



INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE INGENIERÍA MECÁNICA Y ELÉCTRICA,
UNIDAD PROFESIONAL ADOLFO LÓPEZ MATEOS, ZACATENCO
DEPARTAMENTO ACADÉMICO DE INGENIERÍA EN COMUNICACIONES Y
ELECTRÓNICA

SISTEMA INTELIGENTE DE GESTIÓN INTELIGENTE EN
REDES DEFINIDAS POR SOFTWARE MEDIANTE ANÁLISIS DE
MÉTRICAS EN TIEMPO REAL POR MEDIO DE MACHINE
LEARNING.

T E S I S

QUE PARA OBTENER EL TÍTULO DE:

INGENIERO EN COMUNICACIONES Y ELECTRÓNICA

PRESENTA:

HERNANDEZ LUGO CHRISTIAN ALEJANDRO

ASESORES DE TESIS

M. EN S. I. FERNANDO NOYA CHÁVEZ
DRA. MARÍA ELENA ACEVEDO MOSQUEDA

Ciudad de México, a XX de XX 2025



A mis padres

Contenido

Lista de Figuras	iii
Lista de Tablas	v
1 Introducción	1
1.1 Planteamiento del problema	1
1.2 Justificación	3
1.3 Objetivos	5
1.3.1 General	5
1.3.2 Particulares	5
2 Estado del Arte	7
2.1 Contexto histórico	7
2.1.1 Inicios de SDN	7
2.1.2 Influencia de la IA en las SDN	8
2.2 Trabajos relacionados	8
2.2.1 Análisis del Tráfico de Red Utilizando Técnicas de Machine Learning	9
2.2.2 Gestión y Control de Calidad de Servicio de Redes SDN en Tiempo Real	10
2.2.3 Avanzando hacia una red auto-adaptativa: simulación de redes definidas por software (SDN) mediante el simulador GNS3	10
2.2.4 Escalado Automático de Recursos de Red mediante Aprendizaje Automático para Mejorar la QoS y Reducir Costos	11
2.3 Discusión	11
3 Marco Teórico	13
3.1 Redes Definidas por Software SDN	13
3.1.1 Arquitectura de una red SDN	13
3.1.2 Componentes Básicos de la Arquitectura SDN	14

CONTENIDO

3.1.3	Controladores SDN	15
3.1.4	Características principales de una SDN	15
3.2	Aprendizaje Automático	16
3.2.1	Elementos del aprendizaje automático	16
3.2.2	Tipos de aprendizaje automático	17
3.2.3	Algoritmos de aprendizaje automático	17
3.2.4	Simuladores de Red	22
3.2.5	Calidad de Servicio(QOS)	24
4	Metodología y Desarrollo	27
4.1	Modelo principal del sistema	27
4.2	Métricas a evaluar	29
4.3	Captura y Análisis de Métricas de Red en Switches SDN	31
4.4	Ejecución de algoritmo de Machine Learning	33
4.4.1	Generación del Data-Set	33
4.4.2	Entrenamiento del modelo	35
4.5	Commutación de rutas	40
4.5.1	Implementación en el sistema	41
5	Pruebas y Resultados	45
5.1	Funcionamiento	45
5.1.1	Generación de Tráfico	45
5.2	Resultados del experimento	52
6	Conclusiones	57
6.1	Conclusiones Finales	57
6.2	Trabajo Futuro	58
A	Creación de hosts por medio de contenedores de Docker	59
A.0.1	Construcción de la imagen Docker	59
A.0.2	Carga de la imagen como appliance en GNS3	60
A.0.3	Despliegue y conexión de contenedores	61
B	Instalación del controlador Ryu en Ubuntu	63
B.0.1	Preparación del entorno	63
B.0.2	Instalación de Ryu	64
B.0.3	Integración de switches Open vSwitch (OVS) en GNS3	64
	Referencias	67

Lista de Figuras

1.1	Red Backbone del IPN	2
1.2	Tráfico de datos creados,capturados y consumidos a nivel global(1) . .	2
3.1	Comparación de arquitectura tradicional vs arquitectura SDN(2) . .	14
3.2	Relación entre las capas de la Arquitectura SDN(2)	15
3.3	Ejemplo de instancias de un set de datos(3)	21
4.1	Modelo general del sistema	28
4.2	Arquitectura de la Aplicación	29
4.3	Captura de Métricas	32
4.4	Topología Mancuerna	34
4.5	Topología Árbol	34
4.6	Distribución de clases antes y después de SMOTE TOMEK	36
4.7	Métricas de rendimiento Random Forest	37
4.8	Métricas de rendimiento KNN	38
4.9	Métricas de rendimiento Regresión Logística	38
4.10	Métricas de rendimiento Random Forest	39
4.11	Métricas de rendimiento KNN	39
4.12	Métricas de rendimiento Regresión Logística	40
4.13	Diagrama de subprocesso conmutación de rutas	41
5.1	Generación de tráfico	45
5.2	Host Servidor Iperf	46
5.3	Ejecución de los módulos de la aplicación tanto de monitoreo como de gestión.	47
5.4	Flujos por ruta principal	47
5.5	Flujos por ruta alterna	48
5.6	Consulta de métricas de red	49
5.7	Gráficas en tiempo real	49
5.8	Proceso de conmutación	50
5.9	Topología de pruebas	51

LISTA DE FIGURAS

5.10 Topología Red BackBone IPN	52
5.11 Captura de métricas de red en distintos niveles de ancho de banda (I)	53
5.11 Captura de métricas de red en distintos niveles de ancho de banda (II)	54
5.12 Conmutación de rutas al detectarse congestión en la red	55
A.1 Dockerfile utilizado para la creación de la imagen base	60
B.1 Proceso paso a paso para la instalación del appliance Open vSwitch en GNS3	65
B.2 Switch Open vSwitch con el controlador configurado	66

Lista de Tablas

2.1 Comparación de trabajos relacionados	12
4.1 Dispositivos utilizados en la topología y su función	31

LISTA DE TABLAS

Capítulo 1

Introducción

1.1 Planteamiento del problema

El Instituto Politécnico Nacional (IPN) es una de las instituciones educativas más importantes de México y una de las mas grandes. Debido a esto, el funcionamiento de la comunicación interna entre las diferentes escuelas que lo componen es de vital importancia. La estabilidad y capacidad de la infraestructura tecnológica especialmente de su red backbone se convierte en una prioridad. La red backbone del IPN tiene el papel de ser el núcleo central para la comunicación institucional, soportando múltiples servicios, tales como plataformas educativas, bases de datos académicas y administrativas, comunicación por voz, videoconferencias, streaming, correo electrónico, así como el acceso a internet para decenas de usuarios. Actualmente, esta infraestructura la cual puede verse en la figura 1.1 ,se encuentra organizada en una topología delta, compuesta por tres nodos principales: Zacatenco, Santo Tomás y UPIICSA. Sin embargo, la creciente demanda de servicios ha llevado a un incremento significativo en el tráfico de red, provocando problemas frecuentes como es el caso de la congestión. Esta problemática trae consigo distintos obstáculos que perjudican la calidad del servicio, como la degradación del rendimiento, pérdida de paquetes, aumento de la latencia, e incluso caídas ocasionales de los enlaces.

1. INTRODUCCIÓN

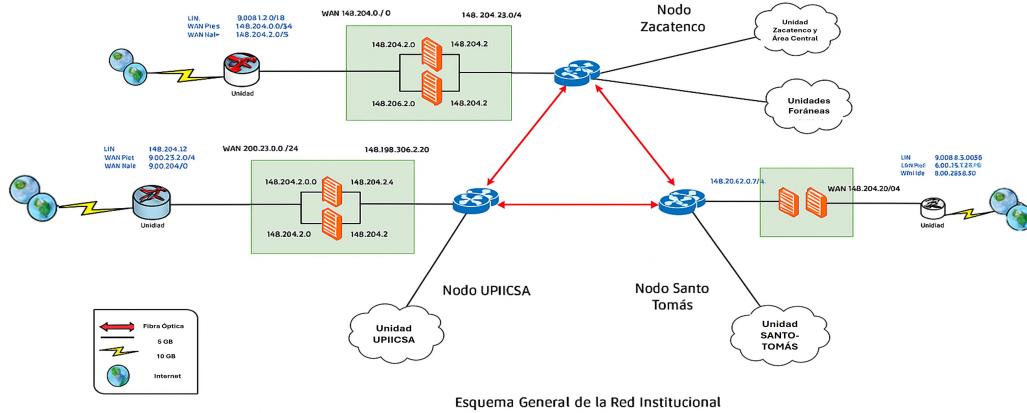


Figura 1.1: Red Backbone del IPN

A pesar de que este trabajo esté centrado a la red backbone del IPN, las problemáticas ocurren en diferentes organizaciones ya que el aumento del tráfico de datos es bastante notorio. Según un reporte, la cantidad total de datos generados durante el año 2024 a nivel mundial fue de 149 zettabytes, y se proyecta que aumente a 394 zettabytes para 2028(1). Esto evidencia la creciente inquietud y necesidad de prestar especial atención en la administración de redes centrándose en su optimización y dar una garantía de la calidad de servicio (QoS).

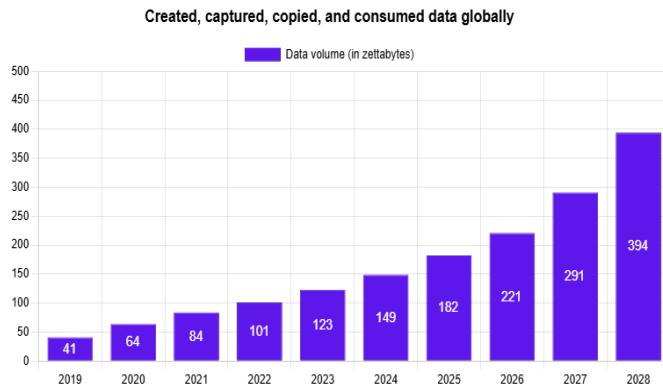


Figura 1.2: Tráfico de datos creados,capturados y consumidos a nivel global(1)

Aunque el IPN cuenta con sistemas tradicionales de monitoreo de red, estos sistemas suelen presentar limitaciones significativas en cuanto a temas de escalabilidad,

automatización y capacidad para anticipar proactivamente eventos ocasionados por la congestión. Es por este motivo que surge la necesidad de implementar soluciones mas robustas y avanzadas las cuales permitan no solo el monitoreo de la red, sino también gestionar de manera autónoma y eficiente situaciones de congestión, es decir tener una nueva generación de aplicaciones de gestión de red que nos puedan brindar estas características.

1.2 Justificación

Como se menciono anteriormente la congestión es una problemática que afecta directamente al rendimiento de las redes. Dado el constante crecimiento del tráfico de datos a distintas escalas de red, se propone una solución basada en el desarrollo de un sistema de gestión y monitoreo de red en tiempo real con la arquitectura de una SDN. Con esto en mente, desde sus comienzos, las redes definidas por software (SDN) han probado ser una opción viable, eficiente y segura para la gestión y monitoreo del tráfico de red. Esta nueva tecnología da la posibilidad a los administradores de red programar de manera dinámica los dispositivos de red, facilitando y optimizando la capacidad de respuesta para la mitigación de problemáticas en las redes de datos. En años recientes han surgido nuevos mercados derivado de esta tecnología. El mercado de infraestructuras SD-WAN ha experimentado un crecimiento acelerado. De acuerdo con datos de IDC, en 2022, el mercado de infraestructuras SD-WAN creció un 25.0%, y se espera que crezca alcanzando un valor de 7.5 mil millones de dólares. Bajo estos datos vemos como las SDN reflejan una presencia significativa en la gestión del tráfico en redes WAN y LAN. (4) Es en este punto donde una herramienta basada en la combinación de nuevas tecnologías surgen como una alternativa prometedora. La combinación de técnicas de inteligencia artificial con SDN, representa una evolución para los métodos tradicionales de administración de redes, dándonos mejoras sustanciales como las escalabilidad y flexibilidad respecto a métodos tradicionales de administración de redes colocándose como un candidato para ser una nueva generación de herramientas para la gestión de redes. Esta integración nos permite afrontar estas problemáticas y mitigarlas para maximizar el rendimiento de las redes de datos aparte de mejorar la experiencia de usuario. Como primer punto se tienen las Redes definidas por Software o "SDN" las cuales en años recientes se han vuelto un referente para la administración del tráfico en redes de datos. En comparación con los métodos tradicionales, las SDN aparte de la versatilidad y flexibilidad que ofrecen, estas permiten configuraciones adaptativas en tiempo real diseñadas específicamente para las características del alguna red. Estas características son fundamentales para el desarrollo de nuestro proyecto ya que da una velocidad de reacción mucho mayor permitiéndonos aplicar acciones correctivas para evitar problemáticas como es el caso de la congestión. Implementando el uso

1. INTRODUCCIÓN

de modelos entrenados de inteligencia artificial nos brinda la oportunidad de añadir una herramienta de clasificación en el tráfico de la red para que basándonos en métricas monitoreadas, estos modelos requieren un conjunto de datos históricos etiquetados previamente para poder decir que se considera tráfico real o si se tiene congestión. Bajo estos datos, el modelo tiene la capacidad de identificar patrones y características para poder tener una clasificación adaptativa y no estática para un posible congestión de alguna parte de la red.

Específicamente para este trabajo se planteo hacer la comparativa de uso de tres de algoritmos de aprendizaje supervisado para poder observar el rendimiento y precisión de cada uno de ellos. Con esto se puede tomar una decisión de cual sería implementado en el sistema final. Los algoritmos que se consideraron son: Random Forest; Algoritmo basado en la combinación de múltiples árboles de decisión para realizar predicciones precisas y robustas. En cuanto a la clasificación, este algoritmo es efectivo cuando se trabaja con tráfico en redes debido a su capacidad para manejar grandes volúmenes de datos y múltiples características relevantes. K-Nearest Neighbors (KNN); Algoritmo basado en la similitud entre datos, donde una nueva instancia se clasifica en función de la mayoría de sus vecinos más cercanos en el espacio de características. En el análisis de tráfico en redes, KNN es útil debido a su simplicidad y capacidad de adaptarse a patrones de congestión en tiempo real. Regresión Logística: Algoritmo de clasificación que da como resultado probabilidad de que un dato pertenezca a una clase específica, utilizando una función logística Sigmoide para transformar una combinación lineal de las características de entrada. En nuestro contexto nos puede ser útil por su capacidad para identificar patrones binarios, como la presencia o ausencia de congestión. El propósito de usar la combinación sinérgica de estas dos tecnologías esta englobada en los siguientes 3 puntos:

- Aplicar sistemas de gestión modernos para la gestión del flujo de datos en redes.
- Incorporar modelos de inteligencia artificial para poder anticiparse ante problemas y aplicar acciones preventivas antes de que impacten al rendimiento general de una red.
- Dar un sistema flexible y escalable a cualesquieras condiciones de red. Esto nos permite dar una respuesta a la creciente presión sobre las capacidades de gestión de las infraestructuras de red para que de igual manera la necesidad del surgimiento de nuevas soluciones y sistemas que puedan escalar en conjunto con el aumento del tráfico de datos en redes.

Para el desarrollo del proyecto, se opto por la utilización de la plataforma GNS3 como entorno para la simulación de las topologías con las que se trabajo. GNS3 es

una herramienta utilizada debido a la capacidad que cuenta para poder trabajar sin la necesidad de hardware físico aparte de la integración de dispositivos de red virtualizados con software de terceros, lo que permite la implementación de gestión de redes por medio de redes definidas por software y el uso de hipervisores como VMware. En general al tener una topología virtualizada se cuenta con una plataforma robusta y escalable para la simulación de redes, teniendo la ventaja de ser replicable con diferentes elementos permitiendo evaluar el impacto del sistema en la gestión del tráfico de la red. Específicamente para el hecho del backbone del IPN se tendrá la posibilidad de contar con un sistema capaz de detectar anticipadamente situaciones de congestión, redistribuir eficientemente el tráfico según su tipo (voz, datos, streaming, usuarios generales) y utilizar rutas alternas menos saturadas, mejorando así la calidad y eficiencia operativa del backbone institucional.

1.3 Objetivos

1.3.1 General

Desarrollar un sistema de predicción y gestión dinámica del tráfico de red para mitigar problemas de congestión mediante la implementación y combinación de Redes Definidas por Software y modelos de IA.

1.3.2 Particulares

- Analizar antecedentes y estudios enfocados en SDN en conjunto algoritmos de aprendizaje supervisados , en los cuales se trate la calidad de servicio.
- Examinar los indicadores fundamentales de la calidad de servicio (QOS) en una red, tales como el uso del ancho de banda, la pérdida de paquetes, el retraso y el jitter, para detectar patrones vinculados con la congestión.
- Tomar acciones proactivas en una red basados en la detección de congestión por medio de algoritmos de aprendizaje supervisado.
- Analizar el rendimiento de la red simulando eventos de congestión y de tráfico normal.

1. INTRODUCCIÓN

Capítulo 2

Estado del Arte

2.1 Contexto histórico

2.1.1 Inicios de SDN

Previo a entender el proyecto en si, es importante tener una perspectiva y noción histórica de la evolución de los pilares en los que se basa. Pero antes se puede establecer un concepto básico de las SDN las cuales son un enfoque de redes que utiliza controladores de software que pueden ser impulsados por interfaces de programación de aplicaciones (API) para comunicarse con la infraestructura de hardware para dirigir el tráfico de red. (5) Se puede centrar su evolución en tres etapas específicas la primera que surgió a partir de 1990, donde las primeras versiones traían consigo funciones programables en la red mediante una interfaz de programación siendo esta una API(Application Programming Interface). Para la segunda etapa y con el paso del tiempo en los años de 2001 a 2007, se introdujo la idea de separar el plano de control que en esencia se encarga de gestionar, mantener y modificar el estado de la red el cual tiene la función del plano de datos el cual cumple con la función de realizar lo establecido en el plano de control siendo una mejora importante para las SDN. Esto aplicado en los planos de control y datos en redes telefónicas, evolucionando con hitos como el protocolo ForCES del IETF (2004) y el proyecto Ethane de Stanford (2008), que derivó en OpenFlow.(6) A partir del periodo de 2007 a 2010 y ultima etapa, se introducen los API OpenFlow la cual fue una tecnología la cual se vuelve un estándar de comunicación entre los elementos de una SDN. Esta estandarización facilitó a los administradores la programación y administración de la red de forma más flexible y adaptable, acorde con las necesidades de aplicaciones actuales que demandan un gran uso de ancho de banda y una naturaleza dinámica. Superando las limitaciones de las redes estáticas tradicionales. (7, 8) Las SDN surgen como alternativa a las arquitecturas de redes tradicionales las cuales presentaban rigidez y complejidad en su administración.

2. ESTADO DEL ARTE

Hablando de un caso en particular se veía que los data center no tenían la capacidad de responder patrones de tráfico impredecibles, por lo que se encontró con dos posibles soluciones o se escalaba a una red más grande y más cara o se tomaba la solución de adaptarse a una red dinámica. Con la llegada de las SDN trajeron consigo mayor velocidad, infraestructura ágil, mayor calidad y reducción de costos.

2.1.2 Influencia de la IA en las SDN

Las SDN actualmente están surgiendo como la nueva generación de alternativas a los métodos para la gestión de redes tradicionales y poco a poco se han ido ajustando nuevas herramientas tecnológicas emergentes creando una sinergía entre tecnologías, tal es el caso de la inteligencia artificial.

El campo de la inteligencia artificial en los últimos años ha experimentado un desarrollo a una velocidad sin precedentes permitiendo desarrollar modelos complejos capaces de realizar tareas complejas como el caso de la clasificación de manera completamente autónoma. Para aplicaciones del trabajo presente y el uso de esta área en las redes, la inteligencia artificial se ha usado para diferentes aspectos desde el mejoramiento de la seguridad de las redes durante la transmisión de datos y evitar vulnerabilidades, optimización del rendimiento para poder dar una mejor experiencia de usuario, reducción de costos, facilidad de configuración entre otras cosas. Esto ha permitido a las redes desarrollar una evolución constante la cual ha permitido a los sistemas de red adaptarse y tener capacidad de respuesta a la variación de factores presentes que puedan generar obstáculos para el rendimiento de la red. Esta emergió como respuesta a la complejidad de gestionar redes dinámicas. Sistemas como SD-WAN y plataformas de Cisco/Arista Networks emplean estos modelos para optimizar rutas, reducir latencia (35%) y mitigar amenazas mediante detección de problemáticas presentes en la red.(9, 10) Ambas tecnologías se presentan como redes autoajustables, basadas en datos históricos y predicciones, las reconfiguran topologías sin intervención humana adaptando sus políticas y comportamiento de la red, marcando un punto de inflexión en la gestión de infraestructuras críticas ante el crecimiento acelerado del tráfico de datos.

2.2 Trabajos relacionados

Se llevó a cabo una búsqueda en las siguientes repositorios de universidades y bases de datos encontrando una variedad de trabajos similares. Estos repositorios siendo por ejemplo la Universidad de Valladolid, Universitat Politècnica de Catalunya, Universidad de los Andes y la de la Universidad de Cornell (Arxiv). Dentro de ellos se centró en identificar estudios relevantes en el ámbito de la optimización en redes mediante inteligencia artificial (IA) y redes definidas por software (SDN). Se

2.2 Trabajos relacionados

centro la búsqueda en trabajos relacionados con la predicción de congestión en redes de alta demanda. Se priorizaron artículos que incluyeran mejoras en la precisión del análisis del tráfico de red y el uso de técnicas de inteligencia artificial, como modelos avanzados de aprendizaje automático. También se tuvo en cuenta estudios que exploraran el uso de algoritmos dinámicos y tecnologías de redes definidas por software(SDN) para gestionar de manera más eficiente el tráfico en diferentes escenarios y situaciones.

Se hizo uso de las siguientes palabras clave para la localización de trabajos similares:

- *SDN*
- *QoS*
- *Inteligencia Artificial/Machine Learning*
- *Mecanismos de Balanceo de Carga*

Con el uso de estas palabras clave se pueden sintetizar 4 trabajos relevantes para su estudio:

- *Desarrollo de aplicaciones para redes definidas por software mediante el controlador ONOS por Rubén Blanco Pérez. (11)*
- *Pathfinder: Aplicación para la gestión de QoS en redes SDN por Daniel Guija Alcaraz. (12)*
- Por lo que de una manera mas profunda se explican a continuación cual fue el enfoque que usaron y que es el objetivo general de cada una de las propuestas para que con esto podamos tener los puntos mas importantes de cada propuesta y entender de manera completa y profunda propuestas similares con la que se esta trabajando.

2.2.1 Análisis del Tráfico de Red Utilizando Técnicas de Machine Learning

Dentro de este trabajo, el autor Behdad Bibak establece la propuesta la cual esta basada en la utilización de redes neuronales profundas las cuales permiten aprender por experiencia y bajo estas adaptar decisiones basados en patrones presentes en los datos. El enfoque que se sigue dentro de el análisis de los datos históricos del tráfico de una red con el cual es entrenada la red neuronal dándole la capacidad de poderse anticipar haciendo una predicción

2. ESTADO DEL ARTE

ante posibles eventos de congestión y poder tener como poder reaccionar estableciendo acciones de corrección como el redireccionamiento del tráfico a rutas alternativas para poder mitigar estos eventos que comprometen el rendimiento de la red. El set de datos esta compuesto por métricas de la red como patrones de flujo, tasas de envío de paquetes y la utilización del ancho de banda. Como complemento se hizo uso de un algoritmo de control de Mijumbi, este tiene la tarea específica de ejecutar decisiones de re configuración de rutas en función de la predicción generada por el modelo de red de neuronal profunda. Con esta metodología se puede observar mejoras significativas de la red desde un buen rendimiento del tráfico que pasa por toda la red hasta evitar la pérdida de paquetes probando el desempeño que se puede lograr en la combinación de inteligencia artificial con SDN.

2.2.2 Gestión y Control de Calidad de Servicio de Redes SDN en Tiempo Real

Este trabajo realizado por el autor Daniel Guija Alcaraz establece la propuesta de el desarrollo de un aplicación implementando SDN con el nombre de "pathfinder" el cual hace la integración de un algoritmo para la gestión de la calidad de servicio (QoS) en tiempo real y de manera dinámica. Este enfoque busca el que se elijan las rutas mas óptimas considerando las implicaciones del tráfico de red así como las restricciones de las mismas. Las propuesta de solución que se presenta esta en establecer una asignación dinámica de recursos haciendo uso de la métricas de QoS, como pérdida de paquete o como la latencia mínima. De igual manera, se utilizan datos en tiempo real para encontrar estas rutas mas optima y que pueda redistribuir el flujo de datos cuando se detectan problemáticas de congestión. Este sistema probo mejorar la eficiencia de la red en entornos simulados.

2.2.3 Avanzando hacia una red auto-adaptativa: simulación de redes definidas por software (SDN) mediante el simulador GNS3

Con respecto al trabajo del autor Rubén Blanco Pérez, se propone el desarrollo de 8 aplicaciones para SDN mediante el controlador ONOS estos con el objetivo de poder ser un precedente y se pueda avanzar a redes auto-adaptativas. Estas aplicaciones se centran en la gestión de la red desde el control de tráfico, hasta la detección de servidores DHCP falsos y el balanceo

de carga en enlaces redundantes. Cada solución fue diseñada, implementada y probada que se cumpla la mejora de problemas comunes en las infraestructuras de redes tradicionales. El enfoque metodológico se basa en la combinación de la simulación de redes en el software de GNS3 junto con una imagen de switch que acepte los protocolos de comunicación necesario para trabajar con SDN como es el caso de Open vSwitch. Los resultados de las aplicaciones reflejan como la implementación del enfoque de las SDN mejoran significativamente la flexibilidad, eficiencia y la seguridad en comparativa con la gestión tradicional de redes.

2.2.4 Escalado Automático de Recursos de Red mediante Aprendizaje Automático para Mejorar la QoS y Reducir Costos

[24] Para este trabajo de los autores Sabidur Rahman, Tanjila Ahmed, Minh Huynh, Massimo Tornatore y Biswanath Mukherjee se establece la propuesta de un enfoque para el escalado automático de redes virtualizadas, con el objetivo de mejorar la calidad del servicio (QoS) y permitir reducir costos operativos en redes. La metodología que se siguió en este proyecto inicio con la utilización de datos históricos de tráficos de red usando 27 características relacionadas con el tráfico, como patrones temporales y carga de datos y clases establecidas previamente para entrenar un clasificador de inteligencia artificial. Este modelo tiene la tarea de genera decisiones preventivas, permitiendo anticipar cambios en el tráfico pudiendo garantizar requerimientos del QoS. Dentro de este estudio se uso Random Forest mostrando un precisión del 95% teniendo un gran rendimiento general sin la presencia de un gran número de falsos positivos. Lo que se concluye de este estudio es el visualizar un enfoque para la resolución de escalado automático de redes y como este puede ayudar a la optimización de la red a la reducción de costos de empresas implementando el uso de SD-WAN.

2.3 Discusión

Como se puede observar el Tabla 2.1, se tiene presente una síntesis que establece una comparación, el cual sirve como una base referencial para el desarrollo del presente proyecto. Cada uno de los trabajos seleccionados aborda diferentes componentes desde las SDN,QOS, uso de distintos algoritmos de inteligencia artificial, mecanismo de balanceo de carga o

2. ESTADO DEL ARTE

Tabla 2.1: Comparación de trabajos relacionados

Trabajos	SDN	QoS	IA/ML	Gestión de Flujos	Gráficas Métricas
Rubén Blanco Pérez (11)	✓	✓		✓	
Daniel Guija Alcaraz (12)	✓	✓		✓	
Sabidur Rahman et al. (13)	✓	✓	✓		
Behdad Bibak (14)	✓	✓	✓	✓	
Propuesta	✓	✓	✓	✓	✓

ruteo dinámico para mejorar el rendimiento de la red, viendo como todos estos elementos están integrados en la propuesta establecida. Aunque estos trabajos tienen objetivos similares el que mas se acerca es el de Behdad Bibak pero nuestro elemento diferenciador es que no se incluyen herramientas de monitoreo en tiempo real aparte de que se difiere en la comparación de distintos algoritmos de inteligencia artificial para comparar su funcionamiento y rendimiento con la red.

Capítulo 3

Marco Teórico

3.1 Redes Definidas por Software SDN

Red Definida por Software (SDN) "Software-defined networking" es un método de administración de red que admite la configuración de red programable dinámica.[9] Gracias al uso de software las SDN tienen la posibilidad de proveer entornos de aplicaciones como código y el poder minimizar el tiempo de manipulación que se debe emplear para la administración de la red.

3.1.1 Arquitectura de una red SDN

Para explicar la arquitectura de las SDN debemos hablar de su antecedente que son las redes tradicionales, en estas los dispositivos de red trabajan en 3 planos los cuales son los siguientes:

- **Gestión:** encargado de establecer las políticas de la red y coordinar servicios
- **Control:** El cual se encarga del funcionamiento del sistema operativo y calcula utilizando varios algoritmos.
- **Reenvío:** Este cumple con la tarea de recibir y reenviar paquetes de datos.

Esta arquitectura en cambio a una tradicional desacopla las funciones de control y reenvío de la red, permitiendo que el control de la red sea directamente programable y que la infraestructura subyacente sea abstraída para aplicaciones y servicios de red.

3. MARCO TEÓRICO

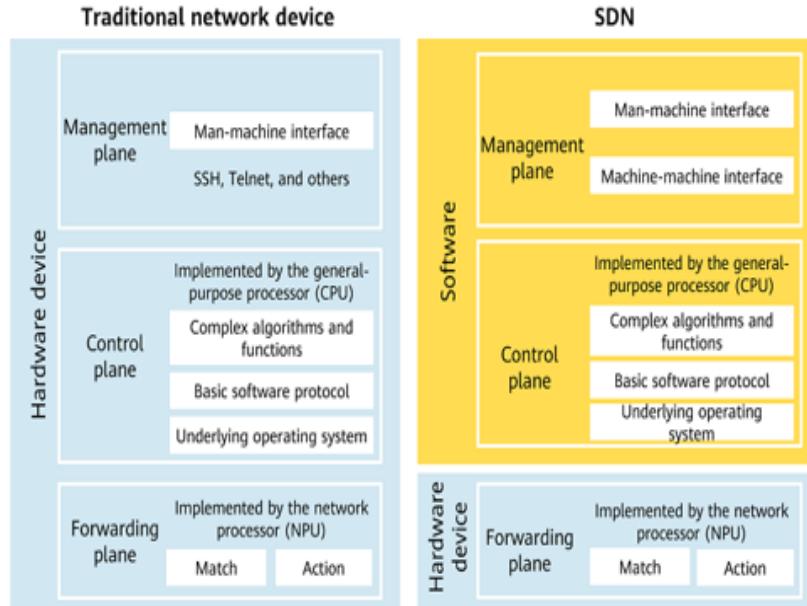


Figura 3.1: Comparación de arquitectura tradicional vs arquitectura SDN(2)

3.1.2 Componentes Básicos de la Arquitectura SDN

- **Capa de Infraestructura:** Consiste en dispositivos de reenvío, como conmutadores de centros de datos.
- **Capa de Control:** Consiste en software de control SDN y se comunica con dispositivos de reenvío a través de protocolos estándar para controlar la capa de infraestructura.
- **Capa de Aplicación:** Normalmente alberga plataformas en la nube basadas en OpenStack. Los usuarios pueden crear sus propias plataformas de gestión de la nube basadas en el software OpenStack.

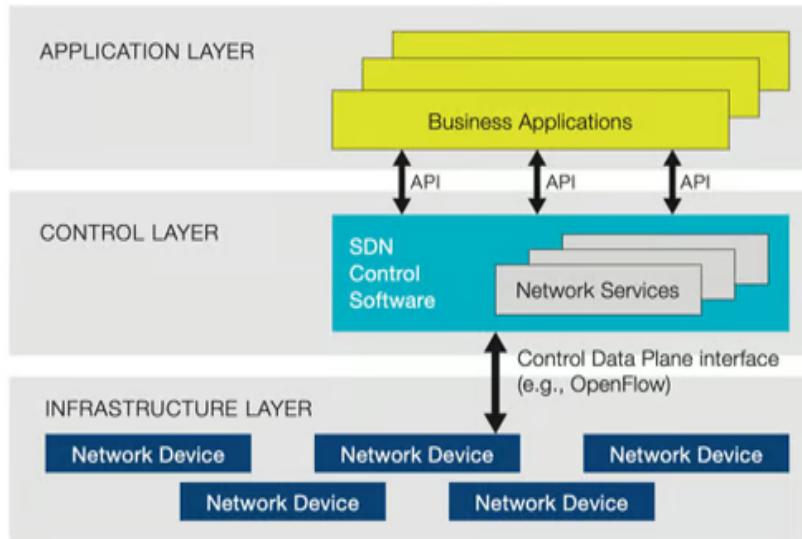


Figura 3.2: Relación entre las capas de la Arquitectura SDN(2)

3.1.3 Controladores SDN

Un controlador SDN lo podemos definir como el software que proporciona una vista centralizada y control sobre toda la red. Los administradores de red utilizan el controlador para gestionar cómo el plano de reenvío de la infraestructura subyacente debe manejar el tráfico y para aplicar políticas que dictan el comportamiento de la red.[12] Las SDN utilizan interfaces de programación de aplicaciones(API) de dos tipos:

- **API Northbound(Hacia el norte)** responsables de la comunicación entre la capa de aplicación y la capa de control.
- **API Southbound(Hacia el Sur)** Responsables de la comunicación entre la capa de infraestructura y la capa de control.

3.1.4 Características principales de una SDN

- **Programabilidad:** Las políticas de red se pueden configurar a través de aplicaciones escritas en lenguajes de programación de alto nivel desarrollando herramienta de automatización.

3. MARCO TEÓRICO

- **Gestión Centralizada** La inteligencia de la red SDN se centra en el software controlador de la red presentando políticas e instrucciones de control de flujo a los diferentes dispositivos de red.
- **Agilidad** Las redes SDN permite a los administradores de red proporcionar nuevas aplicaciones en caso de cambio en la estructura de la red. Permitiendo actualizaciones constantes de manera sencilla

3.2 Aprendizaje Automático

El aprendizaje automático(Machine Learning) es un subconjunto de la inteligencia artificial que permite que un sistema aprenda y mejore de forma autónoma con redes neuronales y aprendizaje profundo, sin necesidad de una programación explícita, a través del análisis de grandes cantidades de datos(15).Después estos datos se usan para crear un modelo de datos que puede hacer predicciones. Con más experiencia y datos, los resultados del aprendizaje automático son más precisos, de forma muy similar a cómo los humanos mejoran con más práctica.

3.2.1 Elementos del aprendizaje automático

Para entender como funcionar el aprendizaje automático, podemos dividirlo en 5 elementos clave que lo constituyen:

- **Set de Datos de Entrenamiento:** Colección de datos relacionados que generalmente se organizan en un formato estandarizado para el entrenamiento de modelos de IA.
- **Algoritmos de aprendizaje:** Fragmentos de código que ayudan a los usuarios a explorar y analizar conjuntos de datos complejos y a buscar significado en ellos. Cada algoritmo es un conjunto finito de instrucciones paso a paso que puede seguir una máquina para lograr un determinado objetivo.
- **Modelo:** Representación matemática que establece las relaciones entre las variables de los datos.
- **Características:** Son las propiedades que son medibles, o los atributos que se extraen de los propios datos y que el modelo analiza para identificar los patrones que le permiten ‘aprender’.

3.2.2 Tipos de aprendizaje automático

Una vez vistos estos elementos es importante entender que existen diferentes tipos de aprendizaje automático que nos pueden servir de mejor o peor manera dependiendo del tipo de problemática que se este abordando.

- **Aprendizaje Supervisado:** Consiste en un modelo de aprendizaje automático que utiliza datos etiquetados para entrenar. La salida esperada se conoce previamente, y el modelo aprende a partir de esa correspondencia.
- **Aprendizaje no Supervisado:** Consiste en un modelo de aprendizaje automático que usa datos sin etiquetar (datos no estructurados) para aprender patrones. A diferencia del aprendizaje supervisado, la “precisión” de la salida no se conoce de antemano. El algoritmo aprende de los datos sin intervención humana y los categoriza en grupos según los atributos.
- **Aprendizaje por Refuerzo:** Modelo de aprendizaje automático que se puede describir como “aprende haciendo” a través de una serie de experimentos de prueba y error. Un “agente” aprende a realizar una tarea definida a través de un ciclo de retroalimentación, recibiendo refuerzo positivo cuando lo hace bien y negativo cuando el desempeño es bajo.

3.2.3 Algoritmos de aprendizaje automático

Podemos establecer una definición para diferenciar con el punto anterior y este. Los algoritmos de aprendizaje automático consisten en fragmentos de código que ayudan a los usuarios a explorar y analizar conjuntos de datos complejos y a buscar significado en ellos. Cada algoritmo es un conjunto finito de instrucciones paso a paso inequívocas que puede seguir una máquina para lograr un determinado objetivo. En un modelo de aprendizaje automático, el objetivo es establecer o detectar patrones que los usuarios puedan usar para hacer predicciones o clasificar información. Por lo tanto los modelos son el resultado del proceso que lleva a cabo un algoritmo en relación con los datos de entrenamiento.

Los aprendizajes automáticos tienen un amplia gama de aplicaciones hacia que tareas puedan ser aplicados, a continuación se enlistan las descripciones de sus usos:

3. MARCO TEÓRICO

- **Clasificación:** Permiten la división de datos a categorías pre establecidas. Son útiles para problemáticas con salidas ya conocidas. Por ejemplo: ¿ Se romperá este neumático en los próximos 1500 kilómetros? ¿Sí o no?
- **Regresión:** Predicción de valores nuevos en función de datos históricos (Por ejemplo con la predicción de el precio de una casa en 3 años o cantidad de enfermos en 2 meses en una pandemia).
- **Agrupamiento (Clustering):** Organización de datos los cuales no estan etiquetados pero son relacionados en grupos con características similares.
- **Detección de anomalías:** Identificación de comportamientos inusuales o datos atípicos que se encuentran fuera de los parámetros. (como lotes de una linea de producción defectuosos o compras con tarjeta con una crédito irreconocibles).
- **Reconocimiento de patrones:** Nos dan la posibilidad de tener la habilidad de detección de estructuras o secuencias recurrentes dentro de un conjunto de datos.
- **Optimización:** Permite la toma de mejores decisiones en entornos complejos y dinámicos (como en logística, redes o planificación de recursos).

Como se puede ver hay una amplia gama de aplicaciones para estos algoritmos pero dentro de este trabajo vamos a hacer uso de 3 específicos los cuales son los siguientes.

1. **Random Forest**
2. **KNN(Vecinos mas Cercanos)**
3. **Regresión Logística]**

Pero son explicados de manera mas detallada desde su definición,funcionamiento y el fundamento en el que se basan.

1. Random Forest

Es un algoritmo de aprendizaje automático supervisado, registrado por Leo Breiman y Adele Cutler, que combina el resultado de múltiples árboles de decisión para llegar a un resultado único.

La idea principal de este algoritmo esta en la construcción de múltiples árboles de decisión durante la fase de entrenamiento para poder tener un resultado único. Su facilidad y flexibilidad ha permitido tener una eficiencia elevada ya se en tareas de clasificación al determinar de manera

correcta la clase de una instancia o de regresión siendo como salida la media de las predicciones de cada uno de los árboles individuales.

Funcionamiento General

- (a) **Creación de Subconjuntos mediante clasificadores (Bootstrapping):** Random Forest hace uso de una técnica llamada bagging o Bootstrap Aggregating, en la que se generan aleatoriamente diferentes subconjuntos a partir del set de datos original.
- (b) **Construcción de Árboles de Decisión:** Una vez generado cada subconjunto, se construye un árbol de decisión. Cada nodo del árbol tiene asignado un subconjunto permitiendo variedad en cada uno de los nodos. Dentro de estos se separan los datos en entrenamiento y prueba, dependiendo de si el problema es de regresión o clasificación.
- (c) **Predicción dada por cada árbol:** Una vez entrenados, cada árbol realiza una predicción individual.
- (d) **Combinación de Resultados:** Finalmente, las predicciones se combinan para generar una predicción final. En clasificación se utiliza votación por mayoría, y en regresión se calcula el promedio de los valores predichos.

Fundamento Como se vio anteriormente este algoritmo esta basado en arboles de decisión, el cual se puede definir individualmente como un modelo predictivo que divide el espacio de las variables predictivas en subregiones cercanas a la variable objetivo buscando una separación optimas (Algoritmo de Hunt) ya sea para problemas de clasificación o regresión. La construcción de este mismo esta basado en las instancias de entrenamiento de un nodo, si estos pertenecen a la misma clase se considera como un nodo terminal, pero si pertenecen a mas, se dividen los datos en conjuntos mas pequeños basados en una variable y así de manera iterativa. Para la selección de que variable elegir se usar el índice de Gini para problemas de clasificación, este mide el grado de pureza de un nodo es decir mide la probabilidad de no sacar dos instancias de una misma clase del nodo en cuestión. A menor indice menor pureza por que se selecciona la variable con menor indice ponderado. Este se define de la siguiente manera:

$$G(D) = 1 - \sum_{i=1}^k p_i^2$$

Donde:

- $G(D)$:Impureza de Gini para un conjunto de datos D .

3. MARCO TEÓRICO

- p_i : Proporción de elementos en D que pertenecen a la clase i , con
 k : Número total de clases.

Por lo que en pocas palabras esto permite al algoritmo minimizar esta impureza en cada división, buscando homogeneidad entre clases. De igual manera se hace uso de la entropía la cual nos permite determinar como los nodos se van a dividir en el árbol de decisión. En esencia la entropía es una medida usada para cuantificar el desorden un sistema. Para este caso si un nodo es puro su entropía es 0 y por lo tanto solo tiene instancias de una clase, pero si la entropía es igual a 1 se encuentra que existe la misma frecuencia para las clases. La entropía nos ayuda a definir la ganancia de información que busca la división de nodos. La entropía se defino como:

$$\text{Entropía} = \sum_{i=1}^C -p_i \log_2(p_i)$$

Donde P_i es la probabilidad de que un ejemplo sea de una clase específica.

2. KNN

K-Nearest Neighbors o K vecinos mas cercanos es un algoritmo supervisado de aprendizaje automático el cual basa su fundamento en encontrar la similitud de características para realizar predicciones en diferentes problemáticas, ya sean de clasificación o regresión. La idea básica esta en que los datos parecidos suelen encontrarse muy juntos("Vecinos") y los que presenten una menor distancia suelen ser de la misma clase. Como se puede observar en la siguiente figura como dentro de los óvalos hay objetos con características similares cuya distancia entre ellos es corta en comparativa con los que se encuentran mas alejados.

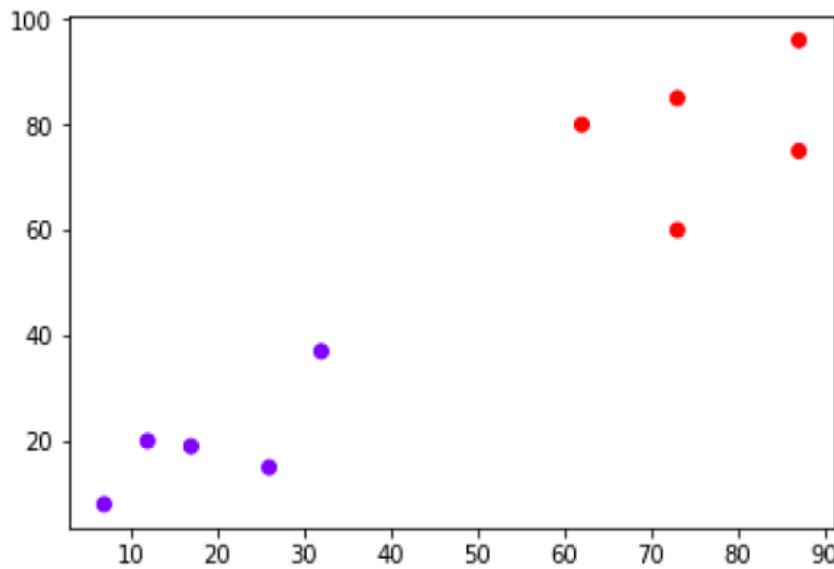


Figura 3.3: Ejemplo de instancias de un set de datos(3)

Funcionamiento General

- Selección de K:** Se establece un numero de vecinos mas cercanos (K) los cuales serán consideradas para establecer la predicción.
- Calculo de distancias:** Para cada instancia o ejemplo, se calcula la distancia hacia otra que en este caso a los datos de entrenamiento(Puede ser distancia Euclidiana, Manhattan o Hamming)
- Identificación de vecinos:** Se procede a seleccionar los puntos K de entrenamiento mas cercanos al dato que se desea realizar la predicción.
- Predicción:** Finalmente, las predicción se asigna la clase mas frecuente entre los K vecinos

3. Regresión Logística

Es un modelo estadístico basado en un tipo de regresión muy similar a regresión lineal ampliamente utilizado para analizar y predecir variables de respuesta categóricas. La regresión logística modela la probabilidad de que ocurra un evento específico, transformando una combinación de variables predictivas basándose en la función logística o Sísmoide.

Este modelo tiene especial uso en situaciones donde el objetivo es clasificar observaciones en categorías mutuamente excluyentes es decir que de estas dos categorías no pueden suceder al mismo tiempo como si

3. MARCO TEÓRICO

hay tráfico en un autopista o no.

Funcionamiento General

- (a) **Transformación de Probabilidades:** Se utiliza la función logística para transformar la salida de una combinación lineal de variables predictivas en una probabilidad binaria es decir 0 o 1. Esta función esta definida de la siguiente manera:

$$P(Y = 1) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X_1 + \dots + \beta_k X_k)}}$$

Donde:

- β_0 : Intercepto (Logaritmo de la probabilidad de un resultado cuando todas las variables independientes son cero).
- β_1, \dots, β_k : Coeficiente asociado a variables predictivas.
- X_1, \dots, X_k : Variables Predictivas

- (b) **Estimación de Parámetros:** Se hace una estimación de los coeficientes del modelo mediante el método de máxima verosimilitud, el cual esta basado en buscar los valores idóneos para maximizar la probabilidad de observar los datos basados en los parámetros del modelo
- (c) **Interpretación de Coeficientes:** Cada coeficiente β_i representa el cambio en el logaritmo de las probabilidades de que ocurra el evento que se busca por unidad de cambio en la variable X_i , manteniendo constantes las demás variables.
- (d) **Clasificación:** Una vez estimadas las probabilidades, se puede establecer un umbral (por ejemplo, 0.5) para clasificar

3.2.4 Simuladores de Red

Los simuladores de red se definen como herramientas de software las cuales permiten crear, modelar y analizar el comportamiento de redes de computadoras en un entorno virtual. Estos simuladores facilitan el diseño y prueba de topologías de red sin la necesidad de implementar físicamente los dispositivos, lo que resulta útil para la educación, investigación y planificación de infraestructuras de red. Los simuladores de red pueden ser clasificados basados en sus características de la siguiente forma:

- **Gráfica de Usuario (GUI):** Proporciona una interfaz visual para la creación y gestión de topologías de red.

- **Soporte Multiplataforma:** Compatibilidad con sistemas operativos Windows, macOS y Linux, ofreciendo flexibilidad en diversos entornos.
- **Integración con Múltiples Tecnologías:** Permite la integración con diversas herramientas y tecnologías, incluyendo Wireshark para el análisis de tráfico y Docker para la implementación de contenedores.
- **Escala labilidad:** Capaz de manejar desde pequeñas topologías en una sola máquina hasta redes complejas distribuidas en múltiples servidores.
- **Simuladores de Propósito General:** Permiten la simulación de una gran variedad de protocolos y topologías de red. Algunos ejemplos son: *NS-3*, *OMNet++*.
- **Simuladores Específicos por Fabricante:** Diseñados específicamente para crear entornos de simulación de dispositivos propios del fabricante. Ejemplo: *Cisco Packet Tracer*.
- **Emuladores de Red:** Sistemas que permiten ejecutar sistemas reales de dispositivos de red en un entorno virtual de pruebas. Ejemplo: *GNS3*.

Aunque actualmente existen variedad de simuladores en el mercado, en el presente proyecto se hizo uso de GNS3 por su facilidad de trabajar con software de terceros. **GNS3 Graphical Network Simulator-3**(16) Este es una plataforma de simulación de redes de código abierto que permite a los profesionales de redes diseñar, configurar, probar y solucionar problemas en entornos de red virtuales y reales. Es ampliamente utilizada para emular redes complejas sin la necesidad de hardware físico, facilitando el aprendizaje y la experimentación en el ámbito de las redes de computadoras. GNS3 esta basado en las herramientas de Dynamips, PEMU (incluyendo el encapsulador) y en parte en Dynagen, fue desarrollado en Python a través de PyQt para el desarrollo de la interfaz gráfica(GUI). GNS3 de igual manera esta apoyado en la tecnología SVG (Scalable Vector Graphics) la cual cumple con la tarea de proveer símbolos de calidad para los diferentes dispositivos presentes para representar en la topología de la red. **Herramientas de GNS3:**

- **DYNAMIPS:** Este es un emulador de routers de cisco. Emula las plataformas 1700,2600,3600,3700 y 7200. Permite ejecutar imágenes IOS teniendo acceso a una versión virtual de dispositivos cisco.
- **DYNA GEN:** Interfaz de texto específicamente diseñada para DYNAMIPS la cual simplifica la configuración de los dispositivos emulados.
- **QEMU(Quick Emulator):** Hipervisor de PC para ejecutar sistemas operativos y dispositivos virtuales y otros sistemas que no están basados directamente en IOS.

3. MARCO TEÓRICO

Características de GNS3:

- **Interfaz Gráfica de Usuario (GUI):** Proporciona una interfaz visual para la creación y gestión de topologías de red.
- **Soporte Multiplataforma:** Compatibilidad con sistemas operativos Windows, macOS y Linux, ofreciendo flexibilidad en diversos entornos.
- **Integración con Múltiples Tecnologías:** Permite la integración con diversas herramientas y tecnologías, incluyendo Wireshark para el análisis de tráfico y Docker para la implementación de contenedores.
- **Escalaabilidad:** Capaz de manejar desde pequeñas topologías en una sola máquina hasta redes complejas distribuidas en múltiples servidores.
- **Funcionalidades:** Dentro de GNS3 se cuenta con la posibilidad de la simulación de dispositivos de red como routers, switches, y servidores, así como dentro de ellos podemos configurar protocolos de red, pruebas de escenarios de fallos etc. Al poder cargar imágenes de dispositivos es especialmente útil para emular dispositivos CISCO, aunque también puede ser integrado con otras tecnologías como soporte para virtualización.

Aunque GNS3 ofrece una gran cantidad de funcionalidad y es ideal en múltiples ambientes, es importantes mencionar la limitación que presenta al requeriría un gran cantidad de recursos significativos y su uso puede ser un poco complejo si no se tiene experiencia previa en redes.

3.2.5 Calidad de Servicio(QoS)

La **calidad de servicio (QoS)** es el uso de mecanismos o tecnologías que funcionan en una red para controlar el tráfico y garantizar el rendimiento de aplicaciones críticas con capacidad de red limitada. En el contexto de redes que presentan alta demanda y problemáticas de congestión, QoS juega un papel crucial al permitir que el tráfico de mayor prioridad, mantenga una calidad de servicio y rendimiento correctos incluso bajo condiciones de sobrecarga.

Es importante entender que la congestión ocurre cuando múltiples flujos de datos intentan acceder los mismos recursos, lo que da como resultado distintas problemáticas como retrasos, pérdida de paquetes y bajo rendimiento de la red en cuestión. El hecho de tener implementado QoS permite gestionar esta congestión al dar prioridad al tráfico crítico y al reservar el ancho de banda para aplicaciones con tareas específicas.

3.2 Aprendizaje Automático

Estas métricas en las que el QoS esta presente son las siguientes y como el QoS mitiga estos efectos:

1. **Ancho de banda:** La velocidad de un enlace. QoS puede indicar a un enrutador cómo usar el ancho de banda. Por ejemplo, asignar una cierta cantidad de ancho de banda a diferentes colas para distintos tipos de tráfico.
2. **Demora (Latencia):** El tiempo que tarda un paquete en pasar de su origen a su destino final. Esto puede verse afectado por retrasos en la cola durante la congestión. QoS permite evitar esto mediante la creación de colas de prioridad para ciertos tipos de tráfico.
3. **Pérdida de paquetes:** La cantidad de datos perdidos debido a la congestión en la red. QoS permite definir qué paquetes deben descartarse en caso de saturación.
4. **Fluctuación (Jitter):** Variación irregular en la velocidad de llegada de los paquetes debido a congestión, lo cual puede causar distorsión o interrupciones en audio y video.

3. MARCO TEÓRICO

Capítulo 4

Metodología y Desarrollo

En este capítulo vamos a mostrar la implementación y configuración de nuestro modelo para administrar tráfico de red. Para ello seguiremos los siguientes pasos:

1. Describiremos el modelo principal del sistema
2. Definiremos y justificaremos las métricas a evaluar.
3. Procederemos a la captura de las métricas de red en los switches SDN y se procederá a analizarlas para encontrar valores de retardos, jitter, utilización y variación del throughput.
4. Posterior a esta parte empezaremos con estos datos, a la parte de la inteligencia de artificial donde entrenaremos los algoritmos de Machine Learning para comparar su precisión.
5. Por último ya con el modelo entrenado, implementaremos la conmutación de rutas basados en las métricas analizadas.

4.1 Modelo principal del sistema

El modelo del sistema propuesto representa el funcionamiento general del sistema, el cual está presentado en la figura 4.1.

Como entrada principal del sistema, se tiene la extracción de métricas de red del tráfico de cada una de las interfaces de los switches individuales, donde contenedores de Docker cargados con el sistema operativo Ubuntu generan tráfico con diferentes parámetros los cuales pueden ser configurados como el tiempo, ancho de banda o tipo de tráfico. En cuanto a salida del sistema, se

4. METODOLOGÍA Y DESARROLLO



Figura 4.1: Modelo general del sistema

genera una predicción de niveles de congestión cada segundo, a la par de ello se guarda un historial de que tantas veces se mantiene congestionado una interfaz. Basados en este historial se manda la señal de activación el proceso de adaptación de la interfaz donde transita el tráfico de la red que pueda estar o no congestionada.

El proceso de este modelo es dividido en tres sub-procesos los cuales se encargan de la captura de tráfico, ejecución del modelo de Machine Learning y finalmente la generación de las predicciones de congestión de cada interfaz individual. Aparte de la posible commutación de rutas en la red la cual retroalimenta el comportamiento del sistema el cual buscar ofrecer rutas de respaldo en caso de desborde la capacidad de algún switch.

A grandes rasgos estos sub-procesos son posibles debido a los dos módulos que componen la arquitectura de la aplicación de gestión de la red los cuales son: el Controlador de Switch (Basado en el framework Ryu) y el Monitor de Congestión. El controlador Ryu dirige lo switches SDN presentes en la red, definiendo reglas de flujo y en general el comportamiento de la red. De manera simultanea, el monitor de congestión consulta y recopila métricas definidas sobre el desempeño de la red, las cuales son almacenadas en una Base de Datos externa como se ha mencionado para su posterior consulta y análisis. Además se ponen a disposición del controlador las predicciones hechas por el modelo de Machine Learning, las cuales son usadas para tomar decisiones en tiempo real del estado de la red.

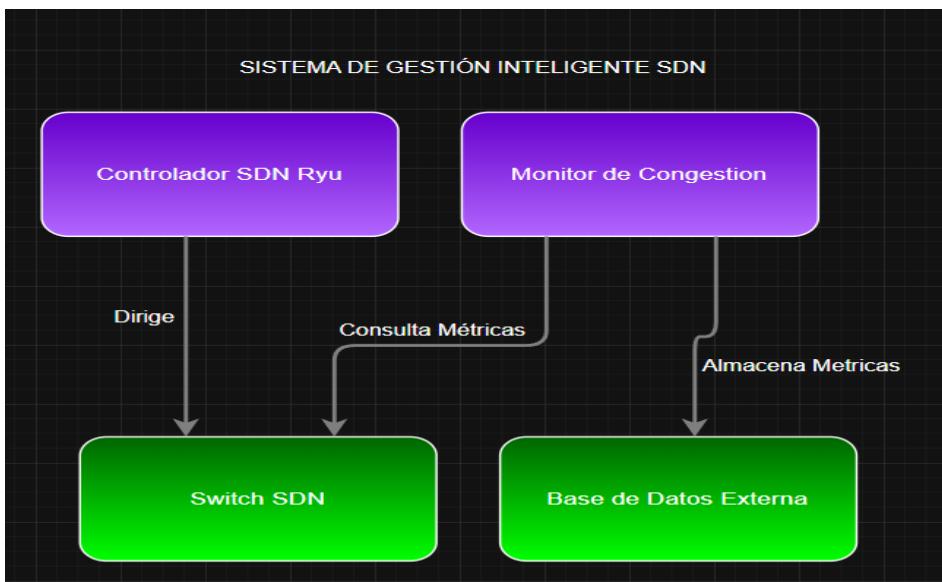


Figura 4.2: Arquitectura de la Aplicación

4.2 Métricas a evaluar

Como ya se menciono anteriormente las métricas que fueron utilizadas como entrada del modelo de Machine Learning y de ahí poder tener una clasificación. Estas métricas fueron calculadas a partir de la consulta en tiempo real de los switches SDN a través del protocolo OpenFlow.

- **Utilización del puerto (%)**, Representa el porcentaje de uso del ancho de banda disponible en un puerto y se calcula con la fórmula:

$$U = \left(\frac{(TX + RX) \times 8}{\Delta t \times B_{\max}} \right) \times 100$$

donde TX y RX son los bytes transmitidos y recibidos durante el intervalo de monitoreo Δt (en segundos), y B_{\max} es el ancho de banda máximo del puerto (en bits por segundo).

- **Delay (Retardo)**: Se calcula enviando paquetes especiales con una estampa de tiempo inyectada en la capa de enlace. El controlador mide el tiempo transcurrido entre el envío y la recepción de dicho paquete, aplicando la fórmula:

$$D = t_{\text{recepción}} - t_{\text{envío}}$$

4. METODOLOGÍA Y DESARROLLO

- **Jitter:** Variación del delay entre dos paquetes consecutivos y se define como:

$$J = |D_i - D_{i-1}|$$

donde D_i y D_{i-1} son los retardos medidos en los últimos dos paquetes recibidos con estampa de tiempo

- **Variabilidad del Throughput**, J_{TP} , es la desviación estándar del throughput medido en los últimos n intervalos de tiempo. Siendo TP_1, TP_2, \dots, TP_n los valores históricos:

$$J_{TP} = \sqrt{\frac{1}{n-1} \sum_{i=1}^n (TP_i - \bar{TP})^2}$$

donde \bar{TP} es el promedio del throughput registrado.

Para fines de testeo el desarrollo de este proyecto fue creado un ambiente de red emulado en el software de GNS3, se usaron diferentes topologías con propósitos específicos, las topologías tienen los mismo elementos tan solo cambia la forma y el propósito en porque están así. A grandes rasgos estos elementos son presentados en la tabla 4.1 con sus características específicas necesarias para el desarrollo del proyecto.

4.3 Captura y Análisis de Métricas de Red en Switches SDN

Dispositivo	Características	Descripción
Open vSwitch(17)	<ul style="list-style-type: none"> Imagen: Open vSwitch with Management Linux (Debian-based) Administración vía interfaz CLI Versión: 2.15.0 RAM 256 MB 	Switch virtual programable utilizado como nodo central en la topología SDN. Permite instalar reglas de flujo desde el controlador y reenviar paquetes según esas reglas.
Contenedores Docker	<ul style="list-style-type: none"> Imagen base: Ubuntu 22.04 Herramientas: iproute2, net-tools, iperf3, tcpdump, etc. Configuración de red optimizada para pruebas Interfaz eth0 habilitada 	Hosts virtuales configurados para generar tráfico de red con diferentes características. Simulan usuarios que consumen servicios como voz, datos o streaming.
VM (Ubuntu Desktop)(18)	<ul style="list-style-type: none"> SO Ubuntu Desktop linux 20.04.4 LTS RAM 3GB Procesadores 6 Almacenamiento 50GB Adaptador de red Intel PRO/1000 MT Desktop 	Máquina virtual que actúa como estación de administración o controlador del sistema. Ejecuta programas de monitoreo enviando mensajes OpenFlow, también ejecuta el modelo de IA y tiene el panel de visualización de métricas.
Ethernet Switch (GNS3)	<ul style="list-style-type: none"> Switch virtual por defecto en GNS3 Usado para conexión central entre nodos SDN Compatible con enlaces múltiples 	Switch virtual de GNS3 que interconecta los distintos dispositivos de la topología. Facilita el tráfico entre nodos sin requerir hardware físico.

Tabla 4.1: Dispositivos utilizados en la topología y su función

4.3 Captura y Análisis de Métricas de Red en Switches SDN

En el subprocesso representado en la Figura 4.2, se lleva a cabo la captura y análisis de diversas métricas operativas provenientes de los switches que conforman la topología de red. La comunicación entre los switch y el controlador SDN se realiza a través de la interfaz de control dedicada `eth0`, la cual opera estrictamente en la capa 2 (enlace de datos), permitiendo el

4. METODOLOGÍA Y DESARROLLO

intercambio directo de tramas.

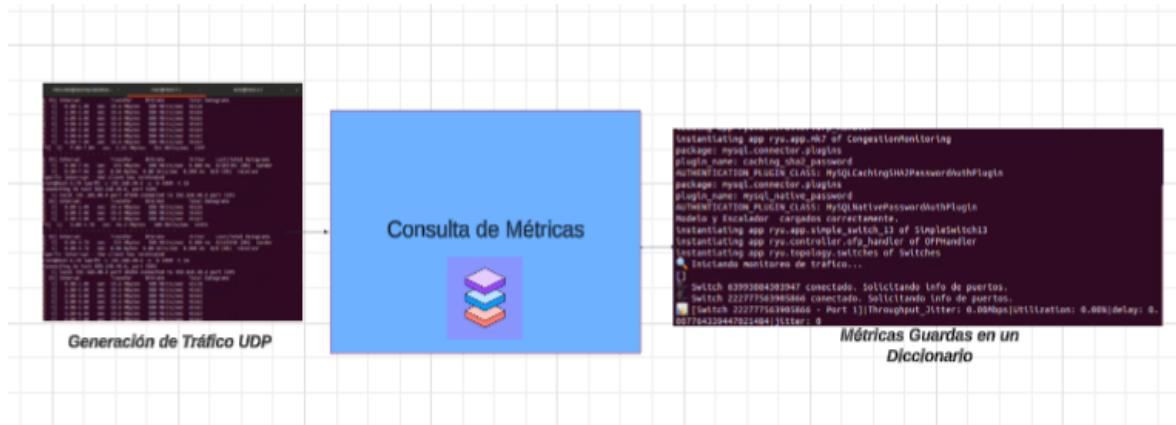


Figura 4.3: Captura de Métricas

Como entrada principal a este proceso, se define primero la conexión a switch activos detectados por el controlador, para después las posteriores solicitudes periódicas del controlador a los switch mediante mensajes OpenFlow(*PortStatsRequest*), específicamente para este caso se utilizaron paquetes UDP al no tener control de congestión de manera nativa, nuestro proyecto mostrará una mejora.

Para el proceso como tal que se realiza, el controlador SDN comienza un monitoreo continuo con los switch activos registrados activos. Esto se lleva a cabo a partir del envío periódico de mensaje Open Flow que solicitan métricas detalladas de los puertos, incluyendo número de bytes transmitidos y recibidos. Con estos valores la aplicación calcula métricas derivadas apartir de estos como el throughput, utilización del puerto en porcentaje, tamaño promedio de los paquetes etc.

De manera paralela se analizan los paquetes inyectados previamente con la marca de tiempo , para que al recibir un paquete alterado se pueda comparar el retraso del tiempo del switch que está recibiendo el paquete, permitiendo generar un historial. Todo este proceso ocurre de manera autónoma al administrador de red, permitiendo tener un reporte del comportamiento de la red sintetizado.

Finalmente a la salida de este subprocesso, se obtienen las métricas recolectadas por cada uno de los switch activos y cada uno de los puertos pertenecientes:*throughput*, utilización, *retraso*, *jitter*, tamaño promedio de paquetes (TX/RX), *jitter* del *throughput*. También todas estas métricas guardadas en una base de datos mysql para su análisis posterior ya que con ellas podremos construir un centro de gráficas basado en Grafana para análisis

histórico de cada uno de los puertos así como su estatus de congestión.

4.4 Ejecución de algoritmo de Machine Learning

4.4.1 Generación del Data-Set

Debido a que no se pudo encontrar ningún conjunto de datos relevante para definir la congestión, por lo que se procedió a implementar y configurar dos topologías distintas para generar diferentes escenarios. Dentro de cada una de las topologías hay una máquina virtual cargada con un programa para generar tráfico con diferentes características y durante distinto tiempo, de igual manera nos sirvió para automatizar esta generación de tráfico mediante la herramienta iperf, estableciendo hosts como servidores que escuchan y otros que generan tráfico.

Topología 1: Mancuerna

La primera topología, consiste en tres switch Open vSwitch (OVS), tres contenedores Docker que simulan host finales basados en Ubuntu cargados con herramientas para generar tráfico de red y con su respectiva red ip configurada aparte de una maquina virtual cargada con el mismo sistema operativo, estos se conectan directamente a los switch Estos dos switch están conectados entre sí mediante un tercer switch , el cual a su vez conecta otros dos hosts adicionales. Esta configuración de topología permite un flujo bidireccional entre ambos extremos, teniendo un enlace intermedio el cual es propenso a congestión inter switch. De igual manera para la conexión entre el controlador SDN y los switch , estos están conectados a través de un switch Ethernet virtual para su control respectivo.

Topología 2: Árbol

La segunda topología tiene el mismo propósito que la anterior ya que se quiere medir congestión inter switch y un traspaso de un mayor número de nodos de todo el tráfico. Esta topología consiste en una estructura similar a un árbol de tres niveles. De igual manera hay hosts conectados a cada uno de los switches. Vemos que de igual manera el controlador esta conectado a un switch virtual para que funciones como un nodo central de estructura.

4. METODOLOGÍA Y DESARROLLO

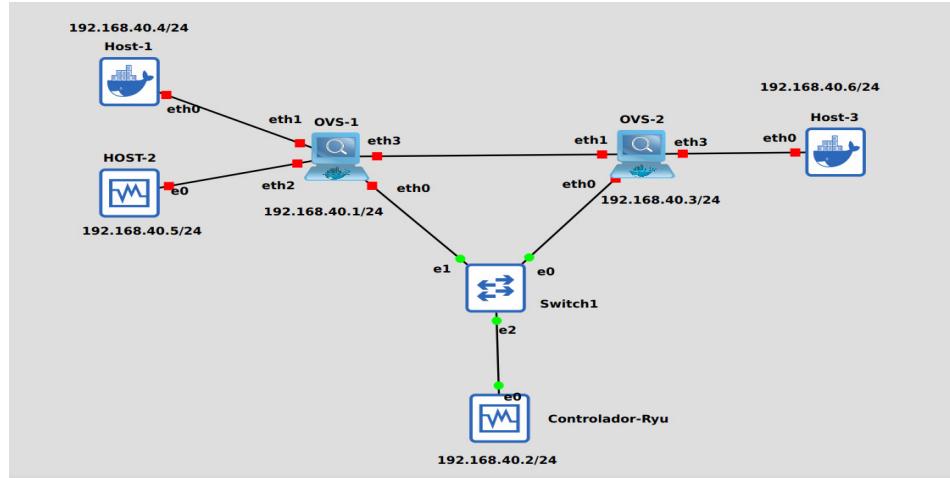


Figura 4.4: Topología Mancuerna

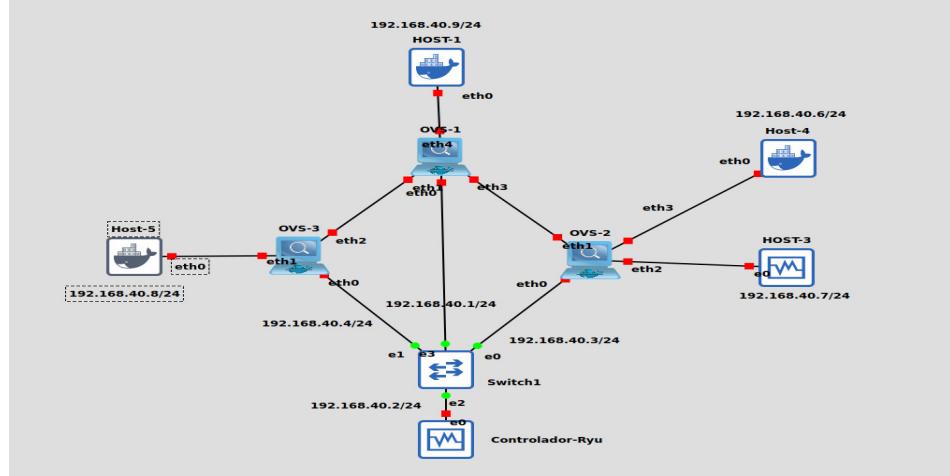


Figura 4.5: Topología Árbol

Es importante mencionar que estas topologías son redes LAN funcionando como switches convencionales aunque la ventaja de ser una red SDN es que puede ser analizado el tráfico y modificar su comportamiento. Una vez generado nuestro conjunto de datos para un etiquetado entre que sería congestión o no se procedió a establecer umbrales con las métricas de qué se consideraría o no congestión, por lo que para mejorar la identificación de cada una las instancias se agrego un campo más para saber el escenario de la topología en la que fue generado y saber si es o no congestión.

4.4.2 Entrenamiento del modelo

Una vez que se tiene el conjunto de datos generado y disponible en un formato para ser usado se utilizo Google Collab la cual es una herramienta gratuita que permite escribir, ejecutar y compartir código Python en la nube en "cuadernos".

El entrenamiento del algoritmo consiste de los siguientes pasos: De primera instancia se tiene el preprocesamiento del conjunto de datos el cual tiene la tarea de limpiar y adecuar los datos, para que el algoritmos pueda interpretarlas de manera clara. Se usaron librerías adecuadas ya que nos proporcionan las herramientas ya codificadas para llevar acabo esta tarea, estas fueron pandas y scikitlearn, aunque se tuvo que hacer uso de una versión específica de scikitlearn la cual es la 1.3.2 debido a problemas de compatibilidad dentro de Linux. Como se vio anteriormente el conjunto de datos, se registro con diferentes escenarios de las instancias pero es necesario etiquetar el modelo con valores binarios para que pueda funcionar la clasificación: 0 para "sin-congestión" y 1 para "congestión moderada o severa" aumentando la sensibilidad ante condiciones de congestión y actuar de manera proactiva. Los datos fueron procesados usando pandas como se mencionaba para manipular el conjunto de datos para convertirlo en un conjunto de datos el cual es una estructura de datos propia de la librería. Los datos fueron normalizados es decir, se metieron en un rango específico para que no haya problemáticas de escalado y estar en el mismo rango. Una vez realizado este proceso se eliminan las columnas irrelevantes que no aportan información a la clasificación Posterior a esta etapa se divide en conjuntos de entrenamiento con un 80% para el entrenamiento y un 20% para pruebas posteriores al entrenamiento para que pueda clasificarlo sin tener acceso a las etiquetas. Una vez teniendo los datos separados es relevante saber las distribución de clases, y como se puede ver en la figura 4.6 es implementada vemos que las clases están extremadamente desbalanceadas por lo que es necesario el uso de un mecanismo para mitigar esto ya que podría generar un sesgo en el modelo. Para abordar esta problemática se utilizo el método **SMOTE TOMEK**, la cual consiste en la combinación de dos técnicas, la primera siendo SMOTE (Synthetic Minority Over-sampling Technique) teniendo la tarea de generar ejemplos sintéticos de cada una de las clases que tengan menores instancias, y TOMEk la cual elimina ejemplos ambiguos que no representen información útil para el modelo.

Como se menciono al principio de este trabajo se usaron 3 diferentes tipos de algoritmos todos estos fueron optimizados através de los hiperparámetros,

4. METODOLOGÍA Y DESARROLLO

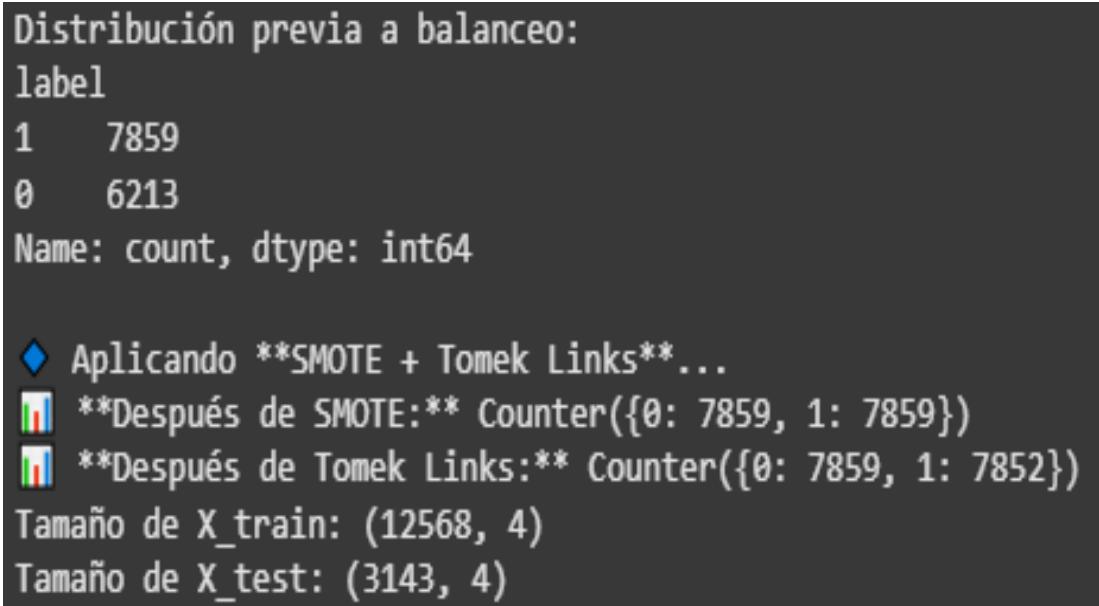


Figura 4.6: Distribución de clases antes y después de SMOTE TOMEK

buscando los valores más óptimos para cada uno de los algoritmos, para ello se utilizo la herramienta GridSearchCV, permitiendo tener la mejor configuración y así el mejor rendimiento para las fase del entrenamiento.

- **Random Forest:** Algoritmo basado en múltiples árboles de decisión entrenado cada uno de ellos con subconjuntos de datos seleccionado aleatoriamente, el mejor desempeño que se obtuvo fue con lo siguientes hiperparámetros: n_estimators=100 (número de arboles), min-samples-split=5 (número mínimo de muestras requeridas para dividir un nodo (valor más alto evita sobreajuste)), min-samples-leaf=1 (número mínimo de muestras en una hoja, más alto da árboles más simples), max-depth=None (Sin limitación para crecimiento de arboles) y class-weight='balanced' (Ajusta el peso de las clases en función de la frecuencia en el set de datos).
- **KNN:** Basa su clasificación en función de la distancia de vecinos mas cercanos refiriendo a que las instancias con características mas cercanas pertenecen a la misma, su mejor resultado fue obtenido con n-neighbor=5 (número de vecinos más cercanos que se considerarán para determinar que pertenezcan o no a una clase), weights='distance' (los vecinos más cercanos tienen mayor influencia) y p=2 (uso de distancia Euclidiana para calcular la cercanía entre puntos).
- **Regresión Logística:** Modela la probabilidad de que una instancia

4.4 Ejecución de algoritmo de Machine Learning

pertenecza a una instancia específica en función de una combinación de variables de entrada, tuvo su mejor configuración con C=0.1(inverso de la fuerza de regularización (menor valor = mayor regularización = modelo más simple)), max iter=100(número máximo de iteraciones), solver='lbfgs'(método de optimización utilizado para ajustar los pesos.

Para evaluar la eficiencia, se analizaron métricas clave como la precisión: la cual refleja la proporción de predicciones correctas sobre el total de predicciones positivas realizadas, recall: siendo el número de instancias que fueron correctamente clasificadas, F1-score: es una media entre los dos anteriores. Los resultados de los modelos ya entrenados, se presentan en las figuras 4.7,4.8 y 4.9.

Reporte de Clasificación:				
	precision	recall	f1-score	support
0	1.00	1.00	1.00	1572
1	1.00	1.00	1.00	1571
accuracy			1.00	3143
macro avg	1.00	1.00	1.00	3143
weighted avg	1.00	1.00	1.00	3143

Figura 4.7: Métricas de rendimiento Random Forest

4. METODOLOGÍA Y DESARROLLO

```
F1 Score (Weighted): 0.9990

Accuracy: 0.9990454979319122

Reporte de Clasificación:
precision    recall   f1-score   support

          0       1.00      1.00      1.00      1572
          1       1.00      1.00      1.00      1571

accuracy                           1.00      3143
macro avg                           1.00      1.00      3143
weighted avg                          1.00      1.00      3143
```

Figura 4.8: Métricas de rendimiento KNN

```
Accuracy: 0.9825007954183901

Reporte de Clasificación:
precision    recall   f1-score   support

          0       0.98      0.98      0.98      1572
          1       0.98      0.98      0.98      1571

accuracy                           0.98      3143
macro avg                           0.98      0.98      3143
weighted avg                          0.98      0.98      3143
```

Figura 4.9: Métricas de rendimiento Regresión Logística

Vemos que para los 3 modelos la precisión recall y f1 score es muy cercano a 1.0 en ambas clases (no congestión y congestión) lo que indica que los modelos identifican casi todas las instancias cometiendo muy pocos errores demostrando que de las 3143 instancias disponibles no hubo discrepancias. De igual manera para evaluar el análisis del rendimiento del modelo se hizo una matriz de confusión para visualizar falsos positivos o negativos que estén arrojando los modelos. Las matrices de confusión se ven en la figura 4.10,4.11 y 4.12.

4.4 Ejecución de algoritmo de Machine Learning

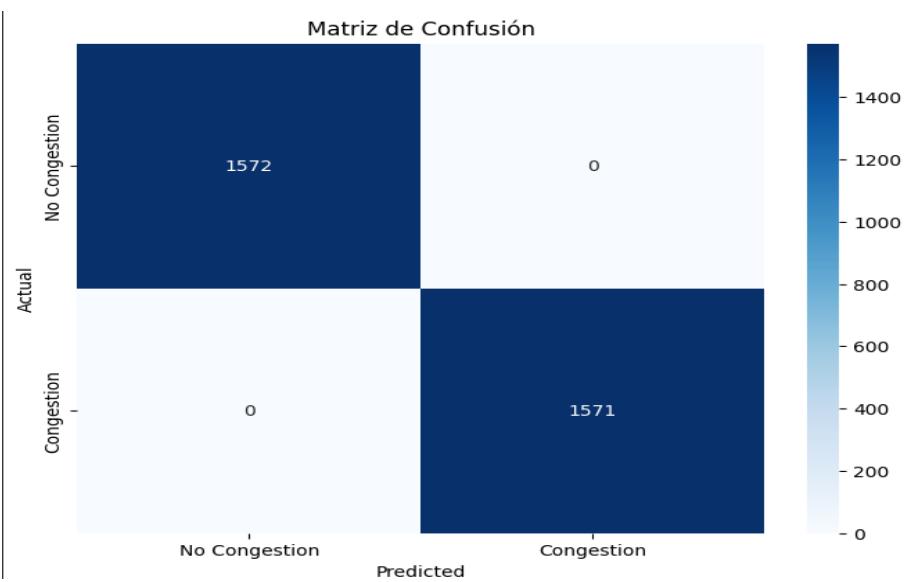


Figura 4.10: Métricas de rendimiento Random Forest

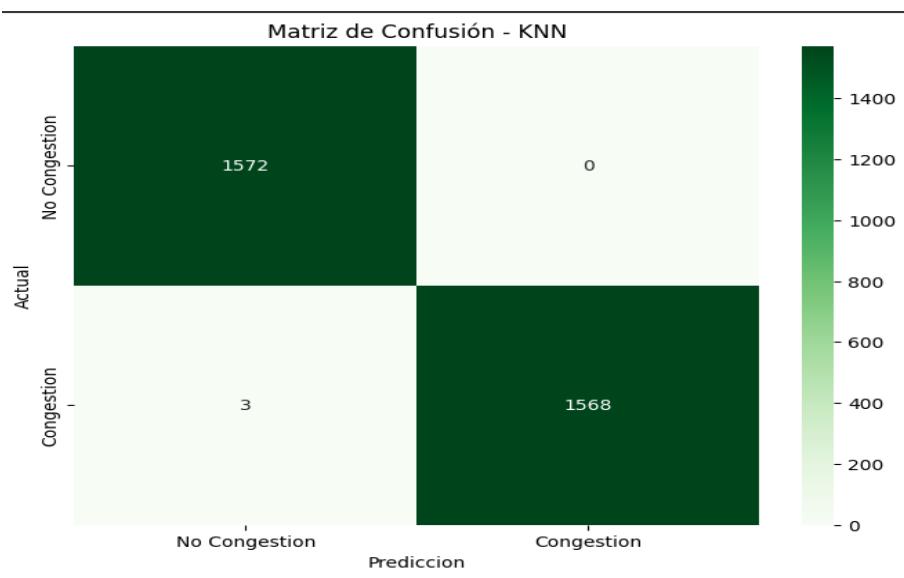


Figura 4.11: Métricas de rendimiento KNN

4. METODOLOGÍA Y DESARROLLO

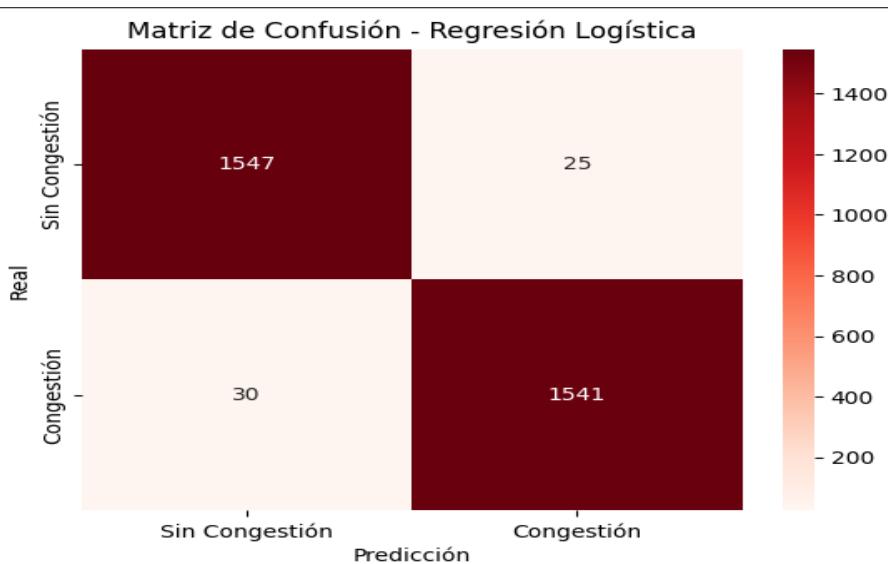


Figura 4.12: Métricas de rendimiento Regresión Logística

Algunos puntos importantes a mencionar para la interpretación de las matrices de confusión esta en que:

- En Random Forest: De las 3143 muestras de instancias del tráfico de red, se clasificaron correctamente todos los casos en ambas clases tanto sin congestión como cuando se presentaba congestión.
- En KNN: De las 3143 muestras de instancias del tráfico de la red, se clasificaron casi todas las instancias de manera correcta presentando solo 3 falsos negativos.
- En Regresión Logísticas: De las 3143 muestras de instancias del tráfico de la red, se presentaron 30 falsos negativos y 25 falsos positivos.

Viendo los resultados de los 3 modelos podemos concluir que claramente el que presenta una mejor precisión de clasificación esta presente con Random Forest mostrando un rendimiento mayor al de los demás modelos, aparte de no presentar ningún falso positivo ni negativo, pudiendo presentar una gran cantidad de datos en tiempo real.

4.5 Conmutación de rutas

Finalmente este subprocesso se encarga de tomar acciones depende de la cantidad de tiempo en que un puerto de un switch este congestionado

para poder mitigar este estado y sus posibles efectos en la red como el desbordamiento de la red.

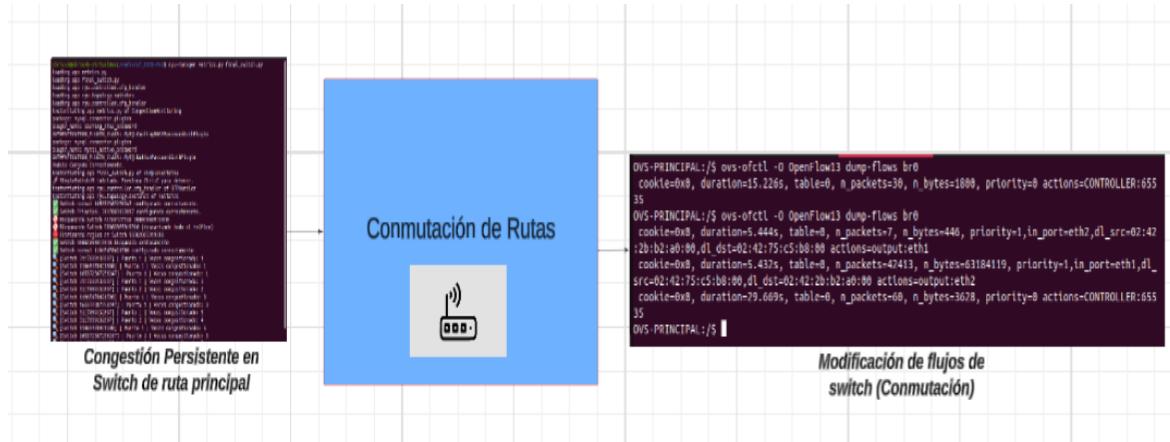


Figura 4.13: Diagrama de subprocesso conmutación de rutas

Este subprocesso se basa en la supervisión periódica de las predicciones hechas por el modelo. Cuando uno de los switch muestra un historial con niveles de congestión persistente, activa un proceso el cual conmuta todo el tráfico de la red a el otro switch debido a la exposición prolongada a la congestión de la ruta principal y ofrecer una ruta no congestionada en caso de fallos. Se tiene un registro de los identificadores que será el switch principal y los alternos que serán los que modificaran su comportamiento.

La lógica general en la que se basa este subprocesso consiste en identificar los flujos que se encuentran activos por el puerto congestionado, después se selecciona la ruta alternativa que al estar desactivada presentara congestión nula. Con esto se procede a redirigir los flujos hacia el switch alterno seleccionado, tomando las reglas de reenvío necesarias que se tengan registradas, para que de igual manera se pueda actualizar el registro de tablas de direcciones MAC, garantizando la comunicación entre toda la topología.

Este enfoque nos permite mantener el servicio incluso en el hecho de una falla total de la ruta principal aplicando medidas correctivas ante tales eventos.

4.5.1 Implementación en el sistema

Con esto podemos pasar a la integración de los subprocesos, los cuales generalizando, comienzan con la ejecución del modelo de Machine Learning entrenado apartir de métricas consultadas y procesadas. Esto permite al

4. METODOLOGÍA Y DESARROLLO

sistema clasificar niveles de congestión en puertos específicos de cada uno de los switches presentes en la red. De igual manera la aplicación tiene la capacidad de darle a los switches el comportamiento de switch convencional de capa 2, por lo que de igual manera se maneja toda la gestión del tráfico dentro de la lógica de nuestro código. Es importante mencionar que debido a las métricas seleccionadas solo se aplica el modelo a el switch principal, aunque se siguen monitoreando las métricas y se siguen guardándose en la base de datos para su posterior análisis. A la entrada del sistema se reciben los datos recolectados en tiempo real en los switch presentes en la topología y conectados al controlador, estas incluyen métricas de desempeño general de la red como el porcentaje de utilización del puerto, el retraso, el jitter, el jitter del throughput. Estas métricas se obtienen a través de consultas mediante mensajes OpenFlow y de la recepción de paquetes especiales los cuales se les inyecta una marca de tiempo para poder medir el retraso de la red. Los datos capturados son concentrados en un diccionario de métricas por puerto.

El primer paso consiste en tener las métricas en una estructura de datos para poder ingresarlos al modelo, lo cual haremos con la biblioteca `pandas`. En base de las métricas y como se van a clasificar los datos, nuestro conjunto tiene los siguientes campos

A continuación, se realizan los siguientes pasos para preparar los datos:

- **Manejo de valores nulos e infinitos:** Se utiliza la función `replace` de `pandas` para sustituir valores infinitos (`inf` y `-inf`) por `NaN`. Posteriormente, mediante la clase `SimpleImputer` de `sklearn.impute`, los valores faltantes se imputan utilizando la media de cada columna, asegurando así la continuidad y estabilidad del conjunto de datos.
- **Selección y ordenamiento de características:** Una vez limpios los datos, se toman las columnas relevantes que representan las métricas más significativas, siendo este el caso de `utilization_pct`, `retraso`, `jitter` y `throughput_jitter`.
- **Normalización de atributos:** Se normalizan los datos utilizando `MinMaxScaler` para mantener todas las variables dentro de un rango similar, para que no haya una discrepancia muy notoria en los datos.
- **Almacenamiento:** Finalmente, las métricas procesadas son almacenadas en una base de datos MySQL, lo cual permite su consulta para visualización y análisis históricos por medio de herramientas como Grafana para generar gráficas de las métricas de red.

4.5 Conmutación de rutas

Por último, el modelo genera como salida, una predicción: 0 si en uno de los puertos se identifica tráfico normal sin posible congestión, y 1 si el modelo encuentra instancias que puedan considerarse congestión. Al tener el modelo entrenado y programado con las métricas de retardo, jitter, etc. Podrá tomar acciones de conmutación del tráfico para mejorar los niveles de calidad de servicio en la red.

4. METODOLOGÍA Y DESARROLLO

Capítulo 5

Pruebas y Resultados

5.1 Funcionamiento

Con el sistema en funcionamiento se realizan las siguientes pruebas para comprobar su eficacia y rendimiento en diferentes escenarios de tráfico.

5.1.1 Generación de Tráfico

1. Como primera instancia se genera el tráfico mediante una herramienta llamada Iperf la cual es utilizada para hacer pruebas en redes informáticas. El funcionamiento habitual de esta herramienta es crear flujos de datos TCP o UDP. Esta herramienta establece un host como servidor que escucha y otro el cual genera el tráfico especificando parámetros como el tipo de tráfico, el ancho de banda, el tiempo, etc.

```
root@HOST-1:/# iperf3 -c 192.168.40.40 -u -b 100M -t 60
Connecting to host 192.168.40.40, port 5201
[ 5] local 192.168.40.3 port 48285 connected to 192.168.40.40 port 5201
[ ID] Interval Transfer Bitrate Total Datagrams
[ 5]  0.00-1.00  sec  11.9 MBytes  99.9 Mbits/sec  8625
[ 5]  1.00-2.00  sec  11.9 MBytes  100 Mbits/sec  8633
[ 5]  2.00-3.00  sec  11.9 MBytes  100 Mbits/sec  8632
[ 5]  3.00-4.00  sec  11.9 MBytes  100 Mbits/sec  8633
[ 5]  4.00-5.00  sec  11.9 MBytes  100 Mbits/sec  8633
[ 5]  5.00-6.00  sec  11.9 MBytes  100 Mbits/sec  8632
[ 5]  6.00-7.00  sec  11.9 MBytes  100 Mbits/sec  8633
[ 5]  7.00-8.00  sec  11.9 MBytes  100 Mbits/sec  8632
[ 5]  8.00-9.00  sec  11.9 MBytes  100 Mbits/sec  8633
[ 5]  9.00-10.00 sec  11.9 MBytes  100 Mbits/sec  8633
[ 5] 10.00-11.00 sec  11.9 MBytes  100 Mbits/sec  8632
[ 5] 11.00-12.00 sec  11.9 MBytes  100 Mbits/sec  8633
[ 5] 12.00-13.00 sec  11.9 MBytes  100 Mbits/sec  8632
[ 5] 13.00-14.00 sec  11.9 MBytes  100 Mbits/sec  8633
[ 5] 14.00-15.00 sec  11.9 MBytes  100 Mbits/sec  8632
[ 5] 15.00-16.00 sec  11.9 MBytes  100 Mbits/sec  8633
[ 5] 16.00-17.00 sec  11.9 MBytes  100 Mbits/sec  8633
[ 5] 17.00-18.00 sec  11.9 MBytes  100 Mbits/sec  8632
[ 5] 18.00-19.00 sec  11.9 MBytes  100 Mbits/sec  8633
```

Figura 5.1: Generación de tráfico

5. PRUEBAS Y RESULTADOS

2. El servidor permanece en escucha a través del puerto 5201 y en la dirección 192.168.40.40.

```
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^'.
HOST-2 console is now available... Press RETURN to get started.
ip: RTNETLINK answers: Network is unreachable
root@HOST-2:/# iperf3 -s
-----
Server listening on 5201
-----
Accepted connection from 192.168.40.3, port 42398
[ ID] Interval      Transfer     Bitrate    Jitter  Lost/Total Datagrams
[  5]  0.00-1.00   sec   22.7 MBytes   191 Mbits/sec  0.096 ms  0/16472 (0%)
[  5]  1.00-2.00   sec   23.8 MBytes   200 Mbits/sec  0.053 ms  0/17267 (0%)
[  5]  2.00-3.00   sec   23.8 MBytes   200 Mbits/sec  0.007 ms  0/17265 (0%)
[  5]  3.00-4.00   sec   23.8 MBytes   200 Mbits/sec  0.047 ms  0/17271 (0%)
[  5]  4.00-5.00   sec   23.8 MBytes   200 Mbits/sec  0.046 ms  0/17259 (0%)
[  5]  5.00-6.00   sec   23.9 MBytes   200 Mbits/sec  0.057 ms  0/17282 (0%)
[  5]  6.00-7.00   sec   23.8 MBytes   200 Mbits/sec  0.019 ms  0/17248 (0%)
[  5]  7.00-8.00   sec   23.9 MBytes   200 Mbits/sec  0.007 ms  0/17280 (0%)
[  5]  8.00-9.00   sec   23.8 MBytes   200 Mbits/sec  0.005 ms  0/17268 (0%)
[  5]  9.00-10.00  sec   23.8 MBytes   200 Mbits/sec  0.033 ms  0/17263 (0%)
[  5] 10.00-11.00  sec   23.8 MBytes   200 Mbits/sec  0.008 ms  0/17250 (0%)
[  5] 11.00-12.00  sec   23.9 MBytes   200 Mbits/sec  0.007 ms  0/17283 (0%)
[  5] 12.00-13.00  sec   23.8 MBytes   200 Mbits/sec  0.062 ms  0/17265 (0%)
[  5] 13.00-14.00  sec   23.8 MBytes   200 Mbits/sec  0.009 ms  0/17265 (0%)
[  5] 14.00-15.00  sec   23.8 MBytes   200 Mbits/sec  0.007 ms  0/17265 (0%)
[  5] 15.00-16.00  sec   23.8 MBytes   200 Mbits/sec  0.016 ms  0/17248 (0%)
[  5] 16.00-17.00  sec   23.8 MBytes   200 Mbits/sec  0.061 ms  0/17267 (0%)
[  5] 17.00-18.00  sec   23.9 MBytes   200 Mbits/sec  0.048 ms  0/17281 (0%)
[  5] 18.00-19.00  sec   23.8 MBytes   200 Mbits/sec  0.005 ms  0/17265 (0%)
[  5] 19.00-20.00  sec   23.8 MBytes   200 Mbits/sec  0.052 ms  0/17265 (0%)
[  5] 20.00-21.00  sec   23.8 MBytes   200 Mbits/sec  0.050 ms  0/17266 (0%)
[  5] 21.00-22.00  sec   23.8 MBytes   200 Mbits/sec  0.008 ms  0/17264 (0%)
[  5] 22.00-23.00  sec   23.8 MBytes   200 Mbits/sec  0.051 ms  0/17265 (0%)
[  5] 23.00-24.00  sec   23.8 MBytes   200 Mbits/sec  0.007 ms  0/17249 (0%)
[  5] 24.00-25.00  sec   23.8 MBytes   200 Mbits/sec  0.051 ms  0/17265 (0%)
[  5] 25.00-26.00  sec   23.9 MBytes   200 Mbits/sec  0.020 ms  0/17282 (0%)
[  5] 26.00-27.00  sec   23.8 MBytes   200 Mbits/sec  0.013 ms  0/17248 (0%)
[  5] 27.00-28.00  sec   23.8 MBytes   200 Mbits/sec  0.021 ms  0/17265 (0%)
[  5] 28.00-29.00  sec   23.8 MBytes   200 Mbits/sec  0.012 ms  0/17266 (0%)
[  5] 29.00-30.00  sec   23.9 MBytes   200 Mbits/sec  0.032 ms  0/17279 (0%)
[  5] 30.00-30.04  sec   1.05 MBytes   198 Mbits/sec  0.080 ms  0/763 (0%)
-----
[ ID] Interval      Transfer     Bitrate    Jitter  Lost/Total Datagrams
[  5]  0.00-30.04  sec   715 MBytes   200 Mbits/sec  0.080 ms  0/517941 (0%)  receiver
```

Figura 5.2: Host Servidor Iperf

3. Ejecución del programa del sistema de gestión inteligente de la red.

```
chrisvb@chrisvb-VirtualBox:/media/sf_DOCS-RYU$ ryu-manager metrics.py final_switch.py
loading app metrics.py
loading app final_switch.py
loading app ryu.controller.ofp_handler
loading app ryu.topology.switches
loading app ryu.controller.ofp_handler
instantiating app metrics.py of CongestionMonitoring
package: mysql.connector.plugins
plugin_name: caching_sha2_password
AUTHENTICATION_PLUGIN_CLASS: MySQLCachingSHA2PasswordAuthPlugin
package: mysql.connector.plugins
plugin_name: mysql_native_password
AUTHENTICATION_PLUGIN_CLASS: MySQLNativePasswordAuthPlugin
Modelo Cargado Correctamente.
instantiating app final_switch.py of SimpleSwitch13
🔗 SimpleSwitch13 iniciado. Presiona Ctrl+C para detener.
instantiating app ryu.controller.ofp_handler of OFPHandler
instantiating app ryu.topology.switches of Switches
✓ Switch normal 165372387250247 configurado correctamente.
✓ Switch Principal 3117009132097 configurado correctamente.
🚫 Bloqueando Switch Alternativo 90802005919310
🚫 Bloqueando Switch 90802005919310 (descartando todo el tráfico)
🚫 Eliminando reglas en Switch 90802005919310
✓ Switch 90802005919310 bloqueado exitosamente
✓ Switch normal 11065478421580 configurado correctamente.
```

Figura 5.3: Ejecución de los módulos de la aplicación tanto de monitoreo como de gestión.

4. Se instalan reglas de flujo en la ruta principal basados en el protocolo ARP generando las tablas MAC y se bloquea todo tráfico por la ruta alternativa.

```
OVS-PRINCIPAL:$ ovs-ofctl -O OpenFlow13 dump-flows br0
cookie=0x0, duration=15.226s, table=0, n_packets=30, n_bytes=1800, priority=0 actions=CONTROLLER:655
35
OVS-PRINCIPAL:$ ovs-ofctl -O OpenFlow13 dump-flows br0
cookie=0x0, duration=5.444s, table=0, n_packets=7, n_bytes=446, priority=1,in_port=eth2,dl_src=02:42
:2b:b2:a0:00,dl_dst=02:42:75:c5:b8:00 actions=output:eth1
cookie=0x0, duration=5.432s, table=0, n_packets=42413, n_bytes=63184119, priority=1,in_port=eth1,dl_
src=02:42:75:c5:b8:00,dl_dst=02:42:2b:b2:a0:00 actions=output:eth2
cookie=0x0, duration=29.669s, table=0, n_packets=60, n_bytes=3628, priority=0 actions=CONTROLLER:655
35
OVS-PRINCIPAL:$
```

Figura 5.4: Flujos por ruta principal

5. PRUEBAS Y RESULTADOS

```
OVS-ALTERNO:/$ ovs-ofctl -O OpenFlow13 dump-flows br0
cookie=0x0, duration=20.072s, table=0, n_packets=0, n_bytes=0, priority=32769,dl_type=0x88cc actions
=CONTROLLER:65509
cookie=0x0, duration=20.072s, table=0, n_packets=40, n_bytes=2400, actions=drop
OVS-ALTERNO:/$
```

Figura 5.5: Flujos por ruta alterna

5. Se hace la consulta de métricas de red, para después realizar el preprocesamiento de las mismas. Se ejecuta el modelo de Machine Learning, obteniendo dos posibles clasificaciones: 0 sin congestión (tráfico normal) y 1 congestión.

5.1 Funcionamiento

```

Actividades Terminal ▾ 2 de may 00:03
chrsvb@chrsvb-VirtualBox:/media/sf_DOCS-RYU

|Delay: 0.1235661506652832|Jitter: 0.01353824615478516
|[Switch 165372387250247] | Enlace: (165372387250247, 1) | Predicción: 0 | Historial: 0
|Throughput: 213.46Mbps|Utilization: 34.99%|Avg_packet_tx:60.0|Avg_packet_rx:1490.0|throughput_jitter:10.916053617447362
|Delay: 0.1229548454284668|Jitter: 0.028619766235351562
|[Switch 165372387250247] | Enlace: (165372387250247, 2) | Predicción: 0 | Historial: 0
|Throughput: 0.00Mbps|Utilization: 0.00%|Avg_packet_tx:60.0|Avg_packet_rx:60.0|throughput_jitter:0.0002148251210081005
|Delay: 0.12313675880432129|Jitter: 0.028813838958740234
|[Switch 11065478421580] | Enlace: (11065478421580, 1) | Predicción: 0 | Historial: 0
|Throughput: 193.91Mbps|Utilization: 31.79%|Avg_packet_tx:1489.9171782694314|Avg_packet_rx:0|throughput_jitter:11.715372814847141
|Delay: 0.0782480239868164|Jitter: 0.015845775604248847
|[Switch 11065478421580] | Enlace: (11065478421580, 2) | Predicción: 0 | Historial: 0
|Throughput: 0.00Mbps|Utilization: 0.00%|Avg_packet_tx:60.0|Avg_packet_rx:60.0|throughput_jitter:5.5868612035370736e-05
|Delay: 0.07797455787658691|Jitter: 0.016224145889282227
|[Switch 11065478421580] | Enlace: (11065478421580, 1) | Predicción: 0 | Historial: 0
|Throughput: 226.48Mbps|Utilization: 37.13%|Avg_packet_tx:1489.9186390532545|Avg_packet_rx:60.0|throughput_jitter:13.357460000649018
|Delay: 0.137120246887220703|Jitter: 0.058872222906390625
|[Switch 11065478421580] | Enlace: (11065478421580, 2) | Predicción: 0 | Historial: 0
|Throughput: 0.00Mbps|Utilization: 0.00%|Avg_packet_tx:60.0|Avg_packet_rx:60.0|throughput_jitter:6.109628867116186e-05
|Delay: 0.1372437477118164|Jitter: 0.05926918983459473
|[Switch 3117009132097] | Enlace: (3117009132097, 1) | Predicción: 0 | Historial: 0
|Throughput: 197.94Mbps|Utilization: 32.45%|Avg_packet_tx:60.0|Avg_packet_rx:1489.837287364169|throughput_jitter:9.425959455820859
|Delay: 0.1374971866607666|Jitter: 0.014040231704711914
|[Switch 3117009132097] | Enlace: (3117009132097, 2) | Predicción: 0 | Historial: 0
|Throughput: 198.13Mbps|Utilization: 32.48%|Avg_packet_tx:1489.9187176718012|Avg_packet_rx:60.0|throughput_jitter:9.440086386993643
|Delay: 0.13776373863220215|Jitter: 0.014197587966918945
|[Switch 165372387250247] | Enlace: (165372387250247, 1) | Predicción: 0 | Historial: 0
|Throughput: 196.80Mbps|Utilization: 32.26%|Avg_packet_tx:60.0|Avg_packet_rx:1489.9187176718012|throughput_jitter:10.916160247060635
|Delay: 0.09503817558288574|Jitter: 0.027916669845581055
|[Switch 165372387250247] | Enlace: (165372387250247, 2) | Predicción: 0 | Historial: 0
|Throughput: 0.00Mbps|Utilization: 0.00%|Avg_packet_tx:60.0|Avg_packet_rx:60.0|throughput_jitter:0.00021488562948290523
|Delay: 0.0951461791921875|Jitter: 0.02799657960510254
|[Switch 11065478421580] | Enlace: (11065478421580, 1) | Predicción: 0 | Historial: 0
|Throughput: 205.59Mbps|Utilization: 33.70%|Avg_packet_tx:1489.9172645221013|Avg_packet_rx:60.0|throughput_jitter:13.11162392813631
|Delay: 0.041525840759277344|Jitter: 0.09559440612792969

```

Figura 5.6: Consulta de métricas de red

- Las métricas son guardadas en una base de datos, las cuales son usadas para la generación de gráficas y tener un registro si se desea hacer un posible análisis.



Figura 5.7: Gráficas en tiempo real

5. PRUEBAS Y RESULTADOS

7. En caso de sobrepasar el límite de congestión para evitar el posible fallo del switch, el sistema cambia la ruta cambiando los flujos configurados en el switch.

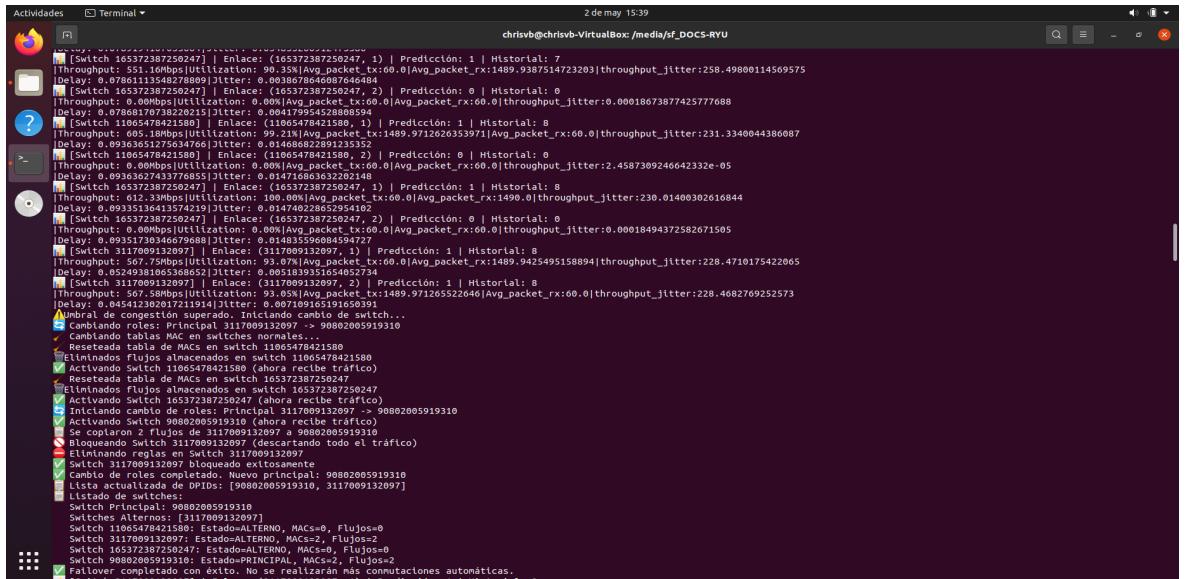


Figura 5.8: Proceso de conmutación

Topologías de pruebas

La topología mostrada en la figura 5.9, fue usada para la comprobación de la efectividad del sistema, estableciendo dos switches que tendrán 2 roles distintos, uno será la ruta principal y otro quedará en reposo hasta que sea activado por el sistema. Cada uno de los switch están conectados al controlador, el sistema identifica de manera autónoma si es necesario redirigir el tráfico a partir de la clasificación de congestión generadas por el modelo de Machine Learning.

5.1 Funcionamiento

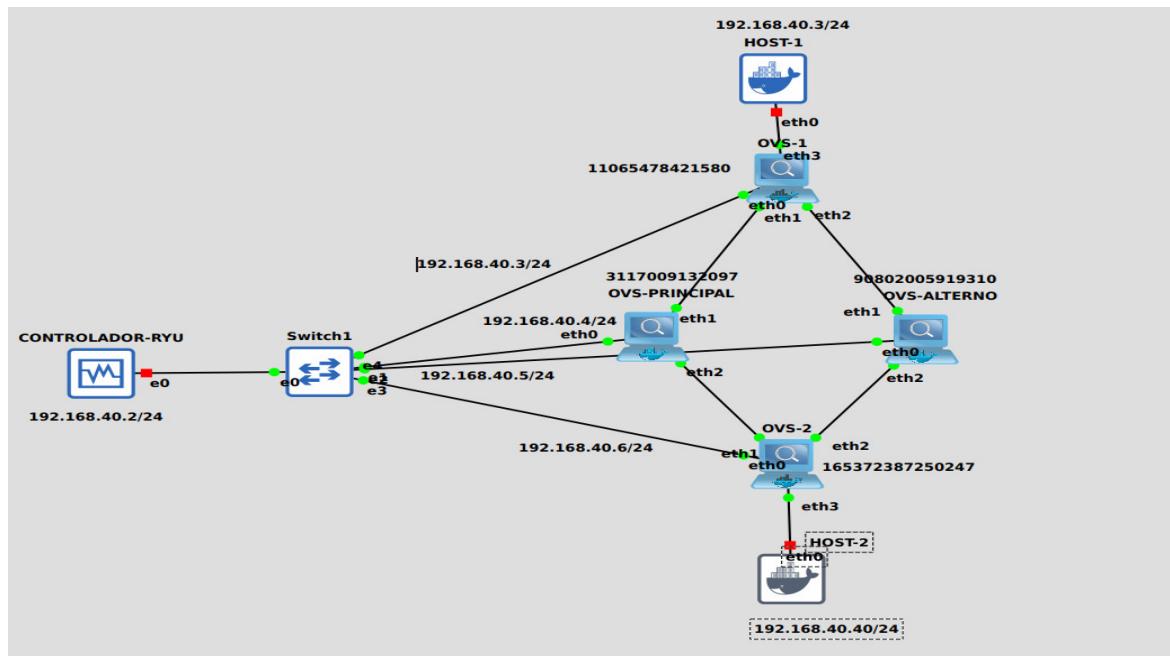


Figura 5.9: Topología de pruebas

A grandes rasgos esta ultima topología comprueba la capacidad de la aplicación para el monitoreo y gestión automática de la red. Esta aplicación puede ser aplicada para la topología de la red backbone del IPN la cual se muestra en la figura 5.10.

5. PRUEBAS Y RESULTADOS

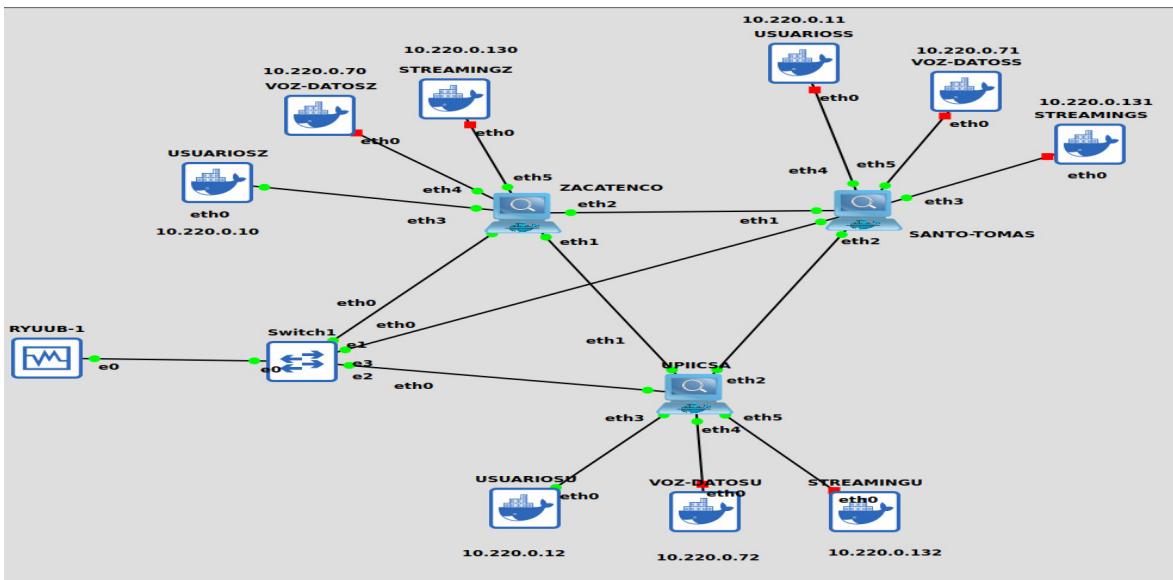


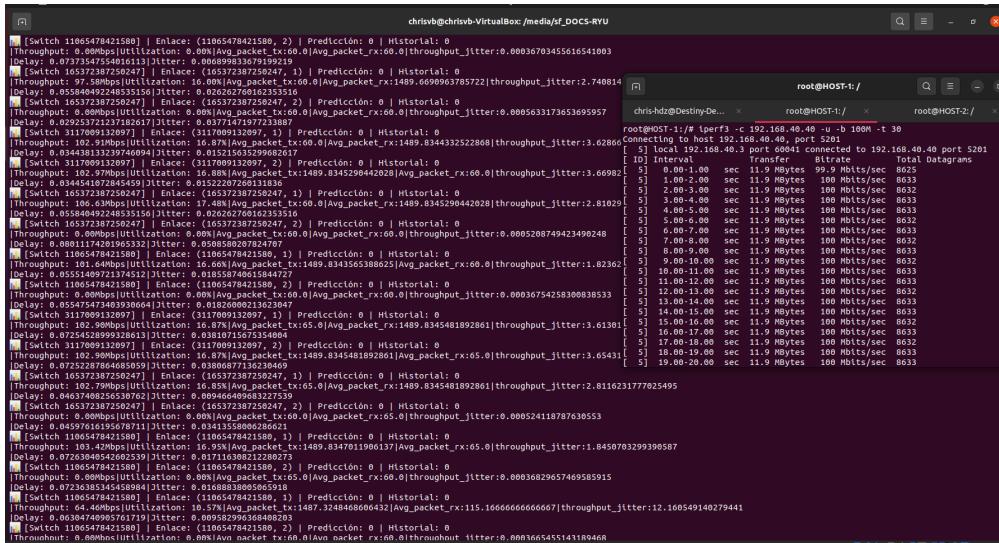
Figura 5.10: Topología Red BackBone IPN

Lo que se hace para esta topología es declarar rutas estáticas para cada tipo de tráfico (usuarios, voz/datos y streaming). Estos flujos son definidos a través de reglas que están compuestas por puertos de entrada/salida y direcciones IP. Las reglas consideran tanto rutas principales como las alternativas. La commutación de estas rutas esta basada en el monitoreo en tiempo real por parte del modulo encargado de ello, el cual al igual que en la topología anterior se hace la captura métricas clave presentes en los puertos de los switch. Si se detecta que un enlace específico supera el umbral predefinido de congestión, el sistema activa el proceso de commutación de rutas basado en una prioridad del tipo de tráfico para esos 3 grupos. Por lo que si los 3 grupos están presentes en la ruta principal y la congestionan, se van cambiando a la ruta alterna uno por uno, con el objetivo de reducir la carga de la ruta principal.

5.2 Resultados del experimento

Como objetivo de este trabajo se busco realizar un sistema de gestión y monitoreo para redes SDN. Para la comprobación de que el sistema detecte de manera correcta la congestión se realizo un experimento en el cual, se probaron diferentes anchos de banda y observar la respuesta del sistema. A continuación, se presentan las métricas capturadas para cada uno de los escenarios de ancho de banda:

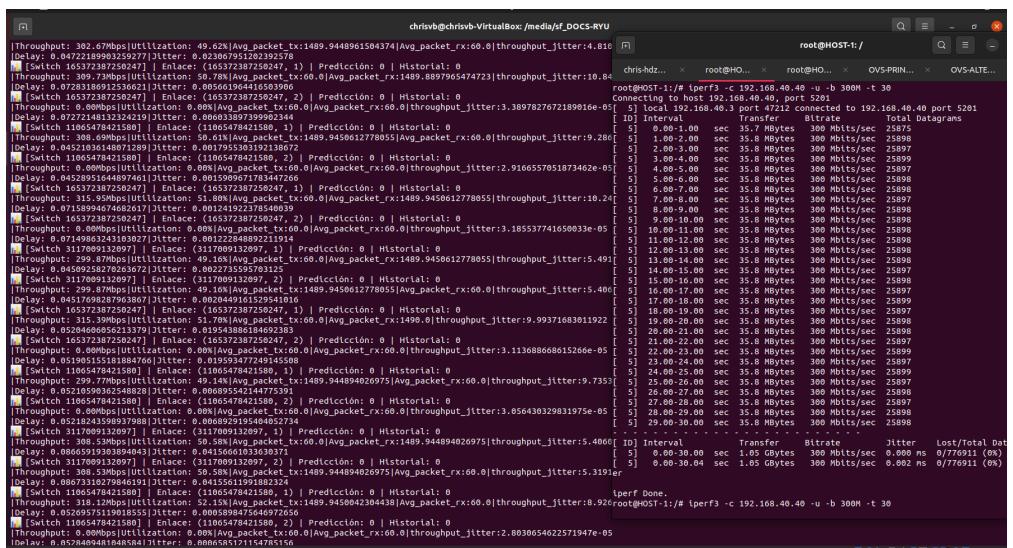
5.2 Resultados del experimento



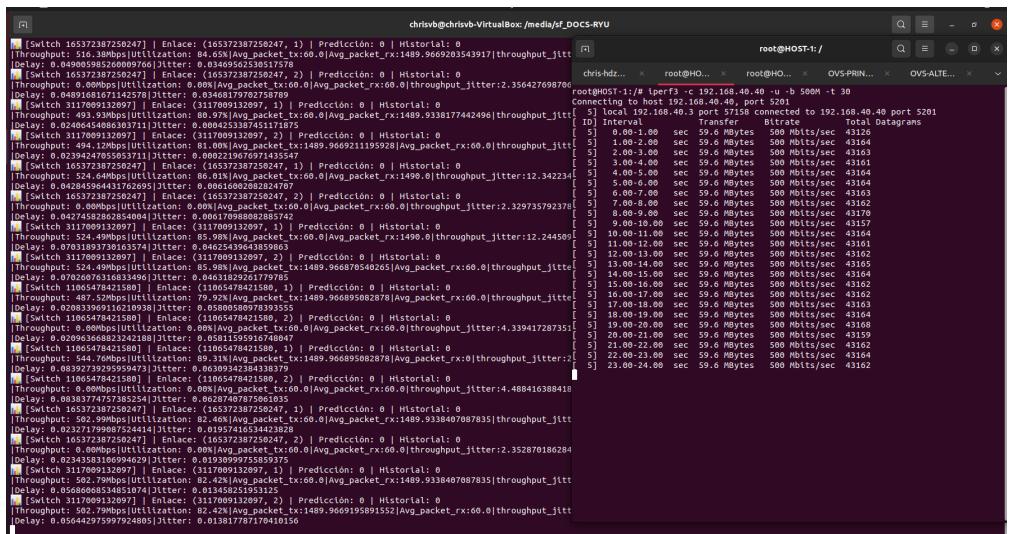
(a) Ancho de banda: 100 Mbps

Figura 5.11: Captura de métricas de red en distintos niveles de ancho de banda (I)

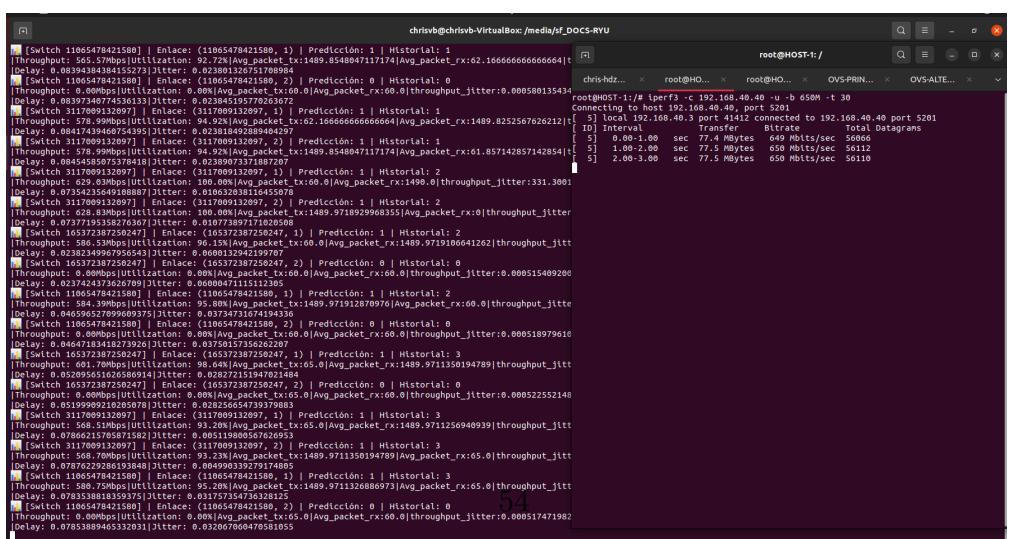
5. PRUEBAS Y RESULTADOS



(b) Ancho de banda: 300 Mbps



(c) Ancho de banda: 500 Mbps



(d) Ancho de banda: 650 Mbps (congestión detectada)

Figura 5.11: Captura de métricas de red en distintos niveles de ancho de banda (II)

5.2 Resultados del experimento

En las Figuras 5.11a a 5.11c, se observa que la red opera de manera estable sin indicios de congestión, es decir, se presenta tráfico normal que la red puede soportar. Sin embargo, al alcanzar un ancho de banda de 650 Mbps (Figura 5.11d), se detecta un aumento considerable en las métricas asociadas a congestión y, por lo tanto, la clasificación del modelo de Machine Learning los clasifica como congestión. Por ello, se establece un historial de cuántas veces se puede mantener así el switch antes de que sea necesario activar el mecanismo de conmutación de rutas.

A continuación, se muestran los flujos de tráfico antes y después de la conmutación tanto en el switch principal como en el switch alterno:

```
OVS-PRINCIPAL:/$ ovs-ofctl -O OpenFlow13 dump-flows br0
cookie=0x0, duration=15.226s, table=0, n_packets=30, n_bytes=1800, priority=0 actions=CONTROLLER:655
35
OVS-PRINCIPAL:/$ ovs-ofctl -O OpenFlow13 dump-flows br0
cookie=0x0, duration=5.444s, table=0, n_packets=7, n_bytes=446, priority=1,in_port=eth2,dl_src=02:42
:2b:b2:a0:00,dl_dst=02:42:75:c5:b8:00 actions=output:eth1
cookie=0x0, duration=5.432s, table=0, n_packets=42413, n_bytes=63184119, priority=1,in_port=eth1,dl_
src=02:42:75:c5:b8:00,dl_dst=02:42:2b:b2:a0:00 actions=output:eth2
cookie=0x0, duration=29.669s, table=0, n_packets=60, n_bytes=3628, priority=0 actions=CONTROLLER:655
35
OVS-PRINCIPAL:/$
```

(a) Flujos antes y después de la conmutación (switch principal)

```
OVS-ALTERNO-90802005919310:/$ ovs-ofctl -O OpenFlow13 dump-flows br0
cookie=0x0, duration=15.508s, table=0, n_packets=0, n_bytes=0, priority=32769,dl_type=0x80cc actions=CONTROLLER:65509
cookie=0x0, duration=15.508s, table=0, n_packets=32, n_bytes=1884, actions=drop
OVS-ALTERNO-90802005919310:/$ ovs-ofctl -O OpenFlow13 dump-flows br0
cookie=0x0, duration=11.045s, table=0, n_packets=0, n_bytes=0, priority=1,in_port=eth2,dl_src=02:42:2b:b2:a0:00,dl_dst=02:42:75:c5:b8:00 actions=output:eth1
cookie=0x0, duration=11.045s, table=0, n_packets=25848, n_bytes=38513520, priority=1,in_port=eth1,dl_src=02:42:75:c5:b8:00,dl_dst=02:42:2b:b2:a0:00 actions=output:eth2
cookie=0x0, duration=11.045s, table=0, n_packets=22, n_bytes=1320, priority=0 actions=CONTROLLER:65535
OVS-ALTERNO-90802005919310:/$
```

(b) Flujos antes y después de la conmutación (switch alterno)

Figura 5.12: Conmutación de rutas al detectarse congestión en la red

En las Figuras 5.12a y 5.12b se aprecia el flujo de tráfico antes y después de la conmutación efectuada, donde la mayoría de los paquetes transitan por la ruta principal. Tras la detección de congestión persistente y la activación de la conmutación, el sistema redirige el tráfico hacia la ruta alterna disponible, lo cual permite que el switch que se encontraba en la ruta principal sea revisado y quede disponible en caso de que el alterno falle.

5. PRUEBAS Y RESULTADOS

Capítulo 6

Conclusiones

6.1 Conclusiones Finales

El presente trabajo se presento una propuesta de aplicación para aplicaciones de monitoreo de nueva generación, vimos que tan viable y efectivo es el hecho de integrar técnicas de Machine Learning en un entorno compuesto enteramente por redes definidas por software, con el objetivo de optimizar la calidad de servicio. Al este proyecto ser un prototipo la topología para ser probado es sencilla contando con dos 2 rutas redundantes, pero aplicando el sistema el cual esta en un monitoreo constante de métricas como el retraso,jitter,throughput y su variación vimos como es posible identificar congestión es decir cuellos de botella en la interfaces de los switch para poder tomar decisiones proactivas como cambiar la ruta para que la red cuente con tolerancia a fallos. Aunque esa parte del proyecto esta hecha para que trabaje de manera autónoma reduciendo la intervención manual y teniendo un tiempo de respuesta ante problemáticas más rápido, también la característica del sistema que permite almacenar y analizar métricas en tiempo real para que pueda ser analizado el comportamiento de la red incluso históricamente para que el administrador de red pueda tomar decisiones. Pero con todo esto podemos ver el potencial que podría tener en entornos mas grandes con topologías mas complejas.Durante este trabajo vimos la facilidad de generar aplicaciones para redes SDN para diferentes propósitos pudiendo tener un preámbulo para pasar a redes auto adaptativas que puedan reaccionar antes problemáticas de mejor manera en comparativa a métodos tradicionales de gestión de red. Se determinó también que el aprendizaje supervisado ofrece ventajas considerables para identificar cuellos de botella y comportamientos anómalos en la red que afectan a la calidad de servicio, aparte de poder

6. CONCLUSIONES

determinar cual es la mejor arquitectura de modelo de Machine Learning para este caso. Vimos que al no tener un data set en linea disponible que sirviera para métricas de red relevantes para caracterizar congestión fue difícil el tener y generar uno debido a recursos computacionales, suficientes instancias para generalizar casos de congestión o incluso el no tener acceso a métricas relevantes como la pérdida de paquetes debido al entorno virtualizado, sin embargo a pesar de eso se logró etiquetar y generar uno con las instancias necesarias para el entrenamiento del modelo. Finalmente, se concluyó que con este desarrollo de implementación de un sistema de detección de problemáticas de congestión, apoyado de redes SDN, Machine Learning y el uso del lenguaje de programación python demuestran el potencial que tienen estas tecnologías juntas para las comunicaciones en redes de ordenadores para que puedan ser adaptados ante constantes cambios y evolución.

6.2 Trabajo Futuro

Finalmente el trabajo abre la puerta a nuevos desafíos que podrían abordarse en investigaciones futuras como lo son:

- **Uso de datos de tráfico reales:** Tendría una mejor adaptación a entorno reales ya que sería métricas medidas reales y adaptadas a la red en particular
- **Ampliación del sistema de monitoreo:** Tendría la implicación de incluir métricas adicionales como el uso de CPU de los switch, estados de las colas de tráfico de salida, consumo energético, etc.
- **Reportes de las amenazas:** Creación de una plataforma para la gestión interactiva de la red y más amigable con el usuario, de igual manera permitiendo ampliar las funcionalidades como monitoreo remoto, alertas, acceso a diferentes topologías, etc.
- **Implementación de Balanceo de carga:** Crear un algoritmo que permita el balanceo de carga y no solo se tenga un sistema en caso de desbordamiento si no que se trabaje en conjunto para mejorar el rendimiento de la red.

Apéndice A

Creación de hosts por medio de contenedores de Docker

Para facilitar la generación de tráfico y simulación de nodos dentro de la red SDN, se implementaron contenedores ligeros basados en imágenes personalizadas de Docker. Estos contenedores fueron cargados con herramientas de generación de tráfico con el objetivo de hacer pruebas. Dicha imagen fue posteriormente exportada y cargada como appliance en GNS3, permitiendo su uso como un host.

A.0.1 Construcción de la imagen Docker

1. Se utilizó como base la imagen oficial de Ubuntu 22.04(18).
2. Se evitó la interacción durante la instalación mediante la variable de entorno:

```
ENV DEBIAN_FRONTEND=noninteractive
```

3. Se instalaron herramientas esenciales de red:
 - **iputils-ping**: prueba de conectividad.
 - **iproute2**: configuración avanzada de red.
 - **net-tools**: herramientas clásicas de red.
 - **iperf3**: generación de tráfico.
 - **hping3**: envío de paquetes personalizados.
 - **tcpdump**: captura de paquetes.
 - **tshark**: análisis de tráfico.

A. CREACIÓN DE HOSTS POR MEDIO DE CONTENEDORES DE DOCKER

- **tcpreplay**: reenvío de capturas pcap.
4. Se configuraron parámetros del sistema para mejorar el rendimiento en la recepción de paquetes:

```
RUN echo "net.core.rmem_max=26214400"  
    >> /etc/sysctl.conf  
RUN echo "net.core.rmem_default=26214400"  
    >> /etc/sysctl.conf
```

5. Finalmente, se construyó la imagen con el siguiente comando:

```
docker build -t host-red .
```

```
GNU nano 6.2  
FROM ubuntu:22.04  
  
# Evitar preguntas interactivas en instalación  
ENV DEBIAN_FRONTEND=noninteractive  
  
# Instalar herramientas necesarias  
RUN apt update && apt install -y \  
    iputils-ping \  
    iproute2 \  
    net-tools \  
    iperf3 \  
    hping3 \  
    tcpdump \  
    tshark \  
    tcpreplay \  
    curl \  
    wget \  
    && rm -rf /var/lib/apt/lists/*  
  
# Configurar parámetros de red para aumentar buffers de recepción  
RUN echo "net.core.rmem_max=26214400" >> /etc/sysctl.conf  
RUN echo "net.core.rmem_default=26214400" >> /etc/sysctl.conf  
  
# Comando por defecto al iniciar el contenedor  
CMD ["/bin/bash"]
```

Figura A.1: Dockerfile utilizado para la creación de la imagen base

A.0.2 Carga de la imagen como appliance en GNS3

Una vez construida la imagen de Docker localmente, esta fue exportada y cargada como appliance dentro del simulador GNS3, permitiendo la creación de múltiples instancias de contenedores, simulando hosts.

A.0.3 Despliegue y conexión de contenedores

1. Se desplegaron varios contenedores desde la imagen cargada en GNS3, nombrándolos de acuerdo a su función.
2. Cada contenedor fue conectado a un switch virtual Open vSwitch y fue configurado con una dirección IP estática.
3. Estas instancias permitieron realizar pruebas de conectividad, congestión, generación de tráfico.

A. CREACIÓN DE HOSTS POR MEDIO DE CONTENEDORES DE DOCKER

Apéndice B

Instalación del controlador Ryu en Ubuntu

El controlador SDN utilizado en este proyecto es Ryu, la cual es una plataforma basada en Python que permite la implementación de aplicaciones de control de red de forma modular. A continuación se describe el procedimiento seguido para instalar Ryu en una máquina con sistema operativo Ubuntu Desktop 22.04 (18).

B.0.1 Preparación del entorno

1. Se utilizó una máquina virtual con Ubuntu 22.04 completamente actualizada:

```
sudo apt update && sudo apt upgrade -y
```

2. Se instaló Python 3:

```
sudo apt install -y python3 python3-pip  
python3-dev
```

3. También se instalaron dependencias adicionales necesarias para el funcionamiento del controlador:

```
sudo apt install -y libffi-dev libssl-dev  
libxml2-dev libxslt1-dev zlib1g-dev
```

B. INSTALACIÓN DEL CONTROLADOR RYU EN UBUNTU

B.0.2 Instalación de Ryu

1. Se utilizó ‘pip3‘ para instalar Ryu desde PyPI:

```
pip3 install ryu
```

2. Una vez instalado, se verificó que el comando ‘ryu-manager‘ estuviera disponible:

```
ryu-manager --version
```

3. Para la ejecución de cualquier aplicación se hace uso del siguiente comando, así de igual manera vemos su correcto funcionamiento:

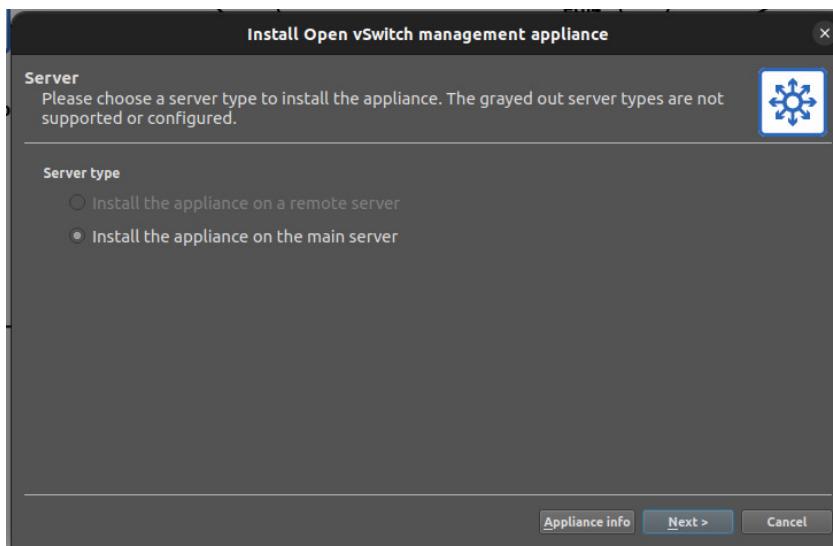
```
ryu-manager ryu.app.simple_switch_13
```

4. En este punto, el controlador quedó a la espera de switches que hablen el protocolo OpenFlow 1.3.

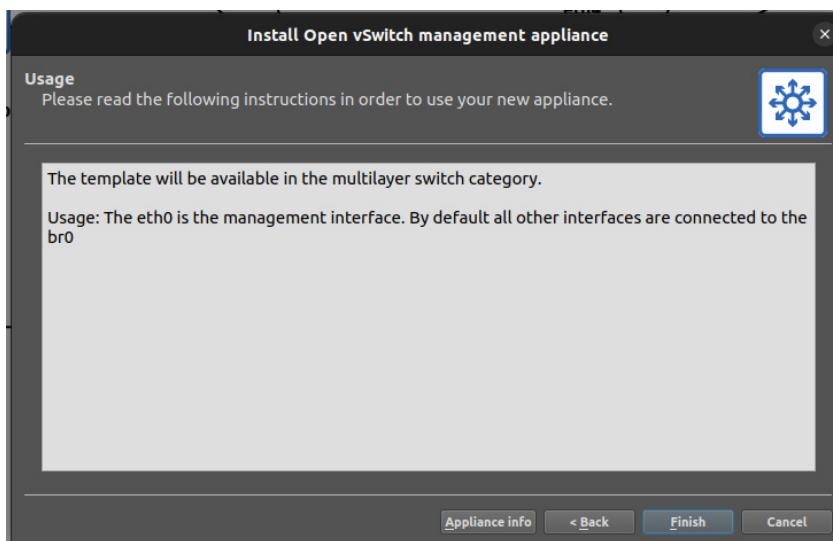
B.0.3 Integración de switches Open vSwitch (OVS) en GNS3

Para emular una red definida por software en GNS3, se utilizaron dispositivos virtuales basados en Open vSwitch (OVS). A continuación se detalla el proceso de instalación y configuración de los switches gestionables.

1. Se descargó el appliance oficial de **Open vSwitch with Management** desde el marketplace de GNS3 o directamente desde el sitio web oficial(17).
2. En GNS3, se procedió a importar el archivo ‘.gns3a‘ correspondiente al appliance mediante el menú: *File > Import Appliance*.



(a) Selección del servidor para la instalación del appliance. Se elige el servidor principal (local) para gestionar el switch virtual.



(b) Confirmación de que la plantilla estará disponible como switch multicapa. Se indica que la interfaz `eth0` será usada para administración.

Figura B.1: Proceso paso a paso para la instalación del appliance Open vSwitch en GNS3

3. Una vez instalado el appliance, se agregó a la topología de red y se conectaron las interfaces deseadas a otros dispositivos (hosts o switches).
4. Finalmente, se estableció la conexión entre los switches y el controlador

B. INSTALACIÓN DEL CONTROLADOR RYU EN UBUNTU

Ryu configurando manualmente la dirección IP del controlador SDN:

```
ovs-vsctl set-controller br0  
tcp:192.168.40.2:6633
```

5. El enlace fue verificado usando el siguiente comando:

```
ovs-vsctl show
```

```
ZACATENCO:/ $ ovs-vsctl show  
c137cb27-c59a-46ca-8941-29c2c2e5d6d1  
  Bridge br0  
    Controller "tcp:192.168.40.2:6633"  
    fail_mode: secure  
    datapath_type: netdev  
    Port br0  
      Interface br0  
        type: internal  
    Port eth7  
      Interface eth7  
    Port eth6  
      Interface eth6  
    Port eth5
```

Figura B.2: Switch Open vSwitch con el controlador configurado

Referencias

- [1] Matt. How Much Data Is Generated Per Day, 2024. iii, 2
- [2] iii, 14, 15
- [3] K-Nearest Neighbors in Machine Learning. iii, 21
- [4] Butler, Brandon. IDC MarketScape: Worldwide SD-WAN Infrastructure 2023 Vendor Assessment. 3
- [5] ¿Qué son las redes definidas por software (SDN)? | IBM, 2024. 7
- [6] Feamster, N.; Rexford, J.; Zegura, E. The road to SDN: an intellectual history of programmable networks. *ACM SIGCOMM Computer Communication Review* **2014**, 44, 87–98. doi:black10.1145/2602204.2602219. 7
- [7] Espinosa, C. The Evolution of SDN: Redefining Network Management and Security - Blue Goat Cyber, 2024. 7
- [8] Facultad de Informática-Universidad Complutense de Madrid. 7
- [9] Arista's WAN Routing System targets routing use cases such as SD-WANs – IEEE ComSoc Technology Blog. 8
- [10] Rohit, K. AI-Powered Network Optimization: Transforming Operations, 2024. 8
- [11] Blanco Pérez, R. Avanzando hacia una red auto-adaptativa: simulación de redes definidas por software (SDN) mediante el simulador GNS3 **2019**. 9, 12
- [12] Guija Alcaraz, D. Gestió i control de qualitat de servei a xarxes SDN en temps real. Bachelor thesis, Universitat Politècnica de Catalunya, 2014. 9, 12

REFERENCIAS

- [13] Rahman, S.; Ahmed, T.; Huynh, M.; Tornatore, M.; Mukherjee, B. Auto-Scaling Network Resources using Machine Learning to Improve QoS and Reduce Cost, 2019. arXiv:1808.02975, doi:black10.48550/arXiv.1808.02975. 12
- [14] Bibak, B. An SDN-based traffic load prediction using machine learning. Master's thesis, École de technologie supérieure, 2023. 12
- [15] Algoritmos de aprendizaje automático | Microsoft Azure. 16
- [16] GNS3 Team. GNS3 0.4.1 - Documentación (Español), 2012. Accedido: 2025-05-04. 23
- [17] Open vSwitch Appliance for GNS3. 31, 64
- [18] Ubuntu 22.04.4 LTS (Jammy Jellyfish) Release. 31, 59, 63
- [19] ¿Qué son las redes definidas por software (SDN)? | IBM, 2024.
- [20] Facultad de Informática.
- [21] Software-Defined Networking (SDN) Definition.
- [22] Accelerating Your Growth from R&D to ROI, 2015.
- [23] Servicios de informática en la nube | Microsoft Azure.
- [24] Consultores IA - Consultores Inteligencia Artificial.
- [25] ¿Qué es Random Forest? | IBM, 2021.
- [26] Qué son los árboles de decisión y para qué sirven, 2020.
- [27] Schott, M. Random Forest Algorithm for Machine Learning, 2020.
- [28] What is the K-Nearest Neighbors (KNN) Algorithm?
- [29] 12.1 - Logistic Regression | STAT 462.
- [30] Sperandei, S. Understanding logistic regression analysis. *Biochimia Medica* **2014**, 24, 12–18. doi:black10.11613/BM.2014.003.
- [31] Logistic Regression | Stata Data Analysis Examples.
- [32] ¿Qué es la calidad de servicio (QoS) en las redes?

REFERENCIAS

- [33] Quality of Service (QoS) Configuration Guide, Cisco IOS XE Everest 16.6.x (Catalyst 9400 Switches) - Configuring QoS [Support].
- [34] Ruipérez Cuesta, J. Seguridad en Redes definidas por software (SDN). Proyecto/Trabajo fin de carrera/grado, Universitat Politècnica de València, 2021.
- [35] Noboa Romo, J.J. Seguridad para redes IoT usando Machine Learning : implementación de un sistema de detección de amenazas utilizando redes definidas por software. **2023**.
- [36] La historia de la ciberseguridad | NordVPN, 2022.
- [37] ¿Qué es Random Forest? | IBM, 2021.
- [38] M Devendra Prasad, Prasanta Babu V, C.A. Machine Learning DDoS Detection Using Stochastic Gradient Boosting. *International Journal of Computer Sciences and Engineering* **2019**, 7, 157–166. doi:black<https://doi.org/10.26438/ijcse/v7i4.157166>.
- [39] ¿Qué es una API? - Explicación de interfaz de programación de aplicaciones - AWS.
- [40] Software-Defined Networking (SDN) Definition-OpenNetworking.