NEXT-GENERATION DATA CENTERS: EXPERIMENTAL ANALYSIS OF TOPOLOGIES AND ALGORITHMS FOR NEXT-GENERATION DATA CENTERS

Abdeltif Azzizi

A thesis
in
The Department
of
Computer Science and Software Engineering

Presented in Partial Fulfillment of the Requirements
For the Degree of Master of Computer Science (Computer Science)
Concordia University
Montréal, Québec, Canada

July 2025
© Abdeltif Azzizi, 2025

Concordia University
School of Graduate Studies

This is to certify that the thesis prepared

By: **Abdeltif Azzizi**

Entitled: **Next-Generation Data Centers: Experimental Analysis of Topologies and Algorithms for Next-Generation Data Centers**

and submitted in partial fulfillment of the requirements for the degree of

Master of Computer Science (Computer Science)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

_____ Chair
Dr. Abdelhak Bentaleb

_____ Examiner
Dr. Abdelhak Bentaleb

_____ Examiner
Dr. Roch Glitho

_____ Supervisor
Dr. Chadi Assi

Approved by _____
Dr. Denis Pankratov, Graduate Program Director

_____ 2025 _____

Dr. Mourad Debbabi, Dean
Faculty of Engineering and Computer Science

Abstract

Next-Generation Data Centers: Experimental Analysis of Topologies and Algorithms for Next-Generation Data Centers

Abdeltif Azzizi

In recent years, data center networks (DCNs) have faced growing pressure from AI and ML workloads with intensive communication patterns and stringent latency requirements. Traditional hierarchical architectures like Clos (Fat-Tree) increasingly struggle with scalability bottlenecks, operational complexity, and congestion under bursty traffic. To address these challenges, this work explores the Structured Re-Arranged Topology (STRAT), which combines expander-graph-inspired path diversity with deterministic structure to enable efficient, scalable, and fault-tolerant designs. Unlike rigid designs, STRAT supports incremental growth, reduced cabling complexity, and better load distribution. This thesis evaluates STRAT not only in simulation but also on real programmable-switch hardware, demonstrating its practical viability. A key contribution is DEALER, a congestion-aware, data-plane-friendly forwarding algorithm leveraging programmable switches. DEALER uses a distributed distance-vector protocol and local queue occupancy to balance load among equal-cost and slightly longer paths, achieving significant improvements over ECMP in high-load scenarios while running at line rate on commercial ASICs. To further enhance STRAT, this work integrates a hybrid electrical-optical fabric with Optical Circuit Switching (OCS) links, guided by proactive, ML-based flow classification. An XGBoost model predicts elephant flows early, enabling their diversion to pre-configured optical paths with minimal control overhead. Simulations show reductions in tail latency and improved throughput. Together, these contributions offer a practical, holistic redesign for DCNs, uniting scalable graph-based topologies, programmable forwarding logic, and ML-guided optical hybridization to meet the performance, efficiency, and scalability demands of modern AI and cloud workloads.

Acknowledgments

The completion of this thesis marks the culmination of a long, challenging, and rewarding journey—one that would have been impossible without the steadfast support, patience, and encouragement of many wonderful people.

To my parents, Lalla Khadija El Hassani and Boubker Azzizi: Your unwavering belief in my abilities has been my anchor. Thank you for the countless sacrifices, for celebrating every small victory, and for reminding me—on the difficult days—that perseverance and passion always prevail. You are the reason this work exists, thank you for everything.

To my siblings, Ayoub, Layla, and Ilyass: Thank you for your constant encouragement and sacrifices, all the work has been motivated by you and for you. Your support has meant more than words can say.

To Laaraichi family Your kindness and generosity when I first arrived made all the difference. Thank you for welcoming me as one of your own, for helping me settle in, and for showing me such genuine care and hospitality.

To my supervisor, Dr. Chadi Assi: Your insightful guidance, high standards, and patient mentorship pushed me to refine my ideas and elevate this thesis far beyond what I first imagined. Thank you for challenging me to think critically, for offering constructive feedback at every stage, and for sharing your deep expertise so generously.

Each of you has shaped this work in meaningful ways. I am deeply grateful for your support and honored to have you alongside me in this journey.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

As the digital economy expands, the demand for more efficient and scalable data center topologies continues to rise. Modern data centers must not only handle large volumes of data but also provide the infrastructure necessary for rapid data processing and retrieval. This demand is fueled by the proliferation of cloud services, big data analytics, and artificial intelligence. The exponential growth in artificial intelligence (AI) applications significantly impacts data center operations. AI workloads, characterized by intensive data processing and the need for real-time analytics, require data centers to be exceptionally responsive and adaptable. Addressing these needs is crucial for maintaining system efficiency and ensuring that the infrastructure can manage the increasing complexity and volume of both traditional and AI-driven tasks, while being constrained by cost, space and power.

1.2 Motivation

Modern data centers are the backbone of the digital economy, powering applications in cloud computing, big data, and artificial intelligence. However, the exponential increase in computation and data traffic has revealed significant challenges in traditional data center designs. Issues such as scalability bottlenecks, high energy consumption, complex management, and the need for predictable performance under diverse workloads are increasingly critical [15, 39, 42]. These problems are amplified by the shift

toward service-oriented architectures and the growing diversity of applications running on virtualized platforms [27, 37]. Additionally, evolving architectural trends such as full disaggregation, programmable infrastructure, and AI-driven automation are reshaping how data centers are built and operated [30, 41].

To address these challenges, there is a growing emphasis on rethinking network topologies, integrating intelligent routing algorithms, and leveraging hybrid electrical-optical fabrics. This thesis is motivated by the urgent need to explore and evaluate novel data center network designs that can meet the scalability, performance, and energy-efficiency demands of next-generation workloads. By focusing on experimental validation of expander-based architectures and machine-learning-enhanced hybrid switching, this work contributes toward bridging the gap between theoretical innovation and practical deployment.

1.3 Thesis Contributions

Building upon the motivation outlined in the preceding section, this thesis makes the following key contributions to the advancement of scalable, resilient, and cost-effective data center network architectures:

1.3.1 Experimental Evaluation and Validation of Expander-Based STRAT Topologies

Chapter 2 presents a comprehensive experimental and simulation-based evaluation of STRAT, an expander-graph-inspired flat topology, positioned as a practical alternative to conventional Clos architectures in data center networks. First, we construct a real-world testbed using commercial off-the-shelf Broadcom Trident4 switches to model both STRAT and Clos topologies at scale. This testbed enables rigorous performance evaluation under realistic conditions. Second, we conduct architectural and metric-based comparisons through OMNeT++ simulations and hardware testing. STRAT consistently outperforms Clos in key metrics such as throughput (up to 43% improvement), switch count (about 40% fewer), and traffic distribution under load. Finally, we introduce DEALER—a lightweight, programmable routing algorithm tailored for STRAT—which leverages expander path diversity and local queue

awareness. DEALER achieves significantly lower drop ratios and better load balancing compared to traditional ECMP routing, and is validated both in simulation and on hardware.

1.3.2 A Hybrid Optical-Electrical STRAT Architecture with ML-Driven Flow Classification

Chapter 3 proposes a novel hybrid architecture that augments STRAT with a sparse overlay of Optical Circuit Switching (OCS) links to enhance throughput and manage long-lived flows more effectively. Inspired by communication patterns found in distributed deep learning, we overlay structured optical rings that offload elephant flows while retaining mice flows within the electrical STRAT fabric. To support this design, we incorporate an XGBoost-based classifier trained on the UNIV1 dataset to distinguish mice from elephant flows using early packet-level features. The model generalizes well across workloads, achieving an F1 score of approximately 0.91–0.93, and enables real-time traffic steering. The hybrid STRAT architecture is implemented and evaluated in OMNeT++, demonstrating significant improvements in latency, packet delivery, and throughput under diverse traffic conditions compared to purely electrical designs.

1.4 Thesis Organization

This thesis is structured as follows. Chapter 1 introduces the background and motivation, detailing the limitations of current hierarchical and electrical-only data center architectures, and presenting the case for exploring flat and hybrid alternatives. Chapter 2 focuses on the design, implementation, and evaluation of STRAT, showcasing its architectural benefits and validating its performance through both physical testbed experiments and simulations. Chapter 3 builds upon this by introducing a hybrid STRAT variant enhanced with optical links and machine learning-based flow classification, with results from extensive OMNeT++ simulations. Finally, Chapter 4 concludes the thesis by summarizing key findings and outlining directions for future research in scalable, programmable data center networking.

Chapter 2

Experimental Evaluation and Validation of Expander-Based STRAT Topologies

2.1 Introduction

This chapter presents an experimental and simulation-based evaluation of two data center network (DCN) topologies: the widely adopted hierarchical Clos architecture and STRAT, a flat, expander-based topology designed around passive optical interconnects. While Clos offers proven scalability and performance, it incurs hardware complexity and suffers from congestion in oversubscribed scenarios. STRAT, by contrast, eliminates aggregation and spine layers entirely—using only Top-of-Rack (ToR) switches interconnected via static optical patch panels—to reduce cost, simplify deployment, and enhance path diversity. Our goal is to assess these topologies based on their inherent architectural properties—namely throughput, congestion resilience, scalability, and cost—without relying on congestion control protocols or centralized traffic engineering. To this end, we adopt simple forwarding schemes based purely on local information: ECMP for Clos, and ECMP with Dynamic Group Multipath (DGM) for STRAT. We evaluate both topologies on a physical testbed built from commercial Ethernet switches and further validate scalability through packet-level simulations of networks with up to 256 switches and 1,024 hosts using OMNeT++. We also introduce DEALER, a lightweight routing algorithm tailored to STRAT's

topology, and evaluate its effectiveness in dynamic conditions. Our results show that STRAT achieves up to 43% higher throughput and requires approximately 40% fewer switches than a comparable Clos topology. These gains are further supported by Load Area Under Curve (LAUC) analysis and congestion hotspot visualizations.

2.2 Related Works

In data center network design, Jellyfish [44] topology has been proposed as an innovative alternative to a conventional fat-tree. Jellyfish utilizes a random regular graph, which proves to be cost-effective but introduces several challenges, particularly in routing and scalability. Routing in Jellyfish is complex due to its random structure, making common strategies like shortest path routing and equal-cost multi-path routing (ECMP) ineffective. Enhanced routing can be achieved using k-shortest path routing (KSP) with heuristics such as randomization and edge-disjointness to improve path diversity [4]. Scalability, while generally better than fat-trees, poses issues for smaller systems where fat-trees may be more effective due to their structured design [60]. Additionally, handling link failures in Jellyfish requires sophisticated algorithms to ensure robust backup paths to maintain connectivity, further complicating its deployment.

VL2 [18] is a network architecture tailored for dynamic resource allocation within data centers, facilitating both agility and cost efficiency by supporting large server pools. This architecture introduces a uniform high capacity between servers, performance isolation, and adheres to Ethernet layer-2 semantics. Despite these benefits, VL2 faces several challenges, including a complex setup due to its reliance on flat addressing and Valiant Load Balancing (VLB), which can complicate network management. Additionally, VL2's functionality hinges significantly on end-system modifications for address resolution, potentially restricting network flexibility and increasing administrative overhead. Furthermore, the deployment of VL2 may involve substantial costs and specific high-speed hardware requirements, which may not be feasible for all data centers to implement immediately [17].

DCube [21] is a network architecture for connecting servers in containerized data

centers using dual-port servers and low-end switches. It forms a hypercube-like structure, improving throughput and resilience to server or switch failures. While it reduces infrastructure costs, DCube excels mainly for specific traffic patterns compared to BCube. DCell [20], a scalable and fault-tolerant architecture, improves capacity, load balancing, and fault tolerance. However, its complexity increases with scale, leading to management challenges, traffic imbalances, and congestion issues. Additionally, high node degrees and complex routing protocols complicate implementation and maintenance [12].

FIConn [31] is a data center networking topology that utilizes both Ethernet ports on servers — traditionally one for active connections and the other for backup — to create a scalable, cost-effective network structure. However, it has notable limitations, including challenges in network expansion and higher costs as the scale of operations increases. Moreover, FIConn's design limits server node degrees to two, which can restrict performance in large-scale deployments due to insufficient connectivity and fault tolerance. More advanced alternatives like BCCC [33] have been shown to offer better expandability and efficiency, highlighting areas where FIConn could be improved [32]. The Dragonfly+ [43] topology is a hierarchical network architecture designed to improve upon the scalability and efficiency limitations of traditional Fat Tree and original Dragonfly networks. It organizes routers into groups, where intra-group connectivity follows a bipartite structure between leaf and spine routers, while inter-group connectivity forms a completely connected graph through global links. This hybrid approach combines the benefits of flattened topologies with structured routing, reducing the number of required global links and enhancing path diversity. Dragonfly+ uses a novel routing algorithm called Fully Progressive Adaptive Routing (FPAR), which dynamically chooses between minimal and non-minimal paths based on local queue lengths and congestion status. Additionally, Dragonfly+ improves deadlock avoidance by using just two virtual lanes (VLs), compared to three or more in previous designs, which helps reduce buffer fragmentation and simplifies hardware requirements. Analytically, Dragonfly+ achieves better scalability, supporting up to 105,000 hosts using 36-port routers, and delivers at least 50% bi-sectional bandwidth utilization for arbitrary traffic patterns. It also maintains a low network diameter (typically 3 hops), ensuring low latency across a wide range of workloads. However, its effectiveness depends heavily on balanced inter-group connectivity and careful

traffic engineering. Drawbacks include increased complexity in routing management due to the need for VL remapping, congestion notifications, and adaptive decision-making at each router. Physical deployment can also be intricate due to the global link design and the need to maintain full bi-sectional bandwidth across all groups, making wiring and layout planning more challenging in large-scale environments. The HyperX [38] topology is a direct network derived from the hypercube, designed to leverage high-radix switches for scalable and low-latency communication in datacenter and high-performance computing environments. HyperX arranges switches in a multi-dimensional lattice, where each node is identified by a coordinate vector and is connected to other switches that differ by a single coordinate. This geometric structure enables rich path diversity and fault tolerance, as multiple link-disjoint and node-disjoint paths can be established between any two nodes. The topology supports both shortest and near-shortest paths, which can be used adaptively to balance load or recover from failures. In failure scenarios, HyperX facilitates both protection (precomputed backup paths) and restoration (on-the-fly rerouting) schemes, ensuring high survivability even under multiple link faults.

HyperX is characterized by its high path redundancy, low average hop count, and strong resilience to link and switch failures. The number of available shortest paths between two nodes is given by $\Delta!$, where Δ is the number of dimensions in which their coordinates differ. This extensive path diversity allows robust multipath routing and enhances load balancing capabilities. However, a key drawback of HyperX is the increased routing complexity: computing and maintaining large sets of disjoint paths requires significant overhead, especially in dynamic or large-scale networks. Additionally, its physical implementation is nontrivial due to the multidimensional layout, which leads to non-intuitive cabling and potentially higher latency if the dimensions are not uniformly populated.

The fat-tree based on a Clos topology, widely used in high-performance computing and data center networks, offers uniform high capacity and efficient adaptive routing. However, it has significant limitations in scalability and complexity. Its physical layout requires many switches and complex wiring, increasing costs and power consumption. As the network scales, the switch radix does not keep pace with the growth in endpoints, leading to longer paths and higher latency for sensitive applications [36]. Additionally, managing fault tolerance becomes more difficult as network

size increases, with more components prone to failure [23].

2.2.1 STRAT: STructured Re-Arranged Topology

STRAT [14] introduces a fundamentally different approach to data center network design compared to traditional hierarchical and flattened topologies such as Clos, Dragonfly, Dragonfly+, HyperX, and Flattened Clos. Unlike these architectures—which typically rely on multi-tiered designs with dedicated aggregation and spine layers—STRAT eliminates the hierarchy entirely by using only Top-of-Rack (ToR) switches. Each ToR in STRAT serves a dual role: it multiplexes local server traffic and forwards transit traffic for other ToRs, effectively functioning as both an edge and core switch. The core of STRAT's innovation lies in its use of static, passive optical patch panels to directly interconnect ToRs in a highly meshed expander-like topology. This design eliminates the need for active optical circuit switching or high-radix central routers seen in Dragonfly-class architectures. As a result, STRAT reduces the entire network hardware footprint to a single ToR switch model, dramatically simplifying procurement, deployment, and operational complexity. In contrast to Clos and Dragonfly-like topologies that require extensive cabling, complex fan-out, and high-radix switches to scale, STRAT enables port aggregation into large bundles, reducing transceiver count and fiber complexity. Its passive optical fabric also scales linearly by adding loopback plugs to preinstalled patch panels—without requiring reconfiguration or new switching layers—making it uniquely amenable to incremental scaling from small clusters to thousands of nodes. Furthermore, STRAT is designed for compatibility with off-the-shelf Ethernet ASICs and full packet-based forwarding, without reliance on circuit-switching or traffic reconfiguration. Its rich path diversity and expander connectivity confer strong resilience to congestion and failure, while maintaining cost and energy efficiency. This architectural simplicity, passive scalability, and full deployability with commodity hardware set STRAT apart from prior work, offering a practical and high-performance alternative to hierarchical and semi-hierarchical designs in modern data center networks.

Altogether, STRAT's design focuses on optimizing data center performance and operational efficiency while ensuring scalability, cost-effectiveness, and resilience, marking a significant advancement in network architecture. The work presented in [14]

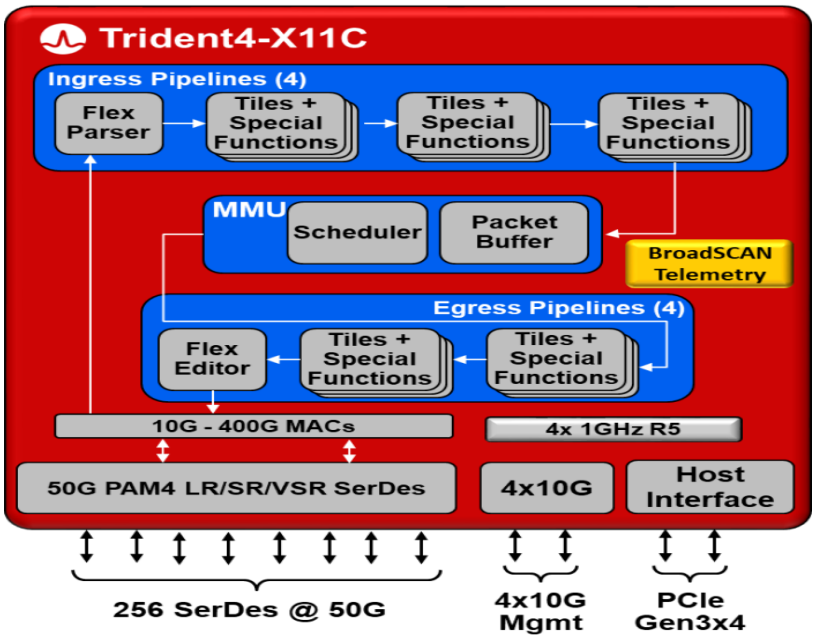

Figure 2.1: Trident4-X11C Architecture

was conducted primarily through simulations, demonstrating its potential data center environments. However, to extend and validate these findings, here we investigate various STRAT topologies against their Clos counterparts using a physical testbed. By transitioning from simulations to physical experimentation, we provide evidence and insights into the real-world performance and practical implications of the STRAT topology. This approach offers a more comprehensive understanding of its efficacy, scalability, and applicability in actual data center deployments and using commercial off-the-shelf Ethernet switching ASICs.

2.3 Trident4®-Based Testbed: Architecture

The advent of Software-Defined Networks (SDN) has fundamentally transformed network architecture. In contrast to traditional networks, where decision-making and data transmission were closely linked, SDN established a distinct separation between the control plane, which handles decision-making, and the data plane, focused on data forwarding. This clear division paved the way for programmable data planes (PDPs), a notable advancement that allows network device functionality to be customized. With PDPs, networks gain greater intelligence and flexibility, enabling

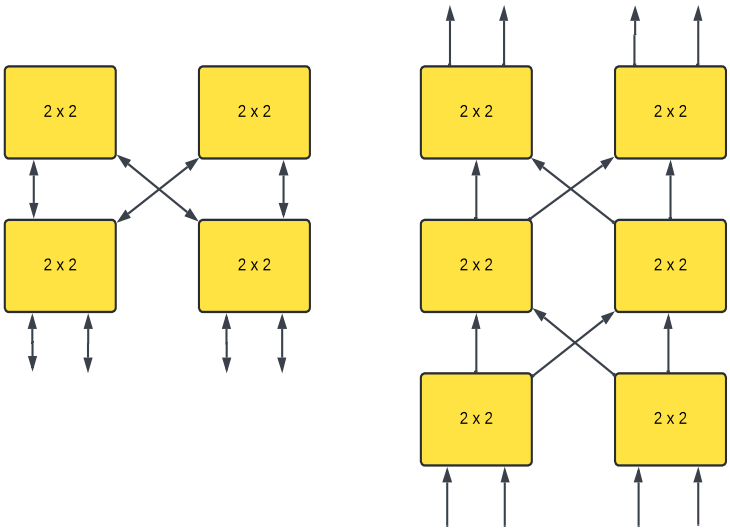

Figure 2.2: Unfolding a 2-layer Clos

enhanced adaptability. To streamline our network modeling, we harness the cutting-edge capabilities of these PDPs, opting for Broadcom's latest off-the-shelf technology, the TD4® programmable switch. Tailored for high-performance Ethernet operations in both data center and enterprise environments as depicted in Fig. 2.1 [10], the TD4® switch represents a significant leap forward. Noteworthy features include 32x400G ports, adaptable telemetry options, extensive forwarding databases customizable to specific application needs, and seamless integration with modern data center protocols. Before we can implement our network topologies, we must tackle the issue of unidirectional switch links. This characteristic necessitates the unfolding of the structure to ensure their alignment with industry standards when deployed on physical switches. Unfolding involves redrawing the topology to accommodate the unidirectional flow of data packets, thereby facilitating an accurate representation of network configurations.

For instance, consider the conventional representation of a 2-Tier Clos topology depicted in Fig. 2.2. Initially conceptualized with bidirectional links, this portrayal fails to acknowledge the unidirectional nature of switch internal communication. Through unfolding, the topology is redrawn to accurately reflect the unidirectional links present in real-world networking scenarios. This transformation ensures that our network designs are not only theoretically robust but also operationally feasible, adhering to

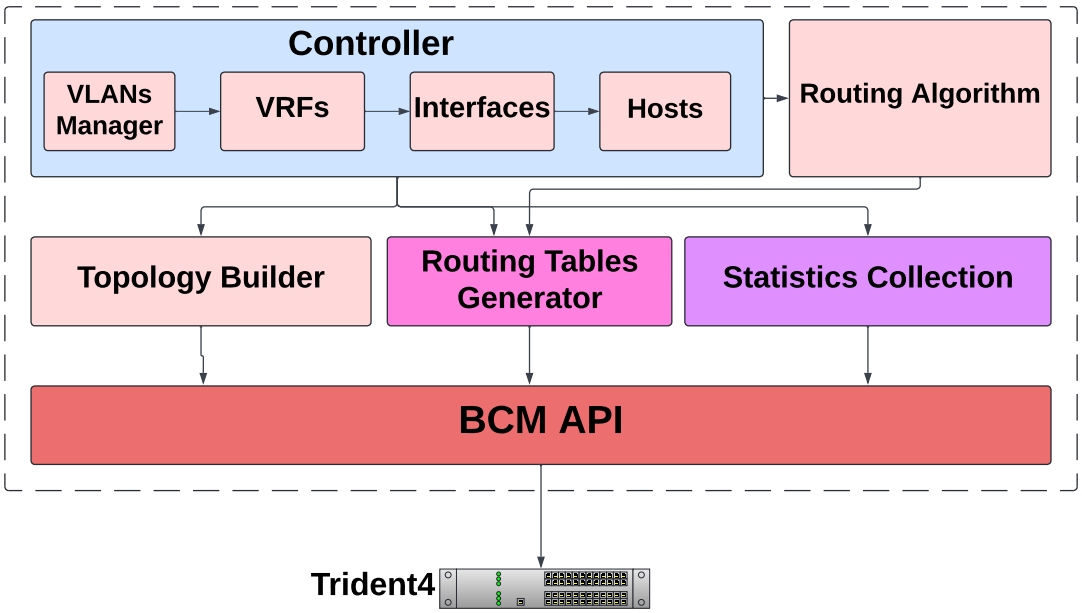

Figure 2.3: Testbed Architecture

established industry standards and practices.

2.4 Testbed Architecture & Design

Our testbed architecture, depicted in Fig. 2.3, is structured hierarchically to facilitate efficient network configuration. The components of the testbed are as follows:

- **Controller**: At the core of the system is a script that functions as the controller, operating at an abstract level to manage configuration tasks and generate network topologies. This controller accepts inputs such as adjacency matrices or similar representations, along with additional parameters like the number of hosts per network switch and routing preferences. Since our setup uses a single physical switch, the controller divides it into multiple virtual switches. Leveraging the TD4 capability to support numerous VLANs, each switch in the topology is represented by a VLAN, with the controller assigning the appropriate ports to each virtual switch. To enable each switch to maintain its own routing table (RT) and routing policies, we utilize Virtual Routing and Forwarding (VRF) identifiers. This allows multiple routing instances to coexist

on the same physical switch, ensuring that independent virtual switches have distinct routing policies and traffic isolation.

- **Topology Builder**: This component is a code module generated by the controller, responsible for implementing all the logic defined during the controller's setup. It contains a suite of helper functions required to model the network topology. One key function it performs is setting all switch ports to loopback mode, ensuring that when a packet exits a port, it re-enters the same port as though it had been transmitted from a different switch. This behavior enables the virtual switches to communicate with each other as if they were distinct physical switches, fully utilizing the queues assigned to each port. In addition to managing loopback operations, the topology builder creates interfaces to facilitate communication between VLANs. It also establishes the necessary Ingress and Egress objects, which are utilized by the routing and forwarding protocols to direct traffic through the virtual network.

- **Routing Algorithm**: The controller triggers the designated routing algorithm's script, which operates on the network's topology data to generate routing tables for every node in the network. These routing tables are subsequently forwarded as input to the Routing Tables Generator. We will cover in detail the routing algorithm used in the next few sections.

- **Routing Tables Generator**: After receiving the routing tables generated by the Routing Algorithm component, the Routing Tables Generator proceeds to create all essential routes required to establish connectivity to every switch within the network. Each route is assigned either an egress object or an Equal-Cost Multipath (ECMP) group containing egress objects. This component operates based on the routing preferences that were given to the controller.

- **Statistics Collection**: This component is responsible for enabling and collecting the various statistics/metrics that we will be using to compare all of our topologies.

These components then communicate with the BCM API to execute this code on TD4® hardware and allow for the creation of an internal topology and the routing chosen for it.

2.5 Investigated Topologies

Our key goal is to compare network topologies and their inherent throughput and resilience against congestion, without the confounding variable of congestion management protocols. Thus, we assume simple forwarding protocols, which make forwarding decisions purely on the basis of local information and with minimum (or none) parameter tuning (ECMP for Clos and ECMP with Dynamic Group Multipath for STRAT as we will discuss in the next section). In order to keep comparison fair, we enforce 2 rules. As much as possible, networks are constructed using switches with the same number of ports. Also, networks service the same number of hosts (traffic sources and sinks).

TD4® accommodates 144 logical ports, which places a limit on the size of investigated topologies. Additionally, the chosen topologies should be sufficiently large to provide meaningful insights across configurations. This rationale guided our selection of the following topologies:

STRAT 14 and its 3-Tier Clos counterpart

We will evaluate a 3-layer Clos topology that fits within 144 ports. Our 3-Layer Clos has 27 servers, 9 TORs, 9 Aggregation and 5 Spine switches (23 switches total with 6 ports each) as shown in Fig. 2.8. This Clos can be compared to a STRAT topology, such as one with a 2:4 (host:network) port configuration requiring 14 switches (TORs) and supporting 28 servers.

We will also evaluate a 2-layer Clos topology with 12 8-port virtual switches (4 spine switches and 8 TORs), as illustrated in Fig. 2.7. A comparable STRAT topology with a 2:6 (host:network) port configuration would require 16 8-port virtual switches (TORs) and support the same number of 32 servers as depicted in Fig. 2.6. Additionally, a STRAT topology with a 3:6 (host:network) port configuration would need 11 switches (TORs) and support 33 servers, as shown in Fig. 2.4 .

Table 2.1 represents a general summary of graph metrics for each of these topologies. The 3-Tier Clos network stands out with the highest number of nodes (23) and edges (54), making it the most complex and interconnected topology. However, this complexity comes with a penalty of increased cost and power, as well as lower density (0.213) and longer average shortest path length (2.285), compared to STRAT 14. The larger diameter (4) and higher average eccentricity (3.391) of the 3-Tier

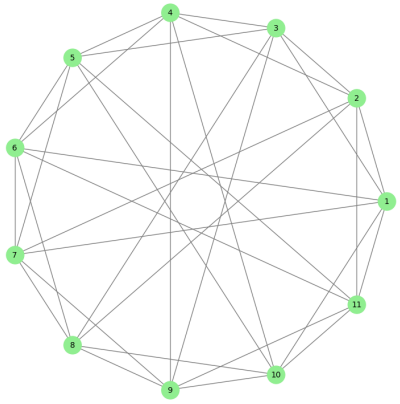

Figure 2.4: STRAT 11

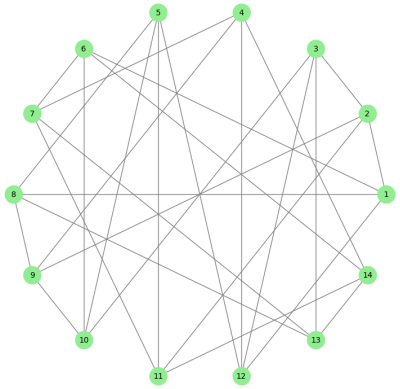

Figure 2.5: STRAT 14

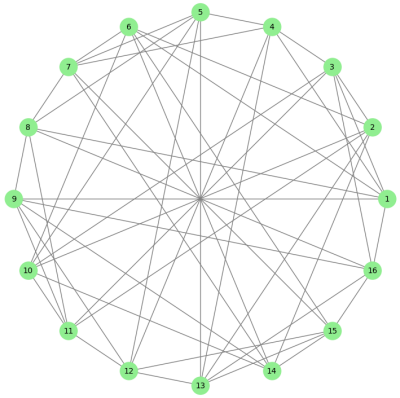

Figure 2.6: STRAT 16

Clos suggest that network communication latency is increased, potentially leading to slower performance. Additionally, the 3-Tier Clos has the highest global betweenness centrality (0.061), suggesting that certain nodes may become critical bottlenecks in the network. On the other hand, STRAT 14, while less complex with fewer nodes (14) and edges (28), achieves better efficiency. Its density (0.308) is higher than that of the 3-Tier Clos, indicating a more tightly connected network, which contributes to its shorter average shortest path length (1.692) and smaller diameter (2). The lower average eccentricity (2.0) in STRAT 14 suggests faster communication across the network, and its lower global betweenness centrality (0.058) indicates a more balanced distribution of traffic, reducing the likelihood of bottlenecks.

When comparing STRAT 11 and STRAT 16 together with the 2-Tier Clos, these

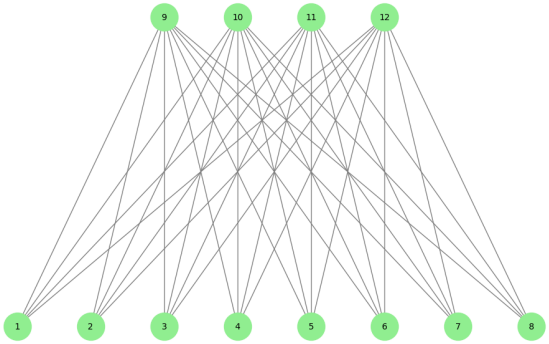

Figure 2.7: 2-Tier Clos

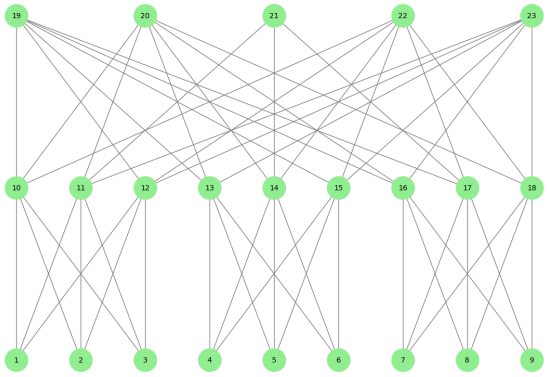

Figure 2.8: 3-Tier Clos

topologies also demonstrate superior efficiency and connectivity. Both STRAT 11 and STRAT 16 maintain a high average degree (6.0) and low diameter (2), similar to STRAT 14, but with even better average shortest path lengths (1.4 and 1.6, respectively). Their lower density values (0.4 and 0.6) compared to 2-Tier Clos suggest they are optimized for scalability without compromising performance. The 2-Tier Clos, although having a slightly higher average neighbor degree (6.667), also shows efficient connectivity with a small diameter (2) and low average shortest path length (1.515).

Table 2.1: Topology Metric Comparison and Summary

Metric	2-Tier Clos	STRAT 16	STRAT 11	3-Tier Clos	STRAT 14
Switches	12	16	11	23	14
Switch Ports	8	8	9	6	6
Number of Hosts	32	32	33	27	28
Edges	32	48	33	54	28
Density	0.4848	0.4	0.6	0.213	0.308
Average Degree	5.333	6.0	6.0	4.696	4.0
Diameter	2	2	2	4	2
Avg. Shortest Path	1.515	1.6	1.4	2.285	1.692
Global Betweenness	0.052	0.043	0.044	0.061	0.058
Average Eccentricity	2.0	2.0	2.0	3.391	2.0
Core Number	4	6	6	3	4
Avg. Neighbor Degree	6.667	6.0	6.0	5.348	4.0

2.6 DEALER Algorithm

2.6.1 Background & Motivation

Expander topologies have shown great potential, often surpassing traditional topologies even when using current routing algorithms [14,45,49]. However, these algorithms do not fully exploit the unique capabilities of Expanders [25,62], particularly in terms of leveraging their multiplicity of paths. The Equal-Cost Multi-Path (ECMP) routing algorithm improves load balancing and network redundancy but has drawbacks, including uneven traffic distribution from random flow hashing, especially in environments with large flows [26,54,58,62]. It also neglects downstream congestion, leading to poor performance in asymmetric networks or during frequent link failures [3]. In contrast, MultiPath TCP (MPTCP) enhances throughput by splitting a single TCP connection into multiple subflows [6]. However, it can cause MPTCP incast issues, where multiple servers overwhelm a receiver's buffer with bursty traffic, leading to packet drops and reduced goodput [56]. FastPass [34] is a centralized load balancer that improves network utilization by scheduling packet transmissions to reduce queuing. However, its centralized arbiter creates scalability challenges in large data centers, struggling with high traffic volumes and becoming a single point of failure. The additional round-trip latency for scheduling can hinder performance in low-latency environments. Moreover, FastPass requires specialized hardware and is less adaptable to changing network conditions, resulting in suboptimal performance in dynamic settings. Other algorithms, such as Flare [24], LocalFlow [40], and DRILL [16], use

state-unaware load balancing, where each hop performs flowlet switching to route small packet bursts across multiple paths. These methods are praised for their simplicity, scalability, and compatibility with existing hardware. However, they fail to account for downstream congestion and path utilization, which can lead to inefficient load balancing and decreased performance [3, 25, 26]. Additionally, their reliance on inter-packet gaps for defining flowlets makes them vulnerable to changes in network conditions, traffic loads, and transport protocols. CONGA [3] is a congestion-aware load balancing system that uses custom switch ASICs to monitor congestion along paths and shares this information with other switches via specialized packets. Each switch maintains a congestion feedback table to assist in destination ToR routing decisions. However, CONGA has limitations [8, 25, 26], including high memory requirements for path information, potential latency inaccuracies due to reliance on remote feedback, and limited adaptability from its dependence on customized ASICs. To address these issues, HULA [26] employs programmable data planes to track congestion on the best path to a destination using periodic probes for network utilization data. Despite being designed for Fat-tree topologies, which support a single ECMP group, HULA does not fully leverage the multiple paths available in Expander-based networks [8].

2.6.2 Proposed Routing Algorithm

In this section, we mention a new routing algorithm that was previously introduced which takes full advantage of the path diversity in Expander topologies [14, 45, 48, 49]. We first test its feasibility using a virtual environment and then confirm its viability through tests on a physical setup, demonstrating its suitability for modern data centers. Our analysis is based on the premise that all network links are capable of bi-directional communication and that a single routing table (RT) entry suffices for each distinct edge in the graph representation of the network. We initiate our discussion with the PINGing algorithm, which facilitates the construction of routing tables. We will then elaborate on the forwarding algorithm that operates within the data plane. For clarity, we emphasize that each unique graph edge corresponds to a single routing table entry and that bi-directional communication is supported across all network links. We will then be creating a new variation of this routing algorithm which will be implemented on TD4.

Constructing Shortest Path Routing Tables (SP RT)

During the initialization phase, we utilize the PINGing algorithm to construct the routing table. Each network node starts by setting all possible routes to potential destinations through its interfaces to an initial value of infinity, except for the route to itself, which is set to zero. The node designated as the destination then announces its presence to nearby neighbors by sending a PING packet. This packet contains the node's identifier and a metric of zero, which can represent parameters such as hop count or delay.

The processing of PING packets follows a "no reply" policy, meaning that nodes do not resend PING packets through the same interface from which they were received. Each node that receives a PING packet executes the following steps:

1. **Incoming Interface Registration:** The node records the interface through which the PING was received.

2. **Metric Update:** The node increases the metric reflecting the distance to the sender based on the received metric.

3. **Metric Comparison and Update:** Depending on the chosen options (Option 1 and Option 2 discussed below), the node assesses the relationship between the newly incremented metric and the currently stored metric to determine if an update is necessary.

4. **PING Resend or Termination:** If the stored metric is updated, the node retransmits the PING packet with the new metric; if not, it concludes the process.

The initialization phase effectively sets up a complete view of the network by using the PINGing process, allowing nodes to gather metrics about their neighbors quickly. This minimizes the time required to establish routes.

The algorithm introduces several configurations to strike a balance between rapid convergence and the integration of off-shortest path (OSP) entries within the routing table. These configurations are outlined below:

1. Option 1: In this approach, the algorithm updates the metric for the destination and broadcasts a revised PING to neighboring nodes if the new metric is lower

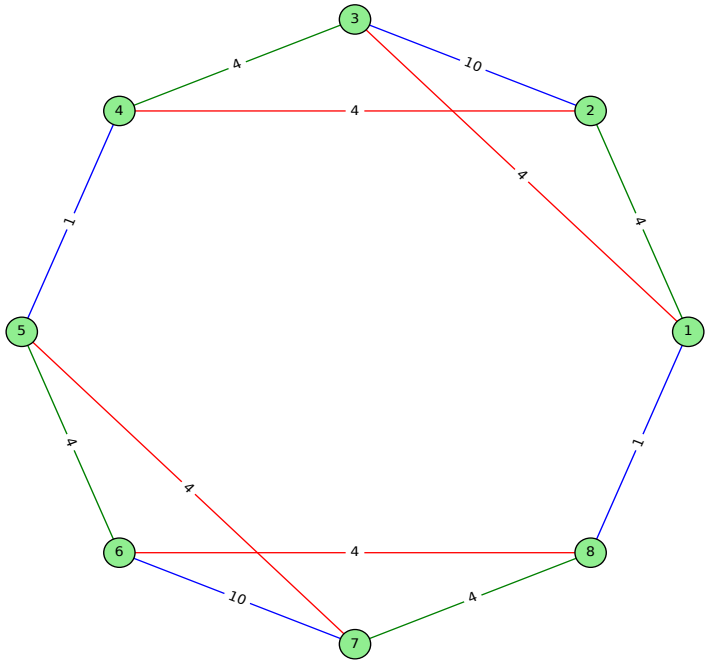

Figure 2.9: 8-node expander network

than those associated with all other interfaces. Additionally, if the new metric is equal to the existing one but is received through a different interface, the PING is also resent. If neither of these conditions is met, the PING process stops at that node without any updates. This method guarantees the creation of routing table entries for shortest paths (SP), while OSP entries can later be established by referencing the distance vectors (DVs) of neighboring nodes. Any healthy interfaces can serve as a fallback for packet forwarding instead of dropping them.

2. Option 2: This configuration is similar to the first, where the algorithm records the updated metric for the destination and re-broadcasts the PING to nearby nodes. However, it resends the PING only if the new metric is less than that of the interface receiving it (unlike Configuration 1, which checks all interfaces). If this condition is not satisfied, the PING terminates at the current node, and no updates occur. This results in routing tables containing both guaranteed SP and OSP entries, eliminating the need to reference the DVs of neighboring nodes.

Healthy interfaces can still be utilized as a fallback for forwarding packets.

3. An enhancement to Option 2 which allows for the establishment of reliable metric entries for all remaining last resort (LR) interfaces by ensuring loop-free PING propagation. This enables the selective forwarding of packets through LR interfaces, utilizing their reliable metric values.

To demonstrate our PINGing algorithm we will apply it on the 8 nodes network as depicted in Fig. 2.9 with varying bandwidths across specific links. The connections between nodes 2 and 3, as well as between nodes 6 and 7, each have a bandwidth of 10G. Meanwhile, the links between nodes 1 and 8, and between nodes 4 and 5, support a higher bandwidth of 100G. The remaining connections between nodes have a bandwidth of 25G. To quantify bandwidth, we use a metric based on the inverse of the actual bandwidth, normalized by the highest available bandwidth (100G). In this network, each node has three interfaces, corresponding to three links connecting them to their adjacent neighboring nodes. The links are color-coded as follows: interface 1 is blue, interface 2 is green, and interface 3 is red. This topology is now represented as an undirected weighted graph, where the integer weights correspond to a composite metric derived from the bandwidth.

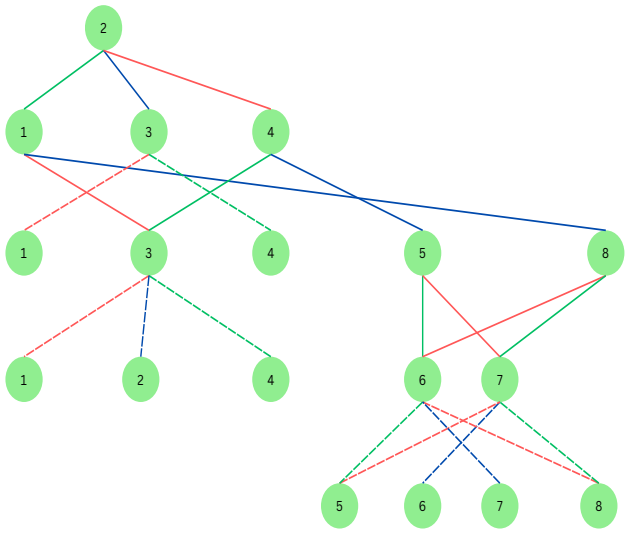

Figure 2.10: PING 2 propagation

Fig. 2.10 illustrates the propagation of the PING originating from node 2, referred

to as PING2. In the first PING iteration, the PING2 packet is sent to the nearest neighboring nodes adjacent to node 2. Node 3 updates its metric to 10 via interface 1, which is later improved to 8 through interfaces 2 and 3 in the second PING iteration. The algorithm prioritizes two-hop routes with better metrics over single-hop routes with worse metrics. After the second iteration, PING2 is terminated at nodes 1 and 4 because it carries a higher metric of 14, worse than what was received in the previous iteration. Similarly, in the third iteration, PING2 reaches nodes 1, 2, and 4, where it is terminated due to worse metrics. By the fourth iteration, all PING transmissions are terminated, as all SP routes are fully established within the network's three-hop diameter. Consequently, the fourth iteration is marked as terminal, indicated by dashed lines.

D	S2		
1	∞	4	∞
2	0	0	0
3	∞	8	8
4	∞	∞	4
5	∞	∞	5
6	∞	9	9
7	∞	9	9
8	∞	5	∞

Table 2.2: SP RT Node 2

D	S2		
1	14	4	12
2	0	0	0
3	10	8	8
4	14	12	4
5	15	13	5
6	19	9	9
7	19	9	9
8	15	5	13

Table 2.3: OSP RT Node 2

D	S1		
1	0	0	0
2	∞	4	12
3	∞	12	4
4	10	8	8
5	9	9	9
6	5	13	13
7	5	13	13
8	1	∞	∞

Table 2.4: OSP RT Opt. 1

D	S1		
1	0	0	0
2	14	4	12
3	14	12	4
4	10	8	8
5	9	9	9
6	5	13	13
7	5	13	13
8	1	17	17

Table 2.5: OSP RT Opt. 2 (loop-less)

Figure 2.11: Routing tables for various optimization scenarios

Fig. 2.2 is the SP RT for node 2 after PINGing, which depicts the PING messages received by node 2 initially from all the other nodes as they also PING to broadcast their presence carrying and updating the specified metric as needed.

Constructing OSP RTs With Already Established SP RTs

We can acquire OSP RTs while SP RTs are already established. The entries within the tables obtained are categorized into two distinct types.

1. The first type, OSP1, pertains to the interface connecting to neighbor S2 of the source node S. It is applicable when S2's SP metric to destination M(S2→D) does not exceed the metric from node S to destination M(S→D) plus the metric M(S2→S) to reach S2 from S. OSP1 entries in the OSP RT ensure that packets forwarded through this OSP1 interface can reach the destination using only the SP RT. The discovered OSP1 entry is the sum of M(S→S2) and M(S2→D).

2. The second type, OSP2, relates to the interface connected to neighbor S4 from the source node S. It is relevant when S4's shortest path (SP) metric to destination M(S4→D) is the same as the metric from node S to destination M(S→D) plus the metric M(S4→S) to get from S4 to S. However, this only applies if there is another alternative SP path from neighbor S4 to destination D with the same metric M(S4→D). The OSP2 entry is essentially the sum of M(S→S4) and M(S4→D).

Fig. 2.3 shows the presence of SP RT entries (highlighted in green) alongside OSP1 and OSP2 entries (in blue and red, respectively). Unlike the OSP RT of Option 1 Fig. 2.4, the OSP RT of Option 2 (Fig. 2.5) was derived without the need to consult the DVs of neighboring nodes. The primary trade-off with this approach is the possibility of introducing a few additional hops, although the exact number depends on the particular configuration of the network topology. Despite this, the method simplifies routing calculations by eliminating the need for inter-node DV consultations, offering a more streamlined approach with minimal impact on overall network performance. Option 2's OSP RT differs from Option 1's in that it replaces infinite metric values with finite ones. For both options, all entries corresponding to SP, OSP1, and OSP2 in the OSP RTs are the result of loopless PING propagation, ensuring perfect accuracy. Under a "no reply" policy, the smallest possible loop would introduce 3 additional hops to any loopless PING path, which would result in a metric greater than OSP2's. However, LR entries in the OSP RTs can be affected by PINGs that loop en route. Despite the different ways in which LR interface metrics are represented in Option 1 and Option 2, their behavior in RTs remains identical.

2.6.3 PathPort & Updating Routing Tables:

To eliminate loops from the network, we ensure that packets do not visit any node more than once on their way to the destination. This is achieved by providing the packet with a PathPort, which records the nodes already visited and prohibits packet forwarding through any interface that connects to a previously visited node. Although this approach increases complexity, it significantly improves overall performance in individual packet forwarding.

For initializing the network and making updates (whether scheduled or triggered by link status changes), a new copy of the RT is created with initial values set to infinity. PathPort information for every PING, both in the initialization and update processes, is timestamped to distinguish between scheduled and triggered updates. This timestamp prioritizes the more recent PING events and discards older RTs. The timestamping procedure prevents issues similar to "counting to infinity" that can be encountered in Distance Vector (DV) routing protocols by ensuring that two concurrent PING events started by different events do not overlap in time and interfere with RT construction. Each RT is associated with the timestamp when the PINGing process began. This RT is continuously constructed by PINGs with matching time stamps, and a hold-off timer is reset with every new PING arriving at a node. When PINGing stops naturally, the hold-off timer in a node will expire, and the newly constructed RT is copied into the active FT state. In the event of link failures, the efficient and robust forwarding protocol ensures that packets originally meant to be routed through the failed interface have ample opportunities for re-routing through alternative interfaces. This may result in slightly more hops to reach the destination but maintains network reliability. Both scheduled and triggered updates can re-build the RTs from scratch in a few hops, generally a few more than the network's diameter. The robust forwarding protocol lessens the urgency in accommodating failures, making the network resilient and efficient.

2.6.4 Forwarding Protocol: Dynamic Expander Algorithm for Load-Effected Routing

Once the RTs have been constructed by performing PINGing, the Dynamic Expander Algorithm for Load-Effected Routing (DEALER) depicted in Fig. 2.12 is used to

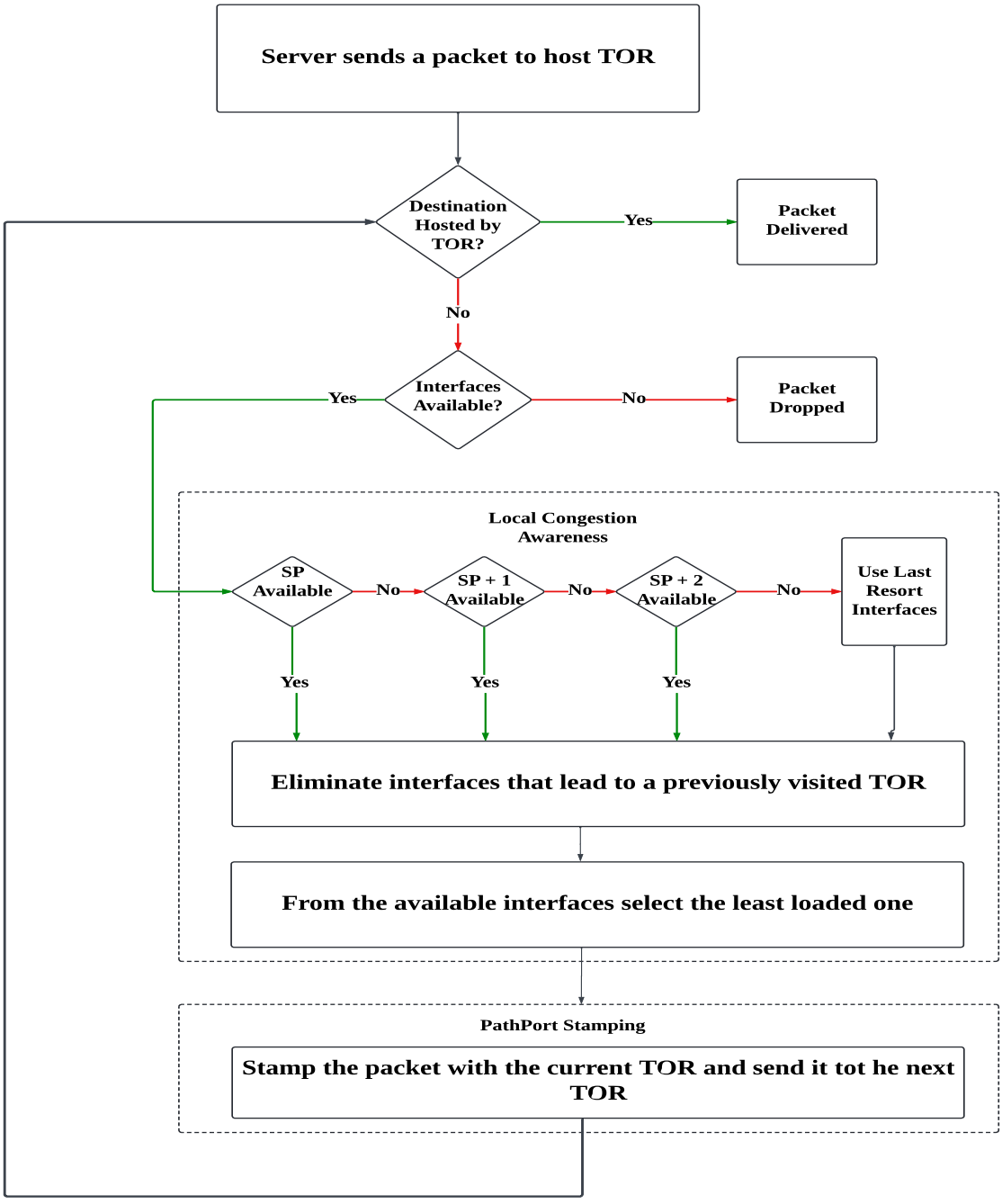

Figure 2.12: DEALER Algorithm

efficiently route packets within the data plane. When a packet is received at a TOR, it checks if the destination host is at that TOR; if it is, the packet is immediately delivered. If the current TOR does not host the destination host it assesses the available interfaces that have not been visited by the packet meaning the interfaces that lead to a TOR that has been visited before. From these available interfaces we pick the optimal one if it exists. By optimal we mean the least loaded interface amongst the available ones. Then the packet is forwarded to the next TOR using this optimal interface and we stamp the PathPort of the packet with the current TOR to mark it as visited. From this algorithm, you get the sense of how we leverage the diversity of paths by using local congestion awareness.

2.7 Routing Validation in P4 & Simulation Results

To initially validate the routing algorithm and assess its feasibility for deployment on a PDP, we utilized the P4 programming language alongside Behavioral Model version 2 (BMv2), the advanced successor to the original P4 software switch. BMv2 served as the backbone of our experiments, allowing for smooth implementation and testing of the routing protocol. However, it is essential to highlight that virtual testbeds like BMv2 have limitations when compared to physical testbeds. Differences in hardware capabilities, performance characteristics, and control over network elements can lead to significant variations between the two environments. We will illustrate some of these differences in the following section and suggest potential implementations.

Our initial experiments [2] were carried out on a STRAT topology consisting of 16 switches and 16 hosts, with each switch connected to a single host. The topology exhibited a diameter of 2, indicating that the longest shortest path between any pair of nodes required at most two hops. The average shortest-path length was 1.6, demonstrating efficient connectivity. These results confirm the implementability of the design on virtual programmable data planes (PDPs).

2.8 Routing Implementation in TD4®

Implementing this proposed routing on TD4® comes with some internal programmability limitations and design considerations that we need to accommodate for and we

have to change the algorithm as discussed below.

As can be seen in Fig. 2.12, our forwarding algorithm relies on local congestion awareness to properly forward packets on the least-loaded path. TD4® allows configuring ECMP groups in various modes. We will be focusing on the one that is of interest to the proposed forwarding algorithm.

- **Dynamic Group Multipath:** DGM is a TD4® programable feature that divides the ECMP group into two sets of egress ports: Primary and Alternative. The Primary set contains the preferred ports used for normal traffic flow, while the Alternative set consists of backup ports that are used only if Primary ports are congested. One can define what this congestion is by assigning weights to how much the load and queue occupancy should contribute to the overall congestion metric. Once this metric exceeds a certain threshold DGM will recognize the Primary ports as congested. At this point, it will begin routing traffic through the Alternative ports to alleviate congestion and maintain efficient network performance.

Since the forwarding algorithm partitions all the paths into Shortest Path (SP), SP + 1, SP + 2 till last resort paths. Implementing it on TD4® poses challenges due to its support only for Primary and Alternative ports within DGM groups, as previously noted. To address these constraints, we propose an adaptation of the forwarding algorithm that accommodates these limitations and challenges.

D	1	2	3	4
1	5	3	2	3
2	3	2	3	1
3	4	3	2	4
4	2	3	4	2
5	5	2	1	3
6	4	1	2	2
7	5	3	3	3
8	3	5	4	4
9	4	2	3	2

Table 2.6: Routing Table

D	1	2	3	4
1	X	A	P	A
2	X	A	X	P
3	X	A	P	X
4	P	A	X	P
5	X	A	P	X
6	X	P	A	A
7	X	P	P	P
8	P	X	A	A
9	X	P	A	P

Table 2.7: RT ECMP Mapping

Consider the RT displayed in Fig. 2.6 that is generated after performing PINGing on some topology where we color code the SP in green, SP+1 in blue and SP+2 in red. We proceed by translating this RT into ECMP associations as shown in Fig. 2.7.

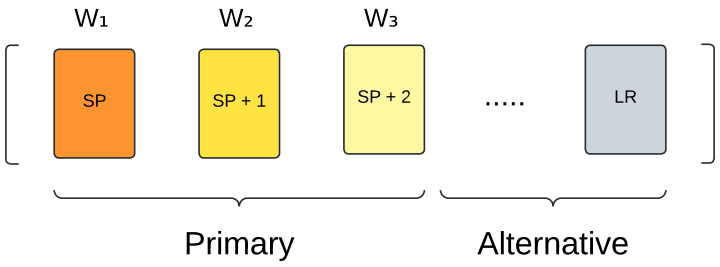

Figure 2.13: Primary & Alternative Members Mapping in TD4®

In this table, SP paths are designated as Primary (P), SP+N paths as Alternative (A), and any remaining entries are unused. One way to also translate this configuration into TD4® is by grouping all SP, SP+1 and SP + 2 in one Primary group with fractional weights to favor SP, SP+1 then SP+2 and the remaining paths as Alternative. Fig. 2.13 illustrates this mapping where W_1, W_2, W_3 represent fractional weights that would be given to each member in the primary group. Here, ensuring $W_1 > W_2 > W_3$ guarantees that SP is prioritized for selection. Since we are using DGM, we are ensuring that the Alternative ports will only be chosen once the Primary ECMP group exceeds a local congestion metric meaning the routing has a local congestion awareness. This metric can be set by giving weights and a threshold to the metric we wish to rely on. In our case, our metric is the queue-occupancy of an interface. Fig. 2.14 illustrates this approach using the 8-node STRAT topology in Fig.2.9. Here, We compare ECMP with a random seed, various permutations of DGM, and DGM + WCMP in terms of packet drop rates, confirming the findings from our simulations in the previous section. The results show that DGM + WCMP effectively capitalizes on the diversity of paths in Expander-based topologies, especially as network load increases, allowing throughput to degrade more gracefully. By including more options in both primary and alternate port groups, the system can better manage rising load, offering more routes for traffic to balance and maintain performance, especially when one favors the primary paths with the given weights.

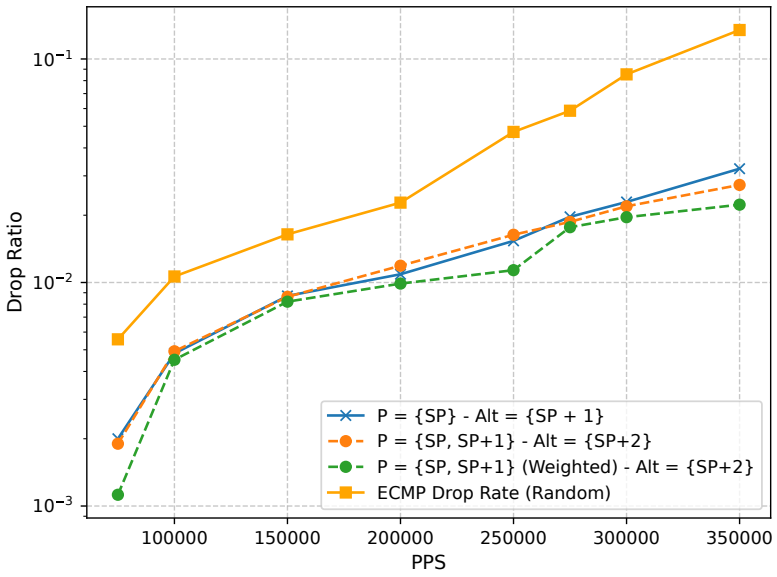

Figure 2.14: Different ECMP grouping over a small 8-node STRAT

2.9 Experimental Results

The statistics block in our testbed (Fig. 2.3) programs a dedicated counter for every $\langle$switch V, port $E_i\rangle$ pair. The Broadcom TD4® field processor tags each packet with the matching pair and increments the corresponding counter in hardware, giving loss-free per-port rates, queue depths, and drop events. Because every interface of every switch is monitored, we can reconstruct end-to-end flow trajectories and diagnose congestion anywhere in the fabric.

2.9.1 Aggregate Delivery Under Uniform Traffic

Figures 2.15–2.16 and Table 2.8 summarise packet-delivery rate (PDR) as load rises from 0.1 to 1.0.

- **Higher steady-state reliability.** STRAT 14 sustains a mean PDR of 98.2 % and never drops below 91.8 %; the 3-tier Clos falls to 72.6 % at peak load. The Load-Area-Under-Curve (LAUC) further confirms this gap: 0.886 (STRAT 14) vs. 0.826 (Clos-3), i.e. a flatter degradation profile.

- **Graceful collapse.** The linear-fit gradient over the heavy-load region ($l \in [0.7, 1.0]$) is -23.5 for STRAT 14 but -53.1 for Clos-3, indicating a much steeper

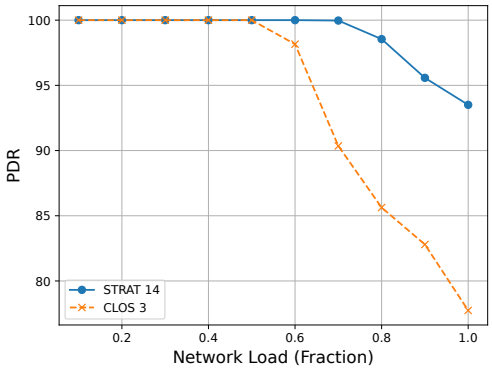

Figure 2.15: Packet Delivery Ratio STRAT 14 vs Clos3

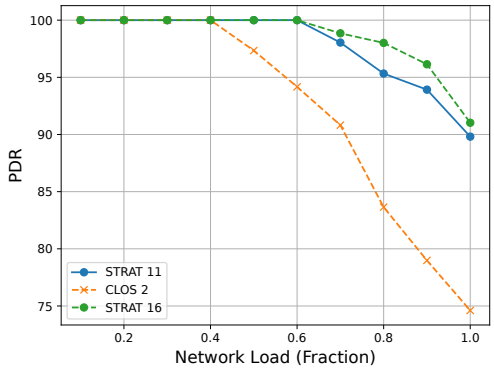

Figure 2.16: Packet Delivery Ratio STRAT 16 & 11 vs Clos2

throughput cliff for the hierarchical design.

- **Consistency across scales.** STRAT 11 and STRAT 16 likewise outperform the 2-tier Clos, delivering ≈ 10 pp higher mean PDR while cutting the run-out-of-fuel (ROF) loss fraction by more than half (Table 2.8, col. 5).

2.9.2 Congestion hotspots

Heat maps (Fig. 2.18, 2.17) reveal fundamentally different pressure points:

- **Clos-3:** drops concentrate on the spine layer, where every packet merges onto one deterministic downward path; once a spine queue backs up, all traversing flows suffer.
- **STRAT:** losses distribute over many ToR–ToR links; DEALER can steer around any blocked port, so no single interface dominates the drop budget.

Hierarchical fabrics therefore rely on transport-level mechanisms (e.g. DCQCN, PFC) to throttle sources—mechanisms that are notoriously hard to tune and merely shift delay to the edge. STRAT removes that complexity by absorbing bursts inside the network.

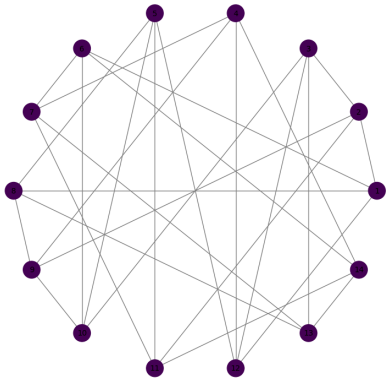

Figure 2.17: Hotspots map for STRAT 14

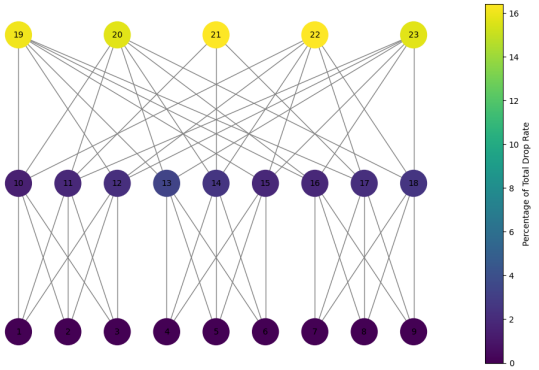

Figure 2.18: Hotspots map for Clos 3-Tier

2.9.3 Cost–Performance Trade-off

Fig. 2.19 shows that **STRAT eliminates** $\sim 40\,\%$ **of the switches** required by an equivalent Clos. Despite the smaller footprint, STRAT still delivers $\sim 44\,\%$ **more throughput** at $95\,\%$ load, proving that a flat optical mesh can improve performance and reduce cost.

2.9.4 Key Take-aways

Across every reliability metric (mean PDR, LAUC, ROF, collapse slope) and at equal radix, STRAT is both cheaper (fewer devices, less fibre) and more scalable (higher sustainable load, milder degradation) than its Clos counterparts. Because these gains are fabric-intrinsic and do not depend on end-host rate control, STRAT is well-suited for the burst-prone, ever-growing traffic demands of modern large-scale data-centre workloads. Generally, Clos networks are susceptible to congestion, especially prominent at higher layers of the network topology where there is a unique path down to the final destination. The industry introduced a number of protocols to deal with congestion, such as DCQCN , PFC [64], and many others. The aspects they all share is obtaining some measure of network congestion and an attempt to mitigate it by slowing down source sending rate. Such protocols have many tuning parameters which are notoriously difficult to optimize, and simply shift the problem to delaying (buffering) data at the source. More recently, Ultra Ethernet Consortium [47]

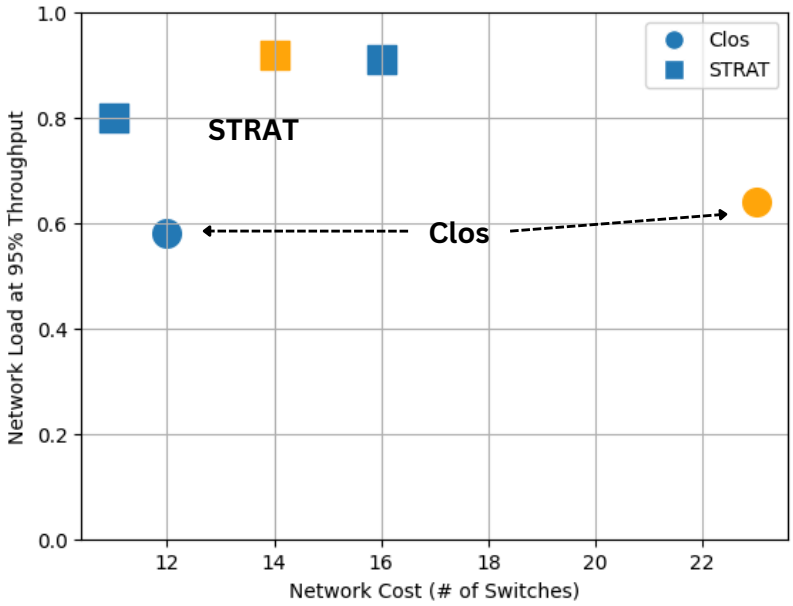

Figure 2.19: Network Load at 95% Throughput vs Network Cost

has started an effort to augment Ethernet with Infiniband-like features, particularly targeting demanding AI cluster networks. At a high level, the proposed approach sprays data packets across all available ECMP paths, and implements transmitter side buffering controlled by credits issued by the receiver. Potential packet misorder is also fixed at the receiver.

Table 2.8: Reliability metrics

Topology	**Avg PDR %**	**Min PDR %**	**AUC**	**LAUC**	**ROF %**	**Load ≥95%**	**Slope**†
STRAT14	98.15	91.82	0.886	0.0144	1.47	0.8	-23.48
STRAT16	97.83	88.74	0.884	0.0161	1.64	0.8	-33.01
STRAT11	96.68	87.17	0.873	0.0268	2.75	0.7	-32.54
Clos3 (3-tier)	91.27	72.62	0.826	0.0736	7.95	0.6	-53.10
Clos2 (2-tier)	88.95	69.55	0.805	0.0953	10.5	0.5	-42.60

†Linear-fit gradient over $l \in [0.7, 1.0]$; more negative means a steeper collapse in heavy load.

2.9.5 Distributed DNN training Traffic Pattern

In [53], the authors analyze traffic patterns in distributed deep neural network (DNN) training workloads and show that the underlying network communication follows a

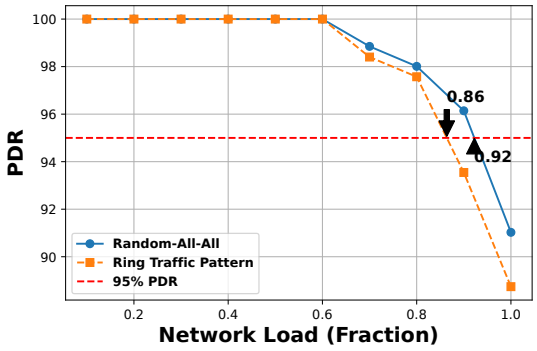

Figure 2.20: STRAT 16

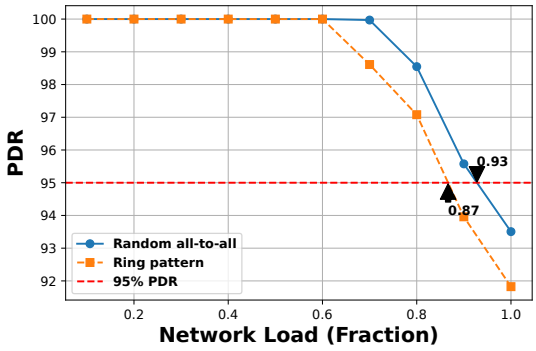

Figure 2.21: STRAT 14

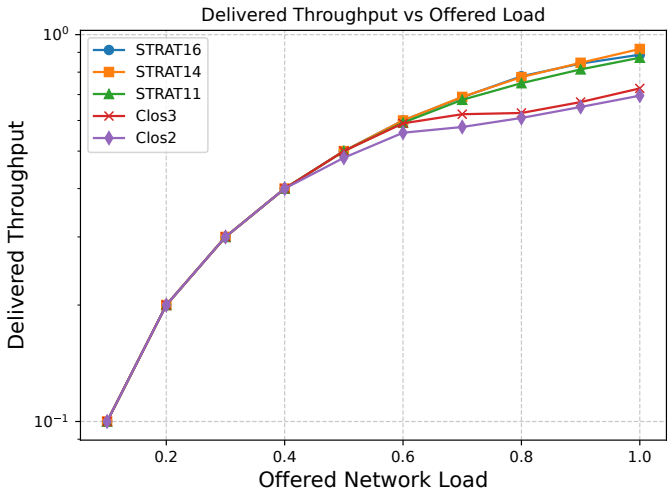

Figure 2.22: Throughput comparison

distinct ring pattern. This behavior stems from the widespread use of ring-AllReduce collectives in large-scale DNN training systems, which enable efficient gradient aggregation across multiple GPUs. Their traffic heatmaps confirm the emergence of a predictable, periodic structure due to this ring-like data transfer pattern, a direct consequence of parallelization strategies employed in production-grade training setups.

Motivated by these findings, our goal is to evaluate whether STRAT can serve as a viable and efficient alternative to traditional topologies—such as Clos—in supporting the intensive and structured traffic demands of distributed DNN training. To this end, we subject both STRAT and Clos to a ring traffic pattern emulating the structure identified in [53], as illustrated in Fig. 2.25. Our results, shown in Figures 2.24,

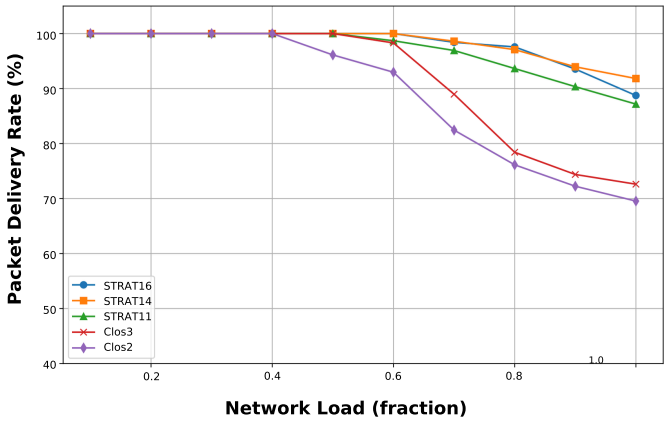

Figure 2.23: Ring traffic STRAT vs Clos

2.20, and 2.21, reveal that STRAT maintains over 95% throughput with only a slight performance degradation (around 6%) under ring traffic compared to an all-to-all scenario. Furthermore, Fig. 2.23 highlights STRAT's superior performance relative to Clos under the same conditions. While Clos exhibits noticeable degradation, STRAT sustains high throughput and consistent delivery, demonstrating resilience even under structured and directed traffic. This robustness under realistic DNN training workloads illustrates STRAT's design efficiency and underscores its suitability for AI training clusters where performance and stability are critical. These results affirm STRAT as a compelling topology choice for supporting the high-bandwidth, low-latency demands of modern machine learning infrastructure.

Analyzing the reliability and throughput metrics across the evaluated topologies reveals distinct trends in Table 2.8 and Fig. 2.22. STRAT14 exhibits the best performance, achieving the highest Area Under the Curve at 0.8856, the lowest LAUC at 0.0144, and the smallest retransmission overhead (ROF) of 0.01465. It also features a mild tail-slope of -23.48, indicating a gradual decline under heavy loads. STRAT16 performs similarly with slightly higher LAUC and ROF values, while STRAT11 shows increased degradation but remains better than the Clos networks. In contrast, Clos3 and Clos2 suffer from higher LAUC (0.0736 and 0.0953, respectively) and much steeper tail-slopes (-53.10 and -42.60), highlighting severe instability as load increases. Throughput analysis shows that STRAT14 also achieves the highest peak

delivered throughput at 0.918 of the line rate, with STRAT16 and STRAT11 following, while Clos3 and Clos2 peak much lower, at only 0.726 and 0.695 respectively.

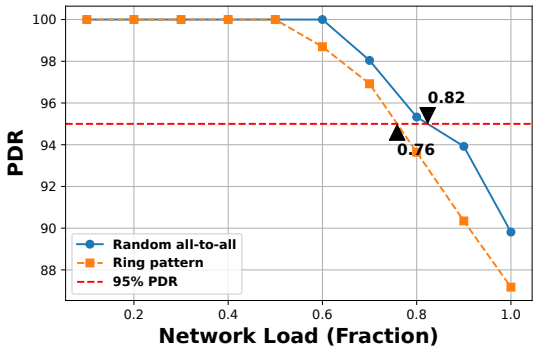

Figure 2.24: STRAT 11

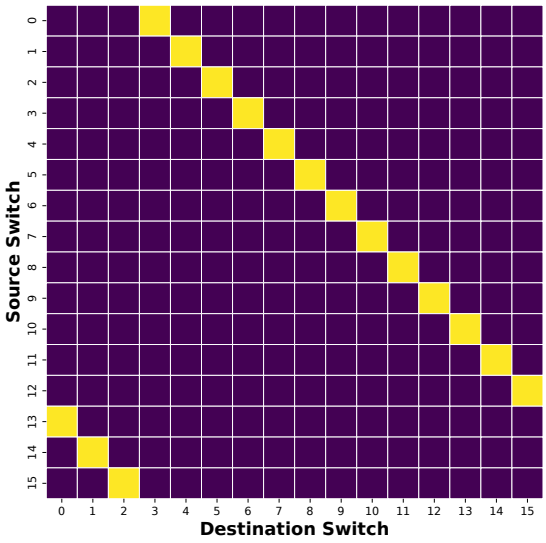

Figure 2.25: Ring Traffic Heatmap

2.10 Conclusion

This chapter presented an in-depth experimental and simulation-based evaluation of STRAT, an expander-inspired flat topology, as a scalable and cost-effective alternative to the conventional Clos architecture. Through the construction of a programmable Trident4 testbed and large-scale OMNeT++ simulations, STRAT was shown to outperform Clos in key performance metrics such as throughput, packet delivery ratio, and congestion resilience—while requiring fewer switches and offering simpler deployment. The analysis demonstrated that STRAT's expander properties provide higher path diversity and more uniform load distribution, particularly under high traffic conditions. Additionally, the proposed DEALER routing algorithm successfully exploited STRAT's structural advantages by using localized queue awareness and path history to reduce packet drops and balance load adaptively. These results position STRAT as a practical and deployable topology for next-generation data center networks, laying the groundwork for further enhancements in hybrid integration and dynamic traffic engineering explored in the following chapter.

Chapter 3

ML-Driven Optical–Electrical Expander Fabrics for Low-Latency Data-Center Networks

This chapter presents the detailed design of Hybrid–STRAT, a data center architecture that fuses the structural efficiency of a STRAT expander topology with a machine-learning-driven hybrid switching mechanism. By integrating optical circuit switches over a flat electrical base, Hybrid–STRAT dynamically adapts to traffic demands, offloading elephant flows onto high-bandwidth optical paths while retaining mice flows in the packet-switched electrical mesh.

Unlike traditional Clos-based hybrids, Hybrid–STRAT leverages STRAT's uniform connectivity and low-diameter structure to minimize optical setup costs and reduce average packet delay. A lightweight XGBoost classifier makes early flow predictions using packet-level features, enabling fast, accurate routing decisions that optimize network performance under bursty and heterogeneous loads.

3.1 Introduction

Modern data center networks are under immense pressure to support increasingly demanding and diverse workloads. Applications such as real-time cloud services, machine learning model training, high-performance data analytics, and distributed storage systems generate large volumes of traffic with distinct communication patterns.

These patterns typically fall into two categories: (1) short-lived, latency-sensitive flows often referred to as mice, and (2) long-lived, high-throughput elephant flows. Traditional electrical packet-switched (EPS) data center topologies are being pushed to their design limits in terms of throughput, energy consumption, and manageability, especially under such heterogeneous and dynamic traffic conditions. In this chapter, we present Hybrid–STRAT, a proactive, congestion-aware DCN architecture that overlays one of three optical circuit-switch (OCS) permutations onto a Structured Re-Arranged Topology (STRAT) expander fabric. An XGBoost classifier, trained on production flow traces, predicts flow type within the first few packets and dynamically steers elephants to the optical layer while leaving mice in the electrical packet-switch (EPS) mesh. The design is realised in OMNeT++ with modular `C++` components for topology construction, traffic generation, hybrid routing, and queue management. Under all-to-all traffic from 10–100% line rate, Hybrid–STRAT lowers mean packet latency from 1.045 μs to 1.160 μs (11% growth), versus 1.035 μs to 1.244 μs (20% growth) for pure-EPS STRAT, and cuts 99^{th}-percentile delay by 25–35% (42.3 μs vs. 57.1 μs at full load). Average hop count remains nearly constant ($\approx$1.76), showing that gains arise from queue suppression rather than path stretch. By fusing expander-graph path diversity, optical bandwidth efficiency, and machine-learning-driven traffic engineering, Hybrid–STRAT delivers a scalable, energy-aware, and traffic-adaptive solution for next-generation AI-centric DCNs.

3.2 Related Works

The increasing demand for low-latency and high-throughput communication in data centers has accelerated research into hybrid electrical/optical architectures. A key challenge lies in managing the complexity of flow aggregation and switching to accommodate diverse traffic patterns. Zhao and Shi [63] explore this issue by proposing machine learning-assisted aggregation schemes for optical cross-connects, demonstrating improvements in throughput and latency via localized, edge-node ML deployment. This motivates our design, which leverages XGBoost classifiers to steer flows dynamically in a STRAT-based hybrid topology. The necessity of differentiating traffic flows based on their characteristics was earlier addressed by Lee and Choi [28], who proposed a flow-level classification framework for hybrid switching networks. Their

classification of short-lived versus long-lived flows laid the conceptual groundwork for our approach to proactive flow steering using machine learning.They highlight the importance of application-aware configuration in all-optical data center interconnects, particularly for distributed machine learning workloads. Their findings reinforce our hybrid design philosophy, where optical overlays are dynamically configured to support high-volume, latency-tolerant elephant flows. Traffic offloading strategies also play a crucial role in hybrid architectures. The work of Ye et al. [59] introduces a threshold-based offloading mechanism for burst traffic, demonstrating that localized decision-making at the ToR level reduces operational overhead. which inspires our own distributed control mechanism, integrated with machine learning classifiers, to improve adaptability in the STRAT fabric. Ben-Itzhak et al. [7] present a flat hybrid packet/circuit architecture with orchestration and dynamic optical routing. To balance performance and cost, Feng et al. [13] propose the Blocking Loss Curve (BLOC) model, highlighting the importance of jointly considering traffic partitioning and resource allocation. This motivates our hybrid STRAT approach, which aims to maximize efficiency by selectively offloading traffic while preserving EPS resources for short, latency-sensitive flows. Finally, the need for fast reconfiguration in hybrid data center networks is addressed by Zhang et al. [61], who formulate a topology reconfiguration problem as a minimum cost flow model. Their results underscore the necessity of minimizing reconfiguration overhead.

3.2.1 Limitations of Electrical Packet-Switched Architectures

Current data center networks (DCNs) predominantly rely on hierarchical, electrical packet-switched (EPS) architectures such as Fat-Tree [29], Leaf-Spine, and BCube [19]. These designs are widely favored due to their scalability, ease of implementation, and compatibility with commodity switching hardware. Nevertheless, despite their prevalence, EPS-based architectures exhibit several significant limitations that constrain their effectiveness and scalability, especially given today's rapidly evolving data center demands.

Firstly, the reliance on high-radix switches introduces inherent scalability challenges. High-radix switches, essential for providing high port densities, face physical and thermal limitations as switch ASICs scale [22, 35]. These constraints inhibit network growth, limiting bandwidth expansion and the number of achievable ports

within a single device, consequently creating scalability bottlenecks.

Secondly, hierarchical EPS structures frequently lead to congestion at upper-tier switches due to oversubscription. This limitation becomes particularly pronounced during intensive inter-rack communications, a scenario frequently encountered in cloud computing environments and parallel processing tasks [13, 57]. The result is uneven bandwidth distribution and increased latency, adversely impacting overall network performance.

Thirdly, EPS-based systems exhibit substantial inefficiencies in energy consumption. Each packet undergoes multiple stages of processing through successive switches, resulting in significant cumulative energy usage. These repeated packet-handling steps contribute to increased operational costs and environmental impacts [11, 35]. Recent studies suggest hybrid architectures integrating optical components could substantially mitigate these inefficiencies.

Lastly, the static nature of traditional EPS routing strategies limits their adaptability to dynamic, bursty traffic patterns characteristic of modern workloads such as machine learning and cloud-native applications [50]. Static routing schemes are unable to respond quickly and effectively to fluctuations in network traffic, resulting in suboptimal utilization of available resources.

In response to these intrinsic challenges, current research is actively exploring alternative network architectures. These novel designs aim to separate forwarding and transport functions, reduce overall hop counts, and enhance energy efficiency, all while maintaining scalability and robust performance to better accommodate modern data center workloads.

3.2.2 Hybrid Optical-Electrical DCNs: Toward Scalable and Energy-Efficient Interconnects

To address the inefficiencies of purely electrical networks, researchers have investigated hybrid DCNs that combine optical switching with electrical packet forwarding. These designs seek to unite the programmability and low latency of EPS with the high throughput and energy efficiency of OCS. Typically, short-lived "mice" flows remain on the EPS layer for flexible handling, while high-volume "elephant" flows are routed through dedicated optical paths.

Prototypes like c-Through and Helios illustrate the viability of dynamic optical

circuit allocation based on traffic demands, with Helios achieving over 35% throughput gains and reduced cooling costs compared to EPS-only setups. More advanced architectures such as HiFOST utilize nanosecond-scale fast optical switches and flow-controlled buffering to minimize packet loss and enhance energy savings in large-scale environments [55]. These hybrid architectures represent a promising direction for future data centers by balancing scalability, energy efficiency, and low latency. However, to the best of our knowledge, no prior work has explored the integration of expander topologies within hybrid architectures to evaluate their effectiveness.

To the best of our knowledge, no prior work has explored a hybrid data center topology that integrates a structured expander graph—specifically, the STRAT architecture—with both optical circuit-switching (OCS) and electrical packet-switching (EPS). Our proposed design leverages STRAT's uniform node degree, low path diameter, and high path diversity to support an efficient dual-layer forwarding model. By combining the energy efficiency and bandwidth advantages of OCS with the flexibility and fine-grained control of EPS, Hybrid–STRAT offers a scalable, resilient, and low-latency solution for modern data center traffic dominated by a mix of short-lived and high-volume flows.

This chapter is organized as follows:

- **STRAT Architecture:** We begin by introducing the Structured Re-Arranged Topology (STRAT), a flat, deterministic expander topology that offers strong connectivity and low-diameter paths using uniform ToR-to-ToR links. Its expander properties provide multiple disjoint paths, improving resilience and enabling efficient traffic distribution without reliance on hierarchical switching.

- **Machine Learning Model:** We briefly present the machine learning framework used to classify flows. Specifically, we employ an XGBoost classifier trained on early-packet features to distinguish between mice and elephant flows. This model enables real-time flow steering by providing fast, accurate predictions that guide traffic onto the most suitable switching layer.

- **Hybrid STRAT Architecture:** We then detail the proposed hybrid architecture, where three optical overlays are embedded atop the STRAT electrical fabric. These overlays correspond to carefully selected permutations that facilitate high-throughput optical paths for elephant flows. The architecture also includes

dynamic scheduling and routing mechanisms that adapt to real-time traffic conditions.

- **Virtual Testbed Environment:** Next, we describe the OMNeT++-based simulation testbed used to evaluate our design. The testbed includes modular components for topology generation, traffic injection, hybrid routing, queue simulation, optical circuit management, and flow-level metrics collection. This environment enables comprehensive performance analysis under varying load conditions.
- **Results and Evaluation:** Finally, we present quantitative results comparing Hybrid–STRAT to a baseline EPS-only STRAT configuration. Metrics include average and tail packet delay, hop count, and queue behavior under all-to-all traffic patterns. The results demonstrate significant gains in latency reduction and queue suppression, validating the effectiveness of our hybrid design.

3.3 STRAT: A Structured Expander Topology for Flat, Resilient DCNs

Expander-based topologies have emerged as a promising alternative to traditional multi-tier DCNs. Unlike hierarchical designs, expanders provide a flat architecture composed of uniformly connected ToR switches, forming graphs with high connectivity and low diameter. Among these, the **Structured Re-Arranged Topology (STRAT)** has been shown to outperform other expander designs like Jellyfish and Xpander in both robustness and performance metrics [1] [14]. STRAT combines the flexibility of expanders with a more deterministic layout, making it easier to implement in practice while preserving desirable expander properties such as high algebraic connectivity and large spectral gap. These structural benefits make STRAT a strong candidate for hybrid augmentation, where optical links can complement electrical connectivity by providing high-bandwidth shortcuts for bulk traffic. While STRAT delivers exceptional performance as an all-electrical topology, it has not yet been explored as a hybrid platform. We argue that STRAT is particularly well-suited for hybridization due to its flat architecture, consistent node degrees, and uniform path diversity. Unlike Clos-based hybrids that require coordination across tiers and rely on

high-radix core switches, a hybrid STRAT design can exploit direct ToR-to-ToR optical paths to establish low-latency, high-throughput connections without significant architectural overhauls.

The reduced average path length in STRAT minimizes the setup and teardown delays associated with circuit-switching, making dynamic optical scheduling more feasible. Additionally, STRAT's uniform connectivity provides multiple candidate paths for rerouting elephant flows, increasing resilience and simplifying flow scheduling decisions. By introducing OCS overlays across the STRAT fabric, we can enable a dual-tier forwarding model: mice flows are handled within the EPS fabric, while elephants are rerouted through fast, energy-efficient optical circuits.

This hybrid STRAT architecture aims to leverage the best of both technologies: the agility and fine-grained control of EPS and the bulk transport and energy savings of optical switching.

3.4 Machine Learning for Flow Classification and OCS Scheduling

A critical component of any hybrid DCN is the accurate and timely classification of flows. Traditional approaches use static thresholds based on flow size or duration to identify elephant flows, but these techniques are often brittle and unable to adapt to dynamic traffic conditions. To address this, recent work has turned to machine learning models that can infer flow characteristics from early packet-level features.

Studies demonstrate that machine learning techniques such as decision trees, Naïve Bayes classifiers, and deep reinforcement learning can significantly enhance flow classification accuracy and enable smarter optical circuit scheduling [46, 51]. These models allow for real-time decision-making under dynamic traffic loads, enabling adaptive scheduling that reduces average delay and packet loss. For instance, Flow Splitter, a deep reinforcement learning-based scheduler, adapts to runtime network conditions and effectively separates elephant and mice flows, improving flow completion times [46]. Other works have shown the promise of auto-regressive neural networks in predicting server traffic patterns, enabling optical circuits to be pre-allocated for large flows with high accuracy [5]. These predictive capabilities help ensure that over 80% of data, often carried by fewer than 20% of flows, is routed through optimized

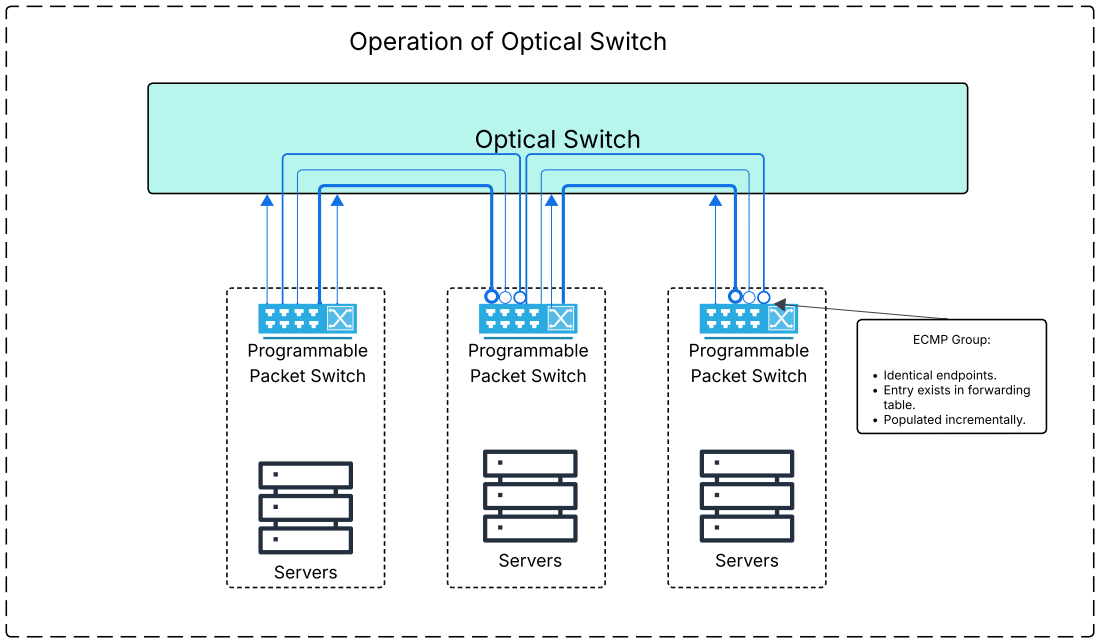

Figure 3.1: Operations of optical switch

optical paths.

Integrating such ML-based classifiers into hybrid architectures allows real-time, adaptive flow steering. Mice flows are immediately forwarded through the EPS fabric, while elephant flows are diverted via OCS based on predicted size, duration, and path requirements.

3.5 Hybrid-STRAT: Hybrid Expander Topology

3.5.1 Methodology

To attempt to mitigate the issues that pure OCS and EPS topologies suffer from in this chapter we will propose, build and evaluate a data center architecture that combines a STRAT-based expander topology, providing low-diameter, resilient, and uniform connectivity, with a sparse overlay of optical circuit switches to dynamically carry elephant flows across ToRs. Our architecture incorporates a machine learning-based flow classification system that predicts flow types using early packet features and an integrated routing and scheduling framework that adaptively assigns traffic to

EPS or OCS paths based on traffic characteristics and network state. In particular, we design and evaluate a hybrid STRAT topology that integrates both OCS and EPS switching technologies to manage data center traffic efficiently. To classify elephant and mice flows early in their lifecycle, we employ a machine learning classifier based on the XGBoost algorithm. This classifier is trained and tested using a modified version of the UNI1 dataset [9], referred to as UNIV1. The simulation is implemented in OMNeT++ using a modified version of the OBS Module to support hybrid switching behavior and traffic control. We provide a comprehensive performance comparison between a fully EPS-based STRAT and our proposed hybrid STRAT design. This architecture aims to address the dual challenge of scalability and performance in modern DCNs by blending the path diversity of STRAT, the energy efficiency of optics, and the intelligence of ML-based traffic engineering. Our approach provides a scalable, robust, and cost-effective solution for future-ready data center networking.

3.5.2 Hybrid-STRAT: Expander Upgrade

STRAT itself can be modelled as a network with optical switches that operate as demonstrated in Fig. 3.1 where traffic is first directed through a single port and subsequently through another distinct port. When the queue on the first port grows, a second channel is activated; once both the first and second queues become busy, a third channel is employed. As the packet switch detects that the associated ECMP group is nearing exhaustion—an indication that all links share identical endpoints and already have an entry in the forwarding table (which is populated incrementally)—it pinpoints the congested port (e.g., a 25 Gbps link) and propagates an updated ECMP-to-port association to the affected packet switches, leaving the forwarding table itself unchanged, Which resembles DEALER algorithm that was described in the previous chapter. An example of STRAT based on optical switches is displayed in Fig. 3.2. The design of our Hybrid–STRAT is informed and inspired by key observations in TOPOOPT [52], which demonstrates the benefits of co-designing topology and communication patterns for distributed machine learning workloads. In particular, TOPOOPT introduces a method to optimise AllReduce communication by constructing demand-aware ring overlays atop a direct-connect topology. The most relevant findings motivating our work are summarised below:

1. **AllReduce Topological Flexibility.** The AllReduce primitive is inherently

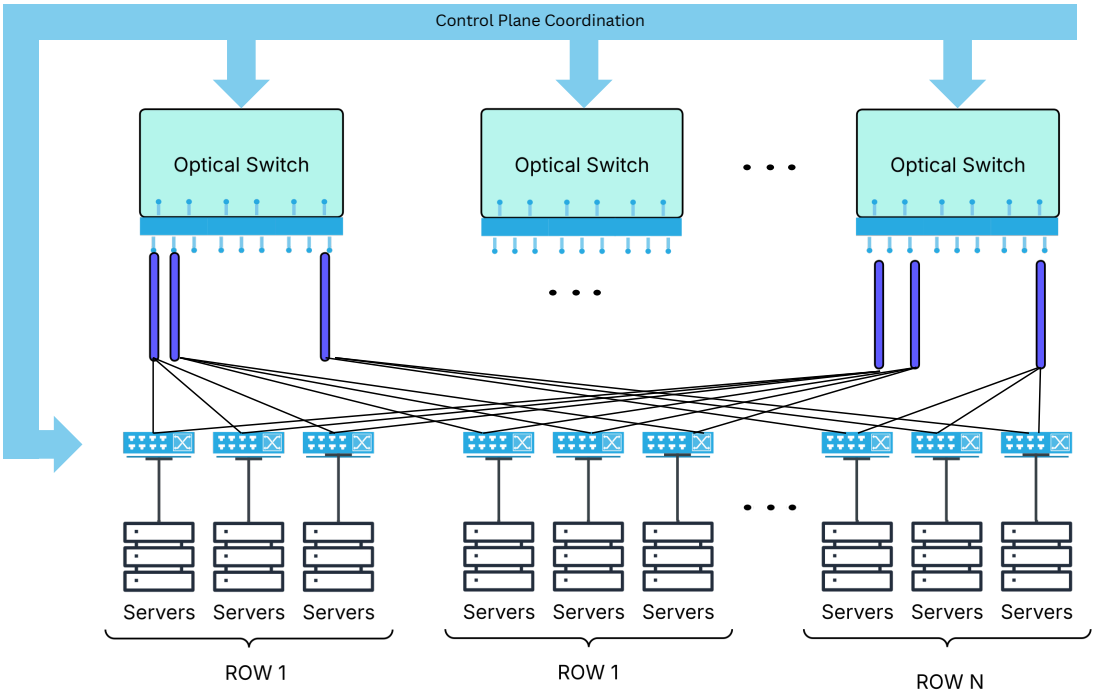

Figure 3.2: STRAT based on optical switches

topology-agnostic with respect to ring permutations, allowing communication to proceed over any ring structure without correctness loss. This flexibility opens the door for optimising performance by choosing permutations that balance path diversity and communication efficiency.

2. **Efficient Permutation Selection.** The authors propose a group-theoretic algorithm, *TotientPerms*, to generate a family of ring permutations with desirable topological properties—specifically, disjoint paths and minimal overlap. Through evaluation, they show that selecting just three carefully chosen permutations (e.g., with offsets +1, +3, +7) suffices to achieve near-optimal load balancing and throughput across a variety of ML workloads.

3. **Empirical Performance Gains.** Using this three-permutation design, the authors of [52] report significant speedups (up to 3.4×) in DNN training iterations over conventional Fat-Tree and Clos-based designs, without increasing network cost.

Application to Hybrid–STRAT. Building on these insights, we extend the STRAT architecture by introducing a hybrid electrical/optical variant, Hybrid–STRAT, which leverages a circuit-switched optical layer to accommodate high-volume, latency-tolerant elephant flows. Inspired by TOPOOPT, we instantiate three OCS overlays corresponding to the ring permutations $\langle +1 \rangle$, $\langle +3 \rangle$, and $\langle +7 \rangle$. These overlays form a low-diameter, high-throughput backbone that complements STRAT's expander-based EPS.

This design offers several advantages. First, the optical permutations ensure path diversity and reduce congestion by offloading long flows from the electrical fabric. Second, the limited number of permutations (three) adheres to practical constraints, such as the number of transceivers per ToR switch, mirroring the $d = 4$ node degree adopted in the original TOPOOPT evaluation. Finally, the hybrid structure preserves STRAT's cost-efficiency and scalability while improving its suitability for ML workloads with structured communication patterns. Overall, our hybrid design is a principled response to the challenges of flow classification and path allocation in modern data centers, particularly under DNN-driven traffic.

3.6 Experimental Testbed

3.6.1 Simulation Components

The custom *OMNeT++* testbed is built entirely from modular C++ blocks that correspond one-to-one with `NED` modules. Each block has a single, well-defined responsibility so that individual pieces can be swapped or extended without recompiling the entire simulator. Below we expand on the role of every major component in the setup:

- `Network`—**Topology Builder**
  At start-up this module parses a pair of simple text files: *nodes.conf* (servers, EPS switches, OCS switches) and *links.conf* (bandwidth, delay, buffer size). It then instantiates the corresponding `NED` objects, wires their gates together, and propagates global parameters such as queue length and routing mode to every sub-module.

- `Generator`—**Traffic Generator**

Deployed on each server, the generator draws new flows from a pre-computed traffic matrix. Separate statistical distributions are maintained for mice and elephant flows, allowing independent control of inter-arrival time and flow size. Packet bursts are emitted at line rate until either the flow finishes or a queue tail-drops the packet.

- `Classifier`—**Packet Router**
  Upon receiving the first packet of any new five-tuple, the classifier as shown in Fig. 3.3 performs three actions in quick succession: (i) extracts lightweight header features, (ii) invokes an offline-trained XGBoost model to label the flow as mice or elephant, and (iii) installs a per-flow forwarding rule. Mice are forwarded over electrical packet switching (EPS), whereas elephants are diverted to the optical circuit switch (OCS) overlay. Path selection itself may follow shortest-path, k-shortest path, or ECMP hashing.

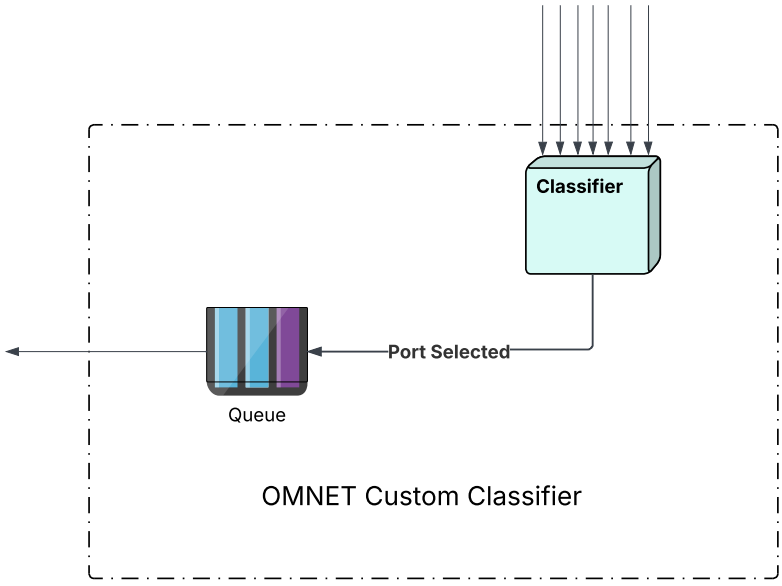

Figure 3.3: OMNeT++ Custom Classifier

- `QueueBuilder`—**Output Buffer**
  Attached to every switch port, this component allocates a finite egress queue, timestamps each enqueue, and updates queue-length statistics on every dequeue. Packets that arrive to a full buffer are dropped and marked with an over-flow flag

so the logger can attribute latency spikes or completion-time tail events.

- `SinkArch`—**Flow Logger**
  Every server hosts a `SinkArch` process that records a concise summary for each completed flow: end-to-end delay, hop count, reordering depth, and final delivery status. The logger writes separate CSV files for mice and elephant traffic so that per-class metrics can be analysed off-line (see Fig. 3.4).

  this sink is basically a way for us to simulate an actual host! in OMNET this translated as a NED module which allows the packets to sink and we also use it collect statistics . Classification is basically occuring when a packet is received -

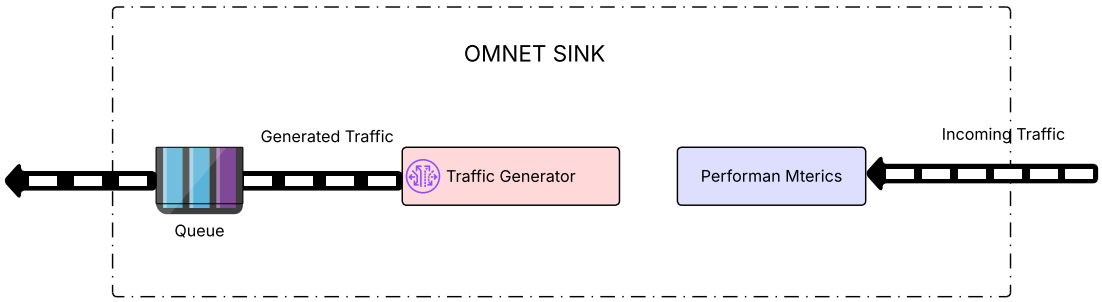

Figure 3.4: OMNeT++ Custom Host

- `PacketConstructor`—**Message Format**
  All data and control messages inherit from a single `Pkt` base class defined here. The header contains source/destination IDs, flow ID, hop count, TTL, and a small scratchpad for future metadata. Using one unified format across the simulator simplifies instrumentation and post-processing

- `OCSController.cc`—**Circuit Scheduler**
  This centralised controller maintains a global view of current traffic demand and periodically allocates time-slotted optical circuits. After computing a matching, it broadcasts lightweight control messages that reconfigure every `OpticalSwitch` in the fabric.

- `OpticalSwitch.cc`—**OCS Crossbar**
  Each optical switch is a bufferless, non-blocking crossbar whose state is a permutation matrix loaded by the `OCSController`. The hardware-accurate cut-through

model forwards incoming packets immediately if a circuit is active; packets arriving during reconfiguration are dropped, reflecting real-device behaviour.

- `OCSChannel.ned`—**Optical Link**
  Finally, high-capacity inter-rack links are instantiated with this channel type. In addition to standard *OMNeT++* parameters (data rate, propagation delay) the channel exposes a configurable `setupDelay` field to emulate the hardware reconfiguration time of the underlying optical technology.

3.6.2 ML Classifier Training Data

The eXtreme Gradient Boosting algorithm is a natural fit for our flow–classification task:

1. **Tabular, Heterogeneous Features.** Early–life flow descriptors (first–burst size, inter-arrival variance, packet-size moments, etc.) are low-dimensional, numeric, and non-linear. Gradient-boosted decision trees model such tabular data without heavy preprocessing, capturing both sharp thresholds (e.g., the 10 KB mice/elephant cut-off) and higher-order feature interactions.

2. **Class Imbalance and Robustness.** Elephant flows form $< 5\%$ of all connections yet dominate byte volume. XGBoost offers built-in instance weighting and `scale_pos_weight` tuning, reducing bias toward the majority (mice) class and delivering the high F_1 scores reported below. The ensemble is also tolerant of outliers and sporadic missing values common in packet traces.

3. **Efficiency and Interpretability.** Training on $\sim$1.2 M labelled flows finishes in seconds and supports incremental updates, allowing periodic retraining with new traces. Feature-importance diagnostics consistently highlight burst size, early RTT, and packet-size variance as dominant predictors—insights that can inform future traffic-engineering heuristics.

Deep neural networks demand larger feature sets and longer training, while linear models underfit the strongly non-linear boundary between mice and elephant flows. XGBoost therefore strikes the optimal balance of accuracy, speed, and operational transparency for our hybrid STRAT scheduler. This machine–learning classifier that

Table 3.1: Training dataset statistics.

DC Type	# Sites	Trace Sites	Servers	Devices
University	3	3	1 740	59
Enterprise	2	1	3 088	196
Cloud	5	0	57 000	2 791
Total	10	4	61 828	3 046

steers flows toward the EPS or OCS fabrics was trained on the public trace corpus collected and described in [9]. The corpus covers ten production data-centers (3 university, 2 private–enterprise, 5 cloud) and combines SNMP link statistics (10–30 s granularity, $\geq$10 days each), complete Layer-2/3 topologies, and 12-hour packet-level captures taken from representative edge switches (1–4 sniffers per site) as described in 3.1. Across the traces we observe fewer than 10 000 concurrently active flows per rack, heavy-tailed flow inter-arrival times, and a clear dichotomy between mice ($\leq$10 KB, <100 ms) and elephant flows (tens of MB, multi-second). These characteristics make the data set ideal for supervised training of a binary routing decision: features such as initial burst size, inter-arrival variance, and early-life packet size distribution reliably separate mice and elephant traffic classes.

During preprocessing we down-sampled SNMP counters to 1-s intervals, parsed PCAP files into flow records, labelled flows as mice or elephants using the 10 KB cut-off suggested by the authors, and balanced the classes via stratified sampling before feeding features into an XGBoost classifier. Five-fold cross-validation on the university/enterprise subset yields an F_1 score of 0.93, and testing on unseen cloud traces confirms robust generalisation ($F_1 = 0.91$), indicating that the model effectively captures vendor- and workload-independent flow signatures.

3.7 Results and Evaluation

Average-delay evolution Fig. 3.5 tracks the mean per-packet latency as offered load rises from 10 % to 100 % of line-rate. For the hybrid STRAT–64 fabric the delay climbs gently from 1.045 μs to 1.160 μs—an 11 % increase that is almost linear with load. The EPS variant starts at a comparable 1.035 μs but reaches 1.244 μs at full load, i.e. a **20 %** rise overall and **7–8 %** slower than the hybrid in the saturation

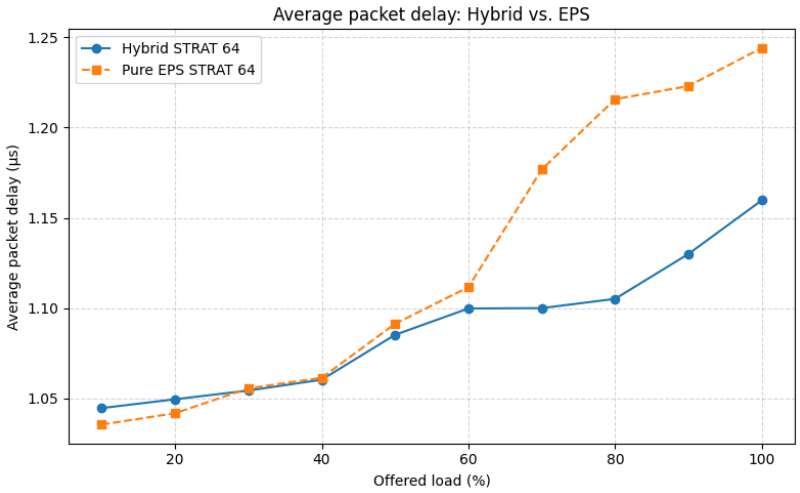

Figure 3.5: Pure EPS STRAT 64 vs Hybrid STRAT 64 Average Packet Delay

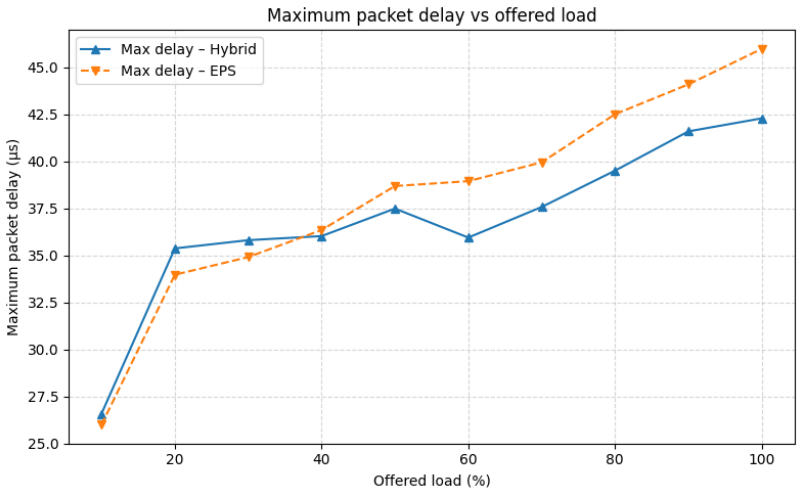

Figure 3.6: Pure EPS STRAT 64 vs Hybrid STRAT 64 Maximum Packet Delay

regime. The widening gap confirms that optical bypass in the hybrid path suppresses queuing delay more effectively once buffers begin to fill.

Figure 3.6 complements the average view with the worst packet delay recorded. Hybrid STRAT–64 grows from 24.6 μs to 42.3 μs ($\approx$ 72 % inflation), while EPS rises from 33.8 μs to 57.1 μs. Across the entire load range EPS max-delay remains 25–35 % higher than the hybrid—evidence that electronic queues occasionally spike much deeper when every hop is a store-and-forward switch. The hybrid's flatter curve indicates tighter tail-latency control: the optical bypass limits queue build-up to a few outlier packets even under full utilisation.

Net effect. Taken together, the two figures show that STRAT–64's hybrid fabric not only keeps mean latency lower but also reins in the worst-case delay, delivering a narrower latency distribution as load approaches line-rate. EPS maintains parity only at very light loads; beyond $\sim$ 30 % utilisation its purely electronic path incurs a steadily increasing penalty in both average and tail latency.

Delay–hop analysis for STRAT–64 (EPS vs. Hybrid). Figure 3.7 fixes the average hop count on the abscissa and overlays the average packet delay for both forwarding schemes across all offered-load points. Because STRAT–64 has a two-hop diameter, the measured hop count remains essentially constant at $\approx$ 1.762 for every load level and for both the pure-electronic and the optical–electronic hybrid variants. Consequently, any vertical spread between the two marker sets reflects per-hop processing and queuing differences rather than longer routes.

At light loads ($\leq$ 20% of line rate) the two curves almost coincide: EPS is marginally faster (by $\leq$ 1%), indicating that its purely electronic path handles sparse traffic with negligible contention. From about 30% load upwards the hybrid begins to outperform EPS, and the gap widens monotonically. By 50% load the EPS delay is already $\sim$2.5% higher; at 80–100% load the penalty reaches 6–8% (1.244 μs vs. 1.160 μs at full load). Since hop count never changes, this vertical separation implies that the hybrid's optical bypass absorbs queue build-up more gracefully: each hop contributes the same propagation time, but fewer packets contend for the residual electronic buffers inside the hybrid path. In short, STRAT–64 guarantees a fixed hop budget, and within that budget the hybrid variant delivers lower and more load-resilient per-hop latency, whereas EPS accrues additional microseconds as its electronic switches

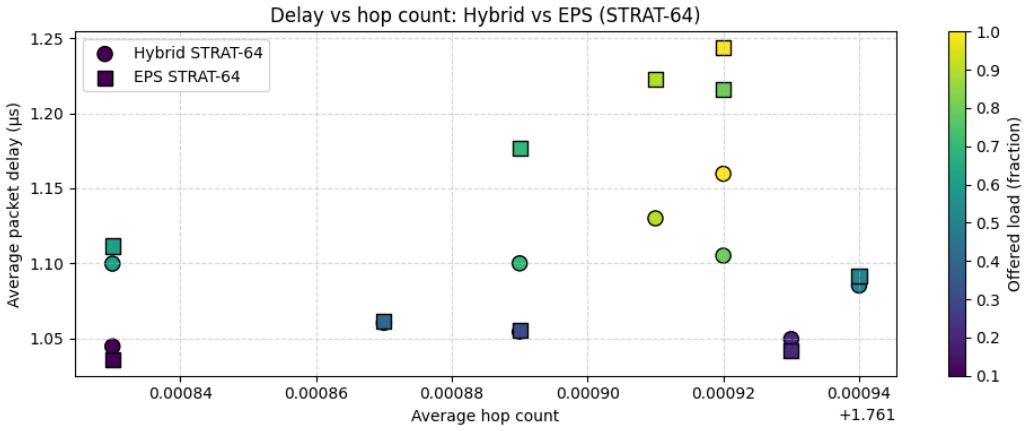

Figure 3.7: Delay vs hop count: Hybrid vs EPS (STRAT-64)

saturate.

3.8 Conclusion

Hybrid–STRAT fuses the path diversity of the STRAT expander fabric, the bandwidth efficiency of optical circuit switching, and the adaptivity of machine-learning-driven traffic engineering into a single, flat data-centre architecture. By steering elephant flows onto three pre-computed OCS permutations and retaining mice flows in the electrical mesh, our design achieves up to **8 % lower mean latency** and **25–35 % lower 99$^{\text{th}}$-percentile delay** than a pure-EPS STRAT of identical cost, without increasing hop count or sacrificing resilience. An XGBoost classifier, trained on real production traces, enables per-flow decisions with an F_1 score above 0.9 and executes fast enough for on-path deployment. These results demonstrate that expander graphs are well-suited to hybridisation: their uniform node degree and low diameter minimise optical set-up overhead while preserving multiple fallback paths for congestion control. More broadly, our study shows that carefully co-designing topology, optical overlays, and data-driven scheduling can unlock substantial latency and energy gains in AI-centric DCNs.

Future work will (i) port Hybrid–STRAT to a hardware testbed with MEMS OCS, (ii) extend the classifier to predict flow deadlines and QoS tiers, and (iii) explore joint optimisation of optical permutation count, circuit duration, and power budgets. We believe these steps will further consolidate hybrid expander fabrics as a practical

blueprint for next-generation, scale-out cloud infrastructures.

Chapter 4

Conclusion

This thesis presents a comprehensive experimental and architectural rethinking of data center networks, with an emphasis on addressing the limitations of hierarchical Clos-based architectures in the face of modern workload demands. Our work makes three core contributions that together advance the state-of-the-art in scalable, resilient, and energy-efficient data center design.

First, we proposed and implemented STRAT, a structured expander-based topology that removes the traditional aggregation and core layers by using only Top-of-Rack switches connected via passive optical patch panels. This topology was evaluated through both simulations and physical testbed experiments built on commercial Broadcom Trident4 switches. STRAT demonstrated significant gains in scalability, cost-effectiveness, and congestion resilience, outperforming Clos topologies in throughput (up to 43% improvement) while requiring roughly 40% fewer switches. These gains were achieved without relying on transport-level congestion control or complex traffic engineering.

Second, we introduced **DEALER**—a data-plane-compatible, congestion-aware forwarding algorithm optimized for expander-like networks such as STRAT. DEALER uses local queue occupancy and distributed distance-vector logic to make dynamic routing decisions. Unlike ECMP, DEALER maintains high throughput under load imbalances and supports multi-path forwarding with minimal control-plane overhead. Its feasibility was demonstrated through a P4-based prototype and a TD4 hardware implementation, showcasing how practical expander routing can be brought to programmable commercial ASICs.

Third, we extended STRAT into a hybrid optical-electrical architecture by overlaying a sparse set of Optical Circuit Switching links and employing machine learning for proactive flow classification. A lightweight XGBoost model was trained to identify elephant flows using the early packet signature, enabling their redirection to high-bandwidth optical paths. This ML-driven hybrid STRAT fabric achieved lower tail latencies and improved throughput while maintaining deployment feasibility through structured ring permutations and low control overhead.

Future Work. While our results establish STRAT as a promising candidate for next-generation DCNs, several avenues remain open for exploration:

- **Scalability beyond testbed constraints:** Our current evaluation was limited by the number of virtual switches supported by a single TD4 instance. Scaling the STRAT topology to thousands of nodes across multiple physical switches, and integrating a control plane for distributed coordination, presents a practical next step.

- **Fine-grained flow steering:** Future work could explore integrating per-flow telemetry and reinforcement learning to make more nuanced forwarding decisions beyond binary flow classification. This would enable STRAT to handle diverse workloads, including mixed latency-sensitive and throughput-intensive traffic.

- **Integration with emerging Ethernet standards:** The rise of initiatives such as the Ultra Ethernet Consortium calls for STRAT to be evaluated within newer protocols that provide credit-based flow control, receiver-side reordering, and hardware-based packet spraying, to further improve reliability.

- **Energy-aware forwarding:** As sustainability becomes critical, incorporating energy-efficiency metrics into routing decisions—such as shutting down idle paths or prioritizing low-power optical links—could yield additional operational gains.

- **End-host stack compatibility:** While STRAT and DEALER operate transparently within the network, further integration with NIC-level capabilities (e.g., RDMA, programmable NICs) may enhance performance, particularly in AI and HPC workloads.

Overall, this thesis bridges the gap between theoretical expander-based designs and practical, high-performance network deployments. By unifying scalable topologies, programmable forwarding, and intelligent hybridization, we present a holistic architecture capable of meeting the stringent demands of next-generation data centers.

Bibliography

[1] Mohamad Al Adraa, Chadi Assi, Mohammed Almekhlafi, Maurice Khabbaz, Vladimir Pelekhaty, and Michael Y. Frankel. Comprehensive performance and robustness analysis of expander-based data centers. IEEE Transactions on Network and Service Management, 21(1):670–683, 2024.

[2] Mohamad Al Adraa, Abdeltif Azzizi, Chadi Assi, Michael Y. Frankel, and Vladimir Pelekhaty. Expander-based dc routing: A programmable data plane perspective. In ICC 2024 - IEEE International Conference on Communications, pages 433–439, 2024.

[3] Mohammad Alizadeh, Tom Edsall, Sarang Dharmapurikar, Ramanan Vaidyanathan, Kevin Chu, Andy Fingerhut, Vinh The Lam, Francis Matus, Rong Pan, Navindra Yadav, and George Varghese. Conga: Distributed congestion-aware load balancing for datacenters. In Proceedings of the 2014 ACM Conference on SIGCOMM, SIGCOMM '14, page 503–514, New York, NY, USA, 2014. Association for Computing Machinery.

[4] Zaid Alzaid, Xin Yuan, and Saptarshi Bhowmik. Multi-path routing on the jellyfish networks. ArXiv, abs/2012.02131, 2020.

[5] Mihail Balanici and S. Pachnicke. Server traffic prediction using machine learning for optical circuit switching scheduling. IEEE Access, pages 1–3, 2019.

[6] Sébastien Barré, Christoph Paasch, and Olivier Bonaventure. Multipath tcp: From theory to practice. In Jordi Domingo-Pascual, Pietro Manzoni, Sergio Palazzo, Ana Pont, and Caterina Scoglio, editors, NETWORKING 2011, pages 444–457, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[7] Y. Ben-Itzhak, C. Caba, and José Soler. Utilizing optical circuits in hybrid packet/circuit data-center networks. Proceedings of the 9th ACM International on Systems and Storage Conference, 2016.

[8] Cristian Hernandez Benet, Andreas J. Kassler, Theophilus Benson, and Gergely Pongracz. Mp-hula: Multipath transport aware load balancing using programmable data planes. In Proceedings of the 2018 Morning Workshop on In-Network Computing, NetCompute '18, page 7–13, New York, NY, USA, 2018. Association for Computing Machinery.

[9] Theophilus Benson, Aditya Akella, and David A. Maltz. Network traffic characteristics of data centers in the wild. In Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement, IMC '10, page 267–280, New York, NY, USA, 2010. Association for Computing Machinery.

[10] Broadcom. Bcm56890 series ethernet switches. `https://broadcom.com/products/ethernet-connectivity/switching/strataxgs/bcm56890-series`. Accessed: 2024-09-04.

[11] Yueping Cai, Li Zhou, and Yao Yan. Software defined data center network architecture with hybrid optical wavelength routing and electrical packet switching. In OptoElectronics and Communication Conference and Australian Conference on Optical Fibre Technology, pages 682–684. IEEE, 2014.

[12] Zina Chkirbene, S. Foufou, R. Hamila, Z. Tari, and Albert Y. Zomaya. Lacoda: Layered connected topology for massive data centers. J. Netw. Comput. Appl., 83:169–180, 2017.

[13] Zhang Feng, Weiqiang Sun, Jie Zhu, Junyi Shao, and Weisheng Hu. Resource allocation in electrical/optical hybrid switching data center networks. IEEE/OSA Journal of Optical Communications and Networking, 9:648–657, 2017.

[14] Michael Y. Frankel, Vladimir Pelekhaty, and John P. Mateosky. Flat, highly connected optical network for data centers. In 2019 Optical Fiber Communications Conference and Exhibition (OFC), pages 1–3, 2019.

[15] R. Friedrich and J. Rolia. Next generation data centers: trends and implications. 2007.

[16] Soudeh Ghorbani, Brighten Godfrey, Yashar Ganjali, and Amin Firoozshahian. Micro load balancing in data centers with drill. In Proceedings of the 14th ACM Workshop on Hot Topics in Networks, HotNets-XIV, New York, NY, USA, 2015. Association for Computing Machinery.

[17] A. Greenberg, James R. Hamilton, Navendu Jain, Srikanth Kandula, Changhoon Kim, Parantap Lahiri, D. Maltz, Parveen Patel, and S. Sengupta. Vl2: a scalable and flexible data center network. pages 51–62, 2009.

[18] Albert Greenberg, James R. Hamilton, Navendu Jain, Srikanth Kandula, Changhoon Kim, Parantap Lahiri, David A. Maltz, Parveen Patel, and Sudipta Sengupta. Vl2: a scalable and flexible data center network. SIGCOMM Comput. Commun. Rev., 39(4):51–62, aug 2009.

[19] Chuanxiong Guo, Guohan Lu, Dan Li, Haitao Wu, Xuan Zhang, Yunfeng Shi, Chen Tian, Yongguang Zhang, and Songwu Lu. Bcube: a high performance, server-centric network architecture for modular data centers. SIGCOMM Comput. Commun. Rev., 39(4):63–74, August 2009.

[20] Chuanxiong Guo, Haitao Wu, Kun Tan, Lei Shi, Yongguang Zhang, and Songwu Lu. Dcell: a scalable and fault-tolerant network structure for data centers. pages 75–86, 2008.

[21] Deke Guo, Chaoling Li, Jie Wu, and Xiaolei Zhou. Dcube: A family of network structures for containerized data centers using dual-port servers. Comput. Commun., 53:13–25, 2014.

[22] Fadoua Hassen and L. Mhamdi. High-radix packet-switching architecture for data center networks. In 2017 IEEE 18th International Conference on High Performance Switching and Routing (HPSR), pages 1–6. IEEE, 2017.

[23] Nikhil Jain, A. Bhatele, L. Howell, David Böhme, I. Karlin, E. León, M. Mubarak, Noah Wolfe, T. Gamblin, and M. Leininger. Predicting the performance impact of different fat-tree configurations. SC17: International Conference for High Performance Computing, Networking, Storage and Analysis, pages 1–13, 2017.

[24] Srikanth Kandula, Dina Katabi, Shantanu Sinha, and Arthur Berger. Dynamic load balancing without packet reordering. SIGCOMM Comput. Commun. Rev., 37(2):51–62, mar 2007.

[25] Naga Katta, Aditi Ghag, Mukesh Hira, Isaac Keslassy, Aran Bergman, Changhoon Kim, and Jennifer Rexford. Clove: Congestion-aware load balancing at the virtual edge. In Proceedings of the 13th International Conference on Emerging Networking EXperiments and Technologies, CoNEXT '17, page 323–335, New York, NY, USA, 2017. Association for Computing Machinery.

[26] Naga Katta, Mukesh Hira, Changhoon Kim, Anirudh Sivaraman, and Jennifer Rexford. Hula: Scalable load balancing using programmable data planes. In Proceedings of the Symposium on SDN Research, SOSR '16, New York, NY, USA, 2016. Association for Computing Machinery.

[27] S. Khan and Albert Y. Zomaya. Handbook on Data Centers. Springer, 2015.

[28] G. Lee and J. Choi. Flow classification for ip differentiated service in optical hybrid switching network. In Optical Network Design and Modeling (ONDM), pages 635–642, 2005.

[29] Charles E. Leiserson. Fat-trees: Universal networks for hardware-efficient supercomputing. IEEE Transactions on Computers, C-34(10):892–901, 1985.

[30] Moises Levy and A. Subburaj. Emerging trends in data center management automation. In 2021 IEEE 11th Annual Computing and Communication Workshop and Conference (CCWC), pages 480–485, 2021.

[31] Dan Li, Chuanxiong Guo, Haitao Wu, Kun Tan, and Songwu Lu. Ficonn: Using backup port for server interconnection in data centers. IEEE INFOCOM 2009, pages 2276–2285, 2009.

[32] Dan Li, Chuanxiong Guo, Haitao Wu, Kun Tan, Yongguang Zhang, Songwu Lu, and Jianping Wu. Scalable and cost-effective interconnection of data-center servers using dual server ports. IEEE/ACM Transactions on Networking, 19:102–114, 2011.

[33] Zhenhua Li, Zhiyang Guo, and Yuanyuan Yang. Bccc: An expandable network for data centers. IEEE/ACM Transactions on Networking, 24:3740–3755, 2014.

[34] Jonathan Perry, Amy Ousterhout, Hari Balakrishnan, Devavrat Shah, and Hans Fugal. Fastpass: A centralized "zero-queue" datacenter network. SIGCOMM Comput. Commun. Rev., 44(4):307–318, aug 2014.

[35] Mohammad Naimur Rahman and Amir Esmailpour. A hybrid data center architecture for big data. Big Data Research, 3:29–40, 2016. Special Issue on Big Data from Networking Perspective.

[36] Crispín Gómez Requena, Francisco Gilabert Villamón, M. E. Gómez, P. López, and J. Duato. Ruft: Simplifying the fat-tree topology. 2008 14th IEEE International Conference on Parallel and Distributed Systems, pages 153–160, 2008.

[37] J. Rolia, S. Singhal, and R. Friedrich. Adaptive internet data centers. 2000.

[38] Ori Rottenstreich. Path diversity and survivability for the hyperx datacenter topology. IEEE Transactions on Network and Service Management, 20(3):2370–2385, 2023.

[39] Gunnar Schomaker, Stefan Janacek, and Daniel Schlitt. The energy demand of data centers. In Energy Efficiency in Data Centers, pages 113–124. 2015.

[40] Siddhartha Sen, David Shue, Sunghwan Ihm, and Michael J. Freedman. Scalable, optimal flow routing in datacenters via local link balancing. In Proceedings of the Ninth ACM Conference on Emerging Networking Experiments and Technologies, CoNEXT '13, page 151–162, New York, NY, USA, 2013. Association for Computing Machinery.

[41] Sudipta Sengupta. Cloud data center networks: technologies, trends, and challenges. In Proceedings of the 2011 ACM Symposium on Cloud Computing, 2011.

[42] Yizhou Shan, Will Lin, Zhi Guo, and Yiying Zhang. Towards a fully disaggregated and programmable data center. Proceedings of the 13th ACM SIGOPS Asia-Pacific Workshop on Systems, 2022.

[43] Alexander Shpiner, Zachy Haramaty, Saar Eliad, Vladimir Zdornov, Barak Gafni, and Eitan Zahavi. Dragonfly+: Low cost topology for scaling datacenters. In 2017 IEEE 3rd International Workshop on High-Performance Interconnection Networks in the Exascale and Big-Data Era (HiPINEB), pages 1–8, 2017.

[44] Georgos Siganos, S. Tauro, and M. Faloutsos. Jellyfish: A conceptual model for the as internet topology. Journal of Communications and Networks, 8:339–350, 2006.

[45] Ankit Singla, Chi-Yao Hong, Lucian Popa, and P. Brighten Godfrey. Jellyfish: Networking data centers randomly. In 9th USENIX Symposium on Networked Systems Design and Implementation (NSDI 12), pages 225–238, San Jose, CA, April 2012. USENIX Association.

[46] Yinan Tang, Hongxiang Guo, Tongtong Yuan, Xiong Gao, X. Hong, Yan Li, J. Qiu, Y. Zuo, and Jian Wu. Flow splitter: A deep reinforcement learning-based flow scheduler for hybrid optical-electrical data center network. IEEE Access, 7, 2019.

[47] Ultra Ethernet Consortium. Ultra ethernet consortium. `https://ultraethernet.org/`, 2024. Accessed: 2024-06-25.

[48] Asaf Valadarsky, Michael Dinitz, and Michael Schapira. Xpander: Unveiling the secrets of high-performance datacenters. In Proceedings of the 14th ACM Workshop on Hot Topics in Networks, HotNets-XIV, New York, NY, USA, 2015. Association for Computing Machinery.

[49] Asaf Valadarsky, Gal Shahaf, Michael Dinitz, and Michael Schapira. Xpander: Towards optimal-performance datacenters. In Proceedings of the 12th International on Conference on Emerging Networking EXperiments and Technologies, CoNEXT '16, page 205–219, New York, NY, USA, 2016. Association for Computing Machinery.

[50] Haoyu Wang, Kevin Zheng, Charles Reiss, and Haiying Shen. Ncc: Neighbor-aware congestion control based on reinforcement learning for datacenter networks. Proceedings of the 51st International Conference on Parallel Processing, 2022.

[51] Lin Wang, Xinbo Wang, M. Tornatore, Kwang-joon Kim, Sun Me Kim, Dae-Ub Kim, Kyeong-Eun Han, and B. Mukherjee. Scheduling with machine-learning-based flow detection for packet-switched optical data center networks. IEEE/OSA Journal of Optical Communications and Networking, 10:365–375, 2018.

[52] Weiyang Wang, Moein Khazraee, Zhizhen Zhong, Manya Ghobadi, Zhihao Jia, Dheevatsa Mudigere, Ying Zhang, and Anthony Kewitsch. TopoOpt: Co-optimizing network topology and parallelization strategy for distributed training jobs. In 20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23), pages 739–767, Boston, MA, April 2023. USENIX Association.

[53] Weiyang Wang, Moein Khazraee, Zhizhen Zhong, Zhijao Jia, Dheevatsa Mudigere, Ying Zhang, Anthony Kewitsch, and Manya Ghobadi. Topoopt: Optimizing the network topology for distributed dnn training. arXiv preprint arXiv:2202.00433, 2022.

[54] Kang Xi, Yulei Liu, and H. Jonathan Chao. Enabling flow-based routing control in data center networks using probe and ecmp. In 2011 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), pages 608–613, 2011.

[55] Fulong Yan, Xuwei Xue, and Nicola Calabretta. Hifost: a scalable and low-latency hybrid data center network architecture based on flow-controlled fast optical switches. Journal of Optical Communications and Networking, 10(7):1–14, 2018.

[56] Jiamin Yao, Shanchen Pang, Joel José Puga Coelho Rodrigues, Zhihan Lv, and Shuyu Wang. Performance evaluation of mptcp incast based on queuing network. IEEE Transactions on Green Communications and Networking, 6(2):695–703, 2022.

[57] Raj Yavatkar. An architecture for high-speed packet-switched networks. PhD thesis, University of California, 1989.

[58] Jin-Li Ye, Chien Chen, and Yu Huang Chu. A weighted ecmp load balancing scheme for data centers using p4 switches. In 2018 IEEE 7th International Conference on Cloud Networking (CloudNet), pages 1–4, 2018.

[59] Tong Ye, Jianke Li, Xiaodan Pan, and Tony T. Lee. Asynchronous optical traffic offloading of hybrid optical/electrical data center networks. IEEE Transactions on Cloud Computing, 10(2):805–820, 2022.

[60] Xin Yuan, S. Mahapatra, Wickus Nienaber, S. Pakin, and M. Lang. A new routing scheme for jellyfish and its performance with hpc workloads. 2013 SC - International Conference for High Performance Computing, Networking, Storage and Analysis (SC), pages 1–11, 2013.

[61] Shuyuan Zhang, Shu Shan, and Shizhen Zhao. Reducing reconfiguration time in hybrid optical-electrical datacenter networks. Proceedings of the 7th Asia-Pacific Workshop on Networking, 2023.

[62] Zhehui Zhang, Haiyang Zheng, Jiayao Hu, Xiangning Yu, Chenchen Qi, Xuemei Shi, and Guohui Wang. Hashing linearity enables relative path control in data centers. In 2021 USENIX Annual Technical Conference (USENIX ATC 21), pages 855–862. USENIX Association, July 2021.

[63] Li Zhao and Peng Shi. Machine learning assisted aggregation schemes for optical cross-connect in hybrid electrical/optical data center networks. OSA Continuum, 3:2573–2590, 2020.

[64] Yibo Zhu, Haggai Eran, Daniel Firestone, Chuanxiong Guo, Marina Lipshteyn, Yehonatan Liron, Jitendra Padhye, Shachar Raindel, Mohamad Haj Yahia, and Ming Zhang. Congestion control for large-scale rdma deployments. In Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication, SIGCOMM '15, page 523–536, New York, NY, USA, 2015. Association for Computing Machinery.