Evaluación de performance en Redes Definidas por Software

Diego Bolatti, Ricardo Calcagno,
Carlos Cuevas, Sergio Gramajo, Reinaldo Scappini

Grupo Ingeniería en Sistemas de Información, Universidad Tecnológica Nacional Facultad Regional Resistencia,
French 414, (3500) Chaco, Argentina. Tel. 362-4432683
{dbolatti, rcalcagno, cac, sergio, rscappini} @frre.utn.edu.ar

Resumen

El objetivo principal de este artículo es presentar una metodología sencilla para la simulación de una SDN (Software Defined Networking) [1], configurada con el protocolo OpenFlow [2] controladores POX [3] y ODL [4], utilizando la herramienta de simulación Mininet [5], y realizar pruebas de performance mediante la utilización de IPERFv3 [6], mostrando los resultados con el software GNUPlot [7], todos ellos con licencia GNU. Además, se describen los requerimientos y recomendaciones fundamentales para dicha emulación, y se presentan los resultados de pruebas sencillas con el fin de verificar la conectividad, transferencia de datos y operación sobre una topología de prueba. Se orienta al desarrollo de modelos de evaluación de performance que ayuden a los administradores de red a tomar decisiones en base a atributos críticos (tipos de tráfico, servicios, usuarios, etc.) identificados previamente.

Palabras clave: Redes Definidas por Software, Análisis de Tráfico, Performance

Contexto

Este proyecto está inserto en una línea de I/D presentada en la Universidad Tecnológica Nacional con código: UTN-2422. Título: "*Modelo para la evaluación de performance mediante identificación de tráfico y atributos críticos en Redes Definidas por Software*". Dicho proyecto se lleva a cabo en el ámbito del Dpto. de Ingeniería en Sistemas de Información perteneciente a la Facultad Regional Resistencia de la Universidad Tecnológica Nacional.

Introducción

Si bien el objetivo del presente trabajo no radica en la descripción de tecnologías aplicadas en SDN, se brinda un breve contexto descriptivo del ámbito de trabajo, a los efectos de una mejor comprensión. Las Redes Definidas por Software proponen un modelo para cubrir nuevas demandas de usuarios y organizaciones. Ésta es una arquitectura de red emergente, donde el control de la infraestructura de red está desacoplado del reenvío de datos y, a su vez, es directamente programable. La inteligencia de red es (lógicamente) centralizada en controladores SDN basados en software que mantienen una visión global de la red. Como resultado, las organizaciones controlan la red independiente del proveedor en un único punto lógico lo que simplifica, en gran medida, el diseño de la red y su operación. A su vez, se simplifica la gestión de los dispositivos de red debido a que ahora no se tienen que entender y procesar múltiples normas de protocolo, sino simplemente aceptar instrucciones de los controladores centralizados de SDN (Fig. 1).

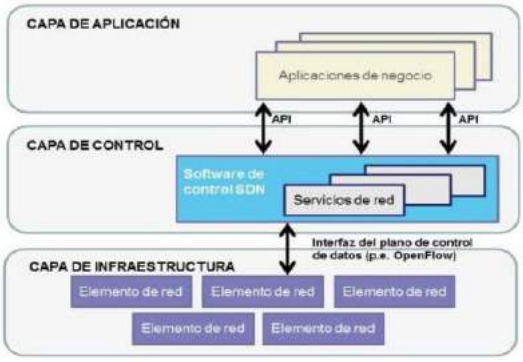

Figura 1. Arquitectura de Software Defined Network

Tal vez lo más importante de esta nueva visión de red, es que se pueda configurar mediante programación esta abstracción simplificada de la red en lugar de tener que configurar manualmente múltiples dispositivos. Además, aprovechando la inteligencia centralizada del controlador SDN, se puede alterar el comportamiento de la red en tiempo real y desarrollar nuevas aplicaciones y servicios de red ágilmente, lo que mejora sustancialmente las posibilidades, a esto se lo conoce por Northbound Interfaces [8]: Una interfaz hacia el norte o capa de aplicación, la cual sigue siendo una cuestión abierta, es muy pronto para estandarizarla, se basará en la experiencia con los diferentes controladores. Además, se debe permitir que las aplicaciones de red no dependan de implementaciones específicas. Para poder concretar la nueva arquitectura de redes fue necesario crear y estandarizar una interfaz de comunicaciones entre el control y el reenvío de datos, conocida por Southbound Interfaces [8]: Son los puentes de conexión entre los elementos de control y los de reenvío, siendo el elemento esencial para separar la funcionalidad del plano de datos y de control. Las API están todavía muy enlazadas a dispositivos físicos de reenvío de paquetes en una infraestructura física. Un switch puede tardar bastante tiempo de desarrollo para estar presente en el mercado, OpenFlow es uno de los primeros estándares de interface de comunicación definida entre las capas de control y forwarding en una arquitectura SDN. Además, permite el acceso directo y la manipulación de la capa de forwarding de los dispositivos de Red (switches, routers), ya sean físicos o virtuales (basadas en un Hypervisor).. Para ello se creó el protocolo OpenFlow que permite el acceso directo a la gestión de datos de reenvío en dispositivos de red como switches y routers, tanto físicos como virtuales y de un modo abierto. Esto contrasta con las arquitecturas de redes tradicionales donde los dispositivos de red son monolíticos y cerrados. Ningún otro protocolo estándar tiene la funcionalidad y finalidad de OpenFlow que transfiere el control de los dispositivos de red a la lógica del software de control.

Una característica particular de SDN basada en OpenFlow es que se puede implementar en las redes existentes, tanto físicas como virtuales. Los dispositivos de red pueden realizar el reenvío basado en OpenFlow, así como el reenvío tradicional, lo que hace que sea muy fácil para las organizaciones introducir progresivamente esta tecnología, incluso en los entornos de red de múltiples proveedores.

Con este panorama es atendible la necesidad de contar con herramientas que permitan evaluar apriori el comportamiento de los cambios introducidos en las aplicaciones y servicios de las redes como también el impacto en la performance de éstas

Líneas de investigación y desarrollo

En el proyecto "*Modelo para la evaluación de performance mediante identificación de tráfico y atributos críticos en Redes Definidas por Software*" se propone el análisis de la arquitectura y estándar de SDN. Además del diseño e implementación de un sistema de soporte a las decisiones con Información Lingüística [9],[10] y [11] para evaluar datos **cualitativos** de SDN y su uso. Este sistema permite que múltiples expertos puedan participar conjuntamente dando sus puntos de vista que ayuden a la toma de decisiones.

Los aspectos **cuantitativos** del modelo se realizan mediante simulaciones y creación de escenarios de comparación con las redes tradicionales. Referencias [1],[2],[3],[4],[5],[6],[7] y [8]

Resultados y objetivos

En este trabajo se han obtenido resultados utilizando los recursos mencionados arriba,

como se verá a continuación. Se utiliza Mininet: ágil y potente simulador de SDN para la creación de escenarios de prueba, teniendo en cuenta particularmente la posibilidad que ofrece en la caracterización de los parámetros de enlaces, tanto en lo que hace a las características del enlace (ancho de banda, retardo etc.) como a parámetros relacionados con calidad de servicio (Ver figura 2). Esto se puede realizar en forma sencilla creando las topologías con la aplicación Miniedit [12], incluida en Mininet, (ver figura 3), que permite tomar y arrastrar los elementos y definir sus propiedades con solo la ayuda del mouse. Lo bueno de Miniedit es que permite construir muy velozmente la topología en un entorno gráfico y genera automáticamente las scripts para utilizarlas con Mininet (Cabe mencionar que tanto en estas scripts como así también en la inmensa mayoría de las aplicaciones en estos entornos se utiliza Python [13]).

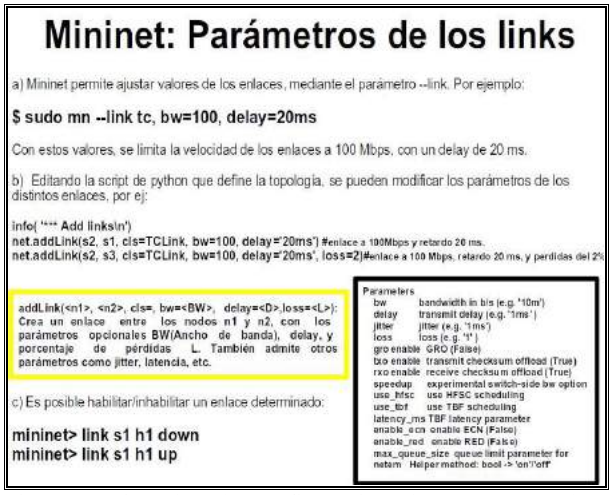

Figura 2 Ejemplos de configuración de parámetros de enlaces (propio).


Figura 3: Interfaz gráfica de Miniedit (propio)

Ejemplo de topología de prueba: a los efectos de realizar una simulación y evaluar la performance se muestra una topología de prueba (figura 4). En la misma se muestran tres locaciones con redes lan en cada una y de las siguientes características:

- Enlaces LAN en las tres locaciones a 100 Mbps con un delay de 5 ms y porcentaje de perdida de paquetes de 1%
- Enlace SW1 SW2 1000 Mbps con delay de 15 ms., porcentaje de perdida de paquetes 1% y Jitter de 1%
- Enlace SW2 SW3 1000 Mbps con delay de 20 ms., perdida de paquetes 1% y Jitter de 1%

Con respecto del controlador, se puede definir para la misma topología, un controlador POX, ODL, o bien cualquiera disponible solo basta con definir las propiedades haciendo un clic con el botón derecho sobre el ícono Controlador.

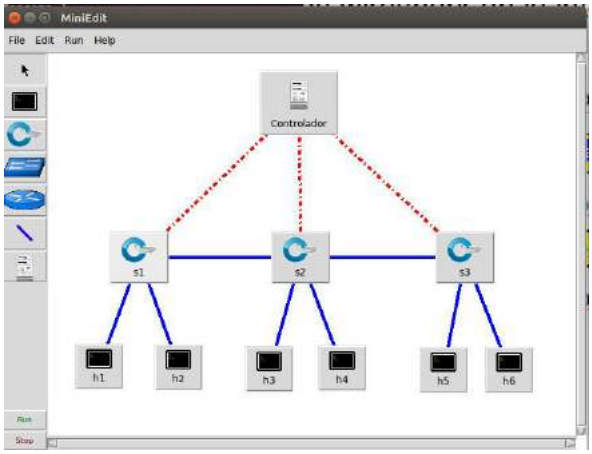

Figura 4: Topología de prueba (propio)

A partir de la topología definida; y ajustada la script con todos los detalles que deseemos, es posible iniciar las pruebas de performance utilizando IPERF3, para ello basta con abrir terminales en los hosts correspondientes y lanzar instancias ya sea de cliente o servidor según el/los enlaces que deseemos probar. Vale aclarar que IPERF3 tiene muchas mejoras y ventajas respecto a la versión original IPERF, las más destacables son la caracterización del tráfico tanto en TCP como

UDP, la posibilidad de lanzar múltiples instancias respecto de un mismo servidor, como así también crear streams en paralelo para cada instancia, se puede ver con detalle en [6]

A continuación, se muestra un ejemplo suponiendo un tráfico TCP de 80 Mbps entre h1 y h3 de la topología de la figura 4.

En h3 un servidor: iperf3 -s -p 5566 -i 1(fig.5)

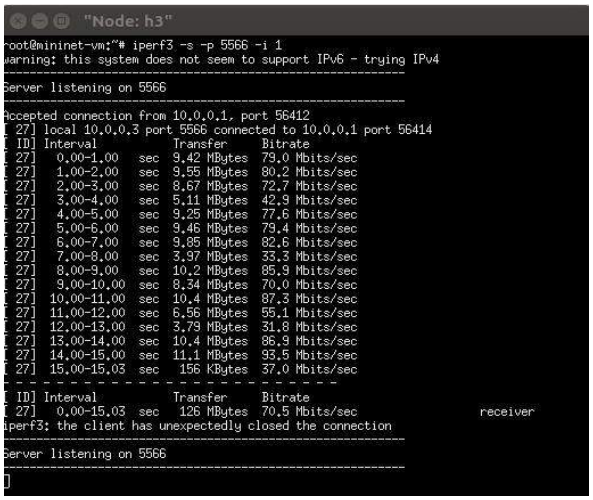

Figura 5: Servidor iperf3 (propio)

En h1 un cliente: iperf3 -c 10.0.0.3 -t 15 -b 80M (fig.6)

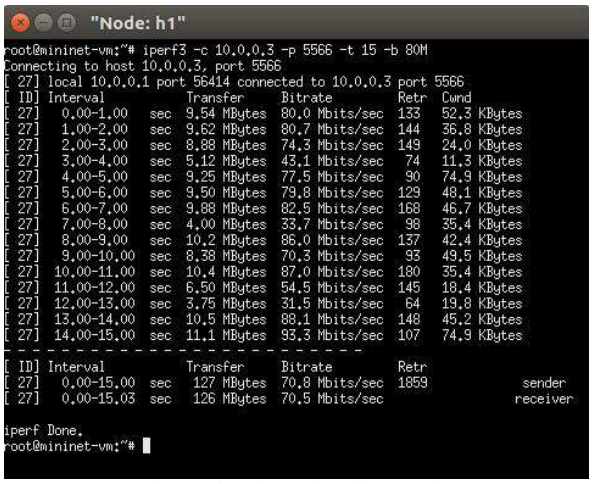

Figura 6: Cliente iperf3 (propio

Para documentar en forma de gráfico los resultados es útil la aplicación GNUPLOT que toma directamente los resultados de la salida de iperf y los convierte en una gráfica a la que se le pueden agregar múltiples detalles, ver referencias [14], [15]. El gráfico correspondiente al ejemplo se muestra en la figura 7.

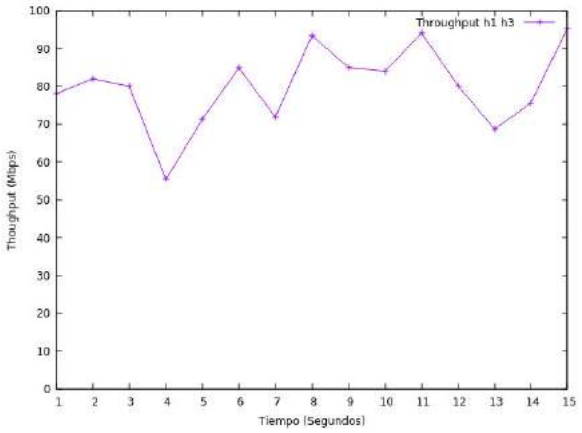

Figura 7: Grafico de trhoughput entre h1 y h3 en Mbp

Con la misma secuencia de operaciones, se pueden graficar múltiples instancias de simulación como así también cambiar muy fácilmente los parámetros de simulación, sencillamente cambiándolos en línea de comandos para luego visualizar los cambios con poco esfuerzo.

Formación de Recursos Humanos

La formación de recursos humanos es la siguiente:

a) **Formación de becarios:** Dos becarios alumnos avanzados de la carrera de ingeniería en sistemas de información y un becario graduado de iniciación a la investigación. Esto hará posible fomentar la actividad de investigación en alumnos que están próximos a recibirse y graduados jóvenes estimulando la actividad de investigación.

b) **Formación de postgrado:** A partir de las líneas de investigación desarrolladas en el proyecto se prevé que el Ing. Carlos Cuevas finalice su Maestría en Redes de la Universidad de La Plata

mediante una tesis vinculada a este proyecto. Así mismo servirá de base para la investigación y formulación de la tesis de Maestría en Administración de Negocios para el Ingeniero Diego Bolatti. Carrera de postgrado cursada en UTN Facultad Regional Resistencia.

Equipo de trabajo:

La estructura del equipo de trabajo es la siguiente:

Director	• Dr. Ing. Sergio Gramajo
Investigadores: (En orden alfabético)	• Ing. Diego Bolatti • Ing. Ricardo Calcagno • Ing. Carlos Cuevas • Ing. Reinaldo José Ramón Scappini
Colaborador Externo	• Dr. Luis Martínez López (Catedrático Universidad de Jaén, España) • Dra. Macarena Espinilla Estévez (Investigadora Universidad de Jaén)

Referencias

1. https://www.opennetworking.org/sdn-definition/
2. https://www.opennetworking.org/software-defined-standards/specifications/
3. https://openflow.stanford.edu/display/ONL/POX+Wiki
4. https://www.opendaylight.org/
5. http://mininet.org/
6. https://iperf.fr/
7. http://gnuplot.info/
8. https://www.opennetworking.org/images/stories/downloads/sdn-resources/technical-reports/SDN-architecture-overview-1.0.pdf
9. Espinilla, M., Liu, J., Martínez, L.: An extended hierarchical linguistic model for decision-making problems. Computational Intelligence 27, 489-512 (2011)
10. Herrera, F., Herrera-Viedma, E.: Linguistic decision analysis: Steps for solving decision problems under linguistic information. Fuzzy Sets and Systems 115, 67-82 (2000)
11. Zadeh, L.: The concept of a linguistic variable and its application to approximate reasoning. Part I, Part II and Part III. Information Sciences 199-249, 301-357, 143-180 (1975)
12. http://www.brianlinkletter.com/how-to-use-miniedit-mininets-graphical-user-interface/
13. https://www.sdnskills.com/learn/learn-python-1/
14. http://www3.fi.mdp.edu.ar/analisis/links/gnuplot-tut.pdf
15. http://csie.nqu.edu.tw/smallko/sdn/iperf_mininet.htm