45

Fecha de presentación: diciembre, 2020
Fecha de aceptación: febrero, 2021
Fecha de publicación: marzo, 2021

REDES BAYESIANAS APLICADAS

A LA PREDICCIÓN DE ERRORES EN LAS REDES DEFINIDAS POR SOFTWARE

BAYESIAN NETWORKS APPLIED TO ERROR PREDICTION IN SOFTWARE DEFINED NETWORKS

Bryon Wladimir Oviedo Bayas[1]
E-mail: boviedo41@uteq.edu.ec
ORCID: https://orcid.org/0000-0002-5366-5917
Cristian Zambrano Vega[1]
E-mail: czambrano@uteq.edu.ec
ORCID: https://orcid.org/0000-0001-8568-8024
[1] Universidad Técnica Estatal de Quevedo. Ecuador.

Cita sugerida (APA, séptima edición)

Oviedo Bayas, B. W., & Zambrano Vega, C. (2021). Redes bayesianas aplicadas a la predicción de errores en las redes definidas por software. *Revista Universidad y Sociedad*, 13(2), 419-429.

RESUMEN

El presente artículo está enfocado al análisis de fallos detectados en las redes definidas por software (SDN) que son un conjunto de redes altamente eficiente, escalable, programable, con gran velocidad y capacidad de gestionar recursos de red. Se empezó la investigación planteando la identificación y selección de controladores y herramientas de emulación que se utilizan actualmente en SDN. Luego se propone diseñar e implementar un escenario para pruebas basado en métricas de evaluación. El sistema propuesto se basa en el aprendizaje de una red Bayesiana a partir de los datos extraídos y procesados de una SDN, buscando relaciones causales entre valores de los datos y estado de la red. Para emular el funcionamiento de una SDN real, se ha diseñado una simulación con varias redes libres de escala unidas, en las que se han inyectado varios tipos de tráfico. La red Bayesiana será utilizada posteriormente para diagnosticar nuevos fallos introducidos en la red, razonando con los datos extraídos de esta. Finalmente se analizan los resultados obtenidos con el controlador OpenDayLight, el protocolo OpenFlow y el emulador Mininet.

Palabras clave: Redes definidas por Software (SDN), red bayesiana, diagnóstico de fallos, Mininet.

ABSTRACT

This article is focused on the analysis of failures detected in software-defined networks (SDNs), which are a highly efficient, scalable, programmable set of networks with high speed and the ability to manage network resources. The research began with the identification and selection of drivers and emulation tools currently used in SDN. Then it is proposed to design and implement a test scenario based on evaluation metrics. The proposed system is based on learning a Bayesian network from the data extracted and processed from an SDN, looking for causal relationships between data values and the state of the network. To emulate the operation of a real SDN, a simulation has been designed with several linked scale-free networks, in which various types of traffic have been injected. The Bayesian network will be used later to diagnose new failures introduced into the network, reasoning with the data extracted from it. Finally, the results obtained with the OpenDayLight controller, the OpenFlow protocol and the Mininet emulator are analyzed.

Keywords: Software Defined Networks (SDN), Bayesian Network, Troubleshooting, Mininet.

INTRODUCCIÓN

Hoy en día existe un nivel alto de complejidad en los servicios de telecomunicaciones, las arquitecturas de red son complejas y no cumplen con las expectativas del usuario para mejorar los servicios, las industrias de las tecnologías de la información y las telecomunicaciones (TIC´S) se han visto en la necesidad de reevaluar los conceptos de las redes tradicionales.

En los últimos años las redes definidas por software se han convertido en uno de los temas más abordados en el ámbito de las (ICT) Tecnología de Información y Comunicaciones, y muchas de las organizaciones internacionales han dado su concepto, pero no se ha logrado establecer una definición exacta.

El fundamento de la arquitectura SDN según la ONF son *"las redes definidas por software se definen como una arquitectura de red dinámica, gestionable, adaptable, de costo eficiente. Lo cual la hace ideal para las altas demandas de ancho de banda y la naturaleza dinámica de las aplicaciones actuales. Esta arquitectura desacopla el control de la red y la funcionalidad de reenvío de información permitiendo que el control de la red pueda ser completamente programable logrando que las aplicaciones y servicios de red se abstraigan de la infraestructura de red subyacente".* (Pérez Tardío, 2018)

La implementación de un sistema de gestión flexible, utilizando características SDN puede agravar algunos problemas en la gestión de redes. En particular, la inyección de los frecuentes cambios en las reglas de redes hechas por los sistemas de software implica posibilidades constantes de las políticas de tráfico defectuosas que se introduzcan en la red. En los sistemas de legado, debido a su Stiff-Ness, las políticas de red tenían que ser diseñados a fondo y se introduce en cada dispositivo con cuidado; Por lo tanto, el riesgo de introducir configuraciones defectuosas fue severamente limitada (Bernayas de los Santos, 2018).

Dado que los cambios de política pueden ser ahora rápida y fácilmente introducidos, menos tiempo y esfuerzo se gasta en la gestión de la red: por lo tanto, la capacidad de recuperación de la SDN se debe mejorar. La seguridad en todos los tipos de red existentes incluyendo las redes definidas por software (SDN) son esenciales, ya que por este medio se abordan temas de protección, disponibilidad, integridad y la privacidad de la información que se transmite. También es recomendable entender que la seguridad en este tipo de redes, aunque ya tienen un buen enfoque, aún se encuentra en definición, ya que no se logra hacer convergencia en este tipo de redes a base de solo enfoques, a pesar de ser simple de configurar por lo cambiante de la red y además son efectiva para lograr asegurar su despliegue en cualquier momento y sitio (Barrera Pérez, et al., 2019).

Durante la fusión del plano de control centralizado, con el canal de control, actividad

encaminada en lograr el intercambio de información con los dispositivos de red, se generan inquietudes a nivel de seguridad las cuales se deben despejar analizando tanto el enfoque del atacante, y del controlador de red, esto debido a su relevancia en la arquitectura. Se debe además tener en cuenta la importancia del análisis a los diferentes protocolos, ya que las soluciones de seguridad para las redes SDN (Redes Definidas Por Software), suelen ser más un tipo de aspecto de aplicación que no dependen tanto del hardware (Vélez Mejías, 2018).

Cabe aclarar que todos estos conceptos y escenarios de red concretos de aplicación en la actualidad, requieren de mucha más investigación para poder comprenderlos desde el ámbito de la seguridad.

El diseño e implementación de un sistema de diagnóstico de fallos basado en el razonamiento bayesiano se presenta en este trabajo. Con el fin de probar el sistema de diagnóstico, una compleja red es simulada, la inyección de tráfico que se asemeja tráficos realistas presentes en las redes actuales, tales como P2P, chat, correo electrónico o la transmisión de vídeo. En esta red, algunas fallas se generan, con el fin de crear un "estado defectuoso" en el que se recogen datos. Una vez que tenemos suficientes datos de cada estado posible en la SDN, una red bayesiana es aprendida de ella, la búsqueda de relaciones causales entre los valores de los datos y el estado de SDN. Entonces, esta red bayesiana se utiliza para diagnosticar posibles fallos en la red (Bernayas de los Santos, 2018).

DESARROLLO

Primeramente se da un análisis de una identificación en la estructura de las redes definidas por software (SDN) de una manera teórica, de esta manera se busca lograr un mejor entendimiento a nivel de seguridad, después se analiza si pueden ser o no vulneradas fácilmente y cuál es la posibilidad que existe de alguna afectación, si ejecutamos una mala implementación desde el comienzo, ya sea por inexperiencia a nivel de protocolos o por falta de práctica, esto se debe a que en la mayoría de los casos no se validan parámetros importantes al momento de la configuración inicial por parte de los desarrolladores. Y es ahí donde con la investigación y la predicción de errores basado en el teorema de Bayes se pretende concluir si la instalación de las redes SDN puedan afectar en parte

la seguridad de la infraestructura donde se realizará la instalación.

Para describir el funcionamiento de una red bayesiana, primero hay que explicar el concepto de razonamiento probabilístico. Este se da cuando tenemos un conjunto de variables que describen nuestro entorno, y queremos la 'razón' sobre ellos (para encontrar un modelo que explica adecuadamente el funcionamiento de nuestro entorno) podemos utilizar varios enfoques.

Logrando así una recopilación de información que pueda ser utilizada como medio de referencia en las redes SDN, la documentación que se encuentra en internet está destinada a redes convencionales y para este caso la seguridad del plano de control está conectada de la seguridad de los protocolos de enrutamiento que incluyen MD5 para EIGRP, IS-IS, OSPFV2, IPsec, AH en el caso de OSPFV3. GTSM/ACLs y contraseñas para Mp-BGP, técnicas muy diferentes a las utilizadas en las redes SDN, generando vacíos en la implementación del plano SDN (Bernayas de los Santos, 2018) (Figura 1).

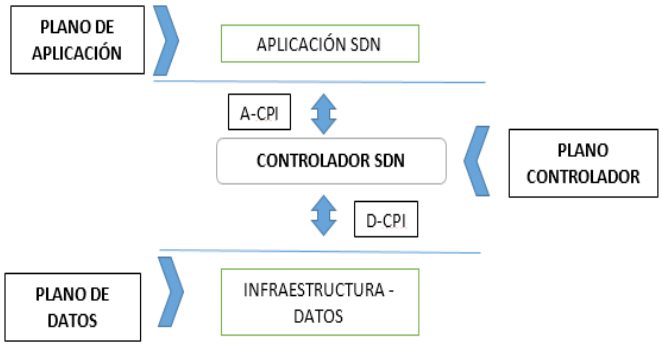

Figura 1. Arquitectura SDN

La arquitectura de las redes definidas por software contiene tres capas las cuales cumplen funciones específicas tal como lo especifica la ONF (Open Nerworking Fundation), fundación sin fines de lucro que se encarga de impulsar el desarrollo de las SDN. Estas etapas se detallarán a continuación:

. Plano de Datos o arquitetura

Es la capa se encarga de incorporar los recursos que interactúan con los clientes, dentro de los recursos que incorpora está la virtualización, conectividad, seguridad, disponibilidad y calidad de servicio. Esta capa cumple la función de enviar datos hacia la capa de controlador y así mismo revivir datos para los clientes y recursos (Figura 2).

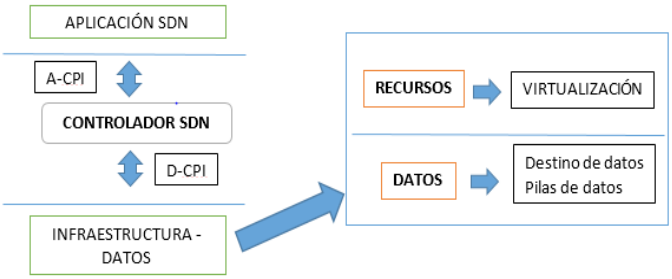

Figura 2. Plano de datos o arquitectura.

. Plano de controlador

El controlador SDN es una entidad de software que tiene control exclusivo sobre un conjunto de abstracto de recursos del plano del control, es decir es la entidad que controla y configura los nodos, se enfoca en la toma de decisiones y la selección del mejor camino para el tráfico según la red lo requiera (Figura 3).

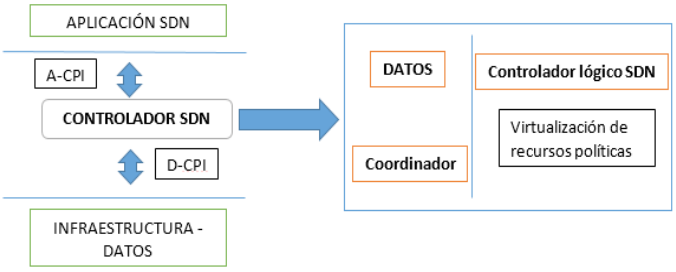

Figura 3. Plano de controlador

. Plano Aplicación

Esta etapa consiste en la implementación de aplicaciones para usuarios finales, las cuales utilizan servicios de comunicación de SDN a través de API (Interfaz de programación de aplicaciones) (Figura 4).

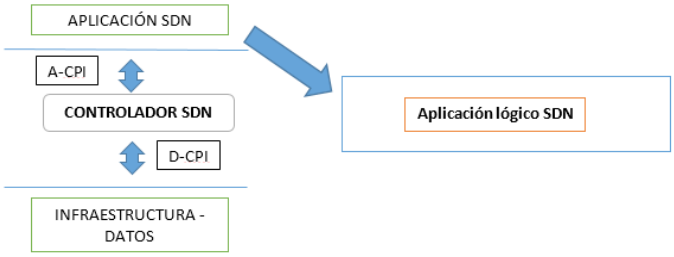

Figura 4. Plano de aplicación

Comparativa de la arquitectura de red tradicional vs arquitectura SDN

En la comparativa entre la arquitectura de red tradicional y la arquitectura SDN se observan las posibles causantes

que sugieren un cambio de tecnología en las redes, así como la tendencia a las SDN (España Tarapuez, 2016).

En la arquitectura de red tradicional una aplicación corre en un servidor e intercambiaba tráfico fundamentalmente con sus clientes. Pero hoy, las aplicaciones están distribuidas a lo largo de múltiples máquinas virtuales, que intercambian tráfico unas con otras y que pueden ser movidas de servidor físico para optimizar los servicios y balancear la carga de los servidores. Mientras que cada aplicación sigue teniendo sus requerimientos de red (Redclara) (España Tarapuez, 2016).

Las redes tradicionales están descentralizadas en el control. La complejidad de las redes de hoy en día hace que sea muy difícil aplicar un conjunto coherente de acceso, seguridad, calidad de servicio y otras políticas a los usuarios cada vez más móviles, lo que deja a la empresa vulnerable a violaciones de seguridad, incumplimiento de las normas y otras consecuencias negativas (España Tarapuez, 2016).

Las SDN permiten a los operadores de red configurar, administrar, proteger y optimizar los recursos de manera flexible (España Tarapuez, 2016). Los diseños y equipos utilizados tradicionalmente en las redes de telecomunicaciones, aunque funcionan correctamente, no están alineados con los objetivos de negocio ni con la lógica de las aplicaciones, sino que forman una estructura cerrada y estática que no se puede adaptar en tiempo real a la demanda de las aplicaciones.

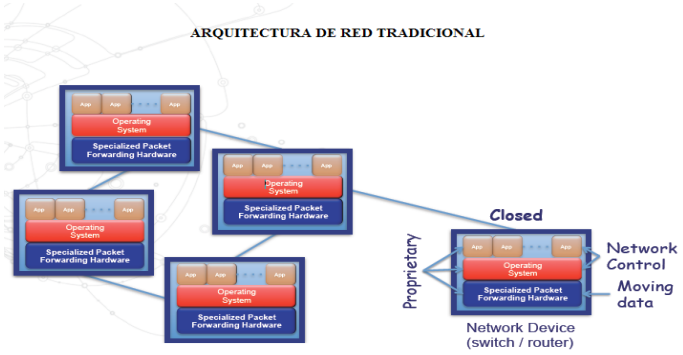

Figura 5. Arquitectura Red Tradicional

Fuente: España Tarapuez (2016).

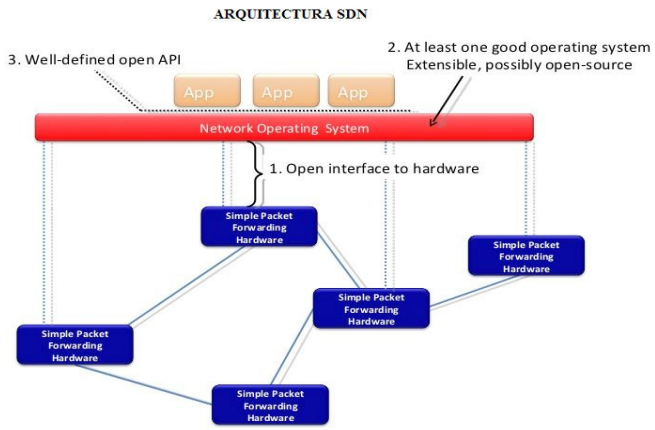

Figura 6. Arquitectura SDN.

Fuente: España Tarapuez (2016).

Diferencias entre la red tradicional y SDN

De lo dicho anteriormente, en la Figura5 y 6 se hace evidente de manera gráfica que el control centralizado programable es el aspecto más importante de las SDN (Figura 7).

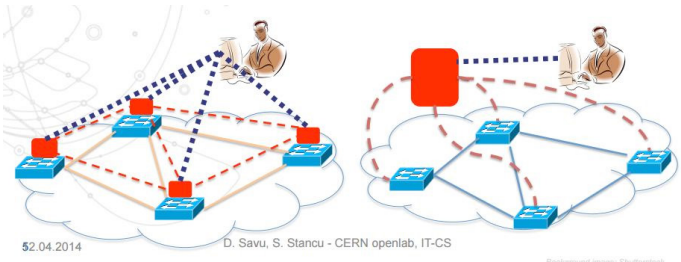

Figura 7. Red tradicional vs SDN.

Fuente: España Tarapuez (2016).

Métricas de evaluación

- Performance.

Se lo define como el rendimiento de una red en función de la velocidad de transferencia al realizar alguna tarea o proceso. Llevando a las redes SDN esta métrica permite ver cuáles son los recursos que utiliza cuando la red esté operativa, así mismo si su rendimiento bajo con su funcionamiento (Guzmán Vélez & Cáceres Miranda, 2019).

- Ancho de banda.

Se define como la capacidad de transmisión de datos dentro de una red, relacionando tanto velocidades de subida como de bajada. Este parámetro se mide en Mbps. (Guzmán Vélez & Cáceres Miranda, 2019).

- Latencia.

Es el parámetro que permite medir el tiempo que tarda un paquete en llegar a su destino. Por lo general los paquetes dentro de una red tienen un retardo impredecible pero afecta considerablemente la calidad del servicio dentro de una red (Guzmán Vélez & Cáceres Miranda, 2019).

- Rendimiento.

Es un atributo fundamental dentro de una red debido a que evalúa la velocidad de la red en función del software y hardware que se esté usando en la misma. Este es uno de los parámetros que más problemas causa ya que es afectado directamente por los equipos y softwares, y si estos no están actualizados pueden generar un rendimiento instable dentro de la red (Guzmán Vélez & Cáceres Miranda, 2019).

Fase 1

Simulación de la red definida por software

Con el fin de simular las redes definidas por software se necesita de dos elementos básicos, un simulador de redes y un controlador SDN. En este proyecto se ha decidido utilizar el software de simulación Mininet y el controlador SDN OpenDaylight, los cuales han sido escogidos por características como tipo de licencia, plataformas que soporta, lenguaje de programación, uso investigativo.

Este proyecto ha sido desarrollado en un computador con sistema operativo Windows 10 PRO usando una máquina virtual que contiene Linux Ubuntu 14.04 LTS, y que a su vez tiene instalado Mininet 2.2.2, la misma que se ha cargado usando VirtualBox.

Escenarios de simulación

El escenario de simulación a emplearse se muestra a continuación (Figura 8).

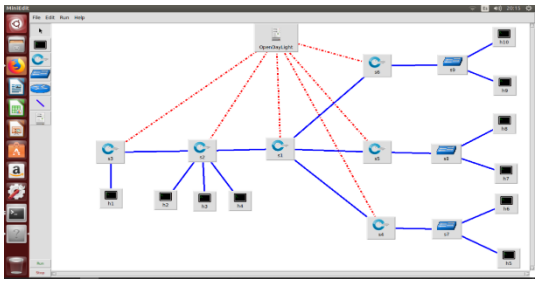

Figura 8. Escenario de simulación.

Para recrear los escenarios mencionados se necesita establecer el direccionamiento IP y un número de puerto para el controlador (Tabla 1).

Tabla 1. Direccionamiento IP para el escenario y puerto para el controlador.

DOMINIOS	IP	PUERTO
OpenDayLight	192.168.1.123	6653
S1	192.168.10.1	
S2	192.168.20.1	
S3	192.168.30.1	
S4	192.168.40.1	
S5	192.168.50.1	
S6	192.168.60.1	
H1	10.0.10.1	
H2	10.0.20.1	
H3	10.0.30.1	
H4	10.0.40.1	
H5	10.0.50.1	
H6	10.0.60.1	
H7	10.0.70.1	
H8	10.0.80.1	
H9	10.0.90.1	
H10	10.0.100.1	

Topología empleada en la simulación

Se ha utilizado una topología personalizada (custom), ya que se requiere incluir parámetros de red en el código evitando repetir el proceso cada vez que se necesite crear la topología. Adicionalmente, se ha incorporado en la programación el llamado de los tres controladores (Figuras 9, 10, 11, 12 y 13).

Para ejecutar una topología personalizada se utiliza el siguiente comando:

$ sudo python [nombre_de_la_topología].py

Se ha generado para el escenario un archivo de topología.

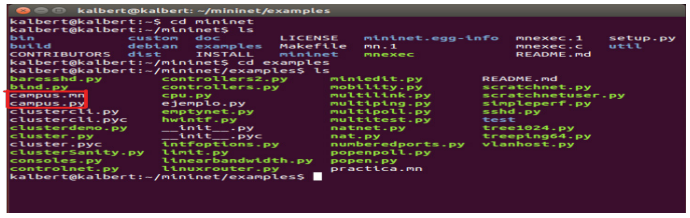

Figura 9. Archivos de la topología para el escenario

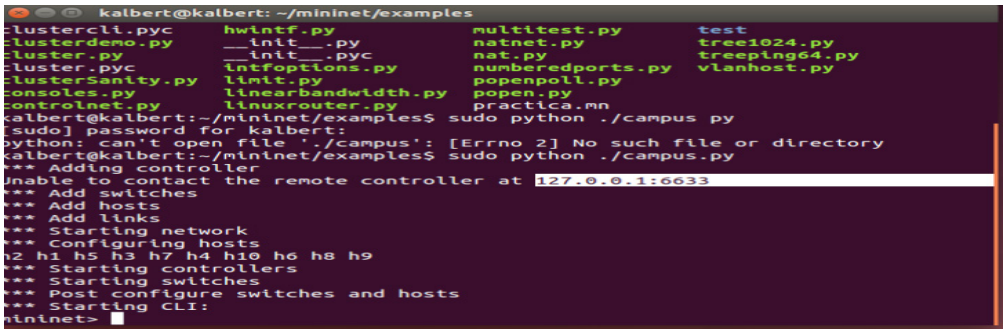

Figura 10. Ejecución de la topología.

Métricas

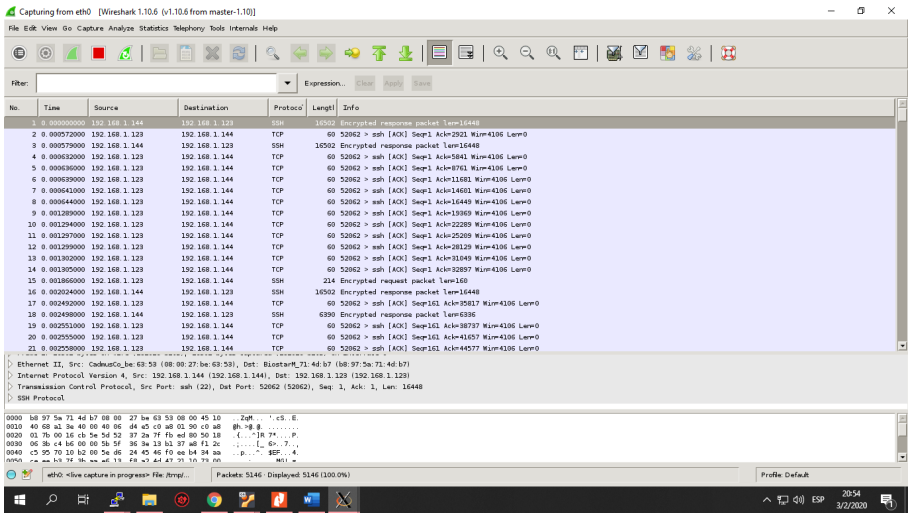

Figura 11. Prueba de Perfomance.

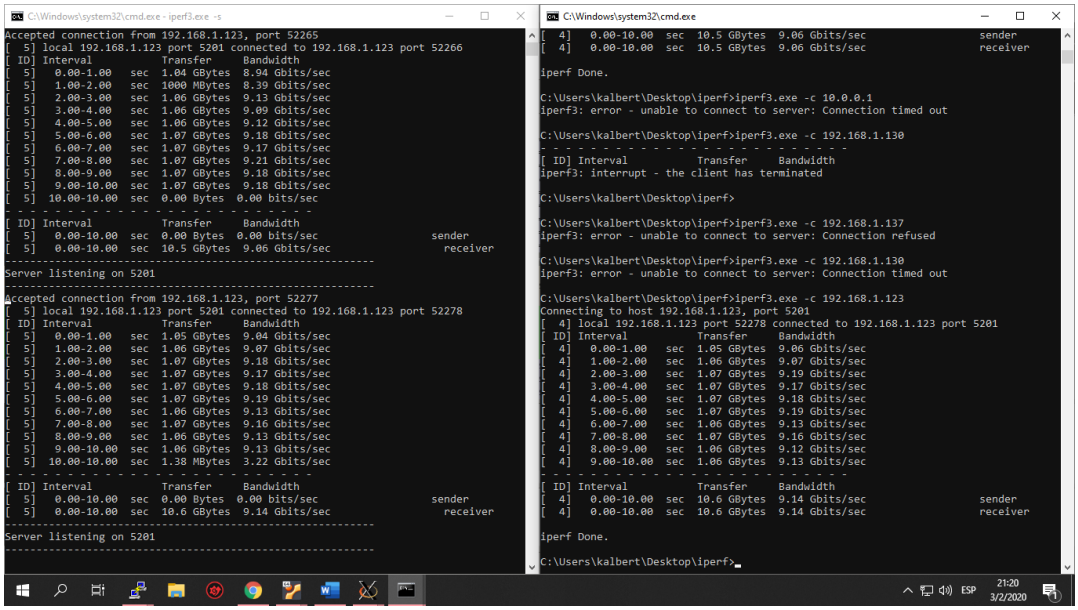

Figura 12. Prueba de Ancho de Banda.

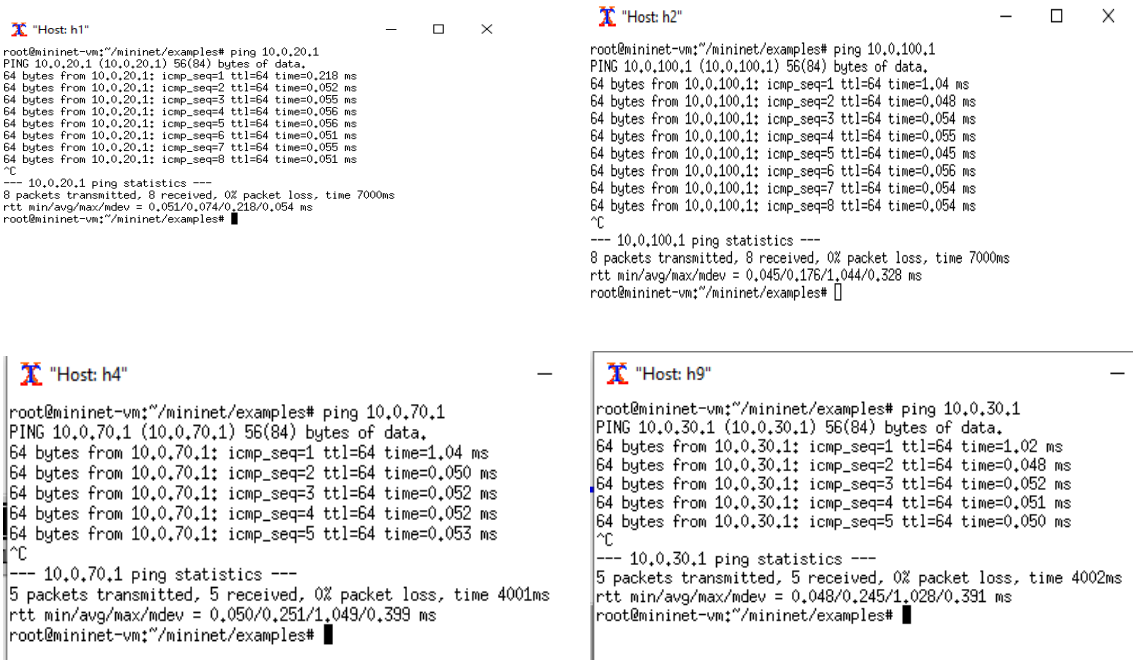

Figura 13. Prueba de Rendimiento.

Fase 2

Generación de fallos

Ahora tenemos una red simulada en pleno funcionamiento. Por lo tanto, el siguiente paso es generar fallos dentro de ella, con el fin de obtener datos suficientes para obtener una red bayesiana que los modelos de nuestro escenario SDN. Específicamente, se definen los siguientes estados posibles de la red:

- S0 - No hay quejas - el estado OK.
- S1 - El cierre de un nodo.
- S2 - Desconexión de un servidor de centro de datos de la red.
- S3 - Modificación de las normas out-port en un nodo.
- S4 - La modificación de las reglas in-port en un nodo.
- S5 - Adición de idle-timeouts de espera en un nodo.
- S6 - Adición de hard-timeouts de espera en un nodo.
- S7 - cambios de las prioridades de flujo en un nodo.
- S8 - Forzar un nodo a perder paquetes LLDP.
- S9 - Modificación de reglas en out-port e in-port dentro de un nodo.

Los fallos se crearon mediante la modificación de las reglas de flujo de corriente dentro de un conmutador a través de solicitud a la transferencia SDN Controller Representational State API (REST) (tales como estados de S1 y S3 a S9) o mediante el uso de la API MiniNet (como el estado S2) para modificar la red topología.

Modelos de redes bayesianas

Basado en los estudios previos realizados sobre redes bayesianas (Oviedo, et al., 2018) y usando nuestro módulo de simulación de la red, hemos obtenido dos conjuntos de datos de dos simulaciones diferentes; el primer conjunto de datos se utilizará como un conjunto de datos “formación”, con el fin de ejecutar el algoritmo bayesiano de búsqueda sobre ella y obtener una red bayesiana. En el proceso de aplicar el algoritmo de búsqueda bayesiano para la

búsqueda de las redes bayesianas, hemos configurado los siguientes parámetros:

- Máxima cantidad de nodos padres: 8. limitar el número de nodos padre de un nodo puede tener, con el fin de limitar el consumo de memoria y el tiempo empleado en el proceso de razonamiento.
- Iteraciones: 20. Como sabemos, la red de bayes lleva a cabo un proceso de búsqueda de escalada partiendo de una red aleatoria. Con el fin de buscar el espacio de posibles redes bayesianas, reiniciamos el proceso de escalada, una vez que hemos hecho una serie de cambios en la red actual. El espacio aumenta con el número de iteraciones buscado.
- Enlace de probabilidad: 0,1. Influye en la conectividad de la red aleatoria de partida.

En cuanto a la función de puntuación del algoritmo de búsqueda bayesiano nos basamos también en el estudio (Oviedo, et al., 2019), hemos seleccionado la exactitud en los datos de entrenamiento como la función de puntuación, y hemos aplicado 10 veces la validación cruzada en el hallazgo de la exactitud. Una vez que hemos utilizado los conjuntos de datos de entrenamiento para generar redes bayesianas, se utilizan los conjuntos de datos de "prueba", para evaluar la calidad general de nuestros modelos bayesianos.

Modelo 1: Modelo de atributo - nivel

En el caso del primer modelo, hemos seguido un método de procesamiento más a fondo de los datos. En concreto, en la etapa de procesamiento basado en el tiempo, en lugar de sólo en busca de cualquier cambio en una regla de flujo, hemos analizado varios atributos que componen la regla de flujo, y su evolución a través y ventana de tiempo especificado. Por lo tanto, estamos en condiciones de cambios inesperados detectados en los atributos específicos de cada regla de flujo (Figura 14).

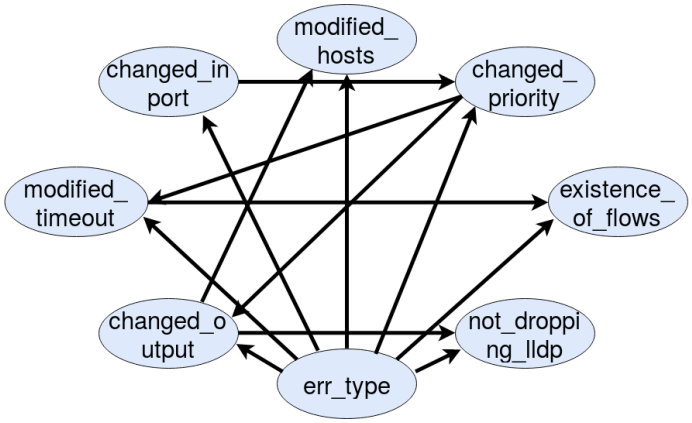

Figura 14. Diagrama de Modelo de atributo – nivel.

Modelo 2: Modelo de rápido aprendizaje

Para el aprendizaje de este modelo hemos evitado proporcionar ningún conocimiento previo. También hemos modificado algunos parámetros del algoritmo de aprendizaje con el fin de que sea más rápido. Específicamente, hemos reducido el número de iteraciones a 10 y el número máximo de nodos padre a 4, con el fin de limitar aún más la duración del proceso de aprendizaje (Figura 15).

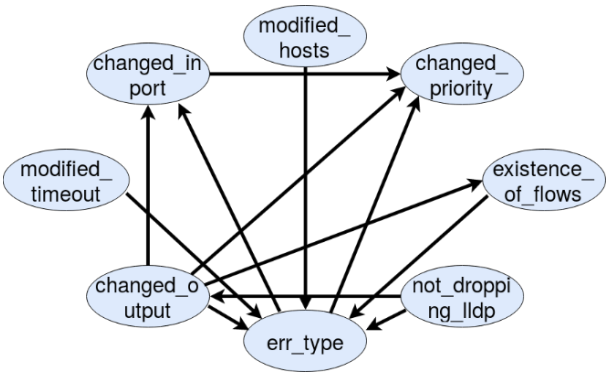

Figura 15. Diagrama de Modelo de rápido aprendizaje.

Modelo 3: Modelo de Flujo de nivel

En este caso, no analizamos qué atributos han cambiado dentro de cada regla de flujo. En su lugar, se analiza la regla de flujo en su conjunto, por lo tanto, siendo sólo es capaz de detectar si la regla de flujo ha cambiado, pero no qué atributo dentro de ella ha cambiado. Como resultado, en lugar de variables tales como "changed_inport" o "modified_timeout" usamos "changed_flow" (Figura 16).

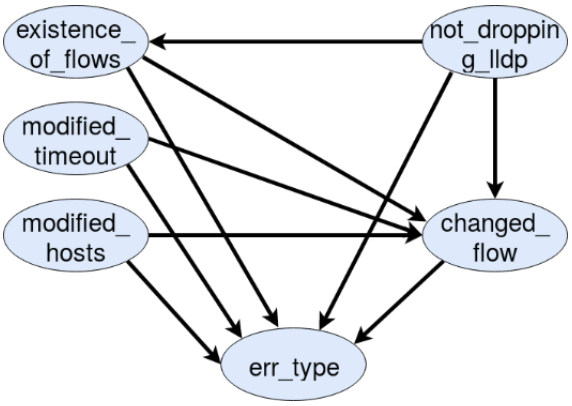

Figura 16. Diagrama de Modelo de Flujo de nivel.

Modelo 4: Modelo Pure Naive Bayes (Khajenezhad, et al., 2021).

En el caso del cuarto modelo, hemos adoptado un enfoque "puros Naïve Bayes": sólo asumimos relaciones condicionales entre la variable a predecir y el resto de las variables en el modelo (Figura 17).

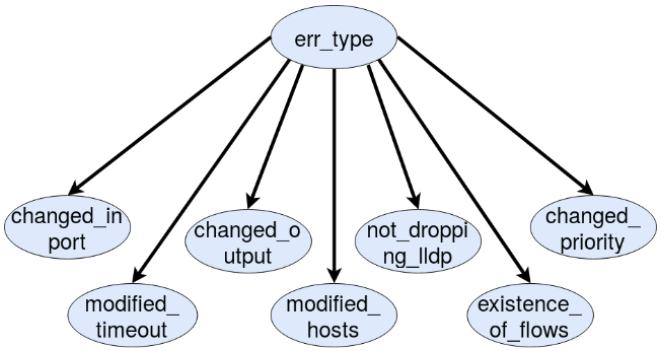

Figura 17. Diagrama de Modelo Pure Naive Bayes.

Resultados esperados

Los estados (fallos de red) que se presentan en esta sección se definen en base a los parámetros utilizados para evaluar cada uno de los modelos antes aplicados: Exactitud (Accuracy), Recordar (Recall), Precision (Precisión) y Puntuación F1 (F1-Score).

- Accuracy. - Representa la fracción de predicciones que nuestro modelo ya acertó en una clase específica; Sin embargo, esta métrica no tiene en cuenta las predicciones que ya ha recibido mal.
- Recall. - La proporción de las entradas de un cierto estado fue identificado correctamente.
- Precision. - Nos muestra la proporción de entradas clasificadas como un cierto estado que eran correctas.
- F1-Score. - Es la media armónica de la retirada y la precisión.

Dado que la F1-Score tiene en cuenta tanto la precisión y recordar, y que no tiene las desventajas de que la precisión tiene, vamos a utilizar la puntuación de la F1 como la referencia con el fin para evaluar los resultados.

Modelo 1: Modelo de atributo - nivel

Como podemos ver en la Tabla 2, en el primer modelo se realiza el diagnóstico de la mayoría de los fallos. Sin embargo, podemos ver que tiene dificultades a la hora de diagnosticar un estado sin fallos de la red, ya que se obtiene un valor de F1-Score para el estado S0 de 0,69, inferior a la 0,95 - 1,00 rango que se obtiene en la mayor parte de los estados restantes (Figura 18).

Tabla 2. Parámetros para el Modelo 1.

Tipo de fallo	S0	S1	S2	S3	S4	S5	S6	S7	S8	S9
F1-Score	0.69	0.75	1.00	0.96	0.95	1.00	0.95	0.94	0.93	0.87
Recall	0.59	1.00	1.00	1.00	1.00	1.00	0.90	1.00	1.00	0.87
Precision	0.83	0.6	1.00	0.92	0.90	1.00	1.00	0.89	0.88	0.87
Accuracy	0.90	0.97	1.00	0.99	0.98	1.00	0.99	0.99	0.98	0.98

Figura 18. Parámetros para el Modelo 1.

Modelo 2: Modelo de rápido aprendizaje

Como podemos ver en la Tabla 3, los resultados obtenidos son similares a los presentados en el Modelo 1, pero con un método de aprendizaje más rápido. Además, no hemos proporcionado ningún conocimiento de fondo (Figura 19).

Tabla 3. Parámetros para el Modelo 2.

Tipo de fallo	S0	S1	S2	S3	S4	S5	S6	S7	S8	S9
F1-Score	0,72	0.89	1.00	0.96	0.95	0.97	0.90	0.94	0.93	0.86
Recall	0.68	0.89	1.00	1.00	1.00	0.94	0.82	1.00	1.00	0.86
Precision	0,77	0.89	1.00	0.92	0.90	1.00	1.00	1.00	0.88	0.86
Accuracy	0.90	0.99	1.00	0.99	0.98	0.99	0.98	0.99	0.98	0.98

Figura 19. Parámetros para el Modelo 2.

Modelo 3: Modelo de Flujo de nivel

En la Tabla 4, podemos ver la evaluación del modelo obtenido a partir del Modelo 3 antes descrito. Como podemos ver, debido al hecho de que estamos salteando algunos pasos en el procesamiento del conjunto de datos, estamos perdiendo la capacidad de detectar algunas fallas en la red.

En concreto, podemos ver que S3, S4, S5 y S9 las fallas probablemente están siendo diagnosticados como S3 (debido al hecho de que S3, S4, S5 y S9 tienen una recuperación de 0,00, y S3 tiene una precisión muy baja) (Figura 20).

Tabla 4. Parámetros para el Modelo 3.

Tipo de fallo	S0	S1	S2	S3	S4	S5	S6	S7	S8	S9
F1-Score	0,77	0.89	1.00	0.31	0.00	0.97	0.90	0.00	0.86	0.00
Recall	0.69	0.89	1.00	1.00	0.00	0.94	0.83	0.00	1.00	0.00
Precision	0.87	0.89	1.00	0.18	0.00	1.00	1.00	0.00	0,76	0.00
Accuracy	0.90	0.99	1.00	0,72	0.86	0.99	0.98	0.96	0.95	0.92

Figura 20. Parámetros para el Modelo 3.

Modelo 4: Modelo Pure Naive Bayes

Como podemos ver en la Tabla 5, los resultados obtenidos para este modelo son similares a los obtenidos para el Modelo 2, cuyos resultados pueden verse en la Tabla 3. Específicamente, los resultados obtenidos son los mismos excepto para los estados S0 y S5, donde se superó este modelo (Figura 21).

Tabla 5. Parámetros para el Modelo 4.

Tipo de fallo	S0	S1	S2	S3	S4	S5	S6	S7	S8	S9
F1-Score	0.69	0.89	1.00	0.96	0.95	0.87	0.90	0.94	0.93	0.86
Recall	0.59	0.89	1.00	1.00	1.00	1.00	0.82	1.00	1.00	0.86
Precision	0.83	0.89	1.00	0.92	0.90	0,77	1.00	1.00	0.88	0.86
Accuracy	0.90	0.99	1.00	0.99	0.98	0.98	0.98	0.99	0.98	0.98

Figura 21. Parámetros para el Modelo 4.

Como se mencionó anteriormente, Modelo 1, Modelo 2 y Modelo 4 tienen una ligera tendencia a encontrar errores incluso cuando la red está funcionando correctamente, como lo señala el F1-Score del estado S0 (estado S0: no hay errores); Sin embargo, esta tendencia no es preocupante, ya que el F1-Score es todavía alto. De hecho, tenemos algunos estados, tales como el estado S2, que es capaz de diagnosticar por completo. Por otro lado, podemos ver en

Modelo 3 una solución de compromiso entre el tiempo de procesamiento y la capacidad para diagnosticar algunos estados (en concreto, la capacidad para diagnosticar estados S4, S7 y S9, y distinguir de otros estados S3).

Por último, podemos ver en la comparación entre Modelo 4 y el resto de los modelos que son capaces de obtener un simple resultado, pero para el modelo Pure Naive Bayes es más utilizado como método de aprendizaje.

CONCLUSIONES

El procesamiento de los datos de los modelos Figuras probabilísticos generados y el conocimiento previo proporcionó una configuración del algoritmo de búsqueda bayesiano. Su evaluación mostró que algunos de estos modelos llegan a un F1-Score superior a 90%. Además, la simulación de las redes SDN constituye un método muy útil para esto pues permite evaluar el desempeño y la funcionalidad de estas redes, para lo cual Mininet es la herramienta de simulación recomendada. En función de las pruebas que se realicen en cada simulación se podrá extraer determinada información relevante del escenario. Por otro lado, se determinó la simulación de un escenario recreando un prototipo de una red de campus SDN, empleando Mininet, permitió obtener las métricas relevantes para las SDN y el desempeño del controlador OpenDaylight.

REFERENCIAS BIBLIOGRÁFICAS

Barrera Pérez, M. Á., Serrato Losada, N. Y., Rojas Sánchez, E., & Mancilla Gaona, G. (2019). ***Estado del arte en redes definidas por software (SDN).*** Universidad Distrital Francisco José de Caldas, 13(1).

Bernayas de los Santos, F. (2018). ***Aplicación de razonamiento bayesiano para el diagnóstico de fallos sobre un controlador de una red definida por software.*** (Trabajo de fin de grado). Universidad Politécnica de Madrid.

España Tarapuez, N. E. (2016). ***Diseño y simulación de una Red Definida por Software (SDN).*** (Trabajo de titulación). Universidad Central del Ecuador.

Guzmán Vélez, D. M., & Cáceres Miranda, C. A. (2019). ***Análisis del desempeño de redes definidas por software (sdn) frente a redes con arquitectura TCP/IP.*** (Tesis de pregrado). Universidad Técnica Estatal de Quevedo.

Khajenezhad, A., AliBashiri, M., & Beigy, H. (2021). A distributed density estimation algorithm and its application to naive Bayes classification. ***Applied Soft Computing***, 98.

Oviedo, B., Puris, A., & Zhuma, E. (2018). Algoritmos meta heurísticos para el aprendizaje de redes bayesianas. ***Revista Lasallista de Investigación***, 15(22).

Oviedo, B., Zambrano-Vega, C, & Gómez, J. (2019). Clasificador Bayesiano Simple aplicado al aprendizaje. ***Revista Ibérica de Sistemas e Tecnologias de Informação***, 18, 74-85.

Pérez Tardío, E. R. (2018). Un acercamiento a las Redes Definidas por Software. (Ponencia). ***Conferencia UCLV.*** Santa Clara, Cuba.

Vélez Mejía, C. L. (2018). ***Análisis de Seguridad en Redes SDN (Redes definidas por software).*** (Tesis de grado). Universidad Nacional Abierta y a Distancia.