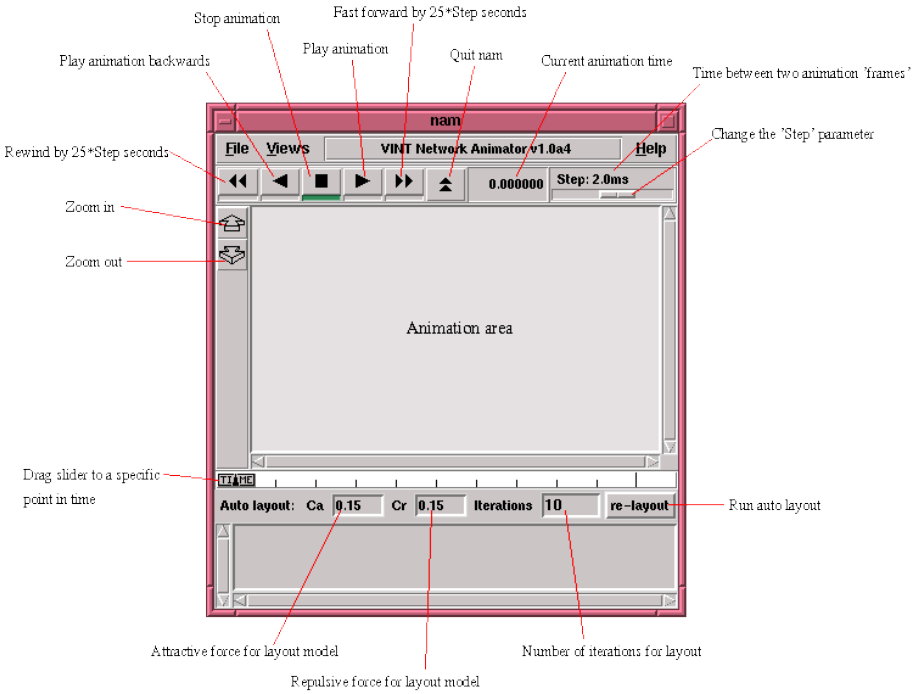

Figura 9 Funciones de NAM

Fuente: Saini, (2017)

NAM para realizar la animación utiliza los siguientes elementos (Henderson, 2011):

- Nodo: representa un origen, un host o un router. Un nodo puede tener tres formas: círculo, cuadrado o hexágono; pero una vez creado no puede cambiar su forma.
- Link (enlace): se crean enlaces entre nodos para formar la topología de red. NAM internamente utiliza dos enlaces simplex, aunque desde el punto de vista del usuario los enlaces son dúplex.

- Colas: las colas necesitan ser construidas entre dos nodos, éstas están asociadas a un solo extremo del enlace y se visualizan como paquetes apilados.
- Paquete: los paquetes se visualizan como un bloque con una flecha, la dirección de la flecha muestra la dirección del flujo del paquete. Los paquetes en cola se muestran como cuadrados pequeños.
- Agente: se utilizan para separar los estados de protocolo de los nodos. Un agente tiene un nombre que es un identificador único, se muestra como un cuadrado con el nombre dentro y se dibuja junto a su nodo asociado.

2.4.4 Xgraph

Xgraph es un trazador de datos de propósito general, traza datos de cualquier cantidad de archivos en un mismo gráfico y tiene la capacidad de manejar conjuntos de datos de tamaño ilimitado (General Purpose, 2016). Se basa en la lectura de archivos de trazas con formato de columnas x-y, obtenidos como resultado de la ejecución de scripts escritos en algún lenguaje, como por ejemplo TCL.

Como se muestra en la Figura 10, xgraph tiene la capacidad de personalizar los gráficos en cuanto a colores, títulos, nombre de los ejes; además tiene la posibilidad de exportar los resultados en distintos formatos de tal manera que puedan ser utilizados posteriormente en reportes u otros documentos (General Purpose, 2016).

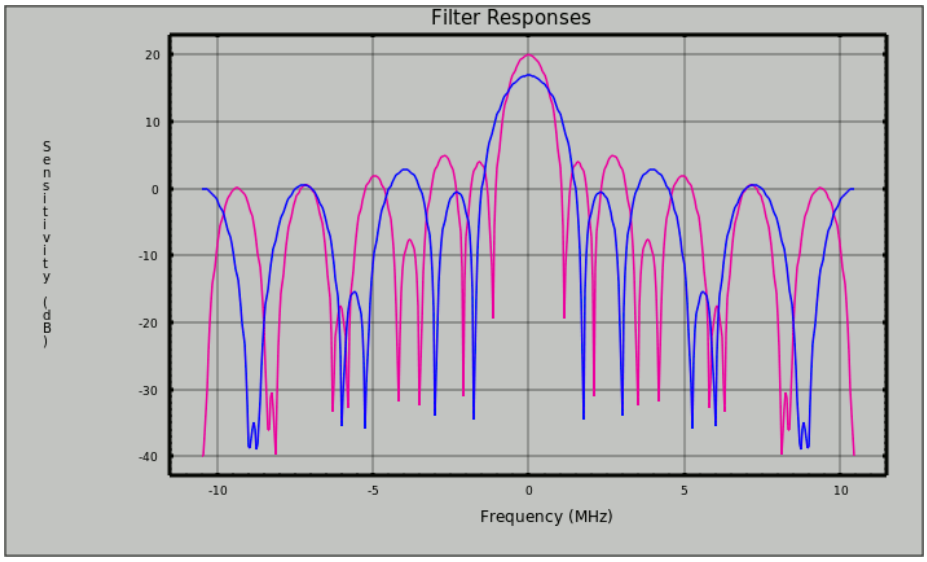

Figura 10 Ejemplo de gráfica obtenida con Xgraph

Fuente: General Purpose, (2016)

2.4.5 AWK

AWK, hace referencia a los nombres de sus autores es un lenguaje de programación diseñado para procesar datos basados en archivos de texto, su función básica es buscar en los archivos líneas o secuencias de caracteres que contengan ciertos patrones, cuando existe una coincidencia con uno de los patrones, AWK realiza acciones específicas (The GNU Awk, 2016).

La ejecución de un programa AWK, sigue la siguiente sintaxis: awk programa.awk archivo-de-entrada.

Por ejemplo, la Tabla 2, contiene notas de estudiantes almacenadas en un archivo de texto, se puede utilizar AWK para procesar el archivo y calcular

el promedio, así como determinar el estado de aprobado o reprobado de cada estudiante.

Tabla 2 Ejemplo de calificaciones para cálculo de promedio con AWK

NOMBRE	NOTA1	NOTA2	NOTA3	NOTA4	NOTA5	NOTA6	NOTA7
Estudiante1	5	9,5	6	9,5	5	9,5	5
Estudiante2	3	7	8,5	7	9,25	10	6
Estudiante3	6,75	8,5	6,75	8,5	6,75	10	6,75
Estudiante4	1	4	7	3	2	8	8,57
Estudiante5	6,7	6,5	10	6,5	8	6,5	6,7
Estudiante6	9	8	7,5	8	9,5	7	2
Estudiante7	10	8,25	9	8,25	10	9,25	10
Estudiante8	4	9,75	1	9,75	2	9,75	9,5
Estudiante9	8,75	10	9	10	7	9,25	8,75
Estudiante10	1	6	3	6,25	5	7	3

El archivo de texto quedaría como se muestra en la Figura 11.

*estudiantes ×

```
Estudiante1 5 9.5 6 9.5 5 9.5 5
Estudiante2 3 7 8.5 7 9.25 10 6
Estudiante3 6.75 8.5 6.75 8.5 6.75 10 6.75
Estudiante4 1 4 7 3 2 8 8.57
Estudiante5 6.7 6.5 10 6.5 8 6.5 6.7
Estudiante6 9 8 7.5 8 9.5 7 2
Estudiante7 10 8.25 9 8.25 10 9.25 10
Estudiante8 4 9.75 1 9.75 2 9.75 9.5
Estudiante9 8.75 10 9 10 7 9.25 8.75
Estudiante10 1 6 3 6.25 5 7 3
```

Figura 11 Ejemplo de archivo de texto que será procesado por AWK

El código AWK para calcular el promedio se muestra en la Figura 12.

ejemplo.awk

```
BEGIN {
        print("\n\n******** CALCULAR PROMEDIO ********\n");
        promedio=0;
}

{
        #Leer el nombre de la primera columna (nombre del estudiantes).
        nombre = $1;
        #Leer las notas de cada estudiante y calular el promedio.
        promedio = ($2+$3+$4+$5+$6+$7+$8)/7;
        if(promedio >= 7) {
                printf("%s %.2f %s \n", nombre, promedio,"Aprobado");
                printf("%s %.2f %s \n", nombre, promedio,"Aprobado") > "resultado.txt";
        }
        else{
                printf("%s %.2f %s \n", nombre, promedio,"Reprobado");
                printf("%s %.2f %s \n", nombre, promedio, "Reprobado") > "resultado.txt";
        }
}
```

Figura 12 Ejemplo de código fuente escrito en AWK

Como resultado se presenta el promedio de las siete notas de los estudiantes y el estado "Aprobado" para los estudiantes cuyo promedio es mayor o igual que 7, y "Reprobado en caso contrario. La Figura 13, muestra la ejecución y el resultado del código escrito en AWK.

```
carlos@carlos-VirtualBox:~$ gawk -f ejemplo.awk estudiantes


******** CALCULAR PROMEDIO ********

Estudiante1 7.07 Aprobado
Estudiante2 7.25 Aprobado
Estudiante3 7.71 Aprobado
Estudiante4 4.80 Reprobado
Estudiante5 7.27 Aprobado
Estudiante6 7.29 Aprobado
Estudiante7 9.25 Aprobado
Estudiante8 6.54 Reprobado
Estudiante9 8.96 Aprobado
Estudiante10 4.46 Reprobado
```

Figura 13 Resultado obtenido de la ejecución del código TCL de la Figura 12

En resumen, este lenguaje es ampliamente utilizado para realizar análisis de gran cantidad de datos obtenidos como resultado de simulaciones, en el presente proyecto ha sido muy útil para extraer desde el archivo de trazas

entregado por NS2, información de rendimiento, retardo, jitter, así como paquetes enviados, recibidos, perdidos y la tasa de entrega de paquetes.

CAPÍTULO 3

3. DISEÑO Y SIMULACIÓN DE UNA RED MÓVIL AD HOC

3.1. Diseño de la Red: Definición de escenario

Para analizar el comportamiento de los mecanismos de TCP para el control de congestión en redes móviles Ad hoc, y con la finalidad de obtener resultados más acertados, se han definido los parámetros que constan en la Tabla 3, los mismos que se basan en los planteados por Kurkowsky, Navidi y Camp (2007).

Tabla 3 Parámetros para el escenario de simulación

Parámetro	Valor
Área de simulación	300m x 600m
Cantidad de nodos	30, 60 y 90
Tiempo de simulación	60s
Protocolo de capa de red	AODV
Protocolo de capa de transporte	TCP/FTP
Tamaño de paquetes	3000 bytes
Variantes TCP	Vegas, Reno, Tahoe
Rango de transmisión	250m
Modelo de movilidad	Random way point
Modelo de propagación	TwoRayGround
Tipo de antena	Omnidireccional

Nota. Fuente: Kurkowsky, Navidi y Camp (2007)

El escenario utilizado consiste en establecer conexiones desde un nodo cliente hacia varios nodos servidores, el cliente recibirá paquetes desde un servicio FTP ofrecido por cada uno de los servidores, como se muestra en la Figura 14.

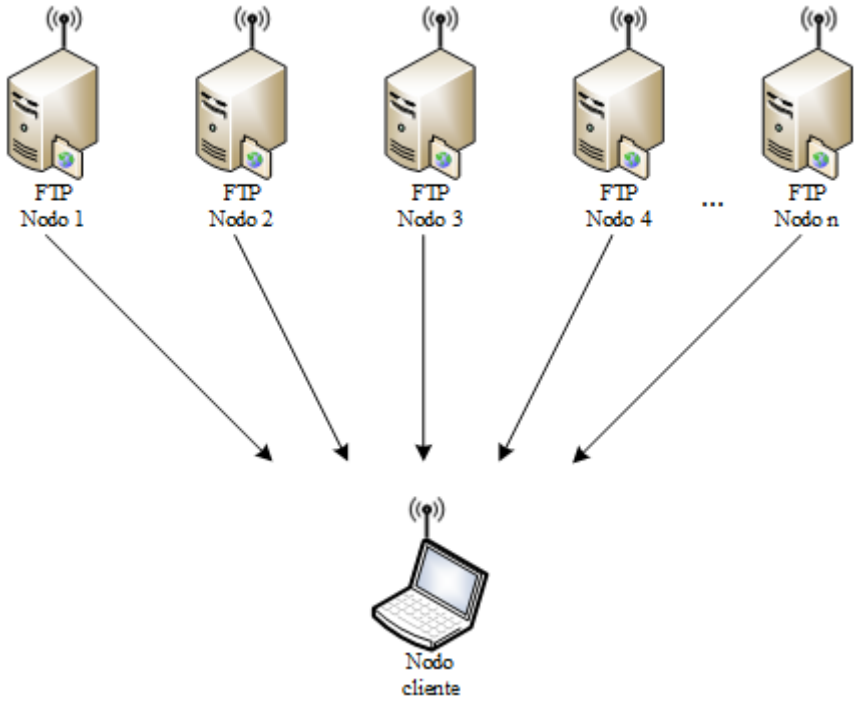

Figura 14 Escenario general para simulación en NS2

Se ha considerado realizar simulaciones para diferentes densidades de nodos, esto es: para 30, 60 y 90 nodos, para cada variante TCP simulada: vegas, reno y tahoe.

3.2. Simulación de la red

La simulación propuesta involucra una serie de actividades que son ejecutadas utilizando las herramientas mencionadas en el capítulo 2 (TCL, NS2, NAM, XGRAPH, AWK). Para obtener los resultados esperados se realizó la secuencia de actividades mostrados en la Figura 15:

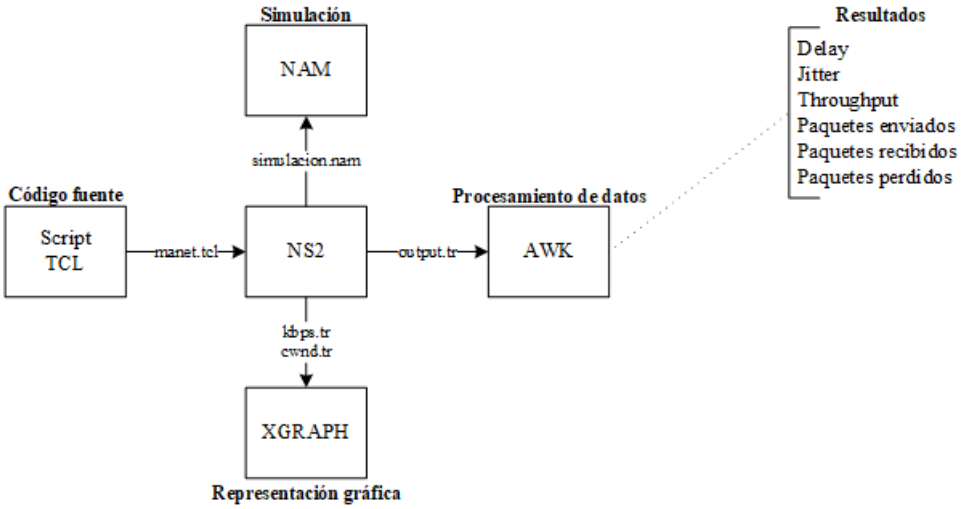

Figura 15 Secuencia de actividades para la simulación

A continuación, se detalla cada actividad:

- Estructuración de código fuente: para ello se utiliza el lenguaje TCL, aquí se obtiene el archivo manet.tcl.

  La Figura 16, muestra la definición de parámetros referentes a la capa física, entre los principales: método de propagación, tipo de antena, estándar 802.11.

```
#-----------------------------------------------
# Definición capa física
#-----------------------------------------------

set val(chan)           Channel/WirelessChannel
set val(prop)           Propagation/TwoRayGround
set val(ant)            Antenna/OmniAntenna
set val(ll)             LL
set val(ifq)            Queue/DropTail/PriQueue
set val(ifqlen)         100
set val(netif)          Phy/WirelessPhy
set val(mac)            Mac/802_11
```

Figura 16 Código fuente, definición de la capa física

La Figura 17, muestra la definición del escenario de simulación, en cuanto a cantidad de nodos, protocolo de enrutamiento (AODV), dimensión del área de simulación (300m x 600m).

```
#---------------------------------------
# Definición de escenario de simulación
#---------------------------------------
set val(nn)             30
set val(rp)               AODV
set val(x)              300
set val(y)              600
set val(energymodel)      EnergyModel;
set val(n_ch)           chan_1
```

Figura 17 Código fuente, definición de parámetros del escenario de simulación

En la Figura 18, se puede ver la creación del área de simulación, así como, la definición de archivos a utilizar durante la simulación para almacenar los resultados (output.tr, cwnd.tr, kbps.tr y simulacion.nam).

```
#-----------------------------------------------------
#  Definición de objetos de simulación
#-----------------------------------------------------
set ns [new Simulator]
set tracefd [open output.tr w]
set cwnd_f [open cwnd.tr w]
set kbps_f [open kbps.tr w]
$ns trace-all $tracefd
$ns use-newtrace
set namtrace [open simulacion.nam w]
$ns namtrace-all-wireless $namtrace $val(x) $val(y)
set topo [new Topography]
$topo load_flatgrid $val(x) $val(y)
create-god $val(nn)
set chan_1 [new $val(chan)]
```

Figura 18 Código fuente, definición de objetos de simulación

En la Figura 19, se detalla el código para la creación de n nodos y la parametrización para la ubicación inicial de los mismos. Se muestra el código para los tres primeros nodos, el código completo se puede ver en el anexo 1.

```
#-----------------------------------------
#Definición de nodos
#-----------------------------------------
for {set i 0} {$i < $val(nn)} { incr i } {
        set n($i) [$ns node]
}

$n(0) color red
$ns at 0.0 "$n(0) color red"

$n(0) set X_ 198
$n(0) set Y_ 346
$n(0) set Z_ 0

$n(1) set X_ 83
$n(1) set Y_ 572
$n(1) set Z_ 0

$n(2) set X_ 59
$n(2) set Y_ 503
$n(2) set Z_ 0
```

Figura 19 Código fuente, creación de nodos

La Figura 20, detalla el código utilizado para la generación de movimientos de los nodos hacia una nueva posición, a una velocidad de 3m/s. Se muestra los movimientos para los diez primeros nodos, el código completo se puede ver en el anexo 1.

```
#-----------------------------------------
#Generación de movimientos
#-----------------------------------------

$ns at 0.2 "$n(0) setdest 223.68 273.38 3.00"
$ns at 0.2 "$n(1) setdest 241.49 188.66 3.00"
$ns at 0.2 "$n(2) setdest 243.13 30.77 3.00"
$ns at 0.2 "$n(3) setdest 274.48 74.78 3.00"
$ns at 0.2 "$n(4) setdest 194.04 16.40 3.00"
$ns at 0.2 "$n(5) setdest 84.62 495.36 3.00"
$ns at 0.2 "$n(6) setdest 243.87 241.01 3.00"
$ns at 0.2 "$n(7) setdest 40.66 412.50 3.00"
$ns at 0.2 "$n(8) setdest 232.99 485.23 3.00"
$ns at 0.2 "$n(9) setdest 3.65 442.87 3.00"
```

Figura 20 Código fuente, generación de movimientos para los nodos

Seguidamente en la Figura 21 se presenta el código para la generación de tráfico TCP tipo vegas, a través de FTP, con paquetes de 3000 bytes y utilizando un tamaño de congestión de 20.

```
#-----------------------------------------
#Genearión de tráfico VEGAS
#-----------------------------------------

for {set i 1} {$i < $val(nn)} {incr i} {

set tcp [new Agent/TCP/Vegas]
$tcp set packetSize 3000
$tcp set window_ 20

set ftp [new Application/FTP]
$ftp attach-agent $tcp

set sink [new Agent/TCPSink]
$ns attach-agent $n($i) $tcp
$ns attach-agent $n(0) $sink

$ns connect $tcp $sink

$ns at 0.1 "$ftp start"

}
```

Figura 21 Código fuente, generación de tráfico vegas con FTP

Para capturar los valores que va tomando la ventana de congestión durante la simulación se utiliza el código mostrado en la Figura 22.

```
#-------------------------------------------------------------
#Procedimiento para capturar la variable de congestión cwnd
#-------------------------------------------------------------
proc plotWindow {tcpSource file} {
     global ns
     set time 0.3
     set now [$ns now]
     set cwnd [$tcpSource set cwnd_]
     set wnd [$tcpSource set window_]
       puts $file "$now $cwnd"
     $ns at [expr $now+$time] "plotWindow $tcpSource $file"
}
```

Figura 22 Código fuente, captura de la variable de congestión CWND

De igual forma se captura el rendimiento en términos de kbps con el código mostrado en la Figura 23.

```
#-------------------------------------------------------
#Procedimiento para almacenar el consumo (kbps)
#-------------------------------------------------------

proc record {} {
        global sink kbps_f
        set ns [Simulator instance]
        set time 0.5
        set now [$ns now]
        set bw0 [$sink set bytes_]
        puts $kbps_f "$now [expr $bw0/$time*8/1000]"
        $sink set bytes_ 0
        $ns at [expr $now+$time] "record"
}
```

Figura 23 Código fuente, registrar el rendimiento en kbps

El código fuente completo, se encuentra en el anexo 1.

- Generación de archivos de simulación: utilizando NS2 se ejecuta el archivo manet.tcl, con el siguiente comando: ns manet.tcl; obteniendo los siguientes resultados:

- Archivo de simulación (simulacion.nam).

simulacion.nam ×

```
n -t * -s 0  -x 198 -y 346 -Z 0 -z 40  -v circle -c green
n -t * -s 1  -x 83 -y 572 -Z 0 -z 40  -v circle -c green
n -t * -s 2  -x 59 -y 503 -Z 0 -z 40  -v circle -c green
n -t * -s 3  -x 214 -y 429 -Z 0 -z 40  -v circle -c green
n -t * -s 4  -x 250 -y 272 -Z 0 -z 40  -v circle -c green
n -t * -s 5  -x 230 -y 77 -Z 0 -z 40  -v circle -c green
```

Figura 24 Resultado utilizado por NAM para generar la topología de red

- Archivo de transferencia de datos (kbps.tr).

kbps.tr ×

```
0 0.0
0.5 0.0
1 0.0
1.5 32.0
2 48.0
2.5 0.0
3 48.0
3.5 0.0
4 96.0
4.5 0.0
```

Figura 25 Archivo de registro de rendimiento. Tiempo(s) y kbps

- Archivo de variación de la ventana de congestión (cwnd.tr).

cwnd.tr ×

```
0.29999999999999999 1
0.59999999999999998 1
0.89999999999999991 1
1.2 1
1.5 1
1.8 3
2.1000000000000001 3
2.3999999999999999 3
2.6999999999999997 3
2.9999999999999996 3
3.2999999999999994 3
3.5999999999999992 6
3.899999999999999 6
4.1999999999999993 6
```

Figura 26 Archivo de registro de la ventan de congestión CWND

- Archivo de trazas (output.tr).

output.tr ×

```
s -t 0.100000000 -Hs 1 -Hd -2 -Ni 1 -Nx 83.00 -Ny 572.00 -Nz 0.00 -Ne
90.000000 -Nl AGT -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 1.0 -Id 0.0 -It
tcp -Il 1000 -If 0 -Ii 0 -Iv 32 -Pn tcp -Ps 0 -Pa 0 -Pf 0 -Po 0
r -t 0.100000000 -Hs 1 -Hd -2 -Ni 1 -Nx 83.00 -Ny 572.00 -Nz 0.00 -Ne
90.000000 -Nl RTR -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 1.0 -Id 0.0 -It
tcp -Il 1000 -If 0 -Ii 0 -Iv 32 -Pn tcp -Ps 0 -Pa 0 -Pf 0 -Po 0
s -t 0.100000000 -Hs 2 -Hd -2 -Ni 2 -Nx 59.00 -Ny 503.00 -Nz 0.00 -Ne
90.000000 -Nl AGT -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 2.0 -Id 0.1 -It
tcp -Il 1000 -If 0 -Ii 1 -Iv 32 -Pn tcp -Ps 0 -Pa 0 -Pf 0 -Po 0
```

Figura 27 Archivo de trazas, en el que se registra la actividad de cada nodo

- Simulación: utilizando NAM se ejecuta la simulación leyendo el contenido del archivo simulacion.nam.

```
exec nam simulacion.nam &
```

Figura 28 Comando para ejecución de la simulación

La instrucción de la Figura 28 se ejecuta desde NS2, al momento de ejecutar el script TCL (manet.tcl) y presenta como resultado el escenario definido en el código fuente. Figura 29.

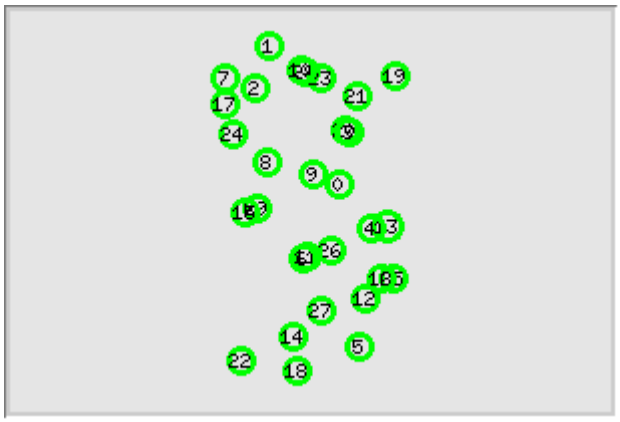

Figura 29 Escenario simulado en NS2 y animado por NAM

- Representación gráfica: mediante xgraph, se crean gráficas de la transmisión de datos en kbps (kbps.tr), con la siguiente línea de código:

xgraph Kbps-Vegas-30Nodos.tr Kbps-Reno-30Nodos.tr Kbps-Tahoe-30Nodos.tr -lw 2 -geometry 600x300 -0 Vegas_30 -1 Reno_30 -2 Tahoe_30 -t "THROUGHPUT" -nb -y BW -x t

Como resultado se obtiene la representación gráfica de los resultados, como se muestra en la Figura 30

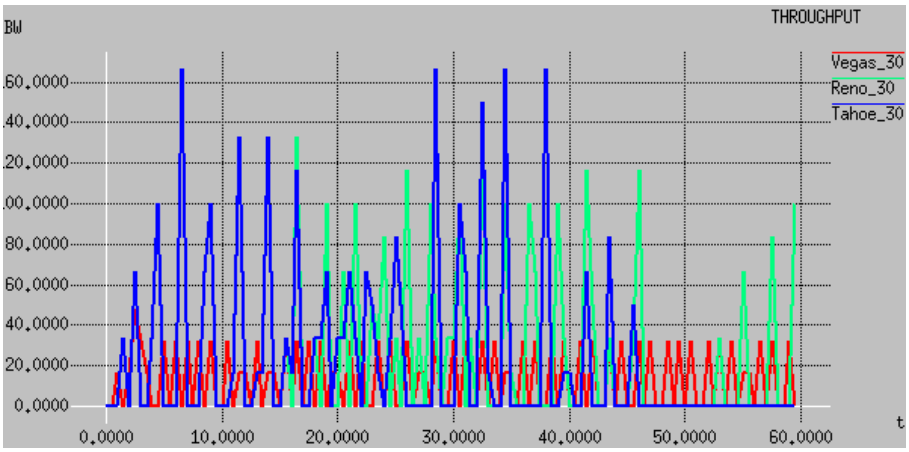

Figura 30 Representación gráfica de rendimiento, realizada con Xgraph

De igual manera se procede con la variación del tamaño de la ventana de congestión (cwnd.tr), con la siguiente línea de código:

xgraph cwnd-Vegas-30Nodos.tr cwnd-Reno-30Nodos.tr cwnd-Tahoe-30Nodos.tr -lw 2 -geometry 600x300 -0 Vegas_30 -1 Reno_30 -2 Tahoe_30 -t "CONGESTION" -nb -y Wnd -x t

- Procesamiento de datos: finalmente se procesa el archivo de trazas (output.tr), para obtener los resultados de la simulación referentes a los parámetros de medición, como son: delay, jitter, throughput, paquetes enviados, paquetes recibidos y paquetes perdidos. El procesamiento de datos se lo realiza con la ayuda de AWK, utilizando la siguiente línea de comandos:

  gawk -f statistics.awk output.tr

La cual recorre el archivo output.tr que resultó de la ejecución del script TCL.

3.2.1. Pruebas en escenario 1, con 30 nodos

La Figura 31, muestra el resultado de la simulación en NAM con un total de 30 nodos móviles, que se desplazan aleatoriamente a lo largo y ancho del área de simulación definida.

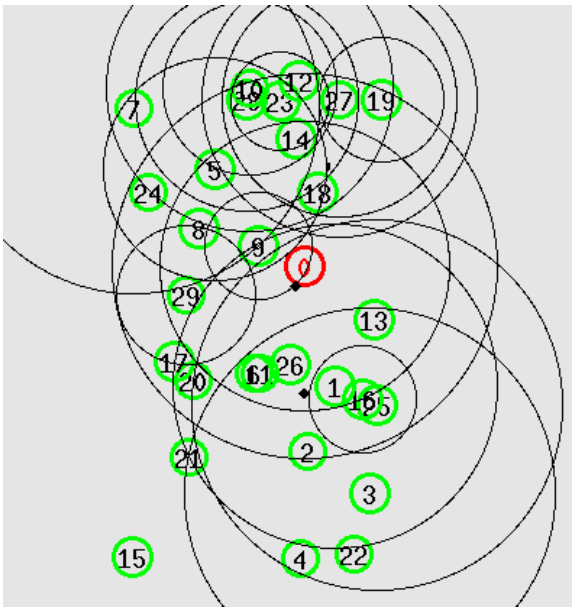

Figura 31 Simulación de escenario con 30 nodos

A continuación, en la Figura 32 se muestra el rendimiento (kbps) alcanzado por cada variante de TCP (vegas, reno y tahoe), en relación al tiempo de simulación.

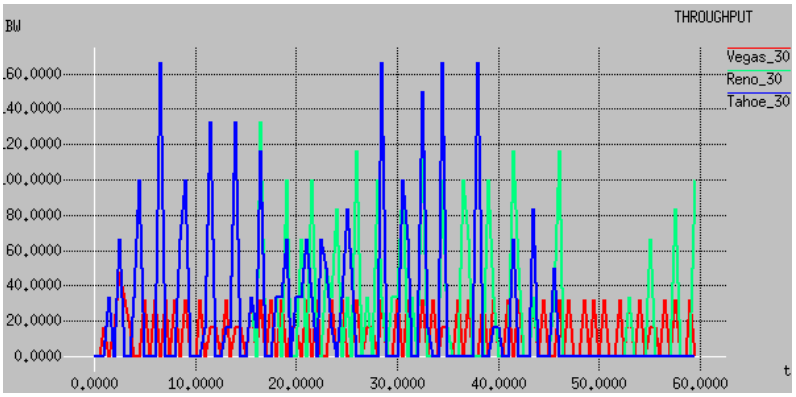

Figura 32 Rendimiento (kbps) para 30 nodos

De acuerdo a la Figura 32, reno y tahoe presentan mejor rendimiento que vegas; en promedio reno alcanza un rendimiento de 35% y 47,47% mayor que tahoe y vegas respectivamente.

A continuación, en la Figura 33 se muestra el comportamiento de la ventana de control de la congestión (CWND), por cada variante de TCP (vegas, reno y tahoe), en relación al tiempo de simulación.

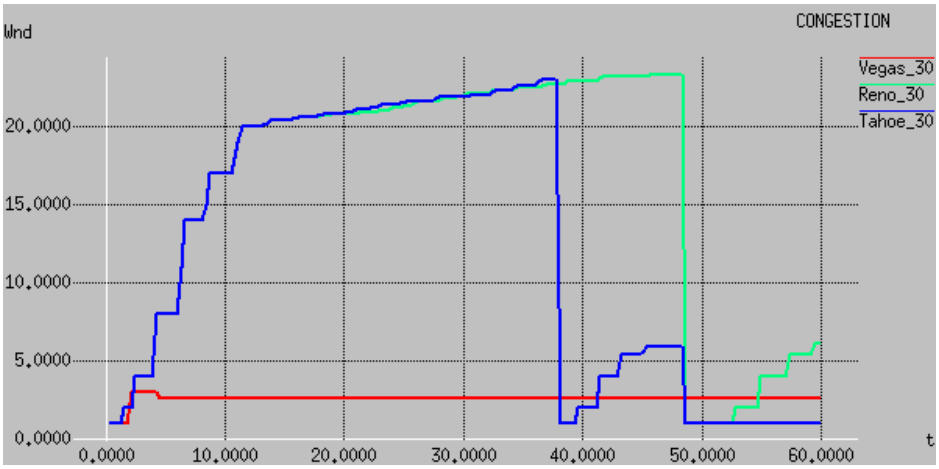

Figura 33 Comportamiento de la ventana de congestión con 30 nodos

En la Figura 33, reno presenta un mejor control de congestión, en t=50s ocurre un timeout, mientras que para tahoe ocurre en t=40s. Si bien, vegas se mantiene con un tamaño de ventana estable, ésta es baja (menor a 4) con respecto al tamaño de la ventana de congestión que es igual a 20, lo que no le permite alcanzar una mayor taza de transferencia, de ahí que en la gráfica 3-20 es la variante TCP con menor rendimiento.

3.2.2. Pruebas en escenario 2, con 60 nodos

Así mismo la Figura 34, muestra el resultado de la simulación para 60 nodos móviles, que se desplazan aleatoriamente a lo largo y ancho del área de simulación definida.

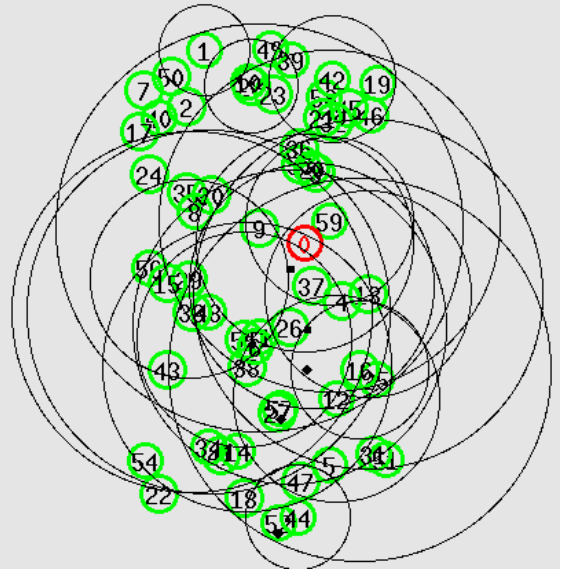

Figura 34 Simulación de escenario con 60 nodos

A continuación, en la Figura 35 se muestra el rendimiento alcanzado por cada variante de TCP (vegas, reno y tahoe), en relación al tiempo de simulación.

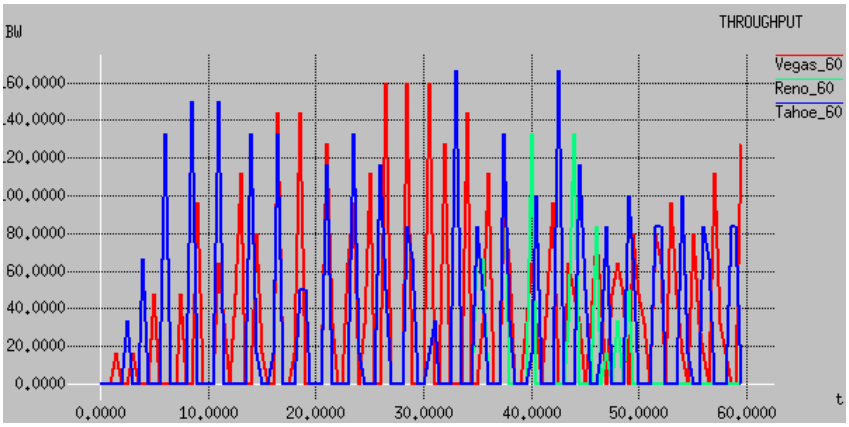

Figura 35 Rendimiento (kbps) para 60 nodos

La Figura 35, evidencia que vegas alcanza en promedio un rendimiento de 30,87% y 49% mayor que tahoe y reno respectivamente.

A continuación, en la Figura 36 se muestra el comportamiento de la ventana de control de la congestión (CWND), por cada variante de TCP (vegas, reno y tahoe), en relación al tiempo de simulación.

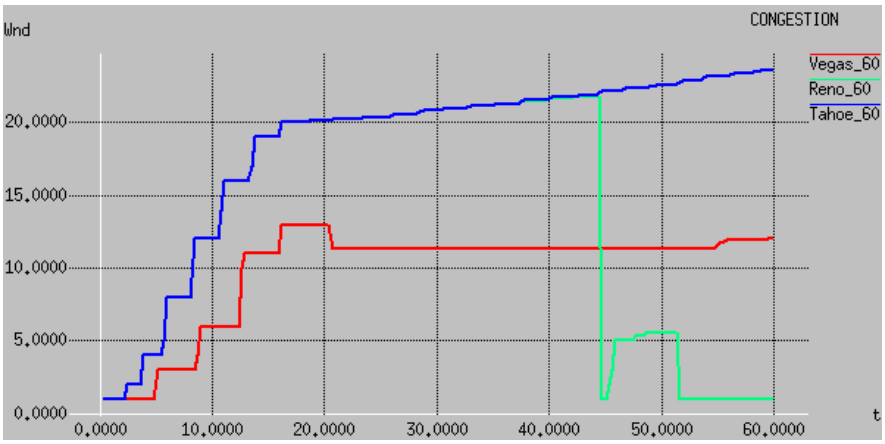

Figura 36 Comportamiento de la ventana de congestión con 60 nodos

Para el escenario de 60 nodos, reno y tahoe llegan a un estado de congestión en t=16s aproximadamente, sin embargo, vegas se mantiene estable con un tamaño de ventana de 11 aproximadamente durante gran parte de la simulación, lo que le permite tener un mejor rendimiento para este escenario.

3.2.3. Pruebas en escenario 3, con 90 nodos

El escenario con 90 nodos se muestra en la Figura 37

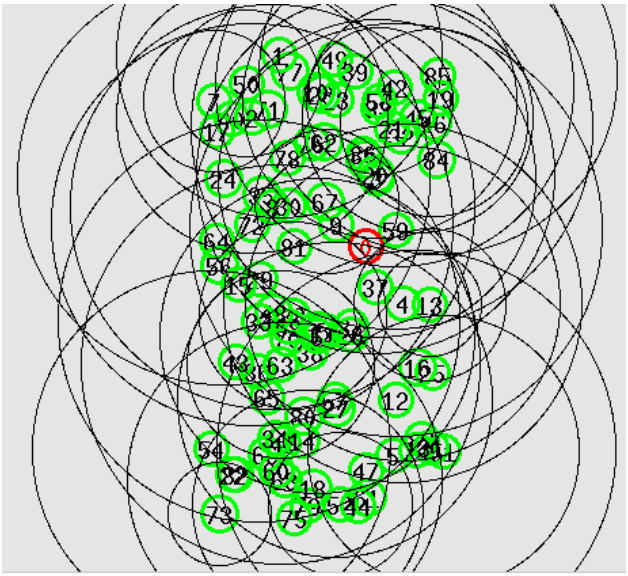

Figura 37 Simulación de escenario con 90 nodos

A continuación, en la Figura 38 se muestra el rendimiento alcanzado por cada variante de TCP (vegas, reno y tahoe), en relación al tiempo de simulación.

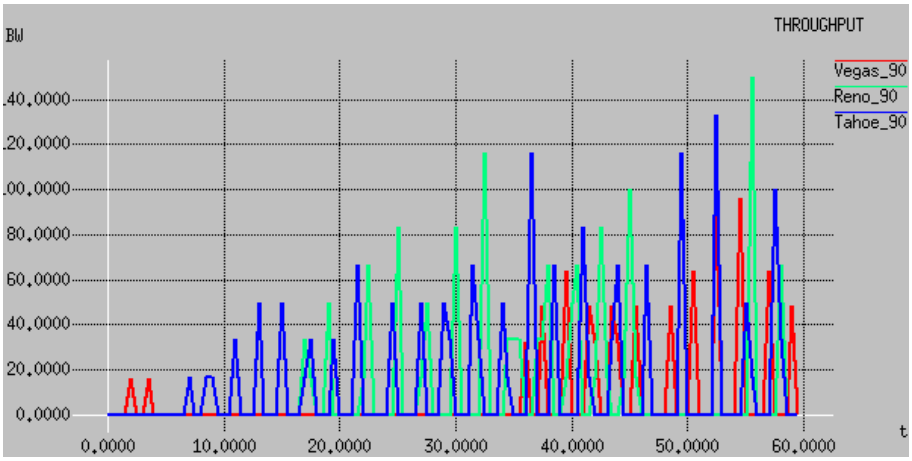

Figura 38 Rendimiento (kbps) para 90 nodos

Según la Figura 38, tahoe presenta mejor rendimiento que reno y vegas; en promedio tahoe alcanza un rendimiento de 11% y 49% mayor que reno y vegas respectivamente.

A continuación, en la Figura 39 se muestra el comportamiento de la ventana de control de la congestión (CWND), por cada variante de TCP (vegas, reno y tahoe), en relación al tiempo de simulación.

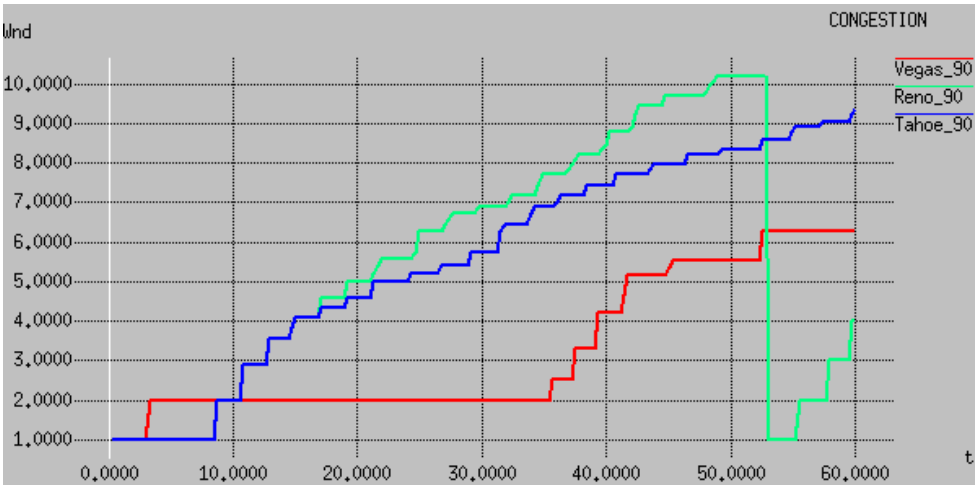

Figura 39 Comportamiento de la ventana de congestión con 90 nodos

En la Figura 39, Tahoe evoluciona linealmente sin llegar a congestionarse, a diferencia de reno que, aunque no alcanza la congestión, presenta timeout en t=53s, esto hace que tahoe tenga mayor rendimiento con respecto a vegas y reno.

CAPÍTULO 4

4. ANÁLISIS DE RESULTADOS

4.1. Métricas o indicadores

Como resultado de la simulación de la red móvil ad hoc en NS2, se obtuvo el archivo de trazas, el cual contiene información detallada de la actividad realizada por cada nodo en la red.

Considerando que, para cada escenario de 30, 60 y 90 nodos se genera un archivo de traza, a continuación, se muestran las tres primeras líneas del archivo de trazas del escenario con 30 nodos, denominado output.tr:

Línea 1:

s -t 0.100000000 -Hs 1 -Hd -2 -Ni 1 -Nx 83.00 -Ny 572.00 -Nz 0.00 -Ne 90.000000 -Nl AGT -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 1.0 -Id 0.0 -It tcp -Il 1000 -If 0 -Ii 0 -Iv 32 -Pn tcp -Ps 0 -Pa 0 -Pf 0 -Po 0

Línea 2:

r -t 0.100000000 -Hs 1 -Hd -2 -Ni 1 -Nx 83.00 -Ny 572.00 -Nz 0.00 -Ne 90.000000 -Nl RTR -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 1.0 -Id 0.0 -It tcp -Il 1000 -If 0 -Ii 0 -Iv 32 -Pn tcp -Ps 0 -Pa 0 -Pf 0 -Po 0

Línea 3:

s -t 0.100000000 -Hs 2 -Hd -2 -Ni 2 -Nx 59.00 -Ny 503.00 -Nz 0.00 -Ne 90.000000 -Nl AGT -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 2.0 -Id 0.1 -It tcp -Il 1000 -If 0 -Ii 1 -Iv 32 -Pn tcp -Ps 0 -Pa 0 -Pf 0 -Po 0

Cada elemento de la traza separado por espacios constituye un parámetro de la comunicación entre un nodo emisor y un nodo receptor, estos elementos son procesados por AWK para obtener los resultados en términos de rendimiento, retardo, tasa de envío y de recepción de paquetes, pérdida de paquetes, latencia y variación de latencia (jitter).

A continuación, se analizan los resultados de la simulación.

4.1.1. Rendimiento

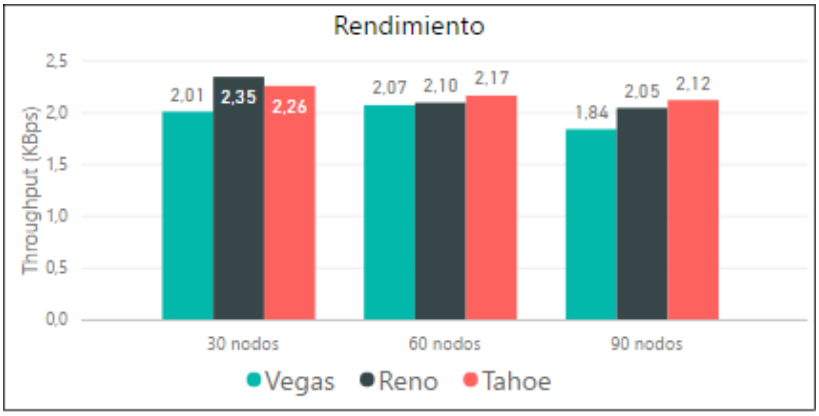

Figura 40 Rendimiento con vegas, reno y tahoe para 30, 60 y 90 nodos

En la Figura 40, se puede apreciar que para el escenario propuesto de 30 nodos reno funciona un 3,75% mejor que tahoe y un 14,32% mejor que vegas. Para los escenarios de 60 y 90 nodos tahoe presenta mejor rendimiento que vegas y reno. 3,14% mejor que reno y 4,25% mejor que vegas para el escenario de 60 nodos. Y para el escenario de 90 nodos, 3,56% más eficiente que reno y 13,22% más eficiente que vegas.

4.1.2. Tasa de envío de paquetes

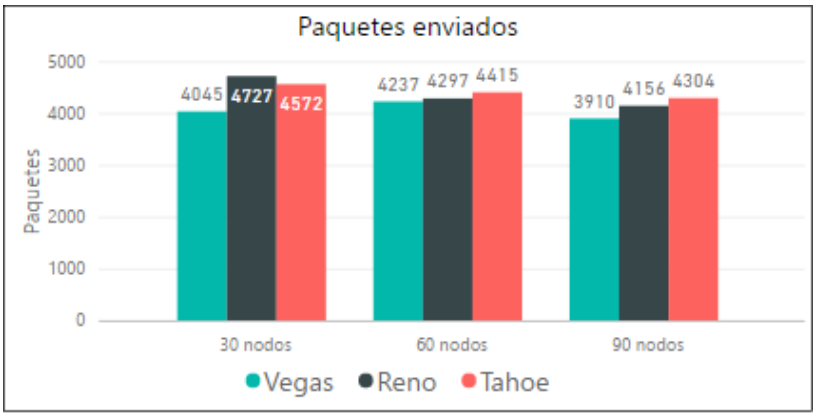

Figura 41 Paquetes enviados con vegas, reno y tahoe para 30, 60 y 90 nodos

Para el caso del escenario de 30 nodos reno realiza el mejor envío de paquetes, mientras que para los escenarios de 60 y 90 nodos, tahoe genera más paquetes, como se muestra en la Figura 41.

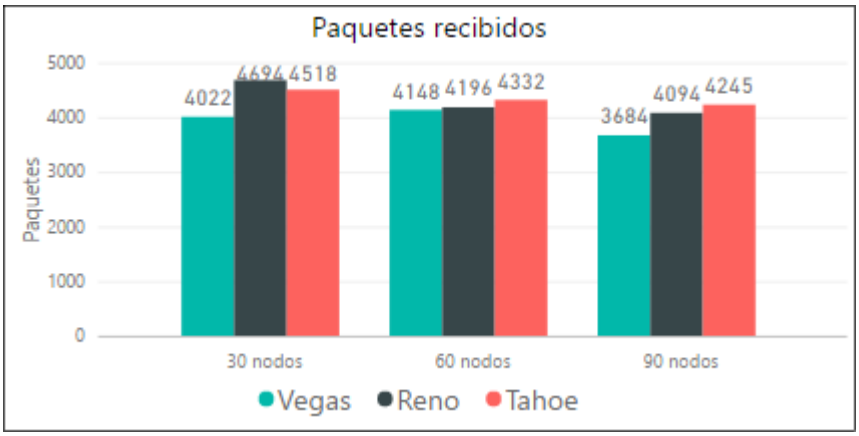

Figura 42 Paquetes recibidos con vegas, reno y tahoe para 30, 60 y 90 nodos.

De igual manera, la recepción de paquetes es mejor con reno para el caso de 30 nodos, y para escenarios de 60 y 90 nodos, tahoe logra mayor recepción de paquetes. Figura 42.

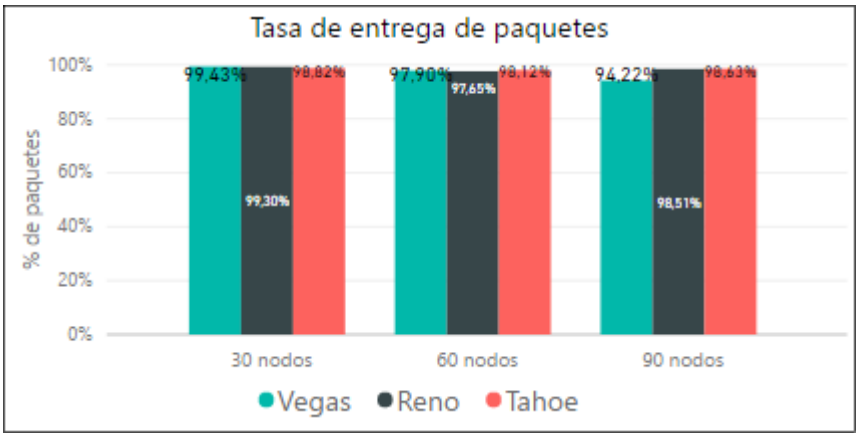

Figura 43 Tasa de entrega de paquetes con vegas, reno y tahoe para 30, 60 y 90 nodos

En cuanto a la tasa de entrega de paquetes, el resultado guarda relación con los obtenidos en envío y recepción de paquetes. Figura 43.

4.1.3. Pérdida de paquetes

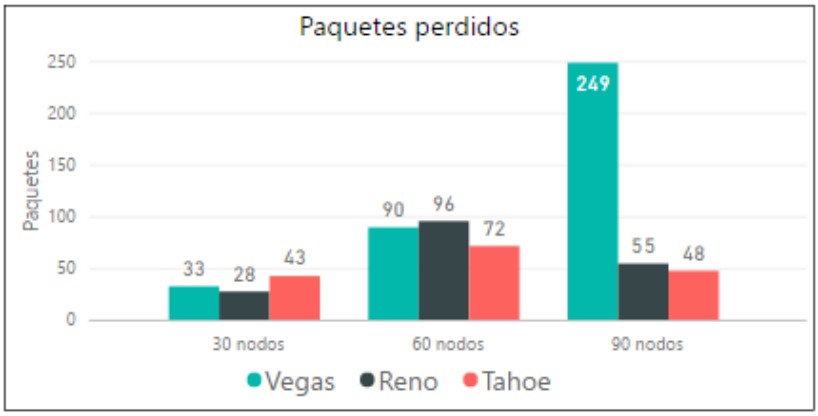

Figura 44 Paquetes perdidos con vegas, reno y tahoe para 30, 60 y 90 nodos

De acuerdo a la Figura 44, para el escenario de 30 nodos reno presenta menor pérdida de paquetes, y para los escenarios de 60 y 90 nodos, tahoe es la variante con menor pérdida de paquetes.

4.1.4. Retardo

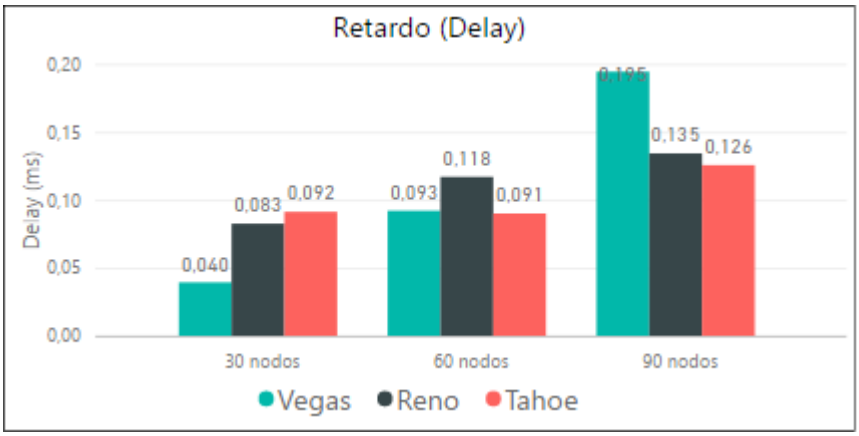

Figura 45 Delay con vegas, reno y tahoe para 30, 60 y 90 nodos

Para los casos de 30 y 60 nodos vegas presenta menor retardo, y para 90 nodos tahoe presenta menor retardo.

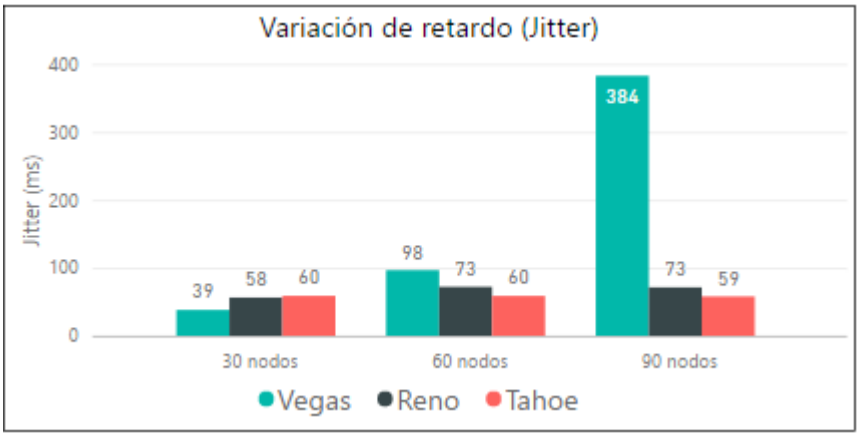

Figura 46 Jitter con vegas, reno y tahoe para 30, 60 y 90 nodos

Si bien en la Figura 45, para el escenario de 60 nodos vegas presentaba menor retardo, al momento de comparar jitter se puede verificar que vegas presenta el mayor valor para dicho parámetro. Para 30 nodos vegas sigue siendo menor y para 90 nodos tahoe es el más bajo.

CAPÍTULO 5

5. CONCLUSIONES Y RECOMENDACIONES

5.1. Conclusiones

La evolución tecnológica y sobre todo en el campo de las redes de comunicaciones ha permitido que más y más dispositivos se conecten a la red ya sea por medios guiados o por medios inalámbricos, permitiendo el despliegue de redes hacia sitios en donde hace pocos años era impensable que se pudiera ofrecer servicios de red; en este contexto las redes móviles Ad hoc se han vuelto indispensables para la implementación de servicios que requieren características propias de este tipo de redes como su autonomía, su capacidad de establecer topologías dinámicas en la medida en que más dispositivos acceden a la red, la no necesidad de una infraestructura física previamente establecida; a pesar de las bondades que presentan estas redes se debe tener presente que tienen limitantes, principalmente en cuanto a energía y capacidad de procesamiento. Estas limitantes son la razón para que los expertos diseñen algoritmos y mecanismos que contribuyan a la optimización de recursos de los dispositivos que operan en las redes móviles ad hoc.

Los recursos de red limitados, así como la creciente demanda de acceso a la red y transmisión de datos en esta, ha conllevado a que los usuarios experimenten numerosos problemas al momento de utilizar un determinado servicio de red, principalmente cuando un canal de datos alcanza su máxima capacidad y comienza a rechazar conexiones o a descartar paquetes, porque ha entrado en un

estado de congestión; debido a esto surge la necesidad de contar con mecanismos para el control de la congestión, los cuales constituyen un elemento fundamental para mantener parámetros aceptables de disponibilidad en un servicio de red, así como para garantizar la entrega confiable de datos en los tiempos requeridos. A lo largo de la historia se han implementado varios mecanismos para el control de la congestión en TCP, y como uno de los objetivos del presente proyecto se logró identificar los principales mecanismos utilizados por las variantes de TCP, así como evaluar su comportamiento frente a una situación de congestión.

Con los conocimientos adquiridos y mediante el uso del simulador NS2 se logró diseñar y simular redes móviles ad hoc en distintos escenarios, concluyendo que es de vital importancia realizar una definición apropiada de los parámetros que intervienen en dicha simulación, ya que de ello depende la obtención de resultados relevantes que permitan tomar una determinada decisión.

La simulación se realizó para tres variantes de TCP: vegas, reno y tahoe, en tres distintos escenarios que varían por la densidad de nodos que intervienen en la red, esto es: 30, 60 y 90 nodos. Para el escenario propuesto de 30 nodos reno funciona un 3,75% mejor que tahoe y un 14,32% mejor que vegas. Para los escenarios de 60 y 90 nodos, tahoe presenta mejor rendimiento que vegas y reno. 3,14% mejor que reno y 4,25% mejor que vegas para el escenario de 60 nodos. Y para el escenario de 90 nodos, 3,56% más eficiente que reno y 13,22% más que vegas. Como resultado de la simulación se concluye que, para los escenarios planteados juntos con sus parámetros, la variante tahoe funciona de mejor manera

sobre todo en los escenarios de mayor densidad de nodos, en términos de rendimiento y tasas de pérdida y entrega de paquetes.

Comparar el comportamiento de las variantes de TCP en los diferentes escenarios requiere del análisis de la variable denominada ventana de congestión, que para la simulación propuesta se ha establecido un tamaño de ventana de 20 paquetes. Con esto la variante vegas presenta problemas al inicio de la simulación en todos los escenarios ya que no inicia el envío de paquetes, por lo que se concluye que vegas no trabaja adecuadamente con el mecanismo de arranque lento o slow start. Por otro lado, la variante tahoe presenta un mejor arranque lento y crecimiento lineal de la ventana de congestión más aún para los escenarios de 60 y 90 nodos. Y finalmente, reno en todos los escenarios alcanza el nivel de congestión, estableciéndose un timeoute que le obliga a reiniciar el envío de paquetes mediante el mecanismo de arranque lento. Por lo tanto, se concluye que, para los escenarios planteados junto con sus parámetros, la variante tahoe presenta un mejor comportamiento frente a una situación de congestión.

De manera general se concluye que, se ha cumplido con los objetivos planteados para el presente proyecto.

5.2 Recomendaciones

Para el desarrollo de una simulación se recomienda definir correctamente el escenario y los parámetros de red a simular, basados en un estándar o marco de referencia, de tal forma que los resultados obtenidos guarden un mínimo margen de error y estén más cercanos a la realidad.

Así mismo, es importante la elección del software de simulación, si bien NS2 integra varias herramientas para la simulación, representación gráfica y análisis de resultados, su curva de aprendizaje es alta, por lo que su uso puede resultar complicado; por otro lado, los gráficos obtenidos con Xgraph son básicos y no permiten mayores cambios. Existen otras herramientas que pueden ser utilizadas para temas investigativos como OPNET, con mayores funcionalidades que NS2, sin embargo, hay que considerar el tipo de licencia.

La creciente demanda de servicios de telecomunicaciones y sobre todo de servicios de voz y video que no admiten pérdida de paquetes, en redes con recursos limitados como las redes móviles ad hoc, obliga a mejorar la entrega de paquetes, por lo que es importante continuar la investigación de mecanismos que aporten a minimizar los problemas de conectividad, con énfasis en la congestión.

BIBLIOGRAFÍA

Altman, E., Jiménez, T. (2012), NS Simulator for Beginners. California, Estados Unidos: Morgan & Claypool.

Houmkozlis, C.N., Rovithakis, G.A. (2012). End-to-End Adaptive Congestion Cotrol in TCP/IP Networks. Florida, Estados Unidos: CRC Press, Taylor & Francis Group.

Issariyakul, T., Hossain, E. (2012). Introduction to Network Simulator NS2. Nueva York, Estados Unidos: Springer Science+Business Media.

Kurose, J., Ross, K. (2010). Redes de Computadoras, un enfoque descendente, 5ta edición. Madrid, España: Pearson.

Law, L.K., Krishnamurthy, S.V., Faloutsos, M. (2009). Transport Layer Protocols for Mobile Ad Hoc Networks. En A. Boukerche (Ed.). Algorithms and Protocols for Wireless and Mobile Ad Hoc Networks. Ottawa, Canadá: WILEY.

Liu, J.J., Chlamtac, I. (2004). Mobile Ad Hoc Networking with a View of 4G wireless: Imperatives and Challenges. En S. Basagni., M.Conti., S. Giordano., I. Stojmenovic. (Ed.), Mobile Ad-Hoc Networking. New Jersey, Estados Unidos: IEEE Press.

Morales, M., Calle, M.A., Tovar, J.D., Cuéllar, J.C. (2013). Simulando con OMNET, Selección de la herramienta y su utilización. Cali, Colombia: Universidad ICESI (Litocencoa SAS).

Sarkar, S.K., Basavaraju, T.G., Puttamadappa, C. (2008). Ad Hoc Mobile Wireless: Principles, Protocols and Applications. Boston, Estados Unidos: Auerbanch Publication Taylor & Francis Group.

Bangnan X., Hischke, S., Walke, B. (2003). The Role of Ad Hoc Networking in Future Wireless Communications. International Conference on Communication Technology Proceedings. doi: 10.1109/ICCT.2003.1209779

Brakmo, L.S., O'Malley, S.W., Peterson, L.L. (1994). TCP Vegas: New techniques for Congestion Detection and Avoidance. ACM Computer Communications Review. doi: 10.1145/190809.190317

Kurkowski, S., Navidi, W., and Camp, T. (2007), Constructing MANET Simulation Scenarios That Meet Standards. Pisa, Italia: 2007 IEEE International Conference on Mobile Adhoc and Sensor Systems. doi: 10.1109/MOBHOC.2007.4428640

Kurkowski, S., Navidi, W., and Camp, T. (2007). Discovering Variables that Affect

MANET Protocol Performance. Washingtpon, DC, Estados Unidos: IEEE GLOBECOM 2007 - IEEE Global Telecommunications Conference. doi: 10.1109/GLOCOM.2007.238

Olasoji, B. O., Olanrewaju, O. M., Adebayo, I. O. (2016). Transmission Control Protocol and Congestion Control: A Review of TCP Variants. International Journal of Computer Science and Information Security. doi: 10.6084/m9.figshare.3153904

Oyeyinka, K., Oluwatope, A., Akinwale, A., Folorunso, O., Aderounmu, G., Abiona, O. (2011). TCP Window Based Congestion Control Slow-Start Approach. Communications and Networks. Communications and Network. doi: 10.4236/cn.2011.32011

Chalmaneta, J. (2009) Estudio y análisis de prestaciones de redes móviles Ad Hoc mediante simulaciones NS-2 para validar modelos analíticos (tesis maestría). Universidad Politécnica de Cataluña, Barcelona, España.

Liu, Ch. (2001). Wireless Network Enhancements Using Congestion Coherence, Faster Congestion Feedback, Media Access Control and AAL2 Voice Trunking (Tesis doctoral). The Ohio State University. Ohio, Estados Unidos.

Torres, R.V. (2011). Contribución a los modelos de gestión de las redes móviles Ad Hoc (Tesis doctoral). Universidad Politécnica de Madrid, Madrid, España.

Ait-Hellal, O., & Altman, E. (2000). Analysis of TCP vegas and TCP reno. Telecommunication Systems, 15(3-4), 381-404. Recuperado de https://search.proquest.com/docview/212013675?accountid=45668

Allman, M., Paxon, V., Stevens, W. (1999). TCP Congestion Control. California, Estados Unidos. Internet Engineering Task Force Recuperado de https://tools.ietf.org/html/rfc2581

Ashish N., Jagannadha J., Mansoor M., Vijay S. (2001). TCP for Wireless Network. Dallas, Estados Unidos: Computer Science Program, University of Texas at Dallas. Recuperado de http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.23.5925&rep=rep1&type=pdf

Chung, J., Claypool, M., (n.d.). NS by Example. Worcester Polytechnic Institute, Computer Science. Recuperado de http://nile.wpi.edu/NS/

Floyd, S., Henderson, T., Gurtov, A. (2004) The NewReno, Modification to TCP, Fast Recovery Algorithm. California, Estados Unidos: Internet Engineering Task Force. Recuperado de https://tools.ietf.org/html/rfc3782

General Purpose 2-D Plotter. (2016). XGRAPH. Recuperado de http://www.xgraph.org/

Henderson, T., (2011). The VINT Project. Berkeley, Estados Unidos: UC Berkeley, LBL, USC/ISI, and Xerox PARC. Recuperado de https://www.isi.edu/nsnam/ns/doc/everything.html

Information Sciences Institute, University of Southern California. (1981) Transmission Control Protocol. California, Estados Unidos: Internet Engineering Task Force. Recuperado de https://tools.ietf.org/html/rfc793

López, A.M., García N. (2010). Simulación de tráfico en redes Inalámbricas mediante NS2. Pereira, Colombia: Scientia et technica, Universidad Tecnológica de Pereira. Recuperado de http://www.redalyc.org/articulo.oa?id=84917316028

Macker, J., Corson, S. (2002). Mobile Ad Hoc Network (MANET). Atlanta Gerorgia, Estados Unidos: Internet Engineering Task Force. Recuperado de https://www.ietf.org/proceedings/55/177.htm

Mascolo S., Casetti C., Gerla M., Lee S., Sanadidi M. (2000). TCP Westwood: congestion control with faster recovery. Los Ángeles, California, Estados Unidos: Computer Science Department – UCLA. Recuperado de

http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.21.9029&rep=rep1&type=pdf

Rodríguez, F., Vidal, L., Alves L. (2004). TCP Sobre enlaces Wireless Problemas y algunas posibles soluciones existentes. Universidad de la República. Recuperado de https://www.researchgate.net/profile/Leonardo_Vidal4/publication/237236848_TCP_sobre_enlaces_wireless_Problemas_y_algunas_posibles_soluciones_existentes/links/55886d4608aeb2994444962d/TCP-sobre-enlaces-wireless-Problemas-y-algunas-posibles-soluciones-existentes.pdf

Saini, G. (2017). NS2 Simulator On Ubuntu. Linoob. Recuperado de http://linoob.com/2011/04/ns2-simulator-on-ubuntu/

Sourceforge. (2011). NS-2 User Information. Recuperado de http://nsnam.sourceforge.net/wiki/index.php/User_Information

Stevens, W. (1997). TCP Slow Start, Congestion Avoindance, Fast Retransmit, and Fast Recovery Algorithms. California, Estados Unidos: Internet Engineering Task Force. Recuperado de https://tools.ietf.org/html/rfc2001

The GNU Awk. (2016). The GNU Awk User's Guide. Free Software Foundation Inc. Recuperado de https://www.gnu.org/software/gawk/manual/gawk.html

S-Logix. (n.d.)How to create Mobile Ad Hoc Network (MANET) in ns2. Kodambakkam, Chennai, Tamil Nadu, India. Recuperado de http://slogix.in/how-to-create-mobile-ad-hoc-network-manet-in-ns2

Researching Wirelessly, (n.d). An AWK Script to Print Network Statistics. Recuperado de https://researchingwirelessly.wordpress.com/2013/07/23/awk-script-an-awk-script-to-print-network-statistics/

ANEXOS

ANEXO 1: CÓDIGO FUENTE PARA 30 NODOS CON TRÁFICO TCP VEGAS

```
#---------------------------------------------------------------
# Definicion capa fisica
#---------------------------------------------------------------
set val(chan)         Channel/WirelessChannel
set val(prop)         Propagation/TwoRayGround
set val(ant)          Antenna/OmniAntenna
set val(ll)           LL
set val(ifq)          Queue/DropTail/PriQueue
set val(ifqlen)       100
set val(netif)        Phy/WirelessPhy
set val(mac)          Mac/802_11


#---------------------------------------------------------------
# Definicion de escenario de simulacion
#---------------------------------------------------------------
set val(nn)           30
set val(rp)                 AODV
set val(x)            300
set val(y)            600
set val(energymodel)    EnergyModel;
set val(n_ch)         chan_1


#---------------------------------------------------------------
#  Definicion de objetos de simulacion
#---------------------------------------------------------------
set ns [new Simulator]
set tracefd [open output.tr w]
set cwnd_f [open cwnd.tr w]
set kbps_f [open kbps.tr w]
$ns trace-all $tracefd
$ns use-newtrace
set namtrace [open simulacion.nam w]
$ns namtrace-all-wireless $namtrace $val(x) $val(y)
set topo [new Topography]
$topo load_flatgrid $val(x) $val(y)
create-god $val(nn)
set chan_1 [new $val(chan)]


#Configuracion de nodos
#---------------------------------------------------------------
$ns node-config  -adhocRouting $val(rp) \
                 -llType $val(ll) \
```

```
		-macType $val(mac) \
		-ifqType $val(ifq) \
		-ifqLen $val(ifqlen) \
		-antType $val(ant) \
		-propType $val(prop) \
		-phyType $val(netif) \
		-topoInstance $topo \
		-agentTrace ON \
		-routerTrace ON \
		-macTrace ON \
		-movementTrace OFF \
		-channel $chan_1   \
		-energyModel $val(energymodel) \
		-rxPower 0.3 \
		-txPower 0.6 \
		-initialEnergy 90 \
			-rxPower 0.3 \
		-txPower 0.6


#-----------------------------------------------------------------
#Definicion de nodos
#-----------------------------------------------------------------
for {set i 0} {$i < $val(nn)} { incr i } {
	set n($i) [$ns node]
}
$n(0) color red
$ns at 0.0 "$n(0) color red"


$n(0) set X_ 198
$n(0) set Y_ 346
$n(0) set Z_ 0


$n(1) set X_ 83
$n(1) set Y_ 572
$n(1) set Z_ 0


$n(2) set X_ 59
$n(2) set Y_ 503
$n(2) set Z_ 0


$n(3) set X_ 214
$n(3) set Y_ 429
$n(3) set Z_ 0
```

```
$n(4) set X_ 250
$n(4) set Y_ 272
$n(4) set Z_ 0


$n(5) set X_ 230
$n(5) set Y_ 77
$n(5) set Z_ 0

$n(6) set X_ 137
$n(6) set Y_ 221
$n(6) set Z_ 0

$n(7) set X_ 8
$n(7) set Y_ 521
$n(7) set Z_ 0

$n(8) set X_ 78
$n(8) set Y_ 380
$n(8) set Z_ 0

$n(9) set X_ 154
$n(9) set Y_ 360
$n(9) set Z_ 0

$n(10) set X_ 134
$n(10) set Y_ 534
$n(10) set Z_ 0

$n(11) set X_ 145
$n(11) set Y_ 226
$n(11) set Z_ 0

$n(12) set X_ 241
$n(12) set Y_ 155
$n(12) set Z_ 0

$n(13) set X_ 278
$n(13) set Y_ 276
$n(13) set Z_ 0

$n(14) set X_ 121
$n(14) set Y_ 94
```

```
$n(14) set Z_ 0

$n(15) set X_ 42
$n(15) set Y_ 300
$n(15) set Z_ 0


$n(16) set X_ 268
$n(16) set Y_ 189
$n(16) set Z_ 0


$n(17) set X_ 8
$n(17) set Y_ 477
$n(17) set Z_ 0


$n(18) set X_ 129
$n(18) set Y_ 39
$n(18) set Z_ 0


$n(19) set X_ 291
$n(19) set Y_ 524
$n(19) set Z_ 0


$n(20) set X_ 208
$n(20) set Y_ 434
$n(20) set Z_ 0


$n(21) set X_ 227
$n(21) set Y_ 490
$n(21) set Z_ 0


$n(22) set X_ 35
$n(22) set Y_ 54
$n(22) set Z_ 0


$n(23) set X_ 168
$n(23) set Y_ 520
$n(23) set Z_ 0


$n(24) set X_ 21
$n(24) set Y_ 427
$n(24) set Z_ 0
```

```
$n(25) set X_ 286
$n(25) set Y_ 188
$n(25) set Z_ 0


$n(26) set X_ 185
$n(26) set Y_ 237
$n(26) set Z_ 0


$n(27) set X_ 169
$n(27) set Y_ 138
$n(27) set Z_ 0

$n(28) set X_ 140
$n(28) set Y_ 530
$n(28) set Z_ 0


$n(29) set X_ 63
$n(29) set Y_ 305
$n(29) set Z_ 0


#---------------------------------------------------------------
#Generacion de movimientos aleatorios
#---------------------------------------------------------------
#for {set i 0} {$i < $val(nn)} {incr i} {
#        set xr [expr rand() * $val(x)]
#        set yr [expr rand() * $val(y)]
#        $ns at 0.2 "$n($i) setdest $xr $yr 50"
#}


#---------------------------------------------------------------
#Generacion de movimientos estaticos
#---------------------------------------------------------------
$ns at 0.2 "$n(0) setdest 223.68 273.38 3.00"
$ns at 0.2 "$n(1) setdest 241.49 188.66 3.00"
$ns at 0.2 "$n(2) setdest 243.13 30.77 3.00"
$ns at 0.2 "$n(3) setdest 274.48 74.78 3.00"
$ns at 0.2 "$n(4) setdest 194.04 16.40 3.00"
$ns at 0.2 "$n(5) setdest 84.62 495.36 3.00"
$ns at 0.2 "$n(6) setdest 243.87 241.01 3.00"
$ns at 0.2 "$n(7) setdest 40.66 412.50 3.00"
$ns at 0.2 "$n(8) setdest 232.99 485.23 3.00"
$ns at 0.2 "$n(9) setdest 3.65 442.87 3.00"
$ns at 0.2 "$n(10) setdest 160.88 564.97 3.00"
$ns at 0.2 "$n(11) setdest 263.95 105.51 3.00"
$ns at 0.2 "$n(12) setdest 190.44 568.72 3.00"
$ns at 0.2 "$n(13) setdest 237.04 473.97 3.00"
$ns at 0.2 "$n(14) setdest 202.84 552.13 3.00"
```

```
$ns at 0.2 "$n(15) setdest 9.34 18.59 3.00"
$ns at 0.2 "$n(16) setdest 217.54 242.58 3.00"
$ns at 0.2 "$n(17) setdest 55.88 234.68 3.00"
$ns at 0.2 "$n(18) setdest 253.14 596.13 3.00"
$ns at 0.2 "$n(19) setdest 142.05 573.77 3.00"
$ns at 0.2 "$n(20) setdest 75.81 214.95 3.00"
$ns at 0.2 "$n(21) setdest 17.60 6.03 3.00"
$ns at 0.2 "$n(22) setdest 253.48 22.13 3.00"
$ns at 0.2 "$n(23) setdest 270.80 589.92 3.00"
$ns at 0.2 "$n(24) setdest 178.77 310.68 3.00"
$ns at 0.2 "$n(25) setdest 204.51 172.48 3.00"
$ns at 0.2 "$n(26) setdest 144.67 43.12 3.00"
$ns at 0.2 "$n(27) setdest 248.54 585.72 3.00"
$ns at 0.2 "$n(28) setdest 16.83 398.99 3.00"
$ns at 0.2 "$n(29) setdest 152.72 380.87 3.00"
for {set i 0} {$i < $val(nn)} { incr i } {
        $ns initial_node_pos $n($i) 40
}


#---------------------------------------------------------------
#Genearion de trafico VEGAS
#---------------------------------------------------------------
for {set i 1} {$i < $val(nn)} {incr i} {

        set tcp [new Agent/TCP/Vegas]
        $tcp set packetSize 3000
        $tcp set window_ 20


        set ftp [new Application/FTP]
        $ftp attach-agent $tcp


        set sink [new Agent/TCPSink]
        $ns attach-agent $n($i) $tcp
        $ns attach-agent $n(0) $sink
        $ns connect $tcp $sink
        $ns at 0.1 "$ftp start"

}


#---------------------------------------------------------------
#Procedimiento para capturar la variable de congestion cwnd
#---------------------------------------------------------------
proc plotWindow {tcpSource file} {
    global ns
    set time 0.3
    set now [$ns now]
    set cwnd [$tcpSource set cwnd_]
    set wnd [$tcpSource set window_]
```

```
#     if { $cwnd < $wnd } {
       puts $file "$now $cwnd"
#        } else {
#        puts $file "$now $wnd"
#        }
    $ns at [expr $now+$time] "plotWindow $tcpSource $file"
  }

$ns at 0.3 "plotWindow $tcp $cwnd_f"

#---------------------------------------------------------------
#Procedimiento para almacenar el consumo (kbps)
#---------------------------------------------------------------
proc record {} {
          global sink kbps_f
          set ns [Simulator instance]
          set time 0.5
          set now [$ns now]
          set bw0 [$sink set bytes_]
          puts $kbps_f "$now [expr $bw0/$time*8/1000]"
          $sink set bytes_ 0
       $ns at [expr $now+$time] "record"
}

#Procedimiento finish
#---------------------------------------------------------------
proc finish {} {
       global ns kbps_f tracefd namtrace
       close $kbps_f
       $ns flush-trace
       close $tracefd
       close $namtrace
       exec nam simulacion.nam &
          exec xgraph kbps.tr -geometry 800x400 -y bandwidth -t
throughput_Vegas_30_nodos -x time &
       exit 0
}

#---------------------------------------------------------------
#LLamada a procedimientos e inicio  de la simulacion
#---------------------------------------------------------------
$ns at 0.0 "record"
$ns at 60.0 "finish"
puts "Starting Simulation..."
$ns run
```

ANEXO 2: CÓDIGO FUENTE AWK

```
BEGIN {
      print("\n\n******** Network Statistics ********\n");

      val=30;
      packet_sent[val] = 0;
      packet_drop[val] = 0;
      packet_recvd[val] = 0;
      packet_forwarded[val] = 0;

      total_pkt_sent=0;
      total_pkt_recvd=0;
      total_pkt_drop=0;
      total_pkt_forwarded=0;
      pkt_delivery_ratio = 0;
      start = 0.000000000;
      end = 0.000000000;
      packet_duration = 0.0000000000;
      recvnum = 0;
      delay = 0.000000000;
      sum = 0.000000000;
      i=0;
      total_energy_consumed = 0.000000;
}

{
state      =     $1;
time       =     $3;

node_num   =     $5;


node_id    =     $9;
level      =     $19;
pkt_type   =     $35;
packet_id  =     $41;

if((pkt_type == "tcp") && (state == "s") && (level=="AGT")) {
      for(i=0;i<val;i++) {
            if(i == node_id) {
            packet_sent[i] = packet_sent[i] + 1; }
}
}else if((pkt_type == "tcp") && (state == "r") && (level=="AGT"))
{
      for(i=0;i<val;i++) {
            if(i == node_id) {
            packet_recvd[i] = packet_recvd[i] + 1; }
}
}else if((pkt_type == "tcp") && (state == "d")) {
      for(i=0;i<val;i++) {
            if(i == node_id) {
            packet_drop[i] = packet_drop[i] + 1; }
}
```

```
}else if((pkt_type == "tcp") && (state == "f")) {
     for(i=0;i<val;i++) {
          if(i == node_id) {
          packet_forwarded[i] = packet_forwarded[i] + 1; }
}
}


if (( state == "s") &&  ( pkt_type == "tcp" ) && ( level == "AGT"
))  { start_time[packet_id] = time; }

 if (( state == "r") &&  ( pkt_type == "tcp" ) && ( level == "AGT"
)) {  end_time[packet_id] = time;  }
 else {  end_time[packet_id] = -1;  }


}

END {
for(i=0;i<val;i++) {
printf("%d %d \n",i, packet_sent[i]) > "pktsent.txt";
printf("%d %d \n",i, packet_recvd[i]) > "pktrecvd.txt";
printf("%d %d \n",i, packet_drop[i]) > "pktdrop.txt";
printf("%d %d \n",i, packet_forwarded[i]) > "pktfwd.txt";

total_pkt_sent = total_pkt_sent + packet_sent[i];
total_pkt_recvd = total_pkt_recvd + packet_recvd[i];
total_pkt_drop = total_pkt_drop + packet_drop[i];
total_pkt_forwarded = total_pkt_forwarded + packet_forwarded[i];

}
printf("Total Packets Sent            :     %d\n",total_pkt_sent);
printf("Total Packets Received        :     %d\n",total_pkt_recvd);
printf("Total Packets Dropped         :     %d\n",total_pkt_drop);
printf("Total Packets Forwarded :     %d\n", total_pkt_forwarded);

printf("Vegas("val") %d\n", total_pkt_sent) > "pks.txt";
printf("Vegas("val") %d\n", total_pkt_recvd) > "pkr.txt";
printf("Vegas("val") %d\n", total_pkt_drop) > "pkd.txt";

pkt_delivery_ratio = (total_pkt_recvd/total_pkt_sent)*100;

printf("Packet Delivery Ratio         :
     %.2f%\n",pkt_delivery_ratio);

printf("Througphut of the network(KBps)    :    %.4f\n",
((total_pkt_recvd/1000)*512)/1024);

# For End to End Delay

for ( i in end_time ) {
 start = start_time[i];
 end = end_time[i];
 packet_duration = end - start;
 if ( packet_duration > 0 )  { sum += packet_duration; recvnum++;
}
```

```
}

delay=sum/recvnum;

printf("Average End to End Delay      :%.9f ms\n", delay);
printf("Vegas("val") %.2f", delay) > "delay.txt";

if(((total_pkt_recvd + total_pkt_drop)/total_pkt_sent)==1) {
printf("Statistics Correct !!!");
}
}
```