MARE INGENII. INGENIERÍAS

Evaluación de conmutadores programables virtuales para redes definidas por software

Virtual programmable switch evaluation for Software Defined Networking

https:/ /doi.org/10.52948/mare.v4i2.673

YANKO ANTONIO MARÍN MURO
http://orcid.org/0000-0001-8837-0169
Empresa de Telecomunicaciones de Cuba (ETECSA), Cuba
yanko.marin@etecsa.cu

FÉLIX F. ÁLVAREZ-PALIZA
http://orcid.org/0000-0003-3243-8923
Universidad Central "Marta Abreu" de Las Villas (UCLV), Cuba
fapaliza@uclv.edu.cu

JAVIER RAFAEL GÓMEZ VALDIVIA
https://orcid.org/0000-0003-0248-7660
Empresa de Telecomunicaciones de Cuba (ETECSA), Cuba
javier.gomez@etecsa.cu

EDUARDO RANCÉS PÉREZ TARDÍO
https://orcid.org/0000-0003-2507-9969
Empresa de Telecomunicaciones de Cuba (ETECSA), Cuba
eduardo.perezt@etecsa.cu

Artículo de investigación
Recepción: 31 de marzo de 2022
Aceptación: 11 de noviembre de 2022

Cómo citar este artículo
Y. Marín Muro, F. Álvarez-Paliza, J. Gómez Valdivia, y E. Pérez Tardío, Evaluación de conmutadores programables virtuales para redes definidas por software", mare, vol. 4, n.º 2, sep. 2022.

Resumen:
En las Redes Definidas por Software, el protocolo OpenFlow implementa una abstracción de los elementos de red y proporciona flexibilidad al extraer el plano de control de los dispositivos de la capa de infraestructura. OpenFlow posee algunas limitaciones que están relacionadas con la poca flexibilidad del protocolo en su interface hacia el sur, por lo que actualmente se demandan soluciones más robustas. Debido a estas limitaciones, el lenguaje específico de dominio P4 (Programación de Procesadores de Paquetes independientes del Protocolo) está ganando impulso en los sectores académico e industrial. P4 proporciona un conjunto básico de herramientas para que el programador implemente protocolos en el hardware de conmutación. P4Runtime es una API abierta del plano de control de las redes definidas por software de próxima generación; tiene como objetivo solucionar las deficiencias de OpenFlow, proporcionando una verdadera independencia de hardware de conmutación para los operadores y proveedores de servicios en la nube. Los procedimientos para medir el desempeño de las redes definidas por software basadas en la API P4Runtime es un tema de investigación abierto en la actualidad. Este artículo proporciona una descripción técnica general de los protocolos P4 y P4Runtime, destacando sus beneficios. Finalmente, es implementado un procedimiento que permite cuantificar el del desempeño de este tipo de redes considerando métricas de escalabilidad, latencia, utilización de memoria y carga de CPU. Para realizar la evaluación fueron creados varios scripts en el Bash Shell de Linux. Se registró un deterioro significativo en el retardo de la red (ICMP RTT) en ONOS-P4Runtime-BMv2 en comparación con la red REF-SDN-OpenFlow.

Palabras clave: redes definidas por software; planos de datos programables; BMV2; OpenFlow; P4; P4Runtime; ONOS.

Abstract:
Through abstracting network elements, the OpenFlow protocol implements the Software Defined Networking (SDN) concept and provides flexibility by moving the control plane from the data plane. The OpenFlow protocol has some limitations that rely on the poorly flexible protocol abstraction as southbound and there has been a need for more robust solutions. Because of these constraints, Programming Protocol-Independent Packet Processors (P4) domain-specific language is gaining momentum in the academic and industrial sectors. P4 provides a basic set of tools for the programmer to implement a network stack in switching hardware. The new language features allow more forwarding devices description, not only programmable but also conventional fixed-function devices. P4Runtime introduced as the next-generation SDN open control plane API, which aims to fix OpenFlow deficiencies and is capable of providing true switching silicon independence for carriers and cloud service providers. Measuring the performance of P4Runtime API-based SDN networks is an open issue at present. This paper provides a technical overview of P4 and P4Runtime, highlighting its benefits. Finally, it performs an analysis of these types of networks considering metrics of scalability, latency, memory utilization and CPU load. This research created several scripts in the Linux bash Shell to perform this evaluation. Finally, this work recorded a significant deterioration of ICMP RTT delay in ONOS-P4Runtime-BMv2 compared to REF-SDN-OpenFlow network.

Keywords: Software Defined Networking; programmable data-planes; BMV2; OpenFlow; P4; P4Runtime; Open Network Operating System.

Introducción

Las redes de comunicaciones tradicionales se construyen a partir de una gran cantidad de dispositivos de red que realizan diversas tareas, como conmutación, enrutamiento, mantenimiento de la calidad del servicio, monitoreo y administración, garantizando la seguridad y la confiabilidad, etc.

Los nodos de red se han desarrollado tradicionalmente para propósitos extremadamente específicos: soportar un conjunto dado, y posiblemente pequeño, de necesidades de comunicación o reenvío [1].

Desafortunadamente, las redes actuales son extremadamente complejas y diversificadas. Los nodos de Internet, históricamente limitados a conmutadores y enrutadores, se han complementado masivamente con una variedad de equipos intermedios heterogéneos como traductores de direcciones de red, entidades de túnel, balanceadores de carga, cortafuegos, sistemas de detección de intrusos, sondas de monitoreo de tráfico, entre otros [2].

OpenFlow es el primer protocolo de comunicación de código abierto estándar entre el plano de control y de datos de la Open Networking Foundation (ONF, por sus siglas en inglés) para promover la adopción de redes definidas por software (SDN, por sus siglas en inglés) [3]. Este protocolo actualmente se enfrenta a algunos problemas entre los que se destaca el costo de las soluciones, la demora al implementar los estándares después de agregar nuevos protocolos, entre otros.

Por otro lado, el plano de datos de OpenFlow posee una estructura pipeline fija cuyo modelo de reenvío se basa en la búsqueda de tablas de flujo y la ejecución de acciones asociadas. La capacidad de programación del plano de datos de OpenFlow se limita al contenido de la tabla de flujo, lo que restringe su flexibilidad [4].

La comunidad SDN ha estado trabajando en función de reducir el tiempo necesario para implementar nuevos protocolos y ampliar los existentes. Las plataformas de red programables son el factor habilitador que permite desarrollar funcionalidades de red complejas sin comprometer los niveles de rendimiento que alcanzan hoy los dispositivos de red de propósito fijo. Estas plataformas deben proporcionar varios aspectos comunes en los diferentes dominios de red (plano de datos, acceso óptico y de radio) [5].

La iniciativa más popular en este nuevo campo es la programación de procesadores de paquetes independientes del protocolo (P4) [3], [6]–[9]. El impulso del P4 comenzó con la observación de que, si bien en el pasado los conmutadores de red tenían un comportamiento fijo y conocido, hoy en día está surgiendo una nueva generación de arquitecturas de conmutadores de alta velocidad totalmente reprogramables [10].

El lenguaje específico de dominio P4 se concibió para expresar cómo son procesados los paquetes por el plano de datos en un conmutador programable por hardware o software, tarjeta de interfaz de red, enrutador o dispositivo de red [8].

Este lenguaje ha surgido como un intento de describir mediante programación el proceso de procesamiento de paquetes a través de "programas de paquetes" escritos en un lenguaje de alto nivel que puede compilarse para diferentes dispositivos.

P4 fue diseñado con la intención de describir el comportamiento de un conmutador. Su objetivo es permitir a los propietarios de redes desarrollar su propio software orientado a aplicaciones para un conmutador programable. Para poder ejecutar la

descripción desarrollada en el hardware se requiere un compilador o un intérprete y P4 lo traduce en un programa ejecutable [11].

P4 aborda un problema más general dentro de la red: en lugar de simplemente crear una interface entre el controlador y el plano de datos define el comportamiento de este último. Las plataformas de red programables deben entonces proporcionar modelos de programación claros que permitan el desarrollo de funciones de red.

El mundo de las redes se ha definido en gran medida por el desarrollo de chips de funciones fijas. Si bien estos dispositivos han sido uno de los medios por los cuales las velocidades de datos han aumentado, esto se ha logrado a expensas de la programabilidad [9].

P4Runtime [11] es una nueva forma para que el software del plano de control controle el de reenvío de un conmutador, enrutador, firewall, equilibrador de carga, etc. Quizás el aspecto más novedoso de P4Runtime es que le permite controlar cualquier plano de reenvío, independientemente de si es construido a partir de un conmutador ASIC de función fija o programable, una matriz de puertas lógicas programable en campo (FPGA, por sus siglas en inglés), una Unidad Neutral de Procedimiento (NPU, por sus siglas en inglés), o un conmutador por software que se ejecuta en un servidor x86.

P4 y P4Runtime están ganando impulso debido a la participación de grandes jugadores como Google AT&T, CISCO, Barefoot, ARISTA y otros. P4 tiene el potencial de causar un cambio significativo en la industria y cumplir con la propuesta de valor de SDN.

La motivación de la investigación es comprender los potenciales SDN-P4Runtime que soportan los pilares de las redes futuras y comparar estos paradigmas con SDN-OpenFlow y las redes tradicionales.

Planteamiento del problema.

Dadas las limitaciones y los valores de las tecnologías existentes, una serie de preguntas fundamentales permanecen abiertas:

- ¿Cuál es el máximo tamaño de una red SDN basada en P4Runtime/OpenFlow y utilizando el emulador Mininet?
- ¿Cuánto tiempo llevará compilar toda la red?
- ¿Cuál es el conmutador con el menor tiempo de compilación de red?
- ¿La topología influye en el tiempo de compilación de la red SDN-P4Runtime?

Por otra parte, nuestras principales contribuciones son:

1. El desarrollo de varias aplicaciones en Linux Bash Shell con el objetivo de automatizar las pruebas de escalabilidad en Mininet a diferentes tipos de topologías de red ONOS-P4Runtime-BVm2; así como la medición en cada prueba de varias métricas en el hipervisor donde se ejecuta el controlador.
2. Una plataforma de pruebas experimentales está disponible. Esta permite el estudio y evaluación de las redes SDN basadas en P4Runtime y OpenFlow; además del desarrollo de aplicaciones.

Metodología

Para evaluar las redes programables y realizar experimentos en la arquitectura SDN es esencial tener un entorno para las pruebas. También es necesario implementar un procedimiento que permita evaluar y comprender el comportamiento de las redes programables en las que se utiliza P4/P4Runtime, comparándolo con el de las redes SDN-OpenFlow y las tradicionales.

Para lograr este objetivo se utiliza una red real para emular este tipo de redes, así como la presentación de diferentes casos de uso o demostraciones.

La investigación se desarrolla en un entorno VM de Open Network Operating System (ONOS) P4 Brigade, configurado con Mininet y ONOS para controlar dispositivos con capacidad P4 a través de P4Runtime. En este escenario se crean varios tipos de topologías SDN-OpenFlow, SDN-P4Runtime y de redes tradicionales, comparándose diferentes métricas de desempeño a través de diferentes casos de uso. La figura 1 muestra el entorno en el que se llevan a cabo los experimentos y es donde se aplica el procedimiento de evaluación creado.

Debido a la naturaleza aún nueva de los paradigmas SDN, P4 y P4Runtime, el proceso de evaluación de un controlador resulta complejo. También se identifican las métricas de desempeño que generalmente se usan para evaluarlas.

Las métricas más importantes que inciden en el desempeño del controlador ONOS utilizando la API P4Runtime son las siguientes: servicio de traducción, operaciones de flujo e interprete pipeline.

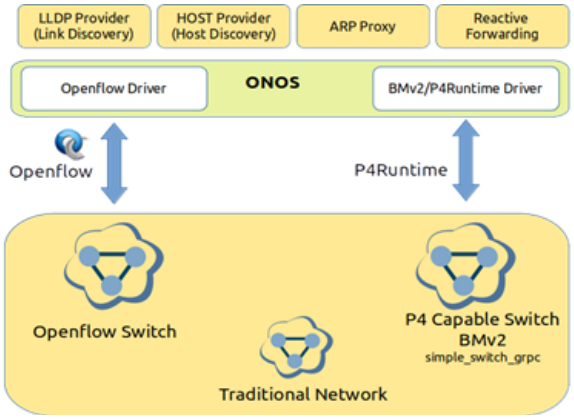

Fig. 1. Entorno de los experimentos.

Las métricas de desempeño de ONOS son provisión inicial del conmutador en caso de reinicio; cambio de ruta al ocurrir una falla; insertar/actualizar rutas; conciliación de reglas de flujo; leer el estado del dispositivo y comparar con el almacenado en ONOS.

Existen muchas métricas definidas para redes tradicionales y SDN-OpenFlow igualmente válidas para SDN-P4: tiempo de ejecución, desempeño de las operaciones de lectura/escritura, latencia para operaciones de escritura por lotes y utilización de memoria. Aunque las métricas que son específicas de tales redes también han sido definidas por la ONF, las métricas de escalabilidad, retardo, la CPU y la memoria se utilizan en este trabajo.

Para eliminar o minimizar el retardo ICMP RTT entre el conmutador y ONOS, idealmente se ejecutó el proceso ONOS y P4Runtime en la misma máquina virtual.

La topología fue diseñada modificando un script Python bmv2.py que es capaz de llevar la configuración a Mininet para simular y comparar indicadores clave de desempeño de SDN-OpenFlow, SDN-P4Runtime y el entorno de red tradicional.

Para realizar la evaluación del desempeño de las redes SDN-P4Runtime se ejecuta el siguiente procedimiento general:

- $ cd $ONOS_ROOT
- $ONOS_APPS=proxyarp, hostprovider, lldpprovider ok clean

Esto inicia ONOS e instala las siguientes aplicaciones: proxy ARP/NDP, proveedor de enlaces como Link Layer Discovery Protocol (LLDP), proveedor de ubicación de host y enlaces LLDP.

Posteriormente se activa el controlador BMv2 y pipeconf a través de ONOS CLI.

- $ onos localhost
- onos> app activate org.onosproject. drivers. bmv2
- onos> app activate org.onosproject. p4tutorial.pipeconf
- onos> app activate org.onosproject. fwd
- onos> cfg set org.onosproject.net. flow.impl.Flow RuleManager fallback FlowPoll Frequency 10

Ahora bien, para crear topología simple SDN-P4Runtime:

- sudo -E mn --custom $BMV2_MN_PY --switch onosbmv2, pipeconf=p4-tutorial-pipeconf --controller remote, ip=127.0.0.1 --topo single,2
- Donde n es: 2, 4, 8, 16, 32, 64...

Por otra parte, para crear topologías lineales SDN-P4Runtime:

- sudo -E mn --custom $BMV2_MN_PY --switch onosbmv2, pipeconf=p4-tutorial-pipeconf --controller remote, ip=127.0.0.1 --topo linear,n
- Donde n es: 2, 4, 8, 16, 32, 64...

Después de crear la topología de red, se invoca el script de la figura 2 utilizando el siguiente mando:

- mininet> h1 bash /home/sdn/simulaciones/ bash_script /host_script/host_script 10.0.0.n
- Donde n es: 2, 4, 8, 16, 32, 64 ...

Con este procedimiento se consultan métricas de desempeño importantes como retardo, CPU y memoria utilizada.

host_script

```
#!/bin/bash
memoria=$(sudo free -m | grep ^Mem)
CPU_UTILIZACION=$(sudo top -b -n2 -p 1 | fgrep "Cpu(s)" /
| tail -1 | awk -F'id,' -v prefix="$prefix" '{ split($1, vs, ",");/
v=vs[length(vs)]; sub("%", "", v); /
printf "%s%.1f%%\n", prefix, 100 - v }')
HORA=$(date "+%Y-%m-%d %H:%M:%S")
rping=$(ping -c 10 $1 | grep "rtt\|packets transmitted")
resultado_final=$(echo "Resultado-;/
$memoria;$CPU_UTILIZACION;$rping" | sed -e 's/\r//g')
echo $resultado_final
```

Fig. 2. Script para consultar memoria, CPU y retardo ICMP RTT.

En este trabajo se desarrolló un script que puede evaluar varias topologías automáticamente y registrar en tiempo real importantes métricas de desempeño.

Aquí se repite el mismo procedimiento, pero se utiliza el controlador de referencia Mininet para evaluar las redes SDN utilizando el protocolo OpenFlow V1.0. Posteriormente, se grafican los resultados.

Planteamiento del problema.

Con el objetivo de evaluar los paradigmas de red P4, P4Runtime, OpenFlow y tradicional, fue creada una VM utilizando VirtualBox 6.0.12 r132055. En la VM se instaló ONOS, Mininet, P4 y Wireshark para evaluar diferentes tipos de redes.

Medir el desempeño de las redes SDN programables basadas en la API P4Runtime es un tema abierto en la actualidad. En este trabajo se propone crear redes de diferentes topologías, tamaños y estudiar métricas de escalabilidad, retardo RTT ICMP, utilización de memoria y CPU para comparar el comportamiento de los diferentes tipos de paradigmas abordados en este trabajo.

ITEM	CHARACTERISTICS
Host Processor	Intel(R) Core(TM) i3-2100 CPU @ 3.10GHz
Host Installed memory:	8 GB
Host Virtualization Activated	SI
Host OS	Arch Linux 2019.11.01 Included Kernel: 5.3.8
Host Hypervisor	VirtualBox Versión 6.0.12 r132055
Guest OS	Ubuntu 16.04.
Number of CPUs assigned to the VM	2 CPU
Memory allocated to VM	4 GB

Tabla 1. Características generales del escenario.

Los casos de uso que se describen a continuación permiten realizar una evaluación cuantitativa del desempeño de estas redes.

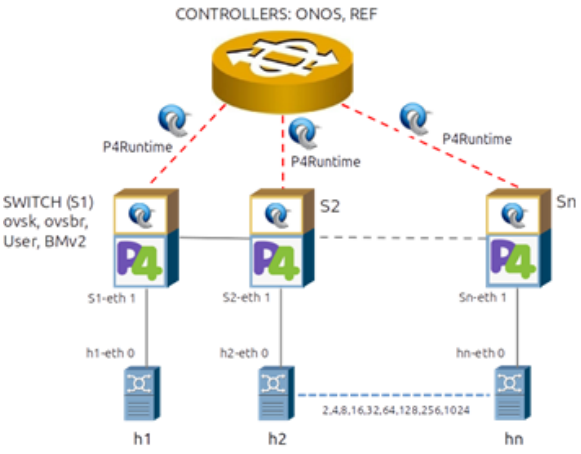

Fig. 3. Modelo del sistema (topología lineal). .

Evaluación de las SDN basada en P4Runtime y OpenFlow.

A continuación, se valida y evalúa la implementación de una red SDN-P4Runtime utilizando conmutadores que soportan P4 y utilizando el controlador ONOS.

La escalabilidad es el número máximo de dispositivos y/o conmutadores que puede manejar un controlador sin reducir su desempeño. Idealmente, la cantidad de conmutadores conectados no debería afectar el desempeño de un controlador SDN. Sin embargo, el aumento en los subprocesos de ejecución, la dinámica del protocolo de control de transmisión (TCP, por sus siglas en inglés) y las tareas que se ejecutan en el núcleo del controlador son factores que pueden degradar su desempeño si hay una gran cantidad de conmutadores activos.

Una prueba computarizada para medir el desempeño de la tecnología se denomina "evaluación comparativa". Las variables para analizar podrían incluir latencia, razón de transferencia de datos, carga de CPU, memoria utilizada, etc. Realizar pruebas de verificación o aceptación hoy en día es algo que las empresas hacen a diario antes de decidir si comprar o no una tecnología, implementar o ejecutar cambios en las topologías de red. Antes de comprar cualquier equipo se someten a pruebas que permiten la simulación de altas cargas de trabajo y, por lo tanto, a través de aplicaciones de software se estudia el comportamiento de cada una de las variables estudiadas.

Para evaluar la capacidad de escalar grandes redes con numerosos hosts y conmutadores, en SDN se realizan pruebas de escalabilidad (evaluación comparativa).

Para realizar estas pruebas se crearon varios scripts en Linux Bash Shell con el fin de crear nodos y topologías de forma automática, simultánea e interactuando con el emulador Mininet.

La realización de los scripts fue motivada porque Mininet permite simular una sola topología en un momento dado. Al mismo tiempo entrega los datos en un formato difícil de procesar cuando muchos escenarios necesitan ser simulados.

Otra cuestión importante es que, por lo general, cuando Mininet compila la topología no proporciona datos relacionados con el impacto del sistema, vinculado con el aumento de subprocesos y la ocupación de la memoria. Por consiguiente, en el script realizado se consulta la ocupación de la memoria en cada compilación para tener referencias al analizar posibles degradaciones del desempeño.

El script creado ofrece el tiempo que lleva crear y destruir la red, además del uso de memoria, CPU e ICMP RTT. En la figura 2 aparece un fragmento del script creado para realizar pruebas de escalabilidad a la topología simple de Mininet.

Estudiar el retardo de la compilación de la red permite investigar el tiempo que Mininet tarda en iniciar el controlador y cada nodo de red; además del tiempo de establecer la conexión de cada conmutador con el controlador a través de P4Runtime u OpenFlow.

Para llevar a cabo esta prueba se han establecido los siguientes objetivos:

- Comprobar la cantidad máxima de dispositivos y/ o conmutadores en una red que puede manejar el controlador SDN lógicamente centralizado sin sacrificar el desempeño al simular diferentes escenarios las SBI OpenFlow y P4Runtime. Para una planificación óptima de la red es clave medir el tamaño máximo de red que un controlador puede descubrir.
- Determinar el número máximo de nodos en una red SDN que se pueden simular utilizando Mininet con topologías simples y lineales y las SBI OpenFlow y P4Runtime.

- Determinar el modelo de switch que muestra los mejores resultados y el escenario con el mejor desempeño al analizar las métricas de escalabilidad y el tiempo requerido para crear y destruir la red SDN.
- Explorar las limitaciones del emulador Mininet cuando aumenta el número de nodos en la red.

Este trabajo se basa en la especificación P4_16 y la evaluación se realizó en el conmutador P4-BMv2.

Los conmutadores OVSK, OVSBR y user de Mininet se consideran en los scripts generados que se conectaron al controlador de referencia formando topologías simples y lineales.

En el momento de utilizar la API P4Runtime todas estas topologías se configuran con el conmutador BMv2 conectado al controlador ONOS. La cantidad de elementos aumenta en cada compilación hasta que se obtiene el valor máximo de los nodos en el sistema sin degradar el desempeño.

Aunque Mininet se considera una herramienta excelente y conveniente, realmente tiene algunas limitaciones porque no puede garantizar una alta fidelidad y desempeño cuando la carga de trabajo es muy alta. En estos experimentos se verificará el número máximo de nodos que el emulador puede manejar en cada escenario. Debido a que Mininet se ejecuta en un solo sistema, existe una limitación de recursos de procesamiento y memoria de hardware que impone una limitación en el tamaño real de las redes que se pueden escalar.

Resultados y discusión

La figura 4 representa el resultado de las pruebas de escalabilidad en las topologías lineal, simple y muestra.

El número máximo de dispositivos y/o conmutadores en una red SDN-P4Runtime-BMv2 SINGLE que puede manejar un controlador ONOS lógicamente centralizado, sin sacrificar el desempeño, es de 65 nodos (1 conmutador BMv2 y 64 hosts). En adición, a partir del 65 la red se vuelve inestable.

El tamaño máximo de una red lineal SDN-P4Runtime-BMv2 del entorno presentado en este trabajo es de 128 nodos (64 conmutadores BMv2 y 64 hosts). El límite de memoria de la VM impide continuar escalando la red. Además, a partir de 128 nodos la red se vuelve inestable.

El tiempo necesario para crear pequeñas redes virtuales SDN-P4Runtime-BMv2 es muy pequeño. El tiempo aumenta a medida que incrementan los nodos virtuales.

En los conmutadores OVSK y OVSBR conectados al controlador de referencia Mininet con SBI OpenFlow se alcanza el mayor número de nodos (1025) en la topología simple.

Por su parte, con el conmutador OVSK, se alcanza el mayor número de nodos (1024) en la topología lineal.

Al comparar los resultados de los dos tipos de topologías y escenarios se obtiene que el tiempo necesario para crear una red virtual aumenta en todos los casos con el incremento en el número de nodos virtuales.

Los resultados anteriores contribuyen a la futura planificación de redes de este tipo al comprobar el tamaño máximo de la red que un controlador ONOS puede descubrir. La figura 5 muestra que ONOS descubre una topología lineal con tres conmutadores BMv2.

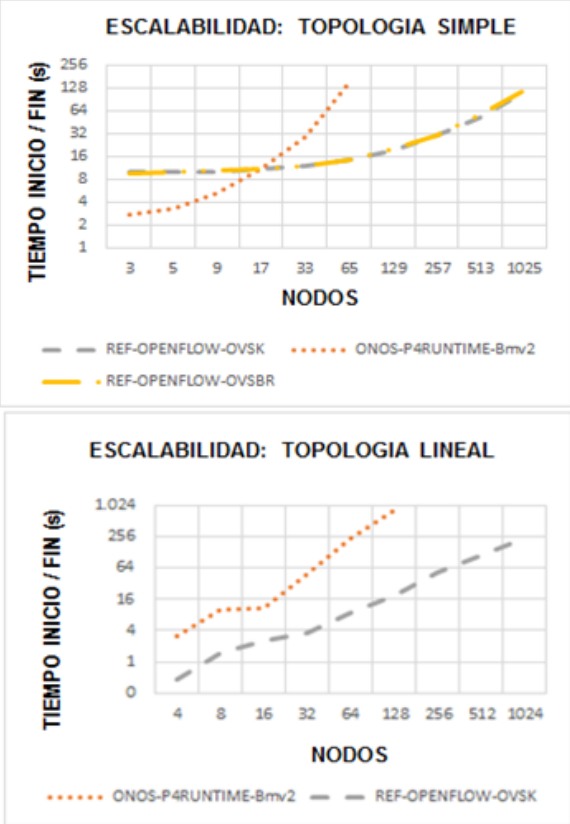

Fig. 4. Prueba de escalabilidad utilizando topología lineal y simple.

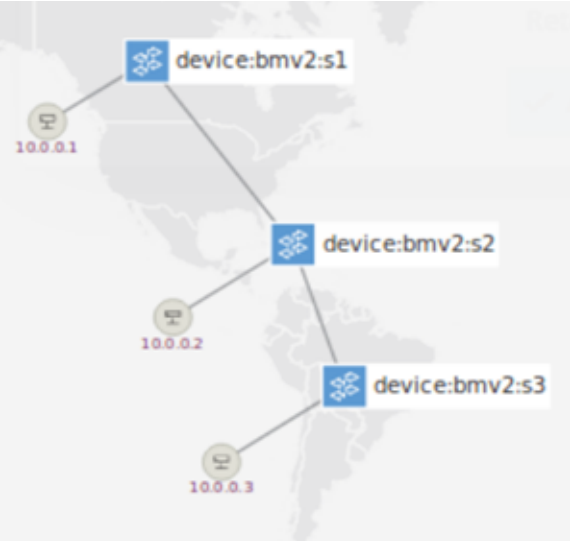

Fig. 5. ONOS descubre automáticamente los conmutadores BMv2 y los hosts de la red.

Con respecto al uso de la CPU (figura 6), en las topologías SDN-P4Runtime-BMv2 se observa que la carga de la CPU aumenta a medida que la red crece. Pero con 128 nodos en una red lineal no supera el 21% de utilización de la CPU.

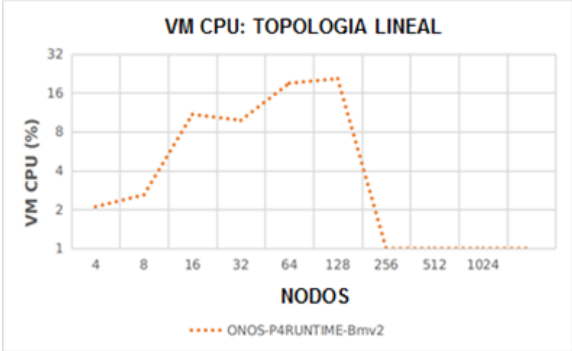

Fig. 6. VM CPU..

Con respecto al uso de memoria (figura 7), se encuentra que para redes pequeñas permanece constante y comienza a crecer linealmente a medida que la red se expande.

El límite de memoria asignado a la máquina virtual impide continuar escalando la red. También es importante enfatizar que Mininet posee una clara dependencia de los recursos de la VM para realizar la emulación. Al asignar más recursos a la VM se pueden escalar redes más grandes. La diferencia en una topología simple con tres y 65 nodos es solamente de 281MB. Por el contrario, en una topología lineal con cuatro y 128 nodos esta misma diferencia es de 1123MB.

La figura 8 muestra que el retardo entre h1 y hn se comporta casi constante a medida que aumenta el número de hosts en la topología simple ONOS-P4Runtime-BMv2. En esta topología el retardo mínimo de la red ONOS-P4Runtime-BMv2 aumentó en un 880,3% en comparación con la REF-OpenFlow-OVSK. El retardo mínimo de la red ONOS-P4Runtime-BMv2 nunca supera los 0,420 ms.

En la topología lineal el valor máximo del retardo mínimo del escenario SDN-P4Runtime-BMv2 aumentó en 110682% en comparación con el del REF-OpenFlow-OVSK. El retardo mínimo de la red ONOS-P4Runtime-BMv2 nunca supera los 26 ms.

La figura 9 muestra el retardo promedio en las redes ONOS-p4Runtime-BMv2 para las topologías simple y lineal. El retardo entre h1 y hn aumenta linealmente en la topología lineal hasta alcanzar 65 nodos. Con 128 nodos el retardo promedio de la red aumenta considerablemente hasta alcanzar valores de 572 ms. En la topología lineal, el valor máximo del retardo promedio en el escenario SDN-P4Runtime-BMv2 aumentó en un 511498% en comparación con el REF-OpenFlow-OVSK.

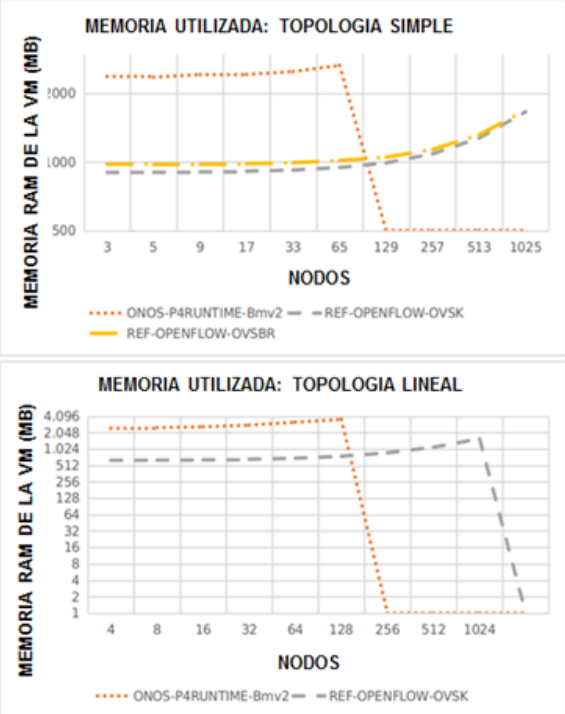

Fig. 7. Memoria RAM utilizada en la VM.

Para la topología simple el valor máximo del retardo promedio en la red Onos-P4Runtime-BMv2 aumenta en 208% cuando se compara con el REF-OpenFlow-OVSK. El retardo promedio de la red en la topología simple ONOS-P4Runtime-BMv2 nunca supera los 3,08 ms.

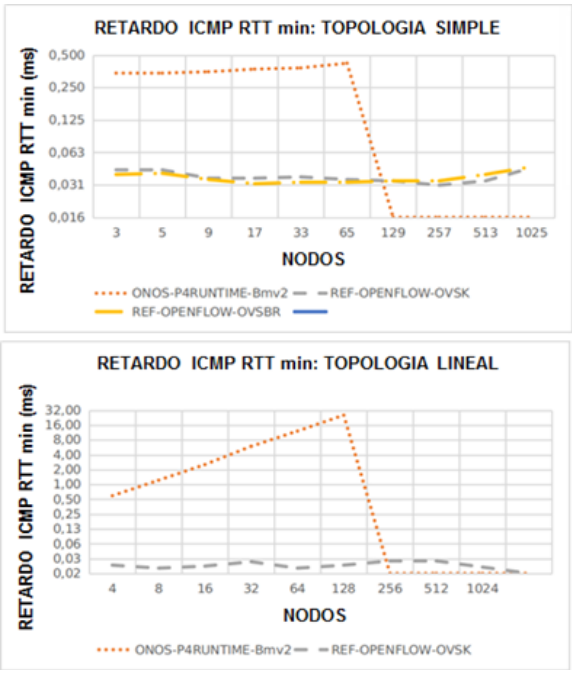

Fig. 8. Retardo ICMP RTT mínimo entre h1 y hn.

Un deterioro significativo en el retardo ICMP RTT fue registrado en la topología ONOS-P4Runtime-BMv2, en comparación con la REF-OpenFlow-OVSK. Este comportamiento se produce porque Bmv2 no es un conmutador para soluciones en producción.

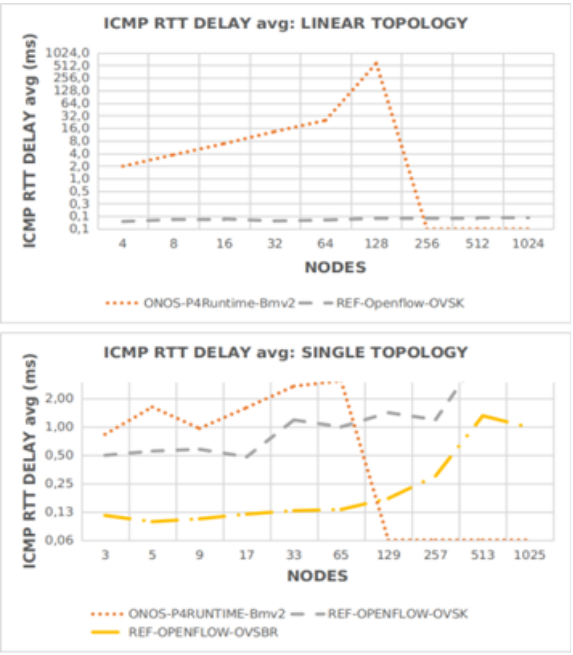

Fig. 9. Retardo ICMP RTT promedio entre h1 y hn.

Conclusión

El lenguaje P4 y la API P4Runtime tienen el potencial de convertirse en un instrumento disruptivo para programar y personalizar el plano de datos de los dispositivos de red SDN.

Las SDN basadas en P4, P4Runtime y OpenFlow se han distinguido en los últimos años como una arquitectura con un gran potencial para reemplazar las redes tradicionales. Esto sugiere la necesidad de profundizar su estudio y procedimientos para su evaluación.

Debido a la naturaleza aún nueva de los paradigmas SDN, P4 y P4Runtime, el proceso de evaluación de un controlador resulta muy complejo.

En los experimentos realizados fue registrado un deterioro significativo del retardo ICMP RTT en la topología ONOS-P4Runtime-BMv2 en comparación con la red REF-SDN-OpenFlow. Este comportamiento se produce porque el conmutador Bmv2 no está destinado a ser un software de producción según expertos de la ONF. Para futuras investigaciones se recomienda una evaluación de los dispositivos de conmutación de producción con soporte para P4 y P4Runtime.

En adición, el número máximo de dispositivos y/o conmutadores en una red ONOS-P4Runtime-BMv2 simple sin deterioro en el desempeño es de 65 nodos (1 conmutador BMv2 y 64 hosts).

Asimismo, el tamaño máximo de una red LINEAL SDN ONOS-P4Runtime-BMv2 en un entorno como el presentado en este trabajo es de 128 nodos (64 conmutadores BMv2 y 64 hosts). El límite de memoria de la VM impide continuar escalando la red. A partir de 128 nodos la red se vuelve inestable.

En la topología simple, ONOS-P4Runtime-BMv2 requirió menos tiempo de compilación que REF-OpenFlow-OVSK hasta nueve nodos. REF-OpenFlow-OVSK en la topología lineal tiene el mejor desempeño en tiempo de compilación de la red.

En este trabajo se verificó el impacto de la topología de las redes SDN en el tiempo de compilación y la latencia en la red.

Al completar este informe se dispone de una plataforma de prueba experimental y un procedimiento que permite el estudio y la evaluación de redes SDN basadas en P4Runtime y OpenFlow; además del desarrollo de aplicaciones.

Por último, los scripts desarrollados en este documento contribuyen a evaluar el desempeño de las redes SDN basadas en OpenFlow y P4Runtime.

Referencias

[1] "CAGRE 2019 Plenary Sessions", en *2019 Algerian Large Electrical Network Conference (CAGRE),* 2019, pp. 1-3. doi: 10.1109/CAGRE.2019.8713309

[2] F. Musumeci, A. C. Fidanci, F. Paolucci et al., "Machine-Learning-Enabled DDoS Attacks Detection in P4 Programmable Networks", *J. Netw. Syst. Manag.*, vol. 30, n.o 21, p. 21, en. 2022, doi: 10.1007/s10922-021-09633-5

[3] Z. Latif, K. Sharif, F. Li et al., "A comprehensive survey of interface protocols for software defined networks", *J. Netw. Comput. Appl.*, vol. 156, abr. 2020, doi: 10.1016/j.jnca.2020.102563

[4] T. Osinski, H. Tarasiuk, L. Rajewski et al., "DPPx: A P4-based Data Plane Programmability and Exposure framework to enhance NFV services", en *2019 IEEE Conference on Network Softwarization (NetSoft)*, París, Francia, jun. 2019, pp. 296-300. doi: 10.1109/NETSOFT.2019.8806625

[5] Q. Zuo, M. Chen, K. Ding et al., "On generality of the data plane and scalability of the control plane in software-defined networking", *China Communications,* vol. 11, n.o 2, pp. 55-64, feb. 2014, doi: 10.1109/CC.2014.6821737.

[6] A. AlSabeh, J. Khoury, E. Kfoury et al., "A survey on security applications of P4 programmable switches and a STRIDE-based vulnerability assessment", *Computer Networks,* vol. 207, abr. 2022, doi: 10.1016/j.comnet.2022.108800

[7] S. Shakeri, L. Veen y P. Grosso, "Multi-domain network infrastructure based on P4 programmable devices for Digital Data Marketplaces", *Cluster Computing,* en. 2022, doi: 10.1007/s10586-021-03501-2.

[8] P. Bosshart, D. Daly, G. Gibb et al., "P4: programming protocol-independent packet processors", *ACM SIGCOMM Comput. Commun. Rev.*, vol. 44, n.o 3, pp. 87-95, jul. 2014, doi: 10.1145/2656877.2656890

[9] Z. Li, Y. Hu, J. Wu et al., "P4Resilience: Scalable Resilience for Multi-failure Recovery in SDN with Programmable Data Plane", *Computer Networks,* vol. 208, may. 2022, doi: 10.1016/j.comnet.2022.108896

[10] E. Kaljic, A. Maric, P. Njemcevic et al., "A Survey on Data Plane Flexibility and Programmability in Software-Defined Networking", *IEEE Access,* vol. 7, pp. 47804-47840, 2019, doi: 10.1109/ACCESS.2019.2910140.

[11] A. Burnic, C. Spiegel, A. Viessmann et al., "Designing Terminals and Infrastructure Components for Cognitive Wireless Networks", en *2007 15th IEEE Workshop on Local & Metropolitan Area Networks,* Princeton, EE. UU, jun. 2007, pp. 117-122. doi: 10.1109/LANMAN.2007.4295985