Modelamiento Formal de una Arquitectura Básica SDN basada en CPN Jerárquicas
Formal Modeling of Basic Architecture SDN Based on Hierarchical CPN

S.Castrillón
Facultad de ingeniería. Universidad de Antioquia
GITA
Medellín, Colombia
sebastian.castrillono@udea.edu.co

Keywords— SDN, OpenFlow, CPN.

Abstract

Las redes definidas por software (SDN) definen un cambio sustancial en la forma como se administran las redes de datos actuales. SDN separa el plano de control y el plano de administración, lo cual se logra adicionando características de software simples a los dispositivos de red actuales. Conjuntamente con SDN fue desarrollado el protocolo OpenFlow para permitir la comunicación entre los dos planos y lograr la administración centralizada de la red. Los sistemas centralizados son fáciles de administrar, pero también existe la probabilidad de errores de implementación, diseño y operación; limitando la escalabilidad y el rendimiento de la red. Este artículo es un primer paso para desarrollar una herramienta de simulación que permita dimensionar y probar la capacidad de expansión óptima de las redes basadas en SDN. Se define una topología de red básica SDN basada en redes de Petri Coloreadas Jerárquicas (HCPN) como técnica formal de modelamiento y análisis del switch y el controlador OpenFlow, el modelo es validado formalmente mediante el análisis del espacio de estados y sus propiedades. Este modelo podrá usado como base para realizar simulaciones de algoritmos de optimización que permitan mejorar el rendimiento de los controladores actuales con tráficos sensibles como el video y la voz.

1 Introducción

Las redes definidas por software representan un método de virtualización de redes basado en un modelo propuesto por el profesor Nick McKeown [1] en la Universidad de Stanford, donde los dispositivos de la red trasmiten la responsabilidad de decisión de envío a un dispositivo central llamado controlador.

Las redes tradicionales están diseñadas para que los equipos tengan propósitos específicos e implementen protocolos, controles de acceso y monitorización. Estos equipos tienen integrado el plano de control y el plano de envío de la red tomando decisiones independientes sin una visión general de la red [2]. El concepto de SDN logra separar el plano de los datos y el plano de control, lo que permite tomar decisiones con conocimiento amplio de la red y da una flexibilidad innovadora en el mundo de las comunicaciones definiendo el concepto de redes programables sin necesidad de cambios de hardware, en ocasiones solo actualizando el firmware de los dispositivos actuales.

En la Figura 1 se presenta una vista lógica de la arquitectura SDN, donde la inteligencia de la red está (lógicamente) centralizada en un controlador SDN, el cual mantiene una vista global de la red y toma las decisiones de procesamiento de los flujos de datos [3].

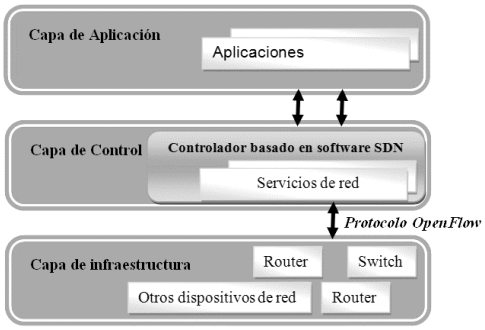

Figura 1. Arquitectura de capas SDN

Uno de los grandes usos para SDN es lograr el control del tráfico End-to-End para garantizar la calidad del servicio (QoS) en comunicaciones sensibles al retardo como la voz y el video para servicios multimedia, permitiendo implementar políticas generales para los diferentes dispositivos intermediarios desde un controlador central [4]. SDN también permite mejorar la aplicación de Calidad de la Experiencia (QoE) y ha sido usado en ambientes de laboratorio para mejorar la negociación de los servicios multimedia [5].

Actualmente se identifica la escasa disponibilidad de herramientas de simulación experimental que permitan dimensionar y probar la capacidad de expansión óptima de las redes basadas en SDN, y conjuntamente la necesidad de realizar pruebas de laboratorio en tiempos adecuados para topologías extensas con diferentes requerimientos de tráfico, se convierten en necesidades actuales de entendimiento y análisis de redes. En consecuencia, entender el rendimiento y las limitaciones de SDN y OpenFlow es un prerrequisito para el desarrollo de

nuevas aplicaciones para las redes definidas por software (SDN)[6].

Por otro lado, la aplicación de métodos formales para describir la semántica de los sistemas de comunicación, permite identificar y analizar patrones como la coordinación o la cooperación de los protocolos de comunicación [7]. Entre los métodos que son ampliamente usados por su facilidad de análisis y representación se encuentran: las redes de Petri (PN), las Redes de Petri coloreadas (CPN), las Labeled Predicate Transition Nets (LPrT) y las Stochastic Timed Petri Nets (STPN) [8], dado que son herramientas formales útiles para describir de manera simplificada sistemas que se pueden descomponer en subsistemas con estructuras y comportamientos autosimilares, permitiendo un modelado robusto para su posterior simulación y análisis.

Como las redes de Petri ordinarias, las CPNs tienen una representación gráfica que está basada en una definición matemática subyacente. Estas han sido ampliamente usadas para modelar y simular protocolos que son la base de muchos servicios actuales como PPPoE [9], SIP [10] y OSPF [11]. Por tanto las CPN representan una excelente herramienta para el modelamiento de un ambiente SDN.

Este artículo está estructurado de la siguiente manera. En la Sección II se realiza una descripción de los conceptos bases para la compresión del modelo propuesto y se describen trabajos relacionados. En la Sección III se realiza la descripción del modelo y la validación formal realizada. En la sección IV y V se definen conclusiones y los planes para trabajos futuros respectivamente.

2 Marco teórico y trabajos relacionados

Para lograr el entendimiento adecuado se realiza una descripción básica de los 4 conceptos base tratados en este artículo que son: redes de Petri, redes de Petri coloreadas, SDN y OpenFlow.

2.1 Redes de Petri (PN)

Una red de Petri (PN) es un término genérico usado para representar sistemas dinámicos a eventos discretos (DEDS), cuya mayor ventaja reside en el soporte matemático subyacente y en la representación gráfica de estas. Mediante una PN, los DEDS se pueden describir mediante una topología distribuida, paralela o concurrente Las redes de Petri más primitivas fueron definidas en la década de los años 1960 por Carl Adam Petri [12], son una generalización de la teoría de autómatas que permite expresar un sistema a eventos concurrentes.

Una PN es un grafo dirigido compuesto por lugares, transiciones, arcos dirigidos y marcas que ocupan posiciones dentro de los lugares como se observa en la Figura 2

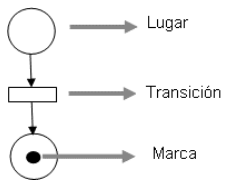

Figura 2. Redes de Petri

Matemáticamente, las PN básicas se definen como una tripleta $\langle P, T, F\rangle$, donde $P = \{p_1, p_2, \dots, p_n\}$ contiene n lugares de la PN. $T = \{t_1, t_2, \dots, t_m\}$ Contiene las m transiciones de la PN, y $F = P \times T \cup T \times P$ Representa los arcos que conectan las transiciones y los lugares. Debido a que estas conexiones representan un flujo y por tanto son direccionadas " $\times$ " representa el producto cartesiano.
Usualmente, dicha tripleta se representa matemáticamente mediante una matriz de incidencia $C = p \times t$ donde:

$$c(i,j) = \begin{cases} W(t_i, p_j) & ,iff\ (t_i, p_j) \in F \\ -W(p_i, t_j) & ,iff(p_i, t_j) \in F \\ 0 & en\ otros\ casos \end{cases}$$

W Representa la magnitud del peso del arco que conecta los elementos indicados en el conjunto F.

Esta representación matricial de las PN trae consigo la representación del marcaje de la PN mediante dos vectores: M que representa el marcaje actual de los lugares de la PN y M_0 que representa el marcaje inicial. Formalmente $M = [|m(p_1)|, |m(p_2)|, \dots, |m(p_n)|]^T$ donde $m(p_i) \geq 0$ es el marcaje del i-ésimo lugar [13].

En su forma más básica, las marcas que circulan en una PN son todas idénticas limitando la escalabilidad de los modelos. Se puede definir una variante de las redes de Petri en las cuales las marcas pueden tener un color (una información que las distingue), un tiempo de activación y una jerarquía en la red.

2.2 Redes de Petri coloreadas (CPN)

Las CPN se caracterizan principalmente porque las marcas que se ubican en los lugares pueden tener un tipo o característica determinada, a cada marca de un tipo se le llama tipo de dato y permite distinguir unas marcas de otras. A cada tipo de dato se le denomina color y al dato representado en una marca se le denomina marca. Las CPN poseen tres diferencias fundamentales con respecto a las PN clásicas: (a) los lugares de una CPN pueden contener elementos de diferentes tipos, (b) los arcos de una CPN poseen expresiones que manipulan los valores de las marcas, y (c) las transiciones poseen la capacidad de evaluar los valores de las marcas mediante una guarda que permite o inhibe su sensibilización según la evaluación sea falsa o verdadera respectivamente. La existencia de tipos de datos en las CPN permite que cada lugar pueda contener uno o más valores correspondientes a un tipo básico o a un producto de tipos. Cada dato presente en un lugar lleva un indicador de su multiplicidad. Es decir, se puede determinar cuántas copias del dato se encuentran en un lugar [14].

Las características particulares de las CPN confieren un mayor potencial para el modelado de sistemas, donde las expresiones de arco y las guardas permiten restringir mejor las condiciones de disparo de las transiciones en una CPN con respecto a las PN clásicas. El empleo de colores y de expresiones de arco hace que el modelo en CPN sea más compacto que el modelo equivalente con PN clásicas [15].

2.3 Redes definidas por software (SDN)

La arquitectura SDN está basada en capas, lo que permite un fácil despliegue de fabricantes y aplicaciones para optimizar de forma flexible los servicios actuales y futuros, estas capas cumplen las siguientes funciones:

Capa de Aplicación: La arquitectura SDN soporta un conjunto de interfaces de programación de aplicaciones (APIs) que hacen posible implementar los servicios comunes de red, incluyendo enrutamiento, multicast, seguridad, control de acceso, administración de ancho de banda, ingeniería de tráfico, calidad de servicio, optimización del almacenamiento y el procesamiento, administración de la energía, y todas las formas de administración de políticas, diseñadas a medida para satisfacer las necesidades actuales[3].

Capa de Control: En esta capa se encuentra el controlador SDN, el cual facilita la administración y la operación de la red permitiendo configurar de manera óptima múltiples equipos, evitando la creación de políticas individuales en cada equipo de la capa de infraestructura. El controlador más conocido académicamente es el NOX [16] el cual ha sido desarrollado en C++ y tiene su variación en Python POX, también se cuentan con otros desarrollos como Beacon[17] que cuenta con interfaz web de administración y Floodlight [18] que ha sido bien acogido por la industria

Capa de infraestructura: En esta capa se ubican los equipos de red (Switch y routers) compatibles con SDN y el protocolo OpenFlow que permite la comunicación con el controlador SDN de la capa de control. La lista de equipos de red compatibles con SDN puede ser encontrada en [19]

2.4 OpenFlow

El protocolo OpenFlow es utilizado como interfaz entre los dispositivos de la infraestructura de red y el software que funciona como controlador SDN. Openflow permite el intercambio de mensajes entre el controlador SDN y el switch como se observa en la Figura 3, logrando mantener actualizada la tabla de flujos del switch mediante las modificaciones de flujo enviadas por el controlador de acuerdo a las API o reglas que este tenga.

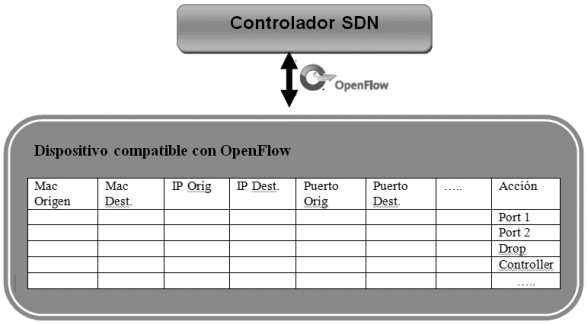

Figura 3. Openflow

Un flujo se define como un conjunto de paquetes con características o encabezados simulares. Estas características normalmente están en las capas 2 a la 4 del modelo OSI y permiten comparar y clasificar múltiples tráficos de red.

OpenFlow es un protocolo que está siendo implementado actualmente en redes empresariales y ha dejado de ser un protocolo solo utilizado en ambientes académicos [20] . Actualmente empresas como Google tiene implementado prototipos de OpenFlow sobre su red de backbone a la cual le ha llamado G-Scale Network[21]. Los principales fabricantes de equipos de redes de telecomunicaciones ya han implementado equipos y sistemas operativos con compatibilidad OpenFlow, al no requerir cambios a nivel de hardware y solo con algunos cambios en el Sistema operativo de los equipos logran soportarlo. Algunas de las empresas que tienen la compatibilidad actualmente en su hardware son HP [22], Juniper [23] y Cisco [24].

2.3 Trabajos relacionados

Recientemente se ha observado artículos como [6], [25], [26] y [27] donde se muestra la funcionalidad de modelar, simular y validar las topologías de las redes SDN. En [6] se efectúan medidas de laboratorio para determinar los tiempos de respuesta y la probabilidad de perdida de paquetes de un controlador OpenFlow, adicionalmente propone un modelo teórico basado en colas que permite el análisis del rendimiento del controlador. Este modelo aunque es un gran avance no permite probar modificando los comportamientos algorítmicos del controlador y simular nuevas estrategias de optimización. En [25] se define un modelo en CPN para el análisis de reglas de seguridad en una red SDN OpenFlow y posterior análisis del espacio de estados para realizar la validación del modelo propuesto, adicionalmente en [26] verifican la interacción del controlador y el switch OpenFlow mediante la modelación del canal con redes de Petri jerárquicas. En [27] se realizan la pruebas experimentales de ambientes multimedia y calidad de servicio utilizando a OpenFlow como estrategia para mejorar las decisiones de enrutamiento.

Actualmente ninguno de los artículos mencionados se enfoca en el diseño formal del switch y el controlador con en el fin de diseñar una herramienta de simulación que sea flexible y escalable para el diseño de diferentes topologías de red y que permita fácil manipulación experimental del controlador como lo permiten las CPN.

3 Modelamiento y análisis formal

A continuación se realiza una descripción detallada de los modelos realizados para el switch y el controlador OpenFlow con la herramienta CPN Tool [28], adicionalmente se valida formalmente mediante la validación de propiedades de CPN y análisis del espacio de estados.

En la Figura 4. Se identifica cada uno de los estados por los que pasa el modelo completo:

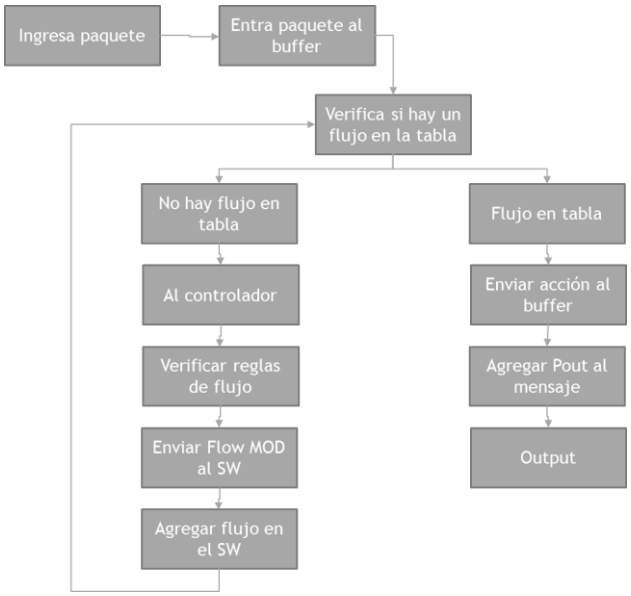

Figura 4 Diagrama de secuencia de estados

3.1 Modelo del switch

El modelo ilustrado en la Figura 5, se represente al switch SDN donde el lugar *Input_Flow* representa el flujo que llega al switch y *sw_table* la tabla de flujos usada para la conmutación de los flujos. La información contenida en *sw_table* es un conjunto de marcas que indican el puerto a través del cual se envía el flujo de paquetes que llegan cuando estos poseen un conjunto de parámetros similares como dirección Mac de Origen, Dirección MAC de destino y puerto de entrada. Si el paquete que ingresa no tiene ninguna coincidencia con la *sw_table*, entonces el modelo envía el paquete al controlador para que este determine la acción necesaria para computar este paquete. Cuando el controlador le entrega un *FlowMod* al Switch, este actualiza su tabla de flujos con el marcaje que le llegue del controlador, el cual representa una entrada en la tabla de flujos. El modelo termina con las marcas acumuladas en el lugar *Output_Flow*, indicando hacia qué puerto deben ir y qué paquetes deben enviar.

ESTADOS		TRANSICIONES	
Input_Flow	Entrada de datos al switch	Input	Aceptación de un flujo para ser analizado
Buffer	Flujo a analizar	To_check	Flujo de datos enviado a comparación en la tabla
in_controller	Flujo analizado en el controlador	In_table	Datos en la tabla Open Flow del switch
table_checker	Comparación del flujo con la tabla Open Flow del switch	Not_in_table	Datos no encontrados en la tabla Open Flow del switch
sw_table	Tabla Open Flow del Switch	Table_change	Recepción de una orden del controlador para modificar la tabla del Switch
is_not_table	Indica que el flujo analizado no está en la tabla		
From_controller	Flujo de entrada al switch desde el controlador Open Flow		
Is_in_table	Indica que el flujo analizado está en la tabla y debe enviarse por un puerto de salida según la tabla	To_controller	Envío de los datos no encontrados en la tabla, hacia el controlador
Output_Flow	Flujo de salida de datos del switch hacia los puertos	Output	Salidas del flujo hacia el controlador.

Tabla 1. Descripción de las partes del modelo del switch

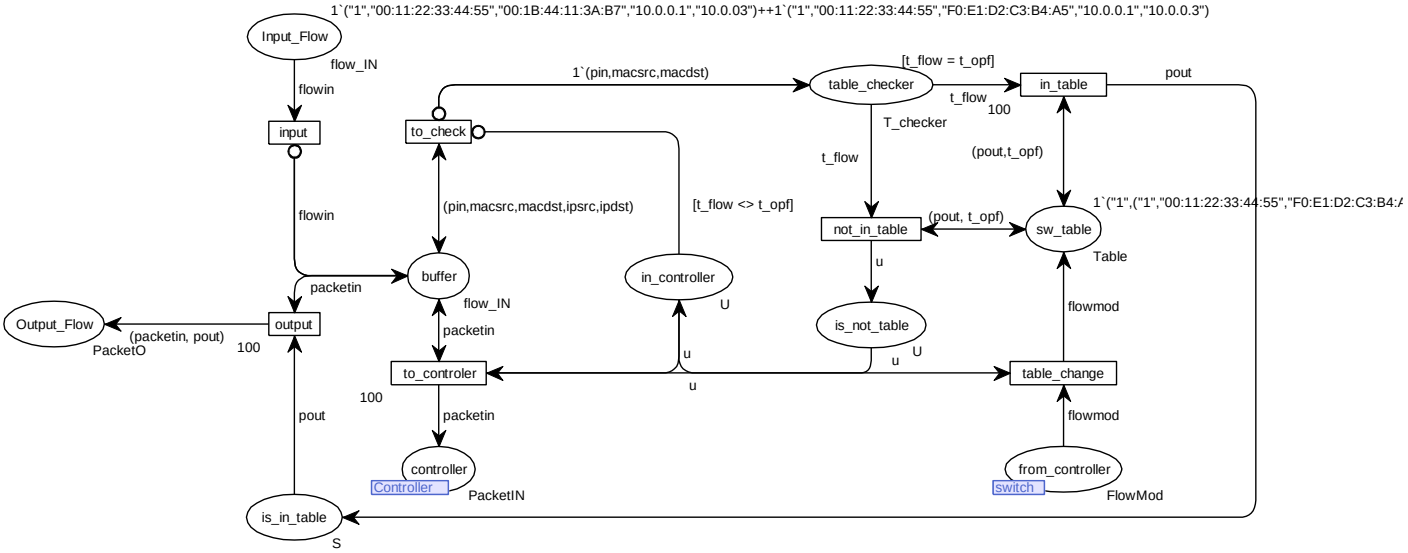

Figura 5. Modelo de switch SDN en CPN

3.2 Modelo del controlador

En el modelo presentado en la Figura 6 se describe el proceso de comparación de reglas y envío del mensaje para la modificación de los flujos del switch. Inicialmente se recibe una marca proveniente del switch cuando este no encuentra una coincidencia en su tabla de flujo, esta marca que lleva información del flujo es comparada con la lista de reglas alojadas estáticamente en el lugar *Rules_set*, si el controlador encuentra una regla que coincida con el flujo que ingresó se genera un mensaje *Flow_Mod* que se envía al switch indicando la acción que debe tomar para ese flujo específico. En esta primera versión del modelo y las reglas son agregadas estáticamente por el administrador, pero el escalado a reglas y acciones más elaboradas se realizará en futuros trabajos mediante la agregación de algoritmos más complejos dentro de la inscripción de arco que conecta el lugar *Rules_set* con la salida del controlador al Switch.

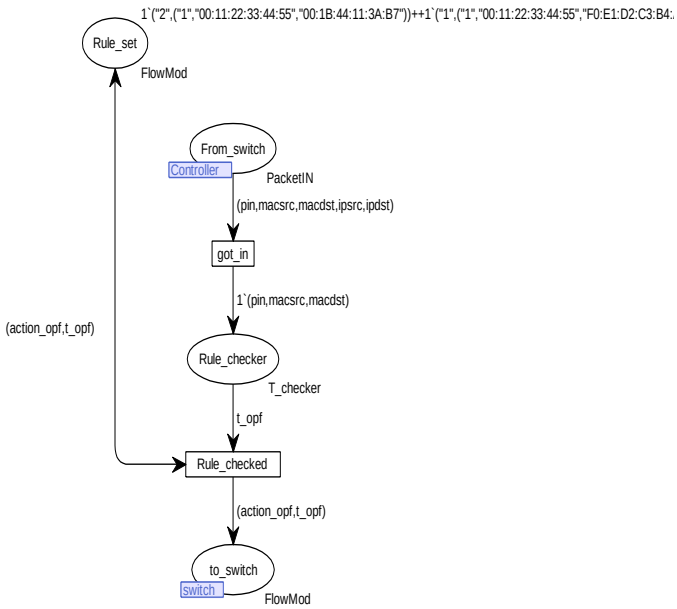

Figura 6. Modelo de controlador SDN en CPN

Lugares		Transiciones	
Rule_set	Conjunto de reglas del controlador	got_in	Recepción del flujo desde el switch por el controlador
From_switch	Flujo de entrada al controlador desde el switch		
Rule_checker	Búsqueda de una regla para el flujo de entrada que se está analizando	Rule_checked	Regla del controlador seleccionada para aplicar al flujo
To_switch	Flujo de salida del controlador hacia el switch		

Tabla 2. Descripción de las partes del modelo del controlador

3.3 Validación formal

Antes de implementar un análisis funcional de un modelo formalizado por medio de redes de Petri, es necesario realizar un análisis de sus propiedades que permita descartar comportamientos no deseados en la simulación. El cumplimiento o no de cualquiera de estas propiedades permite verificar si el modelo puede representar o no funcionalmente cierto comportamiento deseado.

Las siguientes propiedades son verificadas mediante el análisis del árbol de alcanzabilidad que permite validar formalmente el modelo. **(1)Limitación** Una red es limitada cuando todos sus lugares acumulan una cantidad limitada de marcas para cada marcaje alcanzable del árbol de alcanzabilidad. **(2)Vivacidad** Una red es Viva cuando para todo marcaje inicial, existe siempre un conjunto no vacío de transiciones sensibilizadas que permiten el marcaje continuar avanzando en los lugares de la red. **(3) Ciclicidad** describe la capacidad de una red para alcanzar un marcaje previo al marcaje actual mediante determinada secuencia de disparos.

Analysis of properties with CPN		
Places	Upper limit marking	Lower limit marking
Controller'From_switch	1	0
Controller'Rule_checker	1	0
Controller'Rule_set	1	1
Controller'to_switch	1	0
Swtich'Input_Flow	2	0
Swtich'Output_Flow	2	0
Swtich'controller	1	0
Swtich'from_controller	1	0
Swtich'in_controller	1	0
Swtich'is_in_table	1	0
Swtich'is_not_table	1	0
Swtich'packet	1	0
Swtich'sw_table	2	1
Swtich'table_checker	1	0
Analysis of liveness	Marking of lock: Node 28 of States space	
Analysis of cyclicity	Marks home: Node 28	

Tabla 3. Análisis de las propiedades del modelo con CPN Tools

De la tabla 3, al realizar el análisis de limitación, se puede observar que la red posee una cota superior e inferior de marcas para cada lugar de la CPN. Así mismo del análisis de vivacidad, se observa que la red posee un nodo de bloqueo, correspondiente al marcaje en el lugar que representa el flujo de salida del switch. Finalmente, el análisis de ciclicidad indica que la red no es cíclica ya que posee un único marcaje alcanzable desde todos los demás marcajes de la red, correspondiente al mismo marcaje de bloqueo.

4 Conclusiones

En este artículo se propuso un modelo básico de red SDN para el análisis de los diferentes estados del procesamiento de un flujos de datos, CPN demuestra ser una herramienta de fácil modelamiento y comprensión gracias a su intuitivo entendimiento del comportamiento concurrente y la habilidad de analizar patrones. Este modelo nos permite continuar en el proceso de diseño para lograr una aproximación mayor al comportamiento y consolidar a CPN no solo como una herramienta de simulación, sino también como un simulador de protocolos y topologías de comunicaciones.

5 Trabajos futuros

Los trabajos futuros se enfocarán en diseñar un marco experimental basado en CPN para la simulación de algoritmos de optimización. Para lograr este objetivo se requiere mejorar el modelo actual incluyendo: Modelo basado en tiempo utilizando redes de Petri basadas en tiempo, modelo que permita análisis estadístico del comportamiento usando redes de Petri estocásticasticas. Adicionalmente se espera trabajar en crear un esquema fácil para el diseño de múltiples topologías con los dispositivos ya modelados como el Switch y el controlador que pueda ser usado como simulador por la comunidad académica y permita el desarrollo de diseños de experimentos aprovechando las características de simulación de la herramienta CPN tools.

Referencias

[1] N. McKeown, "Software Defined Network." [Online]. Available: http://yuba.stanford.edu/.

[2] N. Foster, A. Guha, M. Reitblatt, A. Story, M. J. Freedman, N. P. Katta, C. Monsanto, J. Reich, J. Rexford, D. Walker, M. R. Harrison, and U. S. M. Academy, "Languages for Software-Defined Networks," *IEEE Comun. Mag.*, no. February, pp. 128–134, 2013.

[3] O. Fundation, "Software-defined networking: The new norm for networks," *ONF White Pap.*, 2012.

[4] H. E. Egilmez and S. T. Dane, "OpenQoS: An OpenFlow controller design for multimedia delivery with end-to-end Quality of Service over Software-Defined Networks," *Signal Inf. Process. Assoc. Annu. Summit Conf. (APSIPA ASC)*, pp. 1–8, 2012.

[5] a. Kassler and L. Skorin-Kapov, "Towards QoE-driven multimedia service negotiation and path optimization with software defined networking," *... Comput. Networks ...*, pp. 1–5, 2012.

[6] M. Jarschel, S. Oechsner, D. Schlosser, R. Pries, S. Goll, and P. Tran-gia, "Modeling and Performance Evaluation of an OpenFlow Architecture," *2011 23rd Int. Teletraffic Congr.*, pp. 1–7, 2011.

[7] A. Societies and S. Simulation, "Why Model ?," vol. 11, no. 4, 2008.

[8] L. Renardikres and M. Loing, "Methodology for LAN modeling and analysis using Petri nets based models," *Model. Anal. Simul. Comput. Telecommun. Syst. 1994., MASCOTS '94., Proc. Second Int. Work.*, pp. 335–342, 1994.

[9] M. Li-li, "analysis and verification of colored petri net in pppoe protocol," *3rd Int. Conf. Adv. Comput. Theory Eng.*, pp. 78–82, 2010.

[10] V. Gehlot and C. Nigro, "Colored Petri Net model of the Session Initiation Protocol (SIP)," *IECON 2010 - 36th Annu. Conf. IEEE Ind. Electron. Soc.*, pp. 2150–2155, Nov. 2010.

[11] J. Wang, J. Yang, G. Xie, and M. Zhou, "OSPFv3 protocol simulation with colored Petri nets," *... Proceedings, 2003. ICCT ...*, no. 60273070, 2003.

[12] M. Silva, "Half a century after Carl Adam Petri's Ph.D. thesis: A perspective on the field," *Annual Reviews in Control*, vol. 37. pp. 191–219, 2013.

[13] X. L. X. Liao, X. Z. X. Zhang, and J. J. J. Jiang, "Petri net-based modeling of switching arrangements and simulation," *IEEE Int. Conf. Mechatronics Autom. 2005*, vol. 3, 2005.

[14] K. Jensen and L. M. Kristensen, "Coloured Petri Nets: Modelling and Validation of Concurrent Systems," *Springer*, vol. 9, p. 384, 2009.

[15] O. López, M. A. Laguna, and F. J. García, "Representación de Requisitos mediante Redes de Petri Coloreadas," 2002.

[16] NOX, "NOX CONTROLLER." [Online]. Available: http://www.noxrepo.org/nox/about-nox/.

[17] D. Erickson, "The beacon openflow controller," *Proc. Second ACM SIGCOMM Work. Hot Top. Softw. Defin. Netw. - HotSDN '13*, p. 13, 2013.

[18] Project Floodlight, "http://www.projectfloodlight.org/floodlight/." .

[19] Open Networking Foundation, "SDN Product Directory," 2013. [Online]. Available: http://sdndirectory.opennetworking.org/products.

[20] R. Klöti, "Openflow: A security analysis," *Proc. Wkshp Secur. Netw. Protoc. (NPSec). ...*, 2013.

[21] U. Hoelzle., "OpenFlow @ Google," 2012. [Online]. Available: http://www.opennetsummit.org/archives/apr12/hoelzle-tue-openflow.pdf.

[22] HP, "HP Networking OpenFlow." [Online]. Available: http://h17007.www1.hp.com/us/en/mobile/solutions/enterprise/openflow.html.

[23] JUNIPER, "Juniper Networks. OpenFlow Switch Application (OF-APP) for Juniper MXSeries Routers." [Online]. Available: https://developer.juniper.net/shared/jdn/docs/ProgrammableNetworks/OpenFLow_APP_JDN_Overview.pdf .

[24] C. Open, N. Environment, and W. Series, "Cisco Open Network Environment Webinar Series An Introduction to OpenFlow :," 2013.

[25] M. Antonio, T. Rojas, E. T. Ueda, T. Cristina, and M. De Brito, "Modelling and Verification of Security Rules in an OpenFlow Environment with Coloured Petri Nets," *Inf. Syst. Technol.*, vol. 9th Iberia, pp. 1 – 7, 2014.

[26] L. Dong, H. Li, N. He, and Y. Xing, "Testing OpenFlow interaction property based on hierarchy CPN," *Proc. - Int. Conf. Netw. Protoc. ICNP*, pp. 4–5, 2013.

[27] H. E. Egilmez, S. Civanlar, and a. M. Tekalp, "An optimization framework for QoS-enabled adaptive video streaming over openflow networks," *IEEE Trans. Multimed.*, vol. 15, no. 3, pp. 710–715, 2013.

[28] K. Jensen and L. M. Kristensen, "Introduction to Modelling and Validation," in *Coloured Petri Nets - Modelling and Validation of Concurrent Systems*, Springer Berlin Heidelberg, 2009, pp. 1–12 LA – English.