UNIVERSIDAD TÉCNICA PARTICULAR DE LOJA

La Universidad Católica de Loja

ÁREA TÉCNICA

TÍTULO DE INGENIERO EN ELECTRÓNICA Y TELECOMUNICACIONES

Simulación de ECMP para redes LAN

TRABAJO DE TITULACIÓN.

AUTOR: Guerrero Loaiza, Pablo Fernando

DIRECTOR: Torres Tandazo, Rommel Vicente, PhD.

LOJA – ECUADOR

2019

2019

APROBACIÓN DEL DIRECTOR DEL TRABAJO DE TITULACIÓN

Doctor.
Rommel Vicente Torres Tandazo,
DOCENTE DE LA TITULACIÓN

De mi consideración:

El presente trabajo de titulación: **Simulación de ECMP para redes LAN,** realizado por **Guerrero Loaiza Pablo Fernando,** ha sido orientado y revisado durante su ejecución, por cuanto se aprueba la presentación del mismo.

Loja, agosto del 2019

f) . . . . . .

DECLARACIÓN DE AUTORÍA Y CESIÓN DE DERECHOS

Yo Pablo Fernando Guerrero Loaiza declaro ser autor del presente trabajo de fin de titulación: Simulación de ECMP para redes LAN, de la Titulación de Electrónica y Telecomunicaciones, siendo Rommel Vicente Torres Tandazo director del presente trabajo; y eximo expresamente a la Universidad Técnica Particular de Loja y a sus representantes legales de posibles reclamos o acciones legales. Además, certifico que las ideas, conceptos, procedimientos y resultados vertidos en el presente trabajo investigativo, son de mi exclusiva responsabilidad.

Adicionalmente declaro conocer y aceptar la disposición del Art. 88 del Estatuto Orgánico de la Universidad Técnica Particular de Loja que en su parte pertinente textualmente dice: "Forman parte del patrimonio de la Universidad la propiedad intelectual de investigaciones, trabajos científicos o técnicos y tesis de grado o trabajos de titulación que se realicen con el apoyo financiero, académico o institucional (operativo) de la Universidad"

f. ……
Autor: Guerrero Loaiza Pablo Fernando
Cédula: 1104203128

DEDICATORIA

Este trabajo es dedicado a mis padres Bartolomé Guerrero y Mireya Loaiza por siempre apoyarme y guiarme en cada momento de mi vida, a ellos que se han sacrificado tanto por mí, que han estado siempre que los he necesitado, jamás podré pagarles todo lo que han hecho por mí, por eso y más ¡Gracias totales!

A mis hermanas Andrea y Paty, a mis amigos por apoyarme y siempre estar pendiente en el logro de esta meta.

AGRADECIMIENTO

Agradezco principalmente a mis padres y hermanas por el apoyo incondicional, por la comprensión y paciencia que han tenido todo este tiempo de espera para verme cumplir con esta meta.

A mis amigos Raúl, Gonzalo, Eduardo, Jorge y Fabricio que siempre estuvieron pendientes de mí y apoyándome en cada momento.

Al Ing. Rommel Torres mi agradecimiento muy especial e inmenso, quien con su experiencia, paciencia, confianza y tiempo supo guiarme de manera adecuada en el desarrollo de este proyecto.

ÍNDICE DE CONTENIDOS

RESUMEN

En la actualidad la demanda de tráfico en las redes de datos es un factor en constante crecimiento, debido al consumo de altas tasas de datos por la variedad de aplicaciones, por ello se vuelve primordial tomar acciones para evitar que el desempeño, rendimiento y calidad del servicio de la red se degrade.

En este tema de titulación se realiza un estudio y análisis de algoritmos multicamino usados para balanceo de carga y prevenir la congestión en una red que posee múltiples rutas de igual costo desde el origen hacia el destino, con el fin de mejorar el rendimiento de la red dividiendo el tráfico por las distintas rutas y aprovechando al máximo la capacidad de ancho de banda de cada enlace hacia el destino.

Una de las técnicas utilizadas para realizar las tareas anteriores es Equal Cost Multipath (ECMP), que trabaja conjuntamente con protocolos de enrutamiento de estado de enlace. ECMP tiene dos modos de operación que son por paquete y por flujo, en este tema de titulación son implementados los dos modos en el simulador NS-3 y comparados con un protocolo IP de un solo camino.

PALBRAS CLAVE: ECMP, algoritmo multicamino, protocolos de enrutamiento, NS-3.

ABSTRACT

At present, the demand for traffic in data networks is a factor in constant growth, due to the consumption of high data rates, to the variety of applications, so it is essential to take actions to prevent the performance and the quality of the data network service gets degraded.

In this work, a study and analysis of multipath algorithms used for load balancing and preventing congestion in a network that has multiple routes of equal cost from the origin to the destination in order to improve the performance of the network dividing traffic through the different routes and taking full advantage of the bandwidth capacity of each link to the destination.

One of the techniques used to perform the above tasks is Equal Cost Multi-Path (ECMP), which works in conjunction with link state routing protocols. ECMP has two modes of operation that are per package and per flow, in this work the two modes are implemented in the NS-3 simulator and compared with a single path IP protocol.

KEYWORDS: ECMP, multipath algorithm, routing protocol, NS-3.

INTRODUCCIÓN

Desde hace mucho tiempo las redes IP han venido en constante crecimiento y con ello también se ha incrementado el tráfico de Internet, el acceso a la red (cableada, inalámbrica, móvil), el consumo de servicios, el número de dispositivos conectados, esto conlleva a que la demanda de requerimientos para el mejoramiento y desempeño de la red de los usuarios finales, la eficiencia de la red de los proveedores de servicios de internet y proveedores de servicios móviles, también vaya creciendo. Dicho esto, existen protocolos y técnicas de enrutamiento que ayudan a optimizar la eficiencia y rendimiento de la red, uno de estos es ECMP.

ECMP es un método o técnica de ingeniería de tráfico usado en redes que poseen múltiples rutas de igual costo desde el origen hacia el destino. ECMP trata de dividir el tráfico equitativamente entre las rutas disponibles hacia el destino, evitando así la congestión en la red e incrementando el ancho de banda (AB) disponible. También actúa como respaldo para las rutas que presenten problemas dentro de la red, es decir, si una de las rutas hacia el destino falla, el tráfico de esta es enviado por otra ruta de igual costo sin tener demasiadas pérdidas.

En la actualidad los protocolos de enrutamiento dinámico de estado de enlace son muy usados para el despliegue de redes de gran tamaño, es aquí donde ECMP es utilizado conjuntamente con este tipo de protocolos. Una de las características más relevantes de los protocolos de estado de enlace es la búsqueda de la ruta más corta (shortest-path) dentro de las múltiples rutas que puede tener un destino, el algoritmo que ayuda a la búsqueda de dicha ruta es el Dijkstra.

El propósito de este tema de titulación es implementar un algoritmo multicamino en un simulador de redes. La implementación de ECMP se lo realizó en Network Simulator 3 (NS-3), un simulador de sistemas de redes con eventos discretos. NS-3 es un software libre que fue instalado en Ubuntu 16.04.

En el Capítulo 1 se realiza un repaso sobre los protocolos de enrutamiento tanto de un solo camino como de multicamino. El Capítulo 2 contiene algunos ejemplos de protocolos y técnicas multicamino que han desarrollado diferentes autores. En el Capítulo 3 se hace un estudio sobre el protocolo ECMP sus características y funcionamiento. El Capítulo 4 describe la implementación de ECMP en el simulador NS-3. En el Capítulo 5 se realizan las pruebas de ECMP mediante la simulación en NS-3. El Capítulo 6 analiza los resultados obtenidos. Finalmente se presentan las conclusiones y recomendaciones del tema de titulación.

OBJETIVOS.

Los objetivos del presente trabajo son:

1. General.

Implementación de la estrategia de enrutamiento ECMP para redes LAN, mediante una herramienta de simulación.

2. Específicos.

- Implementar ECMP para redes LAN en un simulador.
- Desarrollar pruebas de comportamiento de ECMP en escenarios.
- Comparar la estrategia de enrutamiento ECMP con un protocolo de un solo camino.

CAPÍTULO I

MARCO TEÓRICO

1. Marco teórico.

La capa de red es la responsable de llevar los paquetes por todo el camino o los caminos, desde el origen al destino, para esto debe conocer la topología de la red y elegir las rutas apropiadas incluso para rutas más grandes, también debe tener cuidado en escoger las rutas para no sobrecargar algunas de las líneas de comunicación y los enrutadores, y dejar inactivos a otros (Tanenbaum, Wetherall, 2012).

El enrutamiento es el proceso que se encarga de seleccionar la mejor ruta desde un nodo origen hasta un nodo destino dentro de una red, utiliza algoritmos que manejan tablas y realizan las decisiones de selección de la ruta.

1.1 Protocolos de enrutamiento de un solo camino.

Los protocolos de un solo camino se caracterizan por tener o escoger una sola ruta hacia el destino, causando así un bajo rendimiento de la red debido a la congestión. Estos protocolos están implementados en los hosts terminales.

1.1.1 Capa de enlace.

Esta capa tiene como tarea la transferencia de tramas a través del canal físico, transformando un canal físico en un enlace lógico sin errores.

La capa de enlace proporciona una conexión de enlaces punto a punto entre nodos. Realiza los procedimientos y funciones para establecer, mantener y ejecutar estas conexiones.

Entre las funciones que cumple esta capa se tiene (James F. Kurose, Keith W. Ross., 2017):

- Estructura de los mensajes en tramas.
- Direccionamiento
- Control de errores
- Control de transmisión y flujo de datos

Uno de los protocolos de esta capa es el Point to Point (PPP), como su nombre lo indica es un protocolo punto a punto, está definido en RFC 1661, diseñado para dar conexión a nodos punto a punto que estén conectados a Internet. Este protocolo de enlace a su vez puede servir para transporte de protocolos de red de un lugar a otro, principalmente cuando se necesita tener administración de forma remota.

Dentro de esta capa también se encuentran las VLAN (Virtual LAN) usadas para conectar varias redes LAN físicas a través de una sola LAN lógica.

1.1.2 Capa de red.

La principal función de esta capa es enrutar los paquetes desde un nodo origen hacia un nodo destino a través de rutas disponibles. Para realizar el enrutamiento esta capa se basa

en algoritmos que ayudan a elegir que ruta deben tomar los paquetes de origen para llegar hacia su destino.

Los protocolos de enrutamiento se encargan de decidir que rutas utilizar, para el reenvío de paquetes. Dentro de los protocolos de enrutamiento de esta capa tenemos los definidos por vector distancia y por estado de enlace.

- Vector distancia, este tipo de protocolos de enrutamiento utilizan el algoritmo Bellman Ford, su nombre se debe a que usa una métrica (conteo de saltos) para calcular la distancia hacia el destino y una dirección para llegar a este. Es decir, la información de enrutamiento solo se intercambia entre routers vecinos, aquí un router informa a los demás routers vecinos de la topología y de cambios que sufra esta. Cada protocolo basado en vector distancia usa un algoritmo diferente para elegir la mejor ruta. Dentro de este tipo de protocolos tenemos el RIP, RIPv2, IGRP, EIGRP.

- Estado de enlace, en este tipo de enrutamiento, los routers necesitan conocer toda la topología de la red, se basa en los estados de enlace que se informan entre todos los routers de la red, usan LSA (publicaciones de estado de enlace) que intercambian entre ellos, recopilan información necesaria de todos los routers, cada uno calcula su mejor ruta hacia un destino, a partir de esto crean una base de datos de toda la topología de la red. La métrica en este tipo de protocolos de enrutamiento se basa en el retardo, el ancho de banda, la carga y confiabilidad de los diferentes enlaces que posee el nodo origen hacia el destino. Dentro de este tipo de enrutamiento tenemos los protocolos como OSPF, BGP, IS-IS.

1.1.3 Capa de transporte.

Los protocolos en esta capa brindan comunicación lógica entre nodos, es decir que los procesos que se ejecutan en un nodo pueden estar en un lugar diferente a la de su red local en donde está la aplicación a la cual se conectan (James F. Kurose, Keith W. Ross, 2017).

El objetivo de esta capa proporcionar la conexión del host hacia la aplicación deseada. Esta capa se vale de los protocolos de la capa de red.

Hay dos tipos de protocolos en esta capa que es el transporte sin conexión (UDP) y con conexión (TCP).

- UDP, está definido en RFC 768 y toma los mensajes que llegan de la capa de aplicación y los asocia con los campos correspondientes como son el puerto origen y destino, este último puerto, UDP lo usa en el nodo receptor para entregar los datos al proceso de la aplicación enviado desde el origen. UDP transmite segmentos que consisten en un encabezado de 8 bytes seguido de la carga útil (Tanenbaum, Andrew S., & Wetherall. David J., 2012).

El protocolo UDP es sin conexión ya que al momento de enviar el segmento no se establece ninguna conexión previa entre las dos capas de transporte de los nodos de origen y destino.

Un ejemplo de uso de este protocolo es el DNS, cuando un host desea realizar una consulta a través del DNS este construye una entidad UDP que se ejecuta en el destino (James F. Kurose, Keith W. Ross., 2017).

- TCP, este protocolo definido en RFC 793 fue diseñado con el objetivo de brindar una comunicación confiable entre un origen y un destino en una red con diversos parámetros como ancho de banda, diferente topología, retardo, diferentes tamaños de paquete, entre otro. Es así que TCP fue creado para adaptarse a circunstancias adversas de una red y sobreponerse a distintos tipos de fallas (Tanenbaum, Andrew S., & Wetherall. David J., 2012).

TCP tiene tareas como terminar temporizadores, retransmitir los datagramas que no se hayan entregado, enviar datagramas con rapidez para hacer uso de la capacidad del canal sin causar congestión.

A diferencia de UDP, el protocolo TCP es orientado a la conexión, es decir antes del envío de datagramas, TCP establece una conexión entre el origen y el destino, proporcionando así un buen desempeño evitando posible congestión en la red, fiabilidad y control de flujo. Una vez terminado el envío y recepción de datagramas entre los hosts se cierra la conexión.

1.2 Protocolos de enrutamiento Multicamino.

Los protocolos multicamino usan múltiples rutas o caminos entre un origen y el destino, permitiendo explotar los recursos disponibles en la red. Este tipo de protocolos distribuyen el tráfico sobre varias rutas simultáneamente, dando a un nodo origen la posibilidad de usar cualquiera de los múltiples caminos hacia un destino en particular en cualquier tiempo dado. Además, aprovecha la redundancia y diversidad de recursos para proporcionar beneficios como la tolerancia a fallas, balanceo de carga, incremento de ancho de banda y una mejora en las métricas de Quality of Service (QoS), como el retardo (Tsai, J., & Moors, T., 2006).

En la Figura 1 se muestra una red sencilla multicamino que consta de 4 nodos, en donde S1 es el origen y D1 el destino, la línea continua representa el enlace que usaría un protocolo tradicional (de un solo camino), mientras que un protocolo multicamino utilizaría los dos enlaces aprovechando así los beneficios que ofrecen las técnicas multicamino.

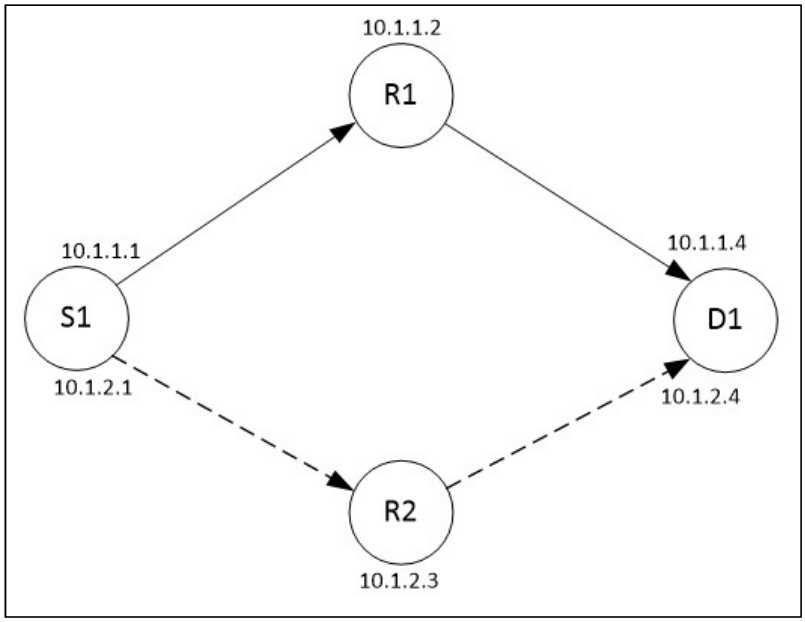

Figura 1. Topología de una red Multicamino.

Fuente: (Garrido Ortiz Pablo, 2013)

Elaboración: El autor

La Figura 2 muestra otro esquema multicamino donde se tiene más nodos y por ende más rutas desde el origen al destino. Aquí, el camino 1 es la ruta principal, los caminos 2 y 3 son rutas de respaldo. Si algún nodo falla o existe congestión en la ruta principal (1) esta se deshabilita y se levantan las rutas de backup, el tráfico es enviado por las rutas alternativas (2 y 3). Este puede ser uno de los casos en los que se aprovechan las ventajas de los protocolos multicamino como es la tolerancia a fallos y balanceo de carga o tráfico.

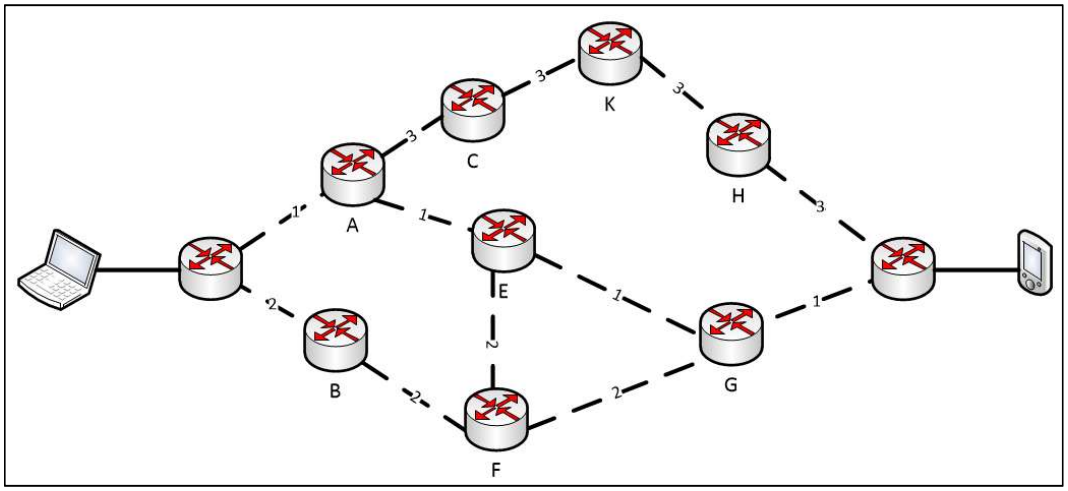

Figura 2. Topología de una red Multicamino.

Fuente: (Li Zhi-yuan, Wang Ru-chuan, and Bi Jun-lei, 2009)

Elaboración: El autor

Se han propuesto varias soluciones de este tipo de protocolos para su implementación en las diferentes capas, a continuación, se detalla en forma general a cada una.

1.1.1 Capa de enlace.

Una solución multicamino para esta capa es LACP (Link Agregation Control Protocol). Este protocolo usa la agrupación de enlaces que se realiza agregando o uniendo los puertos del switch, con el fin de utilizar múltiples conexiones de red para aumentar el rendimiento y crear redundancia en caso de falla del enlace. Una solución que aprovecha un esquema multiruta en esta capa es el SPB (Shortest Path Bridging). El IEEE 802.1aq basado en IEEE 802.1Q Ethernet Bridging, permite el uso de múltiples rutas de igual costo en una red Ethernet en malla. Esta solución es compatible con una solución mucho más grande de topologías de capa de enlace las cuales se pueden utilizar en el centro de datos de los nodos, pero no puede hacer uso de múltiples interfaces disponibles (Barreira, Da Silva, 2014).

Otra solución multicamino para la capa de enlace ha sido sugerido para conseguir un alto rendimiento en WMN (Wireless Mesh Networks), esta solución necesita implementar enlaces multicanal en combinación con enrutamiento multicamino para enrutar eficiente e inteligentemente el tráfico para conseguir un mejor rendimiento (throughput) en este tipo de redes. La capa de enlace es responsable de la programación del canal y paquetes, la primera es usada para controlar el canal en el que la información es recibida, la segunda cuando se envía los paquetes. El esquema multicamino es responsable de seleccionar las mejores dos rutas para la puerta de enlace. Una solución en este nivel tiene un gran potencial para conseguir un buen desempeño y un mayor rendimiento de extremo a extremo, ofreciendo estos beneficios al recurrir a la división del tráfico a través de diferentes canales, programando diferentes tiempos y usando diferentes rutas. Las soluciones de multicamino para la capa 2 están mayormente restringidas por las redes de área local, no haciendo frente a la posible diversidad de multicaminos que están disponibles en la capa de red (Barreira, Fernando Mira da Silva, 2014).

1.1.2 Capa de red.

La implementación de un protocolo multicamino en la capa de red, hace que en la capa de transporte la conexión desde un origen a un destino se divida en varios subflujos, permitiendo que cada subflujo sea enrutado por un camino diferente. El balanceo de carga es realizado en el nivel de conexión y no en el nivel del paquete, junto con el hecho de que esta solución solo ofrece una sola conexión para la capa de transporte, el rendimiento estaría dado por el enlace más congestionado o el más lento, porque el control de congestión de las diferentes rutas son agregadas por el protocolo de transporte. Esta

solución puede mitigar retransmisiones innecesarias en la capa de transporte debido al re-ordenamiento de los paquetes, esto causa una reducción significativa en el rendimiento de la conexión. Unas de las causas de estas retransmisiones innecesarias son debido a que la mayoría de los dispositivos de red dispersos a través de Internet no soportan y no reconocen el tráfico generado. Para la implementación de un protocolo multicamino en la capa de red es necesario recurrir a actualizaciones de los dispositivos de red que se están usando (Barreira, Fernando Mira da Silva, 2014).

1.1.3 Capa de transporte.

La implementación de un protocolo multicamino en la capa de transporte tiene la posibilidad de reunir información como la capacidad, latencia y estado de congestión para cada ruta usada; con esta información es posible reaccionar ante la congestión en la red y mover el tráfico evitando así las rutas congestionadas. Un protocolo multicamino en esta capa permite que sea transparente para las capas superior e inferior, es decir que se usan múltiples flujos similares a las conexiones regulares TCP y el tráfico se mueve sin haber sido retenido en la red. También permite funciones de administración de rutas, programación de paquetes, control de congestión e incluso interfaz de subflujos sin necesidad de modificar capas superior e inferior. Dicho esto, para que la capa de transporte soporte un enlace multicamino, solo se requiere que los puntos finales soporten el protocolo, además, no es necesario actualizar cualquier router o equipo de capa 3 que se encuentre entre los puntos finales (Barreira, Fernando Mira da Silva, 2014).

1.1.4 Capa de Aplicación.

Un ejemplo de soluciones multicamino en la capa de aplicación son los protocolos P2P (Peer-to-Peer) como bittorrent, el multicamino trabaja en granularidad de fragmentos y tiene como objetivo incrementar el rendimiento. Logran alcanzar su objetivo bajando los diferentes fragmentos de un archivo a través de diferentes pares, eligiendo bajar de los servidores disponibles más rápidos. Es posible desarrollar una aplicación multicamino que puede proveer multihoming para servidores y dispositivos disponibles usando protocolo P2P. El problema con este tipo de soluciones es que puede traer limitaciones a otros usuarios de la red, siendo así que una implementación en este nivel ofrecerá una injusta competencia para los recursos disponibles. De hecho, hacen uso de la diversidad de servidores disponibles (Barreira, Fernando Mira da Silva, 2014).

CAPÍTULO II
ESTADO DEL ARTE

2. Estado del arte.

En los diferentes tipos de red sean inalámbrico, sensores, ethernet, celular existen protocolos multicamino que han sido creados a partir de modificaciones de protocolos ya existentes.

En esta sección se tratan algunos de ellos que se encuentran relacionados con el presente trabajo.

2.2 Ejemplos de protocolos multicamino.

En (Sumathi, R., & Srinivas, M. G., 2012), los autores realizan una comparación de varios protocolos de enrutamiento multicamino basados en la QoS, los protocolos evaluados son: SAR (Sequential Assignment Routing), MMSPEED (Multi-Path and Multi-SPEED Routing), MCMP (Multi Constrained QoS Multi-Path), MCBR (Message-Initiated Constrained-Based Routing) y EQSR (Energy Efficient and QoS aware multipath Routing. En NS-2 se analizan varios parámetros como retardo, ancho de banda, energía eficiente, confiabilidad, sobrecarga en control del paquete, tiempo de vida en la red. Los protocolos de enrutamiento basados en multicamino tratan de mejorar la confiabilidad a través de múltiples caminos entre el origen y el destino, en lugar de usar una sola ruta. Esta clase de protocolos se enfocan principalmente en el balance de carga, tolerancia a fallos, agregación de ancho de banda y reducción del retardo.

Los autores (Deepinder, Wadhwa, Tripatjot, & Panag, 2011) en su trabajao presentan una comparación de desempeño de protocolos de enrutamiento para redes adhoc, mediante simulaciones en NS-2. Los protocolos comparados son AOMDV (Adhoc On-demand Multipath Distance Vector), AODV (Adhoc On demand Distance Vector) y DSDV (Destination Sequence Distance Vector). A continuación, se describen brevemente estos protocolos:

- **Protocolo AODV,** es un protocolo reactivo de un solo camino, provee comunicaciones unicast, broadcast y grupos multicast para redes adhoc. Este protocolo mantiene una tabla de enrutamiento en la que almacena información de enrutamiento de los siguientes saltos para llegar al nodo destino (Deepinder et al., 2011).
- **Protocolo DSDV,** es un protocolo proactivo que brinda un solo camino o ruta del origen al destino, usa el algoritmo de enrutamiento del camino más corto del vector distancia, reduce la cantidad de transmisiones de sobrecarga a través de la red (Deepinder et al., 2011).
- **Protocolo AOMDV,** es un protocolo basado en el AODV que pretende vincular varias rutas, libres de lazos o loops, para esto usa técnicas de enrutamiento multicamino con las que se logra establecer varias rutas de un origen al nodo destino. Además, provee de tolerancia a fallas, incremento de ancho de banda, balance de carga (Deepinder et al., 2011).

Según los resultados de las simulaciones, de los tres protocolos comparados, el protocolo AOMDV es capaz de lograr un mejor rendimiento superando a los otros dos protocolos de vector distancia de un solo camino.

En el trabajo de (Tekaya, Tabbane, & Tabbane, 2010), se realiza la propuesta de un nuevo protocolo, LB-AOMDV (Load Balancing-AOMDV) basado en AOMDV con el objetivo de lograr un mejor mecanismo para el balance de carga. Este protocolo es una extensión de AOMDV dotado de cierto mecanismo o técnicas que mejoran su desempeño. AOMDV permite encontrar muchas rutas del origen hacia el destino, pero únicamente usa una para la transmisión de los datos. En LB-AOMDV busca muchas rutas del origen al destino realizando un balanceo de carga en la red.

Los resultados de las simulaciones muestran que el protocolo propuesto tiene un mejor desempeño con respecto al retardo promedio, balance de carga y capacidad, comparado con otros protocolos multicamino como el MSR (Multipath Source Routing) y AOMDV.

En (Cho Aye & Aung, 2014), se propone un protocolo de enrutamiento multicamino con eficiencia energética para elegir la mejor ruta considerando algunas métricas de energía como la potencia de transmisión de los nodos y la energía residual, maximizando así el tiempo de vida de la red, reduciendo el consumo de energía. El objetivo del protocolo es encontrar la mejor ruta basado en dos métricas de energía mientras se elige una ruta para la transferencia de datos. El protocolo es implementado en NS-2 dando como resultado un alto rendimiento comparado con el protocolo AOMDV.

En (Yahya & Ben-othman, 2009), proponen un protocolo de enrutamiento multicamino robusto y de eficiencia energética (REER - Robust and Energy Efficient multipath Routing) que usa la energía residual, el tamaño del búfer disponible del nodo y la relación señal a ruido (SNR) para predecir el mejor salto a través de la etapa de construcción de las rutas. El protocolo propuesto es evaluado mediante simulaciones y comparado con otros protocolos dando como resultado un mejoramiento en energía, un bajo promedio de retardo y una alta tasa de paquetes entregados. Se han propuesto soluciones diseñadas específicamente para WSN (Wireless Sensor Networks). Estas técnicas de enrutamiento pueden ser clasificadas según la operación del protocolo ya sea basados en negociación, basado en consultas, basado en QoS y basado en múltiples rutas.

En el trabajo realizado por (Garrido Ortiz Pablo, 2013), se presenta un estudio del protocolo MPTCP, que es una extensión del TCP, el cual mediante técnicas o mecanismos multicamino permiten la división de una conexión TCP en múltiples flujos simultáneos, logrando una mejora en el rendimiento y en la recuperación frente a fallos de rutas. El autor realiza la evaluación del desempeño del MPTCP frente al TCP mediante simulaciones usando NS-3, dando como resultado una notable diferencia en el rendimiento entre MPTCP y TCP. MPTCP propone usar múltiples caminos para el transporte de los paquetes entre un

origen y destino. Puede hacer uso de diferentes tecnologías a nivel físico para llegar al destino. Los protocolos multicamino permiten una mejor gestión de los recursos de red. Son dos objetivos principales del MPTCP: Uno es aumentar el rendimiento, siendo en el peor de los casos no menor al rendimiento de TCP. El segundo es mejorar la fiabilidad, se usa diferentes caminos como rutas alternativas cuando la principal no está disponible.

En el trabajo de (Saif Saad Hammed, 2018) realiza un estudio y análisis de algoritmos para evitar el congestionamiento en la red con el fin de mejorar el rendimiento dividiendo el tráfico entre las rutas disponibles entre el origen y el destino aplicando ECMP y a la vez que realiza un enrutamiento rápido usando OSPF. Esta implementación es hecha en Matlab y comparada con un protocolo de un solo camino. De los resultados obtenidos en la simulación se tiene que el ECMP tiene mejor rendimiento que el protocolo de un solo camino ya que presenta un mejor rendimiento y el retardo es menor, en el escenario propuesto.

CAPÍTULO III

ESTRATEGIA DE ENRUTAMIENTO MULTICAMINO ECMP

3. Estrategia de enrutamiento ECMP.

Equal-Cost Multi-Path routing (ECMP) es una estrategia o técnica de enrutamiento donde el reenvío de paquetes al siguiente salto hacia un destino puede ser enviado por otras "mejores rutas" de acuerdo al cálculo de métricas de enrutamiento.

El enrutamiento multicamino ECMP puede ser usado en conjunto con otros protocolos de enrutamiento, ya que es una decisión por salto que se limita a un solo enrutador. Este tipo de enrutamiento ofrece un considerable aumento en el ancho de banda por el balanceo de carga sobre múltiples rutas (Cisco, 2018).

Además del balanceo de carga también ofrece una tolerancia a errores para las rutas que presentan errores.

3.1 Funcionamiento.

ECMP es una técnica que dispone del uso de varias rutas o caminos de igual costo en un enrutamiento IP. Esta función ayuda a la distribución del tráfico de manera uniforme en toda la red, además de actuar como un método de protección ante fallos de rutas.

Con ECMP, las rutas de igual costo son instaladas en la tabla de balanceo de carga en la capa de reenvío del router, así luego de detectar una falla en algún enlace, el tráfico es distribuido al resto de rutas sin una pérdida severa de tráfico. ECMP no requiere ninguna configuración especial porque SPF (shortest path first) calcula automáticamente rutas de igual costo las mismas que se anuncian en la capa de reenvío. El único factor variable es el número de rutas ECMP, siendo este un factor limitante ya que tiene un número máximo de rutas ECMP que puede soportar el algoritmo de balanceo de carga. Normalmente, el número de rutas que se puede configurar está entre 1 y su valor máximo que comúnmente está entre 8 y 16. ECMP es implementado en protocolos de enrutamiento de estado de enlace, ya que necesita una pequeña modificación en el cálculo de su ruta, pero también se han publicado implementaciones de ECMP para protocolos vector distancia, como IGRP y EIGRP (Ari Lappetelainen, 2011).

En la Figura 7 se muestra un ejemplo de balanceo de carga con ECMP. El tráfico es distribuido de manera uniforme en toda la red. Además, como se ha mencionado anteriormente las rutas sirven como respaldo entre sí, es decir, si una de las tres rutas falla, el tráfico será dividido en las otras rutas que queden activas. Como se ve en la Figura 7 solo hay un nodo y un enlace que comparte más de una ruta, en este caso, solo el router origen necesita soportar ECMP. Aunque los demás routers solo soportarían enrutamiento de una sola ruta, ECMP seguirá trabajando de la misma forma. Sin embargo, solo el router origen podrá dividir el tráfico y dar protección ante fallos. Si se usa una mezcla de enrutamiento de una sola ruta con ECMP es importante realizar una configuración adecuada para crear una cantidad suficiente de rutas ECMP.

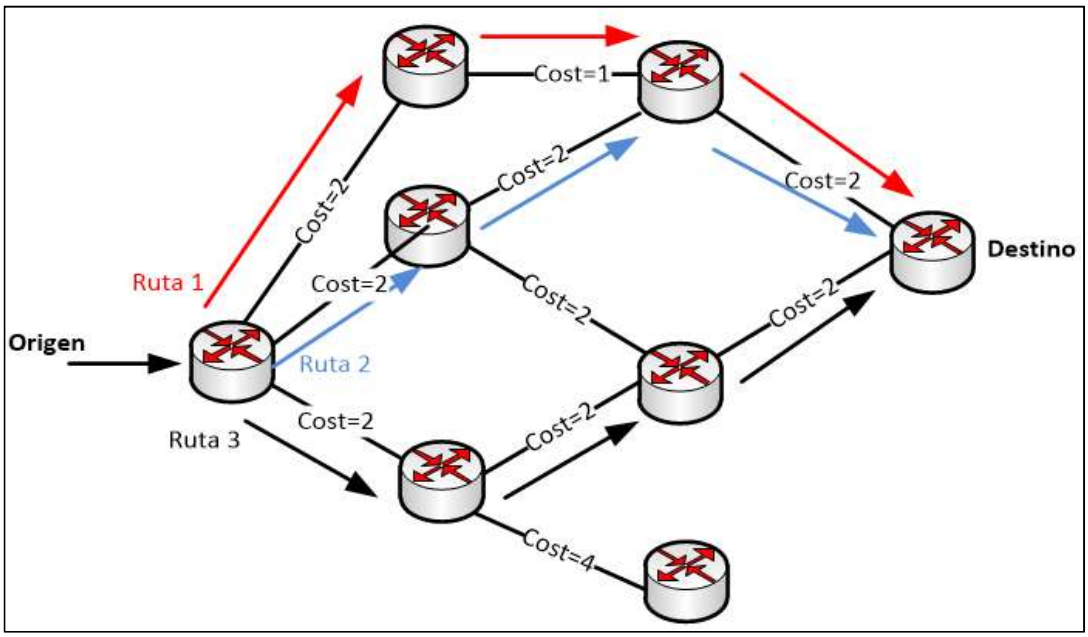

Figura 7. Balanceo de carga ECMP

Fuente: (Ari Lappetelainen, 2011)

Elaboración: El autor

ECMP al ser una técnica de enrutamiento de balanceo de carga, puede trabajar en dos modos por paquete y por flujo.

3.2 ECMP por paquete

En este modo, el nodo envía los paquetes en forma secuencial alternado, usando las diferentes rutas disponibles hacia el destino, es decir, una ruta es seleccionada aleatoriamente entre múltiples rutas de igual costo para enviar los paquetes, el enrutador envía un paquete para un destino por la primera ruta, el segundo paquete para el mismo destino por la segunda ruta, el tercer paquete de nuevo por la primera ruta, así sucesivamente.

En la Figura 8 se muestra el funcionamiento de ECMP por paquete, en donde, en el origen, (router R1) existen seis paquetes (P1, P2, P3, P4, P5, P6) que tienen como destino los routers R3 y R4. El envío de los paquetes hacia su destino lo realiza de forma alternada por los dos enlaces (A y B). Así, el paquete P1 con destino al router R4 es enviado a través del enlace A, luego el paquete P2 con destino al router R3 es enviado a través del enlace B, el paquete P3 hacia R3 lo envía por el enlace A, el paquete P4 que tiene como destino R4 es enviado por el enlace B, el paquete P5 con destino R3 lo envía a través del enlace A y por último el paquete P6 que tiene como destino R4 es enviado a través del enlace B. Es así como ECMP por paquete envía el tráfico de forma secuencial alternada.

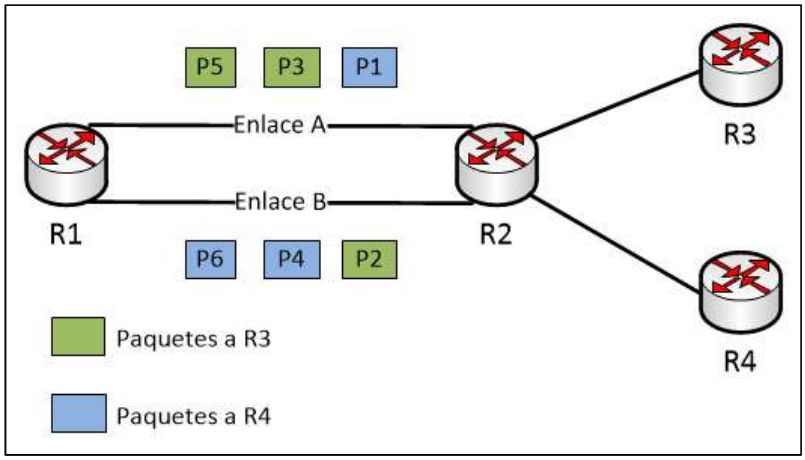

Figura 8. Balanceo de carga ECMP por paquete.

Fuente: (Huawei, Per-Flow and Per-Packet Load Balancing, 2019)

Elaboración: El autor

Este modo, ECMP por paquete, es usado cuando se requiere una utilización igual de las rutas al mismo destino evitando la congestión en la red y para asegurar el uso equitativo de la capacidad de los enlaces. Sin embargo, debido a su forma de dividir el tráfico, da lugar a que en el destino los paquetes lleguen fuera de orden lo cual produce algunos problemas en aplicaciones como voz o video.

3.3 ECMP por flujo

En ECMP por flujo se clasifica los paquetes en diferentes flujos de acuerdo a una regla determinada, que en este caso es la de 5-tupla, es decir que se basa en la IP origen y destino, número de protocolo, puerto de origen y destino. Los paquetes del mismo flujo son enviados hacia el destino por la misma ruta.

En la figura 9 se muestra un ejemplo del funcionamiento de ECMP por flujo. R1 envía seis paquetes (P1 a P6) en secuencia a R2 a través de los enlaces A y B. Los paquetes P2, P3, y P5 son enviados a R3 mientras que los P1, P4, P6 son enviados a R4, Cuando se usa el balanceo por flujo, los paquetes con destino R3 son enviados por el enlace A, mientras que los paquetes con destino R4 con enviados por el enlace B, de forma alternativa los paquetes hacia R3 también pueden ir por el enlace B y los paquetes a R4 por el enlace A.

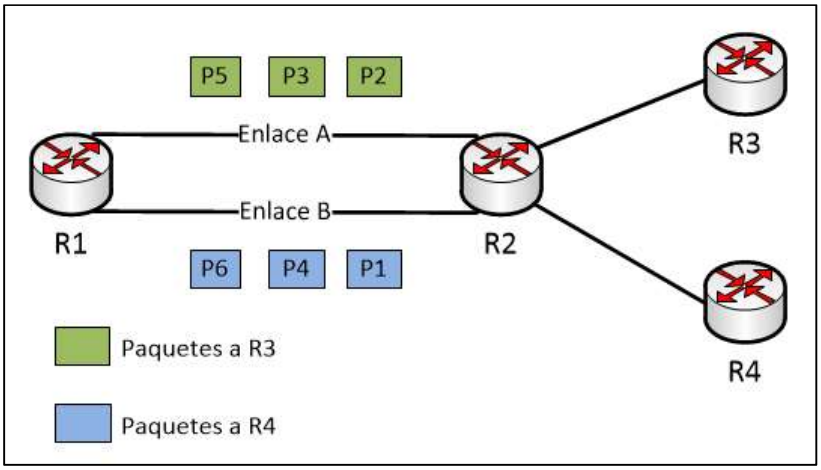

Figura 9. Balanceo de carga ECMP por flujo.

Fuente: (Huawei, Per-Flow and Per-Packet Load Balancing, 2019)

Elaboración: El autor

3.4 Comparación entre ECMP por paquete y ECMP por flujo.

ECMP por paquete garantiza un equilibrio de carga más uniforme que ECMP por flujo entre las rutas de igual costo. En ECMP por flujo, las reglas de balanceo de carga y las características del flujo determinan si el tráfico se puede balancear de forma equitativa entre los enlaces, aunque en la mayoría de los casos la utilización de los enlaces es desigual.

ECMP por flujo evita el reordenamiento masivo de paquetes, es decir, los paquetes que pertenecen a un flujo se envían por la misma ruta, esto garantiza la secuencia correcta de los paquetes a diferencia de ECMP por paquete que en el destino tiene que realizar un reordenamiento de los mismos.

Los paquetes en ECMP por paquete llegan al destino fuera de orden debido a las siguientes causas:

- Los enlaces no son óptimos, causando pérdida de paquetes, retardo, paquetes erróneos.
- Los paquetes son de diferentes tamaños, cuando se transmiten este tipo de paquetes a través del mismo enlace, los paquetes más pequeños son los primeros en llegar al destino incluso si se enviaron luego de los paquetes de mayor tamaño.

3.5 Componentes de ECMP.

Usando múltiples rutas de igual costo o peso, se requieren de algoritmos ECMP para realizar el balanceo de carga.

Un problema de implementación es como distribuir el tráfico de manera uniforme cuando se dispone de varias rutas buenas.

3.5.1 Algoritmos.

Los objetivos de un algoritmo de balanceo de carga son minimizar el consumo de los recursos de red, proporcionar balanceo de carga y una interferencia mínima de manera simple, eficiente e implementable. Se han desarrollado varios algoritmos basados en hash para resolver el problema de la distribución de tráfico. El tráfico se debe dividir utilizando la distribución basada en flujo para evitar el reordenamiento masivo de los paquetes, ya que la mayoría de rutas siempre tienen diferentes retrasos y por lo tanto los paquetes se entregan fuera de orden. Dicho esto, es aconsejable no usar los algoritmos basados en paquetes, como el Round Robin o random. El uso de hashing basado en el flujo puro presenta algunos inconvenientes, en primera instancia, el tráfico total no se distribuye uniformemente, la distribución de tráfico depende de diferentes flujos, si un flujo es mayor y usa una gran cantidad de ancho de banda en comparación con otros, el tráfico no puede estar bien equilibrado, porque los paquetes de un solo flujo seleccionan siempre la misma ruta. Cada algoritmo hash trata de crear una distribución uniforme de flujos entre puertos de salida, por lo tanto, el balanceo de carga trabaja de manera más uniforme cuando aumenta el número de flujos. La selección de flujos no debe ser totalmente predecible, ya que esto podría hacer que la red sea vulnerable a ataques de DoS (Denial of Service). Los algoritmos de balanceo de carga calculan un hash sobre los campos seleccionados en el encabezado del paquete. Normalmente se usan las direcciones de origen (SA) y destino (DA) del encabezado IP, pero también se puede utilizar el campo del protocolo y el tipo de servicio del encabezado IP, el SA y DA de la capa MAC o los puertos origen (SP) y destino (DP) (Ari Lappetelainen, 2011).

Hay varios algoritmos basados en flujo disponibles, los algoritmos basados en flujo como: Hash Modulo-N, Hash-Threshold y Highest Random Weight (HRW). Todos estos mantienen el mismo flujo en la misma ruta. Las principales diferencias son el factor de interrupción y la complejidad computacional. Siendo el factor de interrupción como la medida de cuantos flujos se cambian a otra ruta debido a la adición o eliminación del siguiente salto. De acuerdo a los tres algoritmos mencionados anteriormente el HRW tiene el mejor factor de interrupción, pero tiene la mayor complejidad computacional. Hash-Threshold tiene un factor de interrupción casi tan bueno como HRW y es computacionalmente menos complejo, es así que la mayoría de autores proponen a este último como el mejor para el balanceo de carga estática. El Hash – Threshold es un método para determinar cuál es el siguiente salto a usar cuando el enrutamiento sea con ECMP. El router selecciona una clave realizando un hash (CRC16) sobre los campos del encabezado del paquete que identifican un flujo. A los N saltos siguientes se asigna regiones únicas en el espacio clave. El router usa la clave para determinar qué región y que próximo salto usar (Network Working Group, 2000) (Network Working Group. 2000).

3.5.2 Paquetes.

En la Figura 10 se muetra la estructura por la que está formado un paquete en ECMP.

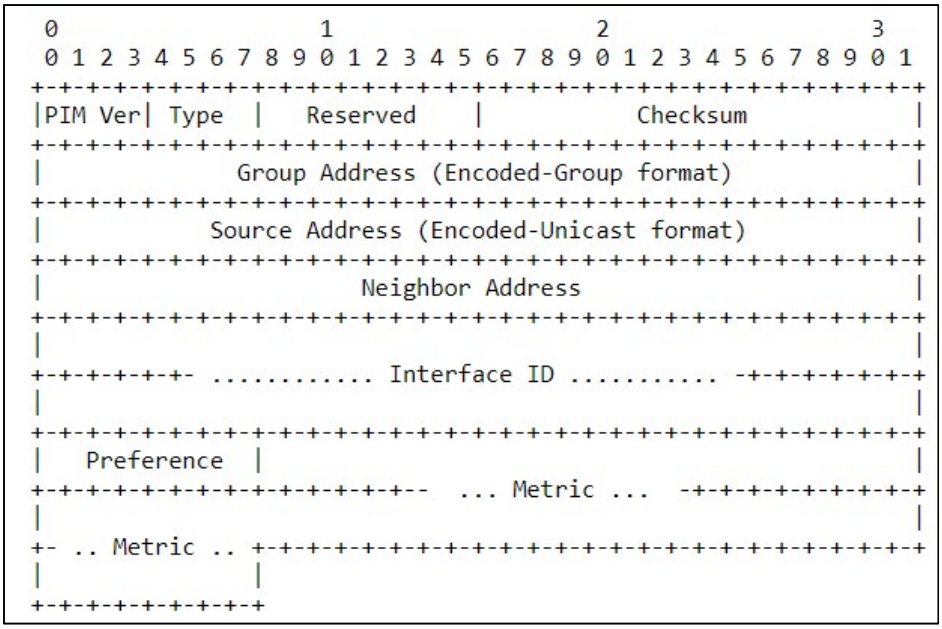

Figura 10. Estructura de paquete ECMP

Fuente: (Network Working Group, 2000)

Elaboración: (Network Working Group, 2000)

De la Figura 8 se tiene que:

PIM Ver. - Protocolo enrutamiento multidifusión que usa la base de información de enrutamiento unicast subyacente o una base de información de enrutamiento independiente compatible con multidifusión.

Type. – Es asignado por IANA.

Reserved. – Pone en cero la transmisión.

Checksum. – Verifica la integridad del paquete.

Group Address (64/160 bits). - Grupo de direcciones codificadas.

Source Address. – Dirección de origen.

Neighbor Address. – Dirección del vecino.

Interface ID (64 bits). – Identifica la interfaz de un router vecino.

Preference (8 bits). – Valor reservado que se usa para indicar que el siguiente valor es de la métrica.

Metric (64 bits). – Es el valor de la métrica. Esta métrica puede contener parámetros de ruta definidos por el usuario.

CAPÍTULO IV
IMPLEMENTACIÓN DE ECMP EN NS-3

4. Implementación de ECMP en NS-3.

En este capítulo se detalla la implementación en NS-3 de la estrategia multicamino ECMP, el protocolo en el que se basa, las clases que contiene, las librerías, algoritmos que se utilizan para lograr el funcionamiento en el simulador NS-3.

Como se ha mencionado en el capítulo 3, ECMP es usado en protocolos de enrutamiento de estado de enlace, ya que estos necesitan una pequeña modificación para el cálculo de la ruta.

En NS-3 no existe ninguna implementación de protocolos de estado de enlace, ante esto existe una clase que se asemeja o trabaja como dichos protocolos, este es el *IPv4 global routing.*

IPv4 global routing permite un enrutamiento dinámico, que hace que cada nodo cree rutas hacia todos los nodos de la red, tienen una vista de toda la topología y pueden construir rutas cortas entre todos los pares de nodos. Estas rutas son almacenadas en una base de datos local, usa el algoritmo Dijkstra para calcular la ruta más corta entre todos los caminos posibles hacia los nodos y configura o publica la tabla de enrutamiento de cada nodo.

En la Figura 11 se puede ver un diagrama general del funcionamiento del enrutamiento global junto con el uso de la ruta más corta. La simulación inicia asignando una dirección IP para cada nodo, esta IP identifica la ruta de enrutamiento IP entre el nodo origen y el nodo destino, luego se establece el tamaño del paquete mientras se fija una etiqueta para cada nodo. La simulación espera la IP y la etiqueta del origen al destino para establecer la IP del remitente y la IP de destino, realiza el cálculo de la ruta más corta para enviar los datos. Con el uso de sentencias condicionales se determina si la transmisión y recepción son exitosas, según esto se obtiene las métricas para evaluar el sistema.

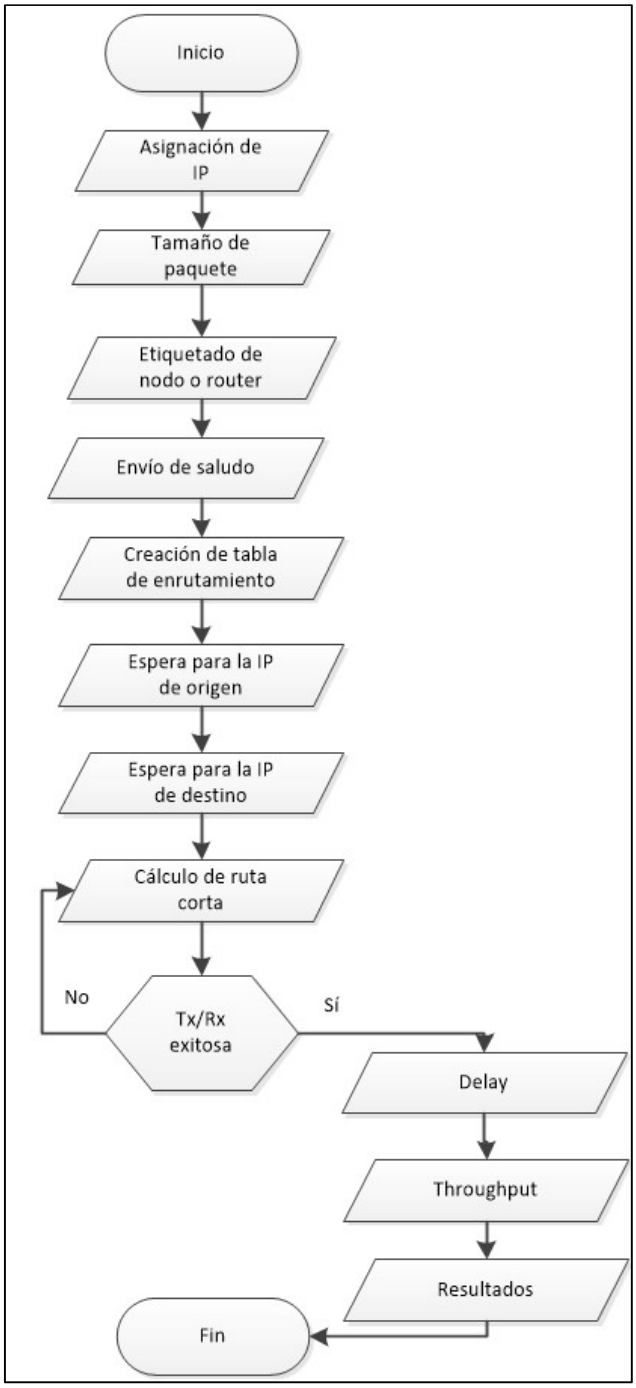

Figura 11. Funcionamiento de global routing con SPF.

Fuente: (Saif Saad Hammed, 2018)

Elaboración: El autor

En la Figura 12 se muestra el diagrama de funcionamiento de ECMP usando el enrutamiento global y utilizando el cálculo de la ruta más corta.

El funcionamiento es similar al descrito en la Figura 11, pero en este caso se agrega ECMP luego del cálculo de la ruta más corta y se establece el balanceo de carga que se utiliza, ya sea por paquete o por flujo.

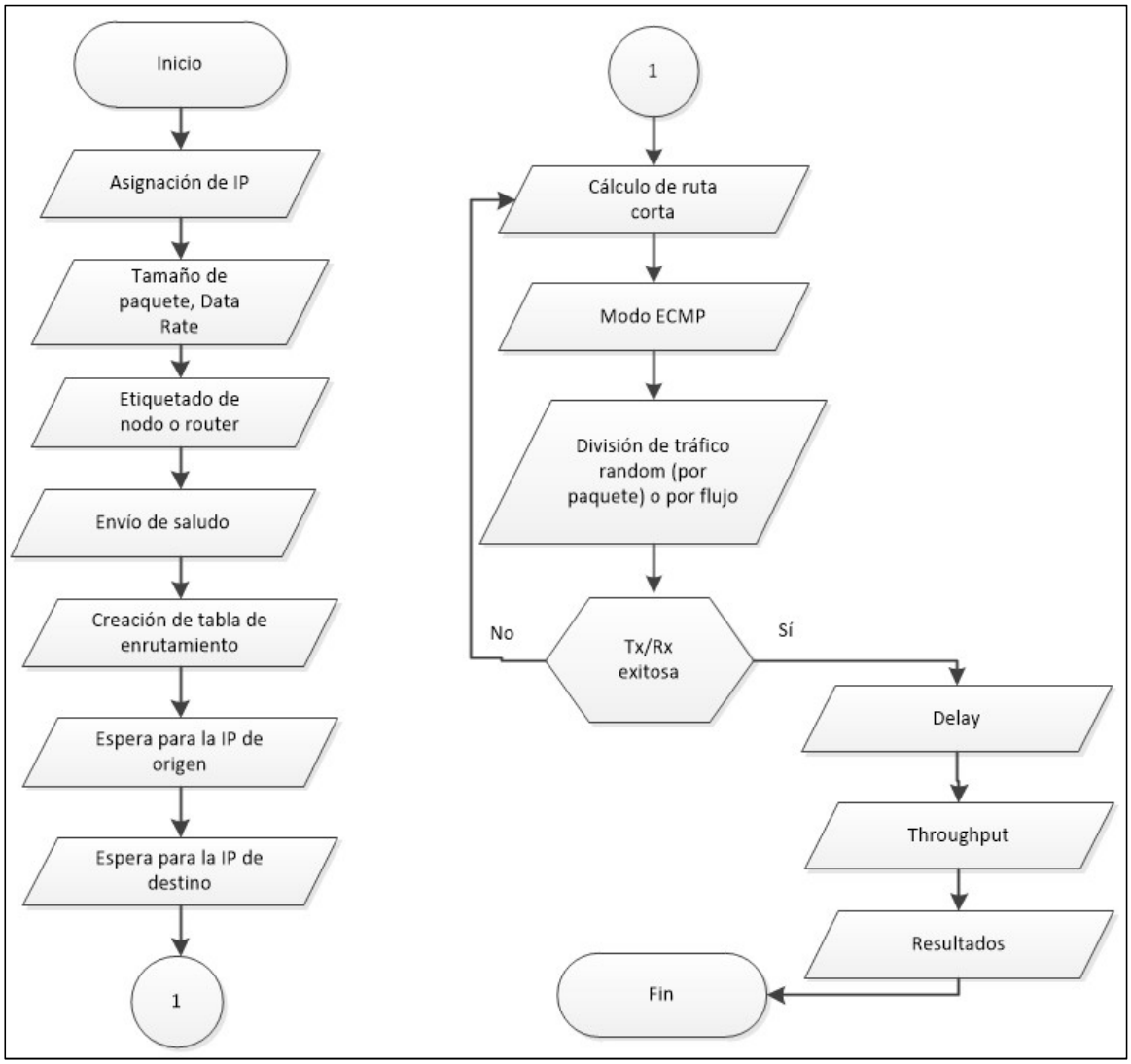

Figura 12. Funcionamiento de ECMP con enrutamiento global con SPF.

Fuente: (Saif Saad Hammed, 2018)

Elaboración: El autor

4.1 Diagrama de clases.

En esta sección se describe la estructura del protocolo que se usa, IPv4 Global Routing, para el funcionamiento de ECMP, se muestra sus clases, atributos y las relaciones entre ellos.

Clase principal.

En la Figura 13 se puede ver la clase **Ipv4GlobalRouting,** esta es la clase principal del protocolo en donde se establecen sus atributos y métodos. En la parte de atributos del diagrama se puede observar que está incluido el ECMP por flujo a través del atributo *m_flowEcmpRouting :bool*, esta característica es la que se agrega modificando los archivos que se mencionan sección 3.2. También se observa el atributo *m_randomEcmpRouting :bool*, el cual permite el uso del modo ECMP random o por paquete.

NetworkRoutes esta clase guarda un registro de una entrada en la tabla de enrutamiento hacia otras redes.

HostRoutes al igual que la anterior, guarda un registro de entrada en la tabla de enrutamiento, pero hacia los hosts.

ASExtenalRoutes esta clase guarda un registro de rutas externas importadas.

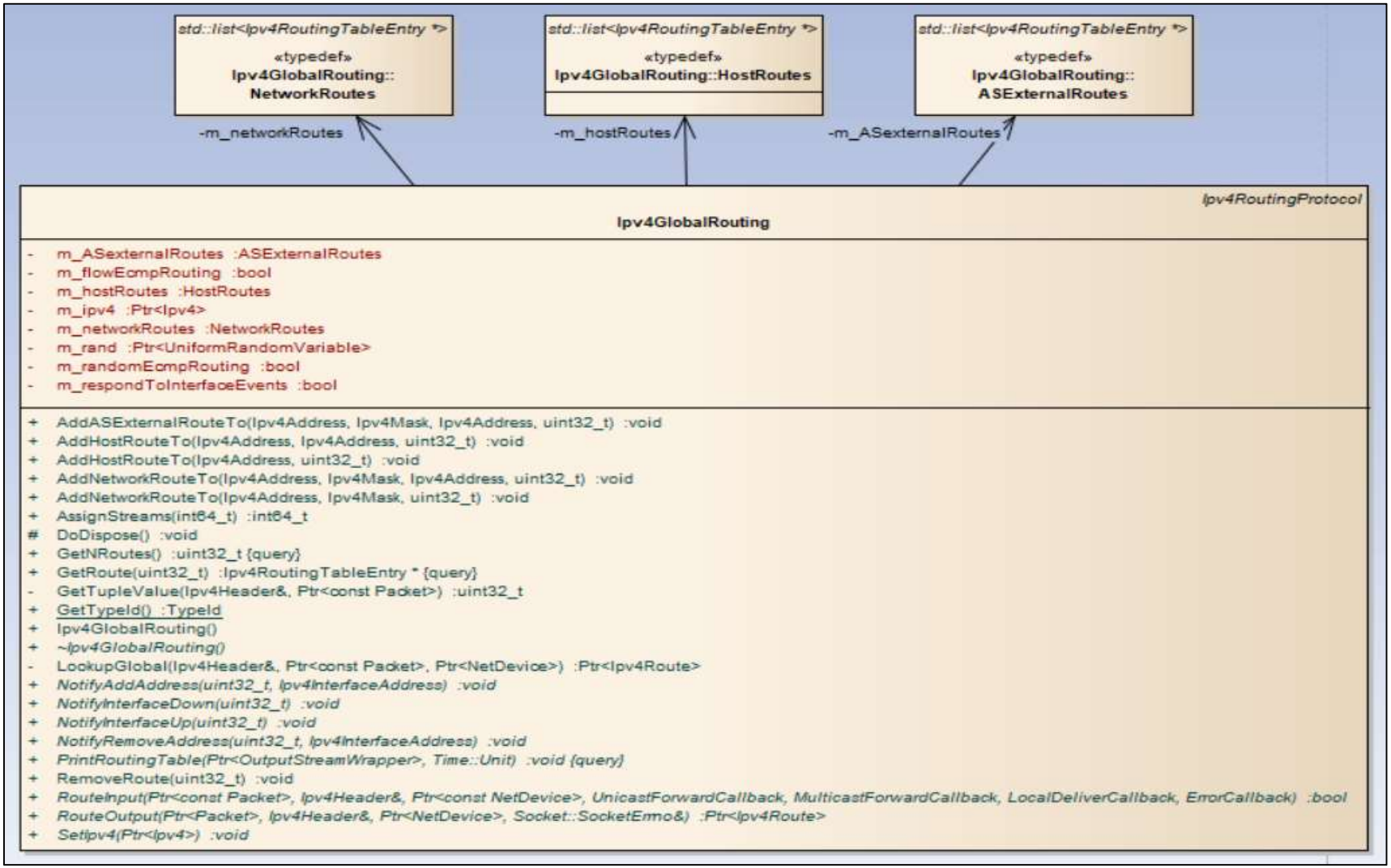

Figura 13. Diagrama de clases.

Fuente: El autor.

Elaboración: El autor.

4.2 Archivos modificados en NS-3.

Dentro de IPv4 global routing se encuentra implementado ECMP random que corresponde al modo ECMP por paquete, por defecto está deshabilitado, para habilitarlo se deberá poner en verdadero (true) al atributo "RandomEcmpRouting", así los paquetes se enrutaran de forma aleatoria por las diferentes rutas disponibles.

Para agregar la funcionalidad ECMP por flujo dentro de IPv4 global routing se debe aplicar el parche[1] “ecmp.patch”, el cual luego de ejecutarlo dentro del directorio del simulador NS-3 se tiene como resultado algunos cambios en archivos como: ipv4-global-routing.cc, ipv4-global-routing.h y wscript.

Las modificaciones que realiza el parche en los archivos anteriores también se los puede realizar de forma manual siguiendo los cambios para cada archivo descritos en el Anexo 2.

En la Tabla 1 están especificados los archivos que se modifican y que se mencionaron anteriormente.

Tabla 1. Archivos modificados.

Archivo	Descripción
ipv4-global-routing.h	En este archivo se agrega la opción ECMP por flujo, definiendo así el acceso a la cabecera de los paquetes ya sea por random o por flujo.
ipv4-global-routing.cc	Este archivo es modificado para activar el ECMP random y además para agregar el ECMP por flujo. Además, especifica métodos, clases y parámetros para realizar la simulación.
wscript	Este archivo es modificado agregando las funciones, archivos de código fuente necesarios para que haga el llamado y se pueda realizar la compilación del protocolo.

Fuente: El autor.

Elaboración: El autor

Al aplicar el parche también se genera un script (ecmp-routing.cc) que se usa como base para trabajar con los dos modos de ECMP.

4.2.1 Ipv4-global-routing.h.

En este archivo se agrega el atributo de ECMP por flujo, que al establecer el atributo en verdadero se realizará el balanceo de carga por flujo. También se define el acceso a la cabecera del paquete para obtener el valor de la tupla. Las líneas que se agregan se detallan en el Anexo 2.

[1] https://www.nsnam.org/bugzilla/show_bug.cgi?id=667. Última revisión julio 2019.

4.2.2 Ipv4-global-routing.cc

Este archivo especifica métodos, clases y parámetros para realizar la simulación. Por defecto el ECMP random o por paquete se encuentra desactivado, para activarlo se deberá poner en “true” el atributo "RandomEcmpRouting". Para el funcionamiento de ECMP por flujo se agregó el atributo “FlowEcmpRouting” que se deberá establecer en “true” cuando se quiera tener el ECMP por flujo. Para esto también se agrega una función que se encarga de calcular un valor entero de la 5 tupla de las cabeceras de los paquetes para clasificarlos por flujo. Las líneas que se añadieron a este archivo se encuentran detalladas en el Anexo 2.

4.2.3 Wscript

Aquí se encuentran las líneas de código que se utilizan para llevar a cabo una función determinada. En este archivo se agregan las cabeceras necesarias para que ECMP pueda compilar.

Define el método en el cual hace los llamamientos a: las funciones para ECMP, las cabeceras y archivos de código fuente para poder realizar la compilación.

CAPÍTULO V
SIMULACIÓN DE ECMP EN NS-3

5. Simulación de ECMP en NS-3.

5.1 Metodología.

Este trabajo de titulación se basa principalmente en la implementación de ECMP en el simulador NS-3, para realizarlo se procedió de la siguiente manera:

- Se analiza el funcionamiento de ECMP en sus modos por paquete y por flujo.
- Se selecciona el protocolo IP de un solo camino con el que se hace la comparación.
- Se especifican las métricas de rendimiento a evaluar en las simulaciones, las cuales son: rendimiento, retardo, jitter.
- Se definen los escenarios en los cuales se implementan los modos ECMP y el protocolo IP de un solo camino.
- Se establece los parámetros de simulación para los escenarios propuestos.
- Se realiza la configuración y codificación de ECMP en NS-3.
- Se implementa las respectivas simulaciones de los escenarios.
- Se obtienen los datos de las métricas de rendimiento establecidas.
- Se compara los modos ECMP con el protocolo IP de un solo camino.

5.2 Métricas de rendimiento.

Las métricas son parámetros usados para evaluar la eficiencia del protocolo de enrutamiento.

En el proceso de pruebas realizado se obtienen datos como resultado de las simulaciones, estos nos servirán para medir la eficiencia de ECMP implementado frente a un protocolo IP de un solo camino, en la misma red. Los protocolos evaluados son los siguientes: un protocolo IP de un solo camino y dos versiones de la técnica multicamino ECMP como son por paquete y por flujo.

Las métricas en este trabajo de titulación que se han utilizado son: rendimiento, retardo promedio y jitter promedio, las cuales se indican a continuación:

5.2.1 Rendimiento.

El rendimiento es la cantidad de datos que se pueden transmitir de un origen hacia un destino en un tiempo determinado, o el total de paquetes recibidos por el destino en una unidad de tiempo. El cálculo de esta métrica está dado por (Ángel, Diosdado, Marta, & Fernández, 2005):

$$Rendimiento = \frac{P_{Rx}}{T_{Tx}}$$

De donde:

P_{Rx} = Es el total de paquetes recibidos en el destino.

T_{Tx} = Es el tiempo de transmisión.

5.2.2 Retardo promedio.

El retardo promedio es el intervalo de tiempo entre la generación de paquetes en un nodo origen y la entrega de los mismos en el destino. Se puede decir que es el tiempo que tarda un paquete en cruzar toda la red desde su origen hacia el destino (Kumar Jha & Kharga, 2015).

El retardo promedio está dado por:

$$Retardo\ promedio = \frac{Sum_{retardo}}{P_{Rx}}$$

$Sum_{retardo}$ = Es la suma de todos los retardos de extremo a extremo.

P_{Rx} = Es el total de paquetes recibidos en el destino.

Con esta métrica se pretende verificar el retardo de un origen hacia un destino varía de acuerdo a la cantidad de routers que hay en el camino

5.2.3 Jitter promedio.

Es la variabilidad que sufre el retardo, es decir, el máximo y el mínimo retardo entre paquetes de la misma comunicación, su cálculo es dado por (Flow-Monitor in NS3, 2016):

$$jitter = \frac{jitter_{Sum}}{P_{Rx} - 1}$$

De donde:

$jitter_{Sum}$ = Es la suma de todos los valores de los jitter de extremo a extremo para todos los paquetes recibidos del flujo (este valor se lo obtiene directamente del flow monitor).

P_{Rx} = Es el total de paquetes recibidos en el destino.

5.3 Simulaciones.

Las simulaciones de los escenarios se los realiza en NS-3, este es un simulador de red de eventos discretos con licencia GNU GPLv2 de licencia libre, desarrollado en C++. En las simulaciones se hace uso del visualizador basado en Python que viene integrado en el simulador. Además, se incluyó las líneas de código necesarias para obtener un archivo xml que lo usamos para la obtención de datos junto con el software flow monitor para el análisis de los resultados.

5.3.1 Escenarios.

En esta sección se describen los escenarios a evaluar. Se han considerado dos escenarios con múltiples rutas de igual costo desde el origen al destino. El primer escenario aplicando el protocolo IP de un solo camino y el segundo escenario aplicando ECMP por paquete y por flujo. También se definen los parámetros de simulación, estos comprenden valores que tendrán ciertas variables que permitirán obtener diferentes datos en cada uno de los escenarios, dichos datos ayudan a realizar la comparación entre el protocolo IP de un solo camino y ECMP.

5.3.1.1 Escenario protocolo de enrutamiento de un solo camino.

El primer escenario consiste en usar un protocolo IP no multicamino, en una red con varios nodos, conectados entre sí y de igual costo para enviar paquetes de un origen hacia un destino. En este escenario se realizan dos pruebas que se describen a continuación:

- La primera, como se puede ver en la Figura 14, genera tráfico del origen en N0 variando su tasa de datos entre 2 - 10 Mbps hacia el destino en el que de forma correspondiente se varía el AB del último enlace N9 y N10 entre 2 - 10 Mbps. Los parámetros están definidos en la Tabla 2.

Tabla 2. Parámetros de simulación.

Parámetro	Valores
Tiempo de simulación	10 seg.
Número de nodos	11
Tamaño de paquete	500
Número de flujos	10
Tipo de tráfico	TCP
AB del enlace N9-N10	2, 4, 6, 8, 10 Mbps
AB de los enlaces intermedios	2 Mbps
Tasa de datos en N0 (origen)	2, 4, 6, 8, 10 Mbps
Protocolo de enrutamiento	IP no multicamino

Fuente: El autor.

Elaboración: El autor

En la Figura 14 se muestra el escenario con los parámetros de la Tabla 2 y el recorrido que tiene un paquete cuando se usa un protocolo no multicamino en una red que tiene varias rutas hacia el mismo destino.

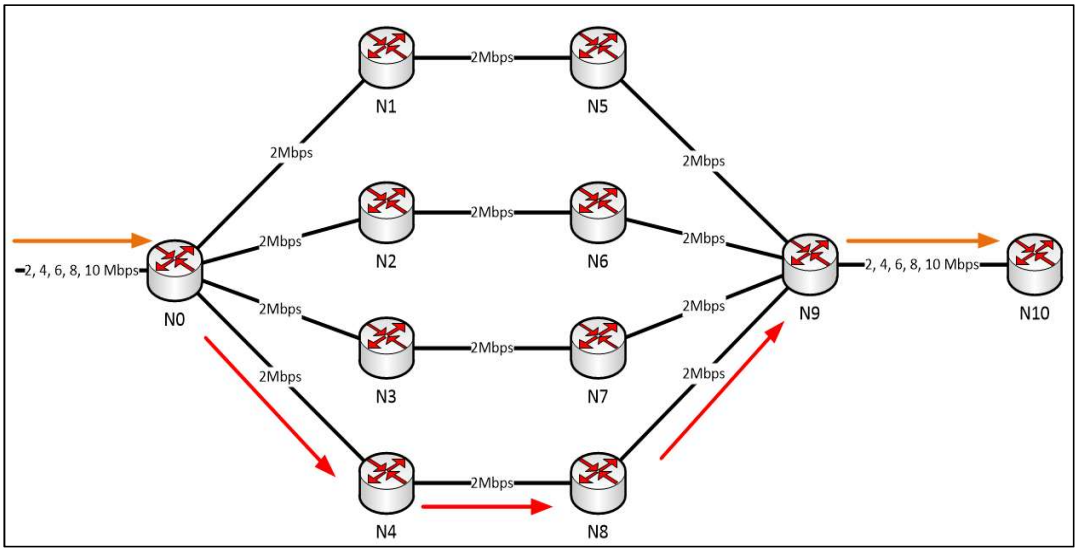

Figura 14. Protocolo no multicamino en una red con múltiples rutas de igual costo variando la tasa de datos en N0 (origen) y el AB en el último enlace (N9-N10).

Fuente: El autor

Elaboración: El autor

- La segunda prueba, ver Figura 15, la tasa de datos del tráfico de origen en N0 varía entre 2 - 10 Mbps mientras que el AB del último enlace entre N9 y N10 se mantiene constante con un valor de 10 Mbps. En la Tabla 3 se definen los parámetros de simulación para esta prueba.

Tabla 3. Parámetros de simulación.

Parámetro	**Valores**
Tiempo de simulación	10 seg.
Número de nodos	11
Tamaño de paquete	500
Número de flujos	10
Tipo de tráfico	TCP
AB del enlace entre N9-N10	10 Mbps
AB de los enlaces intermedios	2 Mbps
Data Rate en el N0 (origen)	2, 4, 6, 8, 10 Mbps
Protocolo de enrutamiento	IP no multicamino

Fuente: El autor.

Elaboración: El autor

La Figura 15 muestra el escenario utilizado con los parámetros descritos en la Tabla 3 y el recorrido que sigue el tráfico cuando se usa un protocolo de enrutamiento no multicamino.

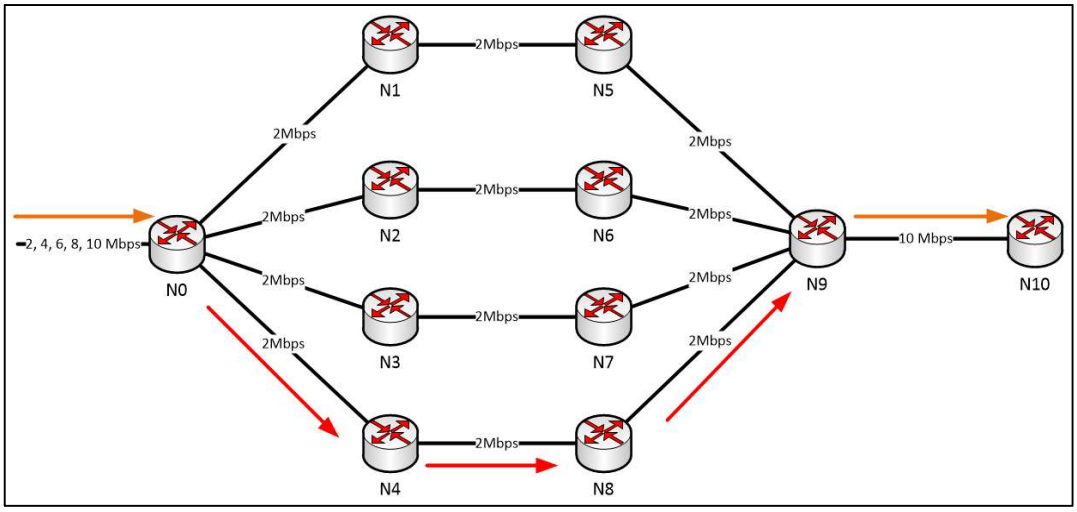

Figura 15. Protocolo no multicamino con tasa de datos variable en el origen N0 y AB constante en el último enlace (N9-N10)

Fuente: El autor

Elaboración: El autor

5.3.1.2 Escenario multicamino ECMP.

El segundo escenario consiste en utilizar la estrategia de tráfico multicamino ECMP por paquete y por flujo, en una red con diferentes nodos conectados entre sí para enviar paquetes TCP desde un origen hacia el destino a través de varias rutas de igual costo. Al igual que en el escenario 1, se realizan dos pruebas que consisten en:

- La primera, ver Figura 16, la tasa de datos en N0 varía entre 2 - 10 Mbps. El AB del último enlace entre N9 y N10 en forma correspondiente al tráfico varía en 2 - 10 Mbps. Esto con el fin de ver la capacidad de utilización de cada uno de los enlaces según vaya aumentando el tráfico de origen, así como el AB del enlace final, los enlaces intermedios se mantienen con un valor constante de 2 Mbps. En la Tabla 4 se describen los parámetros de simulación usados para esta prueba.

Tabla 4. Parámetros de simulación.

Parámetro	Valores
Tiempo de simulación	10 seg.
Número de nodos	11
Tamaño de paquete	500
Número de flujos	10
Tipo de tráfico	TCP
AB del enlace entre N9-N10	2, 4, 6, 8, 10 Mbps
AB de los enlaces intermedios	2 Mbps
Tasa de datos en N0 (origen)	2, 4, 6, 8, 10 Mbps
Técnica multicamino	ECMP por paquete y por flujo

Fuente: El autor.

Elaboración: El autor

En la Figura 16 se observa el escenario con los parámetros de la Tabla 4 y el recorrido que realizan los paquetes cuando se usa un enrutamiento multicamino en una red que posee varias rutas de igual costo desde el origen hacia el destino.

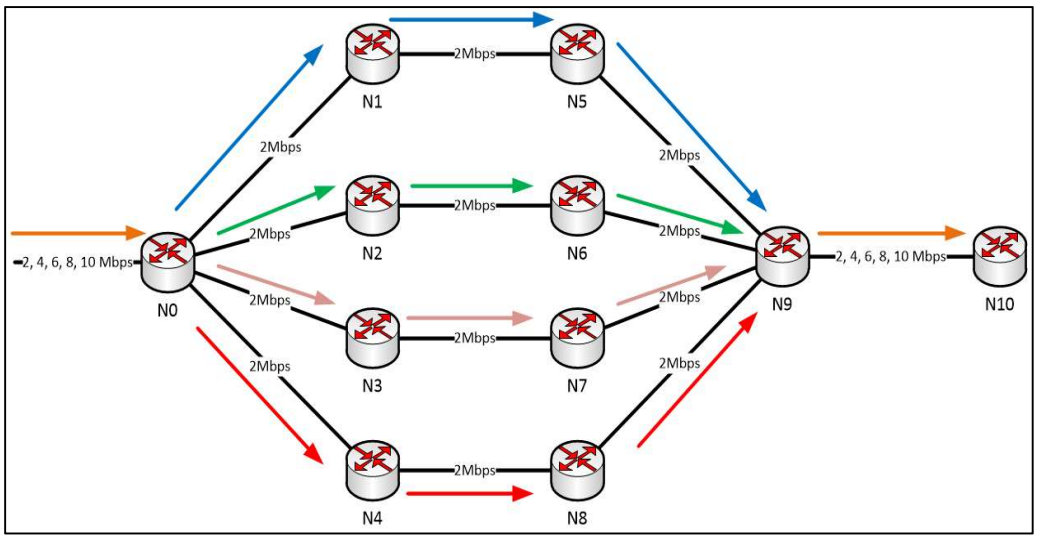

Figura 16. Escenario 2 con ECMP variando la tasa de datos de N0 (origen) y el AB del último enlace (N9-N10)

Fuente: El autor

Elaboración: El autor

- La segunda prueba, N0 genera tráfico desde el origen variando su tasa de datos entre 2 - 10 Mbps. Se mantiene constante el AB (10 Mbps) del último enlace entre N9 y N10, con esto se podrá comprobar la máxima capacidad de uso de los enlaces según como vaya incrementando el tráfico de origen hacia el destino, tomando en cuenta que el AB de los

enlaces intermedios siempre se mantienen con el mismo valor de 2 Mbps. La Tabla 5 de definen los parámetros de simulación para realizar esta prueba.

Tabla 5. Parámetros de simulación.

Parámetro	**Valores**
Tiempo de simulación	10 seg.
Número de nodos	11
Tamaño de paquete	500
Número de flujos	10
Tipo de tráfico	TCP
AB del enlace entre N9-N10	10 Mbps
AB de los enlaces intermedios	2 Mbps
Tasa de datos en N0 (origen)	2, 4, 6, 8, 10 Mbps
Técnica multicamino	ECMP por paquete y por flujo

Fuente: El autor.

Elaboración: El autor

En la Figura 17 se muestra el escenario con los parámetros de la tabla 5 y el recorrido que realiza el tráfico de origen a través de las múltiples rutas que tiene la red para llegar al destino.

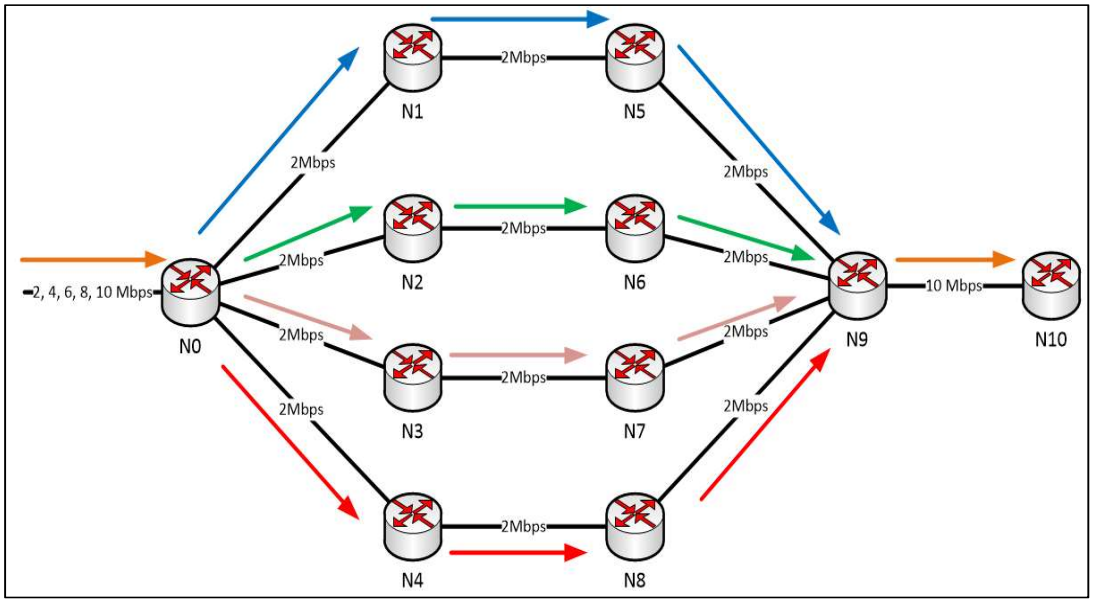

Figura 17. Escenario 2 con ECMP variando el Data Rate de origen y el AB constante en el enlace N9-N10

Fuente: El autor

Elaboración: El autor

CAPÍTULO VI
ANÁLISIS DE RESULTADOS

6. Análisis de resultados.

El análisis de resultados se lo realiza de acuerdo a las métricas definidas en la sección 5.2 y que se usan para evaluar la eficiencia de ECMP frente al protocolo IP de un solo camino.

6.1 Resultados de simulaciones.

A continuación, se muestran los datos obtenidos de las métricas que se usaron para comparar el desempeño de cada protocolo en los escenarios implementados.

6.1.1 Rendimiento

Tabla 6. Protocolo no multicamino, variando la tasa de datos de N0 y el AB entre N9-N10.

NO MULTICAMINO				
Tasa de datos en N0 (Mbps)	**AB entre N9-N10 (Mbps)**	**Paquetes Transmitidos**	**Paquetes Recibidos**	**Rendimiento (Kbps)**
2	2	4130	4130	1927,33
4	4	4118	4118	1929,04
6	6	4120	4120	1929,59
8	8	4120	4120	1929,86
10	10	4120	4120	1930,03

Fuente: El autor

Elaboración: El autor

En la Tabla 6 se aprecia los datos obtenidos mediante el software flow monitor[1] para el primer escenario cuando cambia tanto la tasa de datos en el origen N0 como el AB en el último enlace entre N9-N10, se puede ver que el rendimiento es similar cerca a los 2 Mbps esto debido a que se está usando un protocolo IP de un solo camino, es decir, el tráfico es enrutado por un solo camino para llegar al destino y la capacidad de AB de los enlaces intermedios de ese camino son de 2 Mbps.

[1] https://www.nsnam.org/docs/models/html/flow-monitor.html. Última revisión julio 2019.

Tabla 7. Protocolo no multicamino variando la tasa de datos en N0 y el AB constante entre N9-N10.

NO MULTICAMINO				
Tasa de datos en N0 (Mbps)	**AB entre N9-N10 (Mbps)**	**Paquetes Transmitidos**	**Paquetes Recibidos**	**Rendimiento (Kbps)**
2	10	4136	4136	1931,88
4		4120	4120	1931,91
6		4120	4120	1931,91
8		4120	4120	1931,91
10		4120	4120	1931,91

Fuente: El autor

Elaboración: El autor

La Tabla 7 muestra los datos obtenidos para el primer escenario cuando cambia la tasa de datos en N0 y se mantiene constante el AB en el último enlace, se puede ver que el rendimiento es similar, esto debido a que se usa un protocolo IP de un solo camino que enruta el tráfico de origen a través una sola ruta hacia el destino ocupando la máxima capacidad de los enlaces intermedios que son de 2 Mbps.

Tabla 8. ECMP por paquete variando la tasa de datos de N0 y el AB entre N9-N10

ECMP por paquete				
Tasa de datos en N0 (Mbps)	**AB entre N9-N10 (Mbps)**	**Paquetes Transmitidos**	**Paquetes Recibidos**	**Rendimiento (Kbps)**
2	2	429	429	1751,59
4	4	748	748	3163,29
6	6	1291	1291	5437,86
8	8	1410	1410	5989,2
10	10	1464	1464	6348,3

Fuente: El autor

Elaboración: El autor

En la Tabla 8 se aprecia los resultados obtenidos del rendimiento para el segundo escenario en el que se usa ECMP por paquete, en este caso la tasa de datos en el origen N0 y el AB en el último enlace cambia entre 2 – 10 Mbps, se puede ver que el rendimiento va en aumento de acuerdo al incremento de la tasa de datos en el origen y el AB en el último enlace. Cabe recordar que existen cuatro rutas disponibles entre el origen y el destino cada una con capacidad de 2 Mbps, de acuerdo a esto se hubiera esperado que el máximo rendimiento sea 8 Mbps sin embargo no se llega a tal debido a la congestión que se produce en N9 quien es el que recepta el tráfico de las cuatro rutas y lo reenvía a N10.

Tabla 9. ECMP por paquete variando la tasa de datos de N0 y el AB constante entre N9-N10

ECMP por paquete				
Tasa de datos (Mbps)	**AB entre N9-N10 (Mbps)**	**Paquetes Transmitidos**	**Paquetes Recibidos**	**Rendimiento (Kbps)**
2	10	453	453	1934,94
4		901	901	3884,27
6		1325	1325	5437,34
8		1483	1483	6232,52
10		1507	1507	6348,3

Fuente: El autor

Elaboración: El autor

En la Tabla 9 se aprecia los datos obtenidos para el segundo escenario aplicando ECMP por paquete, variando la tasa de datos del origen y manteniendo constante el AB del último enlace, se puede ver que el rendimiento aumenta mientras se incrementa la tasa de datos en el origen. Al igual que en el caso anterior, no se logra obtener el rendimiento máximo que teóricamente es 8 Mbps debido a la congestión en N9.

Tabla 10. Protocolo ECMP por flujo, variando la tasa de datos de N0 y el AB entre N9-N10.

ECMP por flujo				
Tasa de datos (Mbps)	**AB entre N9-N10 (Mbps)**	**Paquetes Transmitidos**	**Paquetes Recibidos**	**Rendimiento (Kbps)**
2	2	452	452	1734,02
4	4	899	899	3780,6
6	6	1306	1306	5842,13
8	8	1570	1570	6343,8
10	10	1688	1688	6840,24

Fuente: El autor

Elaboración: El autor

La Tabla 10 muestra los datos obtenidos para el segundo escenario aplicando ECMP por flujo, variando la tasa de datos del nodo origen N0 y el AB entre los enlaces N9-N10, se puede ver que el rendimiento se va incrementando de acuerdo el tráfico generado en el origen. En este escenario ECMP por flujo tiene un mejor rendimiento en comparación con ECMP por paquete.

Tabla 11. Protocolo ECMP por flujo, variando la tasa de datos de N0 y el AB constante entre N9-N10.

ECMP por flujo				
Tasa de datos (Mbps)	**AB entre N9-N10 (Mbps)**	**Paquetes Transmitidos**	**Paquetes Recibidos**	**Rendimiento (Kbps)**
2	10	453	453	1934,94
4		901	901	3870,97
6		1349	1349	5681,55
8		1570	1570	6344,37
10		1688	1688	6840,24

Fuente: El autor

Elaboración: El autor

En la Tabla 11 se aprecia los datos obtenidos para el segundo escenario utilizando ECMP por flujo cuando se mantiene constante el enlace entre N9-N10. El rendimiento en este escenario va aumentando de acuerdo al incremento del tráfico en el origen. El rendimiento máximo que alcanza es 6.8 Mbps, no llegando a la capacidad máxima disponible por los enlaces (8 Mbps) debido, como se explicó anteriormente, a la saturación que se produce en N9.

La Tabla 12 muestra la comparación del rendimiento de cada protocolo cuando la tasa de datos en N0 varía entre 2 - 10 Mbps. El AB del último enlace entre el N9 y N10 en forma correspondiente al tráfico varía en 2 - 10 Mbps.

Tabla 12. Comparación del rendimiento de los protocolos implementados.

Rendimiento (Kbps) variando la Tasa de datos en N0 y el AB en N9-10					
Estartegia	**Tasa de datos (2Mbps)**	**Tasa de datos (4Mbps)**	**Tasa de datos (6Mbps)**	**Tasa de datos (8Mbps)**	**Tasa de datos (10Mbps)**
No multicamino	1927,33	1929,04	1929,59	1929,86	1930,03
ECMP por paquete	1751,59	3163,29	5437,86	5989,20	6348,30
ECMP Por flujo	1734,02	3780,60	5842,13	6343,80	6840,24

Fuente: El autor

Elaboración: El autor

De acuerdo a los datos de la Tabla 12 se obtiene la gráfica del rendimiento de la Figura 18. El rendimiento obtenido es en el último nodo N10.

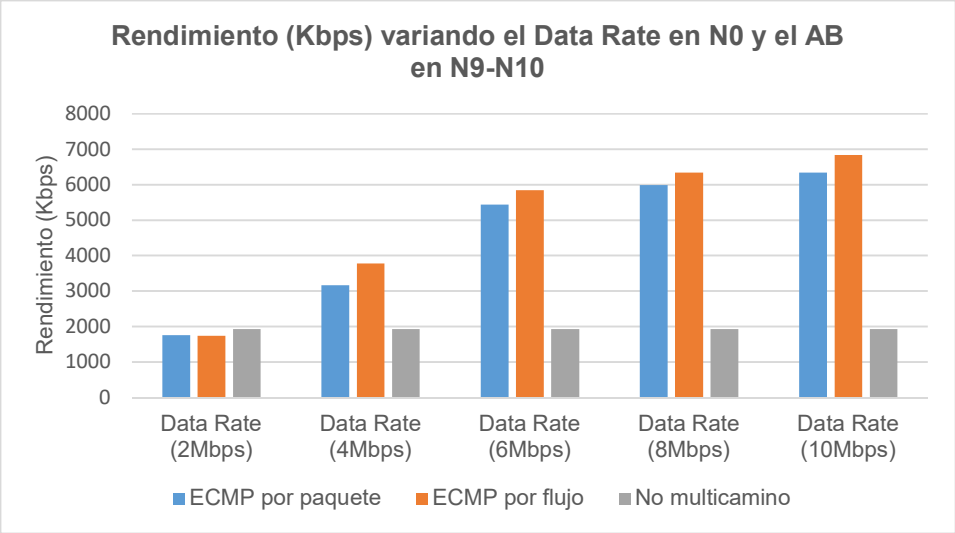

Figura 18. Tasa de datos vs Rendimiento

Fuente: El autor

Elaboración: El autor

En la Figura 18 se ve que el ECMP por flujo es el de mejor rendimiento cuando cambia tanto la tasa de datos de origen como el AB en el enlace de destino. El ECMP por flujo aprovecha un poco más la capacidad de los enlaces que tiene hacia el destino en comparación con el ECMP por paquete que no llega a saturar al máximo la capacidad de sus enlaces, debido a esto el rendimiento de ECMP por flujo es mayor.

La Tabla 13 muestra la comparación del rendimiento de cada protocolo implementado cuando varía la tasa de datos en N0 varía entre 2 - 10 Mbps y el AB del último enlace entre el N9 y N10 permanece constante (10 Mbps).

Tabla 13. Comparación de rendimiento de los protocolos implementados.

Rendimiento (Kbps) variando la tasa de datos en N0 y el AB en N9-10 fijo					
Protocolo	**Tasa de datos (2Mbps)**	**Tasa de datos (4Mbps)**	**Tasa de datos (6Mbps)**	**Tasa de datos (8Mbps)**	**Tasa de datos (10Mbps)**
No multicamino	1931,88	1931,91	1931,91	1931,91	1931,91
ECMP por paquete	1934,94	3884,27	5437,34	6232,52	6348,30
ECMP por flujo	1934,94	3870,97	5681,55	6344,37	6840,24

Fuente: El autor

Elaboración: El autor

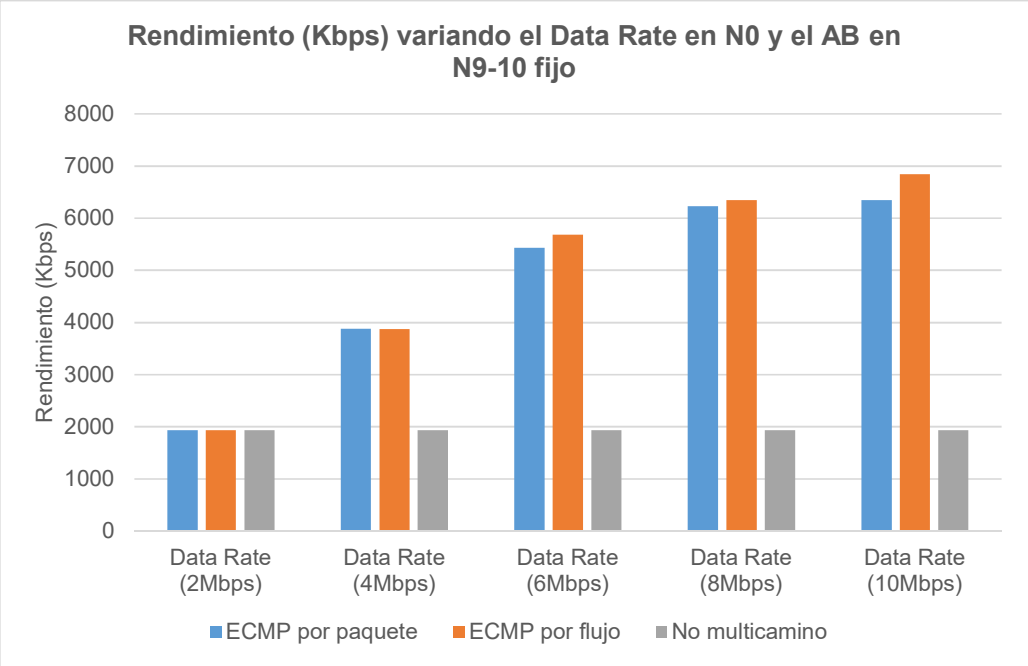

Figura 19. Tasa de datos vs Rendimiento, variando la tasa de datos y el AB constante entre N9-N10.

Fuente: El autor

Elaboración: El autor

En la gráfica de la Figura 19 se observa que, ECMP por paquete y por flujo tienen un comportamiento similar en cuanto al uso de la capacidad de sus enlaces, cuando cambia la tasa de datos de origen entre 2 - 10 Mbps y el AB es constante en el enlace entre N9-N10, ECMP por flujo tiene un mejor rendimiento cuando la tasa de datos de origen es 10 Mbps igual al AB fijado en el último enlace entre N9-N10.

Tanto en la gráfica de la Figura 18 como en la 19 se puede apreciar que, a partir de la tasa de datos de 8 Mbps, en el destino no se logra obtener el mismo rendimiento que el tráfico generado en origen, teniendo en cuenta que en la topología los 4 enlaces intermedios tienen un AB de 2 Mbps, esto debido a la congestión que se produce en el nodo N9 que recepta todo el tráfico de los enlaces provenientes del origen.

6.1.2 Retardo promedio.

Tabla 14. Protocolo no multicamino, variando la tasa de datos de N0 y el AB entre N9-N10

NO MULTICAMINO				
Tasa de datos (Mbps)	**AB entre N9-N10 (Mbps)**	**Paquetes Transmitidos**	**Paquetes Recibidos**	**Retardo promedio (ms)**
2	2	4130	4130	436,618
4	4	4118	4118	518,241
6	6	4120	4120	518,601
8	8	4120	4120	518,766
10	10	4120	4120	518,865

Fuente: El autor

Elaboración: El autor

En la Tabla 14 se observa que el retardo promedio para el primer escenario, variando la tasa de datos en el origen y el AB en el enlace entre N9-N10, se va incrementando conforme se incrementa la tasa de datos del origen, esto debido a que el protocolo no multicamino usa una sola ruta para enviar el tráfico generado en el origen llegando a saturar el enlace.

Tabla 15. Protocolo no multicamino, variando la Tasa de datos de N0 y el AB constante entre N9-N10.

NO MULTICAMINO				
Tasa de datos (Mbps)	**AB entre N9-N10 (Mbps)**	**Paquetes Transmitidos**	**Paquetes Recibidos**	**Retardo promedio (ms)**
2	10	4136	4136	430,612
4		4120	4120	518,486
6		4120	4120	518,77
8		4120	4120	518,916
10		4120	4120	519,001

Fuente: El autor

Elaboración: El autor

En la Tabla 15 se aprecia los resultados del retardo promedio para el primer escenario manteniendo constante el AB del enlace entre N9-N10, al igual que en el caso anterior el retardo promedio aumenta de acuerdo al incremento de la tasa de datos en el origen.

Tabla 16. Protocolo ECMP por paquete variando la Tasa de datos de N0 y el AB entre N9-N10

ECMP por paquete				
Tasa de datos (Mbps)	**AB entre N9-N10 (Mbps)**	**Paquetes Transmitidos**	**Paquetes Recibidos**	**Retardo promedio (ms)**
2	2	429	429	263,031
4	4	748	748	63,982
6	6	1291	1291	30,9045
8	8	1410	1410	19,2382
10	10	1464	1464	19,5377

Fuente: El autor

Elaboración: El autor

La Tabla 16 muestra los datos obtenidos para el segundo escenario con ECMP por paquete variando la tasa de datos en N0 y el AB entre N9-N10, aquí el retardo es menor cuando existe un mayor tráfico de origen, esto debido a que ECMP por paquete envía el tráfico por las rutas disponibles tratando de distribuirlo de forma equitativa y la capacidad de estas rutas va aumentando de acuerdo a la cantidad de tráfico a transmitir.

Tabla 17. Protocolo ECMP por paquete variando la tasa de datos de N0 y el AB constante entre N9-N10

ECMP por paquete				
Tasa de datos (Mbps)	**AB entre N9-N10 (Mbps)**	**Paquetes Transmitidos**	**Paquetes Recibidos**	**Retardo promedio (ms)**
2	10	453	453	11,25
4		901	901	11,89
6		1325	1325	19,92
8		1483	1483	18,97
10		1507	1507	19,05

Fuente: El autor

Elaboración: El autor

En la Tabla 17 se observa el retardo para el segundo escenario con ECMP por paquete manteniendo constante el AB entre N9-10, aquí el retardo va en aumento debido a que el origen aumenta su tasa de datos a transmitir.

Tabla 18. Protocolo ECMP por flujo, variando la tasa de datos de N0 y el AB entre N9-N10.

ECMP por flujo				
Tasa de datos (Mbps)	**AB entre N9-N10 (Mbps)**	**Paquetes Transmitidos**	**Paquetes Recibidos**	**Retardo promedio (ms)**
2	2	452	452	1159,54
4	4	899	899	937,759
6	6	1306	1280	640,184
8	8	1570	1570	1243,51
10	10	1688	1688	1272,4

Fuente: El autor

Elaboración: El autor

La Tabla 18 muestra el retardo generado en el escenario 2 aplicando ECMP por flujo, variando la tasa de datos en el origen y el AB entre N9-N10, se ve que el retardo es diferente entre las diferentes tasas de datos generados en el origen esto debido a que ECMP por flujo clasifica los paquetes por flujo para enviarlos por la misma ruta, llegando en ocasiones a saturar una sola ruta debido al tamaño del flujo.

Tabla 19. Protocolo ECMP por flujo, variando la Tasa de datos de N0 y el AB constante entre N9-N10.

ECMP por flujo				
Tasa de datos (Mbps)	**AB entre N9-N10 (Mbps)**	**Paquetes Transmitidos**	**Paquetes Recibidos**	**Retardo promedio (ms)**
2	10	453	453	11,24
4		901	901	13,402
6		1349	1349	209,361
8		1570	1570	1243,43
10		1688	1688	1272,64

Fuente: El autor

Elaboración: El autor

La Tabla 19 se observa el retardo promedio para el segundo escenario con ECMP por flujo manteniendo constante el AB entre N9-N10, se puede apreciar que el retardo se incrementa a medida que se aumenta la tasa de datos a transmitir desde el origen, esto debido a que ECMP por flujo trata de enviar los paquetes de un mismo flujo por la misma ruta.

En la Tabla 20 se observa la comparación del retardo promedio de los protocolos usados cuando cambia la tasa de datos del nodo origen N0 entre 2 - 10 Mbps y de forma correspondiente cambia el AB del enlace entre el N9-N10.

Tabla 20. Comparación del retardo de los protocolos implementados

Retardo promedio (ms) variando la tasa de datos en N0 y el AB en N9-10					
Protocolo	**Tasa de datos (2Mbps)**	**Tasa de datos (4Mbps)**	**Tasa de datos (6Mbps)**	**Tasa de datos (8Mbps)**	**Tasa de datos (10Mbps)**
No multicamino	436,618	518,241	518,601	518,766	518,865
ECMP por paquete	263,031	63,982	30,905	19,238	19,538
ECMP por flujo	1159,5400	937,7590	640,1840	1243,5100	1272,4000

Fuente: El autor

Elaboración: El autor

La Figura 20 muestra en forma gráfica los datos del retardo promedio que se tiene en la Tabla 20.

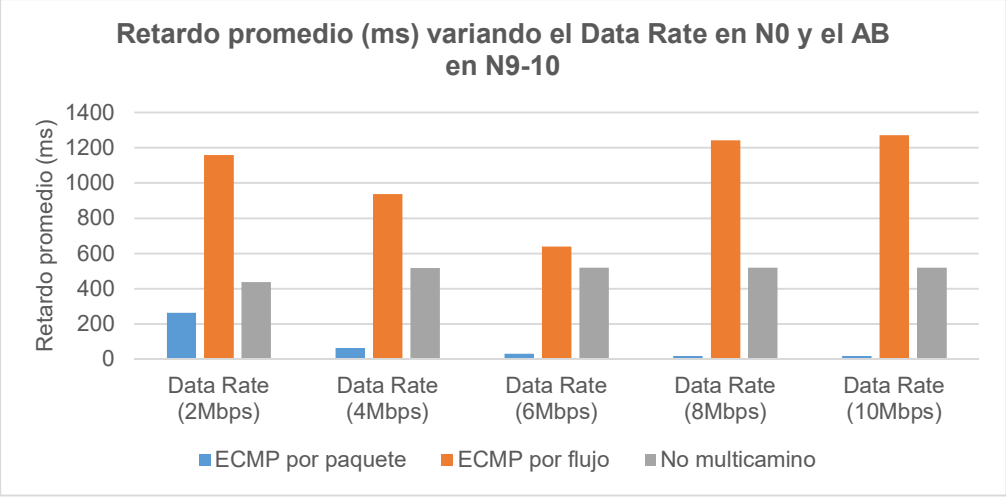

Figura 20. Comparación tasa de datos vs retardo promedio

Fuente: El autor

Elaboración: El autor

En la Figura 20 se muestra el resultado del retardo cuando cambia la tasa de datos en el origen N0 y de forma correspondiente cambia el AB entre los nodos N9-N10, en este caso el retardo de ECMP por flujo es diverso, siendo mayor que ECMP por paquete y mayor que el protocolo no multicamino, esto debido a que los paquetes que pertenecen a un flujo siempre son enviados por la misma ruta.

La Tabla 21 muestra la comparación del retardo de cada protocolo implementado cuando la *tasa de datos* en el N0 varía entre 2 Mbps, 4 Mbps, 6 Mbps, 8 Mbps y 10 Mbps y el AB del último enlace entre el N9 y N10 permanece constante (10 Mbps).

Tabla 21. Comparación del retardo de los protocolos implementados

Retardo promedio (ms) variando la tasa de datos en N0 y el AB en N9-10 fijo					
Protocolo	**Tasa de datos (2Mbps)**	**Tasa de datos (4Mbps)**	**Tasa de datos (6Mbps)**	**Tasa de datos (8Mbps)**	**Tasa de datos (10Mbps)**
No multicamino	430,612	518,486	518,770	518,916	519,001
ECMP por paquete	11,250	11,890	19,920	18,970	19,050
ECMP por flujo	11,240	13,402	209,361	1243,430	1272,640

Fuente: El autor

Elaboración: El autor

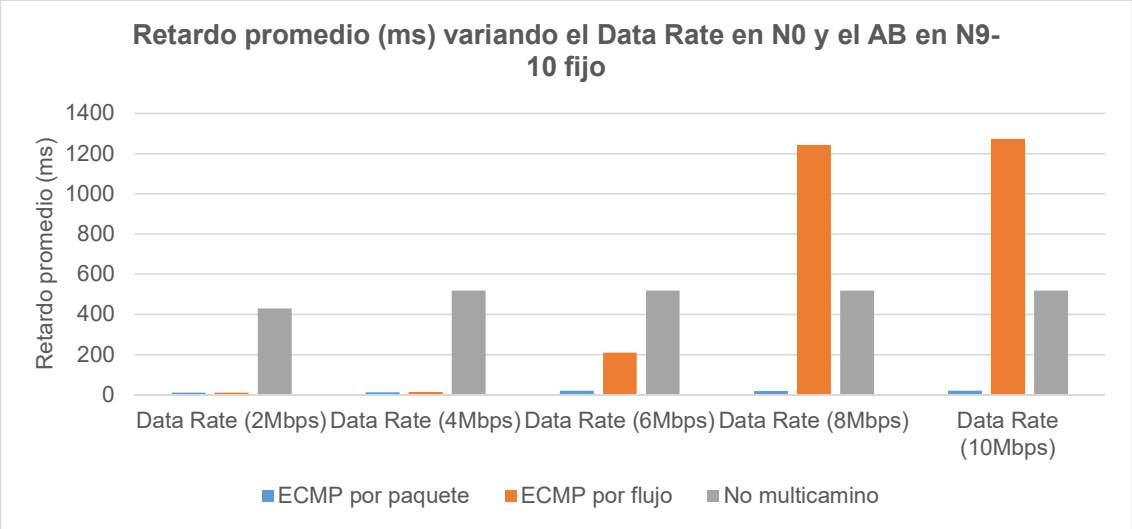

Figura 21. Comparación tasa de datos vs retardo promedio
Fuente: El autor
Elaboración: El autor

En la gráfica de la Figura 21 se puede ver el retardo cuando la Tasa de datos cambia en el origen de 2 - 10 Mbps y el AB entre N9-N10 es constante, en este caso el que menor retardo tiene es el ECMP por paquete y el ECMP por flujo tiene un mayor retardo a partir de 6 Mbps de tasa de datos en el origen, esto puede ser debido a que en este modo los paquetes son clasificados por flujo para ser enviados por un solo camino.

6.1.3 Jitter

Tabla 22. Protocolo no multicamino, variando la Tasa de datos de N0 y el AB entre N9-N10

NO MULTICAMINO				
Tasa de datos (Mbps)	**AB entre N9-N10 (Mbps)**	**Paquetes Transmitidos**	**Paquetes Recibidos**	**Jitter (ns)**
2	2	4130	4130	0,0026
4	4	4118	4118	0,00234
6	6	4120	4120	0,00235
8	8	4120	4120	0,00236
10	10	4120	4120	0,00237

Fuente: El autor

Elaboración: El autor

La Tabla 22 muestra los datos obtenidos para el primer escenario usando un protocolo IP no multicamino cuando se cambia la tasa de datos en N0 y el AB entre N9-N10, se ve que el jitter se incrementa conforme va aumentando la tasa de datos a transmitir, debido a que el protocolo usa un solo camino para enviar el tráfico.

Tabla 23. Protocolo no multicamino, variando la tasa de datos de N0 y el AB constante entre N9-N10.

NO MULTICAMINO				
Tasa de datos (Mbps)	**AB entre N9-N10 (Mbps)**	**Paquetes Transmitidos**	**Paquetes Recibidos**	**Jitter (ns)**
2	10	4136	4136	0,0021
4		4120	4120	0,0022
6		4120	4120	0,0022
8		4120	4120	0,0022
10		4120	4120	0,0022

Fuente: El autor

Elaboración: El autor

En la Tabla 23 se aprecia los datos obtenidos para el primer escenario manteniendo constante el AB en el enlace entre N9-10, se observa que el jitter es similar en las distintas tasas de datos, al igual que el caso anterior el protocolo usa un solo camino para enviar el tráfico.

Tabla 24. Protocolo ECMP por paquete variando la tasa de datos de N0 y el AB entre N9-N10

ECMP por paquete				
Tasa de datos (Mbps)	**AB entre N9-N10 (Mbps)**	**Paquetes Transmitidos**	**Paquetes Recibidos**	**Jitter (ns)**
2	2	429	429	0,0047
4	4	748	748	0,00255
6	6	1291	1291	0,00373
8	8	1410	1410	0,00634
10	10	1507	1507	0,00636

Fuente: El autor

Elaboración: El autor

La Tabla 24 muestra los datos de la simulación para el segundo escenario usando ECMP por paquete, se observa que el jitter es mayor cuando la tasa de datos en el origen aumenta. Esto debido a la congestión que se puede presentar en los enlaces.

Tabla 25. Protocolo ECMP por paquete variando la tasa de datos de N0 y el AB constante entre N9-N10

ECMP por paquete				
Tasa de datos (Mbps)	**AB entre N9-N10 (Mbps)**	**Paquetes Transmitidos**	**Paquetes Recibidos**	**Jitter (ns)**
2	10	453	453	0,0151
4		901	901	0,00917
6		1325	1325	0,00668
8		1483	1483	0,00625
10		1507	1507	0,00653

Fuente: El autor

Elaboración: El autor

En la Tabla 25 se observa el jitter para el segundo escenario cuando se mantiene constante el AB entre N9-N10, para este caso el jitter tiene diferente comportamiento para cada tasa de datos transmitido.

Tabla 26. Protocolo ECMP por flujo, variando la tasa de datos de N0 y el AB entre N9-N10.

ECMP por flujo				
Tasa de datos (Mbps)	**AB entre N9-N10 (Mbps)**	**Paquetes Transmitidos**	**Paquetes Recibidos**	**Jitter (ns)**
2	2	452	452	0,00999
4	4	899	899	0,00288
6	6	1306	1280	0,00305
8	8	1570	1570	0,00325
10	10	1688	1688	0,00334

Fuente: El autor

Elaboración: El autor

En la Tabla 26 se puede apreciar los datos obtenidos para el segundo escenario variando la tasa de datos en el origen y el AB entre N9-N10, se observa que el jitter toma diferentes valores de acuerdo al incremento de la tasa de datos, esto debido a que los paquetes son clasificados y enviados por la misma ruta de acuerdo al flujo al que pertenecen, es decir si un flujo es mayor que otro puede llegar a saturar la ruta.

Tabla 27. Protocolo ECMP por flujo, variando la Tasa de datos de N0 y el AB constante entre N9-N10.

ECMP por flujo				
Tasa de datos (Mbps)	**AB entre N9-N10 (Mbps)**	**Paquetes Transmitidos**	**Paquetes Recibidos**	**Jitter (ns)**
2	10	453	453	0,0131
4		901	901	0,00257
6		1349	1349	0,00512
8		1570	1570	0,00324
10		1688	1688	0,00334

Fuente: El autor

Elaboración: El autor

La Tabla 27 se observa el jitter para el segundo escenario utilizando ECMP por flujo manteniendo el AB contante entre N9-N10, aquí el jitter se va incrementando de acuerdo al aumento de la tasa de datos.

En la Tabla 28 se observa la comparación del jitter de los protocolos usados, cuando se cambia la tasa de datos de N0 entre 2 - 10 Mbps y a su vez cambia el AB del enlace entre N9-N10 en 2 - 10 Mbps.

Tabla 28. Comparación del retardo de los protocolos implementados

Jitter variando la tasa de datos en N0 y el AB en N9-10					
Estrategia	**Tasa de datos (2Mbps)**	**Tasa de datos (4Mbps)**	**Tasa de datos (6Mbps)**	**Tasa de datos (8Mbps)**	**Tasa de datos (10Mbps)**
No multicamino	0,0026	0,00234	0,00235	0,00236	0,00237
ECMP por paquete	0,0047	0,00255	0,00373	0,00634	0,00636
ECMP por flujo	0,00999	0,00288	0,00305	0,00325	0,00334

Fuente: El autor

Elaboración: El autor

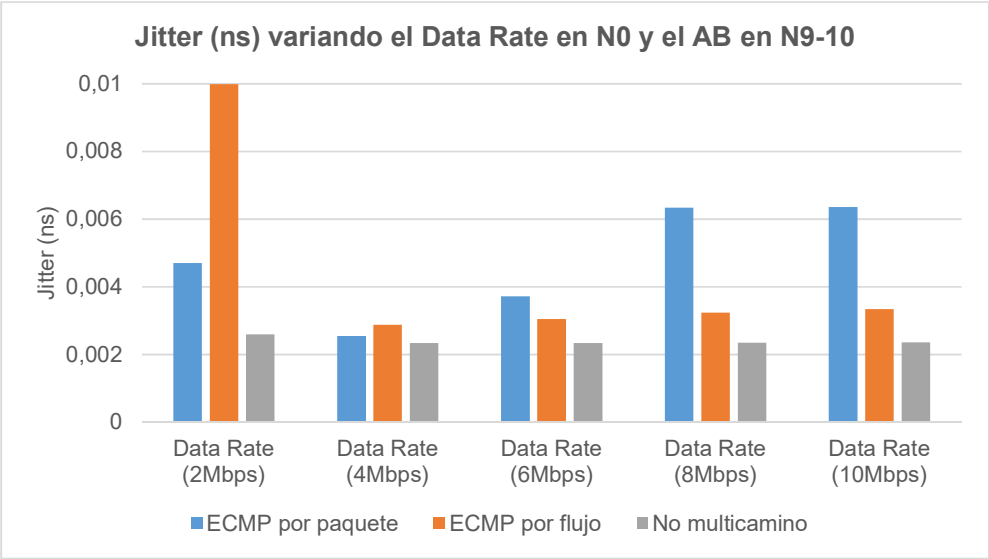

Figura 22. Comparación tasa de datos vs jitter
Fuente: El autor
Elaboración: El autor

En la Figura 22 se muestra el jitter cuando se cambia la Tasa de datos de N0 y el AB del enlace entre N9-N10 en 2 Mbps, 4 Mbps, 6Mbps, 8 Mbps, 10 Mbps, en la gráfica se ve que el jitter en los dos modos de ECMP por paquete y por flujo es mayor que el del protocolo no multicamino.

En la Tabla 29 se observa la comparación del jitter de los protocolos en los escenarios implementados, cuando se cambia la Tasa de datos de N0 entre 2 Mbps, 4 Mbps, 6Mbps, 8 Mbps, 10 Mbps y se mantiene constante el AB del enlace entre N9-N10 en 10 Mbps.

Tabla 29. Comparación del retardo de los protocolos implementados

Jitter (ns) variando la tasa de datos en N0 y el AB en N9-10 fijo					
Estartegia	**Tasa de datos (2Mbps)**	**Tasa de datos (4Mbps)**	**Tasa de datos (6Mbps)**	**Tasa de datos (8Mbps)**	**Tasa de datos (10Mbps)**
No multicamino	0,0021	0,0022	0,0022	0,0022	0,0022
ECMP Por paquete	0,0151	0,00917	0,00668	0,00625	0,0653
ECMP por flujo	0,0131	0,00257	0,00512	0,00324	0,00334

Fuente: El autor

Elaboración: El autor

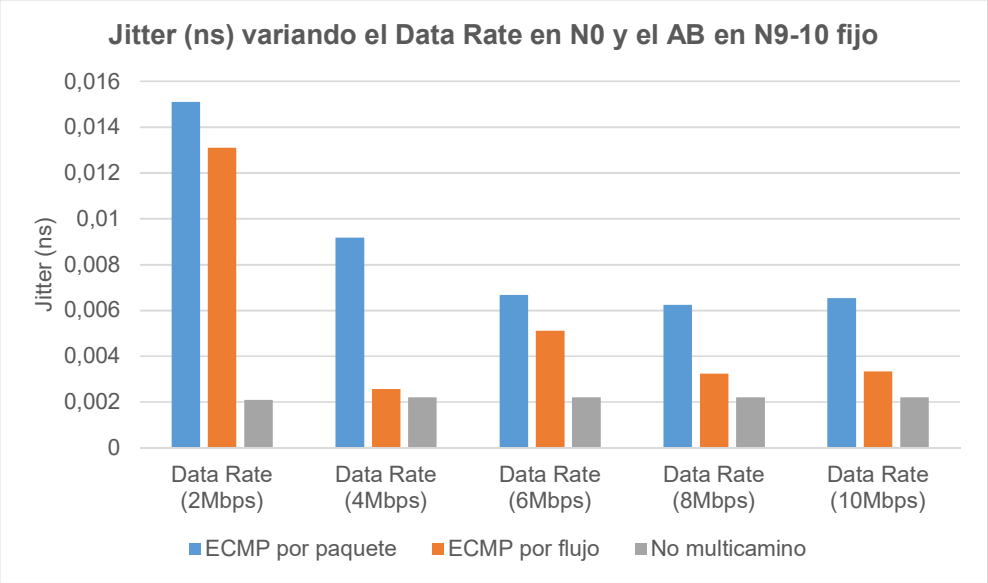

Figura 23. Comparación tasa de datos vs jitter
Fuente: El autor
Elaboración: El autor

En la Figura 23 se observa gráfica del jitter de los protocolos en los escenarios implementados, cuando se cambia la Tasa de datos de N0 entre 2 Mbps, 4 Mbps, 6Mbps, 8 Mbps, 10 Mbps y se mantiene constante el AB del enlace entre N9-N10 en 10 Mbps, en esta prueba el jitter de ECMP por paquete es mucho mayor que la de los otros dos.

CONCLUSIONES

- Se realizó el análisis de un algoritmo multicamino Equal Cost Multipath (ECMP) para redes de datos y se implementó en el simulador NS-3.
- Una vez implementado ECMP se desarrolló pruebas en dos escenarios, el primero variando la tasa de datos en el origen y el AB en el último enlace del destino, el segundo variando la Tasa de datos en el origen y manteniendo constante el AB del último enlace en el destino, con el que se definió el comportamiento del algoritmo multicamino en NS-3.
- Mediante el simulador NS-3 se compararon los dos modos de ECMP por paquete y por flujo con un protocolo IP de un solo camino.
- El ECMP por flujo es el que mejor rendimiento tiene en los escenarios implementados, trata de ocupar al máximo la capacidad de los enlaces disponibles hacia el destino.
- En la métrica de retardo, el ECMP por paquete es el que menor retardo posee en los dos escenarios implementados, además distribuye de forma más equitativa el tráfico, pero con una secuencia incorrecta de los paquetes que llegan al destino.
- El jitter del ECMP por paquete y por flujo son mayores que el del protocolo de un solo camino. Aunque el valor de esta métrica es mínimo (ns) y no tendría mayor efecto en el rendimiento.
- En una red multicamino de igual costo se aconseja usar ECMP por flujo ya que el mismo flujo de datos se envía por una misma ruta garantizando así la secuencia de los paquetes en el destino.
- En NS-3 se comprobó que la capacidad de utilización (ancho de banda) de los canales hacia el destino depende del tráfico de origen generado.

RECOMENDACIONES

- Antes de empezar con la experimentación en NS-3, se debe tomar en cuenta la documentación y manuales brindados por el sitio oficial de NS-3, para de esta manera trabajar adecuadamente durante la integración de los diferentes métodos empleados.

- Para trabajar con el simulador NS-3 se recomienda utilizar un entorno de desarrollo como Eclipse, el cual facilita el análisis de la estructura y funcionamiento de NS-3.

- Se recomienda usar un visualizador en NS-3 que puede ser NetAnim o PyViz para observar gráficamente cómo se comporta el escenario en el que se está trabajando

- Sería importante realizar la implementación de ECMP en un entorno real, es decir implementarlo en equipos que soporten esta técnica de ingeniería de tráfico para tener una idea más clara de su funcionamiento y de las ventajas que brinda una vez puesto en marcha en algo real.

BIBLIOGRAFÍA

Alvarez-Horcajo, J., Lopez-Pajares, D., Arco, J. M., Carral, J. A., & Martinez-Yelmo, I. (2017). TCP-path: Improving load balance by network exploration. *Proceedings of the 2017 IEEE 6th International Conference on Cloud Networking, CloudNet 2017*, 0–5. https://doi.org/10.1109/CloudNet.2017.8071533

Ángel, M., Diosdado, T., Marta, P., & Fernández, L. (2005). *Sistema de Autoreconfiguración para Redes Ad Hoc*. Retrieved from http://eprints.ucm.es/9004/1/Sistema_de_Autoconfiguracion_para_Redes_Ad_Hoc.pdf

Ari Lappetelainen. (2014). Equal Cost Multipath Routing in IP Networks. *Faculty of Electronics, Communications and Automation*, *4*(1), 272–276.

Barreira, D., and Fernando Mira da Silva. Multipath tcp protocols. 2014. *Multipath TCP Protocols*. 1–10. https://pdfs.semanticscholar.org/6fd0/09bdf0a0ddc79a636d69fff26bbe488aa692.pdf

Chihani, B., & Denis, C. (2011). *A Multipath TCP model for ns-3 simulator*. (December 2011). Retrieved from http://arxiv.org/abs/1112.1932

Cho Aye, M., & Aung, A. M. (2014). Energy Efficient Multipath Routing for Mobile Ad Hoc Networks. *International Journal of Information Technology, Modeling and Computing*, *2*(3), 11–18. https://doi.org/10.5121/ijitmc.2014.2302

Cisco. (2018). *ECMP Load Balancing*. 1–12. Retrieved from www.cisco.com/go/cfn.%0Ahttps://www.cisco.com/c/en/us/td/docs/ios-xml/ios/mp_l3_vpns/configuration/xe-3s/asr903/mp-l3-vpns-xe-3s-asr903-book/mp-l3-vpns-xe-3s-asr903-book_chapter_0100.html

Cisco. MPLS: Layer 3 VPN's Configuration Guide, Cisco IOS XE Release 3S (Cisco ASR 900 Series).

Coudron, M., & Secci, S. (2017). An implementation of multipath TCP in ns3. *Computer Networks*, *116*, 1–11. https://doi.org/10.1016/j.comnet.2017.02.002

De, U., & De, N. Y. (2013). *Análisis del rendimiento de técnicas multi- camino y el protocolo MPTCP sobre redes inalámbricas multi-salto*.

Deepinder, E., Wadhwa, S., Tripatjot, E., & Panag, S. (2011). *Performance Comparison of Single and Multipath Routing Protocols in Adhoc Networks*. *2*(5), 1486–1496.

ECMP operation in global routing. (2017). https://www.nsnam.org/bugzilla/show_bug.cgi?id=667

Farrugia, N., Buttigieg, V., & Briffa, J. A. (2018). A globally optimised multipath routing algorithm using SDN. *21st Conference on Innovation in Clouds, Internet and Networks, ICIN 2018*, 1–8. https://doi.org/10.1109/ICIN.2018.8401633

Flow-Monitor in NS3. A Perfect Tool For Evaluation Purposes. (Throughput, Delay, Jitter, Packet Loss and Other Parameters in NS3). (2016). http://mighthelpyou.blogspot.com/2016/04/flow-monitor-in-ns3-perfect-tool-for.html

Garrido Ortiz, Pablo. (2013). Análisis del rendimiento de técnicas multicamino y el protocolo mptcp sobre redes inalámbricas multi-salto. https://repositorio.unican.es/xmlui/bitstream/handle/10902/3144/357933.pdf?sequence=1&isAllowed=y

Hameed, Saif Saad. Evaluation of Equal Cost Multipath "ECMP" Design and Implementation. (2018). *International Journal of Applied Engineering Research*, *13*(02), 1095–1100.

Huawei, Per-Flow and Per-Packet Load Balancing. (2019). https://support.huawei.com/enterprise/en/doc/EDOC1100055041/ebc8ad42/per-flow-and-per-packet-load-balancing

James F. Kurose, Keith W. Ross. (2017). *Redes de Computadoras. Un enfoque descendente.* Madrid, Pearson Educación.

Kheirkhah Sabetghadam, M. (2016). *MMPTCP : a novel transport protocol for data centre networks*. Retrieved from https://ezp.lib.unimelb.edu.au/login?url=https://search.ebscohost.com/login.aspx?direct=true&db=edsble&AN=edsble.690455&site=eds-live&scope=site

Li, Z. Y., Wang, R. C., & Bi, J. L. (2009). A multipath routing algorithm based on traffic prediction in wireless mesh networks. *5th International Conference on Natural Computation, ICNC 2009*, *6*, 115–119. https://doi.org/10.1109/ICNC.2009.43

Network Working Group. 2000. Analysis of an Equal-Cost Multi-Path Algorithm. https://tools.ietf.org/html/rfc2992

Onakome, Utejiro., & Okah-Avae. (2014). *Simulation Study of Aqm Schemes Over Large Scale Networks in Data Centers Using Ns-3*.

Piris Ruiz, Jaime. (2015). Algoritmo para la búsqueda de múltiples rutas sobre redes inalámbricas malladas multiinterfaz. https://repositorio.unican.es/xmlui/bitstream/handle/10902/7732/379707.pdf?sequence=1&isAllowed=y

Rabadan Cebolla, Carlos María. (2013). Algoritmos para el estudio de técnicas de multipath y network-coding en redes inalámbricas multi-salto. https://repositorio.unican.es/xmlui/bitstream/handle/10902/3102/Carlos%20Maria%20Rabadan%20Cebolla.pdf?sequence=1&isAllowed=y

Redes. Tema3: La capa de enlace. Universidad de Oviedo. http://www.isa.uniovi.es/docencia/redes/Apuntes/tema3.pdf

Rodríguez Estévez, B. (2017). *Desarrollo de un sistema de comunicaciones punto a punto con ns3*. Retrieved from https://idus.us.es/xmlui/handle/11441/65496

Sangi, A. R., Liu, J., & Liu, Z. (2010). Performance comparison of single and multi-path routing protocol in MANET with selfish behaviors. *World Academy of Science, Engineering and Technology*, *65*, 28–32.

Su, X., & de Veciana, G. (2007). Dynamic multi-path routing. *ACM SIGMETRICS Performance Evaluation Review*, *29*(1), 25–36. https://doi.org/10.1145/384268.378426

Sumathi, R., & Srinivas, M. G. (2012). A survey of qos based routing protocols for wireless sensor networks. *Journal of Information Processing Systems*, 8(4):589–602. http://jips-k.org/q.jips?cp=pp&pn=241

Tanenbaum, Andrew S., & Wetherall. David J. (2012). *Redes de computadoras*, México, Pearson Educación.

Tekaya, M., Tabbane, N., & Tabbane, S. (2010). Multipath Routing with Load Balancing and QoS in Ad hoc Network. *IJCSNS International Journal of Computer Science and Network Security*, *10*(8), 67–72.

Tsai, J., & Moors, T. (2006). A review of multipath routing protocols: From wireless ad hoc to mesh networks. *Workshop on Wireless Multihop Networking*, *70*, 1–6. https://doi.org/10.1007/s11277-012-0723-2

Tutorial. A first ns-3 Script. (2008). https://www.nsnam.org/docs/release/3.2/tutorial/tutorial_18.html

Yahya, B., & Ben-othman, J. (2009). *REER : Robust and Energy Efficient Multipath Routing Protocol for Wireless Sensor Networks*.

Ye, K., & Zhou, J. (2013). *Simulation of the Global Routing Protocol based on NS-3*. 112–115. https://doi.org/10.2991/icsecs-13.2013.23

ANEXOS

ANEXO 1. Estructura del script ECMP

En este Anexo 1 se hace una descripción del código fuente (generado por el parche) usado para la simulación de ECMP en sus dos modos (por paquete y por flujo).

1.1 Módulos

El código generado por el parche consta de los siguientes módulos para el funcionamiento de ECMP.

```
#include <iostream>
#include <fstream>
#include <string>
#include <cassert>
#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/applications-module.h"
#include "ns3/point-to-point-module.h"
#include "ns3/internet-module.h"
#include "ns3/flow-monitor-module.h"
```

1.2 Espacio de nombre NS-3 (namespace)

Esta línea se encarga de agrupar todas las declaraciones relacionadas con NS-3 en un ámbito fuera del espacio de nombres global ayudando con la integración con otro código, con esta declaración no se tiene que escribir el operador *ns3::scope* (Tutorial NS3). A continuación, su definición:

```
using namespace ns3;
```

1.3 Función principal (main)

En esta parte se hace la declaración de la función principal del script, se define una función principal que será la primera función ejecutada (Tutotrial NS3). Dentro de esta función se define la variable *ecmpMode*, que puede tomar los valores enteros de 0, 1, 2 según el valor fijado hará que se ejecute una de las sentencias que corresponden a los modos de ECMP random (1), por flujo (2) y sin ECMP (0). Mediante el *CommandLine* se puede hacer que uno de los atributos, según el valor fijado, del Ipv4 global routing se cambie de false a *true* directamente desde este código sin necesidad de ir al código origen (ipv4-global-routing) para realizar este cambio.

```
int main (int argc, char *argv[])
{
uint32_t ecmpMode = 1;
CommandLine cmd;
cmd.AddValue ("EcmpMode", "Ecmp Mode: (0 none, 1 random, 2 flow)", ecmpMode);
cmd.Parse (argc, argv);
switch (ecmpMode)
```

```
{
case 0:
break;
case 1:
Config::SetDefault ("ns3::Ipv4GlobalRouting::RandomEcmpRouting", BooleanValue (true));
break;
case 2:
Config::SetDefault ("ns3::Ipv4GlobalRouting::FlowEcmpRouting", BooleanValue (true));
break;
default:
NS_FATAL_ERROR ("Illegal command value for EcmpMode: " << ecmpMode);
break;
}
}
```

1.4 Ayudantes de topología

Estos ayudan a simplificar la configuración de ciertos atributos hacen uso de *Node Container* y *NetDevice Container*.

Contenedor de nodo

Permite crear un conjunto de nodos de igual tipo y de las mismas características. Se usa especialmente cuando se requiere un gran número de componentes en una red. En el código esta defino por:

```
NodeContainer c;
c.Create (11);
```

En donde el valor de 11 es el número de nodos a crear.

Internet stack helper

Agrega funcionalidades de IP/TCP/UDP a los nodos que conforman la red, es decir instala una pila de protocolos de Internet, además proporciona por defecto enrutamiento tanto estático como global (dinámico) (Tutorial NS3). En el código de ECMP esta definido por:

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

En donde *c* corresponde al número de nodos creados anteriormente.

Point to point helper

Ayuda en la creación de redes punto a punto, se puede definir algunas funciones como:

- *Set device Attribute,* permite configurar ciertos valores como el “Data Rate” del dispositivo o interfaz del nodo.

- *Set channel Attribute,* permite configurar valores del canal de comunicación, en nuestro caso fijamos el valor del "Delay" del canal.

En el código tenemos lo siguiente:

```
PointToPointHelper p2p;
p2p.SetDeviceAttribute ("DataRate", StringValue ("2Mbps"));
p2p.SetChannelAttribute ("Delay", StringValue ("1ms"));
```

En donde p2p es la variable definida para el canal.

Contenedor de dispositivo de red

Permite crear un grupo de varios dispositivos de red o interfaces de igual tipo y de las mismas características. Se puede decir que en esta parte del código se definen los enlaces o conexiones que tiene cada nodo con los demás. Así, en el código tenemos:

```
NetDeviceContainer d0d1 = p2p.Install (n0n1);
NetDeviceContainer d0d2 = p2p.Install (n0n2);
NetDeviceContainer d0d3 = p2p.Install (n0n3);
NetDeviceContainer d0d4 = p2p.Install (n0n4);
NetDeviceContainer d1d5 = p2p.Install (n1n5);
NetDeviceContainer d2d6 = p2p.Install (n2n6);
NetDeviceContainer d3d7 = p2p.Install (n3n7);
NetDeviceContainer d4d8 = p2p.Install (n4n8);
NetDeviceContainer d5d9 = p2p.Install (n5n9);
NetDeviceContainer d6d9 = p2p.Install (n6n9);
NetDeviceContainer d7d9 = p2p.Install (n7n9);
NetDeviceContainer d8d9 = p2p.Install (n8n9);
```

Ipv4 address helper

Es un generador de direcciones IPv4, facilita la asignación de direcciones para los dispositivos o interfaces de cada nodo que conforman la red. En el código usado está definido por:

```
Ipv4AddressHelper ipv4;
ipv4.SetBase ("10.0.1.0", "255.255.255.0");
ipv4.Assign (d0d1);
ipv4.SetBase ("10.1.2.0", "255.255.255.0");
ipv4.Assign (d0d2);
ipv4.SetBase ("10.1.3.0", "255.255.255.0");
ipv4.Assign (d0d3);
ipv4.SetBase ("10.1.4.0", "255.255.255.0");
ipv4.Assign (d0d4);
ipv4.SetBase ("10.1.5.0", "255.255.255.0");
ipv4.Assign (d1d5);
ipv4.SetBase ("10.2.6.0", "255.255.255.0");
ipv4.Assign (d2d6);
```

ipv4.SetBase ("10.3.7.0", "255.255.255.0");
ipv4.Assign (d3d7);
ipv4.SetBase ("10.4.8.0", "255.255.255.0");
ipv4.Assign (d4d8);
ipv4.SetBase ("10.5.9.0", "255.255.255.0");
ipv4.Assign (d5d9);
ipv4.SetBase ("10.6.9.0", "255.255.255.0");
ipv4.Assign (d6d9);
ipv4.SetBase ("10.7.9.0", "255.255.255.0");
ipv4.Assign (d7d9);
ipv4.SetBase ("10.8.9.0", "255.255.255.0");
ipv4.Assign (d8d9);
ipv4.SetBase ("10.9.10.0", "255.255.255.0");
ipv4.Assign (d9d10);

Ipv4 global rounting helper

Ayuda con la activación y administración de enrutamiento global, permite llenar las tablas de enrutamiento de los nodos con las rutas mas cortas.

Ipv4GlobalRoutingHelper::PopulateRoutingTables ();

Aplicaciones onoff

Esta clase se usa para definir aplicaciones como generador de tráfico, receptor de tráfico, inicio y finalización de la aplicación. Se tiene algunas funciones como:

- *OnOffHelper,* es un atributo en el que se define el tipo de tráfico, la IP del nodo destino a donde se envía el tráfico y el puerto destino.
- *Onoff.SetConstantRate,* fija el valor del tráfico que se genera en el nodo origen.
- *Onoff.SetAttribute,* aquí se configure el tamaño del paquete.

En el código que se usa las funciones anteriores están definidas por:

OnOffHelper onoff ("ns3::TcpSocketFactory", InetSocketAddress (Ipv4Address ("10.9.10.2"), port));
onoff.SetConstantRate (DataRate ("400kbps"));
onoff.SetAttribute ("PacketSize", UintegerValue (500));

PacketSinkHelper, se define un nodo sink (sumidero) que recepta y consume el tráfico generado a la IP y puerto destino.

Sink Install, se configure el nodo que actuará como sink.

Lo descrito anteriormente se muestra en el código:

PacketSinkHelper sink ("ns3::TcpSocketFactory",

Address (InetSocketAddress (Ipv4Address::GetAny (), port)));

sink.Install (c.Get (10));

1.5 Simulator

Esta clase pública permite el acceso a los eventos programados en el código. Se encarga de ejecutar o iniciar la simulación (Simulator::Run) de acuerdo a los eventos programados en orden desde el más antiguo hasta el más reciente hasta que no haya más eventos en cola o se detenga la simulación (Simulator::Stop)

ANEXO 2. ECMP patch

```
# HG changeset patch
# User Tom Henderson <tomh@tomh.org>
# Date 1356904421 28800
# Node ID c20b5b6de0a1fceb40f7b57ed9a305cf0603da06
# Parent  043544eef3ed4b73924e3f61025b8849fd9b750d
New Ipv4GlobalRouting option for ECMP to split load per-path

diff -r 043544eef3ed -r c20b5b6de0a1 CHANGES.html
--- a/CHANGES.html	Fri Dec 28 04:22:19 2012 +0100
+++ b/CHANGES.html	Sun Dec 30 13:53:41 2012 -0800
@@ -55,7 +55,12 @@

 <h2>New API:</h2>
 <ul>
-<li></li>
+<li>A new attribute Ipv4GlobalRouting "FlowEcmpRouting" has been added.
+Setting this attribute to true will result in equal-cost multipath load
+balancing across paths, with each flow resolving to the same path based
+on its five-tuple.  This mode is in contrast to the existing
+"RandomEcmpRouting" mode which will randomly balance packets across paths
+(irrespective of their flows).  By default, FlowEcmpRouting is false.</li>
 </ul>

 <h2>Changes to existing API:</h2>
diff -r 043544eef3ed -r c20b5b6de0a1 src/internet/examples/ecmp-global-routing.cc
--- /dev/null	Thu Jan 01 00:00:00 1970 +0000
+++ b/src/internet/examples/ecmp-global-routing.cc	Sun Dec 30 13:53:41 2012 -0800
@@ -0,0 +1,156 @@
+/* -*- Mode:C++; c-file-style:"gnu"; indent-tabs-mode:nil; -*- */
+/*
+ * This program is free software; you can redistribute it and/or modify
+ * it under the terms of the GNU General Public License version 2 as
+ * published by the Free Software Foundation;
+ *
+ * This program is distributed in the hope that it will be useful,
+ * but WITHOUT ANY WARRANTY; without even the implied warranty of
+ * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
+ * GNU General Public License for more details.
+ *
+ * You should have received a copy of the GNU General Public License
+ * along with this program; if not, write to the Free Software
+ * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
+ *
+ */
+
+// This script illustrates the behavior of equal-cost multipath routing
+// (ECMP) with Ipv4 global routing, across three equal-cost paths
```

```
+//
+// Network topology:
+//
+//           n2
+//         /  \ all links
+//        /    \  point-to-point
+//  n0---n1--n3--n5----n6
+//        \    /
+//         \  /
+//          n4
+//
+// - multiple UDP flows from n0 to n6
+// - Tracing of queues and packet receptions to file "ecmp-global-routing.tr"
+// - pcap traces on nodes n2, n3, and n4
+
+#include <iostream>
+#include <fstream>
+#include <string>
+#include <cassert>
+
+#include "ns3/core-module.h"
+#include "ns3/network-module.h"
+#include "ns3/applications-module.h"
+#include "ns3/point-to-point-module.h"
+#include "ns3/internet-module.h"
+
+using namespace ns3;
+
+NS_LOG_COMPONENT_DEFINE ("EcmpGlobalRoutingExample");
+
+int
+main (int argc, char *argv[])
+{
+  uint32_t ecmpMode = 1;
+
+  // Allow the user to override any of the defaults and the above
+  // Bind ()s at run-time, via command-line arguments
+  CommandLine cmd;
+  cmd.AddValue ("EcmpMode", "Ecmp Mode: (0 none, 1 random, 2 flow)", ecmpMode);
+  cmd.Parse (argc, argv);
+
+  switch (ecmpMode)
+    {
+      case 0:
+        break;  //no ECMP
+      case 1:
+                        Config::SetDefault    ("ns3::Ipv4GlobalRouting::RandomEcmpRouting",
BooleanValue (true));
+        break;
```

```
+      case 2:
+          Config::SetDefault ("ns3::Ipv4GlobalRouting::FlowEcmpRouting", BooleanValue
(true));
+        break;
+      default:
+        NS_FATAL_ERROR ("Illegal command value for EcmpMode: " << ecmpMode);
+        break;
+    }
+
+  // Allow the user to override any of the defaults and the above
+
+  NS_LOG_INFO ("Create nodes.");
+  NodeContainer c;
+  c.Create (7);
+  NodeContainer n0n1 = NodeContainer (c.Get (0), c.Get (1));
+  NodeContainer n1n2 = NodeContainer (c.Get (1), c.Get (2));
+  NodeContainer n1n3 = NodeContainer (c.Get (1), c.Get (3));
+  NodeContainer n1n4 = NodeContainer (c.Get (1), c.Get (4));
+  NodeContainer n2n5 = NodeContainer (c.Get (2), c.Get (5));
+  NodeContainer n3n5 = NodeContainer (c.Get (3), c.Get (5));
+  NodeContainer n4n5 = NodeContainer (c.Get (4), c.Get (5));
+  NodeContainer n5n6 = NodeContainer (c.Get (5), c.Get (6));
+
+  InternetStackHelper internet;
+  internet.Install (c);
+
+  NS_LOG_INFO ("Create channels.");
+  PointToPointHelper p2p;
+  p2p.SetDeviceAttribute ("DataRate", StringValue ("10Mbps"));
+  p2p.SetChannelAttribute ("Delay", StringValue ("1ms"));
+  NetDeviceContainer d0d1 = p2p.Install (n0n1);
+  NetDeviceContainer d1d2 = p2p.Install (n1n2);
+  NetDeviceContainer d1d3 = p2p.Install (n1n3);
+  NetDeviceContainer d1d4 = p2p.Install (n1n4);
+  NetDeviceContainer d2d5 = p2p.Install (n2n5);
+  NetDeviceContainer d3d5 = p2p.Install (n3n5);
+  NetDeviceContainer d4d5 = p2p.Install (n4n5);
+  NetDeviceContainer d5d6 = p2p.Install (n5n6);
+
+  NS_LOG_INFO ("Assign IP Addresses.");
+  Ipv4AddressHelper ipv4;
+  ipv4.SetBase ("10.0.1.0", "255.255.255.0");
+  ipv4.Assign (d0d1);
+  ipv4.SetBase ("10.1.2.0", "255.255.255.0");
+  ipv4.Assign (d1d2);
+  ipv4.SetBase ("10.1.3.0", "255.255.255.0");
+  ipv4.Assign (d1d3);
+  ipv4.SetBase ("10.1.4.0", "255.255.255.0");
+  ipv4.Assign (d1d4);
```

```
+  ipv4.SetBase ("10.2.5.0", "255.255.255.0");
+  ipv4.Assign (d2d5);
+  ipv4.SetBase ("10.3.5.0", "255.255.255.0");
+  ipv4.Assign (d3d5);
+  ipv4.SetBase ("10.4.5.0", "255.255.255.0");
+  ipv4.Assign (d4d5);
+  ipv4.SetBase ("10.5.6.0", "255.255.255.0");
+  ipv4.Assign (d5d6);
+
+  NS_LOG_INFO ("Populate routing tables.");
+  Ipv4GlobalRoutingHelper::PopulateRoutingTables ();
+
+  NS_LOG_INFO ("Create Applications.");
+  uint16_t port = 9;   // Discard port (RFC 863)
+  OnOffHelper onoff ("ns3::UdpSocketFactory",
+                     InetSocketAddress (Ipv4Address ("10.5.6.2"), port));
+  onoff.SetConstantRate (DataRate ("100kbps"));
+  onoff.SetAttribute ("PacketSize", UintegerValue (500));
+
+  ApplicationContainer apps;
+  for (uint32_t i = 0; i < 10; i++)
+    {
+      apps.Add (onoff.Install (c.Get (0)));
+    }
+  apps.Add (onoff.Install (c.Get (1)));
+  apps.Start (Seconds (1.0));
+  apps.Stop (Seconds (5.0));
+
+  PacketSinkHelper sink ("ns3::UdpSocketFactory",
+                         Address (InetSocketAddress (Ipv4Address::GetAny (), port)));
+  sink.Install (c.Get (6));
+
+  // Trace the right-most (second) interface on nodes 2, 3, and 4
+  p2p.EnablePcap ("ecmp-global-routing", 2, 2);
+  p2p.EnablePcap ("ecmp-global-routing", 3, 2);
+  p2p.EnablePcap ("ecmp-global-routing", 4, 2);
+
+  NS_LOG_INFO ("Run Simulation.");
+  Simulator::Run ();
+  Simulator::Destroy ();
+}
diff -r 043544eef3ed -r c20b5b6de0a1 src/internet/examples/wscript
--- a/src/internet/examples/wscript          Fri Dec 28 04:22:19 2012 +0100
+++ b/src/internet/examples/wscript       Sun Dec 30 13:53:41 2012 -0800
@@ -7,3 +7,9 @@
     obj = bld.create_ns3_program('main-simple',
                                  ['network', 'internet', 'applications'])
     obj.source = 'main-simple.cc'
+
```

```
+    obj = bld.create_ns3_program( 'ecmp-global-routing',
+                                 ['applications', 'point-to-point', 'internet', 'network', 'core'])
+    obj.source = 'ecmp-global-routing.cc'
+
+
diff -r 043544eef3ed -r c20b5b6de0a1 src/internet/model/ipv4-global-routing.cc
--- a/src/internet/model/ipv4-global-routing.cc     Fri Dec 28 04:22:19 2012 +0100
+++ b/src/internet/model/ipv4-global-routing.cc  Sun Dec 30 13:53:41 2012 -0800
@@ -20,6 +20,7 @@
 #include <iomanip>
 #include "ns3/names.h"
 #include "ns3/log.h"
+#include "ns3/abort.h"
 #include "ns3/simulator.h"
 #include "ns3/object.h"
 #include "ns3/packet.h"
@@ -27,6 +28,8 @@
 #include "ns3/ipv4-route.h"
 #include "ns3/ipv4-routing-table-entry.h"
 #include "ns3/boolean.h"
+#include "udp-header.h"
+#include "tcp-header.h"
 #include "ipv4-global-routing.h"
 #include "global-route-manager.h"

@@ -36,6 +39,10 @@

 NS_OBJECT_ENSURE_REGISTERED (Ipv4GlobalRouting);

+/* see http://www.iana.org/assignments/protocol-numbers */
+const uint8_t TCP_PROT_NUMBER = 6;
+const uint8_t UDP_PROT_NUMBER = 17;
+
 TypeId
 Ipv4GlobalRouting::GetTypeId (void)
 {
@@ -46,6 +53,11 @@
                   BooleanValue (false),
                   MakeBooleanAccessor (&Ipv4GlobalRouting::m_randomEcmpRouting),
                   MakeBooleanChecker ())
+    .AddAttribute ("FlowEcmpRouting",
+                   "Set to true if flows are randomly routed among ECMP; set to false for using only one route consistently",
+                   BooleanValue (false),
+                   MakeBooleanAccessor (&Ipv4GlobalRouting::m_flowEcmpRouting),
+                   MakeBooleanChecker ())
     .AddAttribute ("RespondToInterfaceEvents",
                   "Set to true if you want to dynamically recompute the global routes upon Interface notification events (up/down, or add/remove address)",
```

```
                    BooleanValue (false),
@@ -57,6 +69,7 @@

 Ipv4GlobalRouting::Ipv4GlobalRouting ()
   : m_randomEcmpRouting (false),
+    m_flowEcmpRouting (false),
     m_respondToInterfaceEvents (false)
 {
   NS_LOG_FUNCTION_NOARGS ();
@@ -133,12 +146,58 @@
   m_ASexternalRoutes.push_back (route);
 }

+// This function is used to spread the routing of flows across equal
+// cost paths, by calculating an integer based on the five-tuple in the headers
+//
+// It assumes that if a transport protocol value is specified in the header,
+// that a transport header with port numbers is prepended to the ipPayload
+//
+uint32_t
+Ipv4GlobalRouting::GetTupleValue (const Ipv4Header &header, Ptr<const Packet>
ipPayload)
+{
+  // We do not care if this value rolls over
+  uint32_t tupleValue = header.GetSource ().Get () +
+                        header.GetDestination ().Get () +
+                        header.GetProtocol ();
+  switch (header.GetProtocol ())
+    {
+    case UDP_PROT_NUMBER:
+      {
+        UdpHeader udpHeader;
+        ipPayload->PeekHeader (udpHeader);
+        NS_LOG_DEBUG ("Found UDP proto and header: " <<
+                      udpHeader.GetSourcePort () << ":" <<
+                      udpHeader.GetDestinationPort ());
+        tupleValue += udpHeader.GetSourcePort ();
+        tupleValue += udpHeader.GetDestinationPort ();
+        break;
+      }
+    case TCP_PROT_NUMBER:
+      {
+        TcpHeader tcpHeader;
+        ipPayload->PeekHeader (tcpHeader);
+        NS_LOG_DEBUG ("Found TCP proto and header: " <<
+                      tcpHeader.GetSourcePort () << ":" <<
+                      tcpHeader.GetDestinationPort ());
+        tupleValue += tcpHeader.GetSourcePort ();
+        tupleValue += tcpHeader.GetDestinationPort ();
```

```
+        break;
+      }
+    default:
+      {
+        NS_LOG_DEBUG ("Udp or Tcp header not found");
+        break;
+      }
+    }
+  return tupleValue;
+}

 Ptr<Ipv4Route>
-Ipv4GlobalRouting::LookupGlobal (Ipv4Address dest, Ptr<NetDevice> oif)
+Ipv4GlobalRouting::LookupGlobal (const Ipv4Header &header, Ptr<const Packet> ipPayload, Ptr<NetDevice> oif)
 {
   NS_LOG_FUNCTION_NOARGS ();
-  NS_LOG_LOGIC ("Looking for route for destination " << dest);
+  NS_ABORT_MSG_IF (m_randomEcmpRouting && m_flowEcmpRouting, "Ecmp mode selection");
+  NS_LOG_LOGIC ("Looking for route for destination " << header.GetDestination());
   Ptr<Ipv4Route> rtentry = 0;
   // store all available routes that bring packets to their destination
   typedef std::vector<Ipv4RoutingTableEntry*> RouteVec_t;
@@ -150,7 +209,7 @@
        i++)
     {
       NS_ASSERT ((*i)->IsHost ());
-      if ((*i)->GetDest ().IsEqual (dest))
+      if ((*i)->GetDest ().IsEqual (header.GetDestination ()))
         {
           if (oif != 0)
             {
@@ -173,7 +232,7 @@
       {
         Ipv4Mask mask = (*j)->GetDestNetworkMask ();
         Ipv4Address entry = (*j)->GetDestNetwork ();
-        if (mask.IsMatch (dest, entry))
+        if (mask.IsMatch (header.GetDestination (), entry))
           {
             if (oif != 0)
               {
@@ -196,7 +255,7 @@
       {
         Ipv4Mask mask = (*k)->GetDestNetworkMask ();
         Ipv4Address entry = (*k)->GetDestNetwork ();
-        if (mask.IsMatch (dest, entry))
+        if (mask.IsMatch (header.GetDestination (), entry))
           {
```

```
            NS_LOG_LOGIC ("Found external route" << *k);
            if (oif != 0)
@@ -214,15 +273,20 @@
     }
   if (allRoutes.size () > 0 ) // if route(s) is found
     {
-      // pick up one of the routes uniformly at random if random
-      // ECMP routing is enabled, or always select the first route
-      // consistently if random ECMP routing is disabled
+      // select one of the routes uniformly at random if random
+      // ECMP routing is enabled, or map a flow consistently to a route
+      // if flow ECMP routing is enabled, or otherwise always select the
+      // first route
       uint32_t selectIndex;
       if (m_randomEcmpRouting)
         {
           selectIndex = m_rand->GetInteger (0, allRoutes.size ()-1);
         }
-      else
+      else  if (m_flowEcmpRouting && (allRoutes.size () > 1))
+        {
+          selectIndex = GetTupleValue (header, ipPayload) % allRoutes.size ();
+        }
+      else
         {
           selectIndex = 0;
         }
@@ -458,7 +522,7 @@
  // See if this is a unicast packet we have a route for.
  //
   NS_LOG_LOGIC ("Unicast destination- looking up");
-  Ptr<Ipv4Route> rtentry = LookupGlobal (header.GetDestination (), oif);
+  Ptr<Ipv4Route> rtentry = LookupGlobal (header, p, oif);
   if (rtentry)
     {
       sockerr = Socket::ERROR_NOTERROR;
@@ -536,7 +600,7 @@
     }
   // Next, try to find a route
   NS_LOG_LOGIC ("Unicast destination- looking up global route");
-  Ptr<Ipv4Route> rtentry = LookupGlobal (header.GetDestination ());
+  Ptr<Ipv4Route> rtentry = LookupGlobal (header, p);
   if (rtentry != 0)
     {
       NS_LOG_LOGIC ("Found unicast destination- calling unicast callback");
diff -r 043544eef3ed -r c20b5b6de0a1 src/internet/model/ipv4-global-routing.h
--- a/src/internet/model/ipv4-global-routing.h       Fri Dec 28 04:22:19 2012 +0100
+++ b/src/internet/model/ipv4-global-routing.h    Sun Dec 30 13:53:41 2012 -0800
@@ -227,6 +227,8 @@
```

```
 private:
   /// Set to true if packets are randomly routed among ECMP; set to false for using only one route consistently
   bool m_randomEcmpRoting;
+  /// Set to true if flows are randomly routed among ECMP; set to false for using only one route consistently
+  bool m_flowEcmpRouting;
   /// Set to true if this interface should respond to interface events by globallly recomputing routes
   bool m_respondToInterfaceEvents;
   /// A uniform random number generator for randomly routing packets among ECMP
@@ -242,7 +244,9 @@
   typedef std::list<Ipv4RoutingTableEntry *>::const_iterator ASExternalRoutesCI;
   typedef std::list<Ipv4RoutingTableEntry *>::iterator ASExternalRoutesI;

-  Ptr<Ipv4Route> LookupGlobal (Ipv4Address dest, Ptr<NetDevice> oif = 0);
+  uint32_t GetTupleValue (const Ipv4Header &header, Ptr<const Packet> ipPayload);
+
+  Ptr<Ipv4Route> LookupGlobal (const Ipv4Header &header, Ptr<const Packet> ipPayload, Ptr<NetDevice> oif = 0);

   HostRoutes m_hostRoutes;
   NetworkRoutes m_networkRoutes;
```