OpticNet: Self-Adjusting Networks for ToR-Matching-ToR Optical Switching Architectures

Caio A. Caldeira, Otavio A. de O. Souza and Olga Goussevskaia
Universidade Federal de Minas Gerais (UFMG), Brazil
{caio.caldeira, oaugusto, olga}@dcc.ufmg.br

Stefan Schmid
TU Berlin & Fraunhofer SIT, Germany
stefan.schmid@tu-berlin.de

***Abstract*—Demand-aware reconfigurable datacenter networks can be modeled as a ToR-Matching-ToR (TMT) two-layer architecture, in which each top-of-rack (ToR) is represented by a static switch, and n ToRs are connected by a set of reconfigurable optical circuit switches (OCS). Each OCS internally connects a set of in-out ports via a matching that may be updated at runtime. The matching model is a formalization of such networks, where the datacenter topology is defined by the union of matchings over the set of nodes, each of which can be reconfigured at unit cost.**

In this work we propose a scalable matching model for scenarios where OCS have a constant number of ports. Furthermore, we present OpticNet, a framework that maps a set of n static ToR switches to a set of p-port OCS to form any constant-degree topology. We prove that OpticNet uses a minimal number of reconfigurable switches to realize any desired network topology and allows to apply any existing self-adjusting network (SAN) algorithm on top of it, also preserving amortized performance guarantees. Our experimental results based on real workloads show that OpticNet is a flexible and efficient framework to design efficient SANs.

***Index Terms*—Reconfigurable Datacenter Networks, Self-Adjusting Networks, Optical Circuit Switching, Matching Model**

I. Introduction

The design of efficient datacenter networks has received increasing attention in recent years [1]–[9], fueled by data-centric online applications, such as web search, social networks, and multimedia. Augmenting the internal switching capacity of the datacenter networks according to traffic growth has become increasingly cost prohibitive [8]. Traditionally, datacenter network designs rely on static topologies, such as the Clos topology [10], [11], hypercubic topologies like BCube and MDCube [12], [13], or expander-based networks [14], [15]. Alternatively, rotor switches [8], [9] provide periodic direct connectivity. While such architectures perform well for all-to-all traffic patterns, they essentially form demand-oblivious topologies.[1]

Empirical studies show that communication patterns in datacenters feature much spatial and temporal locality [2], [3], [16], i.e., traffic is bursty and traffic matrices skewed. This structure represents an untapped potential for building more efficient communication networks This is the advantage of demand-aware topologies, based on e.g. 3D MEMS optical circuit switches (OCS) [1]–[5], [5]–[7], [17]–[19], which can provide shortcuts to such elephant flows. OCS provide reconfigurations in order of milliseconds and are reconfigurable on-demand, such that a scheduling or matching algorithms can determine the next configuration based on the network state. Several prototypes based on commodity-off-the-shelf OCS have been built, and their advantages have been demonstrated, e.g., [5], [19].

Self-Adjusting Networks (SAN) are networks which optimize their physical topology toward the demand they serve in an online manner, i.e., without prior knowledge of the traffic demand. SAN can be approached from the perspective of self-adjusting data structures [2]. This paradigm shift resembles the process that data structures went through some decades ago [20], [21], evolving from static worst-case designs toward demand-aware and self-adjusting data structures.

When analyzing the performance of SANs, it is common to distinguish between the so-called *adjustment cost* and *service cost*. The former refers to the cost of network reconfiguration (e.g., energy, latency and control plane overhead due to physical network topology adjustments), and the latter refers to the price of serving each communication request (e.g. the delay proportional to source-destination route length). Most of the existing SAN algorithms [2] have been based on *edge-distance* adjustment cost models, which define adjustment cost as the number of edges replaced between consecutive network topologies. This is a useful basic model that enabled the first algorithmic results. In practice, however, switching hardware usually allows to reconfigure the topology on a so-called *per-matching granularity*, i.e., the adjustment cost is proportional to the number of OCS reconfigurations, where each switch internally connects its in-out ports via a *matching*.

ToR-Matching-ToR (TMT) [22] is a two-layer leaf-spine architecture for datacenter networks, in which each top-of-rack (ToR) is represented by a (leaf) static packet switch, and n ToRs are connected by a set of (spine) reconfigurable OCSs. Each spine switch internally connects n in-out ports via a *matching* that may be reconfigured at runtime. TMT can be used to model existing systems, such as RotorNet [9], Opera [8], and ProjecToR [3].

The matching model (MM) is a first formalization of TMT networks [22]. The key property of MM is assuming that any topology can be defined as a union of matchings over the set of nodes and that rearranging the edges of a single matching comes at a fixed cost. Thus the total *adjustment cost* for

[1]Our research work was supported by CAPES, CNPq, Fapemig, and the European Research Council (ERC), grant agreement No. 864228.

adjusting the whole topology to a new one is determined by the number of matchings needed to construct the topology. Early work has shown this model's relevance, e.g. in [23] the authors simulate several so-called lazy SANs in the MM, based on spaced out (less frequent) calls of existing (sequential) SANs, such as SplayNets [4] and ReNets [24].

In this work, we propose a scalable matching model (MM'), which generalizes the MM by adding the constraint that the number of input-output ports of each reconfigurable spine switch is constant, while relaxing the constraint that the number of spine switches is constant. MM' adds flexibility to the network design, depending on the cost and constraints on available switching hardware. MM' uses larger leave-layer (static) switches, which typically come at a lower per-port hardware price than reconfigurable OCS hardware [25], and a greater number of smaller spine-layer switches, which can be incrementally appended to the existing architecture (even at runtime) in case new ToRs are added to the datacenter.

Summary of contributions: In Section II we describe the scalable matching model (MM'), a generalization of the matching model [22], that allows to model TMT architectures in which the number of ToRs (arbitrarily) exceeds the number of in-out ports of the spine-layer switches. In Section III we present OpticNet, a framework that maps a set of (leave-layer) ToR switches to a set of (spine-layer) demand-aware reconfigurable OCSs to form any constant-degree network topology. In Section IV we prove that OpticNet is correct, is optimal with respect to the number of spine switches, and any existing SAN algorithm can run on top of it, while preserving amortized performance guarantees in the MM' model. In Section V we describe the implementation of OpticNet and evaluate its performance in combination with state-of-the-art SAN algorithms. Finally, in Section VI we discuss related work, and in Section VII present our conclusions.

II. Model

SANs: The objective of SAN algorithms is to create a network topology to connect a set V of n communication nodes (e.g., top-of-rack switches, ToRs). The input to the problem is given by a sequence σ of m messages $\sigma_i(s_i, d_i) \in V \times V$ occurring over time, with source s_i and destination d_i; m can be infinite. We denote by b_i the time when a message σ_i is generated, and by e_i the time in which it is delivered. The sequence σ is revealed over time, in an online manner, and can be arbitrary (e.g. chosen adversarially). When serving these communication requests, the network can adjust over time, through a sequence of network topologies $N_0, N_1, \ldots$. Each N_t should belong to some desired graph family $\mathcal{N}$. For scalability reasons, the networks should be of *constant degree*.

Adjustments: In order to minimize the communication cost and adjust the topology smoothly over time, the network is reconfigured (locally) through adjustments that preserve the desired properties of the graph family $\mathcal{N}$. SANs like SplayNet [4] and DiSplayNet [26] are based on Binary Search Trees (BST) and extend the classical *zig, zig-zig* and *zig-zag* rotations, first introduced for splay trees [20]. CBTrees [27] and CBNet [28] leverage *bottom-up* and *top-down* semi-splaying (*semi-zigzig* and *semi-zigzag*). Each splay operation updates a constant number of links at constant cost, and roughly halves the depth of every node along the communication path. This halving effect makes splaying efficient in an amortized sense.

TMT: TMT [22] is a useful architecture to model RDN. It is a two-layer network, in which a set of *static* leaf-layer (ToR) switches is connected using a set of *reconfigurable* spine-layer switches (OCS). Each OCS internally connects its in-out ports via a *matching*, that can be dynamic and change over time.

MM: The matching model [22] is a formalization of the TMT architecture. The network consists of a set of n nodes (leaf layer static switches) are connected using $k' = O(1)$ (spine-layer reconfigurable) switches, $SW = \{sw_1, sw_2, \ldots, sw_{k'}\}$, and each switch internally connects its n *in-out ports* via a *matching*. At each time t, the network is the union of these matchings: $N_t = \mathcal{M}_t = \cup_{i=1}^{k'} M(i,t)$, where $M(i,t)$ denotes the matching on switch sw_i at time t.

Scalable matching model (MM'): In this work, we generalize the MM by adding the constraint that the number of ports of each spine-layer switch is constant, while relaxing the constraint that the number of spine switches is constant. Let $\mathcal{N}$ be the family of constant degree ($\leq k$) graphs on a set V of n nodes (static leaf-layer ToR switches) and $\mathcal{N}_t \in \mathcal{N}$ be a network topology at time t. Let $SW = \{sw_1, \ldots, sw_{n'}\}$ be a set of (reconfigurable spine-layer) switches, of size $n' = f(n, p, k)$, where each switch $sw_i \in SW$ internally connects its $p = O(1)$ in-out ports via a *matching* $M(i,t)$ of size $\leq p$. As in MM, at each time t, the network N_t is equal to the union of all these matchings:

$$N_t = \mathcal{M}_t = \cup_{i=1}^{n'} M(i,t).$$

TMT port configuration: The port-to-port physical connections between leaf and spine layers in a TMT architecture can be realized in different ways, according to the hardware specification (e.g., via free-space optics [3]). In this work, we assume that leaf-spine connections are static *full-duplex* links[2], i.e., for two ToRs (leaf-layer switches) $(u, v) \in N_t$ to establish a connection, we need that $\exists sw_x \in SW | (u, v) \in M(x, t)$. Moreover, there must be a static (physical) link connecting an output port of u to an input port of sw_x and a static link from an output port of sw_x to an input port of v.

Before the system (RDN) starts operating, the TMT architecture must be configured according to these requirements, i.e., we need to find a one-to-one assignment, each representing a full-duplex link, between all in-out and out-in ports of leaf-spine switch pairs.

In the MM, the port mapping between leaf-layer and spine-layer switches is straightforward: each ToR $i, 1 \leq i \leq n$ has k' uplinks, where uplink $j, 1 \leq j \leq k'$ connects to port i in sw_j. The directed outgoing (leaf) uplink is connected to the incoming port of the (spine) switch and the directed incoming (leaf) uplink is connected to the outgoing port of the (spine)

[2]Note that a full-duplex configuration can be easily converted to half-duplex by duplicating each link (switch) of the full-duplex configuration.

switch. Each switch has n input ports and n output ports and the connections are directed, from input to output ports. At any point in time, each switch $sw_j \in SW$ provides a matching of size $\leq n$ between its input and output ports.

In MM', however, because spine-layer switches are limited to a constant number of in-out ports, the problem of connecting leaf-layer and spine-layer switches has a different structure.

MM' port mapping problem (MM'-PMP): Let $\mathcal{N}$ be the family of constant degree ($\leq k$) graphs on a set V of n nodes (static leaf-layer ToR switches), and $SW = \{sw_1, \ldots, sw_{n'}\}$ be a set of $n' = f(n, p, k)$ (reconfigurable spine-layer) switches, where each switch $sw_i \in SW$ internally connects its in-out ports via a *matching* of size $\leq p$.

Let $\mathcal{P}(sw_x) \subseteq V \times V, |\mathcal{P}(sw_x)| = p^2$ be the (static) *in-out port-set* of a spine-layer switch $sw_x \in SW$.

Given a network topology $N_t \in \mathcal{N}$ at time t, the objective is to find a set of switches SW, a set of matchings $\mathcal{M}_t = \cup_{sw_x \in SW} M(x,t) = N_t$, and all in-out port sets $\mathcal{P}(sw_x), sw_x \in SW$, such that $\forall (u,v) \in N_t$:

$$\exists sw_x \in SW |\ (u,v) \in M(x,t) \text{ and } M(x,t) \in \mathcal{P}(sw_x).$$

Time model: In sequential SAN algorithms [4], [20], [23], each message σ_i is processed strictly after the previous one σ_{i-1} has been delivered. In this work we consider a concurrent execution scenario. We assume that time is divided into synchronous rounds, and each round consists of a constant number of time-slots. In a round, multiple (independent) nodes can make adjustments concurrently. We consider that nodes and communication between them are reliable.

Cost Model: Consider a sequence σ of m messages, a SAN algorithm $\mathcal{A}$, any initial network $N_0 \in \mathcal{N}$, and a message $\sigma_i(s_i, d_i) \in \sigma$, generated at time b_i and delivered at time e_i. In terms of time, one objective is to minimize the makespan:

$$Makespan(\mathcal{A}, N_0, \sigma) = \max_{1 \leq i \leq m} e_i - b_1.$$

Let us define the *adjustment cost* adj_i as the number of adjustments performed by $\mathcal{A}$ to deliver σ_i and the *service cost* srv_i to be the price of routing the message along the (shortest) path $dist_{N_{e_i}}(s_i, d_i), N_{e_i} \in \mathcal{N}$. We assume that routing a message incurs a cost of 1 unit per hop. MM allows to define the adjustment cost in different ways, depending on e.g. particular OCS hardware specification:

- **Edge distance**: The case where the adjustment cost is proportional to the number of replaced edges between each consecutive matchings of the same switch $sw_x \in SW$. If the cost of a single edge is α, then the adjustment cost for the entire network is defined as:

$$linkAdj(N_{t-1}, N_t) = \alpha \sum_{sw_x \in SW} |M(x,t) \setminus M(x,t-1)|.$$

- **Switch Cost**: In this case, if a matching (switch) is reconfigured (adjusted), it costs α regardless of the number of edge changes in the matching. Let $\mathbb{I}_{S=S'}$ be an indicator function that denotes if set S is equal to set S' . Then the adjustment cost for the network is:

$$swAdj(N_{t-1}, N_t) = \alpha \sum_{sw_x \in SW} \mathbb{I}_{M(x,t)=M(x,t-1)}.$$

Let $t_{\max} = \max_{1 \leq i \leq m} e_i$. Finally, we define the total service cost, total edge adjustment cost, **total switch adjustment cost**, and total work cost, respectively, as follows:

$$\begin{aligned}
\text{Srv}(\mathcal{A}, N_0, \sigma) &= \sum_{i=1}^{m} (dist_{N_{e_i}}(s_i, d_i) + 1), \\
\text{LinkAdj}(\mathcal{A}, N_0, \sigma) &= \alpha \sum_{t=1}^{t_{\max}} linkAdj(N_{t-1}, N_t), \\
\text{SwAdj}(\mathcal{A}, N_0, \sigma) &= \alpha \sum_{t=1}^{t_{\max}} swAdj(N_{t-1}, N_t), \\
\text{Cost}(\mathcal{A}, N_0, \sigma) &= \text{Srv}(\mathcal{A}, N_0, \sigma) + \text{Adj}(\mathcal{A}, N_0, \sigma).
\end{aligned}$$

III. OPTICNET

OpticNet is a framework for implementing SAN algorithms in the TMT network architecture. Firstly, we describe how OpticNet solves the MM'-PMP problem. Then, we describe how SAN adjustments are implemented in OpticNet.

Firstly, we partition the set of spine-layer switches SW into two subsets, by type: *unit switches* ($SW^{\cup}$) and *union switches* ($SW^{\times}$), as follows:

$$SW = SW^{\cup} \cup SW^{\times}.$$

For simplicity, we assume that $n\%p = 0$ (or, equivalently, that $1 \leq x < p$ non-active vertices are added to the network, so that n is a multiple of p). Next, we partition the set of leaf-layer switches (ToRs) V into $C = \left\lceil \frac{n}{p} \right\rceil$ clusters $C_i \subseteq V, |C_i| = p$, as follows: $\forall (u,v) \in V \times V$, if $u < v$ then $u \in C_i, v \in C_j$, where:

$$i = \min\left(\left\lfloor \frac{u}{p} \right\rfloor, \left\lfloor \frac{v}{p} \right\rfloor\right),\ j = \max\left(\left\lfloor \frac{u}{p} \right\rfloor, \left\lfloor \frac{v}{p} \right\rfloor\right).$$

Each cluster C_i will be assigned to a group of $\lceil k/2 \rceil$ switches in $SW^{\cup}$, and each pair of clusters $(C_i, C_j), i \neq j$ will be assigned to a group of k switches of type $SW^{\times}$. Finally, we define the in-out port-sets as follows:

$$\mathcal{P}(sw^{\cup}_{i \cdot \lceil k/2 \rceil + x}) = C_i \times C_i,\ \forall\ 0 \leq i < C, 0 \leq x < \lceil k/2 \rceil,$$
$$\mathcal{P}(sw^{\times}_{idx(i,j)+x}) = C_i \times C_j,\ \forall\ 0 \leq i < j < C, 0 \leq x < k,$$

where:

$$\begin{aligned}
idx(i,j) &= k \cdot \sum_{x=0}^{i-1} (C - x - 1) + k \cdot (j - i - 1) \\
&= \frac{k}{2} \cdot i \cdot (2 \cdot C - 1 - i) + k \cdot (j - i - 1)
\end{aligned}$$

The total number of spine-layer p-port switches to connect n leaf-layer switches in a constant-degree $\leq k$ network used by OpticNet is:

$$|SW| = |SW^{\cup}| + |SW^{\times}| = C \left\lceil \frac{k}{2} \right\rceil + k \binom{C}{2} = C^2 \left\lceil \frac{k}{2} \right\rceil.$$

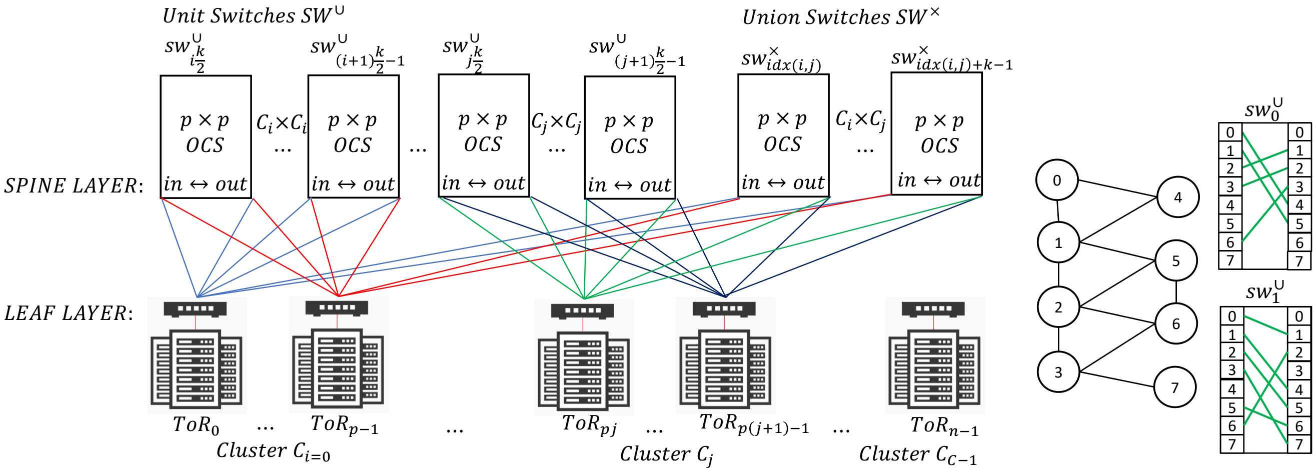

Fig. 1. (a) OpticNet TMT Architecture: Leaf Layer: constant-degree $\leq k$ network on n nodes; Spine Layer: set of unit and union-type p in-out port OCS switches $SW = SW^\cup \cup SW^\times, |SW^\cup| = \lceil k/2 \rceil \cdot C, |SW^\times| = k \cdot \binom{C}{2}, C = \lceil n/p \rceil$. (b) Unit switch example for a $k = 4$- degree network with $n = p = 8$.

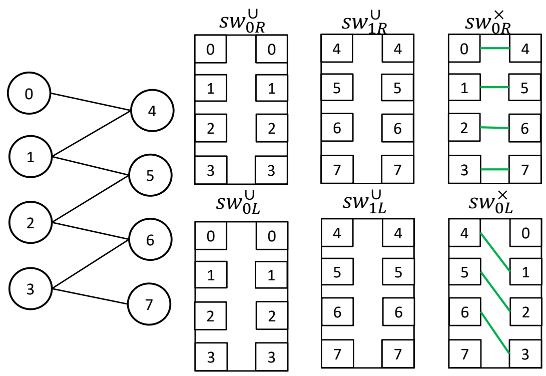

Fig. 2. Unit and union switches example for a BST with $n = 8, p = 4$.

In Figure 1a we illustrate how OpticNet implements a TMT architecture in a datacenter comprised of a k-degree network topology on n racks and a set of p-port spine-layer switches. A set of leaf-layer (ToR) switches is partitioned into C clusters of size p, and each $v \in C_i$ is connected to one input and one output port of $k/2$ unit switches, and to either one input or one output port of k union switches, for each cluster $C_j, j \neq i$. The number of ports of each ToR switch is $k + k(C-1) = k \cdot C$.

Binary Search Trees (BST): SAN algorithms based on splaying adjustments [4], [28] assume that N_t is a BST. In this case, we can reduce the number of spine switches, as follows: We need only 2 unit switches for each cluster C_i and 2 union switches for each pair of clusters $(C_i, C_j), i < j$, one for the *left*-child and one for the *right*-child edges in the BST, which results in $|SW|_{BST} = 2C + 2\binom{C}{2} = C(C+1)$.

In Figure 1b we illustrate an example of two $p = 8$-port unit switches connecting a $k = 4$-degree network of $n = 8$ nodes. In Figure 2 we illustrate a BST network on $n = 8$ nodes, connected via 4 unit and 2 union $p = 4$-port switches.

Network adjustments: We assume that, at each round t, OpticNet receives a set of edges $E = N_{t+1} \setminus N_t$ to be updated (removed or added) between two consecutive network topologies $N_t \in \mathcal{N}$ and $N_{t+1} \in \mathcal{N}$ from a SAN algorithm $\mathcal{A}$. For each edge $(u, v) \in E$: if $(u, v) \notin N_{t+1}$, then OpticNet removes it from the matching $M(i, t+1) \in \mathcal{M}_{t+1}$ on switch $sw_i | (u, v) \in M(i, t)$; and if $(u, v) \notin N_{t+1}$, OpticNet adds it to some matching $M(x, t+1) \in \mathcal{M}_{t+1}$.

Algorithm 1 describes the procedure of adding a new edge $(u, v) \notin N_t$ into $\mathcal{M}_{t+1}$. Firstly, from the subset of $k/2$ unit or k union switches $S_u^v = \{sw_z \in SW | \mathcal{P}(sw_z) = C_i \times C_j, u \in C_i, v \in C_j\}$ (lines $3-6$), two switches are selected: one, where an input port u (or v) is free (sw_x) and one in which the output port v (or u) is free (sw_y) (lines $7-10$). Note that it is possible that $sw_x = sw_y$, but at least one sw_x and one sw_y must exist. Otherwise the addition of (u, v) would imply degree $> k$ for u or v in N_{t+1}. The edge (u, v) (or (v, u)) is added to the matching $M(x, t+1)$ (line 21). Finally, a sequence of edge exchanges is performed between $M(x, t+1)$ and $M(y, t+1)$ (in the case $(sw_x = sw_y)$, the exchange operation is replaced by an edge reversal, i.e., $(in, out) \leftarrow (out, in)$) (lines $22-32$). An exchange (or reversal) is performed every times there is a conflict at one of the two ports connected by the new edge. Note that, by the choice of sw_x and sw_y, at most one (input or output) port might be already occupied at either $M(x, t)$ or $M(y, t)$, and there might be at most $2(p-1)$ edge exchanges on at most 2 switches.

A. OpticNet framework implementation

We implemented OpticNet as a simulation framework for SAN algorithms in the context of TMT architectures, on top of which an interested party can easily add or extend the decision making and network adjustments specification of their SAN algorithm. Like some of the earlier work on concurrent SANs [26], [29], we adopt an optimistic approach to solve concurrency conflicts. Messages that have been in the network for longer are prioritized to avoid deadlocks and starvation.

Centralized control node: All communication between the leaf and the spine layers is implemented via a *centralized*

Algorithm 1 OpticNet adjustments: new edge insertion

Require: $\mathcal{M}_t$, $SW^{\cup}$, $SW^{\times}$, k, $(u,v) \notin N_t, u \in C_i, v \in C_j$.

$\mathcal{M}_{t+1} \leftarrow \mathcal{M}_t$;
$(in, out) \leftarrow (u, v)$; $reverse \leftarrow false$;
if $i = j$ **then** %(unit switch)
  $S_u^v = \{sw_z \in SW^{\cup} |\ i \cdot \lceil \frac{k}{2} \rceil \leq z < i \cdot \lceil \frac{k}{2} \rceil + \lceil \frac{k}{2} \rceil\}$;
else %(union switch)
  $S_u^v = \{sw_z \in SW^{\times} |\ idx(i,j) \leq z < idx(i,j) + k\}$;
$sw_x = sw_z \in S_u^v | (u,w) \notin M(z,t), \forall w \in C_j$;
$sw_{x'} = sw_z \in S_u^v | (w,u) \notin M(z,t), \forall w \in C_i$;
$sw_y = sw_z \in S_u^v | (w,v) \notin M(z,t), \forall w \in C_i$;
$sw_{y'} = sw_z \in S_u^v | (v,w) \notin M(z,t), \forall w \in C_i$;
if $sw_x = \emptyset$ and $sw_y = \emptyset$ **then** %(unit-sw, free out-in)
  $(in, out) \leftarrow (out, in); x \leftarrow y'; y \leftarrow x'$;
if $sw_x = \emptyset$ and $sw_y \neq \emptyset$ **then** %(unit-sw, free out-out)
  $x \leftarrow x'$; $M(x,t) \leftarrow \{(w',w) |\ (w,w') \in M(x,t)\}$;
  if $x = y$ **then** $reverse \leftarrow true$;
if $sw_x \neq \emptyset$ and $sw_y = \emptyset$ **then** %(unit-sw, free in-in)
  $y \leftarrow y'$;
  if $x \neq y$ **then**
    $M(y,t) \leftarrow \{(w',w) |\ (w,w') \in M(y,t)\}$;
  else $reverse \leftarrow true$;
add (in, out) to $M(x, t+1) \in \mathcal{M}_{t+1}$;
$in \leftarrow w | (w, out) \in M(x,t) \in \mathcal{M}_t$;
while $(in, out) \neq \emptyset$ **do**
  if $reverse$ **then** %unit sw, free in-in, $(x = y)$
    reverse $(in, out) \in M(x, t+1)$;
    $out \leftarrow in$; $in \leftarrow w | (w, out) \in M(x,t)$;
  else
    move (in, out) from $M(x, t+1)$ to $M(y, t+1)$;
    $out \leftarrow w | (in, w) \in M(y,t)$;
    if $(in, out) \neq \emptyset$ **then**
      move (in, out) from $M(y, t{+}1)$ to $M(x, t{+}1)$;
      $in \leftarrow w | (w, out) \in M(x,t)$;
return $\mathcal{M}_{t+1}$;

controller node, that maintains a current global view of the network topology and coordinates port-mapping decisions among ToR and OCS switches. Moreover, the controller node handles prioritization rules among nodes participating in current adjustment or service (routing and forwarding) operations, to ensure that they don't participate in conflicting operations.

Rounds and time-slots: The simulation is divided into synchronous rounds, and a message travels up to one hop per round. Each round is divided into 4 time-slots, as follows:

- `nodeInformStep`: Leaf-layer (ToR) switches (nodes) inform the controller node if they have messages to send;
- `controllerStep`: The controller node receives the network messages and computes the current round adjustment operations, informing all leaf and spine-layer switches which edges they must alter, and the network nodes about their new neighbors (e.g., children or parent), and grants permission to selected nodes to forward their messages;
- `nodeRoutingStep`: The nodes that received permission to forward their messages do so;
- `logRoundResults`: Network nodes receive their messages, inform the controller node if they are their message's final destination, and the controller node logs all relevant information about that round.

Multi-round adjustments: In the case of BST-based SANs, most adjustments take 2 rounds, and some (e.g. a *zig-zag* splay) take 3 rounds. In order to increase concurrency, OpticNet handles conflicts around such adjustments differently from some previous work [28]. Nodes participating in an adjustment operation are not locked for more than one round. To prevent multi-round adjustments from being interrupted by messages with higher priority, messages related to unfinished adjustments receive maximum priority.

OpticNet-SAN integration: The integration of OpticNet with SAN algorithms basically involves three steps. First one needs to create a new controller node class that inherits from the `NetworkController` class, add a constructor for it with the SAN algorithm specifications and a call to the parent class constructor. Then all decision making and attribute updating specific to the SAN algorithm (e.g., rank computation, path weight update or random number generation, in the case of a CBNet [29] implementation) is implemented as an override of the adjustment operations or message handling. Finally, it is needed to create a `CustomGlobal` class that will be responsible for calling the network constructor, firing messages into the network, defining the stop condition and any simulation configuration specifications.

Source code: We made the source code of the OpticNet framework available at `https://github.com/caiocaldeira3/OpticalNet`. Note that, in this implementation, we assume $\mathcal{N}$ to be the family of all BST, and switches to be half-duplex, so each switch has a reversed (mirrored) copy added to the spine layer.

IV. Analysis

In this section we analyze the correctness and optimality of OpticNet in the MM'. First we prove that OpticNet can represent any network $N_t \in \mathcal{N}$ at time t.

Theorem 1. *Let $\mathcal{N}$ be the family of constant degree ($\leq k$) graphs on a set V of n nodes. Let SW be a set of spine-layer p-port switches and $\mathcal{M}_t = N_t \in \mathcal{N}$ be the set of matchings computed by OpticNet at time t. If $(u,v) \in N_t$ then $\exists sw_x \in SW |\ (u,v) \in M(x,t), M(x,t) \in \mathcal{M}_t$.*

Proof. The proof is by induction on the number of edges.

Base case: If N_t has no edges, the claim is vacuously true since $(u,v) \notin N_t, \forall u \in V, v \in V$.

I.H.: If $(u',v') \in N'_t = N_t \setminus (u,v)$ then $\exists sw_{x'} \in SW | (u',v') \in M(x',t), M'(x',t) \in \mathcal{M}'_t$.

Induction step: Let $N'_t = N_t \setminus (u,v)$. We will show that if $(u,v) \cup N'_t \in \mathcal{N}$ then $\exists sw_x \in SW$ and $\exists sw_y \in SW$, such that $N_t = (u,v) \cup N'_t = \mathcal{M}_t = \mathcal{M}'_t \setminus \{M'(x,t) \cup$

$M'(y,t)\} \cup \{M(x,t) \cup M(y,t)\}\}$ and either $(u,v) \in M(x,t)$ or $(v,u) \in M(x,t)$.

OpticNet assigns either k *union* or $k/2$ *unit* switches $sw_z \in SW \mid \mathcal{P}(sw_z) = C_i \times C_j, u \in C_i, v \in C_j$. Let's refer to this set as S_v^u.

Case 1 (union switch, $i \neq j$): Since the degrees of nodes u and v must be $< k$ in N_t', there must exist at least one switch $sw_x \in S_v^u$, such that the input port u is free, and at least one $sw_y \in S_v^u$, in which the output port v is free in $\mathcal{M}_t'$. If $x = y$, then $M(x,t) \leftarrow M'(x,t) \cup \{(u,v)\}$, and we are done. If $x \neq y$, let $M = M'(x,t) \cup M'(y,t) \cup (u,v)$. Since $|M| \leq 2(p-1)+1$ and each port $w \in C_i \cup C_j$ appears at most twice in M, M can be partitioned into two matchings $M(x,t)$ and $M(y,t)$ of size $\leq p$ (see Algorithm 1, lines $28-32$).

Case 2 (unit switch, $i = j$): Since the degrees of nodes u and v must be $< k$ in N_t', there must exist at least one switch $sw_x \in S_v^u$, such that port u is free in $M'(x,t)$, either on the input or on the output side of sw_x, and at least one $sw_y \in S_v^u$, in which port v is free in $M'(y,t)$ either on the input or on the output side of sw_y.

If u and v are opposite (in-out) free ports, then we can apply Case 1 and the fact that any edge can be reversed on a unit switch to add either (u,v) or (v,u), depending on which side each free port resides on (i.e., if u is a free out port in sw_x, then we insert (v,u)).

If u and v are both input or both output free ports on sw_x and sw_y, and $x \neq y$, let $M''(x,t) = \{(w',w) \mid (w,w') \in M'(x,t)\}$ be the reversed matching of switch sw_x at time t. Now we insert (u,v) (or (v,u)) in $M''(x,t) \cup M'(y,t)$, in the same way as in the previous case.

Finally, if $x = y$, and sw_x has u and v free ports on the same side (both input or both output), it is enough to reverse every conflicting edge $(w,w') \in M'(x,t) \cup (u,v)$, i.e., $M(x,t) \leftarrow M'(x,t) \setminus (w,w') \cup (w',w)$, starting with $(w,w') = (u,w) \in M'(x,t)$ (or $(w,w') = (w,v) \in M'(x,t)$, if u and v are both free on the input side of sw_x). Note that each edge reversal will result in at most one conflict on either the input or the output port, and there may be no more than $p-2$ such conflicts. $\square$

The following theorem proves the correctness for the special case of BST.

Theorem 2. *Let T_t be any BST on n nodes at time t, and let SW and $\mathcal{M}_t$ be the set of spine-layer p-port switches and the set of matchings computed by OpticNet. If $(u,v) \in T_t$ then $\exists sw_x \in SW | (u,v) \in M(x,t), M(x,t) \in \mathcal{M}_t$.*

Proof. **Case 1 (union switch, $i \neq j$):** W.l.g, let's assume that $i < j$. If $u \in C_i, v \in C_j$ then $u < v$, v is the right-child of u and there is only 1 *union* switch $sw_R^{\times}$, with $\mathcal{P}(sw_R^{\times}) = C_i \times C_j$. Suppose $(u,v) \notin M(sw_R^{\times}, t)$. Since $u < v$, either u input port is matched to a different port, meaning u already has a right child, which is a contradiction, or v's output port is matched, meaning u already has a parent, also a contradiction. The proof is analogous for the case $u \in C_j$ and $v \in C_i$.

Case 2 (unit switch, $i = j$): If u and v belong to the same cluster C_i, then there are 2 *unit* switches $sw_L^{\cup}$ and $sw_R^{\cup}$, for the left and right-child edges, respectively, with $\mathcal{P}(sw_L^{\cup}) = \mathcal{P}(sw_R^{\cup}) = C_i \times C_i$. Suppose that $(u,v) \notin M(sw_L^{\cup}, t)$ and $(u,v) \notin M(sw_R^{\cup}, t)$. If v is the left child of u, than v's output port on $sw_L^{\cup}$ is occupied, a contradiction since v can't have two parents or u's input port on $sw_L^{\cup}$ is occupied, a contradiction since u can't have two left children. This proof is analogous for v being a right child of u. $\square$

We proceed by showing that the number of spine-layer switches is optimal.

Theorem 3. *Let $\mathcal{N}$ be the family of constant degree ($\leq k$) graphs on a set V of n nodes, and OPT we the minimum-size set of spine-layer switches with p in-out ports needed to connect n leaf-layer switches via a TMT two-layer architecture in the MM'. Let SW be a set of spine switches computed by OpticNet and $C = \left\lceil \frac{n}{p} \right\rceil$, then*

$$|OPT| \geq |SW| = C^2 \left\lceil \frac{k}{2} \right\rceil.$$

Proof. For simplicity sake, we assume that loop edges are possible $(v,v), \forall v \in V$ in some network topologies $N_t \in \mathcal{N}$.

Let OPT be any feasible solution for the MM'-PMP. Let $\mathcal{P}(sw_i) \subseteq V \times V, |\mathcal{P}(sw_i)| \leq p^2$ be the (static) *in-out port-set* of a spine-layer switch $sw_i \in OPT$. Let $\mathcal{P}(v)$ be the port-set of a leaf-layer switch $v \in V$. And let $\mathcal{P}(OPT) = \cup_{sw_i \in OPT} \mathcal{P}(sw_i)$ and $\mathcal{P}(V) = \cup_{v \in V} \mathcal{P}(v)$ be the (total) port-sets of spine and leaf layers, respectively.

Note that there must be a one-to-one assignment between leaf and spine-layer ports, so $|\mathcal{P}(OPT)| \geq |\mathcal{P}(V)|$. Moreover, since each spine switch connects p^2 in-out port pairs, we have:

$$|OPT| \geq \frac{|\mathcal{P}(OPT)|}{p^2} \geq \frac{|\mathcal{P}(V)|}{p^2}.$$

At any time t, each leaf switch $v \in V$ might need k simultaneous connections to any subset of k nodes out of n. So we have that $\mathcal{P}(v) \geq k \cdot n, \forall v \in V$. Since edges are not directed (full-duplex), it follows that:

$$|OPT| \geq \frac{\mathcal{P}(V)}{p^2} \geq \left\lceil \frac{k}{2} \right\rceil \cdot \frac{n^2}{p^2} \geq C^2 \left\lceil \frac{k}{2} \right\rceil.$$

$\square$

Finally, we show that any SAN algorithm runs on top of OpticNet, preserving the total cost guarantees.

Theorem 4. *Let $\mathcal{N}$ be the family of constant degree ($\leq k$) graphs on a set of n nodes. Consider any initial $N_0 \in \mathcal{N}$, a sequence of m messages σ and a SAN algorithm $\mathcal{A}$. Let SW be the set of spine-layer p-port switches and $\mathcal{M}_t = \cup_{sw_x \in SW} M(x,t)$ be the set of matchings computed by OpticNet at time t, such that $\mathcal{M}_t = N_t \in \mathcal{N}$. The total service, edge and switch adjustment, and work costs incurred by $\mathcal{A}$ on top of OpticNet to deliver all messages in σ are:*

$$\begin{aligned}
\mathit{Srv}(\mathit{OpticNet}(\mathcal{A}), N_0, \sigma) &= \mathit{Srv}(\mathcal{A}, N_0, \sigma), \\
\mathit{LinkAdj}(\mathit{OpticNet}(\mathcal{A}), N_0, \sigma) &\leq 2p \cdot \mathit{LinkAdj}(\mathcal{A}, N_0, \sigma), \\
\mathit{SwAdj}(\mathit{OpticNet}(\mathcal{A}), N_0, \sigma) &\leq 2 \cdot \mathit{LinkAdj}(\mathcal{A}, N_0, \sigma), \\
\mathit{Cost}(\mathit{OpticNet}(\mathcal{A}), N_0, \sigma) &= O(\mathit{Cost}(\mathcal{A}, N_0, \sigma)).
\end{aligned}$$

Proof. An adjustment operation can be defined in terms of a set of edges to be removed or added to a network topology at each time t. Algorithm 1 implements an edge-addition operation with cost $\leq 2p$ using the *edge-distance* metric and cost ≤ 2 using the *switch-cost* metric. Therefore, the asymptotic amortized cost of a SAN algorithm based on splaying, such as SplayNet [4], DiSplayNet [26] or CBNet [28], combined with OpticNet, remains the same. ☐

V. Evaluation

In this section we present our experimental results. We implemented three SAN algorithms on top of OpticNet:

- SN[3]: SplayNet [4], [30] centralized and sequential generalization of a splay tree;
- DSN: DisplayNet [26] distributed and concurrent version of SplayNet;
- CBN[4]: CBNet [29] distributed and concurrent generalization of CBTrees [27].

We used the following metrics, defined in Section II, to evaluate the simulation results:

- *Service cost*: the number of times a message was shared between adjacent nodes in the path from it's source to it's destination, we also compute the number of times a node, switch or input port was part of a routing operation.
- *Adjustment edge-distance cost*: the number of ports that were altered in an adjustment, caused by a message. For example, *zig-zig* rotation requires up to 12 link changes and a *zig-zag* requires up to 20 link updates.
- *Adjustment switch cost*: the number of switches whose ports were altered. For example, a *zig-zig* rotation requires up to 6, and a *zig-zag* requires up to 10 switch updates.
- *Throughput*: the average number of delivered messages per round.
- *Activity*: the number of times a node, port, switch or request has been used in a routing or adjustment operation.
- *Total work*: the total number of routing and adjustment operations in the network.

A. Workload traces

To measure the locality of reference present in a workload, we use the definition of *trace complexity* [16], which leverages only randomization and data compression operations. The amount of locality present in a workload can be measured based on the entropy of the communication sequence. The concept of entropy is related to the amount of information or the ability to compress the data. Intuitively, workloads with a low locality structure tend to have random sequences of communication pairs and, consequently, a low data compression rate. High-locality sequences tend to have a specific structure between requests that allows for better compression. In particular, we distinguish temporal and non-temporal components.

[3]`https://github.com/caiocaldeira3/SemiDisplayOpticNet`
[4]`https://github.com/caiocaldeira3/CBOpticalNet`

We used the following workload traces in our experiments, grouped according to their locality characteristics:

- **High non-temporal** and **low temporal**: **ProjectToR** [31] describes the communication probability distribution among $8,367$ pairs of nodes in a network of $n = 128$ nodes (top of racks), randomly selected from 2 production groups, executed between map-reduce operations, index builders, database and storage systems. We sample a sequence of $m = 10,000$ independent orders and identically distributed ($i.i.d.$) in time by the given communication matrix and repeat each experiment 30 times. **Skewed** corresponds to an artificial sequence of $10,000$ communication requests in a network of $n = 128$ nodes, using the method from [16]. The non-temporal locality component was produced using the *Zipf* distribution.
- **High temporal** and **low non-temporal**: The traces of **PFabric** [32] were generated by executing simulation scripts in NS2. We sample a sequence of $m = 1,000,000$ requests from a network of 144 nodes. **Bursty** was generated artificially with $m = 10,000$ and $n = 128$, using the method from [16].
- **Low locality**: The **Facebook** trace consists of real *Fbflow* packets collected from the production clusters of Facebook. The per-packet sampling is uniformly distributed with rate 1:30000, flow samples are aggregated every minute, and node IPs are anonymized. We mapped the anonymized IPs to a consecutive value range starting at 0. This resulted in a sequence of $1,000,000$ requests, originated in a 24-hour time window, in a network comprising 159 nodes.

To space the requests in time and make the timestamp sequences more realistic, we used a Poisson distribution with $\lambda = 0.25$ to determine the time of the entrance of each message in the network.

B. Results

Total work and throughput: In Figures 3 and 4, we can see the total work and throughput for all workloads. Note that these metrics are not affected by the OpticNet framework. What we can observe is that CBNet performs almost no adjustments, compared to SplayNet and DiSplaynet, and has a higher total (service) cost only in the Bursty workload. This is due to the extremely high temporal locality of this workload, which gives advantage to more aggressive reconfiguration algorithms, like SplayNet and DiSplayNet. We can also observe that the concurrent SANs have a much higher throughput than the sequential one (SN).

Activity: In Figures 5 and 6 we plot the CDFs of the percentage of active rounds and active ports per switch, for all workloads respectively, (except ProjecToR, which we discuss separately). In all plots, the smallest switch size was used ($p = 8$) for a network with $n \in \{128, 144, 367\}$ nodes, so the number of switches was quite high ($n' \in \{544, 684, 4324\}$).

Analyzing the results in Figure 5, we observe that most of the switches experience a significant amount of idle time. This is more explicit when dealing with SAN algorithms with low

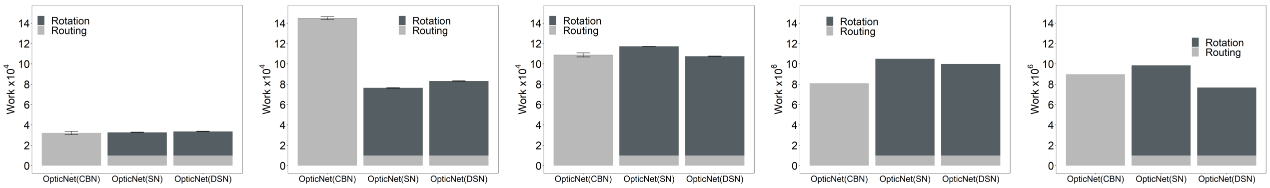

Fig. 3. Total work (all workloads): $m = 10,000$: (a) ProjecToR, (b) Bursty, (c) Skewed; $m = 1,000,000$: (d) Facebook, (e) PFabric.

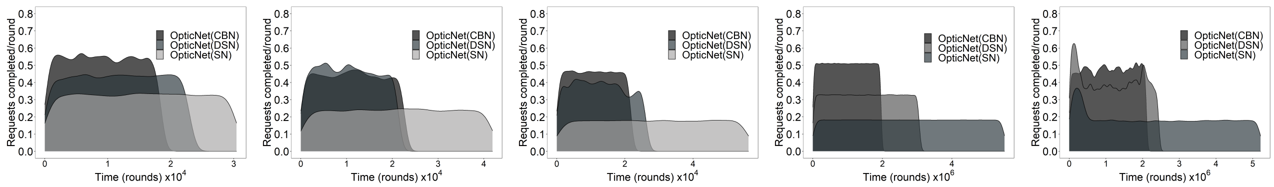

Fig. 4. Throughput (all workloads): $m = 10,000$: (a) ProjecToR, (b) Bursty, (c) Skewed; $m = 1,000,000$: (d) Facebook, (e) PFabric.

levels of adjustment, such as CBNet. In most simulations, the up time of most switches does not go over 60% of rounds. In some cases most switches aren't even up in any round. With SAN algorithms that are more adjustment intensive, like SplayNet and DiSplayNet, we see that the switch activity is better distributed, and fewer switches have no up time.

Analyzing the results in Figure 6, which shows how many ports of a switch have been active at least in one round, we get similar results in nature, with CBNet presenting a higher concentration of active ports, while SplayNet and DisplayNet present a more balanced port activity among switches.

When comparing the network activity over the the Bursty, Skewed, PFabric and Facebook datasets, we would like to point out that since splay operations typically involve more than one switches, we can see that DiSplayNet shows a more even distribution of active rounds per switch than CBNet. SplayNet, however, due to its sequential nature, presents lower percentiles than its distributed counter-part.

Another interesting observation is the discrepancy between active ports percentage between CBNet and DiSplayNet and SplayNet, with CBNet presenting a much higher concentration of active ports, even in the longer workload sequences (PFabric and Facebook, where $m = 1,000,000$). Over this longer sequences (Figures 6c and 6d), we also observe a congruence in the results for DiSplayNet and SplayNet.

Variable switch size (p): In Figures 7 and 8 we plot the CDFs of the percentage of active rounds and active ports per switch, for the ProjecToR workload, with a variable number of switch ports. We can analyze more carefully the impact of a greater number of switches on the network activity. We can see clearly, for example, that by increasing the switch size, the number of completely inactive switches drops dramatically. In particular, the cap for active port percentage of a switch for CBNet decreases with higher number of ports, while SplayNet and DisplayNet actually manage to distribute the workload more evenly over the different switch ports.

VI. Related Work

Reconfigurable network topologies can be grouped in two basic types: demand-oblivious, e.g. [8], [9], [33], [34] and demand-aware, e.g. [3]–[5], [19], [24], [26], [35]–[40], or a combination of both, e.g., Cerberus [41]. Most existing demand-aware architectures rely on an estimation of traffic matrices [5], [17], [37], [38], [42], [43] which can limit the granularity and reactivity of the network, but there are also more fine-grained approaches such as [24], [35], which however rely on a centralized control.

The majority of datacenter networks with on-demand reconfigurations use full crossbar, 3D-MEMS-based OCS [18], or Wavelength Division Multiplexing-based switching [17]. Several prototypes based on OCS have been built, and their advantages demonstrated, e.g., the Helios prototype [5] uses a commercially available Glimmerglass OCS with 64 ports, while OSA [19] provides a testbed that consists of a Polatis Series 1000 32-port OCS. To operate such RDCNs efficiently, the self-adjusting network topology has to be mapped to a set of OCS dynamically in real-time.

While promising performance results have been demonstrated with various prototypes of demand-aware reconfigurable networks, today, it is often challenging to experiment with these technologies, as they are usually based on custom-built prototypes and rely on tailored hardware and software which is not publicly available. One example of a framework that supports experimentation and reproducibility is ExReC [44]. It uses off-the-shelf hardware (Polatis Series 6000n 32×32 OCS [25]) for evaluating different hybrid reconfigurable topologies and applications.

The closest work to ours is probably [23], where online strategies are proposed to adapt SAN algorithms for the matching model [22], based on spaced out (less frequent) network adjustments using existing (sequential) SAN algorithms, such as SplayNets [4] and ReNets [24].

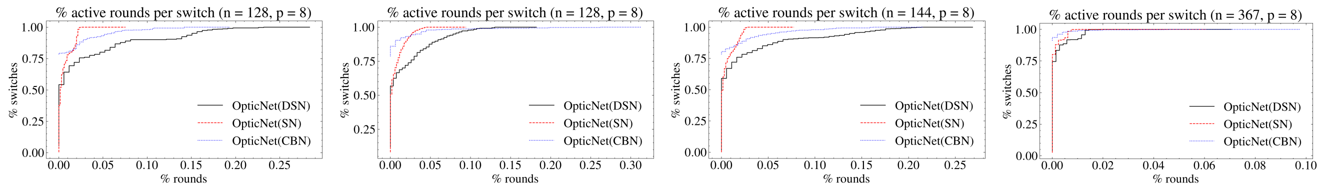

Fig. 5. CDF Active rounds percentage per switch: constant switch size $p = 8$, $n \in \{128, 144, 367\}$ (a) Bursty, (b) Skewed, (c) Facebook, (d) PFabric.

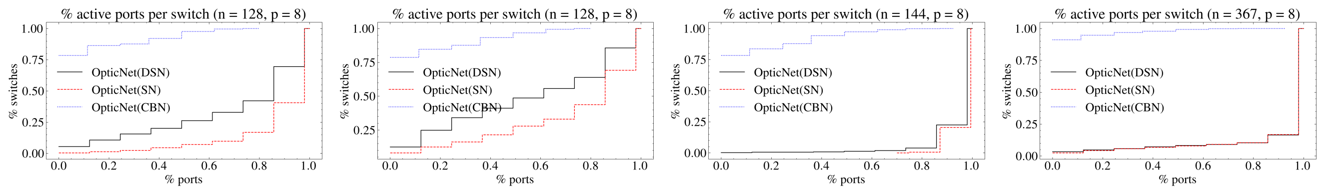

Fig. 6. CDF Active ports percentage per switch: constant switch size $p = 8$, $n \in \{128, 144, 367\}$ (a) Bursty, (b) Skewed, (c) Facebook, (d) PFabric.

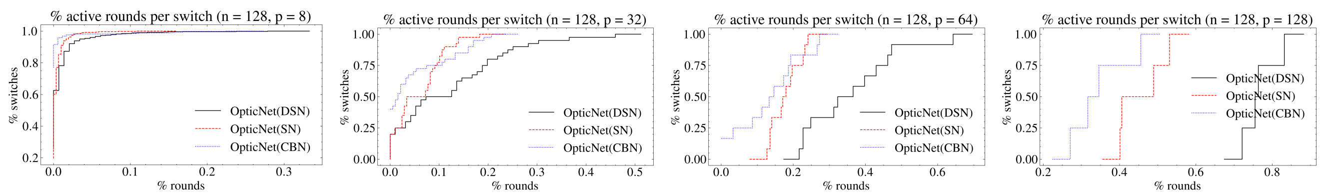

Fig. 7. ProjecToR: CDF Active rounds percentage per switch: variable switch size $p \in \{8, 32, 64, 128\}, n = 128$

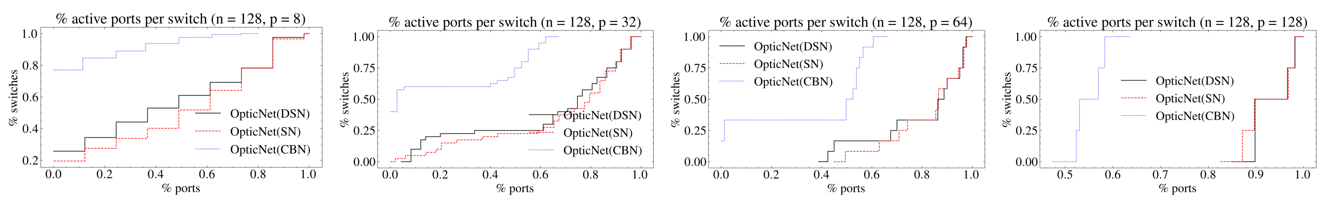

Fig. 8. ProjecToR: CDF Active ports percentage per switch: variable switch size $p \in \{8, 32, 64, 128\}, n = 128$.

VII. Conclusion

Even though emerging hardware reconfiguration technologies introduce an additional degree of freedom to the datacenter network design problem, they still face challenging system and algorithm design problems. Due to additional control logic and several milliseconds latency of the state-of-the-art demand-aware optical switches, their cost can only be amortized for large flows over longer terms.

In this work, we used a synchronous distributed system model. OpticNet assumes a consistent and communication-closed round structure provided by the distributed system. While many solutions to problems in distributed computing assume lock-step rounds, real-world distributed systems are usually not perfectly synchronous. In practice, message loss is present as a result of job dropping by a real-time scheduler or an unreliable communication channel.

While our contribution is still theoretical in nature, we believe it constitutes an interesting step forward toward practical self-adjusting networks. Our work opens interesting avenues for future research, such as considering implications on network and transport layers and understanding the trade-off between maximizing throughput and minimizing latency in both demand-aware and oblivious self-adjusting networks [34].

References

[1] S. Kandula, J. Padhye, and P. Bahl, "Flyways to de-congest data center networks," *Proc. ACM Workshop on Hot Topics in Networks (HotNets)*, 2009.

[2] C. Avin and S. Schmid, "Toward demand-aware networking: A theory for self-adjusting networks," in *ACM SIGCOMM Computer Communication Review (CCR)*, 2018.

[3] M. Ghobadi, R. Mahajan, A. Phanishayee, N. Devanur, J. Kulkarni, G. Ranade, P.-A. Blanche, H. Rastegarfar, M. Glick, and D. Kilper, "Projector: Agile reconfigurable data center interconnect," in *Proc. ACM SIGCOMM*. New York, NY, USA: ACM, 2016, pp. 216–229.

[4] S. Schmid, C. Avin, C. Scheideler, M. Borokhovich, B. Haeupler, and Z. Lotker, "Splaynet: Towards locally self-adjusting networks," *IEEE/ACM Trans. Netw.*, vol. 24, no. 3, pp. 1421–1433, Jun. 2016.

[5] 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," *ACM SIGCOMM Computer Communication Review*, vol. 40, no. 4, pp. 339–350, 2010.

[6] C. Avin, K. Mondal, and S. Schmid, "Demand-aware network designs of bounded degree," in *Proc. International Symposium on Distributed Computing (DISC)*, 2017.

[7] N. Hamedazimi, Z. Qazi, H. Gupta, V. Sekar, S. R. Das, J. P. Longtin, H. Shah, and A. Tanwer, "Firefly: A reconfigurable wireless data center fabric using free-space optics," in *ACM SIGCOMM Computer Communication Review*, vol. 44. ACM, 2014, pp. 319–330.

[8] 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," *arXiv preprint arXiv:1903.12307*, 2019.

[9] 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 *Proc. ACM SIGCOMM*, 2017, pp. 267–280.

[10] 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," *Proc. ACM SIGCOMM Computer Communication Review (CCR)*, vol. 45, no. 4, pp. 183–197, 2015.

[11] V. Liu, D. Halperin, A. Krishnamurthy, and T. Anderson, "F10: A fault-tolerant engineered network," in *Proceedings of the 10th USENIX Conference on Networked Systems Design and Implementation*, ser. nsdi'13. Berkeley, CA, USA: USENIX Association, 2013, pp. 399–412. [Online]. Available: http://dl.acm.org/citation.cfm?id=2482626.2482665

[12] C. Guo, G. Lu, D. Li, H. Wu, X. Zhang, Y. Shi, C. Tian, Y. Zhang, and S. Lu, "Bcube: a high performance, server-centric network architecture for modular data centers," *Proc. ACM SIGCOMM Computer Communication Review (CCR)*, vol. 39, no. 4, pp. 63–74, 2009.

[13] H. Wu, G. Lu, D. Li, C. Guo, and Y. Zhang, "Mdcube: a high performance network structure for modular data center interconnection," in *Proc. ACM International Conference on Emerging Networking Experiments and Technologies (CONEXT)*, 2009, pp. 25–36.

[14] S. Kassing, A. Valadarsky, G. Shahaf, M. Schapira, and A. Singla, "Beyond fat-trees without antennae, mirrors, and disco-balls," in *Proc. ACM SIGCOMM*, 2017, pp. 281–294.

[15] A. Singla, C.-Y. Hong, L. Popa, and P. B. Godfrey, "Jellyfish: Networking data centers, randomly," in *Proc. USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, vol. 12, 2012, pp. 17–17.

[16] C. Avin, M. Ghobadi, C. Griner, and S. Schmid, "On the complexity of traffic traces and implications," in *Proc. ACM SIGMETRICS*, 2020.

[17] L. Chen, K. Chen, Z. Zhu, M. Yu, G. Porter, C. Qiao, and S. Zhong, "Enabling wide-spread communications on optical fabric with megaswitch," ser. NSDI'17, 2017, p. 577–593.

[18] J. Zerwas, W. Kellerer, and A. Blenk, "What you need to know about optical circuit reconfigurations in datacenter networks," in *The 33nd International Teletraffic Congress (ITC 33)*, 2021.

[19] K. Chen, A. Singla, A. Singh, K. Ramachandran, L. Xu, Y. Zhang, X. Wen, and Y. Chen, "Osa: An optical switching architecture for data center networks with unprecedented flexibility," *IEEE/ACM Transactions on Networking (TON)*, vol. 22, no. 2, pp. 498–511, 2014.

[20] D. Sleator and R. Tarjan, "Self-adjusting binary search trees," *Journal of the ACM (JACM)*, vol. 32, no. 3, pp. 652–686, 1985.

[21] D. D. Sleator and R. E. Tarjan, "Amortized efficiency of list update and paging rules," *Commun. ACM*, vol. 28, no. 2, pp. 202–208, Feb. 1985.

[22] C. Avin, C. Griner, I. Salem, and S. Schmid, "An online matching model for self-adjusting tor-to-tor networks," 2020.

[23] E. Feder, I. Rathod, P. Shyamsukha, R. Sama, V. Aksenov, I. Salem, and S. Schmid, "Brief announcement: Toward self-adjusting networks for the matching model," in *33rd ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, 2021.

[24] C. Avin and S. Schmid, "Renets: Statically-optimal demand-aware networks," in *Proc. SIAM Symposium on Algorithmic Principles of Computer Systems (APOCS)*, 2021.

[25] "Polatis Series 6000," www.polatis.com, [Online; accessed 10-May-2022].

[26] B. Peres, O. Souza, O. Goussevskaia, S. Schmid, and C. Avin, "Distributed self-adjusting tree networks," in *Proc. IEEE INFOCOM*, 2019.

[27] Y. Afek, H. Kaplan, B. Korenfeld, A. Morrison, and R. E. Tarjan, "Cbtree: A practical concurrent self-adjusting search tree," in *Proc. DISC*. Berlin, Heidelberg: Springer-Verlag, 2012, pp. 1–15.

[28] O. A. de O. Souza, O. Goussevskaia, and S. Schmid, "Cbnet: Demand-aware tree topologies for reconfigurable datacenter networks," *Computer Networks*, vol. 213, p. 109090, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128622002249

[29] O. A. d. O. Souza, O. Goussevskaia, and S. Schmid, "Cbnet: Minimizing adjustments in concurrent demand-aware tree networks," in *IEEE Int. Parallel and Distributed Processing Sym. (IPDPS)*, 2021, pp. 382–391.

[30] C. Avin, B. Haeupler, Z. Lotker, C. Scheideler, and S. Schmid, "Locally self-adjusting tree networks," in *Proceedings of the 2013 IEEE 27th International Symposium on Parallel and Distributed Processing*, ser. IPDPS '13. Washington, DC, USA: IEEE Computer Society, 2013, pp. 395–406. [Online]. Available: http://dx.doi.org/10.1109/IPDPS.2013.40

[31] "Projector dataset," www.microsoft.com/en-us/research/project/projector-agile-reconfigurable-data-center-interconnect, 2016.

[32] M. Alizadeh, S. Yang, M. Sharif, S. Katti, N. McKeown, B. Prabhakar, and S. Shenker, "Pfabric: Minimal near-optimal datacenter transport," in *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM*, ser. SIGCOMM '13. New York, NY, USA: Association for Computing Machinery, 2013, p. 435–446. [Online]. Available: https://doi.org/10.1145/2486001.2486031

[33] 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 *Proc. ACM SIGCOMM*, 2020, pp. 782–797.

[34] D. Amir, T. Wilson, V. Shrivastav, H. Weatherspoon, R. Kleinberg, and R. Agarwal, "Optimal oblivious reconfigurable networks," in *Proc. of the 54th Annual ACM SIGACT Symposium on Theory of Computing*, ser. STOC 2022, 2022, p. 1339–1352.

[35] J. Kulkarni, S. Schmid, and P. Schmidt, "Scheduling opportunistic links in two-tiered reconfigurable datacenters," in *33rd ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, 2021.

[36] C. Avin, A. Hercules, A. Loukas, and S. Schmid, "rdan: Toward robust demand-aware network designs," in *Information Processing Letters (IPL)*, 2018.

[37] S. B. Venkatakrishnan, M. Alizadeh, and P. Viswanath, "Costly circuits, submodular schedules and approximate carathéodory theorems," *Queueing Systems*, vol. 88, no. 3-4, pp. 311–347, 2018.

[38] G. Porter, R. Strong, N. Farrington, A. Forencich, P. Chen-Sun, T. Rosing, Y. Fainman, G. Papen, and A. Vahdat, "Integrating microsecond circuit switching into the data center," *SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 4, pp. 447–458, Aug. 2013. [Online]. Available: http://doi.acm.org/10.1145/2534169.2486007

[39] A. Singla, A. Singh, K. Ramachandran, L. Xu, and Y. Zhang, "Proteus: a topology malleable data center network," in *Proc. ACM Workshop on Hot Topics in Networks (HotNets)*, 2010.

[40] E. Feder, I. Rathod, P. Shyamsukha, R. Sama, V. Aksenov, I. Salem, and S. Schmid, "Lazy self-adjusting bounded-degree networks for the matching model," in *Proc. IEEE Conference on Computer Communications (INFOCOM)*, 2022.

[41] C. Griner, J. Zerwas, A. Blenk, S. Schmid, M. Ghobadi, and C. Avin, "Cerberus: The power of choices in datacenter topology design (a throughput perspective)," in *Proc. ACM SIGMETRICS*, 2022.

[42] G. Wang, D. G. Andersen, M. Kaminsky, K. Papagiannaki, T. Ng, M. Kozuch, and M. Ryan, "c-through: Part-time optics in data centers," *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 4, pp. 327–338, 2011.

[43] C. Avin, K. Mondal, and S. Schmid, "Demand-aware network design with minimal congestion and route lengths," in *Proc. IEEE INFOCOM*, 2019.

[44] J. Zerwas, C. Avin, S. Schmid, and A. Blenk, "ExRec: Experimental Framework for Reconfigurable Networks Based on Off-the-Shelf Hardware," in *Proc. of the Symposium on Architectures for Networking and Communications Systems*, ser. ANCS, 2021, p. 66–72.