UNIVERSIDAD POLITÉCNICA SALESIANA ECUADOR

POSGRADOS

MAESTRÍA EN
TELEMÁTICA

RPC-SO-01-NO.025-2021

OPCIÓN DE TITULACIÓN:

PROYECTO DE TITULACIÓN CON COMPONENTES DE INVESTIGACIÓN APLICADA Y/O DE DESARROLLO

TEMA:

EVALUACIÓN DEL TIEMPO DE FINALIZACIÓN DE FLUJO DEL TRÁFICO INTERNO DE UNA RED DE CENTRO DE DATOS EN CRECIMIENTO BASADO EN SDN

AUTOR(ES)

CARLOS ALFONSO LUCAS JURADO

DIRECTOR:

JAVIER GONZALO ORTIZ ROJAS

GUAYAQUIL – ECUADOR
2023

Autor(es):

Carlos Alfonso Lucas Jurado
Ingeniero en Sistemas con Mención en Telemática
Candidato a Magíster en Telemática por la Universidad Politécnica Salesiana – Sede Guayaquil.
carloslucasj@gmail.com

Dirigido por:

Javier Ortiz
Ingeniero en Computación (Especialización Sistemas Tecnológicos)
Magister en Sistemas de Información Gerencial
javierortiz24@gmail.com

Carlos Alfonso Lucas Jurado
EVALUACIÓN DEL TIEMPO DE FINALIZACIÓN DE FLUJO DEL TRÁFICO INTERNO DE UNA RED DE CENTRO DE DATOS EN CRECIMIENTO BASADO EN SDN

DEDICATORIA

A mi mamá, Marivel Jurado Ronquillo, por su constante y desinteresado apoyo en mi formación profesional.

AGRADECIMIENTO

Quiero expresar mi más sincero agradecimiento a mi tutor Ing. Javier Ortiz, por su constante orientación, colaboración intelectual, análisis crítico y retroalimentación, que fueron fundamentales para la culminación exitosa de este trabajo. Su experiencia, paciencia y dedicación desempeñaron un papel crucial tanto en la dirección general del proyecto como en la superación de desafíos técnicos.

También quiero expresar mi gratitud al Ing. Miguel Quiroz por sus valiosas observaciones en el código de manipulación de datos y del presente proyecto. Su pericia y disposición para compartir sus conocimientos fueron de gran utilidad.

Además, deseo rendir homenaje a mi madre, Marivel Jurado Ronquillo, PhD., cuyo apoyo incansable, amor incondicional y sacrificios invaluables fueron fundamentales para lograr la consecución de este objetivo.

Tabla de Contenido

INDICE DE FIGURAS

INDICE DE TABLAS

Evaluación del tiempo de finalización de flujo del tráfico interno de una red de centro de datos en crecimiento basado en SDN

Autor(es):

Carlos Alfonso Lucas Jurado

Resumen

El enfoque principal de este trabajo fue la evaluación de los tiempos de finalización de flujo (Flow Completion Time - FCT) en un entorno de simulación de arquitecturas de centro de datos (DCN). Se optó por simular la arquitectura DCN Fat-tree con configuraciones escalables de k=4 y k=6, usando la herramienta de simulación mininet, gestionadas bajo el controlador SDN ONOS y simulando tráfico realista característico para estos entornos. A través de herramientas como Wireshark, se recopilo los datos de los archivos PCAP de las simulaciones, los cuales fueron convertidos a archivos CSV. Posteriormente, con el uso de herramientas como Python y Pandas, se calcularon los FCT de los flujos de las 2 topologías. Finalmente se usó la Función de Distribución Acumulativa (Cumulative Distribution Function - CDF) para el análisis de los datos, comparando los FCT en relación con su tiempo de transmisión teórico "ideal", donde no se consideran retardos ni congestión de la red. Los resultados indican que, con una configuración de k=4, los FCT se alinean en un 80% más estrechamente con los tiempos teóricos ideales. En contraste, los FCT con k=6 presentan una variabilidad y dispersión notoriamente superior, desviándose considerablemente de los tiempos ideales debido a una variación estándar aumentada. La investigación concluye que los FCT se ven afectados cuando el valor "k" de la escalabilidad de Fat-tree es aumentado. Si bien un "k" más alto teóricamente ofrece más rutas y capacidad, la falta de una gestión apropiada, como la que puede proporcionar un controlador como ONOS, puede resultar en que los recursos queden sin explotarse. En base a este estudio se sugiere considerar estrategias de gestión de red para minimizar la variabilidad de los FCT en un DC en constante crecimiento. Esto puede incluir políticas de calidad de servicio y optimización del uso de los enlaces gestionados por el controlador SDN.

Palabras clave:

Data Center Network, SDN, Fat-tree.

ABSTRACT

The primary focus of this study is the evaluation of Flow Completion Times (FCT) within a simulated environment of data center architectures (DCN). The decision was made to simulate the Fat-tree DCN architecture with scalable configurations of k=4 and k=6, utilizing the Mininet simulation tool, all managed under the ONOS SDN controller and emulating realistic traffic typical of these environments. Data from the PCAP files of the simulations was collected through tools like Wireshark, which were then converted into CSV files. Subsequently, tools such as Python and Panda were employed to calculate the FCTs of the flows in both topologies. Finally, the Cumulative Distribution Function (CDF) was used for data analysis, comparing the FCTs in relation to their "ideal" theoretical transmission time, where network delays and congestion are not considered. The results indicate that with a k=4 configuration, the FCTs align 80% more closely with the ideal theoretical times. In contrast, the FCTs with k=6 exhibit significantly higher variability and dispersion, deviating considerably from the ideal times due to an increased standard variation. The research concludes that FCTs are impacted when the “k” value of the Fat-tree scalability is increased. While a higher "k" theoretically provides more paths and capacity, the lack of proper management, which a controller like ONOS could provide, may result in underutilized resources. Based on this study, it is suggested to consider network management strategies to minimize the variability of FCTs in a constantly growing DC, which may include quality of service policies and optimization of the use of links managed by the SDN controller.

keywords:
Data Center Network, SDN, Fat-tree.

1. INTRODUCCIÓN

En los últimos años existe una marcada tendencia en el desarrollo y uso de los servicios de la nube (Bello et al., 2021; Makhlouf, 2020; Srivastava & Khan, 2018), tales como IaaS(Infraestructure as a Service), PaaS(Platform as a Service) y SaaS(Software as a Service) (Jackson & Goessling, 2018), lo que conlleva a que la escala de la red interna de los centros de datos (Data Center Networks – DCN) (Sharma & Mishra, 2020) sean de mayor tamaño, con un crecimiento significativo del volumen de tráfico interno (llamado también tráfico de tipo este-oeste) (Li et al., 2020). Por lo tanto, la red interna de los DC se han vuelto uno de los ejes centrales de esta tendencia al ser los encargados de procesar el tráfico de estos servicios (C. Hu et al., 2018; Zhang et al., 2022). En función de esta premisa, en trabajos previos (C. Hu et al., 2018; Yahyaoui et al., 2020; Zhang et al., 2022), se considera que el propósito primordial en el diseño de la red interna de un DC es minimizar el tiempo de finalización de flujo (FCT - Flow Completion Time). El FCT es una métrica de rendimiento que se define como el tiempo necesario para transferir todos los paquetes correspondientes a un flujo (Yahyaoui et al., 2020). En este ámbito cada vez son más las organizaciones (Ma et al., 2017; Teo et al., 2016) que utilizan la arquitectura de las redes definidas por software (Software-Defined Networking - SDN) (Nunes et al., 2014) para la gestión de la red interna de sus DC. Las SDN desacoplan el plano de control del plano de datos en los dispositivos de red, facilitando una gestión centralizada y adaptable de la red, lo cual facilita la implementación de nuevos protocolos o mecanismos de control que puedan mejorar el rendimiento de la red interna del DC, incluyendo también el FCT.

2. Determinación del Problema

La tendencia creciente y acelerada en el uso de los servicios en la nube tiene un impacto importante en las redes internas de los centros de datos (DC), en especial del tráfico de tipo este a oeste (tráfico de la red interna), afectando al tiempo de terminación de flujo (FCT), la cual es una métrica para medir el rendimiento de la red interna de un centro de datos. Las redes internas de los DC deben estar preparadas para responder a la demanda de la evolución exponencial de la nube, por lo que es importante medir como se ve afectado el FCT en una red interna de un DC en constante crecimiento. Bajo este contexto, las redes definidas por software (SDN) son una tecnología en ascenso que desacopla el plano de control del plano de datos de los dispositivos de red, permitiendo la innovación de soluciones de control de flujo lógicamente centralizadas a través de la programabilidad de la red, lo cual puede mejorar el rendimiento de la red interna de un DC en especial del FCT. En trabajos relacionados (Modi & Swain, 2019) abordan el rendimiento una red de DC basado en SDN, considerando la utilización promedio del ancho de banda y tasa de transferencia efectiva (throughput), mientras (Li et al., 2020) abordan el ancho de banda de bisección y la tasa de utilización del enlace, pero en estos trabajos no se aborda la métrica FCT ni se considera su afectación en varias escalas de una red interna de DC. Los siguientes trabajos (Abdelmoniem et al., 2017; Zaher et al., 2020; Zang et al., 2017) si evalúan el FCT en un escenario de DCN basado en SDN, pero no se considera la afectación del FCT para una red interna de DC en constante crecimiento. Evaluar como el FCT es afectado por el crecimiento de la red interna de un DC, permitirá a futuro desarrollar mejoras al controlador SDN que permitan mejorar su desempeño frente a un escenario en constante crecimiento.

3. Marco teórico referencial

A continuación, se presenta un marco teórico con respecto a las arquitecturas de red de centros de datos (DCN), el comportamiento interno del tráfico de red de los DC. Posteriormente se revisa el tema de controladores SDN y finalmente las opciones de herramientas de Simulación y/o Emulación.

3.1 Arquitecturas de red de centros de datos (DCN)

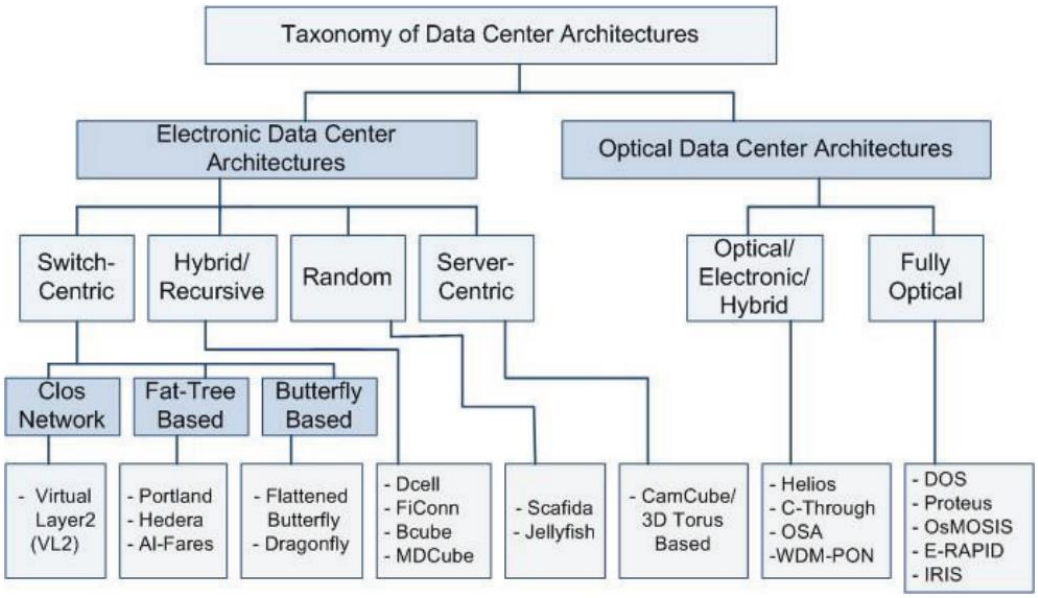
Las arquitecturas de los centros de datos se pueden clasificar en dos grandes categorías: Electronic Data Center Network Architectures (Arquitecturas de Redes de Centros de Datos electrónicos) y Optical Data Center Network Achitectures (Arquitecturas de Redes de Centro de Datos Ópticos).

Las arquitecturas de redes de centros de datos electrónicos se pueden dividir en arquitecturas basadas en switches, basadas en servidores, híbridas y aleatorias, mientras que las arquitecturas de redes de centros de datos ópticos se pueden subcategorizar como híbridas y completamente ópticas. La clasificación detallada se muestra en la Figura 1.

La topología de red básica de un data center se basa en una arquitectura de árbol compuesta de 3 niveles, el nivel más bajo llamado capa de acceso o Edge, el nivel intermedio llamado Aggregation y luego el nivel más alto llamado capa de Núcleo o Core.(Sharma & Mishra, 2020)

Figura 1

Taxonomía de arquitecturas de Data Center



Nota. A Comprehensive Survey on Data Center Network (Sharma & Mishra, 2020)

3.1.1 Arquitecturas de DCN basadas en Switches

En las arquitecturas de redes basadas en switches, los switches son el eje central para el diseño de estas topologías. Entre estas arquitecturas se destacan: Clos Network, fat-tree, butterfly based network (Sharma & Mishra, 2020).

La Clos Network es una topología de tres capas que se basa en una combinación de switches spine y leaf para permitir una mayor escalabilidad y redundancia.

La topología de fat-tree (Al-Fares et al., 2008), se caracteriza por utilizar múltiples capas de switches lo que la hace altamente escalable y también gozar de una alta disponibilidad gracias a la redundancia de la red. Fat-tree utiliza múltiples rutas para conectar servidores y/o dispositivos de red, lo que permite el equilibrio de carga que contribuye a reducir la congestión de la red.

Finalmente, la red basada en butterfly se caracteriza por tener una topología de malla bidimensional, donde cada switch se conecta directamente a un número estático de switches adyacentes para maximizar la conectividad y reducir los cuellos de botella.

3.1.1.1 Arquitectura Clos Network

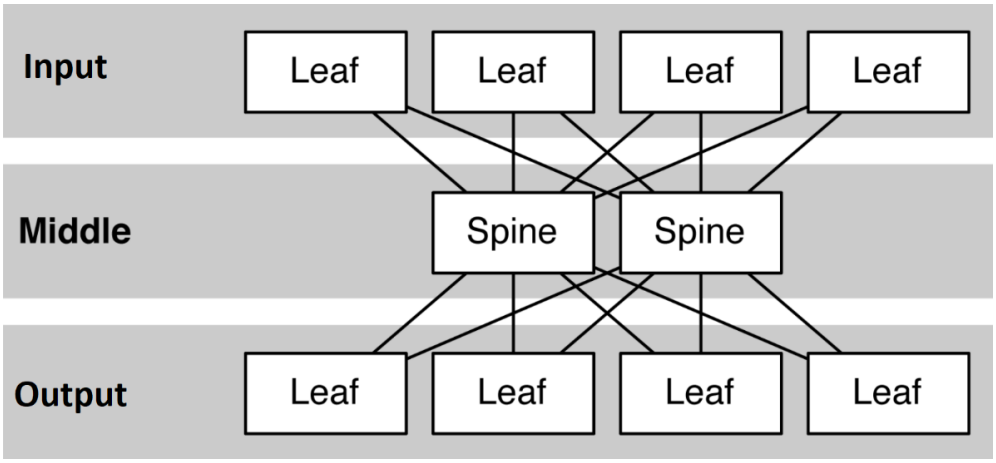
Una CLOS Network (también llamada Spine-Leaf Network) típica consta de tres niveles (o etapas) de switches que se denominan switches de entrada(input), medio(middle) o Interconexión y salida (output) como se ilustra en la figura 2.

Cada switch Leaf se conecta a todos los switches Spine, en consecuencia, cada switch Spine se conectará a cada switch Leaf. No puede haber conexiones entre switches Leaf, y no puede haber conexiones entre switches Spine.

La topología Clos define una red de tal modo que cada switch principal se encuentra conectado a todos los switches en el nivel inferior para aumentar el rendimiento, reducir el diámetro y la latencia de la red.

Figura 2

Clos Network de 3 etapas



Nota. https://www. quora.com/What-is-a-Clos-network

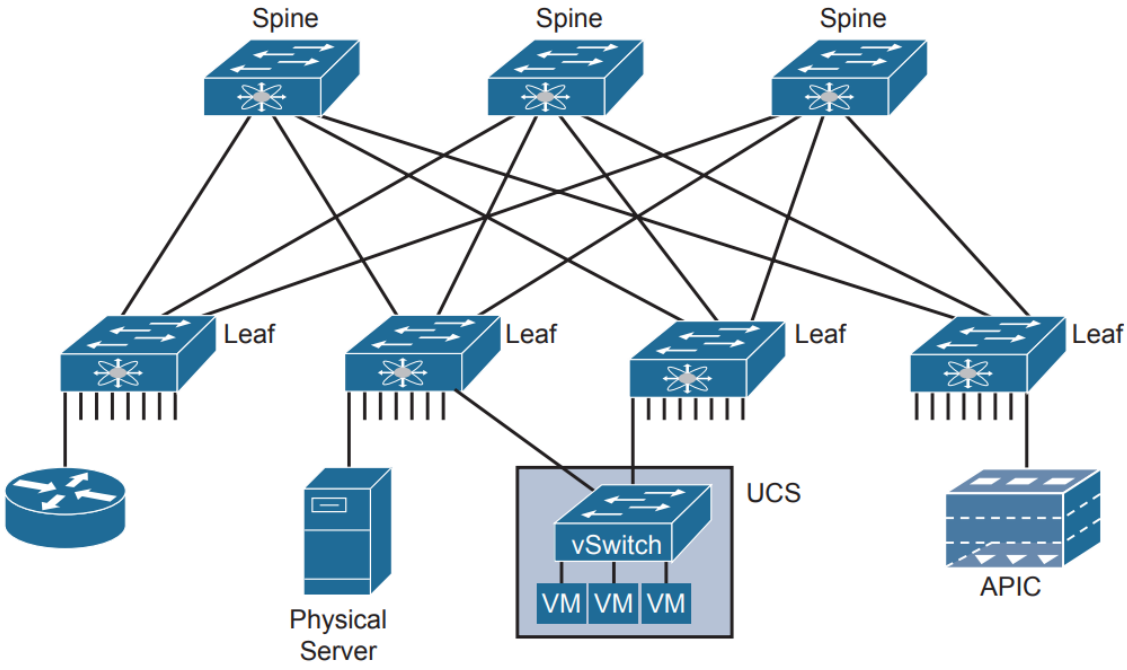
La topología CLOS es escalable horizontalmente, lo que resulta muy rentable. La capacidad de ancho de banda entre servidores se puede aumentar agregando más enlaces spine-leafs y más switches spine.(quora.com, n.d.)

Los servidores, también llamados nodos terminales, se enlazan directamente a los switches Leaf, evitando cualquier conexión con los switches Spine. En cuanto a cantidad, la gran mayoría de estos nodos terminales son servidores físicos operando con sistemas operativos convencionales, o bien, servidores que soportan virtualización, albergando una significativa cantidad de máquinas virtuales y

contenedores. Como se muestra en la figura 3 un servidor Cisco UCS (Unified Computing System) puede estar conectado a más de un switch Leaf, tanto para garantizar la redundancia como para aumentar la capacidad para soportar las máquinas virtuales y contenedores que se ejecutan en el hardware UCS.(Wendell Odom, 2020)

Figura 3

Puntos finales (también llamados nodos terminales) se encuentran solo en los switches Leaf



Nota. CCNA 200-301 Official Cert Guide, (Wendell Odom, 2020)

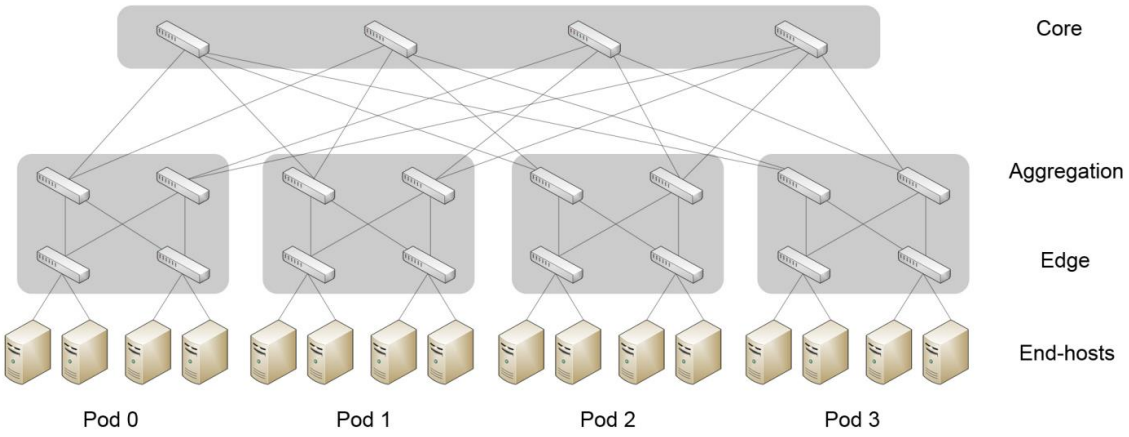
3.1.1.2 Arquitectura Fat-tree

La arquitectura Fat-tree es una arquitectura de red propuesta por (Al-Fares et al., 2008). Y es una variación de la arquitectura CLOS. Se encuentra organizado de la siguiente manera: Hay k pods, cada pod contiene 2 capas de k/2 switches (capa Aggregation y capa Edge), cada switch en la capa inferior (Edge) contiene k puertos, de los cuales K/2 puertos de cada switch Edge están conectados a k/2 end-hosts y los k/2 restantes están conectados a k/2 puertos de los k puertos de cada switch de la capa aggregation. Hay k/2 grupos con k/2 switches Core de k puertos. Cada switch core posee un puerto conectado a cada uno de los k pods. El número total

de host que soporta la topología es de $k^3/4$. En las figuras 4 y 5 se puede observar configuraciones Fat-tree con k=4 y k=6.

Figura 4

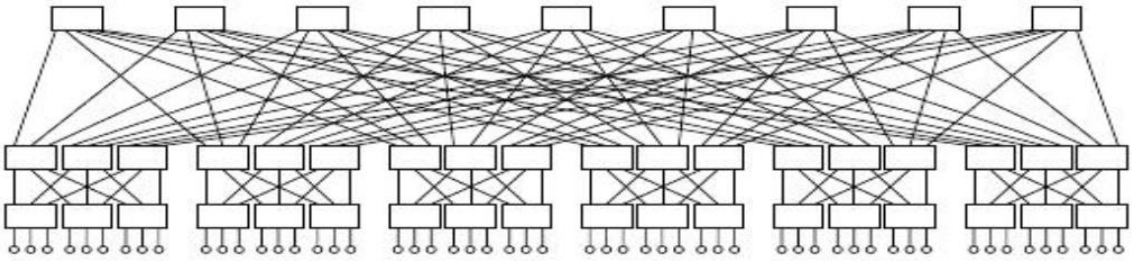
Arquitectura de DCN Fat-tree con k=4



Nota. Towards reproducible performance studies of datacenter network architectures using an open-source simulation approach, (Daji Wong et al., 2013)

Figura 5

Arquitectura de DCN Fat-tree con k=6



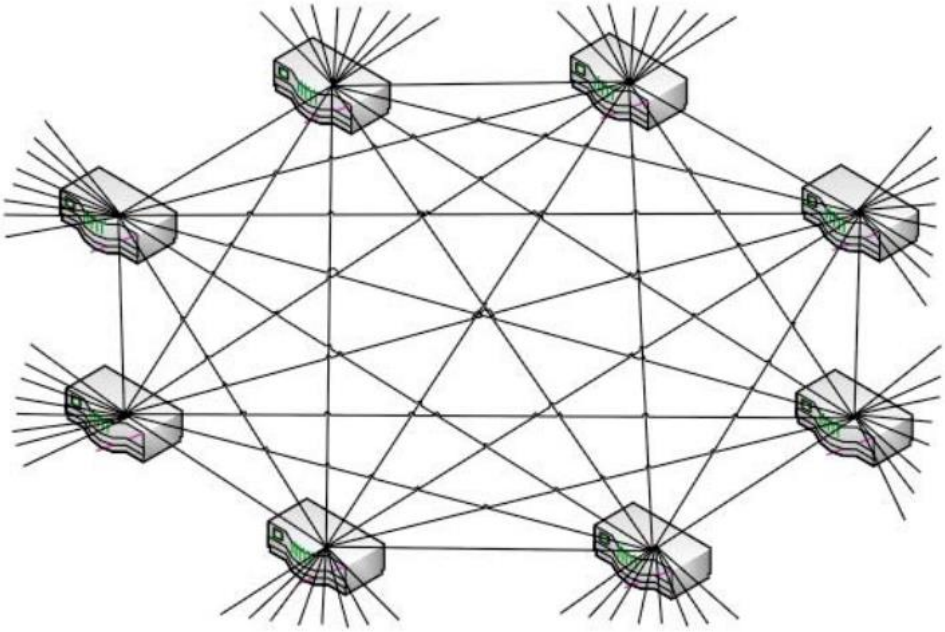
Nota. Data Center Network Topologies: FatTree (Cornell University, 2014)

3.1.1.3 Arquitectura Butterfly Based Network

Entre las arquitecturas Butterfly destaca la Flattened Butterfly (Yao et al., 2014). Los nodos se organizan en una malla bidimensional, y cada nodo está conectado a un número fijo de vecinos cercarnos en un patrón específico. Como se muestra en la figura 6, cualquier par de nodos puede comunicarse entre sí mediante una ruta corta.

Figura 6

Flattened butterfly



Nota. A comparative Analysis of Data Center Network Architectures, (Yao et al., 2014)

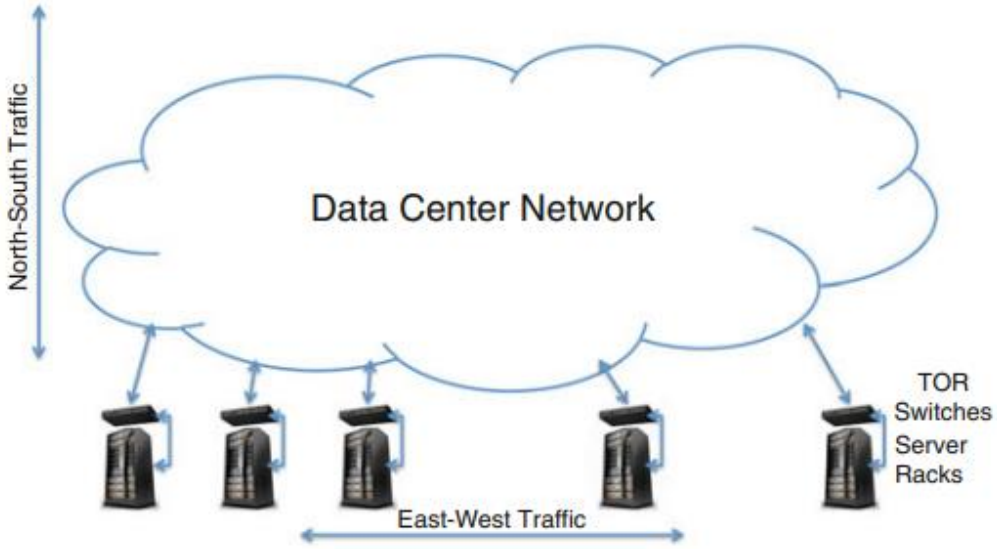
3.1.2 Tipos de trafico de DCN

Como indica (da Fonseca & Boutaba, 2015) el tráfico de DCN se puede dividir en 2 categorías: north/south (norte/sur) y east/west (este/oeste).

El tráfico de red de tipo “este-oeste” se refiere a la interacción entre dispositivos dentro de una red interna, es decir, dentro de un centro de datos o una infraestructura en la nube. A diferencia del tráfico norte-sur, que fluye desde el exterior de la red hacia el centro de datos (entre los usuarios externos y el centro de datos) y viceversa. El tráfico “este-oeste” se produce entre servidores, aplicaciones y servicios que se encuentran dentro de la red interna. Este tipo de tráfico es especialmente importante en entornos en la nube, donde las aplicaciones y los servicios están distribuidos en varios servidores y se comunican entre sí para realizar tareas complejas. El tráfico este-oeste también se utiliza para la replicación de datos, la recuperación ante desastres y el equilibrio de carga. Podemos apreciar estos tipos de tráfico en la figura 7.

Figura 7

Red Básica de Centro de Datos



Nota. Data Center Network Topologies (Liu et al., 2013)

Para gestionar y optimizar el tráfico este-oeste, se utilizan herramientas de monitoreo de redes y de análisis de tráfico, así como técnicas como la segmentación de red. Además, las arquitecturas de red modernas, como las redes definidas por software (SDN) y las redes virtualizadas, pueden ayudar a mejorar la eficiencia y la seguridad del tráfico este-oeste.

Según (da Fonseca & Boutaba, 2015; Fischer e Silva & Carpenter, 2015) el tráfico de tipo este-oeste representa más del 75% del volumen de tráfico total de un DCN.

3.1.2.1 Propiedades de tráfico de DCN

El tráfico de red de un DCN se caracteriza por flujos, donde cada flujo está identificado por secuencias de paquetes. Según (da Fonseca & Boutaba, 2015) , el flujo de tráfico de los DCN se puede clasificar en 2 categorias: elephant y mice. Los flujos de elephant son aquellos que envían una gran cantidad de paquetes en un corto período de tiempo, son de larga duración y tienen un comportamiento intermitente. Por otro lado, los flujos mice son aquellos que envían un número pequeño de paquetes y tienen una duración corta.

Según da Fonseca & Boutaba, 2015, en estudios de medición se ha encontrado que el flujo de tráfico en un DCN proviene de un pequeño número de flujos elephant, mientras que la mayoría de los flujos de tráfico son de tipo mice. Los flujos de tipo "mice" representan aproximadamente el 99% del número total de flujos en la red, y suelen tener un tamaño de menos de 10 Kilobytes, mientras los flujos de tipo "elephant" representan solo alrededor el 1% del número total de flujos en la red y en tamaño pueden alcanzar la decena de megas (da Fonseca & Boutaba, 2015).

Adicionalmente se puede caracterizar el tráfico de la red y sus flujos de la siguiente manera:

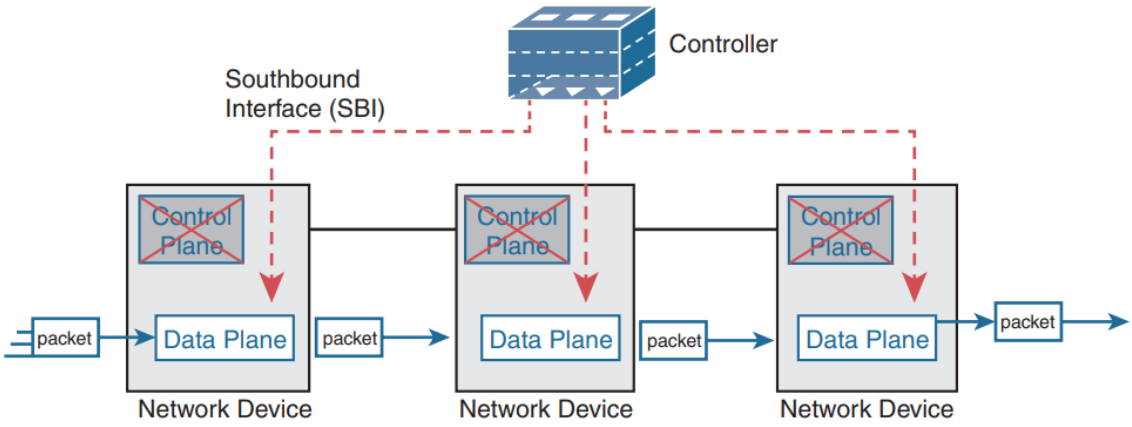
- **Ubicación e intercambio del tráfico general:** La mayoría del tráfico generado por los servidores (aproximadamente el 80%) se mantiene dentro de los racks (da Fonseca & Boutaba, 2015). De tal manera que el tráfico de tipo este-oeste representa esta mayoría.
- **Patrón del flujo de tráfico de un DCN:** El tiempo de transmisión entre paquetes tiene un comportamiento ON/OFF. En el que los periodos de actividad o encendido (ON), se transmiten paquetes IP y en el tiempo de inactividad o apagado (OFF), no se transmiten. El periodo OFF corresponde a los tiempos entre la llegada de un paquete y el siguiente.
    - La duración de los periodos ON/OFF es aleatoria y sigue una distribución exponencial (Daji Wong et al., 2013), que se ha demostrado que esta distribución modela de una manera razonable los flujos de tráfico de un DCN (Benson et al., 2010).

3.2 Redes definidas por software (SDN)

La Redes Definidas por Software (SDN, Software Defined Networking), tienen como objetivo mejorar la flexibilidad y programabilidad de la red. Como se muestra en la figura 8, esto se logra con la separación de la capa de control de la capa de datos en una red, permitiendo que el control se centralice en un controlador que toma las decisiones de cómo se deben reenviar los paquetes.

Figura 8

Plano de control centralizado



Nota. CCNA 200-301 Official Cert Guide, (Wendell Odom, 2020)

3.2.1 SouthBound Interface (SBI)

En una arquitectura de red basada en redes definidas por software, el controlador (Plano de control) necesita comunicarse con los dispositivos de red (Plano de datos). Estos dispositivos de red típicamente se encuentran debajo del controlador. Existe una interfaz entre el controlador y estos dispositivos de red (como se muestra en la figura 8), y dada su ubicación se las denomina SouthBound Interface o SBI.

En el SBI es donde entran en funcionamiento APIs y/o Protocolos para que pueda llevarse a cabo la comunicación entre el plano de control y el plano de datos, entre estos existen OpenFlow, OpFlex, OnePK, NETCONF (Wendell Odom, 2020). Por ejemplo, se puede destacar que NETCONF utiliza el formato de datos XML para obtener los datos del plano de datos. Mientras el protocolo OpenFlow es la interfaz southbound más utilizada (Blial et al., 2016).

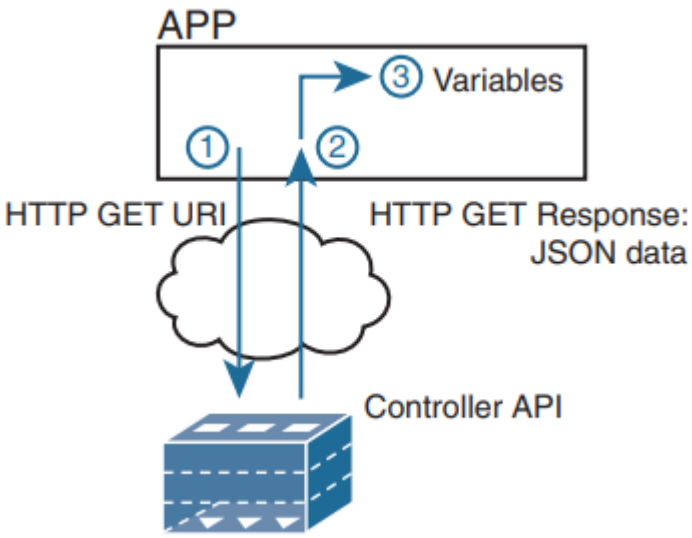
3.2.2 NorthBound Interface (NBI)

Las interfaces NorthBound (NBI) permiten la interacción entre el controlador centralizado de la red y las aplicaciones o servicios externos que se ejecutan en la capa de aplicación. La NBI permite que otros programas utilicen los datos y funciones del controlador para habilitar la programación de redes de manera más rápida. Por ejemplo, una API REST, las API REST se basan en solicitudes HTTP

solicitadas al controlador las cuales se responden en un formato de datos estructurados como JSON o XML, en la figura 9 podemos observar esta interacción.

Figura 9

Ejemplo de la comunicación de una API-REST con un controlador SDN



Nota. Official Cert Guide CCNA 200-301 volume2, (Wendell Odom, 2020)

3.2.3 Arquitecturas de controladores SDN

En las arquitecturas de controladores SDN podemos encontrar 2 tipos: Arquitecturas lógicamente centralizadas y arquitecturas lógicamente distribuidas (Blial et al., 2016).

- Lógicamente centralizada: Es un diseño multicontrolador, pero al mismo tiempo, siempre consideramos que tenemos un único controlador. En otras palabras, tomamos la carga y la distribuimos entre los múltiples controladores; sin embargo, para la capa subyacente, es como si hubiera un solo controlador que controla toda la red (Blial et al., 2016).
- lógicamente distribuida: los controladores están distribuidos física y lógicamente. Además, cada controlador solo tiene una vista del dominio del que es responsable y puede tomar decisiones por él, a diferencia de un diseño lógicamente centralizado, donde cada controlador toma una decisión basada en la vista de la red global (Blial et al., 2016).

En resumen, una arquitectura lógicamente centralizada se mantiene cerca de la tendencia inicial de SDN, que utiliza un solo controlador o un controlador multinúcleo para mejorar el rendimiento (Blial et al., 2016).

3.2.4 Controladores SDN de arquitecturas logicamente centralizadas

Entre los controladores de arquitecturas lógicamente centralizadas, destacan: ONIX, Hyperflow, ONOS, Disco (Blial et al., 2016).

3.2.4.1 ONIX

Es una plataforma de control distribuido para redes a gran escala. ONIX se puede ejecutar en clúster de uno o más servidores físicos, cada uno de los cuales puede ejecutar varias instancias de Onix. Una instancia de Onix también es responsable de difundir el estado de la red a otras instancias dentro del clúster (Koponen et al., 2019). La construcción del controlador de ONIX esta basado en los siguientes desafios: generalidad, escalabilidad, confiabilidad, simplicidad y rendimiento.

ONIX ha sido un paso importante en desarrollo de soluciones SDN, ya que ha ayudado a sentar las bases para la construcción de redes más flexibles y adaptables. No obstante, es importante señalar que ONIX es un proyecto que ha sido descontinuado y sustituido por soluciones de controladores SDN más actuales y sofisticados(Koponen et al., 2019).

Algunas de las caracteristicas clave incluían:

- Capacidad de extensiones: ONIX era altamente modular y extensible, lo que permitía a los desarrolladores agregar y modificar funcionalidades según las necesidades de la red.

- Soporte de multiples protocolos de control de red: ONIX era compatible con varios protocolos de control de red, entre ellos OpenFlow, lo que facilitaba la interoperabilidad entre dispositovos de red de diferentes fabricantes.

3.2.4.2 HYPERFLOW

HyperFlow esta desarrollada sobre el controlador NOX para permitir aquitectutas multicontrolador lógicamente centralizadas. HyperFlow se organiza en tres capas: control, reenvío y aplicación. La capa de control esta compuesto por varios controladores NOX que trabajan en conjunto. En la capa de reenvío, los switches se conectan al controlador más cercano, pero pueden ser reasignados a otro controlador en caso de falla.

Según autores del estudio (Blial et al., 2016; Tootoonchian & Ganjali, 2010) han encontrado que los controladores basados en HyperFlow pueden funcionar con mayor suavidad bajo una carga pesada de sincronización y mantener una latencia mínima en comparación con los controladores NOX. Asi mismo, han demostrado que HyperFlow puede mantener una cantidad aceptable de consistencia entre los controladores para alrededor de 1000 eventos que llegan por segundo, como la conexión y desconexión de switches y hosts en la red y un cambio en el estado del enlace. No obstante, han señalado que hay un ligero aumento en el tiempo de respuesta cuando un controlador converge o alcanza la sincronización de la red.

3.2.4.3 ONOS

ONOS (Open Network Opertating System), es un controlador SDN de código abierto líder para crear soluciones SDN. ONOS fue desarrollado por Open Networking Foundation (ONF) y esta diseñado para ser altamente escalable, flexible y programable. ONOS utiliza un modelo de arquitectura lógicamente centralizada. ONOS puede agregar instancias adicionales para distribuir la carga de trabajo en el plano de control cuando se escala horizontalmente (Berde et al., 2014).

Tolerancia a fallos: La arquitectura distribuida de ONOS permite que el sistema continúe funcionando cuando falla un componente o una instancia de ONOS redistribuyendo el trabajo a otras instancias restantes. ONOS permite que los componentes existan como una sola instancia en tiempo de ejecución, pero proporciona la capacidad de tener múltiples instancias redundantes esperando para tomar el control en caso de que la instancia principal falle. Esto se logra eligiendo

un líder entre todas las instancias en tiempo de ejecución para alojar la instancia principal.(Berde et al., 2014)

ONOS mantiene una vista global de red para gestionar y compartir el estado de la red entre los servidores ONOS en un clúster. Esta abstracción proporciona un modelo de gráfico de la red que se corresponde naturalmente con la estructura de la red subyacente. La topología y el estado de red descubiertos por cada instancia de ONOS se utilizan para construir la vista global de red. En general, ONOS es una plataforma de control SDN distribuida que se enfoca en la escalabilidad y tolerancia a fallos, con una vista global de red y un enfoque en la mejora del rendimiento.(Berde et al., 2014)

Onos y sus funciones principales se encuentran escritos en Java. ONOS se ejecuta sobre Java Virtual Machina (JVM), lo que permite que pueda ejecutarse en varios sistemas operativos. (*Https://Wiki.Onosproject.Org/Display/ONOS/ONOS*, n.d.)

3.2.4.4 DISCO

DISCO es un controlador SDN (Software-Defined Networking) distribuido y de código abierto diseñado para redes definidas por software de múltiples dominios. Se basa en una arquitectura modular y escalable que utiliza un canal de control ligero y altamente manejable para la comunicación entre controladores de diferentes dominios(Phemius et al., 2014).

Cada controlador DISCO es responsable de un dominio SDN y se comunica con los controladores vecinos para intercambiar información de red a nivel global para la gestión de flujo de extremo a extremo. El protocolo de comunicación utilizado por DISCO es AMQP (Advanced Message Queuing Protocol), un protocolo de mensajería de red que permite la transmisión de mensajes entre aplicaciones en una red.

DISCO es un sistema de gestión de redes que tiene como objetivo abordar problemas como enlaces rotos, alta latencia y ancho de banda. Consta de dos partes principales: intradominio e interdominio(Phemius et al., 2014).

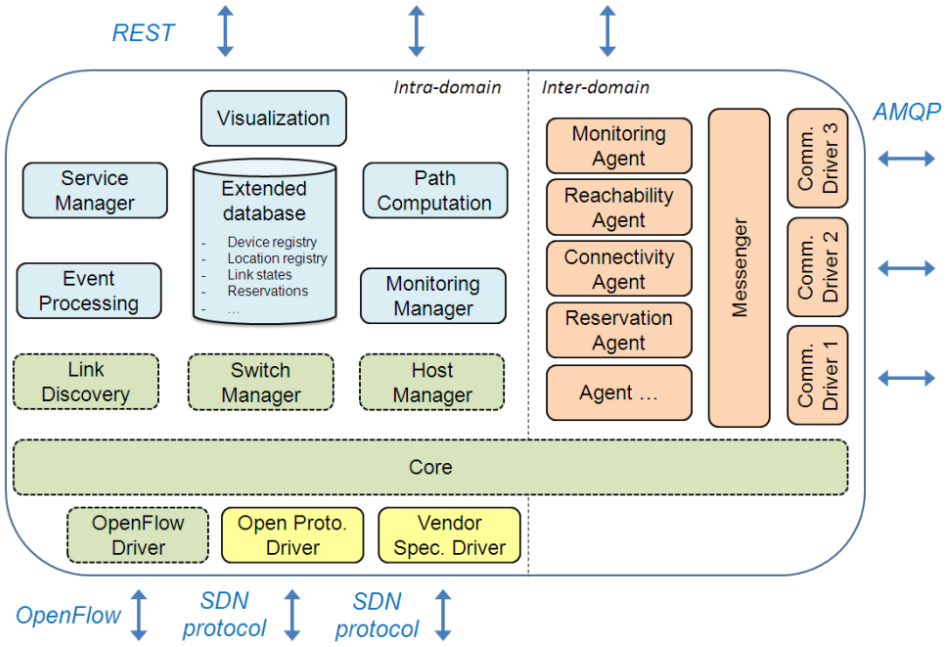
La parte intradominio se encarga de supervisar la red y gestionar la priorización del flujo de datos. Incluye varios módulos, como la Base de Datos Extendida (Extended database), el Módulo de Administración de Monitoreo (Monitoring Manager), el Módulo de Procesamiento de Eventos (Event Processing), el Módulo de Cálculo de Rutas (Path Computation) y el Módulo de Administración de Servicios (Service Manager). Estos módulos trabajan en conjunto para monitorear la red y garantizar un rendimiento óptimo.

La parte interdominio se centra en la comunicación entre los múltiples controladores de la red. Contiene dos módulos principales: el Módulo de Mensajería y el Módulo de Agentes. El Módulo de Mensajería facilita la comunicación entre controladores vecinos, mientras que el Módulo de Agentes contiene cuatro agentes principales que se encargan de diferentes aspectos de la gestión de la red, como conectividad, monitoreo, accesibilidad y reserva de recursos.

Se puede observar la arquitectura de DISCO en la figura 10.

Figura 10

Arquitectura del controlador DISCO



Nota. DISCO: Distributed Multi-domain SDN Controllers, (Phemius et al., 2014)

3.3 Simuladores y emuladores de red

Existe una serie de herramientas de simulación y emulación de redes muy usadas por la comunidad científica, entre estas destacan EstiNet, ns-3 y Mininet.

3.3.1 Estinet 11

EstiNet11 es una innovadora plataforma comercial de simulación/emulación de redes que puede ejecutar una red emulada de forma "simulada".

En el contexto del simulador EstiNet11, la integración efectiva de la pila de protocolos TCP/IP y UDP/IP del kernel de Linux en entornos de red simulados garantiza una representación fidedigna del funcionamiento de los protocolos de las capas 3 y 4, potenciando así el desempeño de las aplicaciones de red en el ámbito de la simulación.(*Estinet Tecnologies Inc.*, n.d.)

Las aplicaciones basadas en Linux, que comúnmente se ejecutan en máquinas reales con dicho sistema operativo, pueden funcionar directamente en los nodos simulados de EstiNet11. La habilidad de ejecutar programas auténticos implica que los comportamientos de las redes del mundo real pueden ser replicados con exactitud en la red simulada de EstiNet11. Esta característica resulta valiosa al desarrollar y evaluar funcionalidades y desempeño de nuevas aplicaciones utilizando EstiNet11, así como al probar una red de reciente creación empleando EstiNet11 y aplicaciones de red de la vida real.

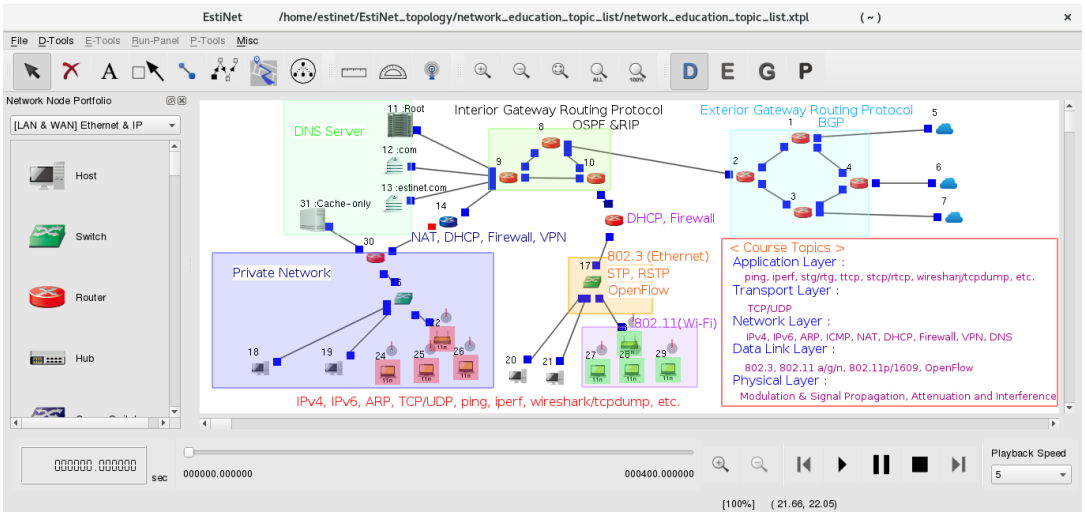
EstiNet11 puede alternar entre el modo de simulación y el modo de emulación en. Al cambiar al modo de emulación, una red o dispositivo simulado puede enviar y recibir paquetes desde y hacia dispositivos reales del mundo. En otras palabras, una red emulada incluye tanto nodos de red simulados como dispositivos de red auténticos. Este enfoque resulta beneficioso para evaluar el comportamiento de dispositivos de red de reciente desarrollo. En la figura 11 se puede observar un ejemplo de la interfaz del simulador EstiNet11.(*Estinet Tecnologies Inc.*, n.d.)

Características clave (*Estinet Tecnologies Inc.*, n.d.):

- Capa de Aplicación:
    - HTTP, FTP, DHCP, NAT, VPN, DNS, Firewall, SSH, telnet, tcpdump and Wireshark.
    - Otras aplicaciones basadas en Linux y generadores de tráfico.
- Capa de Transporte
    - TCP, UDP
- Capa de Red
    - IPv4 & IPv6
    - - ICMPv4 & ICMPv6
    - - Routing Daemon
    - - OSPF, RIP, BGP
    - - OLSRD2
- Capa de enlace de datos
    - IEEE 802.3
    - - CSMA/CD
    - - IEEE 802.11a/g
    - - CSMA/CA & EDCA
    - - Coding & Decoding
    - - IEEE 802.11n
    - - CSMA/CA & EDCA
    - - Coding & Decoding
    - - Block Ack & A-MPDU
- Capa Física
    - Transmission Delay
    - - Propagation Delay
    - - Modulation & Demodulation
    - - Channel Model (for Wireless Channel)
    - - Tx Power
    - - Tx/Rx Antenna Gain Pattern
    - - Signal Attenuation
    - - Rx Sensitivity

Figura 11

Interfaz de Estinet



Nota. https://www.estinet.com/ns/?page_id=21140, (*Estinet Tecnologies Inc.*, n.d.)

3.3.2 NS-3

El simulador de redes ns-3 es un software de eventos discretos diseñado principalmente para investigación y propósitos educativos. Licenciado bajo GNU GPLv2, ns-3 es de código abierto y accesible para su uso en investigación y desarrollo.

El propósito del proyecto ns-3 es establecer un entorno de simulación abierto y preferente en el ámbito de la investigación de redes, alineado con las necesidades actuales en el campo y promoviendo la colaboración comunitaria, la revisión entre pares y la validación del software. (https://www.nsnam.org/, n.d.)

Modelos de simulación

El proyecto ns-3 está dedicado a desarrollar un motor de simulación robusto, ampliamente documentado y de fácil manejo y corrección de errores. Este cubre integralmente las necesidades de simulación, desde la fase de configuración hasta la recolección y evaluación de datos.

La plataforma de software ns-3 impulsa el desarrollo de modelos de simulación lo suficientemente precisos para que ns-3 pueda funcionar como emulador de red en

tiempo real, conectándose con el mundo real y permitiendo la reutilización de numerosas implementaciones de protocolos existentes en ns-3. (https://www.nsnam.org/, n.d.)

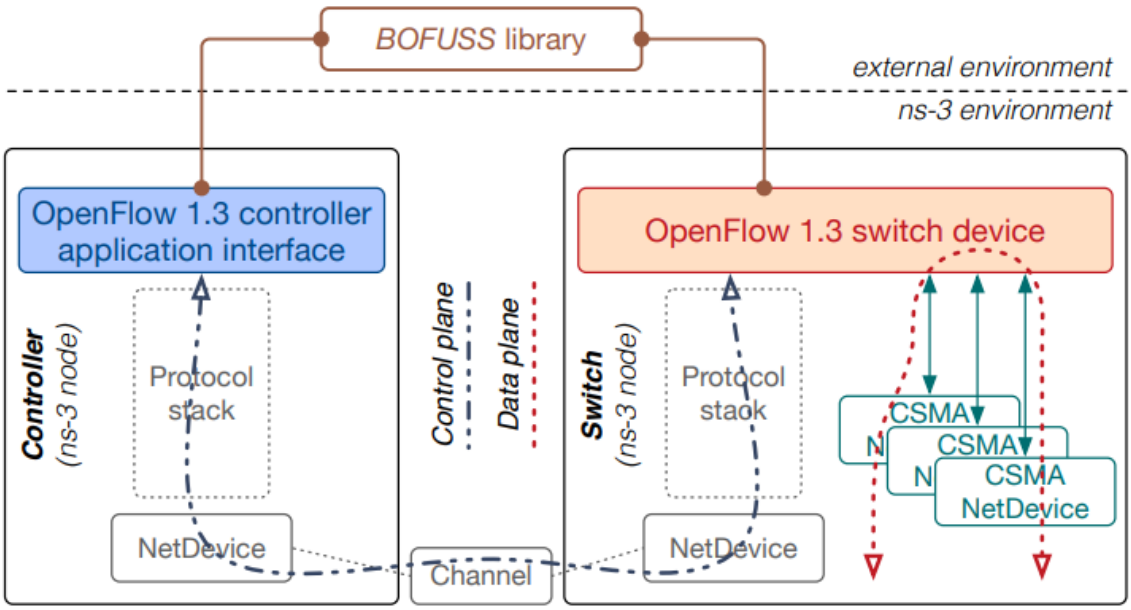
El núcleo de simulación de ns-3 es compatible con investigaciones en redes tanto IP como no IP. No obstante, la mayoría de los usuarios se enfoca en simulaciones de redes inalámbricas/IP, que incluyen modelos para Wi-Fi, WiMAX o LTE en las capas 1 y 2, y diversos protocolos de enrutamiento estático o dinámico, como OLSR y AODV, para aplicaciones basadas en IP. (https://www.nsnam.org/, n.d.)

Compatibilidad SDN

Ns-3 cuenta con un módulo OpenFlow nativo que permite simular switches OpenFlow, sin embargo, este se encuentra desactualizado. Existe una alternativa que es un módulo independiente adicional llamado OFSwitch13 (Luciano Jerez Chaves et al., n.d.), el cual es compatible con la versión 1.3 de OpenFlow. Este módulo permite la comunicación con controladores SDN externos compatibles con OpenFlow1.3. En la figura 12 se puede observar la arquitectura del módulo OFSwitch para ns-3.

Figura 12

Arquitectura del Módulo OFSwitch13



Nota. http://www.lrc.ic.unicamp.br/ofswitch13/ofswitch13.pdf, (Luciano Jerez Chaves et al., n.d.)

3.3.3 MININET

Mininet es una potente herramienta que permite crear una red virtual realista en una sola máquina (ya sea una máquina virtual, en la nube o nativa). Con tan solo un comando, Mininet despliega una red completa que incluye código de kernel, switches y aplicaciones reales. Esto se logra mediante la virtualización de los componentes de la red, lo que permite simular de manera eficiente y rápida diferentes topologías y escenarios.

Una de las ventajas de Mininet es su facilidad de uso e interacción. A través de su interfaz de línea de comandos (CLI) y API, los usuarios pueden interactuar directamente con la red virtual, personalizarla según sus necesidades, compartirla con otros y, si lo desean, implementarla en hardware real. Esto hace que Mininet sea una herramienta muy útil tanto para el desarrollo de soluciones de red, como para la enseñanza y la investigación.

Mininet es una excelente opción para el desarrollo y experimentación de sistemas de Redes Definidas por Software (SDN) utilizando tecnologías como OpenFlow y P4. Proporciona un entorno flexible para el desarrollo, la colaboración y la prueba de conceptos en el campo de SDN.

Es importante destacar que Mininet es un proyecto activamente desarrollado y respaldado por una comunidad comprometida. Se distribuye bajo una licencia de código abierto BSD permisiva, lo que fomenta la colaboración y la participación de la comunidad. Se anima a los usuarios a contribuir con código, informes de errores, correcciones, documentación y cualquier otra mejora que pueda enriquecer el sistema.

En resumen, Mininet es una herramienta versátil y poderosa que permite crear y simular redes virtuales de manera eficiente. Ya sea para desarrollo, enseñanza o investigación, Mininet ofrece una solución flexible y personalizable que facilita el trabajo con redes virtuales y promueve la innovación en el campo de las redes de computadoras.(http://mininet.org/, n.d.)

4. Materiales y metodología

Para llevar a cabo la evaluación del tiempo de finalización de flujo del tráfico interno de una red de centro de datos en crecimiento basado en SDN, se ha seleccionado la metodología Waterfall. Esta elección se fundamenta en su estructura secuencial y en etapas definidas, lo que permite abordar el proceso de simulación de manera organizada y efectiva, siguiendo un flujo de trabajo que comienza con la elección de la arquitectura de DCN y culmina en la evaluación de resultados.

Las etapas clave de la metodología incluyen:

- Elección de la arquitectura de DCN.
- Elección del controlador de redes definidas por software (SDN).
- Elección del software de simulación de redes.
- Simular en varias escalas una arquitectura de red de un centro de datos (DCN).
- Analizar los resultados obtenidos.

Elección de la arquitectura de DCN

En esta etapa inicial, se determina la arquitectura de centro de datos (DCN) que se utiliza como objeto de estudio. Esta elección se basa en una evaluación comparativa que incluye las arquitecturas Clos network, Flattened butterfly y Fat-tree. Se toma en consideración la elección de la arquitectura Fat-tree. Esta arquitectura se origina como una evolución de la Clos network (Al-Fares et al., 2008), y goza de amplio reconocimiento en la comunidad científica (Al-Fares et al., 2008; S. Hu et al., 2021). Además, su influencia se extiende a implementaciones de redes en centros de datos de gran envergadura, como la utilizada por Google (Poutievski et al., 2022). Adicionalmente la arquitectura Fat-tree se destaca por su capacidad de escalabilidad y expansión.

Proceso de elección**:**

Tanto Fat-tree, Clos network y Flattened Butterfly son compatibles y se benefician de la implementación de SDN.(Al-Fares et al., 2008; Curtis et al., 2011; Greenberg et al., 2009; S. Hu et al., 2021; Kim et al., 2008; Poutievski et al., 2022; Vesović et al., 2022). La red definida por software (SDN) puede mejorar la gestión, la escalabilidad y la adaptabilidad de la red en todas estas topologías.

Las diferencias en las arquitecturas se vuelven menos significativas cuando se implementa SDN, ya que la gestión centralizada, la automatización y la capacidad de adaptarse dinámicamente a los cambios en el tráfico son características clave de SDN, que pueden mejorar el rendimiento de cualquier arquitectura de red de Data Center.

A continuación, en la tabla 1 se presenta una comparativa donde se muestra que estas arquitecturas coinciden en características clave como: Escalabilidad, Redundancia y Tolerancia a fallos.

Tabla 1

Tabla comparativa de arquitecturas DCN basadas en switches

Característica	**Fat-tree**	**Clos network**	**Flattened Butterfly**
Topología	Jerárquica, en árbol	Multietapa, en red	Multietapa, en red
Estructura	K-ary tree(Al-Fares et al., 2008)	(M, N, R) conexiones (3 etapas: entrada, interconexión, salida), (M: número de switches de entrada, N: número de switches de salida, R: número de etapas)	K-ary n-fly
Ancho de banda disponible	Alta (agregación de enlaces)	Alta (agregación de enlaces)	Alta (agregación de enlaces)
Escalabilidad	Alta (fácil de expandir)	Alta (modular)	Alta (modular)
Redundancia y rutas paralelas	Múltiples rutas (alta redundancia)	Múltiples rutas (alta redundancia)	Múltiples rutas (alta redundancia)
Balanceo de carga	Eficiente (múltiples rutas y balanceo de carga dinámico)	Eficiente (múltiples rutas y balanceo de carga dinámico)	Eficiente (múltiples rutas y balanceo de carga dinámico)

Tolerancia a fallos	Alta (rutas alternativas y recuperación rápida)	Alta (rutas alternativas y recuperación rápida)	Alta (rutas alternativas y recuperación rápida)
Facilidad de gestión	Fácil (topología regular y configuración homogénea). SDN Facilita la gestión y automatización.	Media (configuración de enrutamiento más compleja). SDN Facilita la gestión y automatización.	Media (configuración de enrutamiento más compleja). SDN Facilita la gestión y automatización.
Latencia	Baja (estructura en árbol)	Baja a media (depende de la configuración, (número de etapas))	Baja (enrutamiento directo y menos saltos). (proporcional al número de dimensiones)
Uso de enlaces	Uniforme (optimizado)	Variable (depende de la configuración)	Uniforme (optimizado)
Algoritmos de enrutamiento	Enrutamiento basado en estado del enlace (link-state) y caminos más cortos (shortest-path). Personalizable con SDN.	Dinámico y personalizable con SDN.	Dinámico y personalizable con SDN.

Se considera la elección de la arquitectura Fat-tree. Siendo esta una arquitectura que parte de la evolución de la arquitectura Clos network (Al-Fares et al., 2008), y muy utilizada en la comunidad científica (Al-Fares et al., 2008; S. Hu et al., 2021). Al ser una arquitectura basada en Clos Network, es también utilizada en implementaciones de grandes redes de centros de datos como la de Google (Poutievski et al., 2022). Adicionalmente es una arquitectura de fácil expansión en su escalabilidad.

Elección del controlador de redes definidas por software (SDN)

En esta etapa se procede a la elección del controlador SDN que supervisará y gestionará la arquitectura de red Fat-tree durante las simulaciones. En consonancia con la tendencia de SDN, que favorece el uso de controladores de arquitecturas lógicamente centralizadas (Blial et al., 2016), se realiza una evaluación de las opciones disponibles. Entre los controladores SDN que cumplen con esta premisa,

se consideran ONIX, Hyperflow, ONOS y Disco. Se llega a la conclusión de que el controlador SDN ONOS es la elección más adecuada para este proyecto. ONOS ha demostrado ser una herramienta de código abierto ampliamente aceptada por la comunidad de desarrolladores e investigadores (T.BAH et al., 2019).

Proceso de elección:

La tendencia de SDN es la de usar controladores de arquitecturas lógicamente centralizadas(Blial et al., 2016). Por lo tanto la elección del controlador SDN se realiza en base a controladores SDN que soporten esta premisa. Entre ellos destacan ONIX, Hyperflow, ONOS, Disco.

A continuación se presenta una tabla comparativa entre controladores de arquitecturas lógicamente centralizadas.

Tabla 2

Tabla comparativa de controladoras SDN

	ONIX	**HYPERFLOW**	**ONOS**	**Disco**
Arquitectura	Basado en un modelo de proceso	Basado en una arquitectura modular	Basado en una arquitectura modular y escalable	Basado en una arquitectura modular y escalable
Lenguaje de Programación	Python	Python y C++	Java	Python
Vigencia	No se encuentra en vigencia desde 2015	Vigente	Vigente	Vigente
Observaciones	Escalabilidad y flexibilidad. Falta de documentación	Alta escalabilidad y flexibilidad, programación nativa con lenguaje C++. Requiere conocimiento técnico avanzado. Falta de documentación para usuarios inexpertos	Amplia comunidad de usuarios y desarrolladores. Alta escalabilidad. Programación nativa con lenguaje Java. Servicio de este controlador con Dockers disponibles.	Fácil de aprender y usar. No es tan escalable como otros controladores, limitaciones de rendimiento

Todos los controladores SDN en la tabla 2 coinciden en su arquitectura escalable, sin embargo, la elección se inclina hacia el controlador ONOS al ser una herramienta de código abierto ampliamente respaldada por la comunidad de desarrolladores e investigadores (T.BAH et al., 2019).

Elección del software de simulación de redes

En esta etapa se aborda la elección del software de simulación de redes. Se realiza el análisis entre las herramientas de simulación/emulación de redes EstiNet, ns-3 y Mininet. Se opta por utilizar la herramienta de emulación Mininet. Esta elección se sustenta en varios factores significativos. En primer lugar, Mininet es una herramienta de código abierto ampliamente adoptada por la comunidad SDN, lo que asegura la disponibilidad de una amplia gama de recursos, tutoriales y soporte en línea. En segundo lugar, Mininet ha sido diseñado desde su concepción para funcionar nativamente con controladores SDN, lo que incluye una flexibilidad destacada en su compatibilidad con ONOS.

Proceso de elección:

A continuación, se presentan 2 tablas comparativas. La primera es una tabla comparativa entre las características de los simuladores/emuladores EstiNet, ns-3 y Mininet (tabla 3). La segunda tabla representa su compatibilidad con el controlador ONOS (tabla 4).

Tabla 3

Tabla comparativa de simuladores/emuladores de redes IP

Característica	EstiNet	ns-3	Mininet
Tipo de simulador	Híbrido	Simulador basado en eventos	Emulador
Lenguaje de programación	C++ y Python	C++ y Python	Python
Modelado de protocolos	TCP, UDP, IPv4, IPv6	TCP, UDP, IPv4, IPv6	TCP, UDP, IPv4, IPv6
Soporte SDN	Sí (OpenFlow)	Sí (OpenFlow)	Sí (OpenFlow)
Escalabilidad	Media	Alta	Baja-Media
Tiempo real	Sí	No	Sí
Interfaz gráfica	Sí (GUI)	No (CLI)	Sí (GUI)

Modelado de enrutadores	Sí	Sí	Sí
Modelado de switches	Sí	Sí	Sí
Modelado de host	Sí	Sí	Sí
Topologías de red	Estáticas y Dinámicas	Estáticas y Dinámicas	Estáticas y Dinámicas
Licencia	Comercial	GPLv2	BSD
Documentación	Media	Alta	Alta
Comunidad	Media-Pequeña	Grande	Grande

A continuación, se detalla la compatibilidad de los simuladores EstiNet, ns-3 y Mininet con el controlador ONOS.

Tabla 4

Tabla comparativa de compatibilidad de simuladores/emuladores con el controlador SDN ONOS.

Característica	EstiNet	ns-3	Mininet
Integración con ONOS	**Directa:** Diseñado para trabajar con controladores SDN, incluyendo ONOS.	**Indirecta:** Requiere un módulo externo para funcionar con ONOS (OFswitch13).	**Directa:** Mininet fue creado pensando en la compatibilidad con controladores SDN como ONOS.
Soporte OpenFlow	**Sí:** EstiNet es compatible con OpenFlow, lo que facilita la integración con ONOS.	**Sí:** ns-3 admite OpenFlow, pero la integración con ONOS requiere un módulo externo.	**Sí:** Mininet es compatible con OpenFlow, lo que facilita la integración con ONOS.
Versiones OpenFlow	EstiNet admite múltiples versiones de OpenFlow, lo que asegura la compatibilidad con diferentes versiones de ONOS.	ns-3 solo admite OpenFlow 1.0 y 1.3, lo que puede limitar la compatibilidad con ONOS.	Mininet admite múltiples versiones de OpenFlow, asegurando compatibilidad con diferentes versiones de ONOS.
Comunicación ONOS	La comunicación directa con el controlador remoto simplifica la integración de EstiNet con ONOS.	La comunicación entre ns-3 y ONOS requiere un módulo adicional, lo que complica la configuración y aumenta la latencia.	Mininet permite la comunicación directa con el controlador ONOS, tanto local como remoto, facilitando la integración.

Latencia	**Baja-Media:** La comunicación directa con ONOS puede resultar en una latencia baja-media. Esto debido a su enfoque hibrido.	**Alta:** La necesidad de un adaptador externo puede aumentar la latencia en la comunicación con ONOS.	**Baja:** La comunicación directa con ONOS puede resultar en una baja latencia.
Configuración	**Moderada:** La configuración de EstiNet para trabajar con ONOS es moderadamente fácil, ya que está diseñado para ser compatible con controladores SDN.	**Compleja:** La configuración de ns-3 para trabajar con ONOS es más compleja debido a la necesidad de un módulo externo.	**Sencilla:** La configuración de Mininet para trabajar con ONOS es sencilla, ya que nativamente fue creado en función de compatibilidad con controladores SDN Externos.
Soporte de aplicaciones ONOS	**Si:** EstiNet es compatible con la mayoría de las aplicaciones ONOS gracias a su diseño orientado a SDN.	**Parcial:** ns-3 puede tener un soporte parcial de aplicaciones ONOS debido a las limitaciones del adaptador externo. Pruebas locales entre ns-3 (utilizando el módulo ofswitch13) y ONOS se encuentran incompatibilidades de intercambio con el protocolo LLDP y paquetes BPDU de Spanning Tree.	**Sí:** Mininet es compatible con la mayoría de las aplicaciones ONOS debido a su enfoque en la compatibilidad con controladores SDN.
Actualización de complementos	Limitada debido a su licencia comercial y a la falta de soporte para la actualización de complementos.	Actualizable a través de módulos adicionales y extensión de código.	Fácilmente actualizable mediante la instalación de nuevos paquetes y complementos.

Consideraciones adicionales durante la elección de la herramienta de simulación/emulación de redes:

Se realizan pruebas con la herramienta de simulación ns-3, utilizando el módulo ofswitch13 para su conexión con el controlador externo ‘ONOS’. Sin embargo, estas

pruebas revelan una interacción irregular debido a errores en el intercambio de paquetes, principalmente con protocolos como LLDP y tramas BPDU de Spanning Tree Protocol (STP). Por otro lado, la herramienta de simulación Estinet, a pesar de ser una herramienta de pago, se constata que no es muy flexible con las más recientes actualizaciones de sus complementos.

En base de la tabla de evaluación y considerando las necesidades del proyecto, se opta por utilizar la herramienta de emulación Mininet. Mininet se destaca como la elección más adecuada debido a ser una herramienta de código abierto, y ampliamente utilizada por la comunidad SDN, lo que garantiza una amplia gama de recursos, tutoriales y soporte en línea. La capacidad de actualizar fácilmente y agregar nuevos paquetes y complementos también hace de Mininet una opción versátil y escalable para trabajar con ONOS en diferentes casos de uso y aplicaciones.

Simular en varias escalas una arquitectura de red de un centro de datos (DCN)

En esta etapa central, se configuran y ejecutan las simulaciones con diferentes escalas y configuraciones de la arquitectura Fat-tree. Esto incluye la simulación de topologías de k=4 y k=6, que se gestionan mediante el controlador ONOS.

- **Consideración para el número máximo de nodos a simularse en Mininet en un entorno SDN:** El límite máximo de nodos a simularse en Mininet en un entorno SDN, particularmente para las topologías de Fat-tree, se determina siguiendo un criterio específico. De acuerdo con el estudio realizado por (Chagas et al., 2021), Mininet fue sometido a pruebas simulando 12 topologías en un entorno SDN, con un número creciente de dispositivos que alcanzó los 109 nodos. En función del estudio mencionado y teniendo en cuenta la configuración de escalabilidad de la arquitectura Fat-tree, se decide simular en un ambiente controlado una configuración de escalabilidad de hasta k=6, donde se alcanza la cantidad de 99 nodos entre switches y host como se detalla en la Tabla 5. En resumen, el límite de nodos a simular en Mininet en un entorno SDN se justifica mediante la referencia

a investigaciones previas (Chagas et al., 2021), que han evaluado la capacidad de la herramienta, así como las consideraciones específicas de la arquitectura Fat-Tree.

- **Consideraciones para la generación del tráfico característico de un DCN**: Para generar el tráfico característico de un DCN y en base a estudios previos (Benson et al., 2010; da Fonseca & Boutaba, 2015) que indican el comportamiento característico del mismo. Se implementa en Mininet un script basado en Python tomando en cuenta las siguientes consideraciones:
    - **Modelo ON/OFF:** El comportamiento de transmisión de paquetes sigue un modelo ON-OFF, donde el estado ON corresponde a la transmisión activa de paquetes y el estado OFF indica la ausencia de tráfico.
    - **Duración de los periodos ON/OFF:** La duración de los periodos ON/OFF es aleatoria y sigue una distribución exponencial (Daji Wong et al., 2013), que se ha demostrado que esta distribución modela de una manera razonable los flujos de tráfico de un DCN (Benson et al., 2010).
- **Tamaño de los flujos:** Los tamaños de los flujos son aleatorios, pero uniformemente distribuidos en el intervalo entre 3000 y 10240 bytes (entre 3kb – 10kb), esta elección de tamaño se basa en el estudio de da Fonseca & Boutaba, 2015.
- **Configuración del controlador ONOS:** En función de los objetivos del presente proyecto, solo se activan funciones básicas como la conmutación del tráfico y prevención de bucles.

Resumen de la configuración para los escenarios de simulación:

A continuación, se presenta una tabla que resume las configuraciones para los escenarios de simulación en la arquitectura Fat-tree con dos configuraciones diferentes: K=4 y K=6 (Tabla 5).

Tabla 5

Configuración de los escenarios de simulación Fat-tree con k=4 y K=6

	Fat-tree k=4	Fat-tree k=6
Número de Switches	20	45
Número de host (servidores)	16	54
Número de enlaces	48	162
Ancho de banda de los enlaces	1 Mbps	1 Mbps
Tamaño máximo de cada trama	100 bytes	100 bytes
Protocolo	UDP	UDP
Tamaño de flujos	Entre 3kB-10kB	Entre 3kB-10kB

La simulación es realizada bajo los siguientes recursos de cómputo (tabla 6):

Tabla 6

Recursos de computo

Hipervisor	Tipo 1 - VMware ESXi
CPU	4 CPU Virtuales
RAM	32 GB
Sistema Operativo	Ubuntu 22.04.2 LTS

Analizar los resultados obtenidos

La evaluación de los tiempos de finalización de flujo (FCT - Flow Completion Time), obtenidos a través de la simulación, se realiza mediante el uso de la Función de Distribución Acumulativa (CDF). La CDF proporciona una visión de la distribución estadística de los FCT en la simulación, lo que puede ser útil para analizar el rendimiento y la eficiencia de la red. En esta evaluación, primero se compara el desempeño individual de los FCT con los tiempos de transmisión teóricos ideales. Posteriormente, se realiza una comparación entre las CDF de los FCT para los valores de k=4 y k=6.

El cálculo del FCT por cada flujo se lo realiza con la siguiente fórmula:

- FCT = Tiempo de llegada del último paquete - Tiempo de salida del primer paquete

El cálculo del tiempo de transmisión ideal teórico por cada flujo se realiza con la siguiente fórmula:

- Tiempo ideal = (Cantidad de bits a transmitir) / (Tasa de transmisión en bits)

5. Resultados y discusión

El escenario contempla la simulación de dos topologías de arquitectura de red fat-tree, una con configuración de escalabilidad de k=4 y otra k=6. Gestionadas bajo un único controlador SDN ONOS, con funciones básicas activadas de conmutación de tráfico y prevención de bucles. La generación de tráfico en la simulación sigue un patrón ON/OFF y utiliza el protocolo UDP. Cada flujo de tráfico tiene un tamaño máximo de 10 kB y los paquetes individuales tienen un tamaño máximo de 100 bytes. Por cada host de la topología se generan 10 flujos entre pares de host aleatorios.

Luego de la generación de tráfico se calcula el FCT resultante de cada flujo. El FCT se calcula como la diferencia entre el tiempo en que llega el último paquete de un flujo y el tiempo en que sale el primer paquete de ese mismo flujo. Este cálculo permite medir el tiempo total que lleva transmitir un flujo de datos a través de la red. Además, se calcula el tiempo de transmisión ideal teórico para cada flujo, que se obtiene dividiendo la cantidad de bits a transmitir por la tasa de transmisión en bits.

Finalmente, se utiliza la librería Pandas de Python para calcular la Función de Distribución Acumulativa (Cumulative Distribution Function, CDF por sus siglas en ingles), que proporciona una visión de la distribución estadística para evaluar los FCT resultantes de las topologías simuladas fat-tree con configuraciones de k=4 y k=6, en relación con su tiempo de transmisión teórico “ideal”, donde no se considera retardos ni congestión de la red. Posteriormente, se realiza una comparación entre las CDF de los FCT para los valores de k=4 y k=6.

Además, se realiza un análisis numérico para cuantificar las diferencias, calculando la media, desviación estándar y máximo de los tiempos de duración. Finalmente se realiza el análisis comparativo de los CDF de los FCT entre configuraciones k=4 y k=6 de fat-tree.

Análisis de la Función de Distribución Acumulativa (CDF) del "FCT" con el CDF "ideal", en la configuración Fat-tree k=4.

Figura 13

CDF de "FCT" e" ideal" con k=4

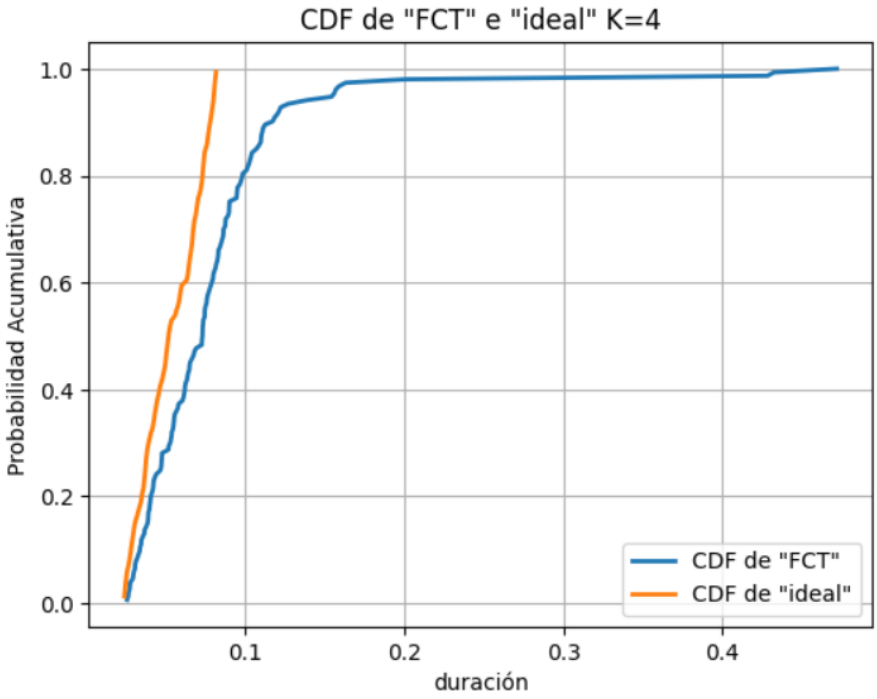


Tabla 1

Análisis numérico para CDF de "FCT" e "ideal" con k=4"

Duración	"Ideal" k=4	"FCT" k=4	Porcentajes de Afectación
Media	0.0516	0.0738	43.0232558 %
Máximo	0.08	0.4721	490.125 %
Desviación estándar	0.0192	0.0278	44.7916667 %

Discusión de resultados (Figura 13 y Tabla 7):

La CDF de "ideal" muestra un crecimiento ligero y continuo a medida que los tiempos de transmisión aumentan. Se observa que el 100% de los tiempos ideales no sobrepasan de los 0.1 segundos de "duración", esto sugiere que la mayoría de las duraciones de tiempo "ideal" tienden a estar cerca de los valores más bajos con

una media de tiempo de 0.0516 segundos, un máximo de 0.08 segundos, y una desviación estándar de 0.0192. La CDF del "FCT", tiene una media de 0.0738 segundos (ligeramente mayor a la duración del ideal) con un máximo de duración de 0.4721 segundos y una desviación estándar de 0.0278.

La probabilidad acumulativa del FCT crece inicialmente de manera pronunciada lo que sugiere tiempos de transmisión bajos, sin embargo, a medida que la duración de los FCT aumenta, la CDF se aplana, lo que sugiere una mayor variabilidad en los tiempos de finalización. Esta variabilidad puede sugerir una congestión en la red. Se puede observar también que cuando la duración es de 0.1 segundos, la probabilidad acumulativa es de aproximadamente 0.80, lo que significa que el 80% de las transmisiones tienen una duración menor o igual a 0.1 segundos.

Análisis de la Función de Distribución Acumulativa (CDF) del "FCT" con el CDF "ideal", en la configuración Fat-tree k=6.

Figura 14

CDF de "FCT" e" ideal" con k=6

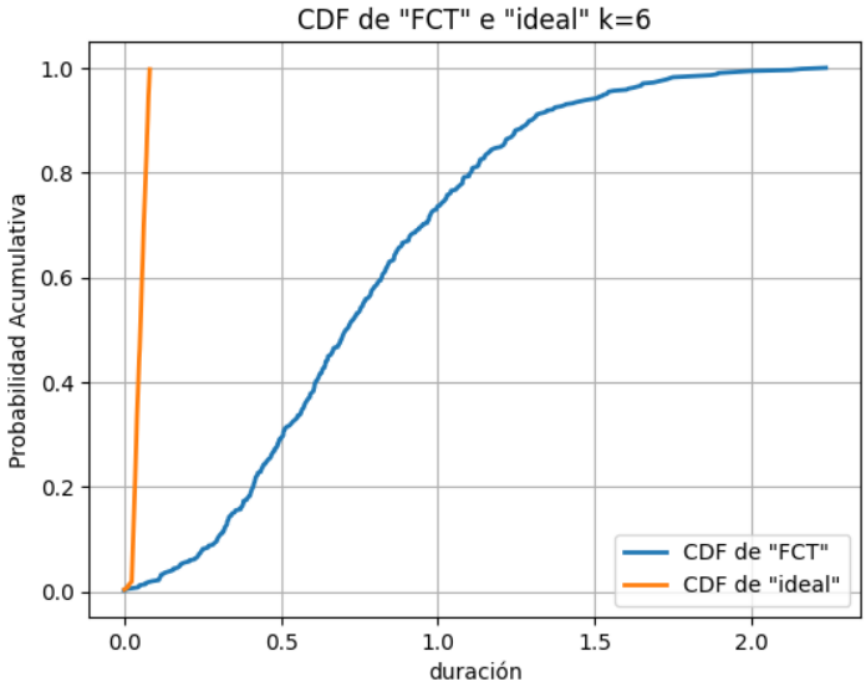


Tabla 2

Análisis numérico para CDF de "FCT" e "ideal" con k=6"

Duración	"Ideal", k=6	"FCT", k=6	Porcentajes de Afectación
Media	0.0552	0.8309	1402.35 %
Máximo	0.0816	3.69548	4424.60 %
Desviación estándar	0.0191	0.4422	2215.18 %

Discusión de resultados (Figura 14 y Tabla 8):

La CDF de "ideal" para el escenario con k=6 muestra una media de duración de 0.0552 segundos, un máximo de 0.0816 segundos y una desviación estándar de 0.0191 segundos. El tiempo máximo de 0.0816 muestra que la mayoría de los flujos tienden a estar en los valores de duración más bajos.

Mientras el CDF del FCT muestra una media de duración de 0.8309 segundos, un máximo de 3.69548 segundos y una desviación estándar de 0.4422 segundos. La desviación estándar de los FCT es mayor que los tiempos ideales, lo que indica una mayor variabilidad y dispersión de los FCT. Observando el gráfico CDF del FCT muestra una curva más extendida hacia la derecha en comparación con los tiempos de transmisión ideales. Esto indica que una proporción significativa de flujos experimenta FCT más largos en comparación con los tiempos ideales. En otras palabras, los flujos experimentan una amplia gama de FCT, desde valores relativamente bajos hasta valores significativamente más altos.

Análisis de la Función de Distribución Acumulativa (CDF) del "FCT" de k=4 con el CDF "FCT" de k=6.

Figura 15

CDF de "FCT de k=4" y "FCT de k=6"

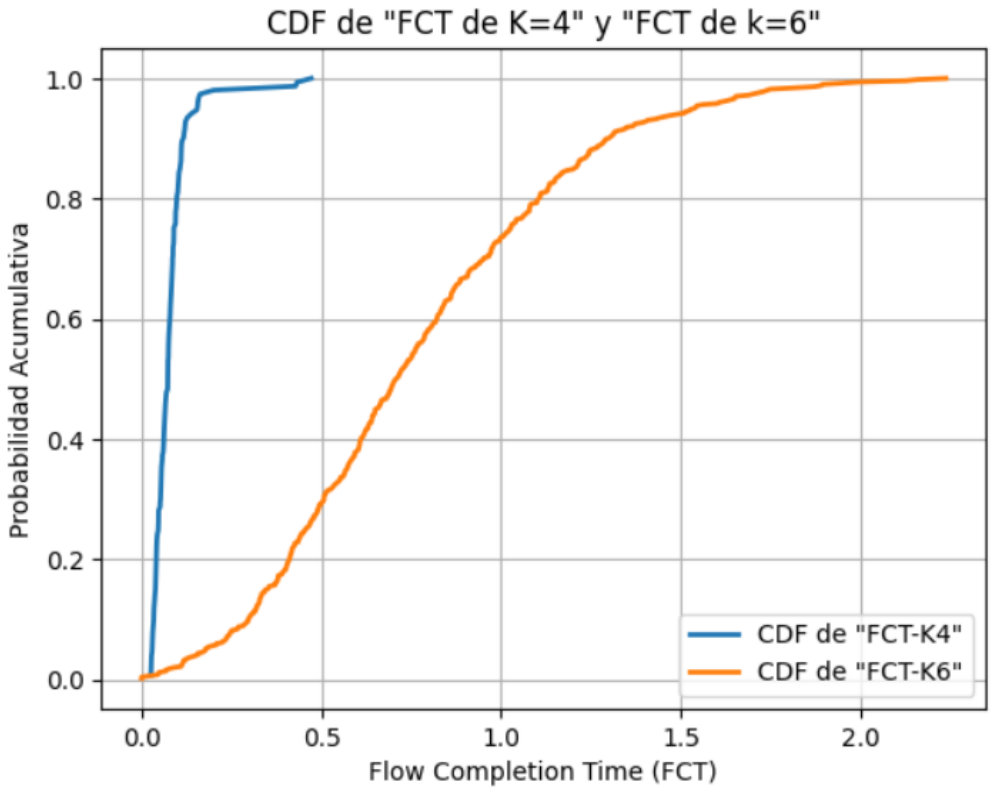


Tabla 3

Análisis numérico para CDF de "FCT de k=4" y "FCT de k=6"

Duración	"FCT", k=4	"FCT", k=6	Porcentajes de Afectación
Media	0.0738	0.8309	1025.88076 %
Máximo	0.4721	3.69548	682.774836 %
Desviación estándar	0.0278	0.4422	1490.64748 %

Discusión de resultados (Figura 15 y Tabla 9):

Al observar los CDF de los FCT de las topologías Fat-tree con configuración de k=4 y k=6, se puede apreciar como varia el FCT con diferentes configuraciones de "k".

Alrededor del 69.48% de los FCT de k=6 están por debajo del segundo de transmisión, mientras el 100% de los flujos de FCT de k=4 se encuentran por debajo de los 0.5 segundos.

La desviación estándar de k=6 es de 0.4422 significativamente mayor que la desviación estándar de k=4 de 0.0278. Este hallazgo sugiere que los tiempos de duración de los FCT con k=6 tienden a ser considerablemente más largos en comparación con los FCT de k=4, lo que podría indicar la presencia de congestión en la red.

6. Conclusiones

La comparación de las funciones de distribución acumulativa (CDF) de los tiempos de transmisión ideales teóricos con los CDF de los tiempos de finalización de flujo (Flow Completion time - FCT) en arquitecturas simuladas de centro de datos (Data Center Network - DCN) basadas en SDN muestra notables diferencias. Los resultados indican que los FCT con k=4 tienden a acercarse más a los tiempos ideales en un 80%, mientras que los FCT con k=6, con una variación estándar considerablemente mayor en comparación con los tiempos ideales, tienden a tener una mayor variabilidad y dispersión en los FCT.

A partir del análisis de los CDF basados en los FCT generados por la simulación de topologías de DCN Fat-tree con configuraciones de k=4 y k=6 bajo la gestión del controlador SDN ONOS, se puede concluir que la métrica de rendimiento FCT se ve afectada a medida que la configuración de escalabilidad del grado “k” es aumentada. Aunque un valor de “k” más alto teóricamente ofrece más rutas y capacidad, sin una gestión adecuada como la que puede proporcionar un controlador como ONOS, los recursos pueden quedar sin explotarse. Este estudio, sugiere la importancia de considerar estrategias de gestión de red para minimizar la variabilidad de los FCT en un entorno de DCN en constante crecimiento. Estas estrategias pueden comprender la implementación de políticas de calidad de servicio y la optimización del uso de los enlaces gestionados por el controlador SDN. Aprovechar las ventajas de un controlador SDN de arquitectura lógicamente centralizada, como ONOS, y su potencial expansión hacia múltiples controladores coordinados para la distribución eficaz de la carga de trabajo puede resultar beneficiosa para optimizar el rendimiento de la red y reducir el FCT.

En cuanto a futuros trabajos, este proyecto se enfocó en la evaluación del FCT del tráfico de tipo este/oeste en una arquitectura DCN en crecimiento basada en SDN en un escenario controlado. Sin embargo, este proyecto se puede expandir a escenarios donde se incorporen tecnologías como VXLAN, que introducen otros

conjuntos de variables en el underlay de la red que pueden tener un impacto directo en el FCT, desencadenando nuevas preguntas.

Referencias

Abdelmoniem, A. M., Bensaou, B., & Abu, A. J. (2017). SICC: SDN-based incast congestion control for data centers. *2017 IEEE International Conference on Communications (ICC)*, 1–6. https://doi.org/10.1109/ICC.2017.7996826

Al-Fares, M., Loukissas, A., & Vahdat, A. (2008). A scalable, commodity data center network architecture. *ACM SIGCOMM Computer Communication Review*, *38*(4), 63–74. https://doi.org/10.1145/1402946.1402967

Bello, S. A., Oyedele, L. O., Akinade, O. O., Bilal, M., Davila Delgado, J. M., Akanbi, L. A., Ajayi, A. O., & Owolabi, H. A. (2021). Cloud computing in construction industry: Use cases, benefits and challenges. *Automation in Construction*, *122*, 103441. https://doi.org/10.1016/j.autcon.2020.103441

Benson, T., Anand, A., Akella, A., & Zhang, M. (2010). Understanding data center traffic characteristics. *ACM SIGCOMM Computer Communication Review*, *40*(1), 92–99. https://doi.org/10.1145/1672308.1672325

Berde, P., Gerola, M., Hart, J., Higuchi, Y., Kobayashi, M., Koide, T., Lantz, B., O'Connor, B., Radoslavov, P., Snow, W., & Parulkar, G. (2014). ONOS. *Proceedings of the Third Workshop on Hot Topics in Software Defined Networking*, 1–6. https://doi.org/10.1145/2620728.2620744

Blial, O., Ben Mamoun, M., & Benaini, R. (2016). An Overview on SDN Architectures with Multiple Controllers. *Journal of Computer Networks and Communications*, *2016*, 1–8. https://doi.org/10.1155/2016/9396525

Chagas, S., Lopes, N., & Portela, I. (2021). Performance Evaluation of Host Scalability in Software Defined Networks. *2021 16th Iberian Conference on Information Systems and Technologies (CISTI)*, 1–6. https://doi.org/10.23919/CISTI52073.2021.9476545

Cornell University. (2014). *https://www.cs.cornell.edu/courses/cs5413/2014fa/lectures/08-fattree.pdf*.

Curtis, A. R., Mogul, J. C., Tourrilhes, J., Yalagandula, P., Sharma, P., & Banerjee, S. (2011). DevoFlow. *ACM SIGCOMM Computer Communication Review*, *41*(4), 254–265. https://doi.org/10.1145/2043164.2018466

da Fonseca, N. L. S., & Boutaba, R. (2015). *Cloud Services, Networking, and Management* (J. Wiley, Ed.).

Daji Wong, Kiam Tian Seow, Chuan Heng Foh, & Kanagavelu, R. (2013). Towards reproducible performance studies of datacenter network architectures using an open-source simulation approach. *2013 IEEE Global Communications Conference (GLOBECOM)*, 1373–1378. https://doi.org/10.1109/GLOCOM.2013.6831265

Estinet tecnologies Inc. (n.d.). Https://Www.Estinet.Com.

Fischer e Silva, R., & Carpenter, P. M. (2015). Exploring interconnect energy savings under east-west traffic pattern of mapreduce clusters. *2015 IEEE 40th Conference on Local Computer Networks (LCN)*, 10–18. https://doi.org/10.1109/LCN.2015.7366278

Greenberg, A., Hamilton, J. R., Jain, N., Kandula, S., Kim, C., Lahiri, P., Maltz, D. A., Patel, P., & Sengupta, S. (2009). VL2. *ACM SIGCOMM Computer Communication Review*, *39*(4), 51–62. https://doi.org/10.1145/1594977.1592576

http://mininet.org/. (n.d.). *http://mininet.org/*.

https://wiki.onosproject.org/display/ONOS/ONOS. (n.d.).

https://www.nsnam.org/. (n.d.). *https://www.nsnam.org/*.

Hu, C., Liu, B., Xing, C., Ding, K., Xu, B., Wei, X., & Zhang, X. (2018). Flow scheduling strategies for minimizing flow completion times in data center networks. *Journal of High Speed Networks*, *24*(1), 63–78. https://doi.org/10.3233/JHS-170581

Hu, S., Wang, X., & Shi, Z. (2021). A Software Defined Network Scheme for Intra Datacenter Network Based on Fat-Tree Topology. *Journal of Physics: Conference Series*, *2025*(1), 012106. https://doi.org/10.1088/1742-6596/2025/1/012106

Jackson, K. L., & Goessling, S. (2018). *Architecting Cloud Computing Solutions* (1st ed.). Packt Publishing, Limited.

Kim, J., Dally, W. J., Scott, S., & Abts, D. (2008). Technology-Driven, Highly-Scalable Dragonfly Topology. *ACM SIGARCH Computer Architecture News*, *36*(3), 77–88. https://doi.org/10.1145/1394608.1382129

Koponen, T., Casado, M., Gude, N., Stribling, J., Poutievski, L., Zhu, M., Ramanathan, R., Iwata, Y., Inoue, H., Hama, T., Hama, T., & Shenker, S. (2019). Onix: A distributed control platform for large-scale production networks. *Proceedings of the 9th*

USENIX Symposium on Operating Systems Design and Implementation, OSDI 2010, 351–364.

Li, H., Lu, H., & Fu, X. (2020). An optimal and dynamic elephant flow scheduling for SDN-based data center networks. *Journal of Intelligent & Fuzzy Systems*, *38*(1), 247–255. https://doi.org/10.3233/JIFS-179399

Liu, Y., Muppala, J. K., Veeraraghavan, M., Lin, D., & Hamdi, M. (2013). *Data Center Network Topologies: Current State-of-the-Art* (pp. 7–14). https://doi.org/10.1007/978-3-319-01949-9_2

Luciano Jerez Chaves, Islene Calciolari García, & Edmundo Roberto Mauro Madeira. (n.d.). *OFSWITCH13*. Http://Www.Lrc.Ic.Unicamp.Br/Ofswitch13/Ofswitch13.Pdf. Retrieved July 22, 2023, from http://www.lrc.ic.unicamp.br/ofswitch13/ofswitch13.pdf

Ma, Y.-W., Chen, Y.-C., & Chen, J.-L. (2017). SDN-enabled network virtualization for industry 4.0 based on IoTs and cloud computing. *2017 19th International Conference on Advanced Communication Technology (ICACT)*, 199–202. https://doi.org/10.23919/ICACT.2017.7890083

Makhlouf, R. (2020). Cloudy transaction costs: a dive into cloud computing economics. *Journal of Cloud Computing*, *9*(1), 1–11. https://doi.org/10.1186/s13677-019-0149-4

Modi, T., & Swain, P. (2019). FlowDCN: Flow Scheduling in Software Defined Data Center Networks. *2019 IEEE International Conference on Electrical, Computer and Communication Technologies (ICECCT)*, 1–5. https://doi.org/10.1109/ICECCT.2019.8869180

Nunes, B. A. A., Mendonca, M., Nguyen, X.-N., Obraczka, K., & Turletti, T. (2014). A Survey of Software-Defined Networking: Past, Present, and Future of Programmable Networks. *IEEE Communications Surveys & Tutorials*, *16*(3), 1617–1634. https://doi.org/10.1109/SURV.2014.012214.00180

Phemius, K., Bouet, M., & Leguay, J. (2014). DISCO: Distributed multi-domain SDN controllers. *2014 IEEE Network Operations and Management Symposium (NOMS)*, 1–4. https://doi.org/10.1109/NOMS.2014.6838330

Poutievski, L., Mashayekhi, O., Ong, J., Singh, A., Tariq, M., Wang, R., Zhang, J., Beauregard, V., Conner, P., Gribble, S., Kapoor, R., Kratzer, S., Li, N., Liu, H.,

Nagaraj, K., Ornstein, J., Sawhney, S., Urata, R., Vicisano, L., … Vahdat, A. (2022). Jupiter evolving. *Proceedings of the ACM SIGCOMM 2022 Conference*, 66–85. https://doi.org/10.1145/3544216.3544265

quora.com. (n.d.). *What-is-a-Clos-network*. Https://Www. Quora.Com/What-Is-a-Clos-Network . Retrieved April 4, 2023, from https://www. quora.com/What-is-a-Clos-network

Sharma, V., & Mishra, R. (2020). A Comprehensive Survey on Data Center Network. *8th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO)*, 222–228.

Srivastava, P., & Khan, R. (2018). A Review Paper on Cloud Computing. *Srivastava, P., & Khan, R. (2018). A Review Paper on Cloud Computing.International Journal of Advanced Research in Computer Science and Software Engineering*, *8*, 17–20.

T.BAH, M., Azzouni, A., Nguyen, M. T., & Pujolle, G. (2019). Topology Discovery Performance Evaluation of OpenDaylight and ONOS Controllers. *2019 22nd Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)*, 285–291. https://doi.org/10.1109/ICIN.2019.8685915

Teo, Z., Birman, K., & Van Renesse, R. (2016). Experience with 3 SDN Controllers in an Enterprise Setting. *2016 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshop (DSN-W)*, 97–104. https://doi.org/10.1109/DSN-W.2016.20

Tootoonchian, A., & Ganjali, Y. (2010). HyperFlow: A distributed control plane for OpenFlow. *2010 Internet Network Management Workshop / Workshop on Research on Enterprise Networking, INM/WREN 2010*.

Vesović, M., Smiljanić, A., & Kostić, D. (2022). Fast and scalable routing protocols for data center networks. *Digital Communications and Networks*. https://doi.org/10.1016/j.dcan.2022.06.010

Wendell Odom. (2020). *CCNA 200-301 Official Cert Guide* (Vol. 2).

Yahyaoui, H., Aidi, S., & Zhani, M. F. (2020). On Using Flow Classification to Optimize Traffic Routing in SDN Networks. *2020 IEEE 17th Annual Consumer Communications & Networking Conference (CCNC)*, 1–6. https://doi.org/10.1109/CCNC46108.2020.9045216

Yao, F., Wu, J., Venkataramani, G., & Subramaniam, S. (2014). A comparative analysis of data center network architectures. *2014 IEEE International Conference on Communications (ICC)*, 3106–3111. https://doi.org/10.1109/ICC.2014.6883798

Zaher, M., Alawadi, A. H., & Molnar, S. (2020). Class-based Flow Scheduling Framework in SDN-based Data Center Networks. *2020 International Conference on Computing, Electronics & Communications Engineering (ICCECE)*, 51–56. https://doi.org/10.1109/iCCECE49321.2020.9231052

Zang, W., Jin, Z., & Lan, J. (2017). An SDN based fast rerouting mechanism for elephant flows in DCN. *2017 8th IEEE International Conference on Software Engineering and Service Science (ICSESS)*, 363–366. https://doi.org/10.1109/ICSESS.2017.8342933

Zhang, T., Zhang, Q., Lei, Y., Zou, S., Huang, J., & Li, F. (2022). Load balancing with traffic isolation in data center networks. *Future Generation Computer Systems*, *127*, 126–141. https://doi.org/10.1016/j.future.2021.09.002