MODELO DEL PROBLEMA DE LA UBICACIÓN DEL CONTROLADOR EN REDES SDN CON CONSIDERACIONES DE COSTO, LATENCIA Y CONFIABILIDAD.

(Trabajo de Grado)

YULIANA ANDREA AMAYA VELASQUEZ

Director:

Ing. Juliana Arévalo, MSc

UNIVERSIDAD SANTO TOMÁS
FACULTAD DE INGENIERÍA DE TELECOMUNICACIONES
MAESTRÍA EN TELECOMUNICACIONES Y REGULACIÓN TIC
BOGOTÁ, 2022

Dedicado principalmente a mi madre Margarita Velásquez quien día a día se sigue esforzando por el bienestar de sus hijas y brinda lo mejor para ello, porque a pesar de las adversidades es un claro ejemplo de fortaleza. A mi padre Helio Amaya por el apoyo y a mis hermanas, como ejemplo de que todo aquello que requiere un gran esfuerzo genera gratificantes cosechas.

Dedicado especialmente a mí, porque muchas veces pensé en desfallecer o en rendirme, pero gracias a las personas correctas logré encontrar las herramientas necesarias para culminar, entre ellas Javier Sánchez.

.

AGRADECIMIENTOS

Agradezco a Dios y al universo por cada obra en mi vida, por cada lección de vida que impacta positivamente mi vida.

Agradezco a mi directora Juliana Arevalo por la paciencia al realizar este proyecto conmigo, por transmitir sus conocimientos en la realización y finalización de este trabajo.

TABLA CONTENIDO

ACRÓNIMOS

SDN Del inglés, Software Defined Networks. Redes definidas por software

CPP Del inglés, Controller Placement Problem. Problema de la ubicación del controlador

CNPA: Algoritmo de partición de Red Basado en Clústeres. Del inglés, Clustering-Based Network Partition Algorithm.

NFV Del inglés, Virtualization Network Function

Capex Del inglés, Capital Expenditure.

Opex Del inglés, Operative expenses.

LAN: Red de área local, Del inglés, Local Area Network.

POCO: Óptima ubicación del controlador basada en Pareto. Del inglés, Pareto-based Optimal Controller-placement.

RAM: Memoria de Acceso Aleatorio. Del inglés, Random Access Memory

WAN: Red de área amplia. Del inglés, Wide Area Network.

RESUMEN

SND dentro de su paradigma al separar el plano de datos del plano de control en una red, busca que un usuario pueda administrar, programar y controlar los recursos de la red para así facilitar y simplificar la gestión de los servicios, con el fin de realizar aprovechamiento de los recursos permitiendo su innovación y evolución en comparación con las redes tradicionales.

De acuerdo con lo anterior en los últimos años se han realizado investigaciones sobre el Problema de la Ubicación del Controlador (CPP), aunque no existe una fórmula única, para cada caso, se debe responder si es necesario uno o varios controladores en la red. Este trabajo tiene como objetivo realizar las pruebas a los estudios sobre la solución del CPP en el proyecto SDNCon de la línea de investigación interconexión y convergencia del grupo INVTEL de la facultad de ingeniería de la Universidad Santo Tomás, teniendo en cuenta los escenarios ya planteados en aquellos estudios, adicionalmente realizar una comparación de los resultados con el fin de proponer un nuevo modelo matemático que incluya condiciones de confiabilidad dentro de la red, teniendo en cuenta parámetros como costo y latencia y así mismo realizar su respectiva evaluación con las herramientas utilizadas en el proyecto para si solventar el problema de la ubicación del controlador.

Palabras clave: SDN, CPP.

ABSTRACT

SND, within its paradigm by separating the data plane from the control plane in a network, seeks that a user can manage, schedule and control network resources in order to facilitate and simplify the management of services, in order to perform taking advantage of the resources that allow its innovation and evolution in comparison with traditional networks.

In accordance with the above, in recent years, investigations have been carried out on the Controller Location Problem (CPP), although there is no unique formula, for each case, it must be answered if one or more controllers is necessary in the network. The objective of this work is to carry out the tests to the studies on the CPP solution in the SDNCon project already proposed in those studies, additionally it made a comparison of the results in order to propose a new mathematical model that would include reliability conditions within the network, taking into account parameters such as cost and latency and also carried out their respective evaluation with the tools used in the project to solve the problem of the location of the controller.

Keywords: SDN, CPP.

INTRODUCCIÓN

En algunos entornos empresariales, se tienen varios dispositivos conectados a la red y día a día se siguen sumando más para convertirse en un medio de transmisión y almacenamiento de datos masivo, buscando siempre una alta calidad del servicio para así permitir el acceso a Internet a millones de usuarios, sin embargo, este proceso va más allá de conectar un ordenador con un modem, se necesita mucho más como routers, switches, firewalls, servidores, entre otros, para poder tener acceso al servicio [1].

De acuerdo con lo anterior las empresas de telecomunicaciones viven en constante reto de mantener a los usuarios conectados brindando sobre todo seguridad y confiabilidad en los sistemas y este concepto se ha reforzado con la crisis del Covid-19 en donde algunos países tuvieron una mayor demanda en el uso de herramientas, principalmente para el teletrabajo, estudios virtuales y también mayor demanda de entretenimiento como streaming de video y juegos en línea, lo anterior genera saturación de las redes y degrado en la calidad de las conexiones [1], [2].

Continuando la idea anterior, las redes definidas por software son un nuevo paradigma en las redes que busca reducir costos y facilitar la implementación e implantación de los servicios de la red, puesto que es una red con equipos menos costosos y listos para ser controlados por software. Estás redes se caracterizan por la separación entre el plano de control y el plano de datos, permitiendo que un usuario pueda programar y controlar los recursos de la red, con el fin de facilitar el funcionamiento de los servicios de red [3],[4].

El principal objetivo de las redes basadas en SDN son los controladores, dado que por su diseño tiene el control absoluto de una red, y la comunicación se realiza mediante un canal de comando, además le permiten tener una vista global de la red [5]. Si el controlador es centralizado permite desarrollar aplicaciones de redes complejas sin embargo tiene dos limitaciones: La primera es la confiabilidad, esta es limitada debido al único punto de falla y la segunda es el tráfico entre el controlador y los switches dado que se concentra en un solo servidor, esto hace que se genere problemas de escalabilidad [6]. Adicionalmente se

ha percibido que la sociedad se ha vuelto dependiente de la fiabilidad de las redes de telecomunicaciones [2].

De acuerdo con el nuevo paradigma SDN, la universidad Santo Tomás está contribuyendo por medio del grupo INVTEL con la línea de investigación interconexión y convergencia con el proyecto Optimización de la ubicación de controladores en redes SDN abreviado SDNCon, dentro del marco de convocatorias FODEIN. Dentro del proyecto se realizó un estado del arte que permitió elaborar el modelo de optimización.

Teniendo en cuenta lo anterior en este trabajo se realizó la réplica de las pruebas a los estudios pertenecientes al proyecto SDNCon para así realizar una comparación de los resultados obtenidos con el fin de proponer un nuevo modelo matemático que incluya condiciones de confiabilidad dentro de la red, teniendo en cuenta parámetros como costo y latencia y así mismo realizar su respectiva evaluación con las herramientas utilizadas en el proyecto para si solventar el problema de la ubicación del controlador. Entonces la pregunta es: ¿Qué modelo del CPP permite incluir los parámetros de confiabilidad, además de costo y latencia, considerando los algoritmos ya desarrollados por el proyecto SDNCon de la línea de investigación interconexión y convergencia del grupo INVTEL de la Facultad de ingeniería de la Universidad Santo Tomás?

1 MARCO GENERAL DEL PROYECTO

1.1 OBJETIVOS

Proponer un modelo del problema de la ubicación del controlador que considere la confiabilidad de la red, a partir de la verificación y comparación del desempeño y costo computacional de los métodos planteados por el proyecto SDNCon de la línea de investigación interconexión y convergencia del grupo INVTEL (siglas) de la facultad de ingeniería de la Universidad Santo Tomás.

- Verificar los resultados obtenidos por los estudios anteriores del proyecto SDNCon, por medio de la replicación de las pruebas realizadas.

- Comparar los resultados obtenidos con el fin de realizar un estudio de los diferentes métodos para así solucionar el problema de la ubicación del controlador (CPP).

- Proponer un nuevo modelo matemático a partir de las comparaciones realizadas, que incluya condiciones de confiabilidad dentro de la red, teniendo en cuenta parámetros como costo y latencia.

- Evaluar el modelo matemático propuesto por medio de herramientas de simulación utilizadas en el proyecto SDNCon con el fin de solventar el problema de la ubicación del controlador.

1.2 ALCANCE

El alcance de este proyecto incluye verificar los resultados obtenidos por los mecanismos implementados en el proyecto SDNCon, repitiendo las pruebas realizadas. Se construirá un estudio comparativo con los resultados. Se planteará un nuevo modelo matemático en el que se considere la confiabilidad como un factor que cuantifique la capacidad de la red para

mantenerse operativa frente a un fallo en el enlace y se realizará la evaluación del modelo planteado sobre la herramienta de simulación utilizada.

1.3 METODOLOGÍA

El desarrollo de este proyecto tiene dos enfoques:

1. **Experimental:** Dado que está basada en la metodología científica y tiene como fin recopilar datos, como ya tenemos generados los algoritmos, lo que se hizo es realizar las pruebas en ambientes totalmente controlados con herramientas de simulación utilizadas en el proyecto SDNCon.

2. **Cuantitativa:** Porque la evaluación de cada algoritmo se hizo bajo unas mediciones en el entorno de pruebas con el fin de realizar la comparación de los distintos algoritmos y así establecer la diferencia entre cada uno a nivel de desempeño computacional y costos.

Para cumplir con el proyecto el equipo tendrá reuniones semanales para determinar los avances al respecto.

Las fases de la investigación se enumeran a continuación:

I. **Revisión del proyecto.** En esta fase se revisará el estado del arte del proyecto SDNCon.
II. **Verificación de resultados.** En esta fase se llevará a cabo la implementación de los algoritmos ya existentes. Posterior a eso se realizará pruebas y de requerir ajustes sobre las interfaces o conjuntos de datos, y los códigos se realizarán, nuevamente se realizarán pruebas para así obtener los datos de comparación.
III. **Estudio comparativo.** Luego de tener los datos se realizará un estudio comparativo de cada uno de los métodos.
IV. **Planteamiento y evaluación del nuevo modelo.** Luego de tener el estudio comparativo se realizará una propuesta de modelo que considere la confiabilidad para así determinar la ubicación del controlador (CPP).

2 PROBLEMA DE LA UBICACIÓN DEL CONTROLADOR EN REDES SDN CON CONSIDERACIONES DE COSTO, LATENCIA Y CONFIABILIDAD.

2.1 DEFINICIÓN GENERAL DE REDES SDN.

Software Defined Network (SDN), es un paradigma relacionado con redes, por la separación del plano de datos del plano de control en una red de computadores [3]. De acuerdo con lo anterior la Unión Internacional de Telecomunicaciones (UIT) en la recomendación *Y.3300* define SDN como un conjunto de procedimientos para que un usuario pueda programar y controlar los recursos de la red, con el fin de facilitar el funcionamiento de los servicios de red [4].

SDN en su arquitectura promete simplificar la gestión de la red, aprovechar los recursos, permite innovar y evolucionar las redes tradicionales. La arquitectura de SDN tiene cinco partes principales: Plano de aplicación (AP), plano de control (CP), plano de datos (DP), interfaz en dirección norte (NBI) e interfaz en dirección sur (SBI) como se muestra en la Figura 1 [7].

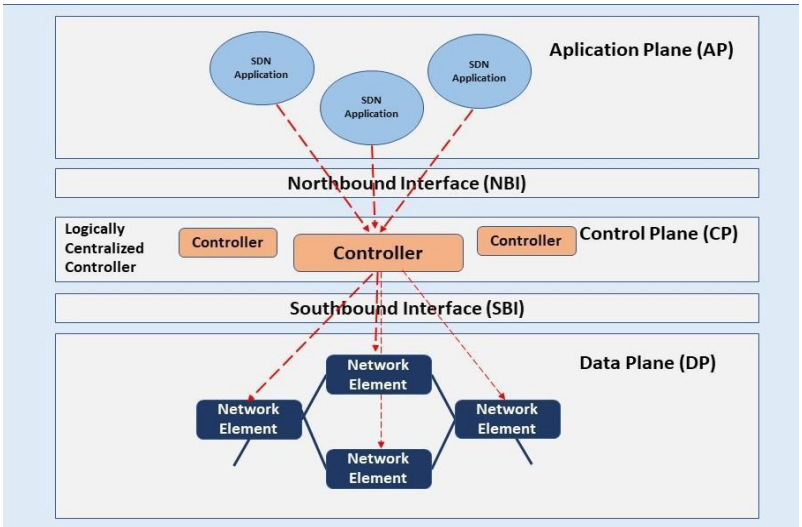


Figura 1. Arquitectura de SDN.

Fuente: Adaptada de la Unión Internacional de Telecomunicaciones (UIT).

Continuando la idea anterior la arquitectura de la red SDN es directamente programable y cada plano tiene las siguientes funcionalidades: En el plano de datos se realiza el transporte y procesamiento de paquetes de datos teniendo en cuenta las decisiones tomadas por el plano de control, como son reenvío o funcionalidad de enrutamiento de datos o incluso ambos. En el plano de control se determina el comportamiento de los recursos, como el transporte y procesamiento de datos según las instrucciones del plano de aplicación, en el cual adicionalmente se especifican los servicios de red o aplicaciones, estas aplicaciones interactúan con el plano de control [4],[8].

SDN ofrece una arquitectura centralizada capaz de realizar una gestión flexible y aplicaciones de Calidad de Servicio (QoS, por sus siglas en inglés), en los últimos años se han realizado investigaciones sobre la ubicación del controlador, aunque no existe una fórmula única, para cada caso, se debe responder si es necesario uno o varios controladores en la red [9], [8],[10], [11], [12], [13].

Las redes SDN permiten al administrador de la red controlar, cambiar, iniciar y gobernar el comportamiento de la red, con el fin de que la red sea más simple, programable, personalizable y manejable [14].

2.2 PROBLEMA DE LA UBICACIÓN DEL CONTROLADOR EN REDES SDN.

Aproximadamente hace una década SDN partió de la dificultad de probar nuevos protocolos utilizando switches de hardware patentados en redes de Campus, adicionalmente se remonta a una colaboración de investigación entre la Universidad de Stanford y la Universidad de California en Berkeley [15]. Con el tiempo los experimentos en el mundo académico pasaron a ser adoptados en redes de área amplia (WAN) y en redes de centros de datos (DC), con lo anterior algunas empresas como Google Inc., implemento SDN para interconectar sus centros de datos en todo el mundo [10].

SDN en su arquitectura se implementa con uno o varios controladores, pero en redes a gran escala un solo controlador no es adecuado y varios controladores se enfrentan al problema de la ubicación del controlador (CPP). En este sentido los ingenieros de redes se enfrentan a este dilema, lo que incita a buscar respuestas sobre preguntas cómo identificar un número

mínimo de controladores, cómo dividir la red, dónde ubicar cada controlador, cómo asignarlos y costos probables [11].

El paradigma moderno de SDN ha inspirado a los investigadores a resolver muchos de los desafíos de red, principalmente el problema de la ubicación del controlador (CPP) dado que esta es una consideración crítica porque afecta directamente la latencia, las políticas de enrutamiento, escalabilidad, equilibrio de carga, eficiencia energética, resiliencia, gestión de fallas, Calidad de Servicio (QoS), seguridad y sobrecarga del controlador [16] [10].

Para resolver lo mencionado anteriormente se han realizado varios estudios con distintos enfoques como son: Minimizar la latencia entre controladores y switches, mejorar la capacidad de recuperación de la red, la confiabilidad, reducción de costos de implementación y consumo de energía. En la figura dos se muestra cómo se han clasificado las distintas investigaciones para encontrar la solución de la ubicación del controlador (CPP) [10], [11], [12], [13].

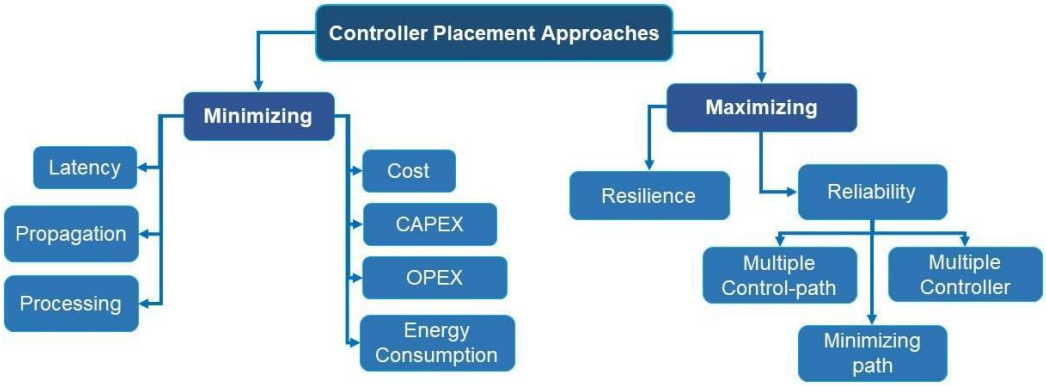


Figura 2. Clasificación investigaciones sobre CPP.

Dentro de este orden de ideas en las redes SND el problema de cuantos y donde colocar los controladores se ha estudiado extensamente a lo anterior se le conoce como *problema de ubicación del controlador* y se ha investigado principalmente los retrasos entre los switches y controladores [17].

El problema de la ubicación del controlador afecta casi todos los aspectos, desde la distribución hasta la tolerancia a fallos y el rendimiento de la red, para las redes de área amplia (WAN) es importante disminuir la latencia de propagación de los paquetes entre los

controladores y el switch, para eso se ha propuesto un algoritmo de Partición de Red Basado en Agrupación de Clústeres (CNPA) con esto una red se divide en subredes y se colocan varios controladores con el fin de acortar la latencia de propagación entre switches y controladores [10].

Una de las características más importantes y el principal objetivo de las redes basadas en SDN son los controladores, dado que por diseño tienen el control absoluto en una red y la comunicación entre los controladores se realiza mediante un canal de comando que puede estar encriptado, adicionalmente a ello se implementan acciones de red como conmutación y enrutamiento de paquetes, esto permite una vista global de la red incluyendo reglas de flujo y estadísticas dentro de los dispositivos de red a las aplicaciones SDN [5].

Teniendo en cuenta lo anterior, el aprovisionamiento de controladores genera un impacto significativo en el rendimiento de la red en un área geográfica extensa, por ende, es necesario utilizar varios controladores en vez de solo uno [18].

Un aspecto importante es proteger a los controladores SDN de aplicaciones maliciosas, para esto se han manifestado varios mecanismos, uno es el sandboxing, esto permite restringir el acceso a funciones críticas del controlador como la manipulación de las estructuras de datos internos, la comprobación de eventos de red ayuda a proteger la topología de la red de ataques [5].

La implementación de un proxy mediante el Protocolo de Control de Transmisión TCP dentro de los dispositivos de red permite reducir los ataques de denegación de servicio que se dirigen desde el plano de datos al controlador SDN, algunos consejos de seguridad son utilizar sistemas operativos modernos con el fin de evitar la instalación de software malicioso[5].

De acuerdo con lo anterior una red con un controlador centralizado permite el desarrollo de aplicaciones de red complejas, pero tiene dos limitaciones: La primera es la confiabilidad, esta es limitada debido al único punto de falla y la segunda es el tráfico entre el controlador y los switches dado que se concentra en un solo servidor, esto hace que se genere problemas de escalabilidad, dado que el procesamiento es limitado [6].Sin embargo, los

controladores SDN distribuidos fueron diseñados para abordar el problema mencionado anteriormente.

En la arquitectura distribuida los controladores son los responsables de interactuar con los switches, generando dos beneficios, primero la carga del procesamiento de cada controlador disminuye, esto ocurre porque el tráfico de control de los switches y controladores se distribuyen entre varios servidores esto, genera un efecto de equilibrio de carga y segundo, la implementación de mecanismos de resiliencia que mejora la confiabilidad de la red en caso de falla del controlador [6]. Debido a que cuando realizaron una medición real de las múltiples redes troncales observaron que aproximadamente el 70% de las fallas de la red son causada porque solo un enlace falla y la probabilidad de que varios enlaces fallen al mismo tiempo es posiblemente baja, por tanto, se presume que las fallas de la red son causadas principalmente por un solo enlace [19].

Complementando lo anterior los operadores también se enfrentan a grandes desafíos, uno son las condiciones de la red debido a que por lo general son muy cambiantes afectando el rendimiento en general. Estos cambios no solo aparecen de forma predecible y regular sino también en escalas de tiempo más largos y cortos, un ejemplo es cada hora. La adaptación es compleja debido a que toca asegurar el buen funcionamiento, para eso es importante calcular ubicaciones que tengan en cuenta la situación actual de la red [20].

Bajo estas condiciones se plantean diversos marcos de trabajo para la solución de CPP. Por ejemplo, **POCO** un marco basado en Pareto para la ubicación del controlador proporciona a los operadores ubicaciones con diferentes métricas de rendimiento, es decir en su configuración se realiza una evaluación de todas las posibles ubicaciones, permitiendo soluciones basadas en software o hardware para el plano de control permitiendo compensaciones entre aspectos como rendimiento, flexibilidad y costos [20].

Continuando la idea anterior sobre el estudio de los algoritmos exactos y heurísticos que se implementan y desarrollan en el marco POCO basado en Pareto para la ubicación del controlador, en el estudio muestra que los beneficios de incorporar enfoques heurísticos son analizar las instancias del problema son demasiado grandes para realizar la evaluación de manera exhaustiva y en entornos dinámicos presentan limitaciones de tiempo, por tanto los algoritmos heurísticos permiten tener un compromiso entre tiempo y precisión [20].

Por otro lado, tener un controlador dentro de la red tiene la ventaja de proporcionar una vista de toda la red, adicionalmente las decisiones de enrutamiento se toman en función de esa vista, pero tener un solo controlador tiene otras desventajas relacionadas con la escalabilidad y el rendimiento. La solución para el problema mencionado anteriormente es tener varios controladores [21].

Si SDN tiene uno o varios controladores es importante resaltar que tendrá un impacto de rendimiento y costo sobre la red eso se mostró en [21] entonces para tener un rendimiento óptimo y un costo mínimo se debe tener en cuenta las diferentes ubicaciones con el fin de mostrar los diferentes retrasos, por tanto, es importante contar con modelos completos que determinen la ubicación o ubicaciones óptimas de los controladores que interconectan los elementos de la red [21].

En [22] se propuso una guía para ayudar a tomar decisiones sobre la cantidad óptima de controladores para una topología teniendo en cuenta su respectiva ubicación geográfica, sin embargo, se presentan varios desafíos respecto al rendimiento (latencia y rendimiento), la escalabilidad de la red, tolerancia a fallas, hay que tener en cuenta que las soluciones dependen de la topología de la red y para un rendimiento óptimo se requiere una cantidad de controladores.

Los autores en [9] proponen para el problema de la ubicación del controlador una formulación matemática, definiendo los costos de la red como la suma del costo del switch y suma del costo del enlace. De acuerdo con lo anterior se formula un problema para minimizar el costo de la red obteniendo al resolver el problema de optimización se encuentre la estrategia óptima de ubicación del controlador.

Una sola instancia de controlador puede sufrir problemas de rendimiento, escalabilidad y confiabilidad a medida que va aumentando el tamaño de la red. Lo anterior se soluciona fragmentando la red en dominios más pequeños, cada uno supervisado por un controlador. Sin embargo, la implementación de múltiples controladores requiere una técnica para colocarlos de manera eficiente [22].

Los controladores centralizados en redes SDN son los encargados de diseñar estrategias para el reenvió de datos, mientras que en el plano de datos los switches o enrutadores solo

realizan el reenvió de paquetes de acuerdo con las estrategias recibidas para el enrutamiento de los controladores [9]. Considerando lo anterior, con la centralización del controlador, la programación de la red se puede automatizar y se puede cambiar dinámicamente según sea necesario. SDN también permite la flexibilidad de red, elimina la dependencia de ciertos proveedores y permite a una organización desarrollar nuevas innovaciones en aplicaciones de red [23].

2.3 PROTOCOLOS EN REDES SDN

Los propietarios pueden configurar la red de manera dinámica y eficiente mediante los controladores. Los controladores de SDN se comunican con el switch a través de protocolos como OpenFlow en él se indica la ruta de los paquetes eligiendo la mejor, también se pueden configurar los dispositivos de la red [24].

En un sistema de comunicación los protocolos son los que permiten que los ingenieros desarrollen sistemas de comunicación complejos, el diagnóstico de fallas o supervisión de la red [25]. OpenFlow es un protocolo clave en SDN pues es el protocolo DE FACTO en la API hacia el sur, es decir que es la interfaz de comunicación entre el controlador y la capa de datos.

Debido a la llegada de la nube y las arquitecturas de redes definidas por software (SDN), los protocolos están presentando numerosas mutaciones. OpenFlow es el protocolo descendente (southbound) desarrollado para SDN. Fue desarrollado en la universidad de Stanford y luego introducido por fabricantes como Nicira y VMware, OpenFlow define dos partes, una es la que contiene las colas, los transmisores y receptores de trama, con las tablas de flujo asociadas a los mismos y la segunda parte la que gobierna la comunicación entre el controlador mediante el protocolo de señalización OpenFlow [26].

La naturaleza de SDN es dinámica debido al tráfico de la red, ocasionando un desequilibrio de carga al momento de que las instancias del controlador estén cargadas provocando lentitud, esto puede ralentizar toda la plataforma de control porque no puede responder consultas en la ruta de datos. Para solucionar lo antes mencionado hay un estudio en el que se propone un protocolo de migración de carga nuevo y eficiente, lo que hace es

cambiar la carga de entrada con instancias de controlador sobrecargadas hacia instancias con carga ligera [27].

En la arquitectura OpenFlow, un controlador lógicamente centralizado administra los conmutadores proporcionando reglas que dictan su comportamiento al momento de manejar los paquetes. Para cubrir aspectos como la escalabilidad y la resistencia, conceptos como HyperFlow permiten la partición de redes OpenFlow en múltiples dominios que son manejados por controladores individuales [21].

OpenFlow es el protocolo SDN más utilizado. Los controladores basados en OpenFlow tienen dos interfaces la primera es la interfaz en dirección norte esta proporciona información para las aplicaciones y la segunda es la interfaz en dirección sur que proporciona la comunicación con los conmutadores [28].

En SDN la interfaz de comunicación representativa es OpenFlow, está inicialmente asume un solo controlador a causa de más simplicidad. Sin embargo, OpenFlow puede sufrir problemas de escalabilidad y rendimiento cuando se expande continuamente la red por tanto se propone luego de numerosos enfoques de varios controladores que lograran una arquitectura básica común con esfuerzos conjuntos [10],[8].

Algunas soluciones se basan en controladores distribuidos con el fin de mejorar la resistencia a fallas y disminuir los tiempos de respuesta en grandes redes [27].

El tema de seguridad en el diseño de OpenFlow debe preocuparse más a fondo en especial en protegerse entre los distintos nodos de los controladores. **Benton** evaluó la debilidad del protocolo, el creyó que tenía un método de certificación preciso y seguro entre conmutadores y controladores, pero el método no era obligatorio y no tenía mecanismos de seguridad en la capa de transporte entonces un atacante puede eludir la autenticación para realizar ataques maliciosos, a pesar de que descubrieron esta amenaza la investigación no introdujo una solución específica [29].

AVISPA es un protocolo de interacción de red que garantiza la seguridad y proporciona pruebas de seguridad completas aplicando el sistema de análisis, lo anteriormente mencionado es importante porque uno de los principales objetivos es abordar los problemas de seguridad de autenticación de la arquitectura SDN con el fin de cerrar la brecha de

confianza mutua entre distintos dominios de confianza proporcionando una base de seguridad para que puedan interactuar diferentes dominios de confianza [29].

El siguiente trabajo proponen un protocolo distribuido llamado SDN **Resource Discovery Protocol (SDN-RDP).** Este protocolo lo que hace en la red es distribuir y dividir los controladores para que cada controlador descubra una parte de la topología de la red con el fin de crear un árbol de latencia mínima enraizado en cada controlador y así se crea la capa de control, teniendo en cuenta que el protocolo no distribuye la carga entre los controladores por sí solo, sino que mantiene actualizado el estado de la red. Entonces con las rutas más cortas restringidas por retardo los controladores recopilan la información sobre el estado de la red de los nodos y deciden distribuir las decisiones de reenvió. El proceso es asincrónico es decir no existe un proceso de inicio global para activar la ejecución del protocolo y por tal motivo no se requiere conocimiento de la red. Los resultados de la simulación evidencian que el protocolo propuesto funciona de forma eficiente en grandes redes en términos de tiempo y carga [30].

Para la verificación y formalización del protocolo Openflow se sugiere la metodología y herramientas matemáticas, estos métodos pueden utilizarse de forma transparente con el fin de analizar las propiedades críticas y los requisitos antes de la implementación del sistema [31].

Los protocolos de enrutamiento heredados como BGP y OSPF, están desarrollados de forma completa sin embargo por su rígido y completo sistema han sido difíciles de adaptar al crecimiento de internet. La aparición de la red definida por software (SDN) ha traído esperanza para solucionar este problema beneficiándose de la ventaja de tener un control centralizado [32].

2.4 PROBLEMA DE LA UBICACIÓN DEL CONTROLADOR BASADO EN COSTOS.

El costo de una red requiere atención principalmente por el costo total, este consiste en el costo de implementación y consumo de energía, respecto al costo de implementación se refiere al costo del equipo de la red (switches y controladores) y el gasto operativo que incluye la instalación de controlador en las redes, el vínculo entre los controladores a los

switches y el consumo de energía representa más del 90 por ciento de los dispositivos de red con una carga baja [8].

En el siguiente artículo [33] se estudió la implementación de controladores capacitados para redes SDN definiendo algunas variables como la transmisión de señalización y el rendimiento de procesamiento de los switches en el peor de los casos, entonces formulan la implementación del controlador capacitado con problema de optimización restringido con el fin de minimizar el costo de la red proponiendo un algoritmo heurístico.

De acuerdo con lo anterior en un escenario práctico de redes SDN, los controladores pueden ubicarse al lado de los switches pero debido al costo de los controladores el número es mucho menor que el de los switches por tanto se puede asociar un controlador con múltiples switches. Entonces el número y ubicación de los controladores teniendo en cuenta la relación entre switches y controladores puede afectar la transmisión de señalización y el rendimiento de los procesos de la SDN, en estudios de implementación los controladores tradicionales asumen que tienen características ideales como suficiente capacidad de procesamiento y capacidad de cálculo entonces teniendo en cuenta los problemas del controlador tradicional este se convierte en uno para la implementación del controlador capacitado [33].

Cada vez que la red se amplía, es decir aumenta la cantidad de switches un solo controlador se vuelve ineficiente para controlar los dispositivos por tanto se requieren múltiples controladores, ya que cada controlador sólo puede controlar una mínima cantidad de switches. En las implementaciones prácticas de SND en redes de área amplia definidas por software trabajan dividiendo la red en muchos dominios y cada dominio contiene muchos switches controlados por más de un controlador o por más para obtener resistencia [34] [35].

Una de las principales vulnerabilidades es la reutilización de códigos. Debido a que el desarrollo de software moderno utiliza bibliotecas existentes, cuando la biblioteca utilizada contiene vulnerabilidades todo el software que depende de la biblioteca puede sufrir amenazas de seguridad, es decir que si dos controladores utilizan la misma biblioteca defectuosa compartirán las mismas vulnerabilidades incluso si se usan diferentes controladores un ataque puede comprometer a ambos [34].

Continuando con la idea anterior un único controlador no logra mitigar la propagación de los ataques por ende es necesario varios tipos de controlador para mejorar la seguridad, sin embargo, teniendo en cuenta un solo factor de seguridad da como resultado altos costos y ese costo no garantiza el requisito de seguridad. En consecuencia, se necesita un mecanismo para implementar un controlador inteligente de múltiples tipos que analice la seguridad y el costo para lograr implementar un controlador seguro y rentable [34].

En [15] referencian que la sobrecarga de sincronización entre controladores en SDN cableada tienen en cuenta dos métricas con el fin de calcular el costo de sincronización entre los controladores; la primera es el retardo de la sincronización y se define como el tiempo que tarde un controlador en darse cuenta de un evento generado por otro controlador, y la segunda es la cantidad de datos de sincronización que se intercambian entre los controladores. Los resultados son experimentales y se mostró que el retardo de sincronización es insignificante en la mayoría de los casos y para ganar más tolerancia a fallas, es necesario comprometer el costo de sincronización entre los controladores.

2.5 PROBLEMA DE LA UBICACIÓN DEL CONTROLADOR BASADO EN LATENCIA.

La clave para tener una SDN altamente adaptable, que pueda responder a las demandas cambiantes o recuperarse después de una falla de la red en un corto período de tiempo, es actualizar la configuración de manera efectiva [36].

La latencia del switch al controlador es fundamental para el rendimiento de la SDN adicionalmente la confiabilidad es un aspecto muy importante para las SDN ya que las fallas pueden ocasionar desconexión entre los controladores y switch o falla en cascada de otros controladores. Tras la falla del enlace se debe realizar una configuración en la ruta de respaldo entre los switch y controladores, es muy importante tener en cuenta que el retardo entre la ruta principal y la latencia de la ruta de respaldo están intrínsecamente en conflicto entre sí. Entonces los controladores tienden a colocarse cerca de los conmutadores para reducir la latencia del switch a controladores en las rutas primarias sin embargo el retraso de la ruta de respaldo puede ser alto debido a las rutas de desvío largas cuando ocurre una falla en el enlace [37].

En las investigaciones sobre la ubicación del controlador orientada a latencia el fin es optimizar el retardo entre los switches y controladores con el supuesto de que hay una red confiable disponible, en el siguiente artículo [37] estudian el problema de la ubicación del controlador para minimizar la latencia de las rutas primarias y de respaldo entre los switches y controladores asumiendo que hay como máximo un fallo de enlace en la red entonces formularon un nuevo problema de ubicación del controlador SDN multiobjetivo para minimizar el retraso de la comunicación entre el switch y el controlador tanto para los casos en que llegue a fallar el enlace.

De acuerdo con lo anterior proponen un algoritmo de la ubicación de controlador orientado a la latencia y confiable basado en metaheurísticas (RALO) para la ubicación de múltiples controladores multiobjetivo, entonces el algoritmo construye una solución inicial factible mediante un método de partición de red, luego se genera repetidamente nuevas soluciones con búsqueda de vecindad variable. Cuando se genera la nueva solución el algoritmo decide si acepta la nueva solución como una solución dominada al problema y realiza la operación de actualización en el conjunto de soluciones óptimas de Pareto. Los resultados de las simulaciones muestran que el algoritmo propuesto podía lograr un rendimiento competitivo de latencias de conmutador a controlador en los casos sin falla de enlace y con falla de enlace único, y el retraso acumulado de las rutas primarias y de respaldo entre los controladores y los switches [37].

En internet la latencia depende de la distancia en que viajan los paquetes en las redes WAN. Esta latencia se puede cambiar si se minimizan los saltos entre los sistemas autónomos [38].

Otra forma de minimizar la latencia de comunicación entre el switch y el controlador es realizar partición de la red. Sin embargo, el controlador no solo se comunica con switches sino también con más controladores con el fin de mantener la configuración de forma uniforme en el entorno de múltiples controladores. En la siguiente investigación [39] modelan una red de comunicación en un gráfico y formulan un problema de minimización de latencia de la comunicación, sin embargo, con el algoritmo propuesto en la investigación no fue posible reducir la latencia de comunicación entre controladores e interruptores obtenida por agrupamiento con K-means, pero dependiendo de los gráficos, este algoritmo puede reducir la latencia de comunicación entre controladores.

En la mayoría de los esquemas de implementación basados en latencia se enfocan en el retardo de la transmisión (TD) o el retardo de propagación (PD). Sin embargo, ningún trabajo ha tenido en cuenta la influencia de la latencia de control en el problema de la ubicación del controlador SDN, esta tiene mucho más sentido en una respuesta rápida a eventos de red. En el siguiente artículo [40] se propuso un algoritmo optimizado de latencia de control mínimo basado en un diseño de control codicioso teniendo como referencia el filtro de Kalman que puede lograr una medición de latencia de alta precisión.

Continuando con la idea anterior en el documento se tomó principalmente la latencia de control como la consideración más importante para resolver el problema de la ubicación del controlador. El principal desafío es minimizar la latencia de control con el número mínimo de controladores en pocas palabras se podía colocar controladores para maximizar la cantidad de nodos controlados dentro de un límite de latencia mientras se minimiza la cantidad controladores, en vez de minimizar la peor o promedio de latencia. La latencia de control es mínima precisa y limitada, mientras que se minimiza el número de controladores. Algunos experimentos han demostrado que el mínimo de latencia de control optimizado puede mejorar el desequilibrio al dividir dominios SDN y lograr el número máximo de nodos por controlador controlado [40].

2.6 PROBLEMA DE LA UBICACIÓN DEL CONTROLADOR BASADO EN CONFIABILIDAD.

Existe numerosa evidencia del aumento e importancia en la utilización de las redes de telecomunicaciones, este concepto se ha reforzado con la crisis de Covid-19 en donde algunos países se tuvo un incremento de la demanda derivada del uso de herramientas principalmente para el teletrabajo, estudios virtuales y también mayor demanda de entretenimiento como streaming de video y juegos en línea, lo anterior genera saturación de las redes degradando la calidad de las conexiones[2]. Adicionalmente se ha percibido que la sociedad se ha vuelto dependiente de la fiabilidad de las redes de telecomunicaciones [41].

Teniendo en cuenta lo anterior, proteger dichas infraestructuras de telecomunicaciones, incluidas las redes definidas por software (SDN), es muy importante dado que los proveedores deben garantizar a la sociedad el acceso constante a servicios confiables. Los

ataques dirigidos a redes SDN afectan su conectividad y así mismo la continuidad del servicio, estos ataques en los nodos de la red permiten dividir sus componentes y como los controladores de la red son limitados, da como resultado el aislamiento de una porción de nodos de los controladores provocando interrupciones en la disponibilidad del servicio [41].

En [41], presentan un enfoque de optimización que puede ser usado por el operador de una red SDN con el fin de ubicar los controladores de manera adecuada teniendo en cuenta un conjunto de ataques predecibles dirigidos en la topología de la red, entonces esos conjuntos se utilizan como datos de entrada para la optimización de la ubicación, realizándose mediante métodos de programación de enteros mixtos, en la optimización el impacto de los ataques se mide mediante la medida de disponibilidad de la red y para minimizar las consecuencias se considera controladores de respaldo adicionales. Es importante resaltar que el efecto de un ataque no se mide por la cantidad de controladores atacados, en realidad normalmente el atacante desconoce la ubicación de los controladores, si no en el número de conmutadores que pierden su conexión con el controlador.

El estudio mencionado anteriormente concluyo en que cuando la ubicación del controlador es tradicional es decir basada en medidas de centralidad y se selecciona ubicaciones de nodos centrales, cuando hay un alto porcentaje de nodos atacados los controladores de respaldo deben colocarse en la periferia de la red, ya que ese tipo de ubicación rara vez son atacadas y tiende a cubrir gran parte de los componentes conectados. La solución propuesta puede ayudar al operador de la red a tomar decisiones más precisas en los niveles de resiliencia de la red considerando una variedad de parámetros básicos como la intensidad del ataque y la cantidad de controladores de respaldo [41].

En [35] utilizan esquemas de SD-WAN para aprovechar la SDN multidominio y está diseñado para la ingeniería de tráfico alcanzando un 70% de utilización del enlace sin realizar bloqueo de los mensajes de control crítico reservando el ancho de banda para aquellos mensajes. Entonces proponen RetroFlow para lograr una sobrecarga de comunicación baja sin interrumpir el procesamiento normal de los controladores activos durante las fallas del controlador.

Continuando el estudio anterior la resistencia del control es una preocupación muy importante dado que un controlador SDN es un software de red instalado en un servidor

físico o virtual. Dado que muchas veces hay problemas inesperados como errores en el hardware, software o fallas eléctricas, un controlador podría fallar por algunos de los factores antes mencionados y luego todos los switches conectados estarían fuera del control es decir fuera de línea. Entre las soluciones existentes para mantener la resistencia se pueden clasificar en dos tipos: (1) reasignación de switches y (2) ubicación del controlador. La solución del primer tipo es cambiar dinámicamente el control de los switches que se encuentran fuera de línea a switches activos. No obstante, si esta una SDN saturada, los controladores alcanzarían su límite de procesamiento, entonces quedaría poco espacio para los controladores activos porque deberían aceptar los switches fuera de línea sin sobrecargar los controladores activos, de lo contrario esto causaría una posible degradación del rendimiento por ejemplo aumentar la sobrecarga de comunicación o incluso causar la falla del controlador en cascada. Por otro lado, la ubicación del controlador elegiría cuidadosamente las ubicaciones físicas de los controladores con el fin de optimizar el rendimiento del control en caso de fallas del controlador como minimizar la latencia entres los switches y controladores de respaldo o minimizar la latencia entre el controlador principal, los switches y controladores de respaldo [35].

En el siguiente trabajo de grado [42] utilizan un algoritmo genético para elegir una ruta factible con criterios de retardos, caudal eficaz y confiabilidad es decir en encontrar la mejor trayectoria para transmitir información dado que este proceso ningún enrutamiento convencional lo realiza, al usar este algoritmo se reduce el proceso de selección de rutas factibles con múltiplos parámetros usando metaheurísticas y métodos de selección basados en máximos y mínimos evitando emplear procesos que requieren de una demanda grande de recursos y tiempo de procesamiento.

3 VERIFICACIÓN DE RESULTADOS DEL PROYECTO SDNCON.

Con el fin de verificar los resultados obtenidos en los estudios anteriores del proyecto SDNCon [43] [44]. Se realizó la replicación de las pruebas ejecutadas con los mismos escenarios que se planteó en cada estudio.

3.1 CONDICIONES DEL TRABAJO

Para la replicación de las pruebas se dejó los mismos escenarios planteados en cada estudio como son los costos de los enlaces y licencias. Entonces a continuación se indican los parámetros que se usó en los estudios y en las pruebas:

ALGORITMO TABÚ

- maxFlowSetupLatency: Latencia de configuración de flujo máximo.
- maxInterControllerLatency: Latencia máxima del Inter controlador.
- percentageNode: Porcentaje de nodo.
- movementType: Tipo de movimiento.
- memorySize: Tamaño de la memoria.
- Iterations: Iteraciones.

Teniendo en cuenta lo anterior los parámetros utilizados en el estudio [43] son los siguientes:

Parámetros: maxFlowSetupLatency: 80, maxInterControllerLatency: 100, percentageNode: 10%, movementType: 0, memorySize: 70, Iterations: 100.

Para las pruebas se tuvo que ajustar los parámetros en las siguientes topologías YourDataServices, Xeex, Digex, Agis,ATTNorthAmerica, ATMnet, Bics, NetworkUsa, Xspedius, BBNplanet quedando de la siguiente manera maxFlowSetupLatency: 50, maxInterControllerLatency: 70, percentageNode: 10%, movementType: 0, memorySize: 70, Iterations: 100 y para CRL, Psinet y Bellcanada maxFlowSetupLatency: 70, maxInterControllerLatency: 100, percentageNode: 10%, movementType: 0, memorySize: 70, Iterations: 100.

ALGORITMO GENÉTICO

Para la simulación del algoritmo genético se consideró los siguientes parámetros: Latencia de configuración de flujo máximo (LCFM): 50 ms, 75 mS, 100 ms ó 200 ms, 225 ms, 250 ms ó 275 ms, 300 ms, 325 ms. Esto depende de la topología que se trabaje. para las topologías ATMNet, YorkDataServices, Digex, Bics, CRL, NetworkUsa y PalmettoNet se tuvo en cuenta un valor de 50ms, 75ms y 100ms que se representa en

milisegundos. Para la topología 60 Packetexchange se consideró los valores de 275ms y 300ms. Para la topología de Darkstrand se tomó los valores de 200ms, 225ms y 250ms y finalmente para Bellcanada se tomó 200ms, 225ms y 250ms.

Para las topologías ATMNet, YorkDataServices, Digex, Bics, CRL, NetworkUsa y PalmettoNet se tuvo en cuenta un valor de 50, 75 y 100 que se representa en milisegundos. Para la topología Packetexchange se consideró los valores de 275 y 300. Para la topología de Darkstrand se tomó los valores de 200ms, 225ms y 250ms y finalmente para Bellcanada se tomó 200ms, 225ms y 250ms.

El porcentaje del nodo expresa la cantidad de controladores que se tendrán en la topología, el porcentaje de nodo en este algoritmo es de 1, 5, 10 o 15 y se aplica cada generación que contiene 10 soluciones, entonces por ejemplo si el PN es de 15 quiere decir que 1 o 2 soluciones de la generación tendrán cambios.

La tasa de mutación, indica la cantidad de mutaciones que se tendrán en la iteración, para este trabajo se maneja al 0.5 y 1. Por ejemplo si se tienen 10 soluciones en una generación y se realiza la mutación al 1, se tendrá una nueva población de 20 soluciones, de las cuales el algoritmo descarta las 10 soluciones más costosas.

Para las pruebas se utilizó los siguientes parámetros:

Para las topologías Digex, NetworkUsa, CRL, ATMnet, Bellcanada tuvo una Latencia de configuración de flujo máximo de 50ms, 100ms y 25ms que se representa en milisegundos.

Las Topologías YourDataServices y Darkstrand tuvo una Latencia de configuración de flujo máximo de 100ms, 150ms y 40ms que se representa en milisegundos.

Las Topologías Bics y ATTNorthAmerica tuvo una Latencia de configuración de flujo máximo de 100ms, 150ms y 45ms que se representa en milisegundos.

El resto de los parámetros se usó los mismos del estudio.

3.2 ESPECIFICACIONES TÉCNICAS DEL AMBIENTE DE PRUEBAS.

Para realizar la respectiva verificación se creó un ambiente de pruebas muy similar al utilizado en los estudios anteriores [43] [44] para eso se instaló una máquina virtual sobre el software Oracle VM VirtualBox Administrador con versión 5.2.44, corriendo el sistema operativo Ubuntu 20.04.2 LTS con memoria RAM de 4GB y capacidad de disco de 78 GB en estado sólido (HDD).

3.3 TOPOLOGIAS UTILIZADAS.

Topology Zoo es un proyecto en curso para recopilar topologías de redes de datos de todo el mundo adicionalmente cuentan con grafos en formato GML y este es el archivo que usamos con el fin de simular un ambiente más realista a las redes WAN que son las utilizadas actualmente. Cada grafo está compuesto por nodos de la red y los enlaces que conectan los distintos nodos.

3.4 REPLICACIÓN DE PRUEBAS MÉTODO HEURÍSTICO ALGORITMO TABÚ.

Teniendo en cuenta la información mencionada anteriormente se realizó las respectivas pruebas a los métodos heurísticos con algunas de las topologías aplicadas en el estudio realizado [43], para esto de las 19 topologías del estudio, se seleccionó 13 para realizar las pruebas de verificación, a continuación, indico las topologías seleccionadas.

Tabla 1. Topologías simuladas- método heurístico Tabú.

Topología	Nodos
YourDataServices	23
xeex	24
Digex	31
Agis	25
ATTNorthAmerica	25
CRL	33
Psinet	24
ATMnet	21
Bics	33
NetworkUsa	35
Xspedius	34
Bellcanada	48
BBNplanet	27

3.4.1 RESULTADOS

En la figura 3 se evidencia los resultados de las pruebas realizadas al método heurístico algoritmo Tabú respecto a costos.

Figura 3. Resultados pruebas método heurístico Tabú.

En la tabla número 2 complementa la información del gráfico anterior tenemos los resultados el tiempo de simulación de cada topología más el costo.

Tabla 2. Resultados de las topologías simuladas- método heurístico Tabú.

Topología	Nodos	Tiempo de Simulación	Resultado Costo
YourDataServices	23	2m 5,14s	1930
xeex	24	2m 37,36s	2030
Digex	31	6m 3,772s	2355
Agis	25	23m 5,857s	2440
ATTNorthAmerica	25	3m 37,53 s	2520
CRL	33	10m 27,827	2895
Psinet	24	8m 32,711 s	1980
ATMnet	21	2m 38, 87 s	1655
Bics	33	10m 40, 25 s	3730
NetworkUsa	35	8m 59,83 s	2795
Xspedius	34	37m 7,08s	3670
Bellcanada	48	55m 7,16s	4250
BBNplanet	27	29m 17, 87 s	2945

3.4.2 COMENTARIOS SOBRE LAS PRUEBAS ALGORITMO TABÚ.

- El tiempo de simulación total de las 13 topologías fue de 202 minutos con 57 segundos más de 3 horas de simulación.
- El tiempo de simulación para las topologías YourDataServices, Xeex y ATMnet tuvo en promedio 2 minutos con 41 segundos.
- La topología ATTNorthAmerica tuvo un tiempo de simulación de 3 minutos con 37 segundos.
- Las topologías Digex, CRL, Psinet, Bics y NetworkUsa tuvo un tiempo promedio de simulación de 8 minutos con 77 segundos.
- Las topologías BBNplanet, Agis y Xspedius tuvo un tiempo promedio de simulación de 30 minutos con 12 segundos.
- La topología Bellcanada tuvo un tiempo de simulación de 55 minutos con 7 segundos siendo uno de los tiempos más altos de simulación.
- En cuanto a los costos Bellcanada fue la que tuvo el costo más alto en comparación con las otras topologías y Bics fue la segunda, el resto de topología tuvo costos muy similares.
- En las topologías ATTNorthAmerica, CRL, Bellcanada y NetworkUsa CRL fue necesario aumentar el MFSL (maxFlowSetupLatency) y MCL (maxInterControllerLatency), porque el tiempo de simulación se estaba demorando más que el promedio de las otras topologías por tanto fue necesario repetir las pruebas en esas topologías.

3.5 REPLICACIÓN DE PRUEBAS MÉTODO HEURÍSTICO ALGORITMO GENÉTICO

Para el algoritmo genético se utilizó algunas de las topologías aplicadas en el estudio anterior [44], para esto de las 10 topologías del estudio, se seleccionó 6 para realizar las pruebas de verificación, a continuación, indico las topologías seleccionadas.

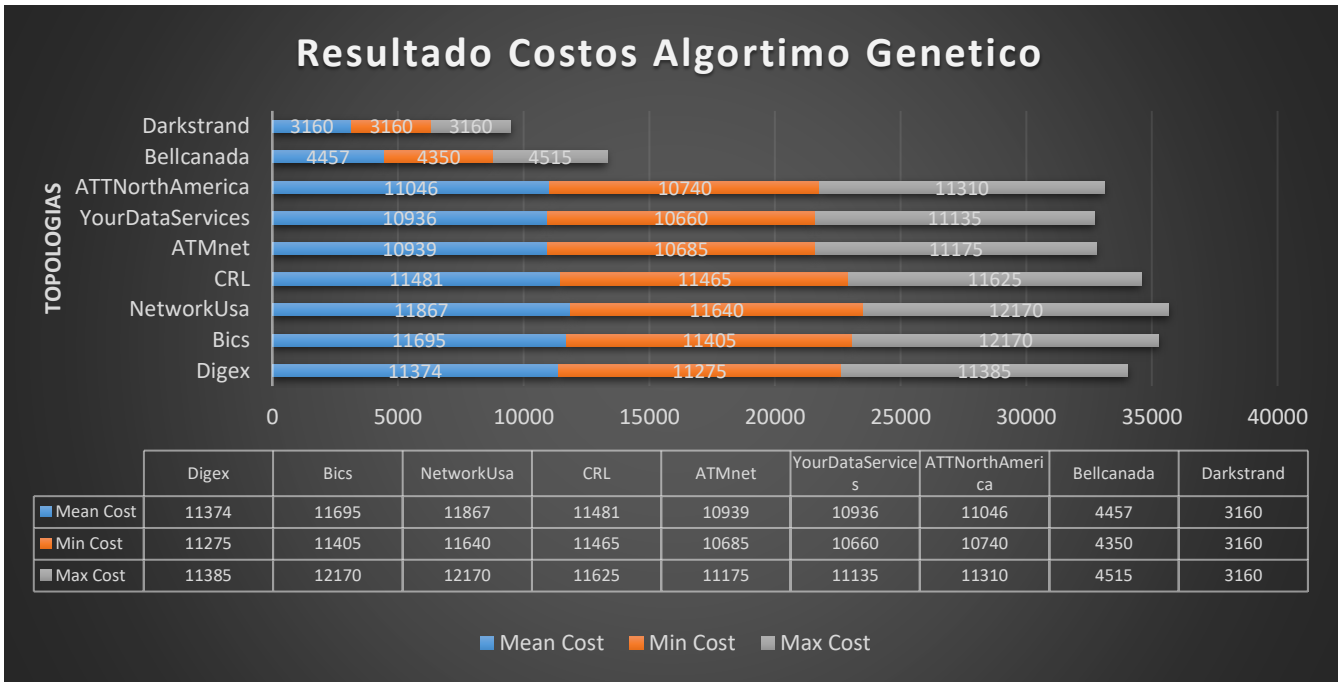
Tabla 3. Topologías simuladas-método heurístico algoritmo genético.

Topología	Nodos
Digex	31
Bics	33
NetworkUsa	35
CRL	33
ATMnet	21
YourDataServices	23
ATTNorthAmerica	25
Bellcanada	48
Darkstrand	28

Es importante resaltar que el algoritmo genético utiliza uno de los archivos que generaba el algoritmo tabú (output.txt) sin embargo como las topologías presentaban errores no fue posible probar algunas, pero para eso lo que se realizó fue reemplazarlas por unas que tuvieran la misma cantidad de nodos. En cuanto a los gráficos mostrados en el estudio no se pudo generar de la misma forma, dado que presentaban errores en el código al momento de ejecutarlos, sin embargo, por consola se obtuvo los resultados que permitió realizar las gráficas que se presentan a continuación.

3.5.1 RESULTADOS

En la figura número 4 se evidencia los resultados de las pruebas realizadas al método heurístico algoritmo genético respecto a costos.



	Digex	Bics	NetworkUsa	CRL	ATMnet	YourDataServices	ATTNorthAmerica	Bellcanada	Darkstrand
Mean Cost	11374	11695	11867	11481	10939	10936	11046	4457	3160
Min Cost	11275	11405	11640	11465	10685	10660	10740	4350	3160
Max Cost	11385	12170	12170	11625	11175	11135	11310	4515	3160

Figura 4. Resultados Costos Método Heurístico Algoritmo Genético.

En la tabla número 4 se complementa la información del gráfico anterior adicionalmente se tiene los resultados del tiempo de simulación de cada topología más el mínimo, medio y máximo costo.

Tabla 4. Resultados topologías simuladas- método algoritmo genético.

Topología	Nodos	Tiempo de Simulación	Mean Cost	Min Cost	Max Cost
Digex	31	2 días, 4 horas 29 minutos 23 s	11374	11275	11385
Bics	33	2 días, 6 horas 59 minutos 63 s	11695	11405	12170
NetworkUsa	35	3 días, 17 horas 04 minutos	11867	11640	12170
CRL	33	1 día, 4 horas 09 minutos 27 s	11481	11465	11625
ATMnet	21	14 horas, 18 minutos,12s	10939	10685	11175
YourDataServices	23	1 día 11 horas, 03 minutos ,44s	10936	10660	11135
ATTNorthAmerica	25	2 días 6 horas 25 minutos 27 s	11046	10740	11310
Bellcanada	48	3 horas 58 minutos 31s	4457	4350	4515
Darkstrand	28	3 días 01 hora 23 minutos	3160	3160	3160

3.5.2 COMENTARIOS SOBRE LAS PRUEBAS ALGORITMO GENÉTICO.

- El tiempo de simulación en total fue de 16 días y 18 horas.
- La topología NetworkUsa fue la que tuvo el tiempo de simulación más alto con 3 días, 17 horas y la segunda Darkstrand con 3 días y 1 hora.
- Las topologías Digex, Bics, ATTNorthAmerica tuvo tiempo de simulación muy similar con un promedio de 6 días, 17 horas y 53 minutos.
- La topología CRL y YourDataServices tuvieron un tiempo de simulación de 2 días, 15 horas y 13 minutos en promedio.
- La topología con menos tiempo de simulación fue Bellcanada con 03 horas y 58 minutos y la segunda ATMnet con 14 horas y 09 minutos.
- En costos la topología Darkstrand y Bellcanada fue la que tuvo menor costo, el resto tuvo costos muy similares.
- En las topologías Bics, YourDataServices, ATTNorthAmerica y Darkstrand fue necesario aumentar el MFSL (maxFlowSetupLatency) porque el tiempo de simulación se estaba demorando más que el promedio de las otras topologías, por tanto, fue necesario repetir la prueba en esas topologías.

3.6 DISCUSIÓN

Se realizo las pruebas a los métodos heurísticos, sin embargo, teniendo en cuenta que lo implemento en código Python y el algoritmo genético tiene como base el código Python del algoritmo Tabú, se obtuvo resultados inesperados frente al tiempo de ejecución. En las pruebas realizadas los tiempos de simulación fueron altamente diferentes (poco minutos en Tabú contra varias decenas de horas en Genético). Así mismo los costos del algoritmo genérico fue más altos respecto al algoritmo Tabú, por lo que se puede concluir que el algoritmo Tabú tiene mejor desempeño y menor costo respecto al algoritmo genético.

Las siguientes topologías usadas en los estudios no fueron posible utilizarlas para replicar las pruebas ya que presentaban errores.

Algoritmo Tabú:

Tabla 5. Topologías no utilizadas método heurístico Tabú.

Topologías Método Tabú	Nodos
Darkstrand	28
HurricaneElectric	24
IntegraTelecom	27
Packetexchange	21
PalmettoNet	45
Claranet	15

Para el caso del algoritmo genético:

Tabla 6. Topologías no utilizadas método heurístico algoritmo genético.

Topologías Método Genético	Nodos
Packetexchange	21
IntegraTelecom	27
PalmettoNet	45

4. COMPARACIÓN DE RESULTADOS

Es importante resaltar que los estudios [43],[44] tuvo como base el mismo modelo matemático, presentado en [45]. A continuación, se realiza la descripción del modelo matemático utilizado.

Cuando los swiches reciben un paquete cuyo destino desconocen, este envía una solicitud para el restablecimiento del flujo (SEF) al controlador que los gobierna, las SEF se encolan en las memorias de los controladores con el fin de ser procesadas. En el estudio consideran que las SEF se procesan bajo la política “First in, first out” es decir no hay ningún tipo de prioridad.

En el momento de la formulación del modelo matemático se asumió que se conocía la siguiente información:

- Retardo máximo permitido en la red para las comunicaciones del switch al controlador (latencia de configuración de flujo) y para las comunicaciones entre switches.
- El costo de asignar una licencia para ejecutar un controlador en un switch.
- Las capacidades de hardware de los switches para ejecutar las licencias de los controladores y manejar las solicitudes de configuración de flujo en términos de la demora de procesamiento para la configuración de flujo.
- La latencia de reenvío y el costo de usar un enlace que pertenece a un proveedor de servicios dado entre cada par de nodos.

Con base en esta información formulo el modelo de la siguiente manera:

- V, Conjunto de nodos.
- E, Conjunto de enlaces *con* $E \subseteq V \, x \, V$.
- $G\ (V, E)$, la red con el conjunto de nodos V y enlaces E .
- C, *C*onjunto de las posibles ubicaciones del controlador con $C \subseteq V$.
- $L_{u,v}^{p}$, la latencia de reenvío (en milisegundos) desde el nodo $u \in V$ al nodo $v \in V$ para un proveedor de servicios P.
- B, el costo (en $) de ejecutar una licencia de controlador.
- α_c, el número de solicitudes promedio presentes en un controlador.
- $\sigma_{\ c}$ el tiempo promedio (en milisegundos) para manejar un número determinado de solicitudes de configuración de flujo en un controlador.
- P_c, que es el retraso de procesamiento (en milisegundos) para la configuración del flujo con $P_c = \alpha_c \, \sigma_c \, \forall \, c \in C$.
- $\lambda_{u,v}^{p}$ el costo (en $) de utilizar una ruta de reenvío del proveedor de servicios P entre el nodo $u \in V$ al nodo $v \in V$.

Adicionalmente tenemos en cuenta las siguientes limitaciones:

- La latencia máxima de configuración de flujo (en milisegundos).
- la latencia máxima entre controladores (en milisegundos).

Variables:

- x_c una variable binaria tal que $x_c = 1$ sí y sólo si un controlador se implementa en una posible ubicación del controlador con $c \in C$.
- $x_{u,c,}$una variable binaria tal que $k_{u,c,} = 1$ sí y sólo si un nodo está asignado al controlador $c \in C$.
- $z_{u,c,}$ una variable binaria tal que $z_{u,c,} = 1$ sí y sólo si un nodo $u \in C$ y un nodo $v \in C$.
- $\delta_{u,}$una variable que representa la latencia entre un nodo $u \in V$ Y y su controlador $c \in C$.

Función de costo

El objetivo es minimizar los costos relacionados con el control de la red dado por:

- El costo de ejecutar licencias de controlador.

$$C_c(x) = \sum_{c \in C} B\, x_c$$

- El costo de enviar información de control entre un conmutador y su controlador asignado.

$$C_s(k) = \sum_{u \in V} \sum_{c \in C} \sum_{p \in P} \lambda_{u,v}^{p} k_{u,c}$$

- El costo de enviar información de control entre controladores.

$$C_w(z) = \sum_{u \in C} \sum_{v \in C} \sum_{p \in P} \lambda_{u,v}^{p} z_{u,v}$$

Teniendo en cuenta lo mencionado anteriormente el modelo para el problema de ubicación del controlador (CPP) quedó de la siguiente manera.

$$Minimize\ (C_c(x) + C_s(k) + C_x(z))$$

Adicionalmente está sujeto a lo siguiente:

- Se debe asegurar que un switch sea controlado por un solo controlador.
- Se debe asegurar que un controlador sea desplegado en un solo switch.
- El retardo máximo establecido de un flujo a través de un SEF $(\emptyset)$, la definió por la siguiente ecuación:

  $$2l_{u,v}^{p} + p_c \leq \emptyset$$

  Es decir que el retardo de ida y vuelta sumado al tiempo promedio para atender un SEF debe ser inferior al valor máximo para establecer el SEF.

- El retardo máximo entre controladores (η) se definió la siguiente ecuación. $2\sum_{u\in C} L_{u,v}^{p} z_{u,v} \leq \eta$, donde C corresponde al conjunto de controladores y $z_{u,v}$ es una variable binaria tal que $z_{u,v} = 1$ si y sólo si hay un nodo $u \in C$ y un nodo $v \in C$, es decir el enlace conecta con dos controladores. Esto garantiza que el retardo de ida y vuelta para un enlace entre controladores debe ser menor o igual al máximo retardo permitido.
- Hay que asegurar que la latencia switch-controlador sea 0, cuando un controlador se despliegue en el mismo switch.

Los dos estudios que se comparó utilizan el modelo presentado para solucionar el problema de la ubicación del controlador por métodos heurísticos. A continuación, se hizo una descripción general de los procesos usados por cada método, junto con la comparación de los siguientes aspectos:

- Costo final de la mejor solución encontrada
- Tiempo de ejecución del algoritmo

4.1 COMPARACIÓN DE RESULTADOS MÉTODO HEURÍSTICO ALGORITMO TABÚ.

En el estudio [43] se definió una red SDN tipo WAN, compuesta por switches, controladores y enlaces, el plano de datos está compuesto por switches y estos están gobernados por controladores (plano de control).

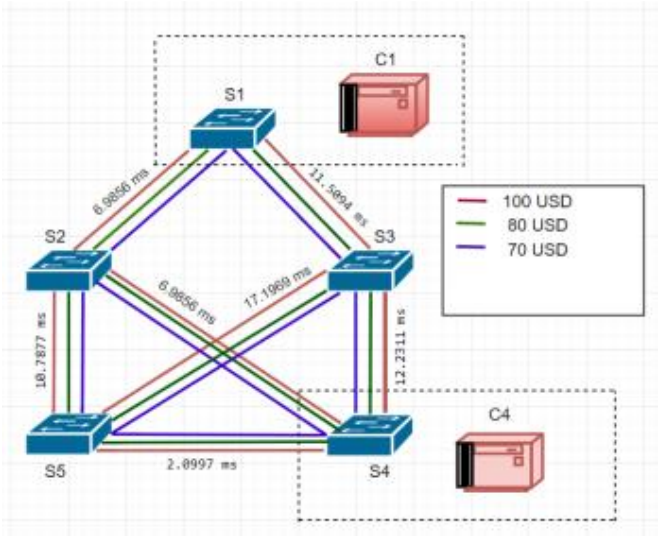


Figura 5. Red SDN.

Fuente: Tabú Search para el problema de ubicación del Controlador en Software Defined Networks.

Para solucionar el CPP, se definió un algoritmo Tabú en una red SND modelada. La calidad de la solución se mide por el costo de esta, es decir el menor costo y la mejor solución.

Para eso se tuvo algunos conceptos claves para trabajar el algoritmo Tabú, la búsqueda tabú es una estrategia local con una estructura flexible de memoria, tiene dos características muy importantes la primera es una memoria adaptativa y la segura es la memoria responsiva, se mueve siempre al mejor punto de solución vecina disponible y la lista tabú es la parte esencial de este método metaheurístico ya que permite guardar soluciones y no volver a visitarlas nuevamente, esta lista es actualizada con base a una memoria de corto plazo.

De acuerdo con lo anterior para la aplicación del algoritmo Tabú realizo la generación de soluciones vecinas porque esta puede ser vista como una búsqueda local o vecindario, iterando de un punto de solución a otra solución.

Generación de vecinos por movimiento de controladores, este movimiento funciona a partir de una solución aleatoria inicial que cumpla las restricciones planteadas en el estudio. Después de esta solución se identifican los controladores activos y no activos, para así mismo escoger un switch en el que no se ha desplegado un controlador es decir si el switch está inactivo cambia a estado activo, al igual de manera análoga se escoge un switch en el

que se ha desplegado un controlador activo se cambia el estado de este a activo. Por último, el listado de switches controlado por el controlador inicialmente activo es asignado al controlador inicialmente inactivo.

Generación de vecinos por variación en los enlaces, este movimiento se basa en la activación/desactivación de enlaces entre dos switches. Cada proveedor de enlaces tiene asignado un precio destino a cada enlace de un nodo A a un nodo B. Teniendo en cuenta de que dos nodos tienen múltiples enlaces y solo uno de ellos está activo.

Resultado algoritmo Tabú

Mediante la programación orientada a objetos (POO) se realizó el modelo de red SDN, los switches, controladores, enlaces y la red se moldeo como clases y Python fue el lenguaje de programación utilizado para implementar y simular el algoritmo Tabú. Para la simulación del algoritmo Tabú utilizo 2 infraestructuras la primera una máquina virtual en la nube de AWS y la segunda una máquina virtual mediante el software Oracle VM VirtualBox, Se consideraron 19 topologías y el tiempo de simulación fue aproximadamente 78.8 horas. La figura 6 presenta el resultado de los costos de cada topología [43]. Cabe resaltar que no hay una indicación clara en el documento sobre los valores exactos de la solución para cada topología, por lo que se deben estimar a partir de la figura.

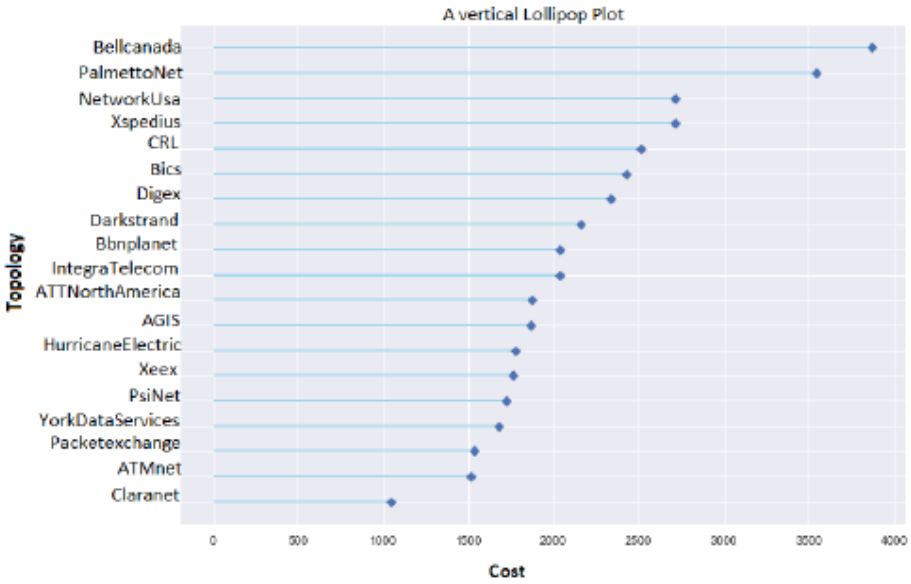


Figura 6. Resultados de los costos para las 19 topologías estudiadas.

Figura 6. *Fuente: Tabú Search para el problema de ubicación del Controlador en Software Defined Networks.*

4.1.1 PRUEBAS DEL ALGORITMO

La verificación de los resultados del algoritmo se realizó según las especificaciones indicadas en la sección 3.2 ESPECIFICACIONES TÉCNICAS DEL AMBIENTE DE PRUEBAS.

La figura 7, junto con la tabla 7 presentan los resultados de las pruebas realizadas.

Figura 7. Resultados estudio método heurístico Tabú.

Tabla 7. Resultados estudio y pruebas método heurístico Tabú.

Topología	Nodos	Tiempo de Simulación	Resultado Costo Prueba	Costo [43][1]
xeex	24	2m 37,36s	2030	1700
Digex	31	6m 3,772s	2355	1750
Agis	25	23m 5,857s	2440	2300
ATTNorthAmerica	25	3m 37,53 s	2520	1800
CRL	33	10m 27,827	2895	1800
Psinet	24	8m 32,711 s	1980	2600
ATMnet	21	2m 38, 87 s	1655	1700
Bics	33	10m 40, 25 s	3730	1500
NetworkUsa	35	8m 59,83 s	2795	2400
Xspedius	34	37m 7,08s	3670	2800
Bellcanada	48	55m 7,16s	4250	2700
BBNplanet	27	29m 17, 87 s	2945	3800
xeex	24	2m 37,36s	2030	2100

[1] Costo estimado a partir de la Figura 6.

Teniendo en cuenta la información mencionada anteriormente y teniendo los resultados de la replicación de las pruebas ver figuras (6 y 7) se puede concluir lo siguiente:

- A pesar de que se usó los mismos escenarios el tiempo de simulación de la replicación de las pruebas fue menor al realizado en el estudio, la diferencia está en 75 horas.
- Algunas topologías en el estudio anterior presentaron menor costo y en las pruebas realizadas mayor costo o viceversa. Esto se debe a la naturaleza heurística del algoritmo, pues en cada iteración obtiene una solución diferente que puede ser más cercana al óptimo real (costo valor exacto más pequeño logrado con las condiciones definidas).

4.2 COMPARACIÓN DE RESULTADOS MÉTODO HEURÍSTICO ALGORITMO GENÉTICO.

En el estudio [44] se utilizó un algoritmo genético aplicado a una red SDN. Se seleccionó un algoritmo genético porque su tiempo de procesamiento es mucho menor respecto a las técnicas exactas dado que estas tienen mayor duración en adquirir un resultado, adicionalmente también opera de manera simultánea con varias soluciones y no de manera secuencias (búsqueda exhaustiva) o de memoria programada (Tabú).

El algoritmo genético es una forma de tratar el problema de la ubicación del controlador, para eso se consideró que parte de un conjunto de soluciones iniciales con un tamaño de población limitada.

Resultado algoritmo genético

En el estudio se planteó cinco clases de resultados, sin embargo, la comparación se realiza sobre los costos y tiempo de simulación.

- Topología inicial, topología optimizada vs Costo

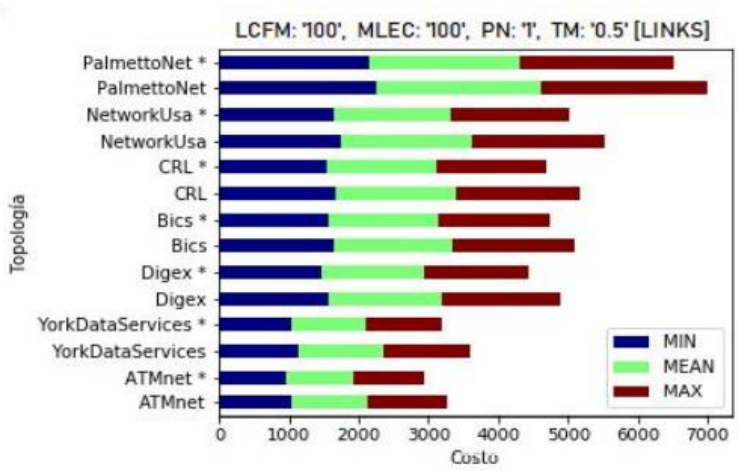


Figura 8.Costos de cada topología. [Resultados al simular en python]

Fuente: Algoritmo genético para el problema de la ubicación del controlador en una red definida por software (RDS).

- Tiempo de cómputo vs topología

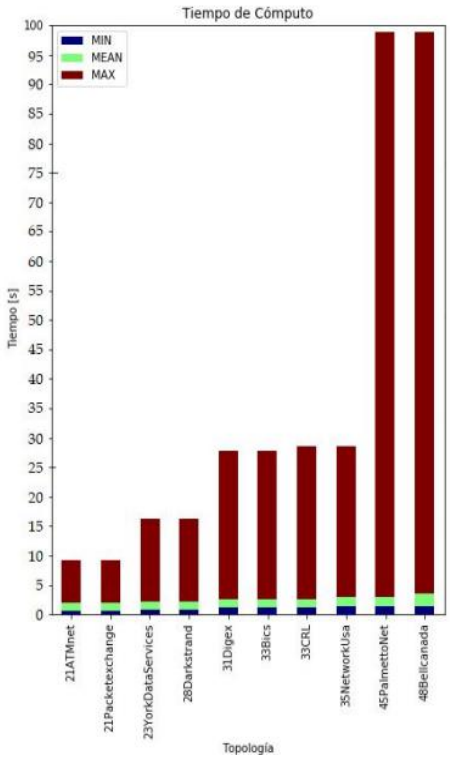


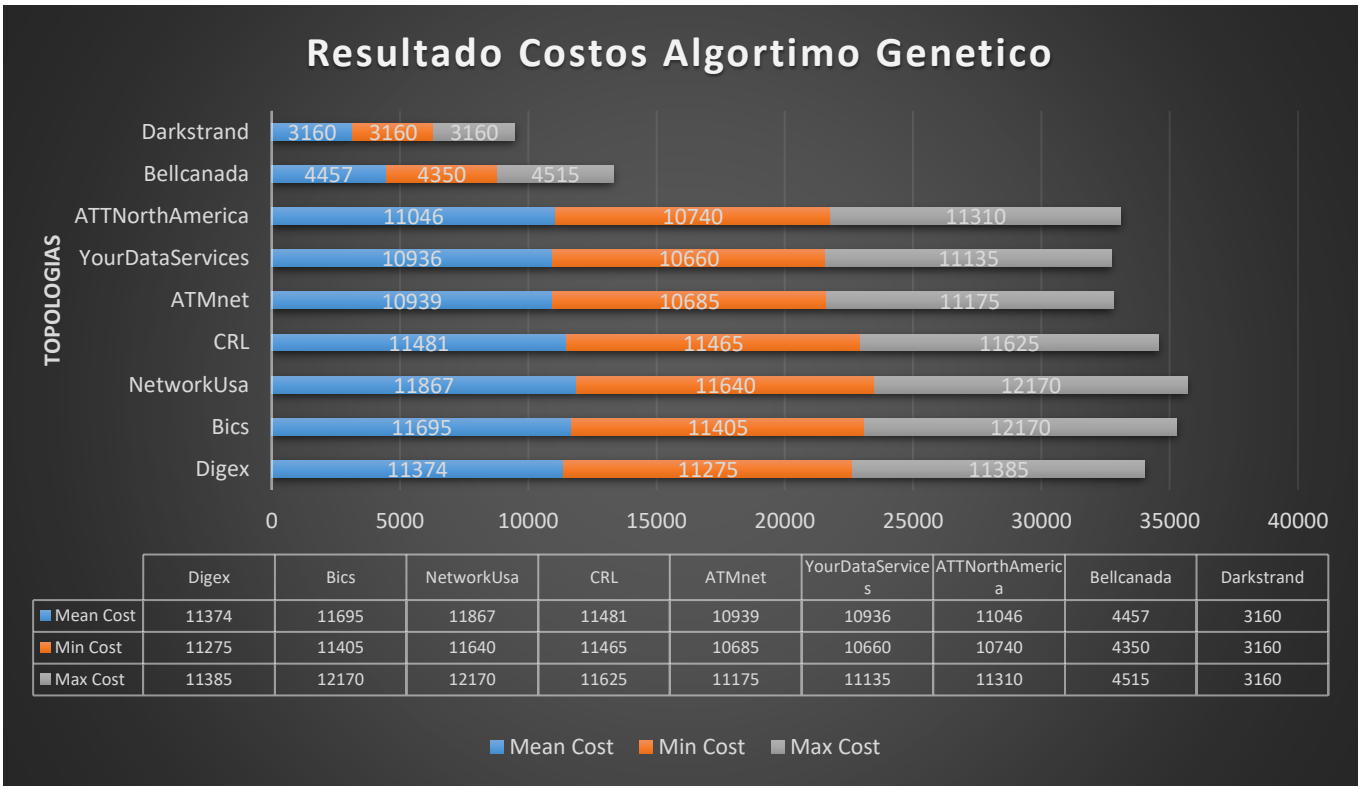
Figura 9.Tiempos de cómputo en las topologías de este trabajo. [Resultados al simular en python].
Fuente: Algoritmo genético para el problema de la ubicación del controlador en una red definida por software (RDS)

Para la simulación del algoritmo genético se utilizó un computador Intel® Core™ i7-8750H Processor 2.2 GHz con 32 Gb de ram, Se considero 10 topologías, el tiempo reportado de simulación en [44] fue de 173 horas. La figura 8 presenta el resultado de los costos de cada topología en [44]. De nuevo, al igual que en [43], no se aclaró el valor exacto de cada solución. Adicionalmente, el tipo de gráfica se debe interpretar de manera diferente, ya que

en este caso se tiene los resultados de 3 soluciones apilados y se debe tomar el valor de solo uno de ellos.

4.2.1 PRUEBAS DEL ALGORITMO

La verificación de los resultados del algoritmo se realizó según las especificaciones indicadas en la sección 3.2 ESPECIFICACIONES TÉCNICAS DEL AMBIENTE DE PRUEBAS.



	Digex	Bics	NetworkUsa	CRL	ATMnet	YourDataServices	ATTNorthAmerica	Bellcanada	Darkstrand
Mean Cost	11374	11695	11867	11481	10939	10936	11046	4457	3160
Min Cost	11275	11405	11640	11465	10685	10660	10740	4350	3160
Max Cost	11385	12170	12170	11625	11175	11135	11310	4515	3160

Figura 10. Resultados método heurístico algoritmo genético.

Tabla 8. Resultados estudio y pruebas método heurístico algoritmo genético.

Topología	Nodos	Tiempo de Simulación	Mean Cost	Min Cost	Max Cost	Mean Cost [44][2]	Min Cost [44][3]	Max Cost [44][4]
Digex	31	2 días, 4 horas 29 minutos 23 s	11374	11275	11385	1600	1600	1600
Bics	33	2 días, 6 horas 59 minutos 63 s	11695	11405	12170	1660	1640	1700
NetworkUsa	35	3 días, 17 horas 04 minutos	11867	11640	12170	1750	1800	1950
CRL	33	1 día, 4 horas 09 minutos 27 s	11481	11465	11625	1750	1800	1950
ATMnet	21	14 horas, 18 minutos,12s	10939	10685	11175	1100	1000	1200
YourDataServices	23	1 día 11 horas, 03 minutos ,44s	10936	10660	11135	1100	1000	1200
ATTNorthAmerica	25	2 días 6 horas 25 minutos 27 s	11046	10740	11310	-	-	-
Bellcanada	48	3 horas 58 minutos 31s	4457	4350	4515	-	-	-
Darkstrand	28	3 días 01 hora 23 minutos	3160	3160	3160	-	-	-

La figura 10, junto con la tabla 8 presentan los resultados de las pruebas realizadas.

- A pesar de que se usó los mismos escenarios el tiempo de simulación de la replicación de las pruebas fue mayor al realizado en el estudio, la diferencia está en días.

- Algunas topologías en el estudio anterior presentaron menor costo y en las pruebas realizadas mayor costo o viceversa.

- Algunos de los resultados mostrados en el estudio no se pudieron comparar dado que al momento de generar las gráficas mostraba inconsistencia debido a que el código tenía documentación incompleta para la generación de las gráficas y datos en general.

[2] Costo estimado a partir de la Figura 9.
[3] Costo estimado a partir de la Figura 9.
[4] Costo estimado a partir de la Figura 9.

- No se pudo comparar todas las topologías debido a que presentaban errores al momento de probarlas porque el archivo GML que se descarga de topology-zoo no tenía la sintaxis correcta.

- Dos de las topologías reemplazadas presentaron menor costo respecto a las probadas del estudio anterior.

- En la gráfica de costos del estudio anterior solo están 7 de las 10 topologías utilizadas, estas no se encuentran incluidas en los costos generales de las topologías Packetexchange, IntegraTelecom.

4.3 DISCUSIÓN

La comparación se realiza respecto a costos obtenidos mediante las pruebas, es importante resaltar que no se pudieron probar todas las topologías dado que al replicar las pruebas algunas presentaban errores por tanto no se podía visualizar el resultado.

El tiempo de simulación fue distinto respecto a los resultados obtenidos en los estudios anteriores, según se mostró en [43] fueron horas y en las pruebas minutos y en [44] fueron segundos y en las pruebas días, el incremento del tiempo pudo haber influido el entorno utilizado en las pruebas, la actualización de las versiones del software tanto para la máquina virtual como para Python y sus librerías.

Respecto a las gráficas no se pudo generar de la misma forma porque al generarlas mostraban inconsistencias, sin embargo, se pudo realizar la comparación en costos y tiempo de simulación.

Respecto a los resultados de costo en las pruebas realizadas fue mucho más alto que el reportado en los estudios.

4.4 COMPARACIÓN GENERAL

Se realiza la comparación de los estudios [43], [44] obteniendo resultados distintos respecto a los estudios iniciales, a continuación, el promedio de los resultados de las pruebas con su

tiempo de ejecución, con la tabla 9 se puede concluir que el estudio [43] tiene mejor desempeño computacional y costos respecto al otro estudio.

Tabla 9. Comparación.

ESTUDIO	COSTO PROMEDIO	TIEMPO EJECUCIÓN
[43]	2700	202 minutos con 57 segundos (aproximadamente 4 horas)
[44]	9600	16 días y 18 horas (402 horas)

5. ADAPTACIÓN DEL MODELO MATEMÁTICO

Para el desarrollo del modelo matemático tuvimos como base lo descrito en el apartado 4 COMPARACIÓN DE RESULTADOS en este apartado se describe el modelo matemático utilizado en los estudios [43], [44] así mismo se explica cómo fue usado el modelo en el desarrollo de los estudios relacionados anteriormente, adicionalmente el modelo utilizado en los estudios fue la base para el desarrollo de esta investigación, dado que este modelo tiene los parámetros de costo y latencia lo que permitió incluir el parámetro de ***confiabilidad*** para una red SDN.

El parámetro confiabilidad es de suma importancia porque cuando la red está expuesta, los componentes de la red pueden fallar en su funcionalidad causando interrupciones en el rendimiento o incluso falla en el enlace afectando el rendimiento de la red. Estas fallas o interrupciones representan una amenaza en los distintos entornos por ejemplo empresariales, por lo que es importante tener en cuenta la confiabilidad de las redes para evitar o minimizar pérdidas posteriores ante los peligros a los que se enfrentan [46].

El parámetro de confiabilidad está dado por el tiempo que tarda la Red en seguir trabajando después de la falla de un enlace. En este caso, después de la falla del enlace tenemos un tiempo igual al de la latencia del establecimiento del flujo (FSL) 50ms o 70ms. Este tiempo es considerablemente menor pues está medido en milisegundos, comparado con lo propuesto por los otros métodos [43] y , que necesitan el tiempo de ejecución del algoritmo para que la Red vuelva a funcionar.

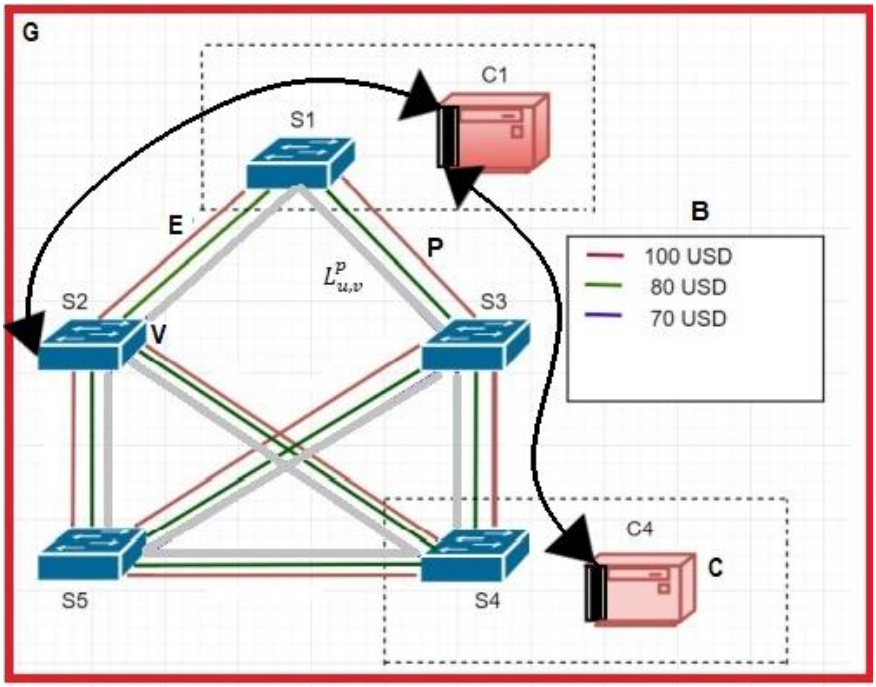


Figura 11. Simulación red SDN.

Adaptada de Tabú Search para el problema de ubicación del Controlador en Software Defined Networks.

Para la formulación del modelo matemático se tuvo en cuenta lo siguiente:

- Retardo máximo permitido en la red para la comunicación del switch al controlador (latencia de configuración del flujo) y para la comunicación entre switches.
- El costo de asignar una licencia para ejecutar un controlador en un switch.
- La capacidad del hardware de los switches para ejecutar las licencias de los controladores y manejar las solicitudes de configuración de flujo en términos de la demora de procesamiento para la configuración de flujo.
- La latencia de reenvío y el costo de usar un enlace que pertenece a un proveedor de servicios dado entre cada par de nodos.

El modelo se basa en el método de grafos y con base en lo anterior se formula el problema de la siguiente manera: se tiene un conjunto de nodos denominado V con un conjunto de enlaces llamado E *con* $E \subseteq V\ x\ V$ *y* esto formaría $G\ (V, E)$ por tanto, está es una red con el conjunto de nodos V y enlaces E . Por otro lado, tenemos P que es el conjunto de posibles

proveedores de servicios y C que es el conjunto de las posibles ubicaciones del controlador con $C \subseteq V$. También tenemos $L_{u,v}^{p}$, la latencia de reenvío (en milisegundos) desde el nodo $u \in V$ al nodo $v \in V$ para un proveedor de servicios P. Adicionalmente consideramos B, el costo (en $) de ejecutar una licencia de controlador, α_c el número de solicitudes promedio presentes en un controlador y σ_c el tiempo promedio (en milisegundos) para manejar un número determinado de solicitudes de configuración de flujo en un controlador. Sumado a lo anterior tenemos P_c que es el retraso de procesamiento (en milisegundos) para la configuración del flujo con $P_c = \alpha_c \sigma_c \forall c \in C$ y $\lambda_{u,v}^{p}$ el costo (en $) de utilizar una ruta de reenvío del proveedor de servicios P entre el nodo $u \in V$ al nodo $v \in V$.

Restricciones del modelo:

- Φ, la latencia máxima de configuración de flujo (en milisegundos).
- η, la latencia máxima entre controladores (en milisegundos).
- τ, **el número mínimo de controladores en la red para la confiabilidad.**

Variables:

- x_c una variable binaria tal que $x_c = 1$ sí y sólo si un controlador se implementa en una posible ubicación del controlador con $c \in C$.
- $k_{u,c,}$una variable binaria tal que $k_{u,c,} = 1$ sí y sólo si un nodo está asignado al controlador $c \in C$.
- $z_{u,c,}$ una variable binaria tal que $z_{u,c,} = 1$ sí y sólo si un nodo $u \in C$ y un nodo $v \in C$.
- $\delta_{u,}$una variable que representa la latencia entre un nodo $u \in V$ Y y su controlador $c \in C$.

Función de costo

El objetivo es minimizar los costos relacionados con el control de la red dado por:

- El costo de ejecutar licencias de controlador.

$$C_c(x) = \sum_{c \in C} B\, x_c$$

- El costo de enviar información de control entre un conmutador y su controlador asignado, incluyendo un segundo enlace para mayor confiabilidad.

$$C_c(k) = \sum_{u \in V} \sum_{c \in C} \sum_{p,q \in P} (\lambda^{\,p}_{u,v} + \lambda^{\,q}_{u,v})\, k_{u,c}$$

- El costo de enviar información de control entre controladores, incluido un segundo enlace para confiabilidad.

$$C_w(z) = \sum_{u \in C} \sum_{v \in C} \sum_{p \in P} (\lambda^{\,p}_{u,v} + \lambda^{\,q}_{u,v})\, z_{u,v}$$

Teniendo en cuenta lo mencionado anteriormente el modelo para el problema de ubicación del controlador (CPP) quedaría de la siguiente manera.

$$Minimize\ \big(C_c(x) + C_s(k) + C_x(z)\big)$$

6. EVALUACIÓN DEL MODELO MATEMÁTICO

Teniendo en cuenta el apartado anterior se realizó la evaluación del funcionamiento del modelo matemático por lo cual se decidió realizar la modificación del código en Python realizado en el estudio [43] que se encuentra disponible en el repositorio del proyecto SDNcon.

Se tuvo como base el diagrama de clases expuesto en el estudio [43] con el fin de incluir el parámetro de confiabilidad se decidió modificar la clase principal Network en la función linkActivacion(self), esta función lo que hace es activar aleatoriamente un enlace entre cada par de Switch, en nuestro caso se debe activar aleatoriamente dos enlaces con el fin de incluir el parámetro confiabilidad.

Respecto a las modificaciones del código se publica en el repositorio de GitHub (https://github.com/Magnadata-USTA/SDNCON) donde se encuentra el código fuente del estudio [43].

6.1 SIMULACIÓN MODELO MATEMÁTICO.

Para la simulación del modelo matemático se utilizó los mismos escenarios ya planteados en el trabajo, a continuación, las topologías utilizadas con los parámetros:

Tabla 10. Topologías utilizadas en la simulación del modelo.

Topología	Nodos	MaxFlowSetupLatency (MFSL)	MaxInterControllerLatency (MCL)
YourDataServices	23	50	70
xeex	24	50	100
Digex	31	50	70
Agis	25	50	70
ATTNorthAmerica	25	50	80
CRL	33	80	100
Psinet	24	50	80
ATMnet	21	50	100
Bics	33	50	100
NetworkUsa	35	50	70
Xspedius	34	50	70
Bellcanada	48	70	100
BBNplanet	27	50	100

EL percentageNode: 10%, movementType: 0, memorySize: 70, Iterations: 100.

6.2 RESULTADOS DE LA SIMULACIÓN.

Se realiza la simulación en las topologías utilizadas en las pruebas del algoritmo Tabú. Teniendo como resultado tiempos más altos de simulación y aumentos en los costos al activar un segundo enlace como se evidencia en la figura 12 y tabla 11.

Figura 12. Resultados de la simulación del modelo.

Tabla 11. Resultados de la simulación del modelo- tiempo.

Topología	Nodos	Tiempo de Simulación	Resultado Costo
YourDataServices	23	3 m 18, 30 s	2275
xeex	24	8m 20, 75 s	2880
Digex	31	6m 23, 64 s	3220
Agis	25	166 m 55, 22 s	3290
ATTNorthAmerica	25	4m 46, 25 s	2880
CRL	33	8m 45, 90 s	3265
Psinet	24	132 m 5,19 s	2855
ATMnet	21	8 m 20, 75 s	2880
Bics	33	12 m 42, 56 s	4250
NetworkUsa	35	15 m 37, 20 s	2825
Xspedius	34	445m 4, 221 s	3885
Bellcanada	48	585 m 53, 68 s	4370
BBNplanet	27	234 m 7, 48 s	3895

6.3 DISCUSIÓN

Las topologías Agis, Psinet, Xspedius, Bellcanada y BBNplanet tuvo tiempos de simulación muy altos en comparación con las otras topologías.

La topología que tuvo menor tiempo de simulación fue YourDataServices y la segunda ATTNorthAmerica.

Las topologías Xeex. Digex, CLR, ATMnet, Bics y NetworkUsa tuvo tiempos de simulación muy similares.

Respecto a costos la topología Bellcanada y Bics tuvo los costos más altos, el resto de las topologías tuvo costos similares.

6.4 COMPARACIÓN ALGORITMO TABÚ CON LA MODIFICACIÓN DEL MODELO.

Tabla 12. Comparación algoritmo Tabú más parámetro de confiabilidad.

Topologías	Costo Pruebas algoritmo Tabú	Costo incluyendo parámetro confiabilidad
YourDataServices	1930	**2275**
xeex	2030	**2880**
Digex	2355	**3220**
Agis	2440	**3290**

ATTNorthAmerica	2520	**2880**
CRL	2895	**3265**
Psinet	1980	**2855**
ATMnet	1655	**2880**
Bics	3730	**4250**
NetworkUsa	2795	**2825**
Xspedius	3670	**3885**
Bellcanada	4250	**4370**
BBNplanet	2945	**3895**

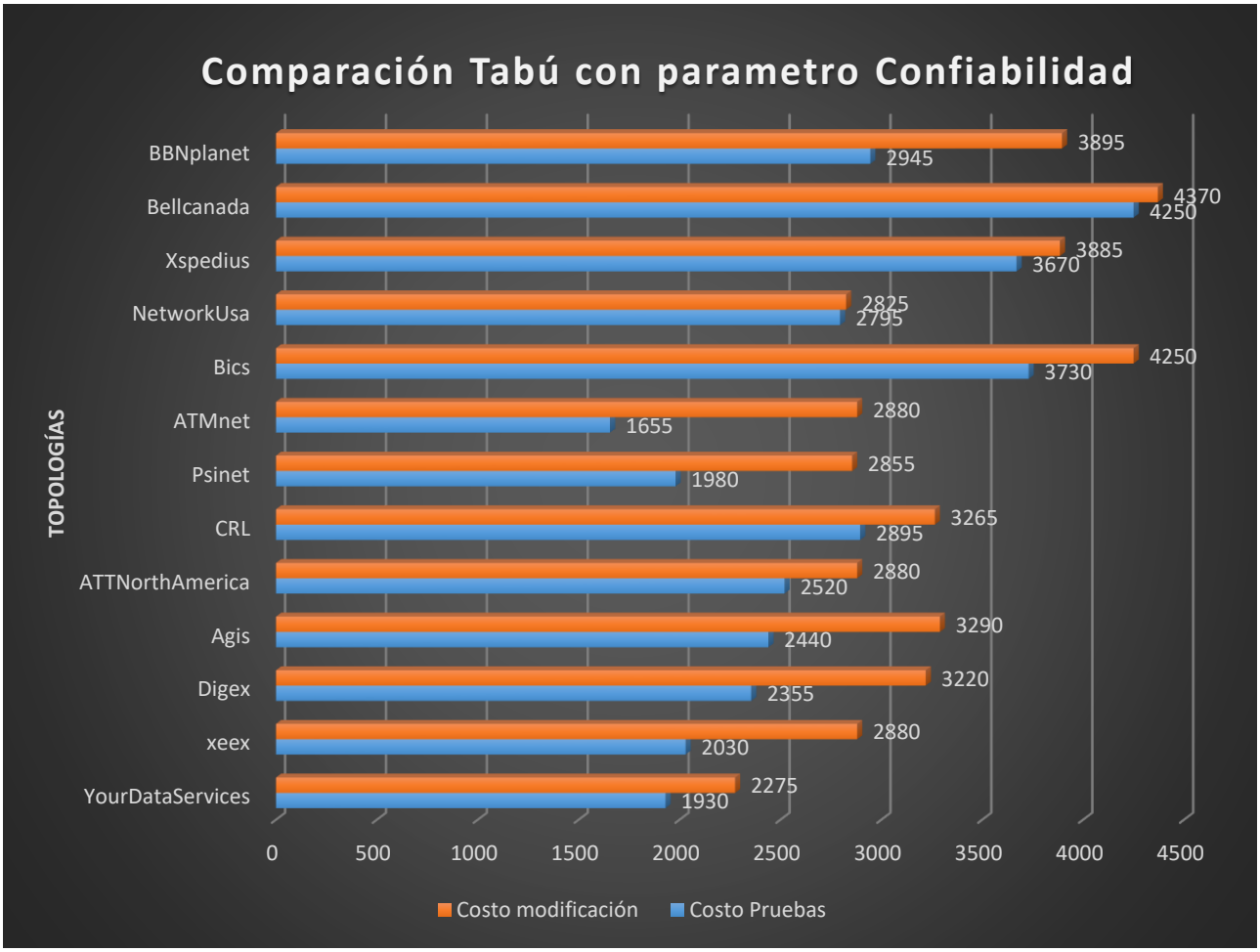


Figura 13. Comparación algoritmo Tabú más parámetro de confiabilidad.

De la figura 12 se puede concluir que incluyendo el parámetro confiabilidad los costos no son tan altos en comparación al algoritmo sin ese parámetro. Esto reafirma que el estudio [43] permite adaptación a los distintos escenarios de la red.

7. CONCLUSIONES

- Al realizar las pruebas se encontró que algunas topologías tenían errores, sin embargo, fue posible realizar las pruebas con las topologías restantes, para el caso del algoritmo genético se reemplazó algunas por otras con la misma cantidad de nodos. Es necesario realizar una revisión de los códigos con el fin de optimizarlo y organizarlo, adicional a ello sería importante dejar documentación sobre el funcionamiento.

- Respecto a la comparación del algoritmo Tabú se obtuvo costos similares al estudio inicial y en cuanto al tiempo de simulación las pruebas tuvieron menor tiempo, esto puede influir la capacidad computacional al momento de realizar las pruebas. Para el algoritmo genético los costos fueron muy altos, en algunos casos se triplico a los reportados en el estudio inicial y respecto al tiempo de simulación fue el doble al indicado en el estudio, por tanto, es necesario revisar el código del algoritmo genético con el fin de validar que los resultados reportados sean confiables.

- Al incluir el parámetro confiabilidad en el algoritmo Tabú, se evidencio aumento de los costos puesto que se está activando un segundo enlace en caso de falla y el tiempo de simulación no fue tan alto en comparación con el estudio original, entonces se puede deducir que el algoritmo Tabú se puede adaptar a los distintos cambios que se generen en la red.

- No se pudo incluir el parámetro confiabilidad en el algoritmo genético porque el funcionamiento del código no fue óptimo, es decir el tiempo de simulación se estaba tardando en algunos casos el doble al que se reportó en las pruebas, por tanto, no se ve viable incluir el parámetro sin antes realizar una revisión del funcionamiento del código.

- Como conclusión general el modelo algoritmo Tabú permitió agregar el parámetro de confiabilidad, a parte del costo y latencia, funcionando correctamente, teniendo

como resultados costos no tan elevados, respecto al tiempo de simulación se podrían mejorar si se utiliza una mejor capacidad computacional.

- A pesar de que no se encontró muchos estudios que utilicen algoritmos heurísticos que incluyan el parámetro confiabilidad, la diferencia de la investigación encontrada con el trabajo propuesto es que se activó un segundo enlace para continuar con el funcionamiento de la red lo que permite ser una red más resiliente y confiable.

CONCLUSIONES POR OBJETIVO

1. Se verificó los resultados obtenidos por los estudios anteriores del proyecto SDNCon, por medio de la replicación de las pruebas realizadas, teniendo como resultado diferencias en los costos y en el tiempo de simulación en ambos estudios.

2. Se realizó la comparación de los resultados obtenidos encontrando que el algoritmo Tabú tiene mejores resultados respecto al genético tanto en costos como en tiempo de simulación.

3. Se incluyó el parámetro de confiabilidad al modelo matemático desarrollado en el proyecto SDNCon, teniendo resultados satisfactorios tanto en costos como en tiempo de simulación.

4. Se evaluó el modelo matemático que incluye el parámetro de confiabilidad por medio de herramientas de simulación utilizadas en el proyecto SDNCon, mediante el algoritmo Tabú, así mismo se comparó los resultados de las pruebas con el algoritmo que incluye el parámetro de confiabilidad, los resultados de la comparación muestran que teniendo en cuenta el parámetro los costos de la red no son tan altos.

REFERENCIAS

[1] A. Singels, *Apropiación, generación y uso edificador del conocimiento*. 2018.

[2] CEPAL, “La Digitalización En América Latina Frente Al Covid-19,” *Cepal Caf Elac*, pp. 2–33, 2020.

[3] W. Zhijun, X. Qing, W. Jingjie, Y. Meng, and L. Liang, “Low-Rate DDoS Attack Detection Based on Factorization Machine in Software Defined Network,” *IEEE Access*, vol. 8, pp. 17404–17418, 2020, doi: 10.1109/ACCESS.2020.2967478.

[4] ITU-T, “Framework of software-defined networking,” 2014, [Online]. Available: https://www.itu.int/rec/T-REC-Y.3300-201406-I/en.

[5] D. Tatang, F. Quinkert, J. Frank, C. Röpke, and T. Holz, “SDN-GUARD: Protecting SDN controllers against SDN rootkits,” *2017 IEEE Conf. Netw. Funct. Virtualization Softw. Defin. Networks, NFV-SDN 2017*, vol. 2017-Janua, pp. 297–302, 2017, doi: 10.1109/NFV-SDN.2017.8169856.

[6] T. Zhang, A. Bianco, and P. Giaccone, “The role of inter-controller traffic in SDN controllers placement,” *2016 IEEE Conf. Netw. Funct. Virtualization Softw. Defin. Networks, NFV-SDN 2016*, pp. 87–92, 2017, doi: 10.1109/NFV-SDN.2016.7919481.

[7] M. Alsaeedi, M. M. Mohamad, and A. A. Al-Roubaiey, “Toward Adaptive and Scalable OpenFlow-SDN Flow Control: A Survey,” *IEEE Access*, vol. 7, pp. 107346–107379, 2019, doi: 10.1109/ACCESS.2019.2932422.

[8] G. Wang, Y. Zhao, J. Huang, and W. Wang, “The Controller Placement Problem in Software Defined Networking: A Survey,” *IEEE Netw.*, vol. 31, no. 5, pp. 21–27, 2017, doi: 10.1109/MNET.2017.1600182.

[9] L. Zhu, R. Chai, and Q. Chen, “Control plane delay minimization based SDN controller placement scheme,” *2017 9th Int. Conf. Wirel. Commun. Signal Process. WCSP 2017 - Proc.*, vol. 2017-Janua, pp. 1–6, 2017, doi: 10.1109/WCSP.2017.8171153.

[10] J. Lu, Z. Zhang, T. Hu, P. Yi, and J. Lan, “A survey of controller placement problem

in software-defined networking," *IEEE Access*, vol. 7, pp. 24290–24307, 2019, doi: 10.1109/ACCESS.2019.2893283.

[11] B. Isong, R. R. Samuel Molose, A. M. Abu-Mahfouz, and N. Dladlu, "Comprehensive review of SDN controller placement strategies," *IEEE Access*, vol. 8, pp. 170070–170092, 2020, doi: 10.1109/ACCESS.2020.3023974.

[12] M. Khorramizadeh and V. Ahmadi, "Capacity and load-aware software-defined network controller placement in heterogeneous envi tienenments," *Comput. Commun.*, vol. 129, no. June, pp. 226–247, 2018, doi: 10.1016/j.comcom.2018.07.037.

[13] A. A. Ateya *et al.*, "Chaotic salp swarm algorithm for SDN multi-controller networks," *Eng. Sci. Technol. an Int. J.*, vol. 22, no. 4, pp. 1001–1012, 2019, doi: 10.1016/j.jestch.2018.12.015.

[14] R. Gopakumar, A. M. Unni, and V. P. Dhipin, "An adaptive algorithm for searching in flow tables of openflow switches," *Proc. 2015 39th Natl. Syst. Conf. NSC 2015*, pp. 1–5, 2016, doi: 10.1109/NATSYS.2015.7489115.

[15] S. Tahmasebi, N. Rasouli, A. H. Kashefi, E. Rezabeyk, and H. R. Faragardi, "SYNCOP: An evolutionary multi-objective placement of SDN controllers for optimizing cost and network performance in WSNs," *Comput. Networks*, vol. 185, no. November 2020, p. 107727, 2021, doi: 10.1016/j.comnet.2020.107727.

[16] N. S. Mohamed Samir,Effat Samir,Mohamed Azab,Mohamed R. M. Rizk, "8 Towards Optimal Placement of Controllers in SDN- enabled Smart Grid.pdf." 2018.

[17] D. Santos and T. Gomes, "Controller Placement and Availability Link Upgrade Problem in SDN Networks," *Proc. 2019 11th Int. Work. Resilient Networks Des. Model. RNDM 2019*, pp. 1–8, 2019, doi: 10.1109/RNDM48015.2019.8949109.

[18] S. Kim, S. Kim, B. Lee, K. Kim, and H. Y. Youn, "Load Balancing for Distributed SDN with Harmony Search," *2019 16th IEEE Annu. Consum. Commun. Netw. Conf. CCNC 2019*, pp. 1–2, 2019, doi: 10.1109/CCNC.2019.8651682.

[19] Z. Fan, J. Yao, X. Yang, Z. Wang, and X. Wan, “A multi-controller placement strategy based on delay and reliability optimization in SDN,” *2019 28th Wirel. Opt. Commun. Conf. WOCC 2019 - Proc.*, no. Wocc, 2019, doi: 10.1109/WOCC.2019.8770551.

[20] S. Lange *et al.*, “Heuristic approaches to the controller placement problem in large scale SDN networks,” *IEEE Trans. Netw. Serv. Manag.*, vol. 12, no. 1, pp. 4–17, 2015, doi: 10.1109/TNSM.2015.2402432.

[21] A. Sallahi and M. St-Hilaire, “Expansion model for the controller placement problem in software defined networks,” *IEEE Commun. Lett.*, vol. 21, no. 2, pp. 274–277, 2017, doi: 10.1109/LCOMM.2016.2621746.

[22] L. Mamushiane, J. Mwangama, and A. A. Lysko, “Given a SDN Topology, How Many Controllers are Needed and Where Should They Go?,” *2018 IEEE Conf. Netw. Funct. Virtualization Softw. Defin. Networks, NFV-SDN 2018*, pp. 2–7, 2018, doi: 10.1109/NFV-SDN.2018.8725710.

[23] E. Husni and A. Bramantyo, “Design and Implementation of MPLS SDN Controller Application based on OpenDaylight,” *2018 Int. Symp. Networks, Comput. Commun. ISNCC 2018*, 2018, doi: 10.1109/ISNCC.2018.8530900.

[24] P. Talhar and A. P. Bhagat, “An Adaptive Approach for Controller Placement Problem in Software Defined Networks,” *Proc. 2018 3rd IEEE Int. Conf. Res. Intell. Comput. Eng. RICE 2018*, pp. 1–11, 2018, doi: 10.1109/RICE.2018.8509059.

[25] C. S. Khin, M. Zin Oo, and A. T. Kyaw, “Packet-in Messages Handling Scheme to Reduce Controller Bottlenecks in OpenFlow Networks,” *17th Int. Conf. Electr. Eng. Comput. Telecommun. Inf. Technol. ECTI-CON 2020*, pp. 502–505, 2020, doi: 10.1109/ECTI-CON49241.2020.9158127.

[26] J. Wiley, “New-generation Protocols,” 2020.

[27] M. A. Beiruti and Y. Ganjali, “Load migration protocol for SDN controllers,” *Proc. - Int. Conf. Netw. Protoc. ICNP*, vol. 2019-Octob, 2019, doi: 10.1109/ICNP.2019.8888102.

[28] S. Yang, L. Cui, Z. Chen, and W. Xiao, “An Efficient Approach to Robust SDN

Controller Placement for Security," *IEEE Trans. Netw. Serv. Manag.*, vol. 17, no. 3, pp. 1669–1682, 2020, doi: 10.1109/TNSM.2020.2994837.

[29] R. Zhou, Y. Lai, Z. Liu, and J. Liu, "Study on Authentication Protocol of SDN Trusted Domain," *Proc. - 2015 IEEE 12th Int. Symp. Auton. Decentralized Syst. ISADS 2015*, pp. 281–284, 2015, doi: 10.1109/ISADS.2015.29.

[30] Y. Jiménez, C. Cervelló-Pastor, and A. García, "Dynamic resource discovery protocol for software defined networks," *IEEE Commun. Lett.*, vol. 19, no. 5, pp. 743–746, 2015, doi: 10.1109/LCOMM.2015.2403322.

[31] A. T. S. Abu-Jassar, "Mathematical tools for SDN formalisation and verification," *2015 2nd Int. Sci. Conf. Probl. Infocommunications Sci. Technol. PIC S T 2015 - Conf. Proc.*, pp. 35–38, 2015, doi: 10.1109/INFOCOMMST.2015.7357262.

[32] H. Zhang and J. Yan, "Performance of SDN Routing in Comparison with Legacy Routing Protocols," *Proc. - 2015 Int. Conf. Cyber-Enabled Distrib. Comput. Knowl. Discov. CyberC 2015*, pp. 491–494, 2015, doi: 10.1109/CyberC.2015.30.

[33] R. Chai, X. Yang, C. Du, and Q. Chen, "Network cost optimization-based capacitated controller deployment for SDN," *Comput. Networks*, vol. 197, p. 108326, 2021, doi: 10.1016/j.comnet.2021.108326.

[34] W. Feng, C. Liu, B. Cheng, and J. Chen, "Secure and cost-effective controller deployment in multi-domain SDN with BAGUETTE," *J. Netw. Comput. Appl.*, vol. 178, no. December 2020, p. 102969, 2021, doi: 10.1016/j.jnca.2020.102969.

[35] Z. Guo and W. Feng, "RetroFlow : Maintaining Control Resiliency and Flow Programmability for Software-Defined WANs."

[36] H. Song, S. Guo, P. Li, and G. Liu, "FCNR: Fast and Consistent Network Reconfiguration with low latency for SDN," *Comput. Networks*, vol. 193, no. March, 2021, doi: 10.1016/j.comnet.2021.108113.

[37] Y. Fan, L. Wang, and X. Yuan, "Controller placements for latency minimization of both primary and backup paths in SDNs," *Comput. Commun.*, vol. 163, no. February, pp.

35–50, 2020, doi: 10.1016/j.comcom.2020.09.001.

[38] L. M. Elguea and F. Martinez-Rios, “A New method to optimize BGP routes using SDN and reducing latency,” *Procedia Comput. Sci.*, vol. 135, pp. 163–169, 2018, doi: 10.1016/j.procs.2018.08.162.

[39] E. Kawaguchi and N. Shinomiya, “Network Partitioning Algorithm for Minimizing Communication Latency in SDN,” *IEEE Reg. 10 Annu. Int. Conf. Proceedings/TENCON*, vol. 2018-Octob, no. October, pp. 1529–1532, 2019, doi: 10.1109/TENCON.2018.8650370.

[40] L. Han, Z. Li, W. Liu, K. Dai, and W. Qu, “Minimum control latency of SDN controller placement,” *Proc. - 15th IEEE Int. Conf. Trust. Secur. Priv. Comput. Commun. 10th IEEE Int. Conf. Big Data Sci. Eng. 14th IEEE Int. Symp. Parallel Distrib. Proce*, pp. 2175–2180, 2016, doi: 10.1109/TrustCom.2016.0334.

[41] E. Calle, D. Martínez, M. Mycek, and M. Pióro, “Resilient backup controller placement in distributed SDN under critical targeted attacks,” *Int. J. Crit. Infrastruct. Prot.*, vol. 33, p. 100422, 2021, doi: 10.1016/j.ijcip.2021.100422.

[42] B. California, “Centro de Investigación Científica y de Educación Superior de Ensenada , Baja California Maestría en Ciencias en Electrónica y Telecomunicaciones con orientación en Instrumentación y Control Diseño de un microaerogenerador de eje vertical,” 2018.

[43] R. Isaac, “Tabu search para el problema de ubicación del controlador en software defined networks,” 2019, [Online]. Available: http://hdl.handle.net/11634/16639.

[44] Y. Moreno Gómez, “Algoritmo genético para el problema de la ubicación del controlador en una red definida por software (RDS),” 2020, [Online]. Available: http://hdl.handle.net/11634/22063.

[45] R. I. Abuabara, F. Díaz-Sánchez, J. Arevalo Herrera, and I. Amigo, “Cost-Effective Tabu Search Algorithm for Solving the Controller Placement Problem in SDN,” pp. 109–130, 2020, doi: 10.4018/978-1-7998-1839-7.ch005.

[46] N. L. Dehghani, S. Zamanian, and A. Shafieezadeh, “Adaptive network reliability analysis: Methodology and applications to power grid,” *Reliab. Eng. Syst. Saf.*, vol. 216, no. April, p. 107973, 2021, doi: 10.1016/j.ress.2021.107973.

LISTA DE FIGURAS

LISTA DE TABLAS