Balancing SDN Control Plane Availability and Traffic Engineering Efficiency in Data Centers

Brian Chang[†]*, Keqiang He[§]*, Shawn Shuoshuo Chen[‡], Jiaxin Lin[†], Mingyang Zhang[||], Wenfei Wu[††], Aditya Akella[†]

[†]Univerity of Texas at Austin [§]Shanghai Jiao Tong University [‡]Carnegie Mellon University [||]Google [††]Peking University

***Abstract*—Many proposals have demonstrated the efficiency advantages of software-defined networking (SDN) in managing data center networks. Common practices employ centralized traffic engineering (TE) in the SDN control plane to optimize load balancing and throughput. Meanwhile, for high availability purposes, the control plane is partitioned to ensure the impact of a single faulty controller is contained. However, the interaction between these two aspects is often overlooked. In particular, we show that the current control plane partitioning approach leads to imbalanced link loads and degraded application performance. To address this issue, we propose *virtual slicing*, a new control plane partitioning scheme. Virtual slicing achieves desirable traffic engineering performance while retaining the availability guarantees from the current approach. Virtual slicing is implemented and evaluated with real-world and synthetic traffic traces on production spine-free data center networks. Results show that virtual slicing reduces tail link utilizations by up to 28.4%, and improves flow completion times by up to 36%.**

***Index Terms*—datacenter networks, software-defined networking, control plane partition, traffic engineering**

I. Introduction

Data center networks (DCNs) need to achieve two crucial objectives: (1) efficient utilization of the network infrastructure and (2) limited impact from failures. Traffic engineering (TE) addresses the efficiency requirement by balancing link loads and maximizing throughput [1]–[4]. Recently, TE implemented in a centralized software-defined networking (SDN) control plane [5]–[11] has seen a growing presence. Compared to traditional networks with distributed control, a failed or faulty control plane in SDN has a global impact on the network. For example, the control plane may route traffic over links with insufficient bandwidth or no reachability to a destination. Unfortunately, current efforts to limit the scope of such impact do not interact well with efficiency-centric TE implementations or may even impose a penalty on TE performance.

Modern SDN designs [7], [10] use an approach named *physical sharding* to divide the data plane network into multiple disjoint sub-networks and manage each sub-network with its own independent SDN controller. The monolithic control plane is effectively divided into several control plane partitions with limited impact under faulty behavior. Ideally, all sub-networks have equal aggregated bandwidth (also called capacity), and TE routes equal volumes of traffic through each sub-network. Therefore, a faulty control plane partition in one sub-network does not affect more traffic than the same fault in another sub-network. However, generating a set of equal-capacity sub-networks proves difficult due to topological asymmetry. Namely, the number of links is not a multiple of the desired number of sub-networks. With unequal partitioning, sub-networks with lower capacity have to support the same traffic volume at the cost of higher link utilization. If TE adjusts the traffic load on each sub-network based on their respective capacity, the load can be balanced, but certain sub-networks will have to serve more traffic than others. This is non-ideal since more traffic will be impacted when these particular sub-networks experience control plane faults.

DCN operators seek to avoid unequal partitioning by enforcing constraints in topology design, such as ensuring that each switch connects to all peer switches with equal link count. With the trend towards spine-free DCNs [8], [12]–[15], this technique faces challenges as it is increasingly difficult to achieve equal partitioning.

How can we partition the network equally such that each control plane partition causes minimal impact while maintaining TE performance close to optimal? In this paper, we propose *virtual slicing*, an effective control plane partition scheme that meets this goal. The idea of virtual slicing is simple: unlike physical sharding, where a whole link is assigned to a control plane partition, virtual slicing slices a link into several "virtual" links and exclusively assigns each virtual link to a control plane instance. In essence, each control plane partition in virtual slicing sees the same complete network topology but at a fraction of the total capacity, hence effectively managing a partition of the network. Isolation of the virtual links is achieved by dedicating a portion of the switch memory to each control plane partition. Each control plane partition installs switch rules in its dedicated switch memory region to control how traffic enters the virtual links it manages.

Virtual slicing offers three benefits compared to physical sharding. First, it does not suffer from unequal partitioning, as it is trivial to slice links equally into the desired number of virtual links with equal capacity. The constraints in topology design mandated by physical sharding are hence unnecessary. Second, virtual slicing is flexible in creating an arbitrary number of partitions, whereas in physical sharding, each sub-topology gets increasingly sparse as the number of aggregation blocks and/or partitions increase. Third, virtual slicing can

*: Work done while at Google.

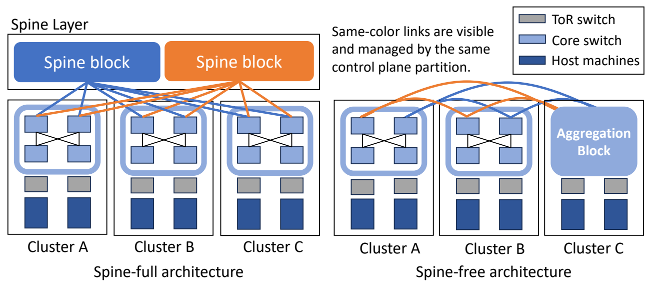


Figure 1: Spine-full and spine-free DCN topologies.

make use of more available capacity in case of controller failures. With physical sharding, a failed controller leaves links in its partition unmanaged. Traffic has to be routed around this partition (and through other partitions) to avoid being dropped or mishandled. This leads to load imbalance because links in the failed partition are idle while links in other healthy partitions are oversubscribed. On the other hand, links in virtual slicing are shared by multiple partitions. Therefore, to avoid a failed partition, traffic only needs to follow a different set of switch rules installed by another controller, but does not have to completely avoid the links.

We evaluate virtual slicing both in production DCNs and in a simulator using a wide range of network topologies and traffic patterns. Results show that virtual slicing achieves up to 28.4% better performance in tail link utilization when compared with physical sharding in a large-scale spine-free DCN with unequal partitioning. Virtual slicing also outperforms physical sharding in application-level performance, improving tail flow completion times by up to 36%.

In summary, this paper makes the following contributions:

- We identify the mismatch and challenges of applying the state-of-the-art control plane partition scheme (i.e., physical sharding) to the emerging spine-free DCNs.
- We design and implement a novel control plane partition paradigm, virtual slicing, to address the TE performance penalty due to unequal partitioning.
- We conduct experiments using both synthetic and real-world spine-free DCNs and traffic traces; demonstrating that virtual slicing addresses the identified challenges and improves over the current approach (physical sharding).

II. Background

A. Spine-free DCNs

At cloud scale, multi-stage Clos (or *spine-full*) networks [16]–[19], face challenges such as high spine layer cost and power consumption, where the spine layer represents 30% capex and 41% power consumption [8]. Multi-stage Clos topologies also face challenges of hindered hardware upgrade speed, where switches in the spine layer need to be upgraded ahead of the switches in aggregation blocks in order to evolve to higher network speed. As an alternative, researchers and cloud providers are exploring flat, spine-free DCN architectures [8], [12]–[15] that enjoy properties such as reduced construction cost and faster hardware upgrade cycles. Figure 1 shows a side-by-side comparison of the two topologies. In both topologies, the clusters are still based on non-blocking Clos networks: aggregation blocks consist of 2-tier Clos, and each ToR switch connects to all the core switches in the aggregation block. In spine-free DCNs, aggregation blocks form a full mesh instead of connecting to the spine blocks.

B. Centralized TE in DCNs

To optimize link bandwidth usage and prevent congestion, data centers employ traffic engineering (TE) to route traffic through a set of routes chosen by a TE algorithm. In SDN-managed DCNs, a centralized controller implements TE by solving a multi-commodity flow problem using linear programming. The primary objective used by most TE systems is to minimize maximum link utilization (MLU) [20]–[22], which is closely tied to application performance [20], [23], [24]. The output of TE is a set of weighted cost multipath (WCMP) [5], [25] weights that determine traffic splitting ratios across multiple paths. Spine-full topologies utilize 2 block-level hops through the spine layer [18] for TE and routing, while spine-free topologies use both direct 1-hop and transit 2-hop paths [8]. This block-level view of the topology also naturally divides the control plane into inter-cluster routing controllers and intra-cluster routing controllers, where inter-cluster controllers solve block-level TE, and the intra-cluster controllers translate WCMP weights generated by TE into switch table rules inside each cluster [5], [25].

The output of TE is implemented in switches with a combination of flow table and group table rules. When a packet arrives at a switch, longest prefix match (LPM) is performed against a flow table containing TCAM entries that implement IP prefix matching rules. The most specific matching rule is selected, pointing to a group in the SRAM-based group table, which contains the group actions for ECMP or WCMP. WCMP weights are represented by replicating group table entries to match each path's weight. This mechanism is further elaborated in Section IV-B and Figure 5.

C. Control plane sharding for high availability

High availability is a crucial design goal in data center networks hosting critical applications and services, as network availability directly impacts application performance, business reputation, and revenue [26], [27]. To achieve this, modern SDN-managed DCNs employ a control plane partitioning strategy we refer to as *physical sharding*. This involves dividing the control plane into multiple independent partitions, each responsible for a disjoint subset of physical links in the network. The combination of a controller partition and its exclusive sub-topology is referred to as a shard. Each cluster and its intra-block links naturally map to an independent shard per cluster, while the inter-block control plane is divided into multiple shards. Each inter-block shard instantiates its own TE pipeline, solving TE and routing traffic within its exclusive sub-topology. This ensures that controller faults in one partition only impact a portion of the total traffic carried by the DCN, minimizing the fault's scope and impact.

Figure 1 shows how physical sharding is applied to spine-full and spine-free topologies respectively. The color of a link indicates the shard to which it is assigned. In spine-full topologies, aggregation block uplinks are split directly to different shards, with all links connected to a non-blocking spine layer. In contrast, spine-free topologies require clusters to distribute aggregation block uplinks among other peer clusters, creating a sparser full mesh with fewer links between each aggregation block pair. The following section will illustrate how the inherent sparsity of spine-free architectures leads to performance issues when implementing physical sharding.

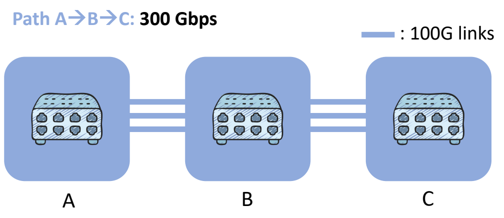
In addition to control plane partitioning, SDN-managed DCNs also employ controller replication techniques to ensure availability. Control plane partitioning contains the impact of corrupted or faulty control plane partitions on traffic forwarding; while controller replication, such as main-follower architectures, ensures high availability within each partition. Control plane partitioning is usually applied first, followed by replication of each shard's controller partition to achieve high availability. This paper focuses on control plane partitioning and its interactions with TE.

III. Challenges

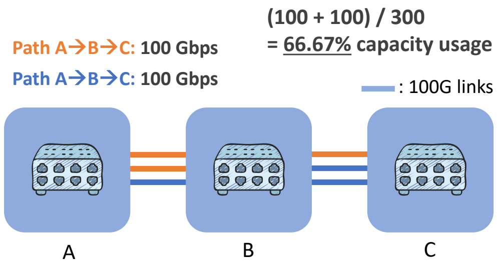
Applying the straightforward sharding scheme above to spine-free topologies, which might be an operator's natural inclination, leads to TE performance penalties, as we will discuss in this section. Under physical sharding, TE performance penalties stem from the *asymmetry of sub-topologies* overseen by different control plane partitions. Unequal partitioning forces sub-topologies with lower capacity to handle the same traffic volume, leading to higher link utilization. As we show next, sub-topology asymmetry leads to several challenges, including inefficient use of the built network capacity and sub-optimal TE solutions. Sub-topology asymmetry is caused by a combination of the following factors:

Limited block radix and large fabric size: The number of uplinks in an aggregation block is called the *radix* of the aggregation block, where the uplinks are used to connect to other blocks in a spine-free DCN. To ensure efficient use of network resources and business capital expenditure, each block has a limited radix (e.g., 256 or 512). When there are a large number of aggregation blocks in a spine-free DCN, links are spread sparsely and there is a limited number of links between block pairs. Partitioning such an already sparse network leads to asymmetric sub-topologies across physical shards. If each aggregation block has a radix r, and the total number of blocks in the network is b, each aggregation block pair will have $\lfloor r/b \rfloor$ links. The number of pairwise connections between blocks decreases as b increases, since r is fixed for each block regardless of the number of blocks in the network.

Increased number of physical shards: As more physical shards are added to further contain the impact of control plane failures, it becomes gradually harder to split the network into identical sub-topologies due to the increased sparsity.



(a) An un-sharded network can fully utilize all the constructed capacity along path A→B→C.



(b) A sharded network with two partitions only utilizes $\frac{2}{3}$ of the constructed capacity along path A→B→C.

Figure 2: Physical sharding leads to loss in usable capacity.

Link or switch failures: In production DCNs, link, port and switch failures are not uncommon [28]. They are either due to scheduled maintenance or unexpected faults. Topology is more irregular when failures occur, and ensuring topology symmetry across physical shards becomes increasingly challenging. The combination of these factors results in sparse and asymmetric sub-topologies, which lead to the performance penalties as described below.

A. Transit path capacity loss

In Section II-B we showed that spine-free topologies rely on non-shortest, 2-hop *transit* paths for routing. However, as we show next, spine-free networks suffer from what we refer to as *transit capacity loss*, where transit paths hold insufficient usable capacity when physical sharding is applied, especially at scale and when the number of control plane shards increases.

As an illustrative example, Figure 2 demonstrates a transit path A→B→C, that might potentially suffer from asymmetric performance penalties. In Figure 2a, path A→B→C utilizes all 300 Gbps of bandwidth; however, in Figure 2b since different shards have disjoint sub-topologies that are asymmetric, the blue shard is bottlenecked on A→B, while the orange shard is bottlenecked at B→C. This results in only a 66.67% utilization in the built network capacity along A→B→C. This limits the usable capacity each shard's TE solver can utilize, causing unnecessary network capacity wastage.

As shown in this example, transit path capacity loss arises due to topology asymmetry across physical shards. To quantify the impact of the capacity loss of transit paths in spine-free DCNs due to physical sharding, we introduce the *fragmenta-*

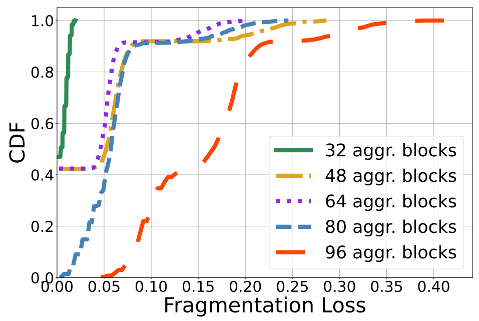


Figure 3: Capacity fragmentation loss as topology scales.

tion loss metric, which measures the degree of underutilization of capacity due to physical sharding:

$$\text{Fragmentation Loss} = 1 - \frac{\text{Usable Capacity}}{\text{Built Capacity}} \qquad (1)$$

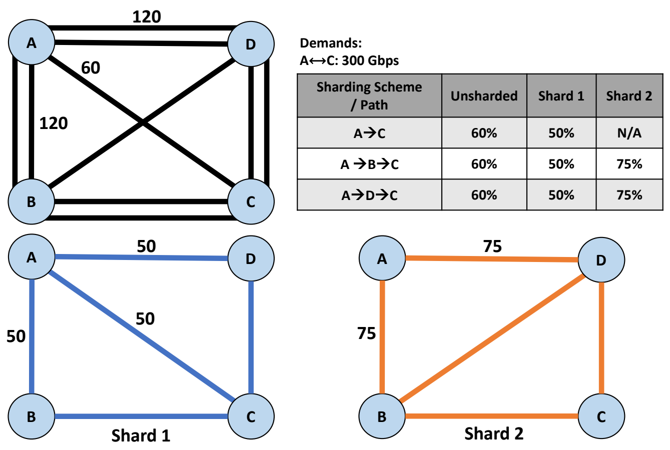
where built capacity refers to the total capacity of the network without sharding. Fragmentation loss increases as the usable capacity decreases, which captures the increased sparsity and asymmetry discussed in Section III.

We then analyzed how fragmentation loss changes under different number of aggregation blocks in the spine-free DCN. To do so, we created synthetic spine-free DCN topologies with various sizes that follow radix and operational constraints identical to our deployed fabric topologies. The generated topologies have 50% of blocks with a radix of 256 and 50% with a radix of 512. Figure 3 shows the fragmentation loss of topologies of different sizes. We can observe a trend where fragmentation loss worsens as the fabric size increases. At 96-block scale, 20% block pairs lose more than 19% transit capacity. Note that topologies analyzed in Figure 3 do not include link failures, which would further worsen transit capacity loss.

B. Suboptimal traffic engineering performance

Physical sharding aims to partition the network into N identical sub-topologies, with each shard carrying an equal amount of traffic to contain failures. However, generating identical sub-topologies in spine-free DCNs is challenging, as we've demonstrated in previous sections. When the topology is unequally partitioned, each shard's controller solves a TE problem based on its local topology, generating a solution that is locally optimal but globally suboptimal.

Figure 4 demonstrates an unsharded topology and two corresponding partitioned topologies with physical sharding using 2 shards. A single traffic demand exists from block A to block C, and each link has a 100 Gbps capacity. The traffic engineering objective is to minimize maximum link utilization (MLU). The numbers along the edges show the optimal traffic split that minimizes MLU. When TE is applied on the unsharded topology, the optimal MLU is 60%. However, splitting the network into two nearly equal physical shards, each handling half the traffic, increases the MLU to 75% on shard B.



Demands:
A↔C: 300 Gbps

Sharding Scheme / Path	Unsharded	Shard 1	Shard 2
A→C	60%	50%	N/A
A→B→C	60%	50%	75%
A→D→C	60%	50%	75%

Figure 4: Unequal partitioning in physical sharding leads to sub-optimal TE performance. Table entries show the link utilization along a certain path.

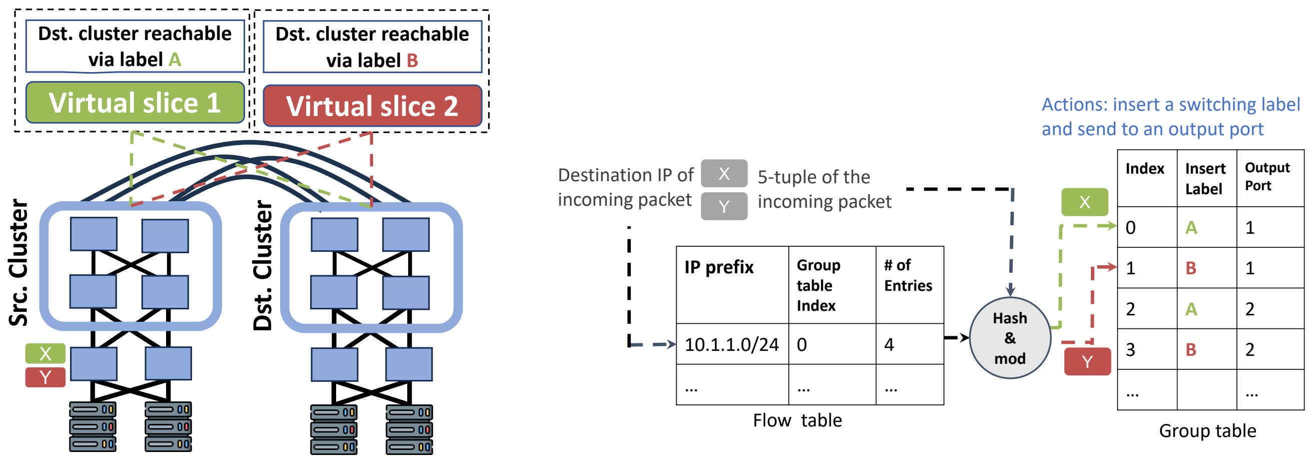
IV. Virtual Slicing

The challenges identified in Section III collectively make physical sharding inherently incompatible with spine-free DCNs. In order to overcome these challenges, we propose *virtual slicing*, an effective control plane partitioning scheme. Unlike physical partitioning that divides the physical network into sub-topologies, virtual slicing maintains a unified topology, therefore avoiding traffic engineering performance penalties while providing equivalent failure containment assurances as physical sharding.

A. Concept

The fundamental concept behind virtual slicing is to partition *switch table resources*, specifically flow tables and group tables, rather than topologically partitioning network links. Each virtual slice is controlled by an independent control plane partition, while all virtual slices share the entire (thus *identical*) physical topology that contains all links when performing TE. Each virtual slice takes this identical topology along with traffic demand as input, and generates its set of WCMP [5], [25] weights to split traffic among multiple block-level paths. Each virtual slice then transforms the generated paths and their weights into flows and groups and programs them into switch tables. By operating on identical, global traffic information and topology inputs across all virtual slices, virtual slicing (1) eliminates the effects of transit path capacity loss and topology asymmetry across shards; and (2) computes a globally-optimal TE solution.

In addition to enhancing performance, control plane failures within each virtual slice only affect traffic routed by that particular slice, achieving similar effect as the isolation provided by physical sharding. With N virtual slices, each controller partition manages $1/N$ of the total traffic volume, confining the impact of a controller failure to the corresponding virtual slice. Failure containment can be further tightened by increasing the number of slices. In contrast, adding more physical shards in spine-free DCNs is challenging because doing so



(a) (b)

Figure 5: Virtual slicing realization with label switching and flow/group tables. (a) Which virtual slice a packet lands on is determined at ToR switches. Dashed lines are controller sessions. (b) The source ToR's group table is populated with one entry per virtual slice per output port, achieving equal traffic split between virtual slices.

will further worsen transit path capacity loss and asymmetry across physical shards.

Furthermore, with physical sharding across N shards, data centers risk losing up to $1/N$ of the fabric capacity when a controller partition fails, as the affected physical shard's capacity becomes unusable. Migrating traffic away from a faulty physical shard is infeasible because other controllers don't have visibility and control over the physical network owned by another partition. In contrast, virtual slicing enables traffic migration across virtual slices (Section IV-B), ensuring no physical network capacity remains unutilized during controller failures.

B. Routing design

Routing in virtual slicing can be realized with classic techniques such as label switching and IP-in-IP tunneling. We elaborate on the use of label switching, though this approach can also leverage IP-in-IP tunneling. Using either technique, packets are tagged upon entering the network to indicate their associated virtual slice. With label switching, each packet is assigned a switching label at the source top-of-rack (ToR) switch and forwarded based on this label until it reaches the destination cluster. There, the packet is eventually forwarded to its destination host via standard IP forwarding based on the destination IP field.

Figure 5 illustrates how a packet is mapped to a specific virtual slice in its source cluster. In the source ToR switch's packet processing pipeline [25], a hash value is first computed based on the packet header's various fields. The pipeline then references the flow table and the computed hash value to identify the corresponding table entry (i.e., actions) in the group table for the packet, as shown in Figure 5b. In this example, packets have an equal probability of being assigned to switching label A or B (thus virtual slice 1 and 2). Flows assigned the same switching label can be sent to port 1 or port 2 with equal probability. Packets of the same flow receive the same deterministic routing decision, landing on the same virtual slice and avoiding packet reordering. Intermediate switches in the aggregation block perform label switching, where traffic from a specific virtual slice destined for a cluster is directed through the network by looking up the assigned forwarding label. Ingress switches of the destination aggregation block ultimately pop the forwarding label and use the inner packet header's destination IP address to route the packets to the correct host.

Figure 5b shows flows from a ToR switch mapped to different virtual slices with equal probability, ensuring each slice handles an equal amount of traffic. Virtual slicing can further achieve weighted traffic splitting across slices by adjusting group table weights via WCMP's entry replication [5], [25]. For instance, doubling the entries that insert label A in Figure 5 results in a 2:1 traffic split between slices 1 and 2. Flexible traffic splitting only requires group table configuration at edge ToR switches, without reprogramming downstream switches. With the usage of group tables, virtual slicing's routing design also enables dynamic traffic migration that allows seamless steering of traffic from faulty to healthy virtual slices. Upon detecting an unhealthy slice, virtual slicing removes the group table entries pointing to the unhealthy slice, migrating all traffic to healthy slices. In the example shown in Figure 5, if the controller in virtual slice 2 is faulty, by removing group table entries pointing to forwarding label B (used by virtual slice 2 for the destination cluster), all traffic would then migrate to virtual slice 1 (using forwarding label A for the same destination), without requiring flow table changes. This process occurs in parallel on ToR switches, which ensures consistent network updates. Dynamic traffic migration ensures no physical network capacity remains unutilized when a virtual slice controller is faulty, unlike physical sharding, where up to $1/N$ of physical capacity is lost with N topology partitions.

C. Switch table usage analysis

We now examine the table resource utilization of virtual slicing in modern commodity switches.

Flow table and label space usage. Each cluster is assigned M forwarding labels for an implementation of virtual slicing with M virtual slices, where each virtual slice is assigned a globally unique label, and the total number of labels required is M multiplied by the number of clusters within the fabric. Considering a maximum practical case of 32 virtual slices in a 96-cluster spine-free fabric, a total of $96 * 32 = 3072$ unique labels required falls well within the capabilities of modern switches [29]–[31]. On the other hand, in a DCN with 96 clusters, there would be 96 prefixes at each ToR for block-level routing (Figure 5b), requiring 96 entries. Thus, virtual slicing's utilization of both unique labels and flow table entries remains well within the capabilities of contemporary switch hardware.

Group table usage. Group tables are the foundation for implementing WCMP. WCMP is usually achieved by replicating ECMP entries in the group table, where the group table's utilization depends on the WCMP reduction algorithm—such an algorithm tries to reduce the number of duplicated entries for efficient table space usage [5], [25], [32]. The design of virtual slicing indicates that the number of entries each slice can utilize is inversely proportional to the total number of slices, i.e., $\frac{M}{\#slices}$. Group table will be under pressure of running out as the number of virtual slices increases. Fortunately, the controllers in each virtual slice often operate on the same topology and traffic input, which results in almost identical WCMP group weights across slices. Switch resource optimization techniques such as group sharing[1] proposed in [5] could help in such scenario. In addition, switch vendors start to support native WCMP weight split techniques in hardware without any ECMP entry replication, e.g., Broadcom Tomahawk4 supports 4,096 WCMP groups with native WCMP weights [31]. New switch hardware with this capability also helps address the resource challenge.

V. Evaluation

In this section, we study the following issues to understand the performance benefits and system characteristics of virtual slicing from different perspectives, and also investigate its practical impact on spine-free DCNs: **Q1.** What is virtual slicing's TE improvement with diverse fabric topologies (e.g., different fabric scales) and traffic patterns (e.g., traffic bursts and real-world traces)? **Q2.** What is virtual slicing's TE improvement with more number of slices? **Q3.** Is virtual slicing's performance gain sensitive to traffic measurement delays? **Q4.** What is virtual slicing's TE improvement under impaired network topologies due to maintenance or unexpected failures? **Q5.** What is virtual slicing's impact on application performance? **Q6.** How does virtual slicing affect TE solving time?

[1]Group sharing enables identical groups to share the same table space/entries.

Exp. group	Fabric characteristics	Traffic characteristics
Synthetic (Section V-B)	Synthesized following block radix constraints.	Synthesized via models that simulate realistic scenarios.
Semi-production (Section V-C)	Synthesized with a topology builder used to design and validate topologies to be deployed.	Same as above.
Production (Section V-C)	Fabrics deployed in production.	Direct traffic collection in production fabrics.
ns-3 [33] (Section V-D)	Synthesized following block radix constraints.	Flows generated via distributions derived from production traces.

Table I: Experiment group configurations.

A. Experiment setup

Implementation. We implemented a virtual slicing and physical sharding evaluation framework modeling a datacenter TE system. The evaluation framework takes topology information and traffic matrix as input, computes TE solutions in the form of block-level path weights and outputs link utilizations. In the physical sharding scheme, direct links between an aggregation block pair are divided, as equally as possible, into k partitions to ensure equal or close to equal network capacities across shards; each physical shard carries $\frac{1}{k}$ traffic. For virtual slicing, we employ k virtual slices, where each slice also carries $\frac{1}{k}$ traffic. We focus on evaluating aggregation block-level link utilization, which is the primary optimization target for TE in spine-free DCNs due to their blocking nature (Section V-B, Section V-C). In addition to TE-related network-level metrics, we also evaluate application-facing metrics with ns-3 [33], a packet-level network simulator (Section V-D).

TE pathing and formulation. The TE system's pathing module allows direct 1-hop and transit 2-hop block-level paths for routing, and employs the widely adapted objective of minimizing the maximum link utilization (MLU) [20]–[22]. TE is then formulated as a multi-commodity flow (MCF) problem [34] and solved by linear programming (Section II-B).

Topologies and traffic traces. We evaluated virtual slicing's performance using synthetic topologies and traces, as well as production spine-free topologies with real-word traces. Table I summarizes the experimental setups used in our evaluation. Specifically, we evaluated (1) synthetic topologies following block radix limitations, (2) topologies generated with a production spine-free topology builder that follow additional operational constraints, and (3) deployed production spine-free topologies. The radix of an aggregation block in these topologies is either 256 or 512.

Parameters for synthesized traffic patterns are chosen to simulate realistic scenarios, and don't represent actual direct production measurements. Our first traffic matrix (TM) follows the gravity model [35], where each aggregation block has a total demand around 40% of its total bandwidth, aiming to represent medium to high traffic load scenarios in data centers. The gravity model states that the amount of traffic between two clusters is proportional to the product of the two cluster's total ingress/egress volume. The second TM is a "gravity+spikes" TM, where random traffic spikes are added on top of the gravity TM to model the burstiness of

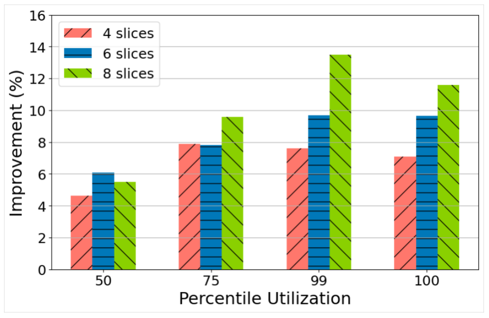


Figure 6: TE improvement w/ more slices/shards.

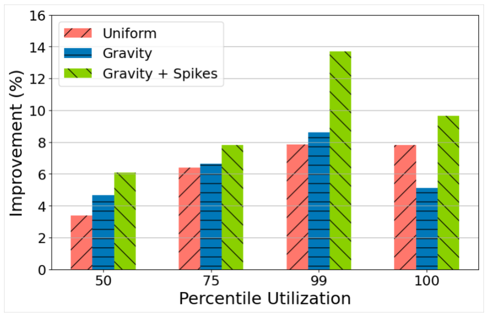


Figure 7: TE improvement w/ different traffic patterns.

DCN traffic [36], [37]. Specifically, with a 15% probability, we increase the volume of a commodity (i.e. traffic demand between a source and destination aggregation block pair) by 50%. We also generated an uniform TM, which has an equal amount of ingress and egress demand that's evenly distributed across all aggregation blocks. Besides synthetic traffic, we also evaluate virtual slicing on traffic collected from production DCN fabrics (Section V-C). Unless otherwise specified, the topologies in evaluation use 4 physical shards/virtual slices. To ensure a fair comparison, the number of physical shards and virtual slices used within each experiment remains constant throughout the evaluation.

B. *TE with different traffic patterns and slices*

Different number of slices. We first evaluate virtual slicing's TE performance under different number of slices by looking at link utilizations at different median and tail percentiles, using the gravity+spikes TM, under a 64-block topology. Figure 6 shows that virtual slicing's link utilization on different percentiles improves over physical sharding, where the level of improvement increases along with the increase in virtual slices/physical shards. When 8 slices/shards are used, virtual slicing improves P99/max link utilization by 13.5%/11.6%. Note that virtual slicing improves not only MLU—the optimization objective of the TE algorithm, but also the utilization of links at other percentiles, particularly at the tail. Though these percentiles were not the target of optimization, virtual slicing's elimination of asymmetry and capacity fragmentation effectively increases the usable capacity within the network, and consequently, improves overall utilization. These results confirms our expectations, as the increase in number of slices exacerbates topology asymmetry, which then impacts TE performance (Section III).

Different traffic patterns. Figure 7 shows virtual slicing's link utilization improvement for the same 64-block topology over different types of traffic. Virtual slicing improves tail link utilization for all types of traffic, with a 13.6%/9.7% P99/max link utilization improvement under the gravity+spikes TM. Tail link utilization improvement is especially noticeable under the gravity+spikes TM, likely due to physical sharding results in sub-topologies with transit capacity loss (Section III), thus having less capacity to accommodate traffic bursts.

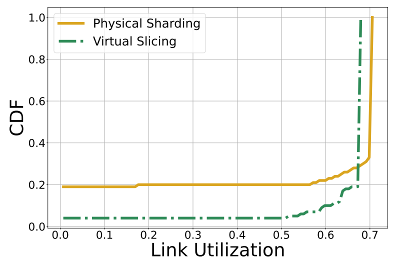
These improvements collectively indicate that virtual slicing load balances traffic more efficiently than physical sharding, due to virtual slicing taking global traffic as input and computing a globally optimal TE solution, as well as the elimination of asymmetry and capacity fragmentation (Section III). Improving overall and tail link utilization distribution in DCNs is crucial to fabric efficiency and application-level performance, as studies have shown that higher utilization worsens packet loss rate and application latency [24], [38].

C. *Production topologies and traffic*

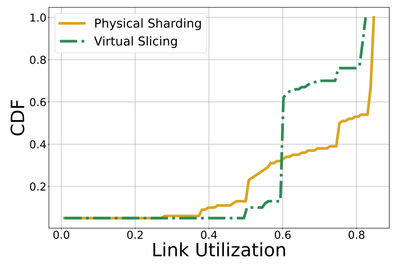
We now study how virtual slicing works with realistic spine-free topologies and traffic. For this part of the evaluation, we generated several topologies of various sizes with our production topology builder for realisticness. We also arbitrarily chose, from our fleet, a deployed medium-size and a large-size example fabric for analysis.

Fabric scale. We generated spine-free topologies with 48, 64 and 72 aggregation blocks with the production topology builder to understand the impact of fabric scale on virtual slicing's performance. Figure 8 shows the link utilization across synthetic topologies under the gravity+spikes TM. Virtual slicing improves tail link utilization, and greatly reduces overall link load in the network. For instance, the 70th percentile link utilization is improved by 3.8% (Figure 8a), 11.8% (Figure 8b), and 25.4% (Figure 8c) for each respective fabric. The average utilization improvement for percentiles from the 50th to the 100th is 3.8%, 16.8% and 13.9% in these fabrics. The TE performance gains of virtual slicing are much more prominent as the fabric size increases (Figures 8b and 8c) because sub-topologies across shards are more asymmetric as fabric size increases in physical sharding. In Figure 8, it can be observed that virtual slicing exhibits higher utilization in the lower percentiles. This is expected, as the objective of traffic engineering is to minimize maximum link utilization by distributing traffic loads more uniformly across links. As discussed in Section III, virtual slicing eliminates fragmentation loss and generates globally optimal traffic engineering decisions, thereby redistributing traffic from more congested links to less congested ones, resulting in a more balanced utilization.

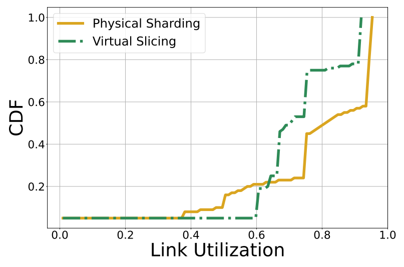
Deployed topologies. Figure 9 demonstrates each deployed production fabric's link utilization of different percentiles with 2 hours of realistic traffic traces. Traffic is collected and aggre-



(a) Fabric 1 (48 Blocks)

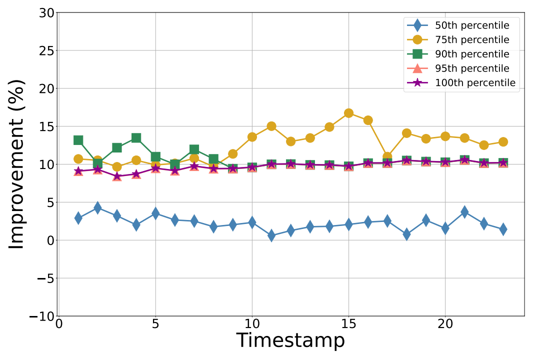


(b) Fabric 2 (64 Blocks)

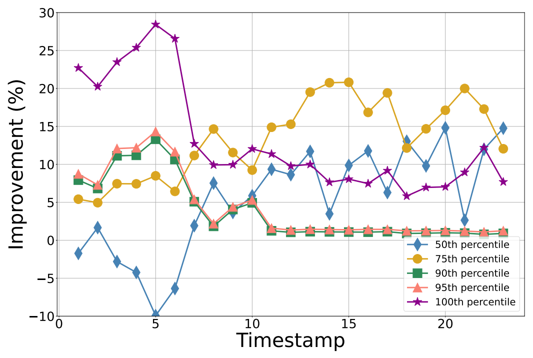


(c) Fabric 3 (72 Blocks)

Figure 8: TE performance improvement as fabric scales.

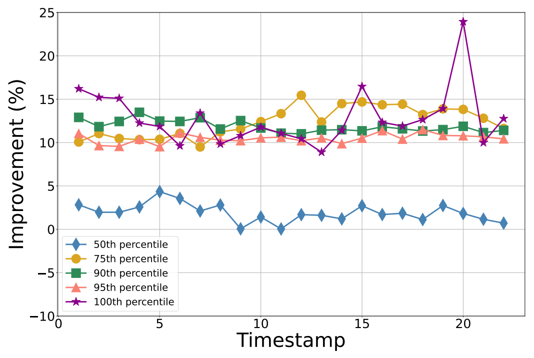


(a) Medium production fabric

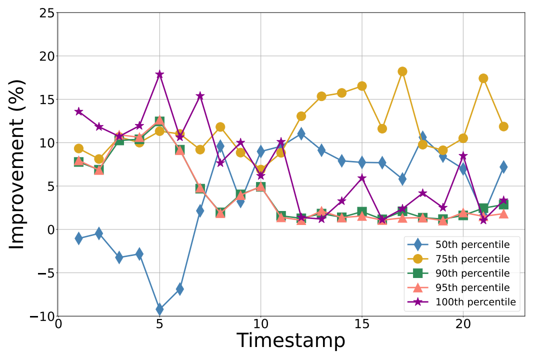


(b) Large production fabric

Figure 9: Real-time TE improvement.



(a) Medium production fabric



(b) Large production fabric

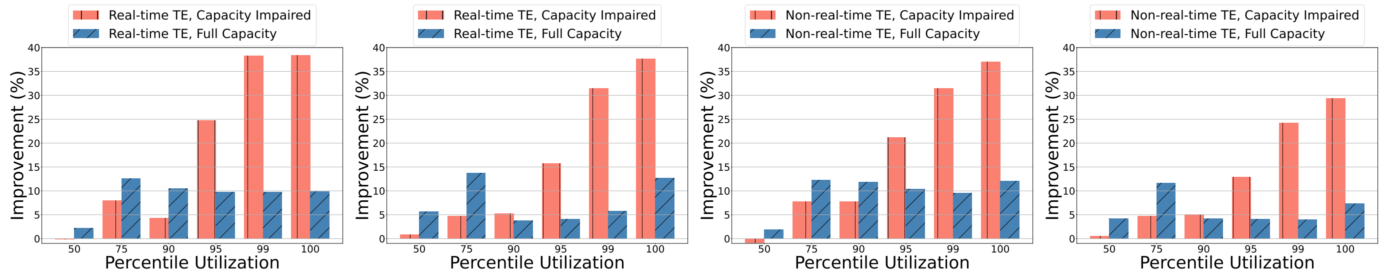
Figure 10: TE improvement under delayed measurements.

gated every 5 minutes[2]. Similar to the results with synthetic traffic, we observe improvements in link utilization across different percentiles, as well as notable tail improvement across all fabrics: the medium fabric has an average of 12.5%, and up to 16.8% link utilization improvement for the 75th percentile; as well as an average of 9.7%, and up to 10.6% max link utilization improvement. The large fabric's corresponding performance improvement numbers are even better (up to 20.9% link utilization improvement for the 75th percentile; and up to 28.4% link utilization improvement for MLU). These results collectively show that the TE improvements of virtual slicing hold, especially at the tail, for realistic topologies and traces as well, where traffic is more bursty, unpredictable and skewed. [3]

TE performance with traffic measurement delays. All previous experiments studied a *real-time* TE setting, where traffic input to the TE solver was identical to the input used for performance evaluation. Real-time TE reflects the optimal TE

[2]Note that these traces were collected from arbitrary chosen time intervals and fabrics, therefore our experiments do not imply a generalized result.

[3]Note that since the objective of our TE system is to minimize maximum link utilization (Section V-A), it is expected that lower-percentile link utilizations, which are not the target of optimization, sometimes do not improve.



(a) Real-time, med. fabric (b) Real-time, large fabric (c) Delayed, med. fabric (d) Delayed, large fabric

Figure 11: TE improvement of virtual slicing under capacity impairments.

solution where traffic measurement delay is omitted, allowing us to better understand the fundamental performance difference between virtual slicing and physical sharding. However, even if the traffic matrix can be measured in real-time, it would be difficult to program the TE solution in real-time, especially in large-scale networks. Therefore, we also study the performance of virtual slicing for *non-real-time TE*, where the traffic input to TE solver is based on *delayed* traffic data. We use the traffic matrix from the past 5 minutes to compute the TE solution, which is then evaluated based on the current traffic matrix.

Figure 10 shows the link utilization percentile improvements of virtual slicing under non-real-time TE. Similar to real-time TE, all fabrics have constant improvement on high percentiles. In particular, both the medium and large fabrics have an averaged 10% improvement for all percentiles 75th and higher. The two fabrics show a MLU improvement of 23.9% and 17.8% across timestamps, respectively. These results show that virtual slicing retains its TE benefits even when facing realistic delayed traffic measurements.

TE performance with capacity impairments. To study virtual slicing's performance under capacity impairments due to scheduled maintenance operations (e.g., switch drains) or unexpected failures (e.g., rack failures), we introduce a scenario where 25% of a randomly chosen aggregation block's links are taken offline. We analyze the TE improvement results under both real-time TE and delayed TE setups.

Figure 11a and Figure 11b demonstrate that with virtual slicing, under real-time TE, both fabrics show additional improvement over physical sharding under large capacity loss. In particular, compared with physical sharding, virtual slicing offers 38.4.% and 37.6% MLU improvement, as well as 38.8% and 31.4% improvement at the 99th percentile link utilization respectively for the two production fabrics. Similarly, in the delayed TE scenario shown in Figure 11c and Figure 11d, virtual slicing improves tail link utilization significantly: the medium and large fabrics respectively show 37.0% and 29.4% MLU improvement. The gains confirm our hypothesis described in Section III—capacity loss, whether caused by scheduled maintenance or unexpected failures, worsens topology asymmetry, and thus leads to exacerbated transit path capacity loss, which is the key reason behind physical sharding's sub-optimality. On the other hand, virtual slicing avoids such problems by design and thus provides much better TE performance.

D. *Impact on flow completion time*

Previous subsections demonstrated the TE (link utilization) improvement of virtual slicing under various scenarios. We now study virtual slicing's impact on flow completion time (FCT) under different scenarios. FCT represents the end-to-end transfer latency of a network flow, which has direct impact on application performance and user experience, especially for time-sensitive distributed applications. We constructed our flow-level traces by sampling flow size CDFs reported by different production networks [36], [39], while ensuring the total size of the sampled flows matches the demand of the traffic matrix. We used the gravity+spikes TM (Section V-A) and a 32-block topology for this study. Run-time load balancing is done at flow-level. TE solutions are programmed following the routing design in Section IV-B, where WCMP weights are implemented via entry replication in group tables.

Figure 12 shows the flow completion time at different percentiles for both virtual slicing and physical sharding. Virtual slicing improves P95 FCT by 62.7%, P99 FCT by 45.3%, and max (P100) FCT by 36.8%. Again, the improvements in tail FCTs can be attributed to globally optimal TE and the increase of usable capacity in virtual slicing, which allows flows to allocate more bandwidth and thus reducing their time to completion. Compared to the link utilization studies in Section V-B and Section V-C, we observe more substantial improvements in tail FCT performance with virtual slicing. We attribute this enhancement to challenges associated with *imperfect load balancing*. To elaborate, load balancing operates at the flow level; therefore, the actual load distributed across each path equates to the cumulative sizes of flows routed through that path, which can diverge from the intended traffic split calculated by the TE system. Such discrepancy intensifies the issue of suboptimal traffic engineering in physical sharding caused by unequal split of traffic and topology (Section III-B). This, in turn, leads to further FCT degradation in physical sharding.

E. *TE solving time microbenchmarks*

TE is solved by linear programming for both virtual slicing and physical sharding, where the solving time is primarily affected by topology size. Although the topology input to

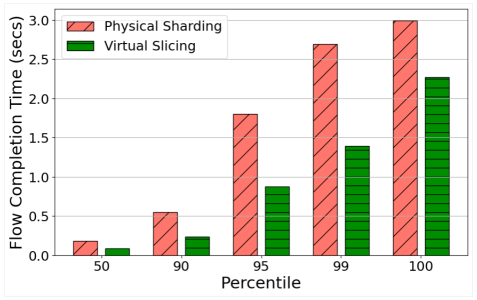


Figure 12: Flow completion times.

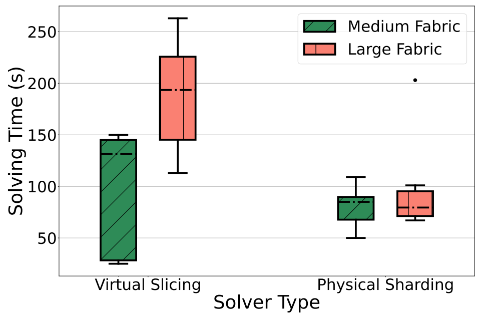


Figure 13: TE solving time benchmark.

the solver has the same number of nodes (i.e., aggregation blocks) in both physical sharding and virtual slicing, the input to physical sharding has less edges (i.e., block-level paths), especially when the topology increases in size, since each physical shard's topology gets sparser with scale. Therefore we expect virtual slicing's TE solving time to be larger than physical sharding, as fabrics scale. To study this, we benchmarked TE solver time for both virtual slicing and physical sharding on a medium and large production fabric. As shown in Figure 13, the median TE solving time is 85/131 seconds for the medium fabric, and 79/193 seconds for the large fabric using physical sharding and virtual slicing, respectively. Since TE solutions in deployed DCNs are reprogrammed at O(10s of minutes) timescale, the TE solution is reprogrammed at a timescale much longer than the solving time, staying within acceptable bounds.

VI. Related Work

Spine-free DCNs Several works [12], [40], [41] have been proposed to directly connect ToRs as a spine-free network to overcome the shortcomings of Clos-based hierarchical networks. However, expander-based solutions face deployment challenges. For instance, Jellyfish [40] and Xpander [41] require MPTCP [42] over shortest-K paths, while Kassing et al. [43] suggest using ECMP-VLB hybrid routing with flowlet switching, which is only available on select hardware platforms. DRing [12] proposes a more practical routing design but it only scales to small to medium-sized data centers. Alternatively, other works [13], [14], [44], [45] suggest connecting ToRs to an optical circuit switching (OCS) layer, leveraging the high reconfigurability and bandwidth of OCSes to provide a reconfigurable flat network. Recently, a large-scale cloud provider has started exploring and adapting a SDN-managed spine-free DCN architecture that directly connects aggregation blocks via MEMS-based OCSes [8], [24], [46]. Our paper focuses on the control plane sharding design for SDN-managed spine-free DCNs where aggregation blocks are connected by an OCS interconnect layer.

TE in WANs and DCNs TE is a well-established technique for enhancing wide-area network (WAN) reliability and efficiency [22], [47]–[51]. Previous studies [4], [8], [24] have also demonstrated the advantages of implementing TE in data centers. For example, TE is able to accomodate more traffic in a spine-free DCN by carefully load balancing traffic demands on multiple paths. TE is often formulated as a multi-commodity flow (MCF) problem [34] and when fractional flow is allowed, MCF problems can be solved in polynomial time through linear programming (LP). In SDN-managed DCNs, TE solutions are first solved in a SDN controller, then the generated TE path splits are translated as WCMP weights and programmed on the switches. However, to the extent of our knowledge, no prior work has systematically investigated the interactions between TE and SDN control plane partitioning in DCNs. Virtual slicing is one of the first attempts to provide a highly available control plane and optimized TE solutions simultaneously.

Network virtualization FlowVisor [52] envisioned an OpenFlow [53]-based network virtualization layer to slice physical networks via partitioning flow tables. Virtual routing and forwarding (VRF) [54] is a technology in routers that allows multiple instances of a routing table to coexist in a router and work simultaneously. Virtual slicing shares a similar vision and shards a network via slicing switch resources (both flow and group tables) as opposed to partitioning physical links. Different from prior works, virtual slicing is not only able to partition a network but also able to provide dynamic traffic migration capability which is essential for control plane failure handling. Furthermore, we demonstrate the TE performance gains of virtual slicing compared with state-of-the-art control plane partitioning scheme, physical sharding, in spine-free DCNs using production topologies and traffic traces.

VII. Conclusion

Spine-free DCNs have received increasing attention in both academia and industry in recent years. In this paper, we show that the state-of-the-art control plane partitioning scheme, physical sharding, inherently creates mismatches with spine-free DCN topologies and impacts TE performance. We propose a novel control plane partitioning approach, *virtual slicing*, that contains impact of faulty contollers while ensuring efficient TE performance. We also propose a concrete routing design for virtual slicing using standard techniques and analyze its feasibility. Experiments using realistic topologies and traffic demonstrate that virtual slicing significantly improves TE performance in spine-free DCNs, especially under network failures, while containing the impact of control plane failures.

References

[1] N. Katta, M. Hira, C. Kim, A. Sivaraman, and J. Rexford, “Hula: Scalable load balancing using programmable data planes,” in *Proceedings of the Symposium on SDN Research*, 2016.

[2] M. Alizadeh, T. Edsall, S. Dharmapurikar, R. Vaidyanathan, K. Chu, A. Fingerhut, V. T. Lam, F. Matus, R. Pan, N. Yadav, and G. Varghese, “Conga: distributed congestion-aware load balancing for datacenters,” in *SIGCOMM*, 2014.

[3] E. Vanini, R. Pan, M. Alizadeh, P. Taheri, and T. Edsall, “Let it flow: Resilient asymmetric load balancing with flowlet switching,” in *NSDI*, 2017.

[4] T. Benson, A. Anand, A. Akella, and M. Zhang, “Microte: Fine grained traffic engineering for data centers,” in *CoNEXT*, 2011.

[5] S. S. Chen, K. He, R. Wang, S. Seshan, and P. Steenkiste, “Precise data center traffic engineering with constrained hardware resources,” in *NSDI*, 2024.

[6] M. Denis, Y. Yao, A. Hatch, Q. Zhang, C. L. Lim, S. Zhang, K. Sugrue, H. Kwok, M. J. Fernandez, P. Lapukhov, S. Hebbani, G. Nagarajan, O. Baldonado, L. Gao, and Y. Zhang, “Ebb: Reliable and evolvable express backbone network in meta,” in *SIGCOMM*, 2023.

[7] U. Krishnaswamy, R. Singh, N. Bjørner, and H. Raj, “Decentralized cloud wide-area network traffic engineering with blastshield,” in *NSDI*, 2022.

[8] L. Poutievski, O. Mashayekhi, J. Ong, A. Singh, M. Tariq, R. Wang, J. Zhang, V. Beauregard, P. Conner, S. Gribble *et al.*, “Jupiter evolving: transforming Google’s datacenter network via optical circuit switches and software-defined networking,” in *SIGCOMM*, 2022.

[9] C.-Y. Hong, S. Mandal, M. Al-Fares, M. Zhu, R. Alimi, C. Bhagat, S. Jain, J. Kaimal, S. Liang, K. Mendelev *et al.*, “B4 and after: managing hierarchy, partitioning, and asymmetry for availability and scale in google’s software-defined WAN,” in *SIGCOMM*, 2018.

[10] A. D. Ferguson, S. Gribble, C.-Y. Hong, C. Killian, W. Mohsin, H. Muehe, J. Ong, L. Poutievski, A. Singh, L. Vicisano, R. Alimi, S. S. Chen, M. Conley, S. Mandal, K. Nagaraj, K. N. Bollineni, A. Sabaa, S. Zhang, M. Zhu, and A. Vahdat, “Orion: Google’s software-defined networking control plane,” in *NSDI*, 2021.

[11] P. Berde, M. Gerola, J. Hart, Y. Higuchi, M. Kobayashi, T. Koide, B. Lantz, B. O’Connor, P. Radoslavov, W. Snow, and G. Parulkar, “Onos: towards an open, distributed sdn os,” in *HotSDN*, 2014.

[12] V. Harsh, S. A. Jyothi, and P. B. Godfrey, “Spineless data centers,” in *HotNets*, 2020.

[13] H. Ballani, P. Costa, R. Behrendt, D. Cletheroe, I. Haller, K. Jozwik, F. Karinou, S. Lange, K. Shi, B. Thomsen *et al.*, “Sirius: A flat datacenter network with nanosecond optical switching,” in *SIGCOMM*, 2020.

[14] W. M. Mellette, R. McGuinness, A. Roy, A. Forencich, G. Papen, A. C. Snoeren, and G. Porter, “Rotornet: A scalable, low-complexity, optical datacenter network,” in *SIGCOMM*, 2017.

[15] W. M. Mellette, R. Das, Y. Guo, R. McGuinness, A. C. Snoeren, and G. Porter, “Expanding across time to deliver bandwidth efficiency and low latency,” in *NSDI*, 2020.

[16] M. Al-Fares, A. Loukissas, and A. Vahdat, “A scalable, commodity data center network architecture,” in *SIGCOMM*, 2008.

[17] A. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta, “Vl2: A scalable and flexible data center network,” in *SIGCOMM*, 2009.

[18] A. Singh, J. Ong, A. Agarwal, G. Anderson, A. Armistead, R. Bannon, S. Boving, G. Desai, B. Felderman, P. Germano *et al.*, “Jupiter rising: A decade of clos topologies and centralized control in google’s datacenter network,” in *SIGCOMM*, 2015.

[19] A. Andreyev, “Introducing data center fabric, the next-generation Facebook data center network.” https://code.facebook.com/posts/360346274145943, 2014.

[20] D. Applegate and E. Cohen, “Making intra-domain routing robust to changing and uncertain traffic demands: Understanding fundamental tradeoffs,” in *SIGCOMM*, 2003.

[21] M. Roughan, M. Thorup, and Y. Zhang, “Traffic engineering with estimated traffic matrices,” in *IMC*, 2003.

[22] S. Kandula, D. Katabi, B. Davie, and A. Charny, “Walking the tightrope: Responsive yet stable traffic engineering,” in *SIGCOMM*, 2005.

[23] H. Wang, H. Xie, L. Qiu, Y. R. Yang, Y. Zhang, and A. Greenberg, “Cope: Traffic engineering in dynamic networks,” in *SIGCOMM*, 2006.

[24] M. Zhang, J. Zhang, R. Wang, R. Govindan, J. C. Mogul, and A. Vahdat, “Gemini: Practical reconfigurable datacenter networks with topology and traffic engineering,” *arXiv preprint arXiv:2110.08374*, 2021.

[25] J. Zhou, M. Tewari, M. Zhu, A. Kabbani, L. Poutievski, A. Singh, and A. Vahdat, “WCMP: Weighted cost multipathing for improved fairness in data centers,” in *EuroSys*, 2014.

[26] S. Hogg, “High Expectations of Network Availability?” https://www.networkworld.com/article/2235262/cisco-subnet-high-expectations-of-network-availability.html, 2009.

[27] E. Voit, “Enterprise Network Availability: How to Calculate and Improve?” https://blogs.cisco.com/networking/enterprise-network-availability-how-to-calculate-and-improve, 2020.

[28] P. Gill, N. Jain, and N. Nagappan, “Understanding network failures in data centers: measurement, analysis, and implications,” in *SIGCOMM*, 2011.

[29] Cisco, “Cisco nexus 9000 series nx-os label switching configuration guide,” https://www.cisco.com/c/en/us/td/docs/switches/datacenter/nexus9000/sw/7-x/label-switching/configuration/guide/b_Cisco_Nexus_9000_Series_NX-OS_Label_Switching_Configuration_Guide_7x/b_Cisco_Nexus_9000_Series_NX-OS_MPLS_Configuration_Guide_7x_chapter_011.html#concept_FBC7A3F15CFE4765AAA5E4E9A6CF7F88, 2024.

[30] Broadcom, “Tomahawk3/BCM56980 Series,” https://www.broadcom.com/products/ethernet-connectivity/switching/strataxgs/bcm56980-series, 2019.

[31] ——, “Tomahawk4/BCM56990 Series,” https://www.broadcom.com/products/ethernet-connectivity/switching/strataxgs/bcm56990-series, 2019.

[32] Y. Xu, K. He, R. Wang, M. Yu, N. Duffield, H. Wassel, S. Zhang, L. Poutievski, J. Zhou, and A. Vahdat, “Hashing design in modern networks: Challenges and mitigation techniques,” in *ATC*, 2022.

[33] NS-3 Consortium, “ns-3 network simulator,” 2023. [Online]. Available: https://www.nsnam.org/

[34] S. Even, A. Itai, and A. Shamir, “On the complexity of time table and multi-commodity flow problems,” in *16th Annual Symposium on Foundations of Computer Science (SFCS 1975)*, 1975.

[35] Y. Zhang, M. Roughan, N. Duffield, and A. Greenberg, “Fast accurate computation of large-scale IP traffic matrices from link loads,” in *SIGMETRICS*, 2003.

[36] T. Benson, A. Akella, and D. A. Maltz, “Network traffic characteristics of data centers in the wild,” in *IMC*, 2010.

[37] T. Benson, A. Anand, A. Akella, and M. Zhang, “Understanding data center traffic characteristics,” *ACM SIGCOMM Computer Communication Review*, 2010.

[38] V. Arun and H. Balakrishnan, “Copa: Practical delay-based congestion control for the internet,” in *NSDI*, 2018.

[39] A. Roy, H. Zeng, J. Bagga, G. Porter, and A. C. Snoeren, “Inside the social network’s (datacenter) network,” in *SIGCOMM*, 2015.

[40] A. Singla, C.-Y. Hong, L. Popa, and P. B. Godfrey, “Jellyfish: Networking data centers randomly,” in *NSDI*, 2012.

[41] A. Valadarsky, G. Shahaf, M. Dinitz, and M. Schapira, “Xpander: Towards optimal-performance datacenters,” in *CoNEXT*, 2016.

[42] D. Wischik, C. Raiciu, A. Greenhalgh, and M. Handley, “Design, implementation and evaluation of congestion control for multipath tcp,” in *NSDI*, 2011.

[43] S. Kassing, A. Valadarsky, G. Shahaf, M. Schapira, and A. Singla, “Beyond fat-trees without antennae, mirrors, and disco-balls,” in *SIGCOMM*, 2017.

[44] N. Farrington, G. Porter, S. Radhakrishnan, H. H. Bazzaz, V. Subramanya, Y. Fainman, G. Papen, and A. Vahdat, “Helios: a hybrid electrical/optical switch architecture for modular data centers,” in *SIGCOMM*, 2010.

[45] W. M. Mellette, R. Das, Y. Guo, R. McGuinness, A. C. Snoeren, and G. Porter, “Expanding across time to deliver bandwidth efficiency and low latency,” in *NSDI*, 2020.

[46] H. Liu, R. Urata, K. Yasumura, X. Zhou, R. Bannon, J. Berger, P. Dashti, N. Jouppi, C. Lam, S. Li *et al.*, “Lightwave fabrics: At-scale optical circuit switching for datacenter and machine learning systems,” in *SIGCOMM*, 2023.

[47] A. Feldmann, A. Greenberg, C. Lund, N. Reingold, and J. Rexford, “Netscope: Traffic engineering for IP networks,” *IEEE Network*, 2000.

[48] B. Fortz and M. Thorup, “Internet traffic engineering by optimizing ospf weights,” in *INFOCOM*, 2000.

[49] R. Zhang-Shen and N. McKeown, “Designing a fault-tolerant network using valiant load-balancing,” in *INFOCOM*, 2008.

[50] C.-Y. Hong, S. Kandula, R. Mahajan, M. Zhang, V. Gill, M. Nanduri, and R. Wattenhofer, "Achieving high utilization with software-driven WAN," in *SIGCOMM*, 2013.

[51] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu *et al.*, "B4: Experience with a globally-deployed software defined WAN," in *SIGCOMM*, 2013.

[52] R. Sherwood, G. Gibb, K.-K. Yap, G. Appenzeller, M. Casado, N. McKeown, and G. M. Parulkar, "Can the production network be the testbed?" in *OSDI*, 2010.

[53] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: enabling innovation in campus networks," *ACM SIGCOMM computer communication review*, 2008.

[54] Cisco, "Virtual Route Forwarding Design Guide," https://www.cisco.com/c/en/us/td/docs/voice_ip_comm/cucme/vrf/design/guide/vrfDesignGuide.html, 2008.