ISSN: 1692-7257 - Volumen 1 – Número 37 - 2021

Revista Colombiana de Tecnologías de Avanzada

Recibido: 10 agosto 2020
Aceptado:11 septiembre 2020

ALGORITMOS DE CALIDAD DE SERVICIO PARA BALANCEAR CARGAS EN REDES DEFINIDAS POR SOFTWARE

QoS algorithms for load balancing in Software Defined Networks

Juan José Guzmán Pineda[*], Javier David Sánchez Calderón[*]
Jorge E. Gómez Gómez, Ph.D.[***]

[*] **Universidad de Córdoba,** Estudiante de Ingeniería de Sistemas/Ingeniería, Montería, Colombia, jguzmanpineda56@correo.unicordoba.edu.co, 0000-0002-1567-314X

[**] **Universidad de Córdoba,** Estudiante de Ingeniería de Sistemas/Ingeniería, Montería, Colombia, jsanchezcalderon49@correo.unicordoba.edu.co, 0000-0002-1567-314X.

[**] **Universidad de Córdoba Monteria,** Departamento de Ingeniería de Sistemas –, Colombia, jeliecergomez@correo.unicordoba.edu.co, 0000-0001-8746-9386

Resumen: Internet ha creado una sociedad digital donde la mayoría están conectados y acceden desde cualquier lugar, sin embargo esto hace que las redes colapsen creando la necesidad de utilizar nuevos paradigmas en las comunicaciones, las redes definidas por software junto con el balanceo de carga presentan una solución a este problema aplicando algoritmos de calidad de servicio utilizan mecanismos para mejorar el tiempo de respuesta de los servidores a los que se conectan los clientes.
Se propuso realizar un balanceador de cargas aplicando algoritmos de calidad de servicios en las redes definidas por software, analizando paso a paso la estructura que arma un balanceador de cargas aplicado en las redes definidas por software, interpretando las políticas de calidad de servicio en cuanto al tráfico de datos por una red se refiere para lograr establecer las herramientas que se usaron para implementar la simulación de la red y finalmente se señala la productividad de las redes definidas por software aplicando el balanceo de cargas por medio de algoritmos de calidad de servicios. Se simuló la red en un entorno virtual utilizando la herramienta Mininet para configurar los parámetros de red y acoplarla con el controlador de red OpenDayLight logrando encontrar la mejor ruta de todos los caminos posibles para que el tráfico circule entre los hosts.
Se logró observar una gran diferencia entre el antes y después de las mediciones en la red, estas mejoras fueron en base al tiempo de retraso y al ancho de banda siguiendo los lineamientos de calidad de servicio, se tabularon los datos y crearon gráficas para observar la tendencias en los estados de la red, llegando a la conclusión que los algoritmos de calidad de servicio aplicado a las redes definidas por software para balancear las cargas mejora considerablemente el funcionamiento de la red..

Palabras clave: Algoritmo, software, Mininet, OpenDayLight.

Abstract: Internet has created a digital society where most are connected and has Access from anywhere, however it causes networks collapse, creating the need of use new paradigms in comunications, software-defined networks along with load balancing present a solution to this problem applying quality of service algorithms they use mechanisms to improve the response time of servers which clients connect.
It was proposed to make a load balancer applying quality of service algorithms in the networks defined by software, analyzing step by step the structure that assembles a load

balancer applied in the networks defined by software, interpreting the quality of service politics regarding data traffic over a network refers to achieve set up the tools that were used to implement the simulation of the network and finally points out the productivity of the networks defined by software applying load balancing by means of service quality algorithms. The network was simulated in a virtual enviroment using the mininet tool to configure the network parameters and attach it with the openDayLight network controller managing to find the best route in all possible ways for traffic circulate among hosts.
Was possible to observe a big difference between before and after of measurements in the network, these improvements were based on the delay time and the bandwidth following the guidelines of quality serve, data was tabulated and was created graphs to observe trends in the network states, coming to the conclusion that the quality of service algorithms applied to software-defined networks to balance loads improve significantly the functioning of the network.

Keywords: Algorithm, software, Mininet, OpenDayLight.

1. INTRODUCCIÓN

Desde el inicio de la computación, siempre se ha venido optimizando los procesos de las máquinas, por lo cual continuamente se desarrollan diferentes tecnologías para los diversos campos; el ámbito de la redes no es ajeno a esta evolución, porque desde tiempos de antaño se ha innovado la forma de comunicación, también los distintos dispositivos que se usan para la misma, igualmente nuevos protocolos y nuevas versiones de estos. Las redes convencionales brindan estabilidad pero muy pocas veces ofrecen flexibilidad y la capacidad de control total, algunos dispositivos avanzados que se encuentran en el mercado podrían asemejar estas funciones, pero son demasiado costosos. A partir de esta idea nacen las Redes Definidas por Software, un paradigma de redes que desacopla el plano de datos del plano de control, tras de eso deja administrar toda la red por medio de un controlador que gestiona con totalidad la red, cabe decir que por la utilización de este controlador se eliminan dispositivos físicos de la red.
Se tiene entendido que en las redes convencionales existen unos mecanismo para agilizar el tiempo de respuesta de los servidores y para evitar la saturación de peticiones en un servidor, a esto se le conoce como balanceador de carga, el cual también se implementa en las Redes Definidas por Software, la idea en este caso es encontrar el camino más óptimo en la red para realizar el recorrido de la petición realizada. Hay unos estándares que mejoran el servicio prestado por la red, esto se llama, Calidad de Servicio; esto brinda algoritmos y políticas que garantizan una red con QoS. Cuando una red cumple con estas políticas se puede decir que es una red optimizada para brindar servicio, con este, se quiere introducir unos algoritmos de Calidad de Servicio en el balanceo de cargas en las Redes Definidas por Software con el fin de perfeccionar el funcionamiento de este tipo de red, cabe destacar que esto se hace en un entorno simulado, ya que este paradigma de redes aún está en desarrollo y están implementadas en pocas empresas del mercado.

2. ESTADO DEL ARTE

Las redes IP tradicionales son complejas y muy difíciles de administrar. Es difícil configurar la red de acuerdo con políticas predefinidas y reconfigurarla para responder a fallas, cargas y cambios. Para complicar aún más las cosas, las redes actuales también están integradas verticalmente: los planos de control y de datos están agrupados. Las redes definidas por software (SDN) son un paradigma emergente que promete cambiar este estado de cosas, rompiendo la integración vertical, separando la lógica de control de la red de los enrutadores y conmutadores subyacentes, promoviendo la centralización (lógica) del control de la red e introduciendo la capacidad para programar la red. La separación de preocupaciones, introducida entre la definición de políticas de red, su implementación en el hardware de conmutación y el reenvío de tráfico, es clave para la flexibilidad deseada: al dividir el problema de control de red en partes manejables, SDN facilita la creación e introducción nuevas abstracciones en redes, simplificando la gestión de la red y facilitando la evolución de la red.(Kreutz et al., 2015).

En SDN, la inteligencia de la red está centralizada lógicamente en los controladores basados en software (el plano de control), y los dispositivos de

red se convierten en simples dispositivos de reenvío de paquetes (el plano de datos) que se puede programar a través de una interfaz abierta como OpenFlow (Valencia et al., 2015).

MPLS genera mayor velocidad, ancho de banda, garantía y potencial de reparación satisfaciendo las crecientes necesidades de datos y comunicaciones bidireccionales, utilizando el mecanismo de reenvío de paquetes eficiente potencializando la capa de enlace de datos y capa de red, este transporta datos de los nodos presentes en una red a otra red utilizando etiquetas en lugar de direcciones basadas en IP. Las redes IP no pueden determinar qué ruta tomará un paquete de datos específico cuando se transporta a través de la red y tampoco pueden determinar la cantidad de tiempo que este tardará en llegar a su destino, MPLS utiliza etiquetas para identificar la ruta que los paquetes deben atravesar por la red. Sin embargo, IP a través de una red MPLS puede resolver muchos problemas, si bien es conocido MPLS no es una alternativa para la comunicación de datos basada en IP, pero contribuye a la expansión de la arquitectura IP incluyendo funcionalidades y aplicaciones dinámicas. (Premkumar & Saminadan, 2018).

Cuando se habla de etiquetas MPLS se refiere a una apreciación simplificada de la cabecera de un paquete IP, aun cuando una etiqueta contiene toda la información asociada al direccionamiento de un paquete hasta su destino final en la red MPLS. A diferencia de un encabezado IP, las etiquetas no contienen una dirección IP, sino más bien un valor numérico acordado entre dos nodos consecutivos para proporcionar una conexión a través de un proveedor de servicios por niveles. (Yury & Jhon, 2016).

Cuando se habla de QoS (Calidad de Servicio) se refiere al conjunto de requisitos de servicio que debe cumplir la red durante el transporte de los datos, las redes trasladan gran cantidad de servicios como datos, voz y video, para asegurar un flujo ágil los proveedores de servicio implementan mecanismos para tratar de una forma diferente el tráfico según lo que requieran los clientes y así satisfacer las necesidades de los usuarios. El objetivo principal de cualquier mecanismo QoS es asegurar que la congestión en la red no interfiera con los paquetes, priorizando el tráfico y la asignación de capacidad para los paquetes cuando la red está congestionada. Uno de los protocolos que usan calidad de servicio es OpenFlow a través de un mecanismo de colas, el ciclo de vida de QoS se simboliza con cinco operaciones que son: crear colas, agregar flujos de QoS, modifica los flujos de QoS, supresión de flujos QoS y Supresión de colas.(Fernández Mora & Ulloa Banegas, 2016).

Estableciendo QoS en la red hay un conjunto de requisitos que se deben cumplir enfocados en el transporte de los flujos que pueden ser implementadas tanto para gestionar la saturación de la red o para evitarla, en general calidad de servicio permite controlar características significativas en la transmisión de paquetes dentro de una red, características que se especifican en términos cuantitativos o estadísticos como ancho de banda, latencia, jitter, pérdida de paquetes de la red. Fortaleciendo la red en pro de cumplir con los requisitos de tráfico en función del perfil y ancho de banda para un determinado flujo de datos.(Jonier & Daniel, 2017).

Los servidores de internet soportan aplicaciones críticas que deben estar corriendo todo el tiempo lo cual conlleva a que las redes necesitan escalar el rendimiento para manejar los grandes volúmenes de peticiones de los clientes y evitar demoras, esto se logra creando cluster de servidores, sin embargo la disponibilidad de estos es amenazada por sobrecargas en la red, el balanceo de carga del lado del servidor es una solución que se adopta a los problemas del funcionamiento y disponibilidad. El balanceo de cargas provee escalabilidad y fácil manejo a servicios de internet, este distribuye el tráfico IP a múltiples copias o instancias de servicios TCP/IP, cada uno corriendo en un host dentro del cluster de servidores.(Castro & Ferrer, 2004).

El balanceador de cargas es un puente entre la red y el servidor, esté generalmente necesita conocer el estado del servidor pero no necesita configurar el protocolo de red en general los servidores de balanceo de carga deben realizar de dos a siete capas de procesamiento de datos lo cual es fácil convertirse en el cuello de botella de todo el sistema de servicio, con la aparición de redes definidas por software la apertura de la red es más grande, propone separar y transmitir los datos para así obtener una visión global de la red. El balanceo de carga basado en SDN no modifica directamente la IP de origen del paquete TCP, el puerto de origen, el puerto IP de destino ni la dirección MAC de destino sino a través de la tabla de flujo distribuida mediante conmutadores SDN, así el balanceo de cargas en SDN solo se una para generar modificar o eliminar reglas en l atabla de flujo y no para reenviar ningún paquete a un cliente

específico. La tabla de flujo de balanceador de cargas SDN se basa en la inspección de estado y el algoritmo de balanceo de carga que se esté utilizando. (Qilin & Weikang, 2015).

La columna vertebral de internet es el protocolo IP el cual describe la estructura en que las redes pueden comunicarse para conformar internet proclamado como la red de redes. El intercambio de paquetes juegan un papel importante en la comunicación a través de la red, los administradores de la red pueden rastrear los paquetes durante el tráfico de la red y así poder solucionar problemas que se presenten y depurar con la implementación de protocolos, existen muchas herramientas de monitoreo de tráfico pero Wireshark es la más popular y una de las mejores herramientas utilizadas para dicho fin, se utiliza para analizar los protocolos, analizador de red, paquetes y como herramienta forense (Elamaran et al., 2018).

3. METODOLOGÍA

Levantamiento de entorno simulado
Antes que nada, se debe tener un entorno de simulación ya creado, es decir, la red va a ser simulada por softwares que imitan el funcionamiento exacto de una Red Definida por Software; ya se ha dicho que este tipo de redes aún no son muy comunes. Cabe destacar que todo será implementado sobre el sistema operativo Ubuntu Desktop en su versión 18.04

Redes con Mininet
Con este software se despliega la Red Definida por Software que se usará en este caso, con herramientas que trae integradas se puede conocer datos de la red, usando comando para saber los datos de la red, tales como: la velocidad de transmisión y el ancho de banda entre dos host de la red, también se pueden conocer lo enlaces de la red, los puertos, las direcciones IP y Mac, entre muchas funciones. En este caso se trabaja con una red personalidad creada con un script en Python. Es necesario decir que se pueden desplegar redes predefinidas, estas redes son las más comunes en esta área, hay comando especiales para cada red junto a eso también cada tipo de red exigen diferentes parámetros numéricos que pueden indicar características únicas de cada red.

Conexión de Mininet con OpenDayLight
En esta ocasión se usa el controlador de SDN llamado OpenDayLight el cual ofrece la administración de la red, este controlador brinda un entorno agradable que se compone de herramientas para la supervisión de la red de una forma centralizada. Es uno de los más estables y uno a los que más aportan las comunidades dedicadas al desarrollo de esta tecnología.

Cuando se tenga la red definida, al momento de ejecutarla se conecta con el controlador SDN, mediante la dirección IP y el puerto, también hay que especificar el protocolo a usar, por ejemplo:

sudo mn –custom ./ArchivoRed.py --controller=remote,ip=192.168.1.10,port=3366 --switch=ovs,protocols=OpenFlow13

Luego de ejecutar el comando se visualiza la topología desplegada en el controlador, ingresando al index del servidor, para esto se ingresa al localhost que es donde está corriendo el controlador. La red desplegada es la siguiente:

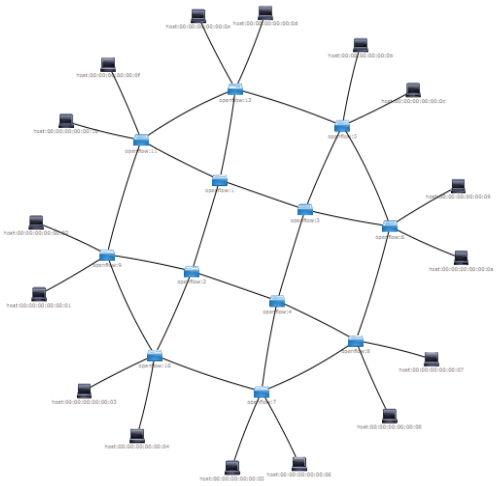


Figura 1. Red desplegada en OpenDayLight
Figure 1. Network deployed in OpenDayLight
Fuente: Elaboración propia

Implementación del algoritmo de Dijkstra en el balance de carga

Algoritmo de Dijkstra
Es un algoritmo para la determinación del camino más corto, dado un vértice origen, hacia el resto de los vértices en un grafo que tiene pesos en cada arista (Algoritmo de Dijkstra - Wikipedia, la enciclopedia libre, n.d.).

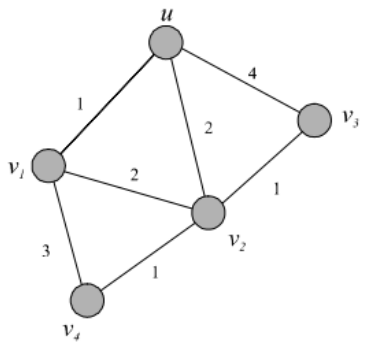


Fig. 2. Grafo G
Figure 2. Graph G
Fuente: (Segura, 2006)

En la Figura 2. Cada nodo v del grafo G (V, E) tiene una etiqueta asociada L(v). Esta etiqueta indica la menor distancia conocida para ir desde un nodo fijado u hasta este nodo. Inicialmente, el valor de L (v) corresponde al peso w (u, v) de la arista que une los nodos u y v, en el caso de que esta arista exista, siendo L (v) = ∞ en caso contrario. El algoritmo funciona creando un conjunto de nodos T ⊂ V, para los cuales se han obtenido hasta ese momento el camino más corto desde u hasta ellos. Al final del algoritmo, L(v) contiene el coste del camino más corto para ir desde u hasta v (Segura, 2006).

Pseudocódigo del algoritmo:

1. para todo v ≠ u L(v) = w(uva)
2. L(u) = 0
3. T = {u}
4. mientras T ≠ V

Inicio

5. encuentra v'∉ T de forma que ∀ v∉ T L(v') ≤ L(v)
6. T = T ∪ {v'}
7. para todo v∉ T de forma que v' es adyacente a v

si L(v) > L(v') + w(v',v)
entonces L(v) = L(v') + w(v',v)
fin si
fin para
fin mientras

En cada iteración, el algoritmo añade un nuevo nodo en la lista T. Esto se consigue escogiendo un nodo v' que todavía no pertenezca a T y que tenga una etiqueta L (v´) mínima. En otras palabras, escogemos el nodo v' más cercano a u y externo a la lista T. Una vez hecho esto, se actualizan las etiquetas de los nodos sobre los que incide v', de manera que hacemos un nuevo cálculo de las distancias de u a estos nodos y añadimos este nodo v' a T. El proceso se repite hasta que todos los nodos del grafo se encuentran en la lista (Segura, 2006).

Aplicación Algoritmo de Dijkstra al balance de cargas

En esta parte se aplica todo el concepto del algoritmo de rutas cortas, codificado en el lenguaje de programación Python, el script se ejecuta después que la red esté desplegada y previamente conectada con el controlador, ya que los flujos de cada ruta será instados por medio de este; de todos los host que están en la red (Figura 1), se escogen dos, cuales quiera para realizar el balance de carga, este algoritmo prueba todas las posibles rutas que unen a los dos hosts, de todas esas rutas escoge el camino más corto o en su defecto el más óptimo, ya que el camino más corto no siempre es el más óptimo, al tener la ruta óptima, el script instala los flujos correspondientes en los switchs en ambas direcciones que conectan a los dos hosts y finalmente se obtiene el balance de cargas, en donde se puede consultar la velocidad y el ancho de banda entre estos dos hosts.

Implementación del algoritmo de Dijkstra con MPLS en el balance de carga

Del mismo modo se usa el mismo algoritmo para buscar la ruta más corta, pero los flujos se van a instalar en los switches OVS mediante el mecanismo MPLS que en vez de direcciones IP, utiliza etiquetas para reconocer los dispositivos, esto para variar las formas de instalaciones de flujos en una SDN y comprobar como es más óptimo el balance de cargas. Es preciso decir que el mecanismo MPLS garantiza la calidad de servicio cuando se le da un uso correcto en la red, este se acomoda a estas políticas.

Para la creación de estos flujos, se utilizan acciones elementales como PUSH, POP y SWAP que son propias del mecanismo MPLS para el proceso de las etiquetas, que a su vez este viene incorporado en las herramientas del comando OVSDB, que se

utiliza para el monitoreo y gestión de los conmutadores OpenFlow (Villas, 2019).

Como se usa el mismo algoritmo para encontrar la ruta óptima, se modifica el script en la parte de instalación de flujos, cabe destacar que esto flujos instalados de esta forma también se ven reflejado en el controlador SDN, que es en este caso es OpenDayLight.

Por consiguiente, se debe hacer una modificación en el sistema para que el script se ejecute con éxito, ya que para usar los comandos de la creación de los flujos se debe pedir permiso de superusuario. Se agrega la ruta del directorio en el en el archivo que otorga los permiso root del sistema, esto con el fin de que los comandos dentro del script no tengan problemas de ejecución debido a que estos piden contraseña de administrador. Finalmente se ejecuta el algoritmo satisfactoriamente pidiendo los datos requeridos para el balance de carga, que son los dos hosts de la red.

4. RESULTADOS

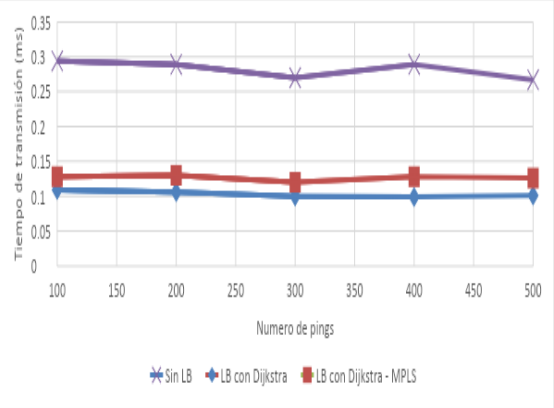
Con el fin de mostrar resultados más claros con respecto a la aplicación de los algoritmos de balanceo de cargas se realizaron pruebas exhaustivas en la red, todo con el comando ping que muestra el tiempo de la latencia de los host participativos en las pruebas, a continuación se muestra cómo se comportó la red con diferentes cantidades de solicitudes de ping entre los dos hosts involucrados, los cuales son h1 y h11, para una mejor aproximación se tomó el promedio de 100 pings hasta llegar a 500 pings el aumento será de 100 por cada prueba, se muestran los datos en ambiente sin balance de carga, balance de carga con Dijkstra y balance de carga con Dijkstra y MPLS.

Tabla 1. Resultados de pruebas de los algoritmos

	Promedios de los tiempos de latencia (ms)		
Pings	Sin LB	LB con Dijkstra	LB con Dijkstra - MPLS
100	0.294	0.109	0.128
200	0.289	0.106	0.130
300	0.270	0.100	0.120
400	0.289	0.99	0.128
500	0.267	0.101	0.126

Fuente: Elaboración propia

Se obtiene una mejora del tiempo de latencia al aplicar los algoritmos implementados para el balance de cargas; en el caso del balanceo de cargas con Dijkstra, la red mejoró aproximadamente 2,7 veces sobre la red sin ningún tipo de balance y aplicación del balanceo de carga con Dijkstra y MPLS la mejora aproximada del tiempo de latencia es de 2,2 veces sobre la red sin balance de carga. A continuación en la Figura X se muestra la tendencia de los promedios de una gráfica, se puede tener un mejor visión de los datos.



.

***Figura 3**. Gráfica de tiempos de latencia*
Figure 3. Latency time graph
Fuente: Elaboración propia

En la gráfica se observa gráficamente los resultados de las pruebas hechas en la red, el algoritmo que tiene un menor tiempo de latencia es el más óptimo para la realización de balance de carga, en este caso el algoritmo de Dijkstra actuando solo, es la mejor apuesta para balancear cargas; aunque usan el mecanismo MPLS también se obtuvo unos muy buenos resultados, consiguiendo así la optimización de la red y brindando una buena calidad de servicio.

En definitiva, se puede decir que estos algoritmos son muy viables para usarlos a la hora de balancear cargas en este tipo de redes, ya que con esto se puede agilizar los tiempos de respuesta de una red y aprovechando al máximo los beneficios que brindan las Redes Definidas por Software.

5. CONCLUSION

En el anterior artículo se emplean algoritmos de calidad de servicio implementados en redes definidas por software para realizar balance de

cargas, estructurando una muestra de balance de cargas basado en redes comunes que se usan en el entorno, obteniendo una distinguida interpretación de las políticas y estándares de calidad de servicio que actualmente se usan en transferencia de datos. Además de ejecutar con éxito algoritmos de balanceo de cargas junto algunas herramientas de implementación de red en un entorno donde se observa el rendimiento de la red, creando un escenario virtual utilizando herramientas de aplicación de redes definidas por software cuando se despliega la red aplicando los algoritmos de calidad de servicio logrando optimizar los tiempos de transferencia entre los hosts relacionados mostrando buen uso de los módulos del balanceo de carga.

Afirmando esto con los datos mostrados en los resultados donde estos se comparan en la red con mesura, para llegar a la conclusión que una red definida por software ajustada a las normas de calidad de servicios para el balance de cargas permite mejorar los sockets entre cliente servidor trayendo como resultado la minimización del tiempo de respuesta ampliando así el ancho de banda y lograr obtener una mejor firmeza en la conexión a una mayor tasa de transferencia..

REFERENCIAS

Algoritmo de Dijkstra - Wikipedia, la enciclopedia libre. (n.d.). Retrieved May 17, 2021, from https://es.wikipedia.org/wiki/Algoritmo_de_Dijkstra

Castro, K., & Ferrer, R. (2004). LOAD BALANCING: BALANCEO DE CARGA. CONCEPTO, ESTADO DEL ARTE Y APLICABILIDAD EN LINUX Y WINDOWS. 55. http://eprints.uanl.mx/5481/1/1020149995.PDF

Elamaran, V., Arunkumar, N., Venkat Babu, G., Balaji, V. S., Gomez, J., Figueroa, C., & Ramirez-Gonzalez, G. (2018). Exploring DNS, HTTP, and ICMP Response Time Computations on Brain Signal/Image Databases using a Packet Sniffer Tool. IEEE Access, 6(c), 59672–59678. https://doi.org/10.1109/ACCESS.2018.2870557

Fernández Mora, M. M., & Ulloa Banegas, R. F. (2016). Despliegue de una red SDN aplicando el protocolo MPLS y generando políticas de QoS para servicios de telefonía IP. 72.

Jonier, P., & Daniel, D. (2017). ANÁLISIS DE LAS CAPACIDADES Y PRESTACIONES DE CALIDAD DE SERVICIO EN REDES DEFINIDAS POR SOFTWARE. 13–14.

Kreutz, D., Ramos, F. M. V., Verissimo, P. E., Rothenberg, C. E., Azodolmolky, S., & Uhlig, S. (2015). Software-defined networking: A comprehensive survey. Proceedings of the IEEE, 103(1), 14–76. https://doi.org/10.1109/JPROC.2014.2371999

Premkumar, S., & Saminadan, V. (2018). Performance evaluation of smart grid communication network using MPLS. Proceedings of the 2017 IEEE International Conference on Communication and Signal Processing, ICCSP 2017, 2018-Janua, 2116–2120. https://doi.org/10.1109/ICCSP.2017.8286779

Qilin, M., & Weikang, S. (2015). A Load Balancing Method Based on SDN. Proceedings - 2015 7th International Conference on Measuring Technology and Mechatronics Automation, ICMTMA 2015, 18–21. https://doi.org/10.1109/ICMTMA.2015.13

Segura, S. D. (2006). Análisis de algoritmos multicast en redes overlay.

Valencia, B., Santacruz, S., & Padilla, L. Y. B. J. J. (2015). Mininet : una herramienta versátil para emulación y prototipado de Redes Definidas por Software 1 Mininet : a versatile tool for emulation and prototyping of Software Defined Networking. 17, 62–70.

Villas, A. D. Las. (2019). Combinación de mecanismos MPLS en una arquitectura SDN. Telemática, 18(1), 1–10.

Yury, M., & Jhon, A. (2016). EMULACIÓN DEL PROCESO DE CONMUTACIÓN/APILAMIENTO DE ETIQUETAS EN REDES MPLS, MEDIANTE UNA HERRAMIENTA DE SIMULACIÓN PARA REDES DEFINIDAS POR SOFTWARE. 2016.