DESARROLLO DE UN SISTEMA ADAPTATIVO PARA EL BALANCEO DE TRÁFICO EN CENTROS DE DATOS MEDIANTE CONTROLADORES SDN Y METODOLOGÍAS KDN

JAVIER GONZALO ORTIZ ROJAS

TESIS DOCTORAL PRESENTADA PARA OPTAR AL TÍTULO DE DOCTOR EN INGENIERÍA

UNIVERSIDAD PONTIFICIA BOLIVARIANA
ESCUELA DE INGENIERÍAS
DOCTORADO EN INGENIERÍA
MEDELLÍN
2026

DESARROLLO DE UN SISTEMA ADAPTATIVO PARA EL BALANCEO DE TRÁFICO EN CENTROS DE DATOS MEDIANTE CONTROLADORES SDN Y METODOLOGÍAS KDN

JAVIER GONZALO ORTIZ ROJAS

TESIS DOCTORAL PRESENTADA PARA OPTAR AL TÍTULO DE DOCTOR EN INGENIERÍA

DIRECTOR
PH.D. JORGE MARIO LONDOÑO PELÁEZ

UNIVERSIDAD PONTIFICIA BOLIVARIANA
ESCUELA DE INGENIERÍAS
DOCTORADO EN INGENIERÍA
MEDELLÍN
2026

DEDICATORIA

A Dios. A la virgen María, A San José y a todos los santos y ángeles del cielo. A Johana, Jesus y Geovanna, mi familia. A Teresa y Gonzalo, mis padres.

AGRADECIMIENTOS

A Dios, por brindarme la paz cuando las circunstancias eran adversas, la claridad en los momentos de duda y el vigor cuando me sentí débil.

A la virgen María, a San José y a todos los ángeles del cielo por su continua intercesión.

A mi familia por su amor incondicional, por tolerar mis ausencias y por estar siempre presentes.

Agradezco profundamente al Doctor Jorge Mario Londoño, por su amistad, por su guía académica, sus observaciones precisas y por desafiarme a ir siempre un paso más allá.

A mis colegas y profesores, por cada conversación técnica, cada análisis crítico y por compartir generosamente su conocimiento.

A la Universidad Pontificia Bolivariana por su excelente programa doctoral y la Universidad Politécnica Salesiana, por el apoyo económico para mis estudios doctorales.

A mis maravillosos padres por todo lo que recibí de ellos.

CONTENIDO

LISTA DE TABLAS

LISTA DE FIGURAS

SIGLAS, ACRÓNIMOS Y ABREVIATURAS

DQN	Deep Q-Learning
DRILL	Distributed Randomized Load Balancing
ECMP	Equal-Cost Multi-Path
FCT	Flow completion time
IA	Artificial Intelligence
KDN	Knowledge-Defined Networking
PER	Prioritized Experience Replay
SDN	Software-Defined Networking

RESUMEN GENERAL DE TRABAJO DE GRADO

TITULO:	DESARROLLO DE UN SISTEMA ADAPTATIVO PARA EL BALANCEO DE TRÁFICO EN CENTROS DE DATOS MEDIANTE CONTROLADORES SDN Y METODOLOGÍAS KDN
AUTOR(ES):	Javier Gonzalo Ortiz Rojas
PROGRAMA:	Doctorado en Ingeniería
DIRECTOR(A):	Jorge Mario Londoño Peláez

RESUMEN

El aumento del tráfio interno en centros de datos, que incluye flujos pequeños (*mice*) y flujos grandes (*elephants*), crea situaciones dinámicas de congestión que mecanismos tradicionales de balanceo de tráfico, como ECMP y DRILL, no gestionan de una manera óptima. El presente trabajo está basado en la metodología de redes definidas por conocimiento (KDN), en la cual sugiere que los controladores de redes definidas por software (SDN) pueden alcanzar capacidades de observación, aprendizaje y decisión adaptativa. Se desarrolló un sistema de control inteligente basado en Deep Q-Learning (DQN) en la cual un agente previamente entrenado, elige las rutas mínimas en una topología FatTree utilizando información del estado de los enlaces. Con esto se logra un mejor balanceo del tráfico y tiempo de finalización de flujos (FCT). Su rendimiento se evalúa con diferentes niveles de tráfico que incluyen procesos ON/OFF con flujos *mice* y *elephants*, y comparándolo con ECMP y DRILL. Los resultados muestran que el agente DQN se adapta de manera rápida y eficiente a las fluctuaciones del tráfico, alcanzando valores de FCT menores y un balanceo más uniforme en comparación con ECMP y DRILL. En el escenario de mayor carga evaluado, el modelo DQN redujo el cociente FCTexp/FCTteo en aproximadamente 52% frente a ECMP y 10% frente a DRILL, al pasar de valores cercanos a 7.7 y 4.1, respectivamente, a un valor aproximado de 3.7 con DQN. Estos resultados evidencian que un controlador SDN con capacidades de aprendizaje puede mitigar la congestión y mejorar el uso de los recursos de red bajo condiciones dinámicas de tráfico.

PALABRAS CLAVE:

SDN, KDN, DQN, Balanceo de tráfico, FCT, FatTree

V° B° DIRECTOR DE TRABAJO DE GRADO

GENERAL SUMMARY OF WORK OF GRADE

TITLE:	DEVELOPMENT OF AN ADAPTIVE SYSTEM FOR TRAFFIC BALANCING IN DATA CENTERS USING SDN CONTROLLERS AND KDN METHODOLOGIES
AUTHOR(S):	Javier Gonzalo Ortiz Rojas
FACULTY:	Doctorado en Ingeniería
DIRECTOR:	Jorge Mario Londoño Peláez

ABSTRACT

The increase in internal traffic within data centers, which includes small flows (mice) and large flows (elephants), generates dynamic congestion scenarios that traditional traffic balancing mechanisms such as ECMP and DRILL do not manage optimally. This work is based on the Knowledge-Defined Networking (KDN) methodology, which suggests that Software-Defined Networking (SDN) controllers can achieve observation, learning, and adaptive decision-making capabilities. An intelligent control system based on Deep Q-Learning (DQN) was developed, in which a previously trained agent selects minimum-path routes in a FatTree topology using link-state information. This approach achieves better traffic distribution and flow completion time (FCT). The system's performance is evaluated under different traffic levels, including ON/OFF processes with mice and elephant flows, and compared against ECMP and DRILL. The results show that the DQN agent adapts quickly and efficiently to traffic fluctuations, achieving lower FCT values and a more uniform load balancing compared with ECMP and DRILL. In the highest-load scenario evaluated, the DQN model reduced the FCTexp/FCTtheo ratio by approximately 52% compared with ECMP and 10% compared with DRILL, decreasing from nearly 7.7 and 4.1, respectively, to approximately 3.7 with DQN. These results demonstrate that an SDN controller with learning capabilities can mitigate congestion and improve network resource utilization under dynamic traffic conditions.

KEYWORDS:

SDN, KDN, DQN, Traffic Load Balancing, FCT, FatTree

V° B° DIRECTOR OF GRADUATE WORK

INTRODUCCIÓN

Debido al acelerado crecimiento de los servicios digitales en la nube [1], los centros de datos han tenido que transformar significativamente su arquitectura y operación. Ciertamente este suceso ha incrementado la demanda por más recursos de red, y particularmente en el plano interno de las infraestructuras que los soportan. A diferencia de las arquitecturas convencionales, en donde el tráfico cliente-servidor (norte-sur) constituía el mayor porcentaje del tráfico de red, los centros de datos actuales presentan un predominio del tráfico este-oeste, es decir, el que se refiere a la comunicación entre servidores, máquinas virtuales y racks dentro del propio centro de datos.

Esta nueva distribución del tráfico impone importantes desafíos en las arquitecturas de red, que deben ser capaces de ofrecer alta disponibilidad, escalabilidad, tolerancia a fallos y, sobre todo, eficiencia en la gestión del tráfico interno para evitar cuellos de botella. Debido a esto, se desarrollaron varias topologías de red (FatTree [2], DCell [3] o Jellyfish [4]), en las cuales se implementan múltiples rutas alternativas entre los nodos. Sin embargo, contar únicamente con rutas no es garantía de que los recursos se usen de forma eficiente; por ello, se necesitan mecanismos inteligentes de balanceo bajo condiciones dinámicas del tráfico en la red.

Balancear el tráfico se convierte en una necesidad esencial para mejorar el rendimiento de la red en los centros de datos. Si bien técnicas tradicionales como ECMP [5], presentan soluciones sencillas y escalables, aún evidencian limitaciones ante la presencia de flujos de tráfico irregulares o de colisiones propias de esta técnica. Debido a esto, es necesario experimentar con métodos más adaptativos que se adecúen a las condiciones variantes de carga y congestión en tiempo real.

De esta necesidad nacen las redes definidas por software [6] (SDN) las cuales revolucionaron la forma de gestionar las redes, separando el plano de control del plano de datos. Esto permite que un controlador centralizado tenga una visión global del estado de la red y pueda programar dinámicamente el comportamiento de los switches. Esta habilidad resulta particularmente útil en centros de datos modernos, donde las condiciones de carga varían rápidamente y de forma muy heterogénea.

De forma complementaria, las metodologías de redes definidas por conocimiento [7] (KDN) extienden el modelo SDN incorporando técnicas de aprendizaje automático para analizar patrones de tráfico, identificar comportamientos emergentes y tomar decisiones óptimas basadas en datos históricos y en telemetría en tiempo real. En este sentido, la red ya no es solo programable,

sino que evoluciona hacia una infraestructura con capacidades autónomas de análisis, adaptación y ajuste de sus decisiones operativas.

En este contexto, los algoritmos de aprendizaje por refuerzo se convierten en una herramienta adecuada para abordar el desafío del balanceo de tráfico. Al interactuar con el entorno de red, el agente aprende políticas que minimizan la congestión y distribuyen mejor los flujos que los métodos tradicionales estáticos o basados en selección local. La combinación de SDN (como plano de control programable) y KDN (como marco de aprendizaje) permite la experimentación con controladores adaptativos que pueden reaccionar y adaptarse a los cambios en el tráfico en tiempo real.

I. PLANTEAMIENTO DEL PROBLEMA

El problema principal tratado en este trabajo es la ineficacia de los mecanismos actuales para el balanceo de tráfico, que no se adaptan óptimamente a las condiciones de tráfico dinámicas existentes en los centros de datos de hoy. Aunque en las topologías se implementan múltiples rutas entre nodos, técnicas como ECMP gestionan las rutas de forma agnóstica a las condiciones reales de la red y mediante algoritmos como *round-robin* [8]. Como consecuencia, se pueden generar saturaciones en unos enlaces y una escasa utilización en otros. Esta situación se vuelve aún más compleja cuando el tráfico de la red está conformado por flujos de gran tamaño, que podrían monopolizar la capacidad de los enlaces de la ruta asignada.

Ante esta realidad han surgido otras técnicas como Flowlets [9], ECMP, o Hedera [10], que pretenden mitigar la congestión dividiendo los flujos en subflujos o haciendo ajustes de las rutas de forma dinámica. No obstante, presentan limitaciones importantes que limitan su aplicación en entornos reales de centros de datos. En el caso de Flowlets, su rendimiento depende de la detección precisa de intervalos de inactividad de los flujos, lo cual resulta muy complejo en tráfico altamente variable. Con DRILL [11], aunque se reduce la congestión eligiendo aleatoriamente entre dos rutas menos cargadas, no guarda memoria del estado histórico del tráfico, por lo tanto, sus decisiones son locales y pueden causar subutilización de los enlaces u oscilaciones [12]. Hedera, por su parte, necesita supervisar el tráfico de forma centralizada, dependiendo del hardware que sea compatible con las mediciones en tiempo real, incrementando la complejidad para su implementación y elevando su costo operativo. Estas limitaciones evidencian la necesidad de investigar métodos más flexibles e inteligentes que aprendan del comportamiento del tráfico y optimicen la asignación de rutas en tiempo real.

El principal reto se encuentra en el desarrollo de un sistema que se adapte al estado actual de la red, con el fin de optimizar la distribución del tráfico entre los enlaces disponibles, reducir la congestión y mejorar métricas importantes como el tiempo de finalización del flujo (*Flow Completion Time* - FCT). A diferencia de los métodos tradicionales, este sistema tiene que analizar en tiempo real el uso de los enlaces y el comportamiento de los flujos, sin depender de situaciones especiales como intervalos fijos, decisiones al azar o uso de hardware especializado. Este análisis le permitirá adquirir conocimiento de forma continua y tomar decisiones autónomas sobre las rutas más apropiadas. De esta forma, se pretende superar las restricciones de las actuales metodologías

mediante un enfoque basado en inteligencia artificial, que ajuste las rutas proactivamente para lograr un balanceo del tráfico más equilibrado y estable en los centros de datos.

Esta investigación parte de la hipótesis de que es posible entrenar a un agente inteligente por medio del desarrollo de un modelo de aprendizaje por refuerzo, y que tendrá la capacidad de aprender con las interacciones que tenga con la red para la toma de decisiones de balanceo más eficientes que las estáticas o heurísticas convencionales. Esta perspectiva nace de los avances de las metodologías KDN que combinan aprendizaje automático, analítica y un control SDN para equipar a la red con habilidades cognitivas y adaptativas. Un agente basado en aprendizaje por refuerzo en una arquitectura KDN puede aprender las dinámicas del tráfico y modificar las rutas con autonomía según el estado actual de la red, lo que permite optimizar la estabilidad y eficiencia del balanceo del tráfico sin requerir de un hardware especializado. Esto contrasta con propuestas como Flowlets, DRILL o Hedera que dependen de reglas fijas, decisiones aleatorias o monitoreo centralizado. De esta manera, se espera superar las restricciones de adaptabilidad y respuesta que exhiben los métodos actuales en ambientes de centros de datos.

II. JUSTIFICACIÓN

La importancia de esta investigación se basa en la contribución para la mejora del rendimiento de los centros de datos en situaciones de alta demanda. Aunque las SDN ha facilitado la gestión centralizada mediante la separación del plano de control del plano de datos, su capacidad para el análisis y la toma de decisiones mediante los controladores aún es limitada al no integrar elementos de inteligencia artificial.

Dentro de este panorama, las KDN se catalogan en una evolución de las SDN, debido a que incorporan una capa de conocimiento que permite observar el estado de la red, procesar la información operativa, aprender patrones de comportamiento y transformar dicho aprendizaje en decisiones de control. En el presente trabajo, la propuesta no se plantea en la forma de una aplicación aislada de inteligencia artificial, sino más bien como un sistema de control adaptativo conforme al ciclo funcional de las KDN: observación del estado de los enlaces y flujos, generación de conocimiento mediante aprendizaje por refuerzo, selección dinámica de rutas y ajuste de la dinámica del tráfico en la topología FatTree. De este modo, el agente DQN actúa como mecanismo de decisión al interior de una arquitectura orientada al conocimiento, mediante el uso de información del entorno para reducir la congestión y distribuir en una forma más equilibrada el uso de los enlaces. Con esta integración, se aplica el principio fundamental de las KDN: convertir los datos del estado de la red en decisiones adaptativas que favorezcan la eficiencia y autonomía del balanceo de tráfico.

No obstante, la aplicación de este enfoque en el marco particular del balanceo de tráfico en topologías FatTree, en condiciones de tráfico realista con flujos de tipo *mice* y *elephant* y en situaciones de congestión, se mantiene como un área escasamente explorada. Aunque los algoritmos de aprendizaje por refuerzo han demostrado resultados favorables en otros campos de las redes, como la calidad de servicio (QoS) o la asignación de recursos, su implementación en entornos controlados que permitan hacer comparaciones directas con otros mecanismos ya establecidos, como ECMP o DRILL, todavía requiere de validación científica.

Este trabajo tiene como objetivo el profundizar en esa necesidad, desarrollando y entrenando un agente basado en aprendizaje por refuerzo, en un ambiente simulado que replica las condiciones operativas de un centro de datos contemporáneo, con el fin de evaluar su capacidad para mejorar la eficiencia, estabilidad y adaptabilidad del balanceo de tráfico frente a los métodos tradicionales.

En relación con el alcance y contribución en comparación con el estado del arte, el aporte principal de esta investigación radica en el desarrollo y evaluación de un sistema adaptativo de balanceo de tráfico en centros de datos, situado en el enfoque KDN, e integrando un entorno de simulación propio, una topología FatTree K=4, un agente DQN y procedimientos de evaluación respecto a ECMP y DRILL en condiciones homogéneas. Mientras que trabajos previos que evalúan algoritmos de inteligencia artificial, heurísticas o mecanismos de optimización en topologías, métricas y plataformas heterogéneas, el presente trabajo formula el problema de selección de rutas a la manera de un proceso de decisión secuencial, define un espacio de estados a partir del uso de enlaces, demanda acumulada y presencia de flujos *elephant*, y diseña una función de recompensa centrada en penalizar congestión y desbalance. Igualmente, la investigación aporta una evaluación comparativa reproducible por medio de las mismas trazas de tráfico, escenarios de carga y métricas de desempeño, teniendo en cuenta FCT, uso de enlaces, balanceo de carga e intervalos de confianza. Con esto, el trabajo aporta a cerrar la brecha identificada en la literatura en relación con la evaluación de DQN para balanceo dinámico de tráfico en topologías FatTree en condiciones de tráfico heterogéneo *mice*/*elephant*.

III. OBJETIVOS

A. Objetivo general

Desarrollar un sistema de control adaptativo fundamentado en metodologías de redes definidas por conocimiento (KDN) para la optimización del balanceo de tráfico en centros de datos, buscando mejorar las condiciones del balanceo comparado con los métodos actuales.

B. Objetivos específicos

- Desarrollar un modelo de aprendizaje automático que permita el balanceo de tráfico adaptativo basado en las metodologías KDN, para la asignación de rutas de flujos entrantes.
- Evaluar el rendimiento del sistema de control adaptativo en ambientes de tráfico variable con presencia de congestión y su influencia en la mejora de los parámetros de red, como el balanceo de los flujos y el FCT.
- Comparar el desempeño de la solución desarrollada contra otros mecanismos tradicionales de balanceo de tráfico, analizando su eficiencia en redes de centros de datos.

IV. HIPÓTESIS

El uso de algoritmos de aprendizaje automático en el contexto de redes definidas por conocimiento (KDN) permitirá que un controlador SDN se adapte de forma rápida y eficiente a las fluctuaciones del tráfico, aplicando un balanceo adaptativo que prevenga en tiempo real la congestión mediante decisiones sobre las rutas de tráfico.

V. MARCO TEÓRICO Y ESTADO DEL ARTE

A. Centros de datos de alta demanda

Los centros de datos modernos ya no son únicamente espacios para almacenar información, sino que hoy son infraestructuras activas que procesan y distribuyen grandes volúmenes de datos sin precedentes. Esto va de la mano con el crecimiento de los servicios en la nube y la necesidad de aplicaciones en tiempo real. Una característica de estos sitios es que normalmente el tráfico interno (este-oeste) supera al tráfico de entrada y salida (norte-sur). Esto justifica la necesidad de desarrollar topologías más eficientes y mecanismos de balanceo de tráfico que permitan la distribución equitativa de los flujos y así minimizar las condiciones de congestión. Por ende, se vuelve esencial que las redes de los centros de datos se diseñen enfocadas en características como: redundancia, escalabilidad y flexibilidad. Para ello se necesita no solo de un diseño topológico apropiado, sino también de la integración de funcionalidades de gestión inteligente para el tráfico y que considere variaciones de la demanda y de los patrones de utilización en tiempo real.

B. Arquitecturas y topologías de red para centros de datos

Dentro de los centros de datos modernos, la infraestructura se distribuye normalmente en servidores físicos agrupados en racks, conectados entre sí mediante switches de acceso o Top-of-Rack (ToR), y niveles superiores de agregación, distribución o núcleo, dependiendo del diseño de la red [2]. Cada servidor puede concentrar múltiples procesadores o procesadores multinúcleo, memoria, almacenamiento local y varias interfaces de red, en los que se ejecutan máquinas virtuales, contenedores o servicios distribuidos [13]. Esta estructura resulta en una alta densidad computacional y genera patrones intensivos de comunicación este-oeste entre servidores, debido especialmente en aplicaciones distribuidas, almacenamiento replicado, procesamiento paralelo e inteligencia artificial [14]. Por ello, la arquitectura de red debe ofrecer múltiples rutas, baja latencia, alta capacidad de bisección, tolerancia a fallos y mecanismos eficientes de balanceo de tráfico [15].

La topología de red impacta en gran medida en las posibilidades del balanceo tráfico. Algunas de las más importantes son:

FatTree [2]: diseñada para el alto rendimiento y escalabilidad en centros de datos, se compone de una estructura jerárquica, con facilidades para la interconexión eficiente entre los servidores y los dispositivos de la red. En la Fig. 1 La conforma switches distribuidos en múltiples niveles, permitiendo que el tráfico se distribuya eficientemente, minimizando la congestión. En su diseño básico, se emplean tres niveles de switches: núcleo (*core*), distribución (*aggregation*) y

acceso (*edge*). Este diseño brinda las facilidades para que cada switch en el nivel inferior se interconecte con múltiples switches en el nivel intermedio, los cuales se conectan a los del nivel superior. De esta manera se crean múltiples rutas redundantes entre todos los nodos, facilitando la distribución de flujos a través de múltiples rutas de igual costo. Esta configuración permite la escalabilidad eficiente al momento de incorporar nuevos dispositivos o servidores sin afectar el rendimiento de la red. Además, el diseño redundante de la red contribuye a mejorar su resiliencia, garantizando que la falla de un componente de la red no impacte significativamente en la conectividad de toda la red. En definitiva, la topología FatTree es una solución eficiente diseñada como una arquitectura de red de alto rendimiento muy utilizada por su simetría y escalabilidad.

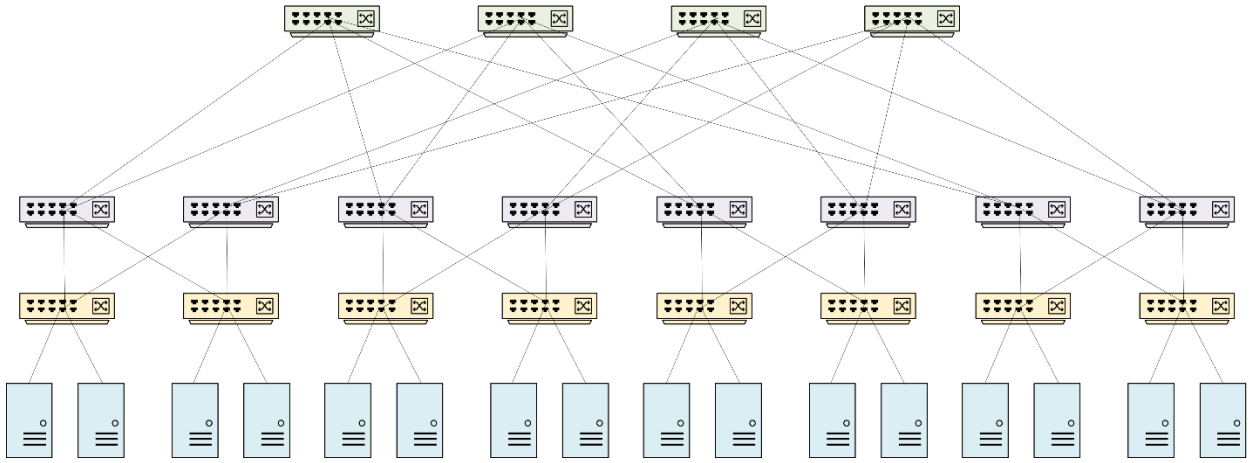

Fig. 1 Topología FatTree

DCell [3] y BCube [16]: constituyen arquitecturas de red diseñadas para mejorar la escalabilidad y la eficiencia en centros de datos. Ambas propuestas abordan la necesidad de interconectar servidores con el objetivo de optimizar los recursos disponibles y reducir los cuellos de botella en el tráfico de datos. DCell se estructura jerárquicamente, por lo que un nodo se puede conectar a múltiples nodos, brindando una alta redundancia y mejor uso del ancho de banda. Es altamente flexible cuando se expande el sistema debido a que no requiere reconfiguraciones de gran relevancia. Por otro lado, BCube (Fig. 2) interconecta sus servidores mediante una red de malla total (*full mesh*), en el cual cada servidor se conecta con múltiples servidores de varios niveles. Esta distribución permite eficiencia en la comunicación reduciendo la latencia con una alta tolerancia a fallos. Las dos arquitecturas ofrecen beneficios de rendimiento y escalabilidad, sin embargo, y en comparación con la topología FatTree, los diseños evidencian una complejidad: la dependencia de los servidores para el balanceo del tráfico, lo cual dificulta su despliegue.

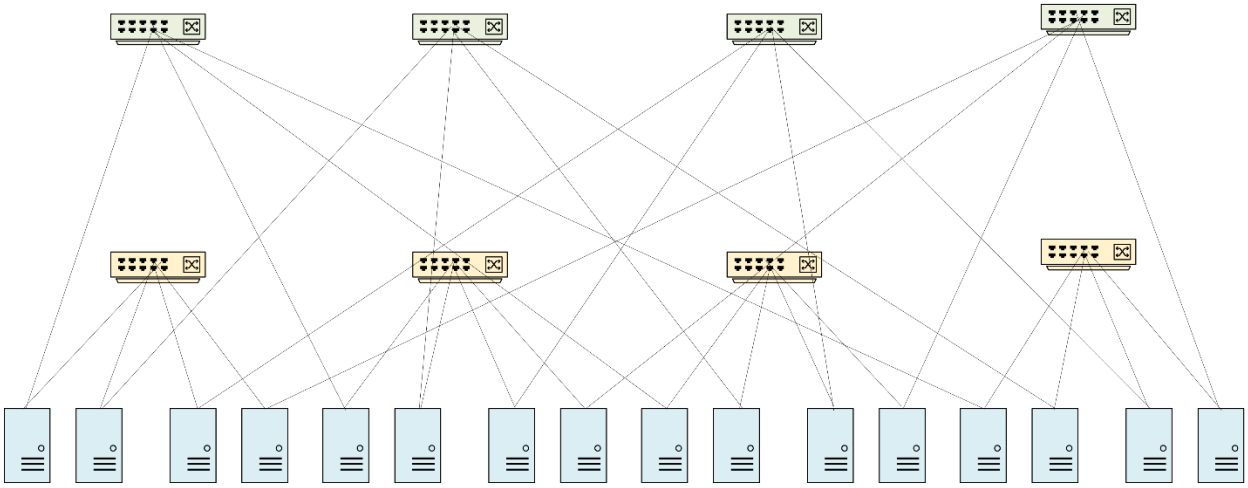

Fig. 2 Topología BCube

Jellyfish [4]: Diseñada como una topología aleatoria para la optimización de la conectividad y la flexibilidad de la red, interconecta los switches de forma pseudoaleatoria implementando un número fijo de enlaces por nodo (Fig. 3). Brinda facilidades para adicionar más switches a la red sin incurrir en rediseños, lo que la vuelve altamente flexible y escalable. Entre sus desventajas se encuentran: difícil gestión y mantenimiento de la red al no haber una jerarquía, complejidad en la selección de rutas al requerir algoritmos avanzados para el cálculo de la ruta óptima y menor compatibilidad con protocolos tradicionales dado que muchos de ellos operan sobre topologías simétricas.

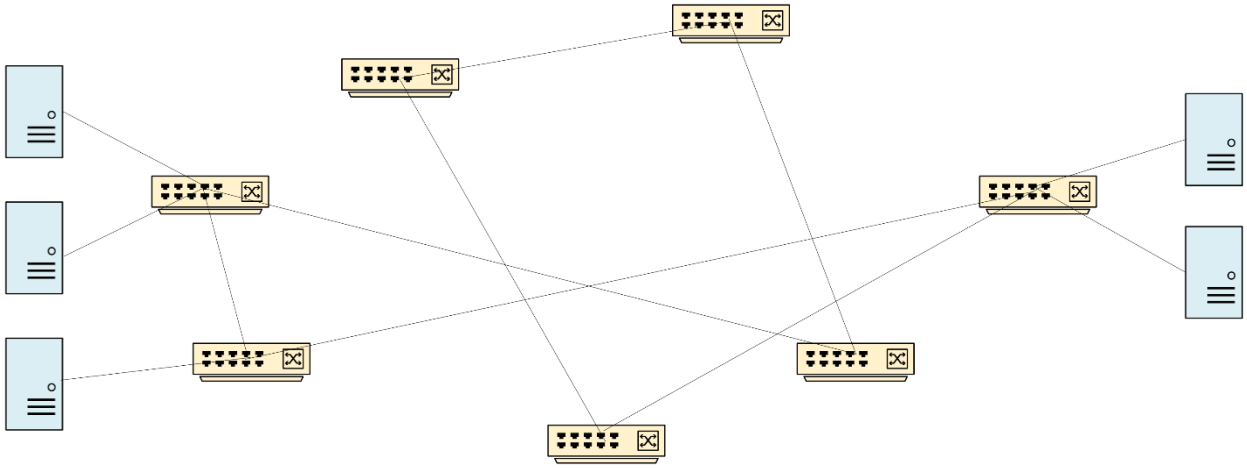

Fig. 3 Topología Jellyfish

En el presente trabajo se utiliza la topología FatTree por su buen equilibrio entre complejidad y diversidad de rutas, lo cual que brinda facilidades en la evaluación de varias estrategias de balanceo de tráfico en ambientes controlados.

C. Perfiles de tráfico en centros de datos reales

El tráfico en centros de datos modernos muestra patrones heterogéneos y altamente variables. En contraste con el tráfico tradicional cliente-servidor, las aplicaciones distribuidas producen una proporción importante de tráfico este-oeste entre servidores [17]. En este sentido se identifican varios patrones de relevancia para el balanceo: los flujos *mice*, caracterizados por tamaños pequeños y alta sensibilidad a la latencia; los flujos *elephant*, de mayor tamaño y duración, con capacidades de ocupación de los enlaces durante periodos prolongados; las ráfagas de tráfico, relacionadas con las fases de alta concentración temporal de flujos; y fenómenos como incast, en los que múltiples servidores envían datos de forma simultánea a un mismo receptor, produciendo congestión localizada y aumento de colas [18] [19]. Análisis recientes se mantienen en la utilización de estos patrones para caracterizar, clasificar y optimizar el tráfico en centros de datos, de manera especial con técnicas de aprendizaje automático, telemetría y balanceo adaptativo [20] [21]. Estos patrones evidencian la necesidad de mecanismos de balanceo adaptativos, con capacidades de reacción ante el estado de la red y no únicamente a reglas estáticas de selección de rutas.

Basándose en estos patrones, la presente investigación adopta un generador de tráfico que incluye llegadas Poisson con ciclos ON/OFF, y una mezcla *mice*/e*lephant*, con el fin de reproducir condiciones de carga variable y congestión emergente en un entorno controlado.

D. Técnicas de balanceo de tráfico

El balanceo de tráfico intenta distribuir el tráfico de manera equitativa entre todos los enlaces disponibles para que no se produzcan rutas con enlaces saturados o subutilizados. Entre las técnicas más comunes se incluyen:

Equal-Cost Multi-Path [5] (ECMP): Posibilita el uso de múltiples rutas de igual costo para la distribución de los flujos, optimizando el ancho de banda disponible y mejorando la resiliencia de la red. Mediante el uso de algoritmos (*round-robin*), ECMP determina la ruta que seguirá cada uno de los flujos, para una distribución uniforme del tráfico. Es utilizada comúnmente en redes a gran escala, como las utilizadas por los centros de datos y proveedores de servicios de Internet, en donde el rendimiento y la redundancia son factores críticos. La principal ventaja de este mecanismo es su simplicidad y escalabilidad, pero puede generar un desbalance importante dentro de escenarios con alto flujo o con patrones de colisión.

Flowlets [9]: Divide los flujos en flujos más pequeños o flowlets, identificando pausas naturales que ocurren entre ráfagas de paquetes. Tratando cada flowlet como una unidad separada, esta técnica permite un balanceo más preciso sin causar un desorden significativo de paquetes. Es especialmente eficaz en redes con tráfico intermitente, dado que puede redistribuir subflujos hacia rutas menos congestionadas sin fragmentar los paquetes de manera forzada. Una de sus principales desventajas es que, al depender de las pausas naturales en el tráfico, si éste es continuo o altamente sostenido (como en flujos *elephant* sin pausas), no se detectan flowlets y por lo tanto el algoritmo no puede redistribuir el tráfico.

Hedera [10]: Lo conforma un sistema centralizado que monitorea el estado de la red y asigna rutas óptimas a los flujos *elephant*. Utiliza un enfoque reactivo, es decir que, basado en las estadísticas de congestión que recopila de la red, decide las mejores rutas en tiempo de ejecución. Aunque ofrece una mejora significativa en el uso de enlaces, su necesidad de un control centralizado y el permanente monitoreo con el requisito de un hardware especializado puede limitar la complejidad para su implementación.

Presto [22]: Desarrollado por Facebook, Presto fragmenta flujos en miniflujos que constituyen unidades más pequeñas que se pueden balanceadas de forma dinámica por múltiples rutas. A diferencia de Flowlets, Presto requiere un soporte explícito por parte del host para ensamblar y etiquetar los miniflujos, lo que implica una mayor complejidad para su implementación, aunque permite un control detallado sobre la distribución del tráfico. Su eficiencia lo hace ideal para entornos con cargas altamente dinámicas.

Valiant Load Balancing [23] (VLB): Se centra en equilibrar la carga de trabajo entre múltiples nodos o servidores. Utiliza una estrategia estocástica donde cada flujo se redirige a un nodo intermedio de manera aleatoria antes de llegar a su destino final. Esta técnica garantiza una distribución más equitativa de los flujos a través de la red, sin embargo, puede inducir a rutas más largas y a un reordenamiento de paquetes. Su simplicidad algorítmica contrasta con su costo en términos de latencia y eficiencia en las rutas.

Distributed Randomized Load Balancing [11] (DRILL): Es una técnica basada en decisiones por paquete que permite la selección dinámica de la ruta menos congestionada entre múltiples opciones. A diferencia de ECMP, DRILL hace un monitoreo del uso reciente de los enlaces para evitar que varios flujos se concentren en la misma ruta. Utiliza el principio conocido como *power-of-two choices* [24], que consiste en seleccionar de dos rutas al azar, la de menor uso

en ese momento, lo cual disminuye la probabilidad de congestión sin necesidad de un análisis exhaustivo de todas las opciones posibles. Aunque su precisión es mayor, puede provocar reordenamiento de paquetes y un mayor uso computacional, especialmente si se implementa a nivel de paquete.

E. Redes definidas por software (SDN)

Software Defined Networking [6] *(SDN*) es un paradigma que promueve un cambio fundamental en la arquitectura de las redes tradicionales mediante la separación del plano de control del plano de datos. En las redes convencionales cada dispositivo (switch o router) toma las decisiones de selección de rutas localmente, mientras que en las SDN se centralizan estas decisiones en un controlador lógico que posee una visión global del estado de la red. Esta separación facilita la gestión de la red de una manera mucho más flexible, dinámica y programable.

Entre las principales ventajas de las SDN se encuentran:

- Programabilidad: Es posible modificar el comportamiento de la red sin requerir modificaciones en el hardware. Esto facilita el despliegue oportuno de nuevos servicios y protocolos.
- Centralización: El controlador SDN tiene la capacidad de recopilar información en tiempo real sobre el estado de los enlaces, el tráfico y los dispositivos. Esta funcionalidad permite una gestión más eficiente con respuestas proactivas ante eventos o fallos. Aunque la arquitectura se basa en un plano de control centralizado, se puede extender a modelos jerárquicos o distribuidos, para mantener la escalabilidad y resiliencia por medio de múltiples controladores cooperando entre sí.
- Automatización: Favorece la implementación de políticas para la automatización de la seguridad, el tráfico, la calidad de servicio y el balanceo de tráfico.
- Experimentación: Al separar el hardware, las SDN facilitan las pruebas de nuevos algoritmos de selección de rutas o control, sin necesidad de modificar la infraestructura existente.

En el ámbito del balanceo de tráfico, las SDN facilitan la implementación de algoritmos adaptativos que, de acuerdo con la carga actual de la red, ajusta dinámicamente las rutas. Un controlador SDN puede identificar enlaces congestionados y redirigir flujos en tiempo real hacia rutas alternas con menor uso; algo complejo de implementar en redes tradicionales basadas en decisiones locales.

Este enfoque ha sido utilizado en entornos reales de alto tráfico, como en Google B4 [25], su red WAN privada global en la que utiliza SDN para distribuir tráfico entre sus centros de datos mediante técnicas de rutas múltiples y utilizando controladores centralizados. Los autores reportan que en su mayoría las conexiones alcanzan un uso cercano al 100 % de utilización, y que en promedio los enlaces se mantienen al ~70 % de utilización, lo que representa una mejora en dos y tres veces en eficiencia comparado con prácticas tradicionales. Otro ejemplo es Facebook Fabric [26], compuesta por una arquitectura SDN interna sobre una topología *fabric* uniforme con *pods* conectados a switches *spines*, lo que le permite disponer de múltiples rutas entre racks y *pods*. Empleando controladores jerárquicos se gestiona el tráfico entre racks de servidores, mejorando la flexibilidad y eficiencia operativa. Con Facebook Fabric se alcanza una rápida adaptación ante fallas o alto tráfico mediante un redireccionamiento programado de flujos. Estas implementaciones ratifican que el paradigma SDN es factible no solo en entornos académicos o simulados, sino también en redes de misión crítica a escala global.

Además, las SDN facilitan la integración con herramientas avanzadas de análisis y toma de decisiones, incluyendo los apoyados en inteligencia artificial (IA), lo cual sienta las bases para el desarrollo de paradigmas más avanzados como las KDN, que no están limitadas a programar reglas, sino que incluyen un proceso de aprendizaje con el fin de optimizar el comportamiento de la red.

F. Redes definidas por conocimiento (KDN)

En las metodologías *Knowledge-Defined Networking* [7] (KDN), el plano de conocimiento agrega una capa cognitiva en la red, permitiendo no solo la automatización, sino también la optimización basada en el contexto. A través de técnicas como minería de datos, predicción de tendencias y algoritmos de IA, es posible pronosticar congestiones, descubrir anomalías y proactivamente adaptar políticas.

Algunas de las características más destacadas de las KDN son las siguientes:

- Predicción: utilizando modelos entrenados con datos históricos, se pueden anticipar determinados patrones de tráfico como picos de carga o cuellos de botella.
- Dinámico: se pueden reasignar flujos o actualizar políticas en tiempo real en función de eventos emergentes.
- Retroalimentación: la red puede aprender a partir de decisiones previas y perfeccionar su actuación al incorporar ciclos de retroalimentación.

- Integración: sistemas de detección de intrusos, gestión de la energía o plataformas de virtualización, son algunas de las tecnologías con las cuales se puede integrar.

Las metodologías KDN establecen las bases para una red verdaderamente inteligente, en la cual el conocimiento se convierte en un activo estratégico. La aplicación de técnicas de balanceo de tráfico con algoritmos de aprendizaje por refuerzo abre las puertas a la construcción de sistemas que se adaptan de forma automática al entorno, optimizando el uso de los recursos disponibles.

G. Propuestas para el balanceo de tráfico en KDN

El uso de la IA en redes ha ganado una relevancia cada vez mayor, particularmente en entornos donde la complejidad y dinámica del tráfico dificultan la gestión convencional. En el ámbito del balanceo del tráfico, se han implementado algoritmos de IA para optimizar las rutas, anticipar congestiones y adaptar las decisiones de selección de rutas en tiempo real.

Dentro del conjunto de técnicas de IA aplicadas, se destacan:

- Algoritmos evolutivos Ant Colony Optimization [27] (ACO), Particle Swarm Optimization [28] (PSO), Genetic Algorithm [29] (GA): adecuados para problemas de optimización compleja con múltiples soluciones. Permite encontrar rutas eficientes en base a sus métricas de red, sin requerir información precisa de su entorno.
- Redes neuronales recurrentes Long Short-Term Memory [30] (LSTM), Bidirectional Long Short-Term Memory [31] (BiLSTM): tienen la capacidad de capturar dependencias temporales en el tráfico, lo cual permite predecir eventos futuros como momentos de congestión o picos de uso.
- Clustering y selección inteligente de rutas [32]: técnica que agrupa flujos o nodos de iguales características para una asignación más eficiente de las rutas. Apropiada para grandes redes en las que no se requiera un entrenamiento intensivo.
- Deep Q-Learning [33] (DQN): especializado en escenarios cambiantes sin modelos precisos del entorno. A partir de la experiencia, puede balancear el tráfico en tiempo real entre múltiples rutas posibles, maximizando métricas como la tasa de transferencia o el FCT.
- Deep Deterministic Policy Gradient [34] (DDPG): puede utilizar en espacios de acción continuo, permitiendo más granularidad que en DQN. Es útil en problemas en donde la elección de rutas para el balanceo depende de ajustes dinámicos de sus parámetros en tiempo real.

- Deep Reinforcement Learning [35] (DRL): optimiza decisiones a largo plazo tomando en cuenta las consecuencias acumuladas de cada acción. Contribuye a la prevención de tomar decisiones de corto alcance, logrando una optimización más sólida en el rendimiento de la red. Incluye a DQN, DDPG y a otros algoritmos.
- Federated learning y blockchain [36]: Muy útil cuando se trabaja entre múltiples controladores, trabajando de forma cooperativa sin depender de una autoridad central. Su aplicación se da en redes 5G o infraestructuras interinstitucionales.
- Algoritmos heurísticos con retroalimentación [37]: facilita una respuesta rápida basado en estado del entorno y en reglas adaptativas, obviando un entrenamiento profundo. Su beneficio está enfocado en entornos de recursos limitados en donde se requieran decisiones rápidas de bajo costo computacional.

En recientes investigaciones, el avance dirigido a redes autónomas apoyadas en SDN y KDN ha incrementado el interés por técnicas de DRL en la resolución de problemas de selección de rutas de manera adaptativa, asignación de recursos y balanceo dinámico del tráfico [38] [39]. No obstante, uno de los principales retos continúa siendo la estabilidad del aprendizaje y la convergencia de las políticas aprendidas, principalmente en entornos con alta variabilidad temporal, múltiples rutas disponibles y espacios de estados y acciones de grandes dimensiones [40]. En este sentido, arquitecturas como Dueling DQN, Double DQN y mecanismos de Prioritized Experience Replay se han utilizado para mejorar la eficiencia del aprendizaje, disminuir la sobreestimación de los valores Q y beneficiar una convergencia más estable frente a despliegues DQN convencionales. Estas mejoras refuerzan la pertinencia del estudio de soluciones basadas en aprendizaje por refuerzo profundo dentro de arquitecturas SDN/KDN, pero también justifican la necesidad de validar su desempeño en escenarios controlados, reproducibles y comparables delante de mecanismos tradicionales de balanceo.

En la Tabla I se presentan algunas de las principales propuestas halladas en la revisión, y clasificadas de acuerdo con el algoritmo utilizado, tipo de escenario de red considerado, la plataforma empleada en su validación experimental y las métricas consideradas en la evaluación del desempeño. Con esta comparación se aprecia cómo diferentes enfoques abordan el problema del balanceo de tráfico en varios contextos operativos, estableciendo las bases para el análisis del enfoque propuesto en la tesis:

TABLA I

PROPUESTAS PARA EL BALANCEO DE TRÁFICO EN KDN

Autor	Algoritmo	Escenario	Plataforma	Métricas	Resultado	Limitación
[41]	SDN + IA; KSP, Greedy, PSO, GA y SA	Red de operador / red mallada de gran escala	Simulación propia	Utilización máxima de enlaces, enlaces congestionados, servicios congestionados, tiempo de convergencia	Reduce la utilización máxima de enlaces por debajo del umbral en 10 escenarios, con convergencia < 10 s y sin enlaces congestionados remanentes	Usa datos sintéticos y red de operador; no evalúa FatTree, tráfico *mice/elephant* ni comparación ECMP/DRILL
[42]	G-ACO: integración de GA y ACO	SDN con topología FatTree y topología de 14 nodos	Mininet + OpenDayLight	Éxito en búsqueda de ruta óptima, RTT, pérdida de paquetes y tiempo de ejecución	Alcanza ~95 % de éxito en la búsqueda de ruta óptima y reduce RTT y pérdida de paquetes frente a RR y ACO; la pérdida queda entre 0.13 % y 0.20 %	Parámetros definidos empíricamente; evaluación en topologías pequeñas, sin tráfico *mice/elephant* ni comparación con DQN, ECMP o DRILL
[43]	ALBRL: DDPG mejorado con SumTree	Red SDN sobre topología NSFNet con distintas intensidades de tráfico	Mininet + Ryu + OpenFlow 1.3 + PyTorch	Throughput, recompensa/convergencia, factor de balanceo y utilización de enlaces	Mejora el throughput entre 4 % y 6 % frente a EARS en alta carga, entre 5 % y 18 % frente a RSIR y entre 10 % y 14 % frente a	Evalúa NSFNet, no FatTree; se centra en throughput/convergencia, sin FCT, tráfico *mice/elephant* ni

Autor	Algoritmo	Escenario	Plataforma	Métricas	Resultado	Limitación
					OSPF; además acelera la convergencia en alta carga	comparación con ECMP/DRIL L
[44]	DRL-TC basado en DDPG	Hybrid SDN en topologías Abilene, Nobel-germany y GEANT	Mininet + Ryu + D-ITG + Keras	MLU, delay, jitter, desviación estándar del delay y tráfico controlabl e	Mejora máxima de MLU de 20.77 % frente a Shortest; reduce delay, jitter y desviación estándar del delay en 68.17 %, 40.1 % y 28.16 %, respectivament e	Se enfoca en Hybrid SDN y QoS; no evalúa FatTree, tráfico *mice/elephant* ni FCT, y no compara ECMP/DRIL L/DQN bajo un mismo entorno
[45]	SINET: DRL con control parcial de nodos mediante *pinning control*	Redes SDN con topologías de 16, 34, 55 y 82 nodos	OMNeT++ 4.6 + Keras/Pytho n	FCT promedio, recompens a de entrenami ento, escalabilid ad y robustez	Reduce el FCT promedio al menos 10 % en una topología de 16 nodos y 32 % en una de 82 nodos; muestra mayor robustez ante cambios menores de topología	Se centra en escalabilidad del DRL; no evalúa FatTree, tráfico *mice/elephant* ni comparación con ECMP/DRIL L bajo un mismo entorno
[46]	Bio-Inspired DRL basado en RBM y aprendizaj	SDN distribuida con múltiples controlado res,	C++/WILL API modificada + datos CAIDA	Throughp ut, overhead de señalizaci ón, retardo	Reduce el overhead de señalización, aumenta el throughput y disminuye el	Escenario reducido; no evalúa FatTree, tráfico *mice/elephant*

Autor	Algoritmo	Escenario	Plataforma	Métricas	Resultado	Limitación
	e emocional	switches y servidores		promedio por salto y error de predicción	retardo promedio por salto frente a OSPF y Deep Learning convencional	ni FCT, y no compara con ECMP/DRILL/DQN
[47]	MLQU y DLQU basados en ML/DL con predicción de utilización de cola	Red de conmutación de paquetes / SDR con 30 nodos y 45 enlaces	Python 3.7 + TensorFlow + grafos regulares aleatorios	Packet Loss Ratio, worst throughput, average delay, queue utilization y error de predicción	Reducen la pérdida de paquetes y mejoran el peor throughput frente a BF y QUBF; el retardo promedio aumenta ~20 % frente a BF como compensación por mayor balanceo	Usa red genérica de 30 nodos; no evalúa FatTree, tráfico *mice/elephant* ni FCT, y no compara con ECMP/DRILL/DQN
[48]	CFR-RL: RL para selección de flujos críticos + LP para reruteo	Ingeniería de tráfico en SDN sobre redes ISP/backbone	TensorFlow + Gurobi; topologías Abilene y Rocketfuel	Maximum Link Utilization, balanceo, retardo extremo a extremo y perturbación por reruteo	Logra desempeño cercano al óptimo reruteando solo 10 %–21.3 % del tráfico total; mejora el balanceo hasta 32.8 % frente a ECMP y hasta 12.2 % frente a Top-K Critical	Enfocado en redes ISP/backbone; no evalúa FatTree, tráfico *mice/elephant* ni FCT, y no compara con DRILL/DQN bajo un mismo entorno

Autor	Algoritmo	Escenario	Plataforma	Métricas	Resultado	Limitación
[49]	BN Q-learning para balanceo de carga de controladores SDN	SDN de centro de datos / IoT con múltiples controladores y migración de switches	Python 3.6 + topología NSFMET	Retardo switch-controlador, grado de balanceo de carga, migraciones de switches y throughput	Reduce el retardo promedio en 74.9 % frente a Q-learning; disminuye el grado promedio de balanceo de carga en 27.5 % y su valor de convergencia en 19 %	Se centra en carga de controladores y migración de switches; no evalúa FatTree, tráfico *mice/elephant* ni FCT, y no compara con ECMP/DRILL/DQN
[50]	KDN-FLB: KDN con Federated Learning y Blockchain	Redes dinámicas y descentralizadas; ingeniería de tráfico con clasificación de flujos largos/cortos	ISCX2016 + FlowMeter + SMOTE + Random Forest	Accuracy, precision, recall, F1-score y umbral dinámico	La clasificación dinámica basada en KDN-FLB con dataset fusionado estabiliza las métricas por encima del 99 %	Arquitectura conceptual; no evalúa balanceo de rutas en FatTree, FCT, ECMP/DRILL/DQN ni control dinámico de tráfico paso a paso

En el análisis comparativo de los resultados reportados en la Tabla I evidencia que las propuestas existentes muestran mejoras relevantes en cuanto a métricas específicas como latencia, throughput, FCT, utilización de enlaces, pérdida de paquetes o balanceo de carga. No obstante, estos resultados no siempre son directamente comparables, por las diferencias en las topologías utilizadas, plataformas de simulación o emulación, perfiles de tráfico, métricas evaluadas y líneas base evaluadas. Esta falta de homogeneidad dificulta la posibilidad de establecer conclusiones generales respecto a la superioridad de una técnica frente a otra, y resalta la necesidad de evaluar DQN, ECMP y DRILL bajo un entorno común, con idénticas trazas de tráfico, condiciones experimentales y métricas de comparación.

En general, la comparación presentada en la Tabla I evidencia que, aunque existen múltiples aproximaciones apoyadas en la inteligencia artificial para optimizar el tráfico en redes SDN/KDN, se mantiene un vacío en la evaluación de modelos DQN destinados concretamente al balanceo dinámico en topologías FatTree, bajo tráfico heterogéneo *mice/elephant*, escenarios reproducibles y comparación directa ante mecanismos tradicionales y heurísticos como ECMP y DRILL. Este vacío fundamenta el desarrollo de un entorno de simulación controlado que facilite la evaluación, bajo condiciones equivalentes, las capacidades de un agente DQN para mejorar el FCT y el balanceo de tráfico en centros de datos.

Esta diversidad de técnicas evidencia la versatilidad de la IA en contextos de red; sin embargo, también demuestra que la aplicabilidad de cada técnica depende directamente del problema a resolver. Por ejemplo, los algoritmos evolutivos como ACO, PSO o GA son apropiados cuando se requiere explorar espacios de solución muy grandes, pero convergen muy lentamente, lo que lo limita en escenarios donde las condiciones del tráfico cambian de forma rápida. Del mismo modo, técnicas como el *clustering* o el aprendizaje no supervisado son efectivas para descubrir patrones globales o agrupar comportamientos similares, pero no capturan dependencias temporales ni responden en tiempo real a congestiones emergentes. En contraste, los métodos basados en aprendizaje por refuerzo profundo, en concreto DQN, son más adecuados en entornos altamente dinámicos, sin modelos precisos del tráfico, con múltiples rutas alternativas y la necesidad de decisiones secuenciales en tiempo real, como los encontramos en las redes SDN con topología FatTree. Por estos motivos, el presente trabajo profundiza en el uso de DQN como técnica de balanceo de tráfico, aprovechando su capacidad para aprender políticas adaptativas, reaccionar a cambios repentinos en la congestión y optimizar rutas sin depender de heurísticas predefinidas ni de información completa del entorno.

H. Simuladores y plataformas experimentales para redes

El estudio de mecanismos de balanceo de tráfico en redes puede realizarse por medio de diferentes plataformas experimentales. Con Mininet se pueden emular redes SDN utilizando hosts, switches y controladores virtuales, por ello resulta útil para la validación de prototipos cercanos a implementaciones reales [51]. OMNeT++ proporciona un entorno de simulación discreta orientado a redes y sistemas distribuidos, apropiado para el estudio de protocolos y comportamientos de tráfico con un alto nivel de parametrización [52]. NS-3 es mayormente utilizado para simulación de redes en modelos detallados de protocolos, enlaces y movilidad, no obstante, su integración

directa con agentes de aprendizaje profundo puede que requiera componentes adicionales [53]. SUMO, aunque se encuentre orientado principalmente a movilidad y tráfico vehicular, suele aparecer en algunos estudios sobre redes inteligentes y control distribuido [54]. En definitiva, los simuladores propios adaptan con mayor precisión la dinámica del problema, controlan las métricas y garantizan condiciones homogéneas entre algoritmos, aunque limitan parte de la validación cercana a entornos reales.

En este trabajo se optó por un simulador propio en Python por la necesidad de controlar simultáneamente la topología FatTree, las rutas precalculadas, la activación temporal de flujos, la dinámica de transmisión por pasos, el registro de métricas internas y la integración directa con el agente DQN. En pruebas preliminares con plataformas de emulación, como Mininet, se detectaron limitaciones para lograr mediciones consistentes en escalas temporales muy pequeñas, en especial con métricas sensibles como el Flow Completion Time (FCT). Debido a esto, se motivó el uso de un entorno de simulación controlado, en donde la evolución temporal de los flujos, la asignación de capacidad por enlace y el cálculo de métricas se pudiera reproducir de forma determinista. Con esta decisión se aseguró una comparación equitativa entre DQN, ECMP y DRILL con las mismas trazas de tráfico, condiciones experimentales y métricas de evaluación, sin embargo, se reconoce que en futuras investigaciones se podría validar el modelo propuesto en otras plataformas de emulación como Mininet o en entornos SDN físicos.

I. Revisión crítica y brechas detectadas

En la revisión del estado del arte se evidencia una importante evolución en las técnicas de balanceo de tráfico, desde métodos deterministas como ECMP hasta estrategias basadas en IA. Sin embargo, existen limitaciones que restringen su aplicación en entornos dinámicos y de alta demanda, como son los centros de datos modernos. Técnicas tradicionales como ECMP, Flowlets o VLB evidencian limitaciones propias de su diseño. En ECMP, la posibilidad de que generen colisiones afecta la equidad en la distribución del tráfico. Flowlets mejora este aspecto mediante la división de los flujos en subflujos con pausas naturales, pero ante tráfico continuo, su eficacia se reduce. Mecanismos más adaptativos se implementan con Hedera y Presto, pero un monitoreo continuo o el procesamiento adicional en los hosts, dificulta su implementación. DRILL introduce mejoras mediante el uso de *power-of-two choices*, pero necesita, visibilidad de la congestión y una sincronización eficiente entre nodos. Por otra parte, las técnicas basadas en IA han mostrado resultados alentadores y a la vez plantean importantes desafíos. Aunque los algoritmos evolutivos

(ACO, PSO, GA) ofrecen soluciones optimizadas, no tienen la capacidad de adaptación en tiempo real. La predicción de patrones de tráfico que se implementan en redes neuronales como BiLSTM necesitan entrenamiento supervisado y conjuntos de datos que sean representativos, lo cual restringe su generalización. Algoritmos de aprendizaje por refuerzo profundo (DQN, DDPG) se caracterizan por su capacidad de adaptarse y aprender de forma continua, sin embargo, los escenarios implementados en las investigaciones suelen utilizar ambientes de simulación muy simplificados, con topologías reducidas y patrones de tráfico artificiales. En estos escenarios se dificulta validar su efectividad en entornos realistas con flujos de diferente tamaño (*mice* y *elephant*), tráfico en ráfagas y condiciones de congestión emergente y variable en el tiempo. En este punto se marca una diferencia fundamental respecto a los algoritmos evolutivos clásicos como GA, ACO y PSO: su enfoque estático asume un estado de red estable durante todo el proceso de cálculo, lo que los limita frente a entornos dinámicos. Estos algoritmos carecen de mecanismos que les permitan ajustar sus decisiones cuando hay cambios en tiempo real, disminuyendo su aplicabilidad en centros de datos modernos con alta variabilidad y cargas impredecibles. Además, el costo computacional puede ser muy alto, y la ejecución muchas veces no se ajusta a los requerimientos de latencia que exige el tráfico en vivo. Además, la comparación de los resultados entre estudios es compleja, debido a la diversidad en las plataformas de experimentación (Mininet [55], OMNeT++ [56], SUMO [57], simuladores propios). Las métricas empleadas también varían (FCT, *jitter*, tasa de transferencia -throughput, tasa de error), y el impacto del algoritmo sobre la eficiencia general de los recursos de red no siempre se evalúa. Como resultado del análisis, se identifican tres brechas fundamentales:

- Limitación en los escenarios de evaluación: son escasos los estudios que utilizan topologías como FatTree bajo situaciones de tráfico realista.
- Incomparabilidad: la diversidad de plataformas y métricas dificulta la creación de comparativas comunes.
- Transferencia limitada del aprendizaje: gran parte de los modelos no se han validado en entornos distintos al que fueron entrenados.

Esta investigación tiene como fin el contribuir al cierre de dichas brechas mediante el desarrollo de un entorno de simulación controlado pero representativo. En éste se evaluarán métricas claves como el FCT y el balanceo de la red, posibilitando la comparación sistemática entre técnicas tradicionales y DQN en topología FatTree con tráfico dinámico.

VI. METODOLOGÍA

A. Enfoque de investigación

El objetivo principal de esta investigación es diseñar, entrenar y evaluar un modelo inteligente para balancear el tráfico en centros de datos utilizando aprendizaje por refuerzo profundo, específicamente el algoritmo DQN, y comparar su desempeño frente a técnicas tradicionales como ECMP y DRILL. Se asume que la arquitectura SDN, junto a metodologías KDN, ofrecen las capacidades requeridas para implementar entornos de decisión basados en datos. En este sentido, el agente inteligente monitorea el estado de la red, aprende de la experiencia pasada y ajusta sus decisiones de selección de rutas de forma dinámica. El enfoque experimental se desarrolla utilizando simulaciones controladas en una topología FatTree evaluando múltiples escenarios de tráfico, que incluyen flujos de varios tamaños (*mice* y *elephant*), así como tráfico en ráfagas. Se construye un marco de trabajo modular que posibilita la integración de diversos algoritmos y la recopilación de métricas claves, asegurando condiciones uniformes -como la topología de red y secuencia de flujos utilizados– para facilitar una comparación consistente de las distintas técnicas. La metodología se organiza en tres fases: diseño del entorno de simulación y del agente de aprendizaje, el entrenamiento del modelo en diferentes condiciones, y la comparación directa con otros mecanismos de balanceo. Cada fase se ejecuta con precisión técnica, lo que permite analizar el impacto del algoritmo propuesto y analizar sus ventajas y limitaciones mediante métricas objetivas.

B. Arquitectura general del sistema propuesto

La estructura de la propuesta está integrada por cinco elementos principales. En la Fig. 4 se muestra su arquitectura por medio de un diagrama de bloques. En el contexto de las KDN, esta arquitectura integra un ciclo de observación, aprendizaje, decisión y actuación sobre la red. El entorno de simulación provee la información del estado de los enlaces y flujos; el módulo de gestión del tráfico produce condiciones variables de carga; el agente DQN convierte dicha información en conocimiento operativo por medio del aprendizaje de una política de selección de rutas; y con la simulación por modelo se ejecutan las decisiones de selección de rutas y su posterior comparación frente a ECMP y DRILL. Así, la propuesta no constituye únicamente una aplicación aislada de inteligencia artificial, sino también una arquitectura experimental orientada al conocimiento, en donde los datos del estado de red se transforman en decisiones adaptativas orientadas al balanceo del tráfico.

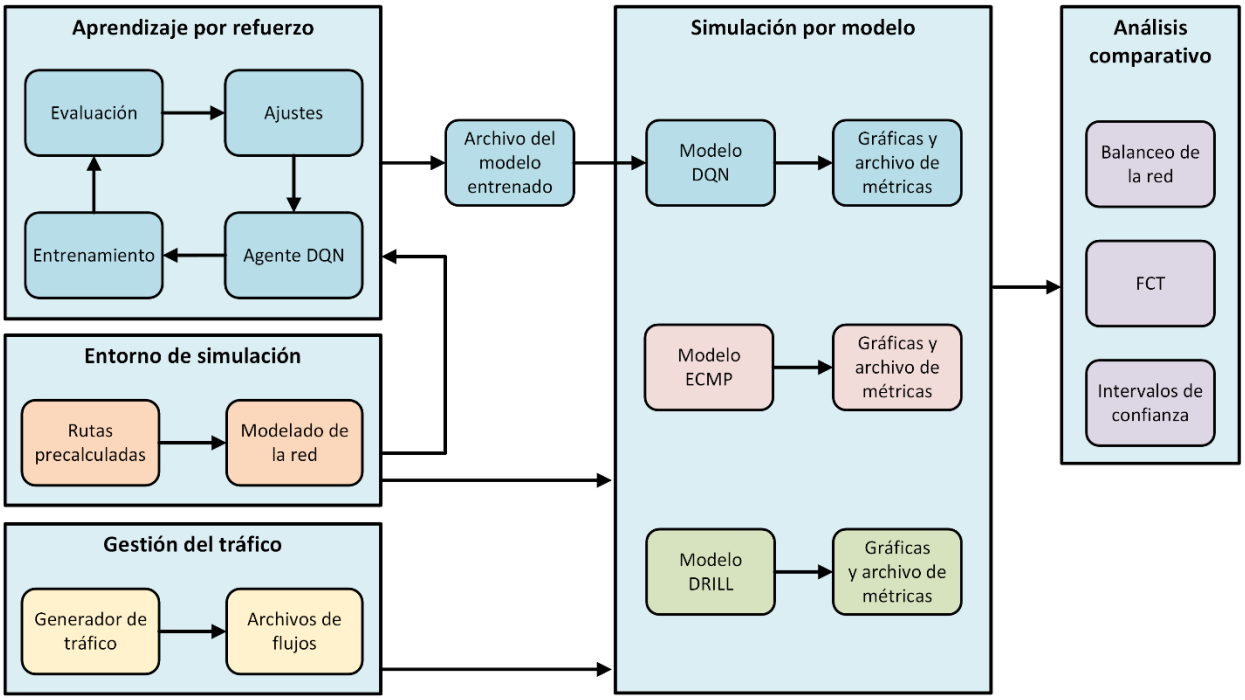

Fig. 4 Arquitectura del sistema

- Entorno de simulación: Implementa una red tipo FatTree con K=4 compuesto por múltiples rutas entre los nodos con enlaces simétricos. Este entorno hace posible la observación el estado de congestión en cada enlace y la simulación del paso de los flujos con comportamiento dinámico.
- Gestión del tráfico: Crea flujos de datos clasificándolos en *mice* y *elephant*, controlando su llegada y duración, además de generar patrones en ráfagas o constantes. Para cada nivel de carga (λ bajo, medio y alto) se crean múltiples escenarios independientes, cada uno con flujos generados aleatoriamente, lo que permite evaluar el sistema en condiciones de tráfico diversas. Esto permite evaluar la fortaleza del agente en condiciones cambiantes en el tráfico.
- Aprendizaje por refuerzo: Efectúa un proceso de entrenamiento del agente DQN. Incluye la generación de episodios, actualización de la política mediante recompensas basadas en congestión y equilibrio del tráfico, y un ciclo de evaluación continua de ajuste de hiperparámetros para mejorar el desempeño. El resultado final es un modelo entrenado capaz de seleccionar rutas de forma adaptativa según el estado de la red.
- Simulación por modelo: Ejecuta de manera independiente los tres mecanismos de asignación de rutas: DQN, ECMP y DRILL. Cada modelo procesa los mismos

escenarios de tráfico y registra las mismas métricas clave como FCT, uso de enlaces y patrones de balanceo. ECMP y DRILL fueron escogidos como líneas base porque figuran como dos enfoques contrastantes: ECMP como mecanismo estático, simple y comúnmente utilizado para rutas de igual costo, y DRILL como heurística dinámica de bajo costo que integra información local de carga. La comparación permite valorar si el agente DQN aporta beneficios ante una política tradicional y respecto a una alternativa dinámica reconocida en balanceo de tráfico. En este módulo se generan las gráficas y los archivos de resultados necesarios para el análisis posterior.

- Análisis comparativo: Integra y compara las métricas que se obtienen en cada modelo, y evalúa su rendimiento en las mismas condiciones de tráfico. Comprende el análisis de los resultados de balanceo de carga, FCT, y el respectivo análisis estadístico e intervalos de confianza, que permiten determinar la efectividad relativa de cada enfoque.

Por medio de esta estructura se pueden realizar pruebas sistemáticas en múltiples escenarios, lo que garantiza consistencia y comparabilidad entre experimentos. Futuras investigaciones que impliquen cambios en los algoritmos o ampliaciones del entorno se facilitan por lo modular del sistema.

C. Diseño del entorno de simulación

El entorno de simulación es el ambiente central sobre el cual se realizan y evalúan los distintos experimentos sobre las técnicas de balanceo de tráfico. Fue implementado en Python y simula una red de centros de datos con topología FatTree de grado K=4. La adopción de un simulador propio en Python satisfizo la necesidad de controlar de modo determinista la evolución temporal de los flujos, la distribución de capacidad por enlace y el cálculo de métricas sensibles como el FCT, característica que se complementa con el análisis valorativo de plataformas experimentales mostrado en la sección de simuladores. Al ser un entorno modular, permite la experimentación controlada en condiciones de tráfico realista y una integración directa con agentes de IA.

La topología FatTree utilizada en la investigación fue presentada previamente en la Fig. 1, mientras que la arquitectura general del sistema se resume en la Fig. 4. Para complementar dichas representaciones, la Fig. 5 detalla el funcionamiento interno del entorno de simulación implementado en Python, exponiendo la secuencia mediante la cual se cargan los flujos, se

inicializa la topología, se ejecuta la simulación discreta por pasos y se registran las métricas utilizadas en la evaluación experimental.

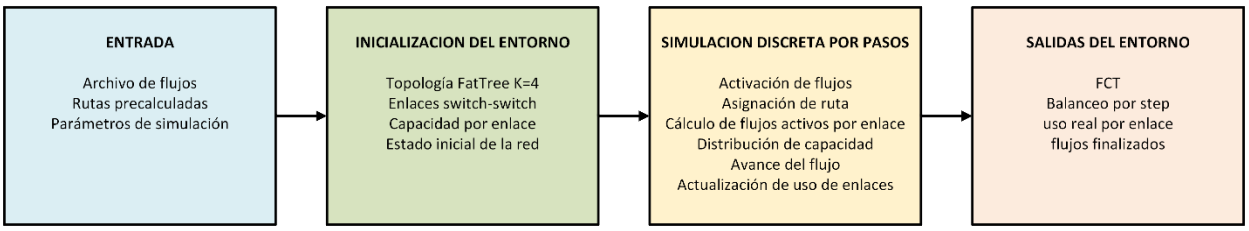

Fig. 5 Funcionamiento interno del entorno de simulación

- Características de la topología FatTree: La topología FatTree fue seleccionada por ser ampliamente utilizada en centros de datos modernos y por su capacidad de ofrecer múltiples rutas de igual costo entre pares origen-destino. En el ambiente experimental la red consta de 20 switches (8 en capa de borde, 8 en distribución y 4 en el núcleo) y 16 hosts etiquetados de Z1 a Z16 e interconectados simétricamente. Esta elección se justifica en que K=4 mantiene las propiedades estructurales relevantes de FatTree —simetría, redundancia y múltiples rutas mínimas—, a la vez que mantiene el tamaño del entorno en un nivel computacionalmente manejable para el entrenamiento y la evaluación del agente DQN. Esto es coherente con la configuración experimental descrita posteriormente, en donde se evalúan los 16 hosts y 20 switches en condiciones homogéneas para DQN, ECMP y DRILL. En la topología se garantizan múltiples rutas entre cualquier par de hosts. Cada enlace funciona bidireccionalmente y para el registro de su utilización, congestión y disponibilidad, se realiza un monitoreo constante.
- Generación de flujos: El generador de tráfico está diseñado para replicar una variedad de condiciones que pongan a prueba al agente en ambientes de red realistas [15]. Los flujos se generan a partir de procesos de Poisson con tasas de llegada configurables. Esto permite ajustar la intensidad del tráfico para la simulación de baja a alta demanda. Cada flujo se genera en base a: Un host origen y destino, seleccionados de forma aleatoria, una tasa de llegada (λ), que define la frecuencia de nuevos flujos asumiendo un modelo de Poisson, y un tamaño, que de forma aleatoria representan a los flujos *mice* (corta duración y tamaño: 20-100 KB) o *elephant* (larga duración y tamaño: 30-50 MB). Además, de la totalidad de los flujos generados, el 90% corresponde al tipo *mice* y el 10% al tipo *elephant*. Esta proporción 90/10 se eligió para representar la asimetría típica

del tráfico en centros de datos, en donde predominan los flujos pequeños, en tanto que una proporción menor de flujos grandes puede concentrar un uso sostenido de enlaces y generar congestión. De esta manera, se emplearon los valores de λ = 100, 200 y 250 para representar tres niveles progresivos de carga —bajo, medio y alto—, cuya configuración se explica en detalle en la sección de resultados experimentales. Todas estas configuraciones permiten recrear momentos de alta concurrencia de flujos, lo que puede dar origen a saturaciones, replicando así ambientes realistas. Los flujos se estructuran en tuplas (origen, destino, tamaño y tiempo de inicio) y se almacenan en archivos que son cargados al momento de la simulación, lo cual favorece a que los experimentos puedan repetirse entre algoritmos y hacer una comparación directa. Igualmente, durante el modo de validación del agente se implementan rutinas que permiten precargar flujos; esto con la finalidad de analizar su comportamiento frente a patrones de tráfico específicos. Esto es fundamental debido a que permite evaluar la capacidad del modelo de generalización ante datos no utilizados en la fase de entrenamiento, esto con el fin de verificar que el entrenamiento del sistema generaliza y no presente problemas de sobre-ajuste.

D. Solución basada en aprendizaje por refuerzo e implementación mediante DQN

El núcleo de este estudio corresponde al desarrollo de un agente basado en aprendizaje por refuerzo profundo, el cual haya sido entrenado para la selección de las mejores rutas para los flujos del tráfico interno en un centro de datos. Fundamentado en el algoritmo DQN, el modelo fue desarrollado para incrementar su precisión, capacidad de generalización y velocidad de convergencia.

Formalmente, el problema de selección de rutas es modelado como un proceso de decisión de Markov, definido por la tupla:

$$\mathcal{M} = (S, A, P, R, \gamma)$$

donde S representa el espacio de estados observables de la red; A es el conjunto de acciones posibles del agente; P se refiere a la dinámica de transición entre estados como consecuencia de la acción tomada y de la evolución del tráfico; R define la función de recompensa que permite evaluar la calidad de la decisión de selección de rutas; y γ es el factor de descuento que valora la importancia de las recompensas futuras.

En este trabajo, el estado $s_t \in S$ se elabora a partir del uso normalizado de los enlaces entre switches, la demanda acumulada relacionada con los flujos activos y la presencia de flujos tipo *elephant* en la red. La acción $a_t \in A$ se asocia a la selección de una ruta mínima precalculada para el par origen-destino del flujo que necesita una asignación. La transición $P(s_{t+1} \mid s_t, a_t)$ resulta de la activación de nuevos flujos, la ruta seleccionada, la competencia por capacidad de los enlaces, el avance de la transmisión y la actualización del uso de los enlaces en el siguiente paso de la simulación. La recompensa $R(s_t, a_t)$ está diseñada para penalizar decisiones que generen congestión, un desbalance en la utilización de enlaces o acciones inválidas. Por último, el factor de descuento utilizado en el entrenamiento es $\gamma = 0.95$, permitiendo que el agente considere el efecto inmediato de una decisión de selección de rutas, además de sus consecuencias futuras sobre la congestión de la red.

El agente tiene como objetivo el aprender una política $\pi(s)$ que le permita escoger la mejor acción disponible para cada estado de la red, maximizando el retorno esperado acumulado. Desde el punto de vista formal, la función óptima de valor-acción puede expresarse como:

$$Q^*(s,a) = \max_{\pi} \mathbb{E}\left[\sum_{t=0}^{T} \gamma^t r_t \mid s_0 = s, a_0 = a, \pi\right]$$

donde $Q^*(s,a)$ representa el valor esperado al tomar la acción a en el estado s y posteriormente continuar con la política óptima; r_t es la recompensa obtenida en el paso t; γ es el factor de descuento; y T representa el horizonte temporal del episodio de simulación.

Puesto que el número de combinaciones posibles entre los estados de congestión y rutas puede ser alto, en la función $Q(s,a)$ no se almacena en una tabla, sino que se aproxima mediante una red neuronal profunda con parámetros θ. Por lo tanto, el agente DQN aprende una aproximación $Q(s,a;\theta)$, que calcula la utilidad esperada de seleccionar cada ruta disponible en función del estado actual de la red. En el entrenamiento, los parámetros θ se actualizan minimizando la diferencia entre el valor Q estimado y el valor objetivo, calculado en función de la recompensa inmediata y la estimación del estado siguiente.

La función de recompensa se estableció para conducir al agente a decisiones de selección de rutas que minimicen el desbalance en el uso de enlaces, eviten el desarrollo de puntos de congestión y penalicen decisiones no válidas en el entorno de simulación. Debido a esto, la recompensa final en el instante t se formuló:

$$R_t = clip(r_{env,t} + \alpha r_{local,t}, -1{,}1)$$

donde R_t corresponde a la recompensa final entregada al agente; $r_{env,t}$ el valor de la recompensa global obtenida a partir del estado completo de la red en el entorno; $r_{local,t}$ representa el valor de la recompensa local asociada a la calidad de la ruta escogida para el flujo; y α para controlar el peso relativo del componente local. Mediante la función $clip(\cdot)$ se restringe la recompensa R_t al intervalo $[-1,1]$, para estabilizar el entrenamiento del agente DQN.

Para la recompensa global se considera el desbalance en el uso de enlaces entre switches, la existencia de puntos de congestión, los picos de utilización y la validación de acciones o rutas no válidas. A su vez, la recompensa local valora la calidad de la ruta asignada a cada flujo, tomando en cuenta la congestión y el balanceo presentes en los enlaces que conforman dicha ruta. Así, la señal de aprendizaje integra un diagnóstico general del estado de la red con un diagnóstico específico de la decisión que se ha tomado para cada flujo, beneficiando a las políticas de selección de rutas que minimicen la congestión y perfeccionen la distribución del tráfico. Las especificaciones de cada componente de la recompensa se detallan en la sección correspondiente al módulo de aprendizaje por refuerzo.

- Espacio de estados: Conjunto de todas las posibles situaciones que el agente puede observar en su entorno. La situación actual de la red se representa en cada estado, en la que se incluye el uso de cada enlace programada como vectores que alimentan la red neuronal.
- Espacio de acciones: Conjunto de acciones posibles que el agente puede llevar a cabo en cada estado. Aquí cada acción implica la elección de una de las rutas precalculadas entre un par origen-destino. Para construir estas rutas se emplea un algoritmo basado en *k-shortest paths*, en donde se generan únicamente las rutas más cortas entre cada par de hosts. La estructura altamente regular de la topología FatTree garantiza que todas las rutas mínimas sean de igual longitud y sigan patrones repetitivos, esto es, con K=4 para cualquier flujo existan como máximo cuatro rutas mínimas. Esta característica permite representar de forma compacta el espacio de rutas, evitando un crecimiento exponencial del número de acciones. Si bien esta reducción del espacio de búsqueda facilita el entrenamiento y la evaluación del agente DQN, puede volverse limitada en redes más grandes, donde el número de pares origen–destino crece cuadráticamente y las rutas mínimas pueden superar ampliamente el número manejable de acciones. Para mitigar este efecto, se podrían investigar estrategias alternativas, como la selección adaptativa

de subconjuntos de rutas relevantes según la congestión actual, la generación de rutas mediante algoritmos heurísticos, o el uso de políticas jerárquicas en la que el agente primero aprende en que dominio/nivel de la topología selecciona rutas, y posteriormente, la ruta específica. Estas estrategias mantendrían un espacio de acciones manejable, conservando la capacidad del modelo para tomar buenas decisiones en ambientes complejos.

- Función de recompensa: Cuantifica que tan buena o mala fue la acción tomada por el agente en un estado dado. Se diseña como un sistema compuesto que penaliza las decisiones que producen desbalance, un uso excesivo de los enlaces y mantener coherentemente las decisiones de selección de rutas. En el desarrollo, se contempló una penalización por desbalance, compuesta por la desviación estándar del uso real de los enlaces entre switches; una penalización por puntos de congestión, que se define como la proporción de enlaces cuyo uso supera un umbral de congestión θ_{hot}; y una penalización más añadida por el pico de utilización, que es aplicada cuando el uso máximo de los enlaces sobrepasa dicho umbral. Así mismo, en el entorno se implementa un mecanismo de validación que penaliza acciones o rutas que no sean válidas para el par origen-destino que se evaluó. Para finalizar, la recompensa se recorta al intervalo $[-1,1]$ con el fin de estabilizar el entrenamiento. Si bien no se otorgan bonificaciones explícitas, si existe un incentivo indirecto hacia las políticas que reparten mejor la carga y reducen la congestión.

El modelo se entrena en episodios simulados, en los cuales, en cada paso temporal, se generan flujos y el agente decide la ruta. El entrenamiento incluye:

- Política ε-greedy [58] : Estrategia que consiste en realizar una acción aleatoria con una probabilidad ε denominada exploración y seleccionar la mejor acción conocida con una probabilidad 1−ε conocida como explotación. Con esto se logra que el agente inicie explorando nuevas rutas al principio y luego conforme avance el entrenamiento, se centre en las mejores. Se emplea una política de exploración con un decaimiento programado que inicia con ε=1.0 y desciende gradualmente hasta un mínimo de 0.08. El decaimiento se define mediante una constante calculada para llegar desde 1.0 hasta 0.08 en un número objetivo de episodios.

- Memoria de experiencia con Prioritized Experience Replay [59] (PER): En lugar de un búfer FIFO o un deque de tamaño fijo, el agente almacena transiciones en una memoria priorizada. Las muestras se seleccionan según su error temporal (TD), medida de la discrepancia entre la estimación actual y la actualización objetivo, otorgando más probabilidad a experiencias informativas. Se utilizan de igual manera los pesos de muestreo por importancia [60] (*importance sampling weights*) para compensar el sesgo generado por el muestreo no uniforme.
- Arquitectura del agente Dueling Double DQN [61] : El sistema utiliza dos redes neuronales: una red Q principal (online network) y una red Q objetivo (*target network*). Ambas redes implementan la arquitectura Dueling DQN, que separa la estimación del valor del estado y la ventaja de cada acción. Adicionalmente, se emplea el esquema Double DQN, donde la red principal selecciona la acción y la red objetivo calcula el valor correspondiente, reduciendo el sesgo por sobreestimación. La red objetivo no se actualiza de forma abrupta, sino mediante una actualización suave (*soft-update*) con coeficiente, lo que permite una transición suave y estable entre políticas sucesivas.

Con respecto a los hiperparámetros, se consideraron los siguientes:

- Capas ocultas: Niveles intermedios que se establecen entre la entrada y salida de la red neuronal para que el modelo aprenda patrones complejos. Se utilizan dos capas ocultas compuestas de 128 neuronas cada una, activadas mediante funciones tipo ReLU [62] . Luego, en la arquitectura Dueling se divide en dos ramas: una de valor del estado y la otra de ventaja de las acciones, conformada cada una con una capa intermedia de 64 neuronas. Esta arquitectura se estableció de forma empírica después de experimentar en pruebas iniciales con varias arquitecturas de red, buscando un equilibrio entre la capacidad de representación y la estabilidad en el entrenamiento. Dado que el escenario posee un espacio de estados discreto y moderadamente acotado, una arquitectura más profunda no mejoraba el rendimiento, y, por el contrario, aumentaba el tiempo de entrenamiento y el riesgo de sobre ajuste. Debido a esto, se optó por una arquitectura sencilla que facilite la estimación de la función Q (s, a) de forma eficiente y con buena convergencia.
- Función de pérdida: Mide el error entre la salida del modelo y el valor esperado, por lo que el modelo aprende minimizando esta función. Se utiliza Huber [63]/Smooth L1 [64]

como criterio de aprendizaje, para minimizar la diferencia entre la Q estimada y la objetivo. Está ponderada por los pesos de muestreo por importancia de PER, con una regularización L2 ligera.

- Optimización: Adapta los pesos de la red minimizando la función de pérdida. El entrenamiento se realiza con el optimizador Adam, el cual ajusta dinámicamente las tasas de actualización de cada parámetro internamente. Se inicia con una tasa de aprendizaje del orden de 10^{-5}, ajustada dinámicamente mediante un scheduler ReduceLROnPlateau [65].
- Tamaño de lote (*batch*): Cantidad de muestras utilizadas en cada paso, en relación directa con la estabilidad y la velocidad de aprendizaje. Se entrena con un tamaño de lote fijo de 128 muestras por iteración, extraídas aleatoriamente desde la memoria de experiencias.
- Factor de descuento γ: La valoración que da el agente a las recompensas futuras frente a las presentes, siendo un valor de 1 la que da más valor a decisiones con beneficios en el largo plazo. Se fija en 0.95.
- Entrenamiento por episodios: El modelo es entrenado consecutivamente en múltiples episodios con flujos generados de forma pseudoaleatoria.

El resultado de este diseño es un agente con la capacidad de aprender políticas que se adaptan en función del estado de la red, evitando congestiones y distribuyendo los flujos de forma eficiente considerando escenarios de carga variable.

E. Fases de implementación

Se diseñó una metodología en tres etapas para construir, validar y comparar un sistema inteligente de balanceo de tráfico basado en DQN. Cada etapa se centró en un conjunto particular de acciones, refinando gradualmente el modelo y manteniéndolo en línea con los objetivos de la investigación.

En la primera etapa se diseñó y desarrolló el entorno de simulación controlado, con una topología FatTree K=4 y una arquitectura modular que divide el entorno de la red, el agente inteligente y los generadores de tráfico. Las principales actividades fueron:

- Creación del entorno bajo un modelo en el que los enlaces entre switches tienen capacidad finita y ocupación variable.

- Implementación del generador de flujos con procesos de Poisson, parámetros ajustables y flujos aleatorios tipo *mice* y *elephant*.
- Programación del espacio de estados en forma de vectores que representan la fracción de ocupación (0-1) de cada enlace entre switches. Dado que todos los enlaces del modelo poseen igual capacidad, esta representación permite comparar directamente los niveles de uso y detectar congestión de manera uniforme.
- Desarrollo de la función de recompensa compuesta y basada en las métricas de saturación y desviación estándar de los enlaces.
- Entrenamiento inicial del agente con un algoritmo DQN usando la política ε-greedy y el búfer de experiencia.
- Diseño de la secuencia de procesos: entrenamiento, evaluación y registro de resultados.
- Esta etapa terminó con la verificación funcional del sistema y las primeras señales de aprendizaje del agente. En la segunda etapa, el modelo se sometió a condiciones más complejas y variadas, con el objetivo de evaluar su capacidad de adaptación y generalización:
- Entrenamiento del agente con múltiples episodios continuos, variando parámetros como tasa de llegada (λ) y el tamaño de los flujos.
- Implementación de instrucciones para la carga de archivos de tráfico generados con anterioridad, con lo que se simulan escenarios reproducibles que incluyen picos de congestión y/o cambios repentinos de carga.
- Afinamiento de hiperparámetros, en la que se incluyen la tasa de aprendizaje, el factor de descuento (γ), tamaño de lote y estrategia de exploración (ε).
- Revisión constante del promedio del FCT y de la desviación estándar por la utilización de los enlaces, para evaluar el rendimiento del modelo.
- Presentación de las métricas relevantes y análisis cualitativo de las rutas seleccionadas por el agente en distintas situaciones.

La etapa final consistió en la comparación del modelo propuesto con mecanismos ampliamente utilizados en la literatura y en centros de datos, en condiciones experimentales controladas y equivalentes:

- Se implementaron como referencia los métodos ECMP y DRILL, bajo el mismo entorno de simulación que el agente DQN. ECMP es el método estático clásico que

distribuye los flujos utilizando algoritmos como *round-robin* que no tienen en cuenta el estado de la red, mientras que DRILL es un método adaptativo basado en decisiones locales que escoge la ruta menos cargada entre dos aleatorias. La comparación contra estos dos modelos permite evaluar el desempeño del modelo DQN frente a una estrategia determinista y estática, como frente a una estrategia dinámica, pero sin aprendizaje acumulativo.

- Para garantizar una comparación equitativa entre los tres modelos, las simulaciones se repitieron para distintos conjuntos independientes de flujos, previamente generados y almacenados. Cada archivo mantiene fijo sus tiempos de llegada de los flujos y sus tamaños, para que ECMP, DRILL y DQN compitan en las mismas condiciones. Múltiples conjuntos de tráfico permiten obtener resultados estadísticamente más robustos y calcular intervalos de confianza para las métricas de red.
- Las métricas utilizadas para la comparación incluyeron FCT promedio y desviación estándar de la ocupación de los enlaces de la topología.
- Se realizó un análisis estadístico de los resultados incluyendo el cálculo de intervalos de confianza (IC-95%) mediante t-Student, con el fin de identificar tendencias y variabilidad entre ejecuciones y comportamientos inusuales en las métricas evaluadas.
- Comparación crítica de las ventajas y desventajas del enfoque basado en DQN frente a los métodos tradicionales en escenarios realistas

F. Métricas de evaluación

Para la evaluación del desempeño del modelo se utilizan métricas cuantitativas clave, las cuales se eligieron por su capacidad para mostrar de manera precisa la eficiencia y adaptabilidad del modelo en condiciones realistas. Estas métricas permiten comparar el esquema propuesto con algoritmos tradicionales como ECMP y DRILL, en igualdad de condiciones. A continuación, se detallan.

- FCT: Es una de las métricas fundamentales del sistema. Mide el tiempo que tarda un flujo desde que es generado hasta completarse su transmisión. Se realiza un cálculo por cada flujo y luego se obtiene el promedio, analizando su desarrollo por cada episodio. Tiene como objetivo el minimizar el FCT promedio y disminuir la dispersión para garantizar eficiencia y estabilidad del sistema. Se extraen valores por episodio y se generan curvas de aprendizaje para comparar entre algoritmos.

- Desviación estándar del uso de enlaces: Indica qué tan balanceada está la carga entre los enlaces. Una desviación alta significa que ciertos enlaces están saturados mientras que otros están subutilizados, creando cuellos de botella y baja eficiencia. Tiene como objetivo minimizar la desviación estándar para garantizar una distribución equitativa del tráfico. La desviación estándar alimenta uno de los términos de la función de recompensa, junto con penalizaciones por enlaces saturados.
- Rendimiento comparativo entre algoritmos: Se hace una comparación de las métricas mencionadas del modelo DQN con las de ECMP y DRILL bajo los mismos escenarios de entrada. Tiene como objetivo evaluar si el modelo basado en IA alcanza un desempeño superior o más estable que los algoritmos tradicionales. Se realizan gráficas comparativas, análisis de tendencias y discusión cualitativa por escenario.

Además de las métricas principales utilizadas para comparar el desempeño de DQN, ECMP y DRILL, el sistema también registra un conjunto más amplio de indicadores internos generados durante la simulación. Estas métricas adicionales desempeñan una función de validación y diagnóstico —permiten verificar la consistencia del entorno, la completitud de los flujos procesados y el uso efectivo de la topología—, mas no forman parte del análisis comparativo entre modelos, debido a que no reflejan directamente los objetivos operativos de la gestión de tráfico. En consecuencia, se las incluye únicamente como información complementaria, mientras que la comparación entre algoritmos se enfoca en métricas que describen de manera directa la eficiencia en la selección de rutas y la distribución del tráfico en la red. Las métricas adicionales son:

- Enlaces sin uso: Mide cuántos enlaces entre switches no fueron utilizados durante la simulación. Un número alto puede indicar rutas poco eficientes o subutilización de la topología, mientras que un uso más amplio de los enlaces refleja una mejor dispersión del tráfico.
- Pasos totales de simulación: Indica el tiempo necesario para que todos los flujos finalicen bajo cada algoritmo. Diferencias significativas pueden revelar congestión, acumulación prolongada de flujos o variaciones en la velocidad de drenaje del tráfico.
- Promedio de flujos por enlace: Se registran dos indicadores, el promedio global de flujos que utilizan cada enlace y el promedio considerando únicamente los enlaces activos. Estas métricas permiten describir la presión de carga sobre la topología y

caracterizar la estructura del tráfico, complementando la métrica de desviación estándar del uso.

- Flujos utilizados en las métricas: Tanto el cálculo de FCT como el de balanceo registra cuántos flujos fueron realmente considerados dentro del rango temporal analizado. Esto asegura la consistencia estadística del análisis y permite verificar la completitud y comparabilidad entre modelos y escenarios.

VII. DESARROLLO DEL SISTEMA PROPUESTO

A. Arquitectura del sistema

El sistema propuesto (Fig. 4) proporciona una plataforma de simulación para evaluar y comparar esquemas de balanceo de tráfico en centros de datos con topología FatTree. Para este trabajo en particular, se lo realiza entre un enfoque basado en aprendizaje por refuerzo, contra dos métodos tradicionales de referencia (ECMP y DRILL).

Con respecto al alcance y contribución, el presente trabajo no plantea un nuevo algoritmo base de aprendizaje por refuerzo, sino una arquitectura experimental propia para adaptar y articular técnicas existentes de DQN en un entorno de balanceo de tráfico en centros de datos. La contribución algorítmica se enfoca en la formulación del problema de selección de rutas como un mecanismo de decisión secuencial, la definición del espacio de estados considerando el uso de enlaces, la demanda acumulada y presencia de flujos *elephant*, la creación del espacio de acciones por medio de rutas mínimas precalculadas, y el diseño de una función de recompensa enfocada en penalizar la congestión y el desbalance. Además, se puso en marcha un entorno de simulación propio en Python para la comparación de DQN, ECMP y DRILL en condiciones homogéneas de tráfico, topología y métricas. En cambio, varios de sus componentes: la arquitectura Dueling DQN, Double DQN, PER, el optimizador Adam y las funciones de pérdida utilizadas, equivalen a técnicas y herramientas existentes en la literatura, que se integraron y adaptaron al problema específico de balanceo dinámico del tráfico en topologías FatTree.

Se diseñó en forma modular, con el objetivo de permitir la experimentación controlada, reproducibilidad de resultados y la comparación directa entre distintos mecanismos de balanceo de tráfico. Cada módulo cubre una etapa específica del proceso: el entorno de simulación y modelado de la red, gestión del tráfico, el entrenamiento del agente basado en aprendizaje por refuerzo, simulación operativa de cada modelo y su análisis estadístico y comparativo del desempeño.

B. Entorno de simulación

El entorno de simulación es el corazón del sistema propuesto, debido a que replica la topología FatTree y la forma en como los flujos de transmiten bajo distintas políticas de selección de rutas. Este módulo fue implementado en Python y diseñado para ofrecer un comportamiento determinista, reproducible y común para todos los modelos evaluados (DQN, ECMP y DRILL), garantizando las mismas condiciones experimentales. Este entorno comprende:

Modelado de la red FatTree: En el simulador se implementa una topología FatTree que la conforman 16 hosts y 20 switches. Los switches están etiquetados alfabéticamente (A–T) en niveles funcionales (de borde, agregación y núcleo), y los hosts se representan como nodos terminales (Z1–Z16) conectados en pares a los switches de borde. Esta topología conserva las propiedades clave de las FatTree utilizadas en centros de datos: rutas múltiples, simetría y redundancia de rutas. Cada enlace de la red se representa de forma explícita como un par (u, v), y aunque se indexan todos los enlaces (incluidos hosts–switch), solo los enlaces entre switches participan en el balanceo y se aíslan mediante una máscara interna. En la inicialización del entorno, todos los enlaces se almacenan en una lista ordenada y se les asignan un índice entero único, generando una estructura link_indices_sorted que sirve como una referencia global. Además, se crean diccionarios auxiliares que mapean cada switch con el conjunto de enlaces incidentes (en forma de lista y en forma de conjunto), para facilitar las operaciones de consulta en tiempo de simulación. La estructura indexada permite que las políticas de selección de rutas (incluido el agente DQN) trabajen directamente sobre acciones específicas como índices de enlace, evitando búsquedas costosas en tiempo de ejecución, y permite el cálculo eficiente de métricas de uso y balanceo, debido a que el estado de cada enlace puede actualizarse y consultarse mediante su índice. De esta manera, el modelado de la red combina una descripción lógica de la topología FatTree con una estructura de datos optimizada que haga eficiente la simulación de los distintos escenarios y el entrenamiento del modelo de aprendizaje automático.

Pre-cálculo de rutas: El entorno emplea rutas precalculadas para determinar las rutas disponibles entre cualquier par de hosts. Estas rutas se calculan previamente mediante un algoritmo de *k-shortest paths* sobre la topología FatTree K=4, la cual es completamente estática en todas las simulaciones. El algoritmo solo obtiene las rutas mínimas entre cada par origen–destino, basados exclusivamente en la estructura del grafo; es decir, sin considerar información dinámica como uso de enlaces, congestión o tráfico en curso. El resultado se almacena en un archivo externo (rutas_precalculadas.pkl) que contiene un diccionario con las rutas óptimas para todos los pares de hosts. De esta manera, los modelos evaluados (DQN, ECMP y DRILL) utilizan exactamente el mismo conjunto fijo, consistente y reproducible de rutas mínimas, garantizando una comparación equitativa entre métodos. En el archivo externo, cada ruta ya está codificada como una lista de índices de enlace. El entorno únicamente las reorganiza y valida, sin necesidad de reconstruir los saltos entre switches. Este proceso crea la estructura ruta_indices_dict, que mapea cada par (src,

dst) con la lista de rutas correspondientes, expresadas como listas de índices de enlaces. Este formato es necesario para permitir que: las políticas de asignación de rutas seleccionen sus acciones directamente sobre la base de índices de enlace, el entorno pueda contabilizar con precisión el uso real de cada enlace, y que las métricas internas de balanceo, puntos de congestión y utilización se procesen sin necesidad de cálculos adicionales. Como parte del proceso de construcción, el entorno asegura que todos los enlaces de las rutas precalculadas correspondan a elementos válidos en el espacio de acciones del agente DQN y que la indexación sea coherente con la estructura link_indices_sorted. Esta validación asegura que la topología modelada, el conjunto de rutas disponibles y las decisiones de selección de rutas que los algoritmos puedan tomar, estén plenamente sincronizados. De este modo, el sistema se beneficia de un conjunto de rutas consistente y previamente filtrado, minimizando el costo computacional y garantizando que la comparación entre los modelos de balanceo se realice en condiciones rigurosamente equivalentes.

Dinámica de transmisión de flujos: El entorno ejecuta la transmisión del tráfico por medio de una simulación discreta por pasos, en la que cada flujo recorre los enlaces asignados según la política de selección de rutas. Esta dinámica simula el comportamiento realista de la competencia por los recursos en una red rutas múltiples y es el mecanismo operativo que determina los FCT, la congestión y el balanceo de tráfico resultante. Para la escala temporal de la simulación, se opera con pasos discretos, en donde cada paso corresponde a un intervalo fijo de tiempo predefinido. Esta relación de conversión entre pasos de simulación y tiempo real se expresa como:

$$t_{real} = step \times tiempo\_por\_step$$

en donde: $tiempo\_por\_step = 0.00001\ seg\ (10\ \mu s)$. De esta forma, los diferentes eventos de la simulación y las variables de medición se pueden expresar de forma coherente en unidades de tiempo físico, garantizando consistencia en la temporalidad durante el análisis comparativo. En cada paso, el entorno realiza las siguientes operaciones:

- Identificación de flujos activos. Se recorren todos los flujos que han llegado (cuya marca de tiempo de inicio sea menor o igual al paso actual) y que aún no han terminado. Cada flujo activo guarda su tamaño restante y la ruta asignada (en forma de secuencia de índices de enlace). Esta información se usa para determinar que enlaces están involucrados en la transmisión durante cada paso en la simulación.
- Cálculo de congestión por enlace. Para todos los enlaces que se encuentren en al menos una ruta activa, en el entorno se calcula cuántos flujos lo atraviesan en este paso. Este

valor define el grado de competencia por capacidad y conforma la base para el cálculo del uso real transmitido por cada enlace y la desviación estándar de uso entre enlaces registrado en cada paso.

- Distribución proporcional de capacidad. La capacidad efectiva que recibe cada flujo se calcula dividiendo la capacidad total del enlace entre el número de flujos concurrentes que lo utilizan. Para un flujo que recorre varios enlaces, la capacidad final asignada corresponde al cuello de botella, es decir, al mínimo de las capacidades efectivas de los enlaces en su ruta. Este mecanismo reproduce un comportamiento real: un enlace saturado limita la velocidad del flujo extremo a extremo.
- Avance del tamaño restante del flujo. Cada flujo disminuye su tamaño restante de acuerdo con la capacidad efectiva que recibe en ese paso. Cuando el tamaño llega a cero, el flujo se marca como completado y se registra su tiempo final.
- Finalización dinámica de la simulación. La simulación seguirá ejecutándose hasta que todos los flujos generados hayan sido completamente transmitidos, aunque el generador dejare de producir nuevas llegadas. En los modelos ECMP y DRILL, el entorno puede extender automáticamente la simulación si todavía quedan flujos pendientes, asegurando que la evaluación se realice sobre la carga completa.
- Registro de estados internos. En cada paso, el entorno actualiza estructuras internas fundamentales para el análisis posterior como el uso real por enlace (kilobytes transmitidos) o el balanceo por paso (desviación estándar del uso real de todos los enlaces).

Métricas internas del entorno: El entorno va almacenando en cada paso de la simulación y al final de la transmisión de los flujos, un conjunto de métricas internas que permiten analizar en detalle el comportamiento de la red en diferentes políticas de selección de rutas. Estas métricas son necesarias para el análisis comparativo posterior, al reflejar tanto el desempeño global del sistema como la eficiencia con la que cada algoritmo reparte los flujos en la infraestructura. Las métricas utilizadas son:

- Uso real por enlace. El entorno lleva un contador acumulado de la cantidad total de datos (en kilobytes) que hayan sido transmitidos por cada enlace durante toda la simulación.

$$u_e(t) = uso\ real\ del\ enlace\ e\ en\ el\ paso\ t\ [kB]$$

Esta métrica permite identificar patrones de congestión, mostrar los enlaces que participaron activamente en la transmisión y detectar enlaces sin utilización. La variable uso_real_por_enlace se actualiza en cada paso para todos los enlaces involucrados.

- Balanceo de tráfico (desviación estándar por paso). Para cada paso, el entorno calcula la desviación estándar del uso real de todos los enlaces activos:

$$B(t) = \sqrt{\frac{1}{|E_t|}\sum_{e \in E_t}\left(u_e(t) - \mu(t)\right)^2}$$

donde:

$$B(t) = \sqrt{\frac{1}{|E_t|}\sum_{e \in E_t}\left(u_e(t) - \mu(t)\right)^2}$$

Este indicador, almacenado en balanceo_por_step_real, refleja que tan desigual se utiliza la red: la presencia de enlaces mucho más cargados que otros, y la calidad de la política de colocación de rutas en relación con la dispersión del flujo. Un valor bajo indica un sistema equilibrado; un valor alto revela congestión localizada.

- Enlaces sin uso. Al terminar la simulación, el entorno guarda en enlaces_sin_uso todos los enlaces que no registraron ninguna transmisión. Esta información es evidencia para conocer la capacidad ociosa de la topología, el grado en que una política aprovecha la estructura de rutas múltiples y que tan disperso es el tráfico. Una política eficiente debería minimizar la cantidad de enlaces sin uso, de manera especial en topologías de rutas múltiples como FatTree.
- Flujos activos por paso. El entorno registra cuántos flujos permanecen activos en cada paso, lo que permite evaluar el comportamiento temporal del sistema bajo condiciones de carga. Esta métrica refleja la progresión de la congestión, permite visualizar la disminución del número de flujos en el tiempo y ayuda a interpretar anomalías en los FCT.

C. Gestión del tráfico

El módulo de generación de tráfico crea trazas sintéticas para representar cargas realistas en un centro de datos. En él se pueden ajustar parámetros como la intensidad de llegada (λ) o la proporción *mice*/*elephant* para el tamaño de los flujos. Todos los flujos se guardan en archivos .pkl

reutilizables, para que DQN, ECMP y DRILL sean evaluados bajo las mismas condiciones experimentales.

Modelo de llegada: La generación de tráfico implementa un modelo híbrido Poisson/ráfagas para simular tráfico estocástico realista en redes de rutas múltiples. Su funcionamiento incluye las siguientes características:

- Distribución Poisson para los tiempos entre llegadas. Este mecanismo agrega jitter natural e inyecta tráfico con un ritmo controlado por el parámetro λ. Se generan mediante una distribución exponencial:

$$\Delta t \sim Exponencial\left(\frac{1}{\lambda}\right)$$

- Patrón en ráfagas (ciclos ON/OFF). El generador contiene periodos ON/OFF en forma de ráfagas. Durante el periodo ON, las llegadas siguen el proceso Poisson; en el periodo OFF, no se genera flujo. Esto simula condiciones de tráfico típicas en centros de datos, con fases de saturación seguidas de periodos de latencia.
- Flujos *mice*/*elephant*. Los *mice* representan la mayoría del tráfico y tienen tamaños reducidos, mientras que los *elephants* son menos frecuentes, pero mucho más grandes.

Estructura del archivo de flujos (.pkl): Cada archivo .pkl es una lista de flujos, en donde cada flujo se representa con exactamente con tres elementos: Tupla con host origen y destino, Tamaño total del flujo en kilobytes y paso dentro de la simulación en el que inicia el flujo.

Reproducibilidad: Para hacer totalmente reproducible las trazas del tráfico, el generador inicializa explícitamente los generadores de números aleatorios np.random.seed(seed) y random.seed(seed), los cuales son necesarios debido a que ambos tienen motores de aleatoriedad separados; el primero utilizado para los tiempos de llegadas de los flujos, mientras que el segundo se utiliza para decisiones probabilísticas, como la clasificación *mice*/*elephant*.

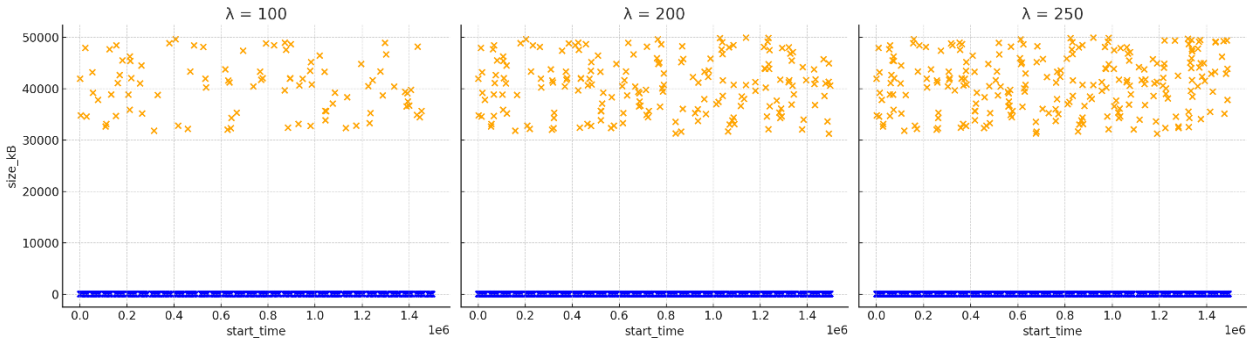

Fig. 6 Distribución de flujos (semilla 101)

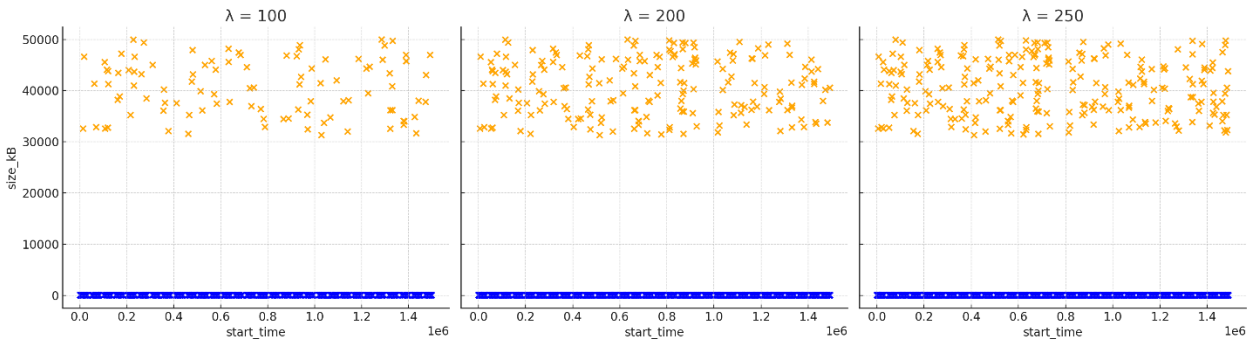

Fig. 7 Distribución de flujos (semilla 102)

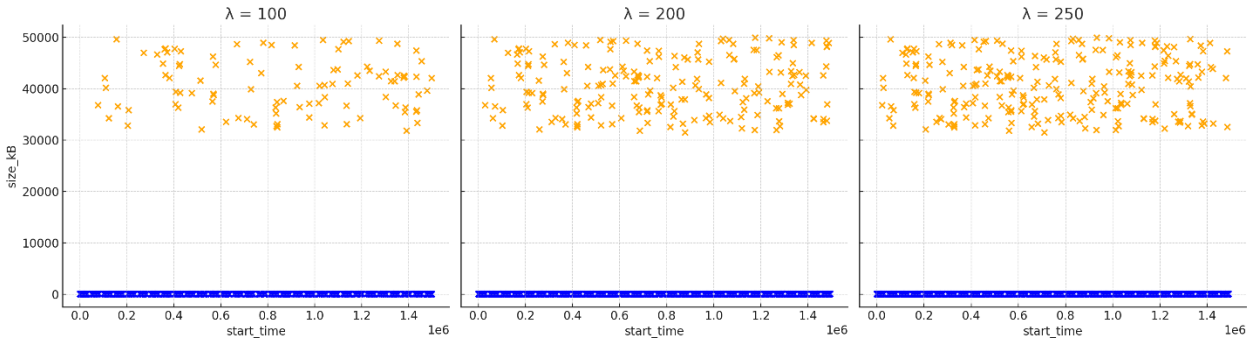

Fig. 8 Distribución de flujos (semilla 103)

En las Figs. 6, 7 y 8 se muestran ejemplos representativos de los archivos de flujos generados por el módulo de tráfico para los valores de λ = 100, 200 y 250, usando tres semillas (*seeds*) distintas (101, 102 y 103). Cada punto corresponde a un flujo individual, caracterizado por su tiempo de arribo (start_time) y su tamaño en kilobytes (size_kB), permitiendo visualizar simultáneamente la distribución temporal, la intensidad del tráfico y la composición *mice* y *elephant*. Los flujos *mice* (representados en azul) se concentran en la región inferior debido a sus tamaños reducidos, mientras que los flujos *elephant* (en naranja) presentan tamaños significativamente mayores y dominan las zonas superiores de cada subgráfico. Las diferencias entre columnas reflejan el incremento de la carga al aumentar el valor de λ: a mayor λ, mayor densidad de puntos y menor separación entre tiempos de llegada, lo que evidencia un patrón de tráfico más intenso y con mayor probabilidad de superposición entre flujos. Asimismo, las variaciones entre filas muestran la naturaleza estocástica del generador: aun cuando se mantiene fijo el valor de λ, el uso de distintas semillas produce secuencias diferentes de tiempos de llegada, tamaños y mezcla de flujos, preservando la reproducibilidad experimental sin perder diversidad en los escenarios evaluados. Estas visualizaciones permiten observar de manera clara el comportamiento del generador en términos de aleatoriedad controlada, presencia de ráfagas y

coexistencia de flujos *mice* y *elephant*, lo cual es fundamental para analizar el rendimiento de los algoritmos de selección de rutas bajo condiciones de tráfico realistas y variables. Al emplear múltiples semillas para cada nivel de carga, se garantiza que las evaluaciones del modelo DQN, ECMP y DRILL se realicen sobre escenarios heterogéneos pero comparables, fortaleciendo la validez estadística del análisis comparativo.

D. Aprendizaje por refuerzo

El módulo de aprendizaje por refuerzo es el núcleo inteligente del sistema propuesto. Su objetivo es aprender, por medio de la interacción con el entorno FatTree, una política de selección de rutas que minimice la congestión mejore el balanceo del tráfico y reduzcan los FCT. A continuación, se describen los componentes que el diseño del módulo contempla:

Representación del estado: La representación del estado es una de las partes más importantes del módulo de aprendizaje por refuerzo, puesto que define la información de la red que está disponible para que el agente DQN tome decisiones de selección de rutas. El entorno hace una captura del estado operativo de los enlaces entre switches, con el objetivo de que el agente aprenda patrones relevantes de congestión, distribución del tráfico y demanda acumulada en la red. El estado se genera internamente en cada paso de simulación, donde el entorno reconstruye el vector de estado con:

- Uso real por enlace (uso_sw). Representa el uso real normalizado de los enlaces entre switches. En cada paso, el entorno calcula el tráfico total transmitido por cada enlace en kilobytes, se lo divide por la capacidad disponible por paso y se lo normaliza al rango [0,1]. El entorno filtra los enlaces host–switch y solo incluye enlaces entre switches en el estado, utilizando la lista preprocesada de índices válidos. Esta métrica refleja la saturación instantánea de cada enlace y constituye un indicativo del estado de congestión de la red.
- Demanda acumulada por enlace (demanda_norm). Almacena la demanda acumulada de flujos activos por cada enlace normalizada en el entorno. Esta métrica no considera el tráfico transmitido, sino la cantidad de demanda “pendiente” o “en curso” asociada a los flujos activos que están utilizando ese enlace. De esta forma, el estado combina congestión efectiva y demanda latente, permitiendo al agente anticipar saturaciones futuras.

- Indicador elephant-flow (flag_elefante). Para cada enlace, el entorno agrega una métrica adicional en donde cada posición contiene un valor binario de 1 si ese enlace es parte de al menos un flujo de tipo *elephant* (≥ 1 MB) o 0 en caso contrario. Esta métrica permite al agente distinguir entre enlaces cargados por flujos *mice* o *elephant*, cuya persistencia tiende a generar congestión prolongada.

  Las tres métricas se combinan en un único vector que refleja de manera compacta y detallada la congestión instantánea, la demanda acumulada sobre los enlaces, la presencia de tráfico pesado y la estructura intrínseca de la topología FatTree. Este diseño del estado presenta ventajas como mayor visibilidad (el agente conoce de forma explícita la carga actual de la red), compatibilidad con redes de rutas múltiples (soporta la saturación selectiva de enlaces) y dimensión estática (aunque el número de rutas posibles cambia por flujo, el estado siempre es del mismo tamaño).

$$state = [uso\_sw, demanda\_norm, flag\_elefante]$$

Espacio de acciones: se compone por todas las rutas completas (desde el switch de origen hasta el switch de destino) que fueron precalculadas para cada par de hosts y almacenadas en la estructura ruta_indices_dict. Cada acción corresponde a la elección de un identificador global de ruta, que se traduce en un conjunto específico de enlaces que conforman la ruta elegida. Este diseño permite al agente tomar decisiones no sobre saltos individuales, sino sobre rutas de extremo a extremo, capturando de manera directa la selección de rutas de la topología FatTree.

- Rutas precalculadas como acciones. Antes del entrenamiento, el entorno carga el archivo externo rutas_precalculadas.pkl que contiene para cada par (src, dst) todas las rutas mínimas disponibles expresadas como listas de índices de enlaces globales y empaquetadas con un identificador único id_global. Cada id_global es el índice de acción que el agente selecciona. De esta forma opera exclusivamente sobre rutas que el entorno reconoce como válidas, lo que garantiza consistencia entre DQN, ECMP y DRILL.

$$ruta\_indices\_dict[(src, dst)] = \begin{bmatrix} (id\_global_1, [e_1, e_2, \dots., e_k]), \\ (id\_global_2, [e_8, e_4, \dots., e_{12}]), \\ \dots \end{bmatrix}$$

- Mapa global de acciones accesible al agente. El script de entrenamiento construye una lista explícita self.training_flows_routes que contiene todos los identificadores globales de todas las rutas de entrenamiento. Entonces el espacio de acciones action_space del

agente considera que el agente tiene un número fijo y global de acciones posibles, cada acción corresponde a un id_global de una ruta más corta precalculada y no se generan rutas nuevas durante el entrenamiento.

- Selección de acciones durante la simulación. Cuando un flujo llega al entorno, este obtiene la lista de rutas válidas actions_validas para ese flujo y contiene únicamente los id_global que aplican al par de hosts correspondiente. Si el agente selecciona una acción a, el entorno verifica si pertenece a actions_validas, la acción es válida y se asigna la ruta; si a no pertenece a las rutas válidas, pero existen rutas disponibles, se aplica una penalización por acción inválida, mientras que, si no existen rutas disponibles para ese par, el entorno registra un "par sin rutas" y también penaliza. Este proceso obliga al agente a aprender a elegir rutas adecuadas, evitando comportamientos incoherentes o inconsistentes con la topología.

La elección de un espacio de acciones basado en rutas globales ofrece ventajas como:

- Evita decisiones inconsistentes. El agente nunca elige un tramo de la ruta de forma aislada sino una ruta completa válida, asegurando rutas físicamente factibles y topológicamente consistentes.
- Comparabilidad directa con ECMP y DRILL. Los tres algoritmos utilizan exactamente el mismo conjunto de rutas más cortas. La única diferencia entre ellos radica en el criterio de cada uno para seleccionar rutas dentro de ese conjunto.
- Simplificación computacional. Como la topología FatTree con K=4 tiene un número relativamente bajo de rutas más cortas por par (2 y 4), el número total de acciones globales es manejable incluso para un agente con Dueling DQN y PER.
- Flexibilidad para el análisis posterior. El registro de acciones inválidas, rutas omitidas y sesgos de selección puede utilizarse para comparar el estilo de selección de rutas de DQN con ECMP y DRILL.

Arquitectura del agente DQN: El módulo de aprendizaje profundo implementa un agente basado en DQN que integra tres mecanismos ampliamente reconocidos por mejorar la estabilidad y eficiencia del aprendizaje en entornos complejos. En primer lugar, Dueling DQN separa la estimación del valor del estado y la ventaja de cada acción, lo que acelera el aprendizaje en situaciones donde algunas acciones no producen cambios inmediatos en la recompensa [66]. En segundo lugar, Double DQN reduce la sobreestimación de los valores Q ocasionada por la

selección y evaluación de acciones con la misma red neuronal, mejorando la precisión del proceso de actualización [67]. Finalmente, el uso de PER permite al agente aprender más rápido al enfocar las actualizaciones en transiciones con mayor TD, lo que incrementa la eficiencia de los datos de entrenamiento y acelera la convergencia [68]. Estas características son especialmente relevantes en un entorno como la topología FatTree, donde el agente enfrenta estados altamente no estacionarios, variabilidad en la congestión y múltiples rutas alternativas por flujo. La combinación de Dueling, Double y PER permite mejorar la estabilidad del aprendizaje y obtener políticas de selección de rutas más robustas frente a fluctuaciones en el tráfico y condiciones dinámicas de la red.

- Arquitectura Dueling. El agente utiliza una arquitectura dueling, en donde la red neuronal se divide internamente en dos ramas, $V(s)$ que calcula el valor global del estado y $A(s,a)$ que estima la ventaja relativa de cada acción. Estas dos ramas se combinan para producir los valores Q finales lo que le permite al agente distinguir entre qué tan bueno es el estado, incluso si la acción no tiene mucho impacto y qué acción es mejor dentro de ese estado. Es especialmente útil en el balanceo de tráfico, donde muchos estados son críticos o congestionados independientemente de la acción específica tomada.

$$Q(s,a) = V(s) + \left(A(s,a) - \frac{1}{|A|} \sum_{a'} A(s,a') \right)$$

- Double DQN. El algoritmo utiliza mecanismos de Double DQN en la reducción de la sobreestimación de valores Q. Para ello la red online selecciona la acción (a^*) con mayor valor Q en el estado siguiente y la red objetivo evalúa esa acción para calcular el valor objetivo (y). De esta forma, la selección y la evaluación se desacoplan, evitando sesgos optimistas y mejorando la estabilidad del aprendizaje. Los valores objetivos se ajustan a un rango acotado ([-20,20]) para estabilizar el entrenamiento.

$$a^* = argmax_a Q_{online}(s', a)$$

$$y = r + \gamma Q_{target}(s', a^*)$$

- PER. El agente utiliza un búfer de experiencias con prioridades basadas en TD:

$p_i \propto (|\delta_i| + \varepsilon)^{\propto}$.

Durante la optimización las transiciones se muestrean con probabilidad proporcional a su prioridad y se aplican los pesos de muestreo por importancia ω_i para corregir el sesgo

introducido por el muestreo no uniforme. La pérdida ponderada utilizada en el entrenamiento es:

$$L = \omega_i \cdot SmoothL1(Q_{pred}, y_i)$$

Después de cada actualización, las prioridades del búfer PER se recalculan utilizando los TD obtenidos en el replay() del agente. Esto mantiene el búfer priorizado sincronizado con las transiciones más informativas para el aprendizaje.

- Función de pérdida, regularización y recorte de gradiente. La actualización de los parámetros del agente se realiza utilizando la pérdida Smooth L1 (Huber Loss), seleccionada por su robustez frente a valores atípicos en el TD especialmente relevante cuando la recompensa presenta variabilidad por picos de congestión). La actualización incluye regularización L2 ligera y recorte de gradiente (*clipping*) con norma máxima 1.0 para evitar explosión de gradientes.
- Actualización suave de la red objetivo. Después de cada paso de optimización, se actualiza la red objetivo mediante una actualización suave con un valor de τ pequeño ($\approx 10^{-3}$). Este mecanismo evita oscilaciones bruscas y estabiliza la convergencia.

$$\theta_{target} \leftarrow \tau\theta_{online} + (1-\tau)\theta_{target}$$

- Arquitectura interna de la red neuronal. La red utilizada por el agente presenta capas totalmente conectadas (*fully connected*), dos capas ocultas con activación ReLU, tamaño de entrada = dimensión del vector de estado (uso_sw, demanda_norm, flag_elefante) y tamaño de salida = número global de rutas precalculadas (action_size). Esta estructura ofrece un balance adecuado entre capacidad de representación y eficiencia computacional.
- Frecuencia y dinámica de actualización de la política. La política del agente se actualiza mediante un mecanismo dinámico que depende tanto del estado del búfer PER como de la carga del episodio, garantizando estabilidad y eficiencia durante el aprendizaje. El proceso incluye una fase de calentamiento (*warm-up*): el entrenamiento solo inicia cuando el agente ha acumulado suficiente experiencia y ha superado los episodios iniciales de calentamiento, evitando actualizaciones prematuras que producirían inestabilidad; Capacidad dinámica (upd_cap): la cantidad de actualizaciones por paso se ajusta automáticamente según el tamaño del búfer PER (a mayor cantidad de experiencias almacenadas, mayor capacidad de actualización); Límite dinámico de

actualizaciones por episodio (dyn_cap): el cual el agente limita el número total de actualizaciones por episodio según la carga real del tráfico simulado, evitando sobre entrenamiento y manteniendo la proporcionalidad con las transiciones generadas; Bucle de entrenamiento: en cada episodio, el agente ejecuta únicamente las actualizaciones permitidas por los límites dinámicos, aplicando replay mientras haya capacidad disponible y deteniéndose cuando se alcanza el máximo permitido.

- Integración con el entorno. La red recibe el estado completo del entorno (el uso real de enlaces, la demanda acumulada, la presencia de flujos *elephant*). La salida de la red determina los valores Q de todas las rutas globales precalculadas. Durante el entrenamiento, el modelo ajusta estos valores de Q utilizando recompensas globales del entorno, recompensas locales por flujo, penalizaciones por acciones inválidas y la evolución temporal de la congestión. Esta arquitectura permite que el agente aprenda patrones complejos de congestión y seleccione rutas de forma adaptativa.

Función de recompensa: La función de recompensa es la forma en la que el entorno le indica al agente DQN que aprenda una política de selección de rutas que minimice la congestión, evite puntos de congestión y distribuya el tráfico de manera más uniforme en la topología FatTree. En esta implementación, la recompensa consta de dos partes complementarias: recompensa global generada por el entorno (evaluación del estado de la red en el paso) y recompensa local ajustada en el script de entrenamiento, dependiente del flujo y de la ruta seleccionada. La recompensa final del agente es una combinación de ambos valores, diseñada para equilibrar la estabilidad de la señal global con la sensibilidad local de la decisión que se ha tomado para cada al flujo.

- Recompensa global. En cada paso de simulación (env.step( )), el entorno implementa una recompensa basada en cuatro métricas internas:

$$r_{env} = -\omega_{bal} \cdot balanceo - \omega_{hot} \cdot penalizacion_{hotspots} + penalizacion_{pico} - penalizacion_{accion}$$

recortada en el rango $r_{env} \in [-1,1]$ con el propósito de mantener la estabilidad del entrenamiento. En la penalización por desbalanceo ($\omega_{bal} \cdot balanceo$) el entorno calcula el uso de los enlaces switch–switch y obtiene su desviación estándar, siendo un valor alto de balanceo un indicativo de que uno o pocos enlaces soportan carga desproporcionada.

Durante la configuración de los experimentos, tanto para el entrenamiento como para la evaluación del modelo DQN, se utilizaron los pesos:

$$\omega_{bal} = 4.0, \omega_{hot} = 2.4$$

Para la penalización debido a puntos de congestión $(\omega_{hot} \cdot penalizacion_{hotspots})$, se considera como punto de congestión a aquel enlace cuyo umbral de uso supera el valor definido en θ_{hot}. Dicho umbral se puede parametrizar durante la implementación, permitiendo que la sensibilidad del entorno se ajuste en presencial de enlaces con alta utilización. La penalización se calcula como:

$$penalizacion_{hotspots} = mean(uso > \theta_{hot})$$

Experimentalmente el umbral se definió como:

$$\theta_{hot} = 0.80$$

Para la penalización por picos de utilización, se toma en cuenta el uso máximo entre todos los enlaces switch-switch y se penaliza al sobrepasar el umbral θ_{hot}. Esta penalización se calcula como:

$$penalizacion_{pico} = -\beta \cdot \frac{max(0, max\_uso - \theta_{hot})}{1 - \theta_{hot}}$$

donde:

$$\beta = 1.8$$

En suma, la penalización por acciones o rutas no válidas $(penalizacion_{accion})$ se emplea en el caso de que en el entorno se detecte que la acción seleccionada no forme parte de una ruta disponible para el par origen-destino del flujo, o cuando la ruta no esté conformada por enlaces válidos entre switches. Esto trabaja como un mecanismo de validación del entorno con el fin de conservar la coherencia de la simulación:

$$penalizacion_{accion} += 5.0$$

- Recompensa local. Se agrega un componente adicional de recompensa para aumentar el valor asociado a cada flujo activo. En este punto se evalúa únicamente los enlaces de la ruta seleccionada lo que permite ser más específico con la decisión de enrutamiento tomada para el flujo. La recompensa local se define como:

$$r_{local} = -\omega_{bal} \cdot std_{ruta} - \omega_{hot} \cdot hotspots_{ruta} - L_{pair} \cdot skew$$

donde:

$$\omega_{bal} = 4.0, \omega_{hot} = 2.4, L_{pair} = 0.020$$

El balanceo local de la ruta (std_{ruta}) se obtiene por medio de la desviación estándar del uso real en los enlaces de la ruta seleccionada:

$$std_{ruta} = std(uso_{ruta})$$

Los puntos de congestión locales ($hotspots_{ruta}$) muestran la proporción de enlaces de la ruta cuya utilización supera el umbral de congestión definido:

$$hotspots_{ruta} = \frac{1}{|\ r\ |} \sum_{e \in r} \mathbf{1}(uso_e > \theta_{hot})$$

donde, en la configuración experimental:

$$\theta_{hot} = 0.80$$

La penalización por sesgo entre rutas $(L_{pair} \cdot skew)$ es utilizada en el caso de que para un mismo par origen-destino y en un mismo paso de simulación, el agente enfoque sus decisiones en una sola ruta equivalente. Debido a esto se obtiene la desviación estándar de la distribución de rutas seleccionadas:

$$skew = \sigma(p_1, p_2, \dots, p_k)$$

Luego, la recompensa local se recorta con el fin de evitar valores extremos:

$$r_{local} = clip(r_{local}, -CLIP_R, CLIP_R)$$

donde:

$$CLIP_R = 3.0$$

Posteriormente, la recompensa local se escala mediante:

$$r_{local} = REWARD\_SCALE \cdot r_{local}$$

con:

$$REWARD\_SCALE = 0.030$$

Con esto se evita que el valor de la recompensa local prevalezca sobre el valor global del entorno.

- Combinación global y local de la recompensa. La recompensa final combina la recompensa global del entorno y la recompensa local escalada:

$$r = clip(r_{env} + \eta \cdot r_{local}, -1{,}1)$$

donde r_{env} almacena la información del estado global de congestión del sistema, r_{local} almacena la información sobre la calidad de la ruta empleada por el flujo activado y η regula la influencia de la recompensa local con respecto a la global. En los experimentos:

$$\eta = R_{ENV\_MIX} = 0.55$$

Proceso de aprendizaje y actualización de la política: combina la interacción del agente con el entorno, donde en cada paso se toma una acción por cada flujo que inicia, la asignación de créditos por flujo, y la actualización de la política mediante Double Dueling DQN con PER. El entrenamiento ocurre episodio por episodio, utilizando flujos sintéticos generados con semillas controladas y parámetros variables. A continuación, se describen las partes que conforman el procedimiento completo:

- Selección de acciones. En cada paso de la simulación, el agente identifica el conjunto de flujos que requieren decisión, esto es, flujos pendientes (start_time < t), y flujos que arriban exactamente en el instante actual. Para cada flujo se obtiene el conjunto de rutas globales válidas registradas en ruta_indices_dict. La selección se realiza mediante un esquema *epsilon-greedy*, definido en dos etapas: exploración con probabilidad ε se elige una ruta válida al azar, y explotación con probabilidad (1-ε), que evalúa los valores Q y agrega una penalización local:

  $$costo(g) = -Q(s,g) + \alpha_{seq} \cdot carga\_local(g)$$

  Donde Q(s,g) se obtiene de la red neuronal, α_{seq} = 0.05 es el parámetro de penalización secuencial que controla cuánto se incrementa el costo de una ruta cuando ya tiene flujos asignados en el mismo paso, y la carga_local(g) es el número de flujos ya asignados en este mismo paso sobre los enlaces de la ruta. El resultado es un diccionario acciones_dict que contiene una acción por flujo activo del paso.
- Ejecución del entorno y recompensa global. El entorno procesa acciones_dict y devuelve: nuevo estado, recompensa global, indicador y métricas internas. La recompensa global proviene exclusivamente del entorno, calculada con los pesos configurados: desviación estándar del uso (balanceo), puntos de congestión según umbral, la penalización por picos de utilización, penalización por acciones inválidas, recorte final al intervalo [-1, 1].
- Cálculo de recompensas locales por flujo. Por cada flujo activado en ese paso, se calcula una recompensa adicional basada solo en la ruta seleccionada: desviación estándar del uso en los enlaces involucrados, proporción de enlaces de la ruta cuya utilización supera el umbral de congestión definido θ_{hot}, penalización por sesgo entre rutas del mismo par en este paso.

- Combinación global/local de la recompensa. La recompensa final se combina. Esta es la señal que se almacena en el búfer PER.

$$r = clip(r_{env} + \eta \cdot r_{local}, -1{,}1)$$

- Generación de transiciones (créditos por flujo). El entrenamiento utiliza una estrategia de asignación de crédito por flujo (*credit assignment per-flow*): por cada flujo activado en el paso, se genera una transición individual, lo que permite al agente aprender con mucha mayor resolución respecto al impacto que tiene cada ruta seleccionada sobre el estado de la red. Las transiciones se almacenan en el búfer PER.

$$(s, a_{fid}, r_{total}, s', done)$$

- Repetición de experiencias y actualización de la política. Cuando existen suficientes transiciones acumuladas (MIN_REPLAY_START = 800), se ejecutan múltiples actualizaciones por episodio, esto es: Se extrae un minilote (*minibatch*) usando PER con enfriamiento progresivo (*annealing*) del parámetro β, se calcula la pérdida Huber ponderada por los pesos de muestreo por importancia, se hacen actualizaciones mediante gradiente descendente con recorte de gradiente y la red objetivo se actualiza mediante una actualización suave con $\tau = 0.05$.
- Actualización de la red objetivo. Después de cada minilote, se aplica una actualización suave implementando estabilidad temporal sin copias completas salvo al inicio o al cargar un punto de control (*checkpoint*).

$$\theta_{target} \leftarrow \tau\theta_{online} + (1 - \tau)\theta_{target}$$

- Control de exploración y del aprendizaje. El valor de ε decae de forma exponencial por episodio. Cuando se reanuda desde un punto de control, el sistema recalcula ε para mantener consistencia con el episodio global. El optimizador Adam incorpora un scheduler ReduceLROnPlateau, que ajusta la tasa de aprendizaje en función del promedio de la pérdida del episodio, permitiendo una convergencia más fina en etapas avanzadas.

$$\varepsilon \leftarrow max(\varepsilon_{min}, \varepsilon \cdot EPS\_DECAY\_CONST)$$

Validación periódica y puntos de control: En el entrenamiento se establece un mecanismo de control periódico cuyo propósito es monitorear la evolución de la política y prevenir que se deteriore el desempeño. Este proceso se encuentra explícitamente programado en el archivo de

entrenamiento mediante un bloque que se ejecuta cada 10 pasos y que no afecta los gradientes ni el estado del agente. El proceso está constituido por los siguientes elementos:

- Frecuencia y activación de la validación. Se evalúa la política cada 10 pasos de actualización. Esta validación no altera el búfer de experiencia ni la red, y se utiliza exclusivamente para obtener métricas estables.
- Evaluación en modo de prueba (test_mode=True). Permite medir el rendimiento real de la política sin estocasticidad ni ruido de exploración. Durante este modo no hay entrenamiento, no se actualiza el búfer de experiencia, no se seleccionan acciones con exploración ε-greedy (ε = 0), el entorno nunca corta por límite artificial de pasos, sino solo cuando todos los flujos terminan.
- Métricas recolectadas en la validación. se obtienen directamente del entorno y los resultados se guardan en listas acumulativas para análisis posterior. Estas son: recompensa total obtenida, FCT promedio, cociente FCT experimental / FCT teórico, balanceo promedio (desv. estándar del uso de enlaces entre switches), porcentaje de flujos completados y número total de flujos evaluados.
- Almacenamiento de puntos de control. Luego de cada validación, el código compara el desempeño con la mejor métrica obtenida hasta el momento y guarda solo si éste mejora. El punto de control incluye los pesos completos de la red principal Q, permitiendo reproducir entrenamientos, analizar políticas intermedias, ejecutar el modo de pruebas completo.

E. Simulación por modelo

La fase de simulación corresponde a la ejecución comparativa de los tres esquemas de balanceo evaluados en la investigación: DQN, ECMP y DRILL. Cada uno se ejecuta sobre el mismo entorno, con idénticos archivos de flujos y la misma topología. El propósito de esta etapa es generar métricas homogéneas de desempeño para analizar la calidad del balanceo de tráfico bajo condiciones equivalentes. Los tres modelos se ejecutan con los mismos insumos: topología fija cargada desde el entorno, rutas precalculadas en rutas_precalculadas.pkl, archivos de flujos .pkl con parámetros (src, dst, size, start_time) y simulación paso a paso, en donde en cada uno se actualizan transmisión de datos, ocupación de enlaces, conteo de flujos activos, y cálculos de balanceo. Cada modelo procesa todos los archivos de flujos en un ciclo externo, produciendo una salida por archivo.

- Simulación con el modelo DQN (modo prueba): En este modo, la red neuronal opera en modalidad determinista, es decir, sin exploración ni actualización de pesos. El proceso de selección de rutas se produce a partir de los valores Q del modelo entrenado, adicionando una penalización secuencial local por la carga en las rutas candidatas, para evitar que varios flujos activados simultáneamente en el mismo paso se concentren en los mismos enlaces. El procedimiento general se describe en el Algoritmo 1.

Algoritmo 1: Simulación DQN (modo prueba)

```
Data: traffic_flows.pkl, rutas_precalculadas.pkl, modelo_entrenado.pth
Resultado: Resultados_DQN
begin
    // Carga del modelo entrenado
    Q ← load_model(modelo_entrenado.pth)
    // Inicialización del entorno
    env ← init_environment(rutas_precalculadas.pkl)
    t ← 0
    // Activación de flujos según start_time
    activar_flujos(env, traffic_flows)
    while env.existen_flujos_activos( ) do
        S ← env.get_state()
        // Selección de acciones válidas por flujo
        acciones_dict ← ∅
        for f ∈ env.flujos_activos(t) do
            A_validas ← env.obtener_rutas_validas(f)
            // Política determinista: selección por menor costo ajustado
            costo_g←Q(S,g)+α_seq·carga_local(g)
            a_f←argmin_g∈A_validas costo_g
            acciones_dict[f] ← a_f
        end
        // Transmisión del tráfico
        env.calcular_avance_por_enlace(acciones_dict)
        // Actualización de métricas por paso
        registrar(env.uso_real_por_enlace(t))
        registrar(env.balanceo_por_step_real(t))
        registrar(env.FCT_parciales(t))
        t ← t + 1
    end
    Resultados_DQN ← env.exportar_metricas()
end
```

Se carga un archivo .pth correspondiente al modelo entrenado. Al estar en modo prueba no se calculan gradientes, no se actualizan pesos, no se aplica exploración. Se inicializa el entorno de simulación que involucra a la topología FatTree, el vector de uso de enlaces (link_usage), las estructuras internas para almacenar métricas, y se cargan los flujos generados por el módulo de tráfico. Por cada paso, en el entorno se crea el vector de estado desde el uso real normalizado de los enlaces entre switches, la demanda acumulada normalizada y el indicador de presencia de flujos *elephant*. Para cada flujo activo en el paso, el modelo evalúa el valor Q de las rutas válidas y realiza un cálculo de costo ajustado para cada alternativa, a la vez que combina el valor estimado por la red en conjunto con una penalización local secuencial relacionada a la carga que ya ha sido asignada en el mismo paso. Así, la selección se mantiene determinista, sin embargo, no corresponde a un argmax puro, sino más bien a una elección que toma en cuenta el menor costo ajustado. Las acciones elegidas se condensan en un diccionario acciones_dict[f]. El entorno ejecuta el avance del tráfico realizando la transmisión real de datos limitada por la capacidad disponible en cada enlace, actualización del uso acumulado de enlaces, reducción del tamaño restante del flujo, detección del fin del flujo (FCT), incremento del contador de acciones inválidas si la ruta seleccionada no pertenece al conjunto de rutas disponibles. En modo prueba, la recompensa proporcionada por el entorno no se usa para actualizar la política ni para el almacenamiento de experiencias; solo se aplica la política entrenada y se almacenan las métricas de desempeño. Finalmente, el entorno genera, almacena internamente y exporta a un archivo .pkl, en cada paso, con las métricas resultantes.

- Simulación con ECMP: en esta simulación el modelo no aprende ni interactúa con una función de valor; su rol es seleccionar rutas predeterminadas (todas de costo mínimo) según una política fija configurada antes de iniciar la transmisión de cada flujo. Cada flujo puede tener varias rutas de igual costo, y ECMP selecciona una sola ruta para todo el flujo mediante el algoritmo *round_robin*, que consiste en la selección circular determinista por cada par origen–destino. El procedimiento general se describe en el Algoritmo 2.

Algoritmo 2: Simulación ECMP

```
Data: traffic_flows.pkl, rutas_precalculadas.pkl
Resultado: Resultados_ECMP
begin
    // Inicialización del entorno
    env ← init_environment(rutas_precalculadas.pkl)
    t ← 0
    // Activación de flujos según start_time
    activar_flujos(env, traffic_flows)
    while env.existen_flujos_activos( ) do
        // Selección ECMP: ruta aleatoria entre rutas mas cortas válidas
        acciones_dict ← ∅
        for f ∈ env.flujos_activos(t) do
            A_validas ← env.obtener_rutas_validas(f)
            a_f ← choose_round_robin_index(A_validas)
            acciones_dict[f] ← a_f
        end
        // Transmisión del tráfico
        env.calcular_avance_por_enlace(acciones_dict)
        // Actualización de métricas por paso
        registrar(env.uso_real_por_enlace(t))
        registrar(env.balanceo_por_step_real(t))
        registrar(env.FCT_parciales(t))
        t ← t + 1
    end
    Resultados_ECMP ← env.exportar_metricas()
end
```

Para cada flujo se extraen las rutas desde el archivo de flujos. Se obtienen los índices de rutas mínimas desde ruta_indices_dict[(src, dst)]. Cada ruta se transforma a una secuencia explícita de enlaces (u,v) coherente con el grafo del entorno y quedan almacenadas en un diccionario interno para acceso rápido durante la simulación. En el instante de activación del flujo (cuando start_time == t), se llama a pick_path_index(src, dst), y se aplica la política basada en el algoritmo *round robin*. Mantiene un contador interno por cada par (src, dst) y selecciona rutas en orden circular, garantizando reparto uniforme cuando múltiples flujos comparten el mismo origen y destino. La ruta elegida se guarda como ruta_asignada[f] = lista_de_enlaces. El ciclo de simulación sigue una estructura similar a la del modo prueba de DQN, esto es, activación de flujos, conteo de

flujos por enlace y avance del tráfico. Finalmente, se generan las mismas métricas que en el modelo DQN.

- Simulación con DRILL: su objetivo es seleccionar dinámicamente una ruta para cada flujo utilizando información local de congestión observada justo antes de su activación. Implementa una variante del enfoque *power-of-two choices*, adaptada al contexto de redes con rutas precalculadas. Para cada flujo (src, dst) se obtiene el conjunto de rutas mínimas y de ese se eligen dos rutas al azar. Para cada ruta candidata r, se calcula un costo local que es igual a la suma de la ocupación local de los enlaces que conforman esa ruta justo antes de activar el flujo. Se selecciona la ruta con menor costo y ocurre en el mismo paso en que el flujo se activa, utilizando la ocupación previa. El criterio es, por tanto, local, *greedy* y dependiente del estado actual de la red. El procedimiento del DRILL se describe en Algoritmo 3.

Algoritmo 3: Simulación DRILL

```
Data: traffic_flows.pkl, rutas_precalculadas.pkl
Resultado: Resultados_DRILL
begin
    // Inicialización del entorno
    env ← init_environment(rutas_precalculadas.pkl)
    t ← 0
    // Activación de flujos según start_time
    activar_flujos(env, traffic_flows)
    while env.existen_flujos_activos() do
        acciones_dict ← 0
        // Selección DRILL por flujo (power-of-two choices)
        for f ∈ env.flujos_activos(t) do
            A_validas ← env.obtener_rutas_validas(f)
            costo_min ← ∞
            a_f ← None
            // Elegir 2 rutas al azar y seleccionar la de menor congestión
            rutas_muestra ← sample(A_validas, 2)
            for r ∈ rutas_muestra do
                costo_r ← env.costo_local_por_congestion®
                if costo_r < costo_min then
                    costo_min ← costo_r
                    a_f ← r
                end
            end
```

Algoritmo 3: Simulación DRILL

```
25                          acciones_dict[f] ← a_f
26                  end
27                  // Transmisión del tráfico
28                  env.calcular_avance_por_enlace(acciones_dict)
29                  // Actualización de métricas por paso
30                  registrar(env.uso_real_por_enlace(t))
31                  registrar(env.balanceo_por_step_real(t))
32                  registrar(env.FCT_parciales(t))
33                  t ← t + 1
34          end
35          Resultados_DRILL ← env.exportar_metricas()
36  end
```

Al iniciar la simulación se carga ruta_indices_dict. Para cada par (src, dst), se construye la lista de rutas como secuencias de enlaces (u, v). Estas rutas se almacenan para rápida selección durante la activación de flujos. Cuando un flujo alcanza su tiempo de inicio, se ejecuta elegir_ruta_drill_links(src, dst, rutas_validas, ocupacion_actual). Con esto se toman dos rutas al azar del conjunto, se calcula su costo local de la ruta, se selecciona aquella con menor carga agregada y se la asigna al flujo. Antes de transmitir, el algoritmo actualiza internamente el conteo de flujos activos por enlace y la información de congestión utilizada para los siguientes flujos que se activen en el mismo paso. Después de la activación, DRILL sigue exactamente el mismo ciclo temporal que ECMP y DQN, esto es: conteo de flujos por enlace, cálculo de avance de tráfico por enlace y registro de métricas por paso. Finalmente, el módulo DRILL genera un conjunto de métricas homogéneo al de los otros dos esquemas.

Para comparar los modelos DQN, ECMP y DRILL se utiliza un conjunto común de métricas diseñadas para describir, de manera objetiva y reproducible, el comportamiento del tráfico en la topología FatTree. Estas métricas se eligieron en base a la literatura sobre balanceo de tráfico en centros de datos, donde el FCT y el balanceo de la red constituyen métricas primarias de eficiencia y estabilidad en redes de alto rendimiento.

- FCT. Cada flujo registra el instante en que inicia su transmisión (start_step) y el instante en que el flujo llega a completarse (finish_time). El FCT experimental se calcula como:

$$FCT_{exp} = (finish\_time - start\_step) \times tiempo\_por\_step\_seg$$

Esta métrica constituye un estándar muy importante en estudios de tráfico en centros de datos, pues es un reflejo de la experiencia del usuario final y la sensibilidad del sistema a la congestión. Por otra parte, el FCT teórico representa el tiempo mínimo posible si el flujo tuviera toda la capacidad de la ruta:

$$FCT_{teo} = \frac{size\_kB}{capacidad\_step\_kB}$$

Y luego FCT_ratio permite comparar rendimiento relativo entre mecanismos. Valores próximos a 1 indican eficiencia óptima; valores elevados evidencian congestión o rutas mal balanceadas.

$$FCT\_ratio = \frac{FCT_{exp}}{FCT_{teo}}$$

- Balanceo real por paso (desviación estándar del uso). El balanceo global por paso se obtiene mediante:

$$B(t) = std(\{uso\_real(e,t)\}_{e \in E})$$

en donde $uso\_real(e,t)$ es la fracción de ocupación del enlace e en el paso t, calculada como los kilobytes efectivamente transmitidos durante ese paso divididos para la capacidad máxima del enlace en un paso (valor comprendido entre 0 y 1). La desviación estándar es una métrica ampliamente utilizada en análisis de dispersión de tráfico, en donde, valores bajos señalan dispersión uniforme y valores altos indican congestión localizada.

En este punto, la simulación comparativa se convierte en una de las contribuciones centrales de la investigación, debido a que permite evaluar un modelo de aprendizaje por refuerzo diseñado específicamente para reflejar condiciones y limitaciones de una red real basada en la topología FatTree. Utilizando métricas de alta relevancia práctica —como FCT y balanceo del uso de enlaces— y comparándolas contra mecanismos ampliamente utilizados (ECMP) y heurísticos modernos (DRILL), se puede demostrar cómo un agente DQN puede responder a dinámicas complejas de congestión y adaptarse a ellas con mayor eficacia. Esto confirma el valor del enfoque propuesto para la gestión inteligente de tráfico en centros de datos.

F. Procesamiento estadístico y análisis comparativo

El procesamiento estadístico es el puente metodológico entre la simulación y la justificación científica del capítulo de resultados. El objetivo es hacer una comparación cuantitativa entre los mecanismos DQN, ECMP y DRILL en términos de procedimientos estandarizados, que garanticen

la objetividad, reproducibilidad y validez estadística. Esta etapa se utilizan los datos generados por los modelos, se los convierte en métricas normalizadas y se aplican herramientas estadísticas que permitan confirmar la hipótesis central de esta investigación.

- Cálculo de intervalos de confianza (t-Student [69]). Para medir la variabilidad entre ejecuciones y obtener conclusiones estadísticamente significativas, se utiliza un análisis basado en la distribución t-Student con nivel de confianza del 95%. Este enfoque es adecuado porque el tamaño de las muestras por escenario suele ser moderado, la distribución del FCT_ratio y del balanceo promedio no es necesariamente normal, pero la t-Student es robusta bajo estas condiciones (propiedad asintótica de la media), y proporciona intervalos para comparar métodos con variabilidad acotada. Para cada métrica se estima: media, error estándar, valor crítico t, e intervalo inferior y superior del 95%.
- Curvas de Distribución Acumulada (CDF). Además de los intervalos de confianza, el análisis gráfico comprende curvas CDF del FCT_ratio, ampliamente utilizadas en redes de centros de datos para visualizar la dispersión y detectar anomalías que no perceptibles en los promedios. Las CDF permiten observar mejoras distribucionales (colas más cortas, menor varianza), estabilidad bajo cargas altas, y qué fracción de flujos termina cerca del óptimo.
- Pruebas de significancia estadística. Adicionalmente al análisis por medio de intervalos de confianza, se aplicaron pruebas formales de significancia para evaluar si las diferencias observadas entre DQN, ECMP y DRILL eran atribuibles al mecanismo de balanceo y no solo a la variabilidad entre trazas de tráfico. Puesto que los tres modelos fueron evaluados con las mismas semillas y archivos de flujos, se adoptó una estructura de medidas relacionadas. Debido a esto, se utilizó la prueba de Friedman para comparar globalmente mediante rangos los tres modelos por cada nivel de carga y métrica evaluada. Del mismo modo, se calculó un ANOVA de una vía como análisis de varianza complementario, además de la prueba de Levene para examinar la homogeneidad de varianzas. Al detectarse diferencias significativas en la prueba global, se realizaron comparaciones por pares mediante Wilcoxon con corrección de Holm, para controlar el error por comparaciones múltiples.

VIII. RESULTADOS

El este capítulo se muestran los resultados experimentales para los tres mecanismos de balanceo de tráfico considerados en esta investigación: DQN, ECMP y DRILL. Cada uno de los experimentos se realizaron bajo condiciones equivalentes, usando la misma topología FatTree, el mismo grupo de rutas mínimas, las mismas capacidades por enlace y la misma semántica temporal determinada en el entorno de simulación. La uniformidad garantiza que sus resultados sean comparables objetivamente entre los tres modelos. El análisis se basa de forma exclusiva en las métricas observadas y exportadas por los scripts de simulación: FCT_ratio y balanceo de tráfico mediante la desviación estándar promedio del uso de enlaces. Igualmente, se utilizaron herramientas de análisis estadístico y de visualización para la comparación, esto es, intervalos de confianza basados en la distribución t-Student. Se presenta el análisis riguroso para validar si el mecanismo basado en DQN posee ventajas significativas en relación con los enfoques tradicionales ECMP y DRILL, en términos de eficiencia en la distribución del tráfico en la red. En estos resultados se encuentra la evidencia experimental que permitirá validar la hipótesis de la investigación.

A. Configuración experimental

La simulación se implementó en una topología FatTree k=4, confirmada por 16 hosts y 20 switches, interconectados por enlaces simétricos en sus diferentes niveles (acceso, distribución y núcleo). Se eligió k=4 porque es el menor valor de k que brinda una estructura de rutas múltiples completa (múltiples rutas mínimas entre cualquier par de hosts), permitiendo la evaluación de los mecanismos de balanceo DQN, ECMP y DRILL. De igual forma, k=4 conserva el tamaño de la red en un nivel computacionalmente manejable, evitando un enorme crecimiento de hosts y enlaces relacionada a valores superiores. Esto es consistente con la literatura, en donde k=4 es ampliamente utilizado como un estándar experimental en los estudios de balanceo de tráfico en centros de datos.

Acerca de la capacidad de los enlaces, todos están configurados a 1 Gbps. El valor se debe a razones de normalización y facilidad numérica. 1 Gbps es una tasa muy común facilitando la conversión a bytes por segundo y por intervalo de simulación. Dado que el objetivo del estudio es hacer una comparación del comportamiento de DQN frente a ECMP y DRILL, y no el de replicar a una red con una tasa de transmisión determinada, el uso de 1 Gbps es suficiente; utilizar escalas mayores (10, 40 o 100 Gbps) solo lograría FCT menores en la misma proporción, mas no alterarían las conclusiones comparativas del estudio.

La simulación funciona con una resolución temporal de 10 μs por paso. El valor se escogió como un balance entre granularidad y coste computacional. Para enlaces de 1 Gbps, en cada paso se puede enviar hasta 1.25 kB, es decir, cerca de un paquete Ethernet de tamaño máximo. Así, el tráfico se simula con una granularidad próxima al tiempo de transmisión de paquetes individuales, lo que permite una mejor captura de eventos como la congestión, sin la exigencia de demasiados pasos que harían inviable el entrenamiento y la evaluación del modelo DQN.

En la generación de tráfico se escogió un modelo ON/OFF con periodos ON de 0.3 s y periodos OFF de 0.2 s. Este patrón corresponde a un *Interrupted Poisson Process* [70] (IPP), frecuentemente utilizado para modelar tráfico en ráfagas en centros de datos. Con la relación ON/OFF = 60%/40% se alcanzan ráfagas lo suficientemente largas para generar una congestión perceptible, con pausas lo suficientemente largas para lograr el drenaje de flujos. Esta proporción produce unas condiciones en la que los mecanismos de balanceo evidencian diferencias significativas. Los valores escogidos son temporalmente coherentes con la simulación (10 μs por paso), lo que evita la congestión sostenida y la ausencia de eventos significativos.

En los periodos ON se usaron tres valores para λ: 100, 200 y 250. Estos se eligieron después de diversas pruebas preliminares con varios valores de λ, con el fin de plantear tres escenarios distintos de carga: uno ligero (λ=100), uno intermedio (λ=200) y uno pesado (λ=250) en el que la red opera muy cerca de su capacidad máxima. Al utilizar valores menores a 100, se generaba escasa congestión, y por ende las diferencias entre los mecanismos de balanceo fueron poco significativas; mientras que para valores mayores a 250, se incurría en una saturación casi constante, dificultando la interpretación de la aplicación de las políticas de selección de rutas. Con el rango elegido se puede visualizar de forma clara el impacto de DQN, ECMP y DRILL en el FCT y en el balanceo de la red.

Con el fin de asegurar la robustez estadística del experimento y el mantener la reproducibilidad de los resultados, para cada uno de los valores de λ se generaron cinco instancias independientes del tráfico, por medio de las semillas 101, 102, 103, 104 y 105; lo cual produce un total de 15 archivos de tráfico por modelo. Al utilizar cinco semillas se mantiene un equilibrio adecuado entre variabilidad y costo computacional, debido a que proporciona suficientes muestras para calcular medias e intervalos de confianza mediante t-Student, disminuyendo la influencia de comportamientos particulares de una sola traza y sin generar una cantidad excesiva de ejecuciones.

Sobre los tipos de flujos, la literatura acerca de centros de datos señala la proporción clásica de *mice* (entre el 80% y 95% de los flujos) y *elephant* (entre 5% y 20% de los flujos). Esta investigación adoptó la proporción 90% *mice* y 10% *elephant*, representativo del comportamiento real para el estudio de la interacción entre flujos pequeños sensibles a latencia y flujos grandes generadores de congestión. Los flujos *mice* se modelaron con tamaños aleatorios entre 20 y 100 kB, intervalo que reproduce el comportamiento observado en la literatura sobre tráfico real en centros de datos, en donde la mayoría de los flujos relacionados a RPCs, pequeñas consultas y mensajes de control son menores de 100 kB. Por otra parte, los flujos *elephant* se generaron con tamaños aleatorios entre 31250 kB y 50000 kB (30 a 50 MB aproximadamente), rango que ofrece carga suficientemente grande para inducir congestión, y que replica el efecto que se tienen en redes de centros de datos reales. De igual forma, estos tamaños mantienen la simulación computacionalmente manejable, y evitan así, volúmenes extremos que dificultarían el entrenamiento y la evaluación del modelo DQN.

Finalmente, el generador de tráfico produce tiempos de llegada de flujos en un intervalo máximo de 1.500.000 pasos (equivalente a 15 segundos), sin embargo, la simulación no termina en ese instante, sino que continúa en ejecución hasta que todos los flujos hayan terminado de transmitirse. De esta forma, el tiempo total simulado depende del nivel de congestión y puede superar ampliamente los 15 segundos nominales.

B. Representación gráfica de resultados

Previo al análisis los distintos escenarios de tráfico, se describen las representaciones gráficas empleadas para evaluar el desempeño de los mecanismos de balanceo. El análisis se apoya en los indicadores principales: el nivel de balanceo del tráfico y el FCT. Para el balanceo se considera la desviación estándar del uso real de los enlaces entre switches, calculada en cada paso de simulación a partir de la ocupación normalizada de dichos enlaces. Mediante un diagrama de barras se muestra el valor promedio de esta métrica a lo largo del intervalo analizado, lo que permite comparar el nivel global de balanceo que se alcanza en cada modelo. La CDF representa la distribución de los valores de balanceo calculados por paso, donde el eje X corresponde a la desviación estándar del uso de los enlaces y el eje Y a la probabilidad acumulada. Debido a que se trata de una métrica agregada por paso de simulación, la CDF se muestra de forma escalonada, reflejando la naturaleza discreta de las observaciones. Así mismo, el desempeño temporal se evalúa mediante el cociente entre el FCT experimental y el FCT teórico de cada flujo. En el diagrama de

barras se muestra el valor promedio de este cociente para cada mecanismo, mientras que la CDF asociada permite analizar la distribución completa de los FCT. Valores menores y CDFs desplazadas hacia la izquierda indican un mejor desempeño en términos de eficiencia temporal y menor penalización respecto al comportamiento ideal. A continuación, se muestran los resultados por escenario.

C. Escenario 1: Régimen de tráfico bajo

En el escenario con tráfico bajo (λ = 100), la red opera con niveles de congestión reducidos y, por ende, las diferencias entre los tres mecanismos de balanceo llegan a ser menos evidentes que en escenarios de mayor carga. Sin embargo, incluso en este nivel de tráfico se observan tendencias claras en el desempeño de DQN, ECMP y DRILL.

Resultados de balanceo de tráfico. La Fig. 9 presenta la desviación estándar promedio del uso de los enlaces. En este escenario, DQN obtiene la menor dispersión (~0.39), seguido de DRILL (~0.41), mientras que ECMP muestra el peor balanceo (~0.46). Si bien las diferencias absolutas son moderadas y coherentes con un escenario de baja congestión, sí muestran que DQN distribuye el tráfico de forma más uniforme. La CDF respectiva refuerza este análisis: las curvas de DQN y DRILL se encuentran en su mayoría por debajo de la de ECMP, indicando una mayor cantidad de pasos con desviación estándar reducida. Esto significa una utilización más homogénea de los enlaces, incluso cuando la red no se encuentra demandada.

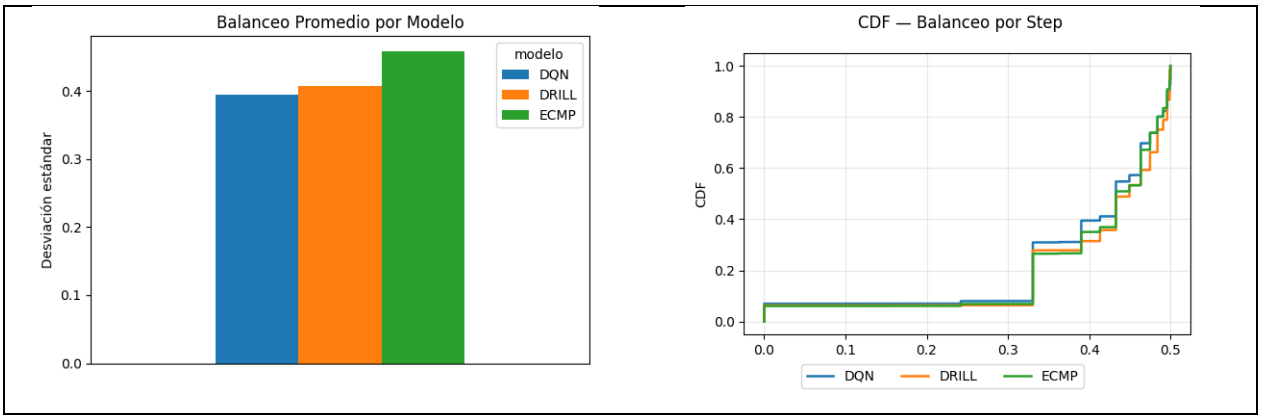

Fig. 9 Resultados de balanceo con tráfico bajo (lambda = 100)

Resultados de FCT (cociente FCTexp/FCTteo). La Fig. 10 muestra el valor promedio del cociente FCTexp/FCTteo para los tres mecanismos. Tanto DQN como DRILL llegan a valores prácticamente idénticos (~1.55), lo que indica que, bajo carga ligera, ambos mecanismos consiguen valores de FCT cercanos al óptimo teórico. En este contexto, el óptimo teórico corresponde a un cociente igual a 1.0, que representa el caso ideal en ausencia de tráfico adicional que genere

contención sobre los flujos. Este valor constituye, por tanto, un límite inferior inalcanzable en escenarios con múltiples flujos concurrentes, pero muy útil como una referencia para cuantificar la penalización introducida por la congestión. Por el contrario, ECMP presenta un cociente significativamente mayor (~2.8), lo que pone en evidencia que, incluso con congestión mínima, la elección estática basada en *round-robin* puede generar rutas desbalanceadas que aumentan innecesariamente el FCT. Este comportamiento se confirma en la CDF, en donde las curvas de DQN y DRILL se superponen en la mayoría del dominio, mientras que ECMP muestra una cola más pesada y una mayor proporcionalidad de flujos con un cociente superior a 2.5. Esto evidencia que ECMP no logra aprovechar apropiadamente la diversidad de rutas, a pesar de que existen recursos suficientes en la red.

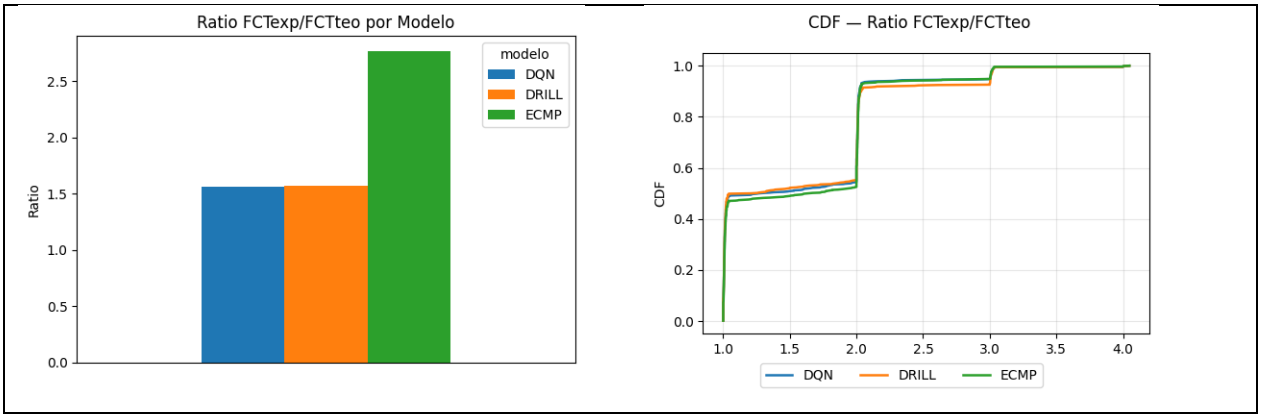

Fig. 10 Resultados de FCT con tráfico bajo (lambda = 100)

Interpretación general del escenario. De los resultados de bajo tráfico permiten se obtienen dos conclusiones principales: DQN y DRILL evidencian desempeños equivalentes en FCT, muy cercanos al óptimo teórico. La red no está lo suficientemente cargada como para que las decisiones adaptativas de DQN demuestren una ventaja significativa sobre DRILL; y DQN sí muestra una ventaja consistente en términos de balanceo, consiguiendo una dispersión menor de carga entre los métodos analizados. Se comprueba que el agente aprende a evitar rutas que podrían aglomerar tráfico sin necesidad, incluso cuando la carga sobre la red es baja. En resumen, el escenario λ = 100 sugiere que, en condiciones propicias, la red está lo suficientemente holgada como para que cualquier método adaptativo funcione apropiadamente, sin embargo, DQN se destaca por mantener un balanceo más uniforme, lo que sugiere que su ventaja se vuelva aún más evidente en niveles mayores de tráfico.

D. Escenario 2: Régimen de tráfico medio

En el escenario de tráfico medio, correspondiente a λ = 200, la red experimenta un nivel de carga mucho mayor que en el caso anterior. En estas condiciones, la distribución de flujos tiene un impacto más marcado tanto en el FCT como en la uniformidad de la utilización de los enlaces. Las diferencias entre mecanismos empiezan a notarse con mayor claridad, manifestando cual es la capacidad que tiene cada uno para reaccionar ante la congestión emergente.

Resultados de balanceo de tráfico. En términos de balanceo, la Fig. 11 muestra que ECMP presenta sorprendentemente la menor desviación estándar promedio (~0.36), levemente inferior que DQN (~0.38) y DRILL (~0.39). Este comportamiento se justifica en la distribución casi uniforme de los flujos empleada por el algoritmo *round-robin*, aunque no necesariamente hacia rutas óptimas desde el punto de vista del FCT. Sin embargo, la CDF del balanceo muestra que DQN se sostiene en una distribución de carga más estable, y con una mayor concentración de valores en el rango medio-bajo. DRILL presenta mayor variabilidad, siendo consistente con su estrategia de selección aleatoria entre dos rutas. ECMP, si bien alcanza valores bajos en promedio, evidencia una dispersión irregular y episodios con desviaciones más altas. La diferencia entre promedio y distribución muestra que ante una menor desviación estándar de ECMP no necesariamente implica una mejor eficiencia, es decir, aunque el uso de los enlaces puede ser simétrico, no precisa que esté correlacionado con rutas libres de congestión.

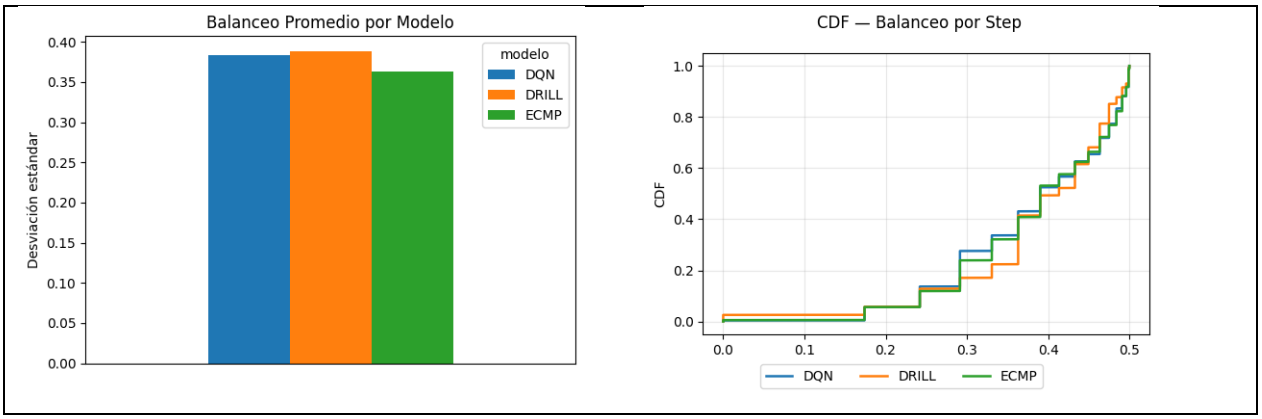

Fig. 11 Resultados de balanceo con tráfico medio (lambda = 200)

Resultados de FCT (cociente FCTexp/FCTteo). En la Fig. 12 se muestran los valores promedio del cociente FCTexp/FCTteo para los tres modelos. DQN logra el mejor desempeño con un valor cercano a 2.6, seguido de DRILL (~2.8). Contrariamente a estos resultados, ECMP evidencia un deterioro significativo, con un cociente superior a 5.2, que equivale a más del doble del tiempo relativo obtenido por los otros mecanismos. Este resultado indica que, con el aumento

de la carga, ECMP no puede distribuir adecuadamente los flujos entre las rutas disponibles, lo que provoca acumulación de tráfico en rutas específicas y aumento de los FCT de manera importante. La CDF correspondiente refuerza lo indicado: DQN mantiene una curva desplazada más hacia la izquierda comparada con DRILL y ECMP, evidenciando mayor proporción de flujos con tiempos cercanos al óptimo. DRILL muestra una ligera desviación hacia cocientes mayores, demostrando cierta inestabilidad al momento que la red empieza a saturarse. ECMP presenta una cola mucho más extendida, con varios flujos que alcanzan cocientes entre 5 y 6, indicando un desempeño significativamente inferior.

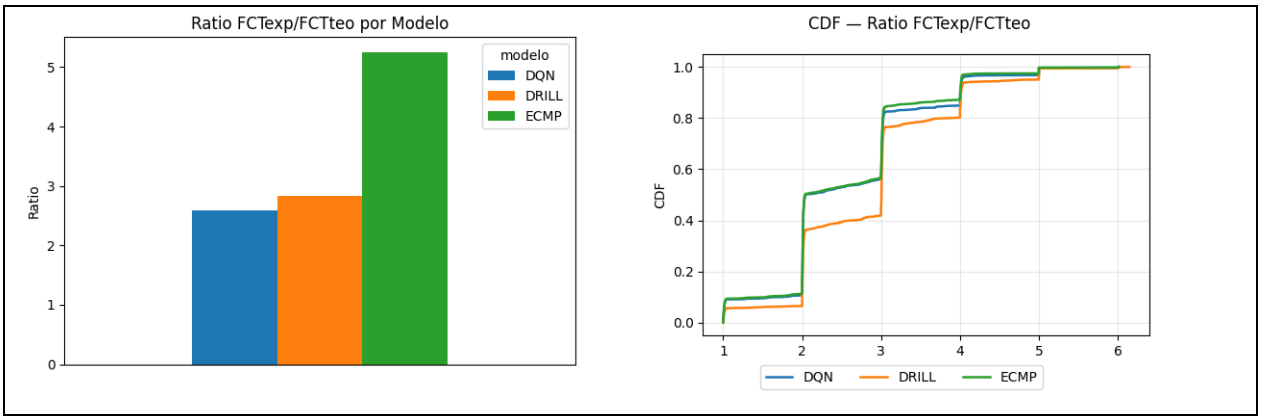

Fig. 12 Resultados de FCT con tráfico medio (lambda = 200)

Interpretación general del escenario. El escenario λ = 200 muestra un punto en donde los mecanismos comienzan a diferenciarse de manera más clara en su funcionamiento. DQN se mantiene con el mejor cociente FCTexp/FCTteo, señal que indica que sus decisiones de selección de rutas evitan acumulaciones innecesarias de tráfico incluso cuando el nivel de carga ya es significativo. DRILL mejora respecto a ECMP, sin embargo, su desempeño ya evidencia desgaste, notoriamente en la cola de la distribución de FCT; esto se debe a que la elección aleatoria entre dos rutas no es garantía de decisiones robustas cuando hay congestión creciente. ECMP evidencia un deterioro considerable en FCT, pese a que su desviación estándar promedio se muestra competitiva. La aparente contradicción indica que no necesariamente un buen balanceo estadístico equivale a un buen desempeño; ECMP distribuye flujos independientemente del estado actual de la red, por lo que se reduce la eficiencia general. Resumiendo, en λ = 200 se nota que las ventajas de DQN comienzan a consolidarse, en eficiencia de extremo a extremo y en consistencia de distribución de carga.

E. Escenario 3: Régimen de tráfico alto

El escenario correspondiente a λ = 250 presenta una situación de alta congestión, mediante la cual la capacidad de la red se encuentra cercano a su límite operativo. Bajo estas condiciones, las decisiones de selección de rutas presentan un impacto significativo en la estabilidad del tráfico, por lo tanto, en este escenario se puede revelar con mayor claridad las fortalezas y debilidades de cada mecanismo de balanceo.

Resultados de balanceo de tráfico. La Fig. 13 muestra que, en este escenario, las diferencias entre los mecanismos son más marcadas. En términos de desviación estándar promedio, DRILL obtiene el valor más bajo (~0.18), con lo cual su balanceo es ligeramente más uniforme en promedio. DQN presenta un valor cercano (~0.19), similar al de DRILL y ECMP contabiliza la peor dispersión (~0.24), con lo que se reafirma que su comportamiento no escala adecuadamente en alta congestión. De forma complementaria, la CDF del balanceo revela que DRILL y DQN se mantienen con distribuciones relativamente estables, una menor proporción de pasos con desviación elevada. ECMP es el más variable, especialmente en valores más altos, indicando desigualdad en la carga de enlaces que no fue visible solo en el promedio. La aparente ventaja de DRILL en el promedio se matiza con lo siguiente: aunque DRILL tiene la menor desviación estándar media, la curva presenta una mayor concentración de pasos con valores intermedios y altos en comparación a DQN, lo que se traduce en inestabilidad en ciertas situaciones. Por otro lado, DQN se mantiene con una distribución más controlada, coherente con un comportamiento reactivo y ajustado al estado real de la red.

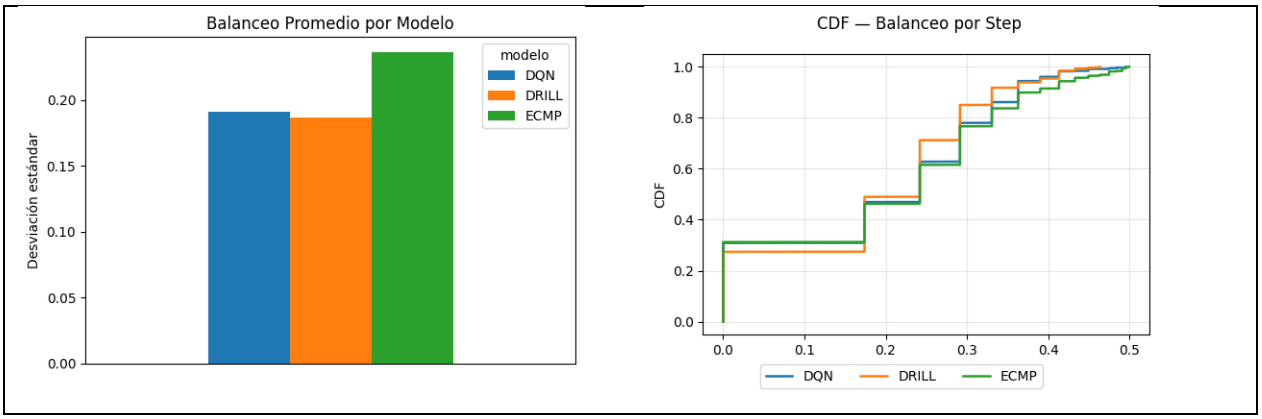

Fig. 13 Resultados de balanceo con tráfico alto (lambda = 250)

Resultados de FCT (cociente FCTexp/FCTteo). La Fig. 14 hace evidente que DQN alcanza el mejor desempeño, con un cociente promedio cercano a 3.7, seguido por DRILL, con un valor alrededor de 4.1. Por el contrario, ECMP muestra un deterioro importante, alcanzando un cociente

de 7.7, lo que representa a más del doble del tiempo relativo obtenido con los mecanismos adaptativos. Las diferencias son más marcadas en la CDF del cociente FCTexp/FCTteo, debido a que la curva de DQN se mantiene consistente a la izquierda de las otras dos, lo que indica una mayor proporción de flujos con tiempos cercanos al óptimo teórico. DRILL muestra un desempeño intermedio, sin embargo, su curva se desplaza hacia valores mayores a medida que aumentan los FCT, lo que corresponde a la toma de decisiones subóptimas bajo condiciones de congestión severa. ECMP evidencia la peor distribución, con una proporción significativa de flujos con tiempos que superan el cociente 6 e incluso llegando a valores cercanos a 8. Esto confirma que cuando la red está operando a niveles cercanos a la saturación, solo un mecanismo de selección de rutas que pueda interpretar el estado dinámico de la red (DQN), mantiene los tiempos bajo control; a diferencia de ello, estrategias estáticas o con decisiones parcialmente aleatorias (DRILL) pierden eficacia.

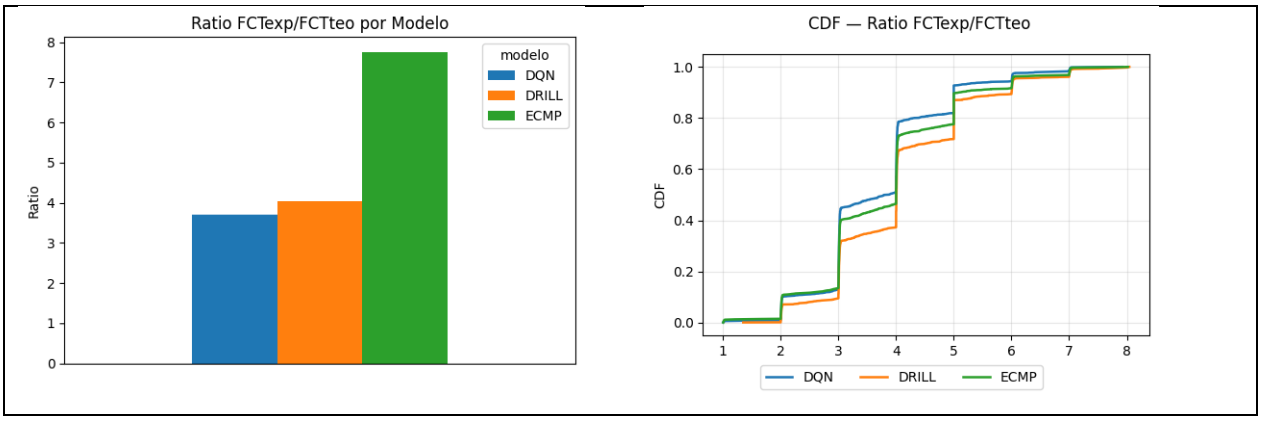

Fig. 14 Resultados de FCT con tráfico alto (lambda = 250)

Interpretación general del escenario. En condiciones de tráfico alto, el resultado de los mecanismos es claramente divergente: DQN es evidentemente el método más robusto, debido a que se mantiene con el mejor FCT y un balanceo estable aún en congestión severa. Con la política aprendida se evita saturar rutas críticas y se distribuye el tráfico acorde al estado actual del entorno. Aunque DRILL mejora el promedio de balanceo, falla en la eficiencia del FCT que es afectada por las decisiones locales que no consideran la dinámica global de la red. ECMP es indiscutiblemente superado por ambos mecanismos adaptativos, con degradaciones importantes en FCT y en balanceo, por ende, se confirma que las políticas estáticas basadas en el algoritmo *round-robin* no se adecuan en escenarios donde los estados de congestión varían rápidamente. En síntesis, en el escenario $\lambda = 250$ son notorias las ventajas de DQN en cuanto a la capacidad que tiene el modelo en la gestión para la toma de decisiones de selección de rutas en centros de datos.

F. Síntesis de resultados experimentales

En el análisis en conjunto de los tres escenarios evaluados, tráfico bajo (λ = 100), tráfico medio (λ = 200) y tráfico alto (λ = 250), se pueden encontrar patrones consistentes en el comportamiento de los mecanismos DQN, DRILL y ECMP. Estos patrones muestran de forma clara las fortalezas y debilidades de cada enfoque, y la relación entre balanceo, congestión y eficiencia en la transmisión dentro de la red FatTree que está sometida a carga variable. En primer lugar, los resultados de balanceo evidencian que no siempre una menor desviación estándar promedio implica un mejor desempeño global. ECMP, por ejemplo, presenta valores relativamente bajos de variabilidad en el escenario de tráfico medio, pero obtiene los peores FCT. Una de las causas es que el balanceo alcanzado por ECMP no responde de forma adaptativa, sino a efectos del algoritmo que distribuye los flujos, pero sin considerar el estado real de congestión. En cambio, DQN alcanza una combinación más equilibrada entre uniformidad y estabilidad, pues mantiene una distribución de carga mucho más controlada incluso cuando el promedio no es el más bajo. DRILL exhibe un comportamiento intermedio: por un lado, logra reducir la desviación estándar en ciertos escenarios (especialmente bajo alta carga) pero su variabilidad es mayor, al mostrar una capacidad de reacción limitada en condiciones locales y con una ausencia de memoria o visión global del estado de la red. En segundo lugar, se observa que con DQN se mantiene el mejor desempeño global en términos del cociente FCTexp/FCTteo, con una ventaja de que se torna más marcada cuando aumenta la carga. En tráfico bajo, DQN y DRILL muestran comportamientos similares, manifestando que, en condiciones favorables, el impacto que tiene el algoritmo de selección de rutas es menos definitivo. No obstante, conforme la red se aproxima a la saturación, DQN continúa seleccionando las rutas más eficientes, distribuyendo flujos de forma inteligente con miras a evitar la congestión acumulada. Por el contrario, con ECMP su rendimiento empeora significativamente en ambientes en donde la congestión es más frecuente, y con DRILL, su comportamiento se torna menos estable conforme aumenta la cantidad de puntos de congestión. La comparación integrada demuestra que la ventaja de DQN no se encuentra en una sola métrica, sino en su capacidad para mantener equilibrado de forma simultánea dos dimensiones críticas: una distribución de carga estable y controlada, y minimizar el FCT incluso bajo congestión severa. Conforme la red opera cerca de sus límites, es más difícil de lograr el equilibrio mediante mecanismos tradicionales o heurísticos locales; a diferencia de ello, el enfoque basado en DQN mantiene su capacidad de adaptación, al evitar puntos de congestión persistentes y atenuando los

impactos acumulativos de decisiones subóptimas. En síntesis, los resultados proporcionan evidencia robusta de que el modelo DQN conforma un mecanismo de balanceo significativamente más eficaz que ECMP y más estable que DRILL, en especial cuando se presentan escenarios realistas en donde la carga varía y la congestión surge de manera dinámica.

G. Análisis estadístico mediante intervalos de confianza (t-Student)

Con el fin de evaluar si las diferencias observadas entre los mecanismos DQN, DRILL y ECMP son estadísticamente consistentes, se hicieron cálculos de los intervalos de confianza al 95% por medio de la distribución t-Student. Este método resulta adecuado por cuanto para cada escenario de carga (λ = 100, 200 y 250) se generaron cinco muestras independientes por modelo, lo que resulta en un tamaño de muestra reducido. Por ello, se necesitó de una distribución que considere la variabilidad adicional asociada a grados de libertad pequeños. Seguidamente, se analizan los resultados para las dos métricas principales: balanceo de carga (desviación estándar promedio) y cociente FCTexp/FCTteo.

Intervalos de confianza del balanceo de carga. La Fig. 15 muestra los IC-95% de la desviación estándar del uso de enlaces por cada modelo y cada valor de λ. Los principales hallazgos con λ = 100 es que los tres modelos muestran intervalos estrechos, reflejando escasa variabilidad entre ejecuciones. DQN y DRILL muestran valores muy similares, mientras que ECMP se ubica por encima de ellos. Esto confirma la tendencia observada en los resultados: con tráfico ligero, las diferencias son pequeñas pero reproducibles. Con λ = 200, los intervalos se vuelven mayores, en especial con DQN y DRILL, reflejando una dinámica más variable del tráfico conforme aumenta la congestión. No obstante, DQN y DRILL se mantienen con medias ligeramente inferiores a ECMP, evidenciando una ventaja estadística leve pero consistente. Finalmente, con λ = 250, la desviación estándar disminuye en los tres modelos debido a que, en alta saturación, la red pierde grados de libertad con tendencia a operar en estados de congestión más homogéneos. Las diferencias entre modelos son evidentes: DQN y DRILL mantienen sus valores inferiores a ECMP, con intervalos que no se superponen en su totalidad, mostrando que el desempeño inferior de ECMP no es por el azar. En general, en los intervalos se muestran que el mejor balanceo observado con DQN y DRILL en escenarios exigentes es estadísticamente consistente, mientras que ECMP queda por encima en los tres niveles de carga.

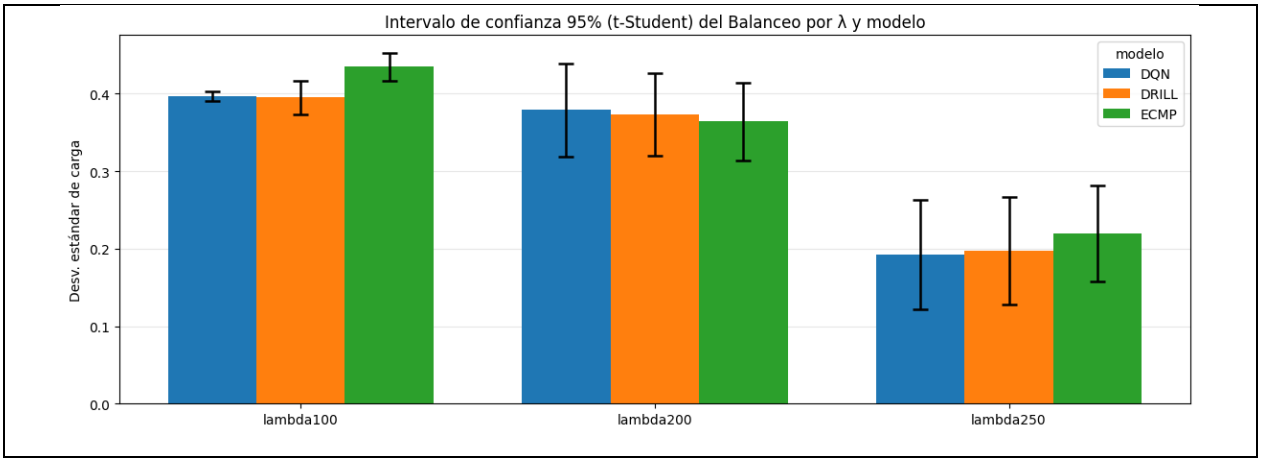

Fig. 15 Gráfica de intervalos t-Student para el balanceo

Intervalos de confianza del cociente FCTexp/FCTteo. La Fig. 16 muestra los IC-95% del cociente FCTexp/FCTteo. Las diferencias en los mecanismos son más marcadas que en el balanceo, confirmando que el FCT es una métrica muy sensible a la calidad de las decisiones de selección de rutas. Con λ = 100, DQN y DRILL muestran intervalos estrechos y muy cercanos entre sí, ratificando que, en baja congestión, ambos mecanismos funcionan similarmente. ECMP presenta valores notablemente mayores, con un intervalo que no se traslapa con los de DQN y DRILL. Con λ = 200 las diferencias se amplían. DQN conserva el mejor FCT, con un intervalo claramente separado hacia abajo respecto a DRILL y ECMP. DRILL tiene un rendimiento intermedio, mientras que ECMP queda muy arriba en promedio y con intervalos también más amplios. Con λ = 250, el patrón observado se intensifica. El intervalo de DQN se desplaza hacia valores mayores que en λ = 200 (debido a la congestión severa), pero se mantiene como el más bajo de los tres, con un rango que no se superpone con el de ECMP e incluso se separa parcialmente de DRILL. DRILL decae, y ECMP muestra el peor comportamiento, con un mayor promedio e intervalo más amplio, indicando ineficiencia y alta variabilidad.

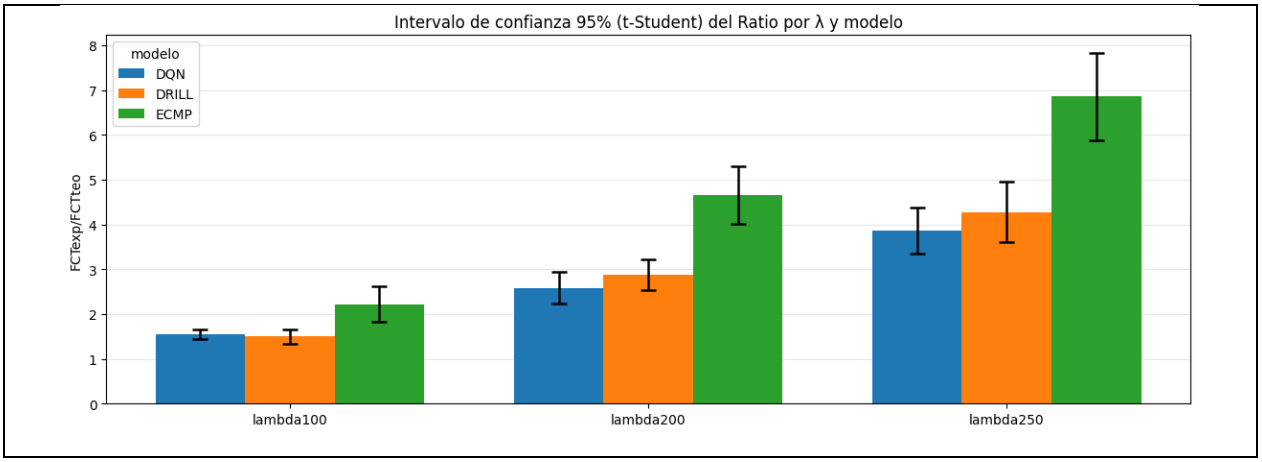

Fig. 16 Gráfica de intervalos t-Student para el FCT

Con los resultados del análisis t-Student se refuerza la validez de los hallazgos, evitando conclusiones basadas en situaciones particulares que se puedan presentar en algunos de los escenarios de simulación. Se puede concluir que las diferencias entre DQN y los demás mecanismos no son producto del azar, sino que se encuentran dentro de intervalos de confianza consistentes en todos los escenarios. DQN alcanza sistemáticamente el mejor desempeño en FCT, y esta ventaja es más pronunciada en la medida en que la carga aumenta. En términos de balanceo, tanto DQN como DRILL superan a ECMP en cada uno de los tres niveles de carga, pero DQN logra un equilibrio superior entre uniformidad y eficiencia temporal, algo confirmado estadísticamente. El deterioro de ECMP en carga media y alta no solo es visual (gráficas), sino también estadísticamente significativo. De manera integrada, este análisis respalda cuantitativamente que el modelo DQN presenta ventajas sólidas y reproducibles sobre ECMP y DRILL, validando la hipótesis central de la investigación y demostrando la eficacia del enfoque de aprendizaje por refuerzo profundo para el balanceo de tráfico en redes de centros de datos.

H. Pruebas de significancia estadística

A fin de complementar el análisis de los intervalos de confianza t-Student, se emplearon pruebas formales de significancia estadística en dos métricas principales: balanceo promedio y cociente FCTexp/FCTteo. El objetivo fue analizar si las diferencias observadas entre DQN, DRILL y ECMP se atribuyen al mecanismo de balanceo utilizado y no solo a la variabilidad propia de las trazas de tráfico. Por esto los tres modelos fueron evaluados con las mismas semillas y archivos de flujos, considerando una estructura de medidas relacionadas; por consiguiente, la prueba de Friedman se usó como comparación principal. A fin de ampliar el análisis, se incluyó ANOVA de

una vía como análisis de varianza y la prueba de Levene para la verificación de la homogeneidad de varianzas. En la Tabla II se resumen los resultados obtenidos para cada métrica y nivel de carga.

TABLA II

PRUEBAS DE SIGNIFICANCIA ESTADÍSTICA POR MÉTRICA Y NIVEL DE CARGA

Métrica	λ	Levene p	ANOVA p	Friedman p	Kendall W	Interpretación
Balanceo	100	0.350104	0.000651	0.022371	0.76	Diferencias significativas
Balanceo	200	0.818732	0.870404	0.449329	0.16	No significativo
Balanceo	250	0.943817	0.705419	0.090718	0.48	No significativo, con tendencia
FCTexp/ FCTteo	100	0.403906	0.000159	0.014996	0.84	Diferencias significativas
FCTexp/ FCTteo	200	0.428628	0.000003	0.006738	1.00	Diferencias significativas
FCTexp/ FCTteo	250	0.482656	0.000008	0.006738	1.00	Diferencias significativas

Con la prueba de Levene se obtuvieron valores p superiores a 0.05 en todos los casos, con lo cual no se evidenció una vulneración significativa de la homogeneidad de varianzas entre modelos. De manera complementaria, con el ANOVA de una vía se identificaron diferencias globales entre DQN, DRILL y ECMP, en tanto que la prueba de Friedman se empleó como método de comparación principal debido a que los modelos se evaluaron con las mismas semillas de tráfico.

En los resultados se evidencia que, para la métrica FCTexp/FCTteo, hay diferencias estadísticamente significativas en los modelos en los tres niveles de carga evaluados. Para $\lambda = 100$, Friedman reportó un $p = 0.014996$; con $\lambda = 200$, $p = 0.006738$; y con $\lambda = 250$, $p = 0.006738$. Estos resultados corroboran que las diferencias observadas en el FCT no corresponden únicamente a variabilidad aleatoria entre trazas, sino que responden al efecto del mecanismo de selección de rutas empleado. También, los valores de Kendall W fueron altos, en especial con $\lambda = 200$ y $\lambda = 250$, en donde alcanzaron $W = 1.00$, lo que muestra un efecto fuerte y consistente en la distinción entre modelos.

Para el caso de la métrica de balanceo, se presentaron diferencias significativas en el escenario de tráfico bajo, con Friedman p = 0.022371 y Kendall W = 0.76. En cambio, en los escenarios de tráfico medio y alto no se registró significancia estadística global al nivel de 0.05, aunque con λ = 250 si se observa una tendencia con p = 0.090718. Esta respuesta tiene coherencia con la discusión experimental previa: la desviación estándar del uso de enlaces no necesariamente refleja por sí misma la eficiencia de extremo a extremo del sistema, debido a que un mecanismo es capaz de distribuir carga de forma aparentemente uniforme sin minimizar el FCT. Es por esto que el FCTexp/FCTteo se perfila como la métrica más sensible para demostrar el impacto del aprendizaje por refuerzo en la calidad de las decisiones de selección de rutas.

En la misma línea, se efectuaron comparaciones por pares post-hoc mediante Wilcoxon con corrección de Holm. Los resultados no se presentan en Tabla II dado que los valores p ajustados no fueron inferiores al nivel de 0.05; sin embargo, en su interpretación se debe tomar en cuenta el tamaño de la muestra reducido de cinco semillas por escenario, lo que evidentemente limita la sensibilidad estadística de las pruebas por pares. Ante estas consideraciones, la prueba global de Friedman brinda una lectura más acorde al efecto conjunto entre modelos, en cambio el post-hoc efectúa una función complementaria. En definitiva, las pruebas de significancia robustecen la validez estadística de los resultados, en especial de la métrica FCTexp/FCTteo, en donde se hallaron diferencias globales significativas entre DQN, ECMP y DRILL en los tres niveles de carga evaluados.

I. Análisis de escalabilidad y retardos no modelados

Como un complemento a la interpretación de los resultados, es preciso considerar dos aspectos que delimitan el alcance experimental del modelo: la escalabilidad de la topología y los retardos no modelados en el entorno de simulación. Sobre la escalabilidad del modelo, los resultados obtenidos en una topología FatTree K=4 permiten avalar el comportamiento del agente DQN en un escenario controlado de múltiples rutas equivalentes, 20 switches y 16 hosts. Sin embargo, al incrementar el grado de FatTree o el número de servidores, el espacio de estados y acciones aumenta considerablemente, a causa del aumento en la cantidad de enlaces, rutas mínimas posibles y combinaciones de congestión que el agente debe considerar. Desde este punto de vista, se esperaría que el modelo conserve su capacidad de adaptación siempre y cuando el entrenamiento incorpore escenarios a mayor escala; no obstante, podría también incrementarse el tiempo de entrenamiento, la memoria requerida y la complejidad en la exploración de las acciones. Debido a

esto, la generalización hacia topologías FatTree de mayor grado forma parte de una línea necesaria de validación futura, en especial para el análisis de la política aprendida y si en una escala determinada puede transferirse o requiere reentrenamiento completo.

De igual manera, hay que considerar que en el entorno de simulación se prioriza el análisis del balanceo de tráfico y del FCT relacionado con la disponibilidad de ancho de banda en los enlaces, por lo tanto, no modela de forma detallada determinados retardos adicionales presentes en una implementación real, tales como el retardo de procesamiento en switches, la latencia de comunicación con el controlador SDN, el tiempo de instalación de reglas de flujo o el exceso de cómputo del agente. Esta elección metodológica viabilizó el aislamiento del efecto de las decisiones de selección de rutas sobre la congestión y el uso de los enlaces, lo que aseguró una comparación homogénea entre DQN, ECMP y DRILL. Aun así, en escenarios reales, dichos retardos podrían influir en el FCT observado, especialmente en flujos *mice*, por su sensibilidad a pequeñas variaciones temporales. Por consiguiente, una validación posterior en emuladores SDN o en infraestructuras físicas haría posible el complementar los resultados e incorporar retardos de procesamiento, señalización y control.

IX. CONCLUSIONES

El propósito de esta investigación fue el desarrollar y evaluar un sistema de control adaptativo basado en las metodologías KDN con el fin de optimizar el balanceo de tráfico en centros de datos, por medio de la integración de algoritmos de aprendizaje automático en un controlador SDN. En respuesta a necesidad, se implementó un agente DQN con capacidades de selección de rutas en forma dinámica a partir del estado de los enlaces, y se comparó su eficiencia contra dos mecanismos tradicionales: ECMP y DRILL, en condiciones de tráfico variable. Los resultados obtenidos permiten aseverar que el sistema desarrollado cumple con la hipótesis planteada y con los objetivos definidos en este trabajo.

En la hipótesis se argumentaba que un controlador SDN potenciado con técnicas de aprendizaje automático podría adaptarse rápida y eficientemente a las fluctuaciones del tráfico, por medio de la aplicación de un balanceo capaz de prevenir la congestión en tiempo real en base a decisiones informadas en la selección de rutas. Los resultados experimentales corroboran esta afirmación. El agente DQN demostró:

- adaptación inmediata ante cambios en la carga del tráfico en cada paso de la simulación,
- prevención efectiva de congestión, evitando la acumulación de flujos en enlaces saturados,
- toma de decisiones en tiempo real, considerando estados actualizados de ocupación de la red,
- mejor rendimiento extremo a extremo, evidenciado por cocientes FCTexp/FCTteo consistentemente menores que los obtenidos con ECMP y DRILL, y
- mayor estabilidad en la distribución de carga, reduciendo desviaciones críticas aún bajo condiciones de estrés.

Esto confirma que el enfoque KDN basado en observación, aprendizaje y decisión puede implementarse de forma exitosa en un controlador SDN por medio de algoritmos DQN.

En los tres escenarios evaluados (λ = 100, 200 y 250), DQN obtiene los menores tiempos relativos de FCT. Su ventaja es más marcada conforme aumenta la congestión, lo que evidencia su capacidad de ajustar la elección de rutas que eviten enlaces críticos. Aunque DRILL alcanza en ocasiones un promedio ligeramente menor de desviación estándar, su distribución muestra variabilidad significativa. Mientras que, DQN distribuye la carga de manera más consistente, evitando picos súbitos de ocupación y manteniendo una menor dispersión en tráfico pesado,

exhibiendo un mayor control sobre la utilización de los enlaces. Con esto se demuestra que la política aprendida favorece el uso equilibrado de los recursos de la red.

El tráfico en centros de datos se caracteriza por una mezcla asimétrica: muchos flujos *mice* y pocos flujos *elephant*. Este patrón genera fluctuaciones rápidas y congestión sostenida que los mecanismos tradicionales no manejan adecuadamente. En este escenario ECMP no distingue entre flujos pequeños y grandes, lo que es causa de posibles colisiones de *elephants* en rutas críticas y degradación severa del FCT. DRILL reacciona de forma parcial, pero su visión local y falta de memoria generan variabilidad cuando coexisten ráfagas de *mice* y *elephants*. DQN, al aprender de la evolución del estado de los enlaces, evita ubicar *elephants* en rutas congestionadas y resguarda rutas despejadas para los *mice*, resultando en un menor FCT y una menor desviación del balanceo. La capacidad del modelo para manejar esta heterogeneidad en tamaños de flujo constituye una evidencia directa de su adaptación efectiva, lo cual es el núcleo de la hipótesis planteada. La naturaleza estática de ECMP, basada en el algoritmo *round-robin*, lo hace incapaz de ajustarse a condiciones cambiantes. Esto se traduce en congestión sostenida, degradación del rendimiento, y variabilidad significativa del FCT. DRILL mejora a ECMP, pero no posee el aprendizaje adaptativo necesario. Su desempeño es intermedio: evita algunos puntos de congestión, pero no llega a una estrategia global estable bajo carga elevada.

Los resultados que se obtuvieron confirman que sí se cumplieron los objetivos de la investigación: Se desarrolló un modelo adaptativo basado en aprendizaje automático dentro del marco KDN para la asignación de rutas, se evaluó su rendimiento en condiciones de tráfico variable y congestión analizando métricas clave como el balanceo y el FCT, y se comparó su desempeño frente a mecanismos tradicionales, demostrando mejoras significativas tanto en eficiencia como en estabilidad.

Con la inclusión de pruebas formales de significancia estadística se confirmó que las diferencias obtenidas en la métrica FCTexp/FCTteo son significativas en los tres niveles de carga evaluados, lo cual respalda la validez cuantitativa de la comparación entre DQN, ECMP y DRILL. Para el caso del balanceo, en los resultados se revelan diferencias significativas sobre todo en baja carga, en tanto que en escenarios de mayor carga la interpretación debe efectuarse en conjunto con el FCT, al ser esta la métrica que evidencia con mayor sensibilidad la eficiencia extremo a extremo del sistema.

Estos hallazgos demuestran que los controladores SDN sí pueden beneficiarse significativamente con la integración de modelos DQN. Las metodologías KDN conforman el marco adecuado para evolucionar hacia redes auto-observables, con capacidades de aprendizaje autónomo y auto-optimizables. Los mecanismos basados en DQN permiten que se incremente la eficiencia de los centros de datos al reducir la congestión y mejorar la utilización de rutas múltiples. Estos hallazgos son relevantes para el diseño de controladores SDN de próxima generación y para la operación de centros de datos modernos con alta demanda.

Las evidencias permiten inferir que el enfoque propuesto podría extenderse hacia escenarios con una mayor complejidad topológica, aunque el crecimiento del espacio de estados y acciones incorpora un desafío importante en términos de escalabilidad computacional. Este aspecto se consideró durante el diseño de la solución basada en DQN y forma parte de una línea relevante hacia futuras investigaciones encaminadas a topologías de mayores dimensiones y entornos SDN más complejos.

Como propuestas de trabajos futuros, se identifican las siguientes oportunidades de investigación:

- Escalar el modelo hacia topologías FatTree mayores ($k > 4$) u otras topologías.
- Integrar modelos de aprendizaje por refuerzo más avanzados (PPO, DDPG, SAC).
- Considerar métricas adicionales como pérdidas, colas y retardo de propagación.
- Evaluar escenarios con múltiples controladores SDN o control distribuido.
- Incorporar mecanismos predictivos que anticipen picos de carga mediante modelos temporales (LSTM, transformers).
- Integrar objetivos de eficiencia energética y balanceo verde (energy-aware load balancing).
- Explorar la implementación y validación del sistema en un entorno real con hardware SDN, cuando se cuente con la infraestructura necesaria, con el fin de contrastar los resultados obtenidos en la simulación con condiciones de operación reales.

Finalmente, con el sistema propuesto se demuestra que un controlador SDN enriquecido con algoritmos de aprendizaje automático puede adaptarse dinámicamente, prevenir congestión y optimizar el balanceo de tráfico en tiempo real, incluso en escenarios exigentes con flujos heterogéneos. Los resultados validan plenamente la hipótesis formulada y consolidan al enfoque KDN como una alternativa sólida para el diseño de redes programables de nueva generación.

REFERENCIAS

[1] A. N. Rage, "The Cloud Computing: A Review Paper," *Architecture Image Studies*, vol. 6, no. 3, pp. 980–989, Nov. 2025, doi: 10.62754/AIS.V6I3.368.

[2] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," *Computer Communication Review*, vol. 38, no. 4, pp. 63–74, Aug. 2008, doi: 10.1145/1402946.1402967;SERIALTOPIC:TOPIC:ACM-PUBTYPE.

[3] C. Guo, H. Wu, K. Tan, L. Shi, Y. Zhang, and S. Lu, "Dcell," pp. 75–86, Aug. 2008, doi: 10.1145/1402958.1402968.

[4] A. Wieder, P. Bhatotia, A. Post, and R. Rodrigues, *Jellyfish: Networking Data Centers Randomly*. 2012. Accessed: Dec. 07, 2025. [Online]. Available: http://ec.europa.eu/eurostat/statistics-explained/index.php/International_trade_in_goods

[5] H. K. Dhaliwal and C. H. Lung, "Load Balancing Using ECMP in Multi-Stage Clos Topology in a Datacenter," *DSC 2018 - 2018 IEEE Conference on Dependable and Secure Computing*, Jan. 2019, doi: 10.1109/DESEC.2018.8625147.

[6] W. Xia, Y. Wen, C. H. Foh, D. Niyato, and H. Xie, "A Survey on Software-Defined Networking," *IEEE Communications Surveys and Tutorials*, vol. 17, no. 1, pp. 27–51, Jan. 2015, doi: 10.1109/COMST.2014.2330903.

[7] A. Mestres *et al.*, "Knowledge-Defined Networking," *ACM SIGCOMM Computer Communication Review*, vol. 47, no. 3, pp. 1–10, Sep. 2017, doi: 10.1145/3138808.3138810.

[8] T. Balharith and F. Alhaidari, "Round Robin Scheduling Algorithm in CPU and Cloud Computing: A review," *2nd International Conference on Computer Applications and Information Security, ICCAIS 2019*, May 2019, doi: 10.1109/CAIS.2019.8769534.

[9] S. Sinha, S. Kandula, and D. Katabi, "Harnessing TCP's Burstiness with Flowlet Switching".

[10] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat, "Hedera: Dynamic Flow Scheduling for Data Center Networks".

[11] S. Ghorbani, Z. Yang, P. B. Godfrey, Y. Ganjali, and A. Firoozshahian, "Drill: Micro load balancing for low-latency data center networks," *SIGCOMM 2017 - Proceedings of the 2017 Conference of the ACM Special Interest Group on Data Communication*, vol. 14, pp. 225–238, Aug. 2017, doi: 10.1145/3098822.3098839;GROUPTOPIC:TOPIC:ACM-PUBTYPE.

[12] F. Tariq, “Micro load balancing with delayed queue lengths,” Apr. 2019, Accessed: Dec. 09, 2025. [Online]. Available: https://hdl.handle.net/2142/105070

[13] J. Singh and N. K. Walia, “A Comprehensive Review of Cloud Computing Virtual Machine Consolidation,” *IEEE Access*, vol. 11, pp. 106190–106209, 2023, doi: 10.1109/ACCESS.2023.3314613.

[14] Q. Zhang *et al.*, “Scalable and Robust East-West Forwarding Framework for Hyperscale Clouds,” *IEEE/ACM Transactions on Networking*, vol. 31, no. 6, pp. 3063–3079, Dec. 2023, doi: 10.1109/TNET.2023.3269772.

[15] J. Alqahtani, S. Alanazi, and B. Hamdaoui, “Traffic Behavior in Cloud Data Centers: A Survey,” *2020 International Wireless Communications and Mobile Computing, IWCMC 2020*, pp. 2106–2111, Jun. 2020, doi: 10.1109/IWCMC48107.2020.9148470.

[16] Z. Han and L. Yu, “A Survey of the BCube Data Center Network Topology,” *Proceedings - 4th IEEE International Conference on Big Data Security on Cloud, BigDataSecurity 2018, 4th IEEE International Conference on High Performance and Smart Computing, HPSC 2018 and 3rd IEEE International Conference on Intelligent Data and Securi…*, pp. 229–231, Nov. 2018, doi: 10.1109/BDS/HPSC/IDS18.2018.00056.

[17] T. Benson, A. Anand, A. Akella, and M. Zhang, “Understanding data center traffic characteristics,” *ACM SIGCOMM Computer Communication Review*, vol. 40, no. 1, pp. 92–99, 2010.

[18] M. Alizadeh *et al.*, “DCTCP: Efficient Packet Transport for the Commoditized Data Center,” Jan. 01, 2010. Accessed: May 12, 2026. [Online]. Available: https://www.microsoft.com/en-us/research/publication/dctcp-efficient-packet-transport-for-the-commoditized-data-center/

[19] F. Yan *et al.*, “Network traffic characteristics of hyperscale data centers in the era of cloud applications,” *Journal of Optical Communications and Networking, Vol. 15, Issue 10, pp. 736-749*, vol. 15, no. 10, pp. 736–749, Oct. 2023, doi: 10.1364/JOCN.494291.

[20] B. Nougnanke, Y. Labit, M. Bruyere, U. Aivodji, and S. Ferlin, “ML-Based Performance Modeling in SDN-Enabled Data Center Networks,” *IEEE Transactions on Network and Service Management*, vol. 20, no. 1, pp. 815–829, Mar. 2023, doi: 10.1109/TNSM.2022.3197789.

[21] G. Wassie Geremew and J. Ding, “Elephant Flows Detection Using Deep Neural Network, Convolutional Neural Network, Long Short-Term Memory, and Autoencoder,” *Journal of Computer Networks and Communications*, vol. 2023, no. 1, p. 1495642, Jan. 2023, doi: 10.1155/2023/1495642.

[22] HeKeqiang, RoznerEric, AgarwalKanak, FelterWes, CarterJohn, and AkellaAditya, “Presto,” *ACM SIGCOMM Computer Communication Review*, vol. 45, no. 4, pp. 465–478, Aug. 2015, doi: 10.1145/2829988.2787507.

[23] R. Zhang-Shen and N. McKeown, “Designing a Fault-Tolerant Network Using Valiant Load-Balancing,” pp. 2360–2368, Jun. 2008, doi: 10.1109/INFOCOM.2008.305.

[24] M. Mitzenmacher, “The power of two choices in randomized load balancing,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 12, no. 10, pp. 1094–1104, Oct. 2001, doi: 10.1109/71.963420.

[25] S. Jain *et al.*, “B4: Experience with a globally-deployed software defined WAN,” *Computer Communication Review*, vol. 43, no. 4, pp. 3–14, Aug. 2013, doi: 10.1145/2534169.2486019;TAXONOMY:TAXONOMY:ACM-PUBTYPE;PAGEGROUP:STRING:PUBLICATION.

[26] A. Roy, H. Zeng, J. Bagga, G. Porter, and A. C. Snoeren, “Inside the Social Network’s (Datacenter) Network,” *Computer Communication Review*, vol. 45, no. 4, pp. 123–137, Aug. 2015, doi: 10.1145/2785956.2787472;PAGE:STRING:ARTICLE/CHAPTER.

[27] M. Dorigo and T. Stützle, “Ant Colony Optimization,” *LNCS*, vol. 5467, no. 2, pp. 2–2, 2009, doi: 10.1007/978-3-642-01020-0_2.

[28] T. M. Shami, A. A. El-Saleh, M. Alswaitti, Q. Al-Tashi, M. A. Summakieh, and S. Mirjalili, “Particle Swarm Optimization: A Comprehensive Survey,” *IEEE Access*, vol. 10, pp. 10031–10061, 2022, doi: 10.1109/ACCESS.2022.3142859.

[29] A. Sohail, “Genetic Algorithms in the Fields of Artificial Intelligence and Data Sciences,” *Annals of Data Science 2021 10:4*, vol. 10, no. 4, pp. 1007–1018, Aug. 2021, doi: 10.1007/S40745-021-00354-9.

[30] A. Graves, “Long Short-Term Memory,” pp. 37–45, 2012, doi: 10.1007/978-3-642-24797-2_4.

[31] R. L. Abduljabbar, H. Dia, and P. W. Tsai, “Unidirectional and Bidirectional LSTM Models for Short-Term Traffic Prediction,” *J. Adv. Transp.*, vol. 2021, no. 1, p. 5589075, Jan. 2021, doi: 10.1155/2021/5589075.

[32] J. Fang, “Clustering and Path Planning for Wireless Sensor Networks based on Improved Ant Colony Algorithm.,” *International Journal of Online & Biomedical Engineering*, vol. 15, no. 1, p. 129, Jan. 2019, doi: 10.3991/IJOE.V15I01.9784.

[33] X. Wang *et al.*, “Deep Reinforcement Learning: A Survey,” *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 35, no. 4, pp. 5064–5078, Apr. 2024, doi: 10.1109/TNNLS.2022.3207346.

[34] H. Tan, “Reinforcement Learning with Deep Deterministic Policy Gradient,” *Proceedings - 2021 International Conference on Artificial Intelligence, Big Data and Algorithms, CAIBDA 2021*, pp. 82–85, May 2021, doi: 10.1109/CAIBDA53561.2021.00025.

[35] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, “Deep reinforcement learning: A brief survey,” *IEEE Signal Process. Mag.*, vol. 34, no. 6, pp. 26–38, Nov. 2017, doi: 10.1109/MSP.2017.2743240.

[36] D. C. Nguyen *et al.*, “Federated Learning Meets Blockchain in Edge Computing: Opportunities and Challenges,” *IEEE Internet Things J.*, vol. 8, no. 16, pp. 12806–12825, Aug. 2021, doi: 10.1109/JIOT.2021.3072611.

[37] G.-G. Wang and Y. Tan, “Improving Metaheuristic Algorithms With Information Feedback Models,” *IEEE Trans. Cybern.*, vol. 49, no. 2, pp. 542–555, Dec. 2017, doi: 10.1109/TCYB.2017.2780274.

[38] E. Gilliard, J. Liu, A. A. Aliyu, D. Juan, H. Jing, and M. Wang, “Intelligent load balancing in data center software-defined networks,” *Transactions on Emerging Telecommunications Technologies*, vol. 35, no. 4, p. e4967, Apr. 2024, doi: 10.1002/ETT.4967;PAGE:STRING:ARTICLE/CHAPTER.

[39] J. Hu, W. Luo, Y. He, J. Wang, and D. Zhang, “Deep Reinforcement Learning Based Load Balancing for Heterogeneous Traffic in Datacenter Networks,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 14489 LNCS, pp. 270–289, 2024, doi: 10.1007/978-981-97-0798-0_16/SAVE-RESEARCH.

[40] Y. LIU, C. WANG, and C. LI, “Survey on Traffic Engineering in Software-Defined Networking Based on Reinforcement Learning.,” *Journal of Computer Engineering & Applications*, vol. 61, no. 24, p. 1, Dec. 2025, doi: 10.3778/J.ISSN.1002-8331.2412-0248.

[41] A. Guo and C. Yuan, “Network Intelligent Control and Traffic Optimization Based on SDN and Artificial Intelligence,” *Electronics 2021, Vol. 10, Page 700*, vol. 10, no. 6, p. 700, Mar. 2021, doi: 10.3390/ELECTRONICS10060700.

[42] H. Xue, K. T. Kim, and H. Y. Youn, “Dynamic Load Balancing of Software-Defined Networking Based on Genetic-Ant Colony Optimization,” *Sensors 2019, Vol. 19, Page 311*, vol. 19, no. 2, p. 311, Jan. 2019, doi: 10.3390/S19020311.

[43] J. Chen *et al.*, “ALBRL: Automatic Load-Balancing Architecture Based on Reinforcement Learning in Software-Defined Networking,” *Wirel. Commun. Mob. Comput.*, vol. 2022, no. 1, p. 3866143, Jan. 2022, doi: 10.1155/2022/3866143.

[44] X. Huang, M. Zeng, and K. Xie, “Intelligent traffic control for QoS optimization in hybrid SDNs,” *Computer Networks*, vol. 189, p. 107877, Apr. 2021, doi: 10.1016/J.COMNET.2021.107877.

[45] P. Sun, J. Lan, Z. Guo, Y. Xu, and Y. Hu, “Improving the scalability of deep reinforcement learning-based routing with control on partial nodes,” *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, vol. 2020-May, pp. 3557–3561, May 2020, doi: 10.1109/ICASSP40776.2020.9054483.

[46] V. Srivastava and R. S. Pandey, “Machine intelligence approach: To solve load balancing problem with high quality of service performance for multi-controller based Software Defined Network,” *Sustainable Computing: Informatics and Systems*, vol. 30, p. 100511, Jun. 2021, doi: 10.1016/J.SUSCOM.2021.100511.

[47] H. Yao, X. Yuan, P. Zhang, J. Wang, C. Jiang, and M. Guizani, “Machine learning aided load balance routing scheme considering queue utilization,” *IEEE Trans. Veh. Technol.*, vol. 68, no. 8, pp. 7987–7999, Aug. 2019, doi: 10.1109/TVT.2019.2921792.

[48] J. Zhang, M. Ye, Z. Guo, C. Y. Yen, and H. J. Chao, “CFR-RL: Traffic Engineering with Reinforcement Learning in SDN,” *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 10, pp. 2249–2259, Oct. 2020, doi: 10.1109/JSAC.2020.3000371.

[49] S. Liang, W. Jiang, F. Zhao, and F. Zhao, “Load Balancing Algorithm of Controller Based on SDN Architecture Under Machine Learning,” *Journal of Systems Science and Information*, vol. 8, no. 6, pp. 578–588, Dec. 2020, doi: 10.21078/JSSI-2020-578-11/XML.

[50] Y. Li, P. K. Donta, X. Wang, I. Murturi, M. Huang, and S. Dustdar, “KDN-FLB: Knowledge-Defined Networking Through Federated Learning and Blockchain,” *Computer (Long. Beach. Calif).*, vol. 58, no. 5, pp. 16–26, 2025, doi: 10.1109/MC.2024.3471984.

[51] B. Lantz, B. Heller, and N. McKeown, “A network in a laptop: Rapid prototyping for software-defined networks,” *Proceedings of the 9th ACM Workshop on Hot Topics in Networks, Hotnets-9*, Oct. 2010, doi: 10.1145/1868447.1868466;CSUBTYPE:STRING:CONFERENCE.

[52] A. Varga and R. Hornig, “AN OVERVIEW OF THE OMNeT++ SIMULATION ENVIRONMENT”.

[53] T. Henderson, “Network Simulations with the ns-3 Simulator,” 2008.

[54] P. A. Lopez *et al.*, “Microscopic Traffic Simulation using SUMO,” *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, vol. 2018-November, pp. 2575–2582, Dec. 2018, doi: 10.1109/ITSC.2018.8569938.

[55] N. Gupta, M. S. Maashi, S. Tanwar, S. Badotra, M. Aljebreen, and S. Bharany, “A Comparative Study of Software Defined Networking Controllers Using Mininet,” *Electronics 2022, Vol. 11, Page 2715*, vol. 11, no. 17, p. 2715, Aug. 2022, doi: 10.3390/ELECTRONICS11172715.

[56] S. Bachmeier, B. Jaeger, and K. Holzinger, “Network Simulation with OMNet++”, doi: 10.2313/NET-2020-11-1_08.

[57] S. Akhter, M. Nurul Ahsan, S. Jafor Sadeek Quaderi, M. Abdullah Al Forhad, S. H. Sumit, and M. Rahatur Rahman, “A SUMO Based Simulation Framework for Intelligent Traffic Management System”, doi: 10.18178/jtle.8.1.1-5.

[58] C. Dann, Y. Mansour, M. Mohri, A. Sekhari, and K. Sridharan, “Guarantees for Epsilon-Greedy Reinforcement Learning with Function Approximation,” Jun. 28, 2022, *PMLR*. Accessed: Dec. 07, 2025. [Online]. Available: https://proceedings.mlr.press/v162/dann22a.html

[59] B. Saglam, F. B. Mutlu, D. C. Cicek, and S. S. Kozat, “Actor Prioritized Experience Replay,” *Journal of Artificial Intelligence Research*, vol. 78, pp. 639–672, Nov. 2023, doi: 10.1613/JAIR.1.14819.

[60] V. Kumar and M. Webster, “Importance Sampling based Exploration in Q Learning,” Jul. 2021, Accessed: Dec. 07, 2025. [Online]. Available: https://arxiv.org/pdf/2107.00602

[61] N. Mohi Ud Din, A. Assad, S. Ul Sabha, and M. Rasool, “Optimizing deep reinforcement learning in data-scarce domains: a cross-domain evaluation of double DQN and dueling DQN,” *International Journal of System Assurance Engineering and Management 2024*, pp. 1–12, May 2024, doi: 10.1007/S13198-024-02344-5.

[62] Y. Xu and H. Zhang, “Convergence of deep ReLU networks,” *Neurocomputing*, vol. 571, p. 127174, Feb. 2024, doi: 10.1016/J.NEUCOM.2023.127174.

[63] X. Xu, X. Li, N. Chen, D. Zhao, and C. Chen, “Autonomous Obstacle Avoidance with Improved Deep Reinforcement Learning Based on Dynamic Huber Loss.,” *Applied Sciences (2076-3417)*, vol. 15, no. 5, p. 2776, Mar. 2025, doi: 10.3390/APP15052776.

[64] C. Liu *et al.*, “Adaptive Smooth L1 Loss: A Better Way to Regress Scene Texts with Extreme Aspect Ratios,” *Proc. IEEE Symp. Comput. Commun.*, vol. 2021-September, 2021, doi: 10.1109/ISCC53001.2021.9631466.

[65] A. Al-Kababji, F. Bensaali, and S. P. Dakua, “Scheduling Techniques for Liver Segmentation: ReduceLROnPlateau vs OneCycleLR,” *Communications in Computer and Information Science*, vol. 1589 CCIS, pp. 204–212, 2022, doi: 10.1007/978-3-031-08277-1_17.

[66] Z. Wang, T. Schaul, M. Hessel, H. Hasselt, M. Lanctot, and N. Freitas, “Dueling Network Architectures for Deep Reinforcement Learning,” Jun. 11, 2016, *PMLR*. Accessed: Dec. 09, 2025. [Online]. Available: https://proceedings.mlr.press/v48/wangf16.html

[67] H. Van Hasselt, A. Guez, and D. Silver, “Deep Reinforcement Learning with Double Q-Learning,” *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 30, no. 1, pp. 2094–2100, Mar. 2016, doi: 10.1609/AAAI.V30I1.10295.

[68] T. Schaul, J. Quan, I. Antonoglou, D. Silver, and G. Deepmind, “Prioritized Experience Replay,” *4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings*, pp. 1–21, Nov. 2015, Accessed: Dec. 09, 2025. [Online]. Available: https://arxiv.org/pdf/1511.05952

[69] H. A. Miot, “Avaliação da normalidade dos dados em estudos clínicos e experimentais,” *J. Vasc. Bras.*, vol. 16, no. 2, pp. 88–91, 2017, doi: 10.1590/1677-5449.041117.

[70] R. Soni and A. K. Pathak, “Generalized Iterated Poisson Process and Applications,” *Journal of Theoretical Probability 2024 37:4*, vol. 37, no. 4, pp. 3216–3245, Aug. 2024, doi: 10.1007/S10959-024-01362-0.