Scheduling Opportunistic Links in Two-Tiered Reconfigurable Datacenters

Janardhan Kulkarni
Microsoft Research
Redmond, USA
jakul@microsoft.com

Stefan Schmid
Faculty of Computer Science
University of Vienna
Vienna, Austria
stefan_schmid@univie.ac.at

Paweł Schmidt
Institute of Computer Science
University of Wrocław
Wrocław, Poland
pawel.schmidt@cs.uni.wroc.pl

ABSTRACT

Reconfigurable optical topologies are emerging as a promising technology to improve the efficiency of datacenter networks. This paper considers the problem of scheduling opportunistic links in reconfigurable datacenters such as ProjecToR. We study the online setting and aim to minimize flow completion times. The problem is a two-tier generalization of classic switch scheduling problems. We present a stable-matching algorithm which is $O(\varepsilon^{-2})$-competitive against an optimal offline algorithm, in a resource augmentation model: the online algorithm runs $2 + \varepsilon$ times faster. Our algorithm and result are fairly general and allow for different link delays and also apply to hybrid topologies which combine fixed and reconfigurable links. Our analysis is based on LP relaxation and dual fitting.

CCS CONCEPTS

• **Networks** → **Network architectures**; • **Theory of computation** → **Online algorithms**;

KEYWORDS

online algorithms, reconfigurable datacenters, switch scheduling, LP relaxation, dual fitting

1 INTRODUCTION

Given the popularity of data-centric applications and machine learning, the traffic in datacenters is growing explosively. Accordingly,

over the last years, great efforts were made to render these networks more efficient, on various layers of the networking stack [1], including the physical network topology [2–6].

The next frontier toward more efficient datacenter networks are *reconfigurable* optical topologies [7–13], and in particular, demand-aware topologies such as [9–12, 14–16] which can dynamically adapt towards the traffic patterns they serve. This is attractive as empirical studies show that datacenter workloads are skewed and bursty, featuring much temporal and spatial structure [17–19], which may be exploited in adaptive infrastructures. For example, these technologies allow to flexibly transmit elephant flows via opportunistic links that provide shortcuts between the frequently communicating datacenter racks.

A key issue for the efficient operation of reconfigurable datacenter networks concerns the scheduling of the opportunistic links. As the number of these links is limited, they should be used for the most significant transmissions. This however is challenging as scheduling decisions need to be performed in an online manner, when the demand is not perfectly known ahead of time.

This paper studies this scheduling problem from a competitive analysis perspective: we aim to design an online scheduling algorithm which does not require any knowledge about future demands, but performs close to an optimal offline algorithm which knows the entire demand ahead of time. In particular, we consider a two-stage switch scheduling model as it arises in existing datacenter architectures, such as ProjecToR which is based on free-space optics [11]. In a nutshell (a formal model will follow shortly), we consider a *two-tier* architecture where traffic demands (modelled as *packets*) arise between Top-of-Rack (ToR) switches, while opportunistic links are between lasers and photodetectors, and where many laser-photodetector combinations can serve traffic between a pair of ToRs. The goal is to minimize the packet (i.e., flow) completion times over all packets in the system.

The problem is reminiscent of problems in classic switch scheduling [20, 21], as in each time step, an optical switch allows to "transmit a matching". However, the two-stage version turns out to introduce several additional challenges (as also pointed out in [11]), which we tackle in this paper.

1.1 Our Contribution

This paper initiates the study of online switch scheduling algorithms for a multi-stage model which is motivated by emerging reconfigurable optical datacenter architectures. Our main contribution is an online stable-matching algorithm attaining the competitive ratio of $O(\varepsilon^{-2})$ against a powerful hindsight-optimal algorithm, in a resource augmentation model in which the online algorithm

runs $2 + \varepsilon$ times faster. Our analysis relies on linear programming relaxation and dual fitting: we formulate primal and dual linear programs to which we will charge the costs of our online algorithm.

Our algorithm and result allow for different link delays and also apply to hybrid datacenter topologies as they are often considered in the literature: topologies which combine fixed and reconfigurable links.

We emphasize that resource augmentation is necessary for obtaining competitive algorithms: Dinitz et al. [22] prove that even in single-tier networks, no randomized algorithm can be competitive against an adversary with matching transmission speed.

1.2 Organization

The remainder of this paper is organized as follows. We introduce our formal model in Section 2 and give an overview of our algorithm and approach in Section 3. Our algorithm is described in more details in Section 4 and analyzed in Section 5. After reviewing related work in Section 6, we conclude our contribution in Section 7.

2 MODEL

We consider a hybrid optical network which consists of a fixed and a reconfigurable topology. We describe this network as a graph $G = (V, E, d)$ where V is the set of vertices partitioned into the following four layers (modelling architectures such as [11]): sources S, transmitters T, receivers R, destinations D. Each transmitter $t \in T$ is attached (has an edge) to a particular source $\text{src}(t)$ and each receiver $r \in R$ is attached to a particular destination $\text{dest}(r)$ (a single source or destination may have multiple transmitters or receivers attached). The edges between transmitters and receivers form an optical reconfigurable network. For transmitter $t \in T$, we denote the set of receivers adjacent to t in G by $R(t)$; and for receiver $r \in R$, $T(r)$ is the set of transmitters adjacent to r in G. The fixed part of the network is a set $E_\ell \subseteq E$ of direct source-destination *links*.

At any time $\tau \in \mathbb{N}_+$, a transmitter t may have at most one active edge connecting it with a receiver from $R(t)$, and each receiver r may have at most one active incoming edge from one of transmitters $T(r)$. For any edge $e \in E$, the *delay* of that edge is defined by $d(e) \in \mathbb{N}$, that is, $s \cdot d(e)$ is the time required to transmit a packet of size s through that edge. If e is a transmitter-receiver connection, then its delay is at least 1.

We study the design of a topology scheduler whose input is a sequence of packets Π arriving in an online fashion. A packet $p \in \Pi$ of weight $w_p > 0$ which arrives at time a_p at source node $\text{src}(p) \in S$, has to be routed to destination $\text{dest}(p) \in D$. For packet $p \in \Pi$, let E_p be the set of transmitter-receiver edges from the reconfigurable network that might be used to deliver p, i.e., $E_p = \{(t, r) \in T \times R : \text{src}(t) = \text{src}(p) \text{ and } \text{dest}(r) = \text{dest}(p)\}$. If there exists a link connecting $\text{src}(p)$ with $\text{dest}(p)$, the delay of that link is $d_\ell(p) = d(\text{src}(p), \text{dest}(p))$. By Π_ℓ we denote the set of all packets that can be transmitted through the fixed network.

In this paper, we assume that packets are of uniform size. However, this assumption is without loss of generality in the speed augmentation model. By standard arguments [23], one can treat a packet p of size ℓ_p as ℓ_p unit-length packets each of weight w_p/ℓ_p. Hence, in the rest of the paper we assume packets of uniform size.

The goal of the algorithm is to route all packets from their sources to destinations. A packet can be transmitted either through the reconfigurable network or the slower direct connection (if available) between source and destination. All transmissions happen only at times $\tau \in \mathbb{N}_+$, but packets may arrive between transmissions. When packet p arrives at time $\tau \in (\tau', \tau' + 1]$ for some $\tau' \in \mathbb{N}_+$, it will be available for transmission in the next transmission slot, namely at time $\tau' + 1$. Therefore, we may assume, that packets arrive only at integral times (i.e., the arrival time of packet is shifted from τ to $\lceil \tau \rceil$) and they immediately can be transmitted through the network.

The weighted latency of packet p is defined as the weight of p multiplied by the time p spent in the system before it was delivered. The cost of an algorithm is a sum of packets' weighted latencies.

In the speed augmentation model, an online algorithm A with speedup s can be transformed into online algorithm B with speedup $s \cdot (1 + \varepsilon)$ such that, the cost of B is at most the $O(1 + \varepsilon^{-1})$ times larger than the fractional latency of A (see e.g., [24]). Therefore, in the analysis we will use the fractional latency (which we define next) to measure the cost of an algorithm.

The *weighted fractional latency* is defined as follows: when a fraction $0 < x \leq 1$ of packet p reaches its destination $\text{dest}(p)$ during transmission step τ, it incurs the weighted latency of $x \cdot w_p \cdot (\tau + 1 - a_p)$ (in this case, $\tau + 1$ is the time when that part of packet p reaches $\text{dest}(p)$).

In particular, when the algorithm transmits packet p through the link that connects $\text{src}(p)$ and $\text{dest}(p)$, the weighted latency of p is $w_p \cdot d_\ell(p)$. On the other hand, routing packet p via path $\text{src}(p) - t - r - \text{dest}(p)$, where $(t, r) \in E_p$ incurs weighted latency of $w_p \cdot (d(\text{src}(p), t) + d(t, r) + d(r, \text{dest}(p)))$.

The goal of the algorithm is to deliver all packets and minimize the total weighted latency of its schedule. The overall performance of the algorithm is measured with the standard notion of the competitive ratio, defined as the worst-case ALG-to-OPT cost ratio, where OPT is the optimal offline solution with limited transmission speed. An example input and transmission schedule are illustrated in Figure 1.

3 OVERVIEW OF ALGORITHM AND TECHNIQUES

Before presenting our approach and analysis in details in the following sections, we give a quick overview here. Our algorithm for the problem is based on a generalization of the stable-matching algorithm [25] for two-tier networks. Informally, in our algorithm, each transmitter maintains a queue of packets that are not scheduled yet. The packets in the queue are sorted in the decreasing order of weights. At each time step, our algorithm finds a stable matching between transmitters and receivers as follows: In the scheduler we are given a bipartite graph $B = \{(T \cup R), E\}$, between the set of transmitters T and set of receivers R; the edge set E denotes the connections between transmitters and receivers. At time step τ, we assign the edge $e = (t, r)$ connecting the transmitter t to the receiver r a weight w_e, which is equal to the highest weight packet in the queue of transmitter t at time instant τ which wants to use the edge e. Taking the weights of edges as *priorities*, our algorithm simply computes a stable matching M in the graph B, and schedules M at time step τ. (Note that since priorities in our algorithm are
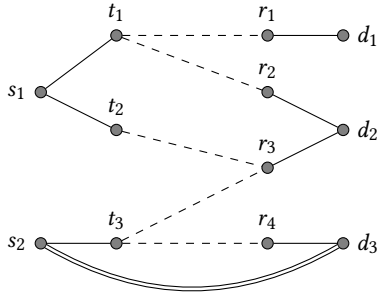
packet	path	arrival	transmission	edge
p_1	$s_1 \to d_1$	1	1	(t_1, r_1)
p_2	$s_1 \to d_2$	1	2	(t_1, r_2)
p_3	$s_2 \to d_2$	1	1	(t_3, r_3)
p_4	$s_2 \to d_2$	2	2	(t_3, r_3)
p_5	$s_2 \to d_3$	2	2	(s_2, d_3)

Figure 1: Input graph G. Solid lines represent edges between sources and transmitters or between receivers and destinations. Available connections in the reconfigurable layer are shown as dashed lines. The double line represents a direct source-destination link in the fixed layer. In this example, $d(e) = 1$ for any edge in the reconfigurable layer, $d(s_2, d_3) = 4$ and $d(e) = 0$ for all other edges. The table contains a set of packets (all of unit weight) and a feasible solution. In the first transmission step packets $\{p_1, p_3\}$ are transmitted, and in the second step $\{p_2, p_4\}$ are sent through the reconfigurable network and p_5 via link (s_2, d_3). The weighted latency of this solution is 9, while the optimal weighted latency of this instance is 7 (sending p_5 in the third step via (t_3, r_4)).

symmetric, one can compute a stable matching by a simple greedy algorithm.)

However, how should we assign an incoming packet to a (transmitter, receiver) pair? In our algorithm, as soon as a packet arrives, it is dispatched to a specific (transmitter, receiver) pair via which our algorithm commits to eventually transmitting the packet. This is the decision that complicates routing in two-tier networks. Our dispatch policy estimates the worst case impact of transmitting a packet via a specific (transmitter, receiver) pair, taking into account the set of queued packets in the system. In particular, we estimate how much latency of the system *increases* if a packet is transmitted via a (transmitter, receiver) pair. Finally, we choose the (transmitter, receiver) pair which has the least impact. We show that this greedy-dispatch policy coupled with stable matching is indeed competitive in the speed augmentation model [26], where we assume that the online algorithm can transmit the packets at twice the rate compared to the optimal offline algorithm. It is not hard to show that without speed augmentation, no online algorithm can be competitive [22].

Our algorithm and its analysis via dual fitting is inspired by scheduling for unrelated machines [23], and combines two disparate research directions: switch scheduling and scheduling for unrelated machines.

Our analysis via dual-fitting works as follows. First we write a linear programming relaxation for the underlying optimization problem, and then we take the dual of the LP. This is done assuming that we know the entire input, which we can do because LP duality is only used in the analysis. The weak duality theorem states that *any feasible solution* to the dual is a lowerbound on the optimal primal solution (which in turn is a lowerbound on the optimal solution to our problem). The crux of the dual-fitting analysis is to relate the cost of our algorithm to a feasible dual solution, thus allowing us to compare our cost to the optimal solution.

The dual of our LP for the problem has a rather interesting form. It consists of variables α_p for each packet p. For each time step τ, we also have variables $\beta_{t,\tau}$ for each transmitter t and $\beta_{r,\tau}$ for a receiver r. We interpret α_p as the latency seen by packet p, and the $\beta_{t,\tau}, \beta_{r,\tau}$ variables as the sets of packets that are waiting to use the transmitter t and receiver r at time step τ. Clearly, the sum over the α_p variables is equal to the total latency seen by packets. It is not hard to argue that the same also holds for the β variables. However, the crucial part of the analysis is to show that indeed such an interpretation of the dual variables is an almost-feasible solution. This is done by showing that our setting of dual variables violates all the dual constraints are exceeded by no more than twice. Verifying the dual constraints crucially uses both our algorithmic decisions regarding dispatch policy and the stable matching algorithm. One could also interpret that our algorithmic decisions were in fact driven by the dual LP, in the sense of primal-dual algorithms [27].

4 ONLINE SCHEDULING ALGORITHM

In this section we present our online scheduling algorithm Alg, and will defer its competitive analysis to the next section. Algorithm Alg comprises two natural components. A scheduler for packets in the reconfigurable network, which relies on the repeated computation of *stable matchings*, and a dispatcher which upon arrival of a packet decides whether the packets will use a direct connection or the reconfigurable links. In the latter case, the dispatcher *assigns* packet p to some edge (i.e., a transmitter-receiver pair) that connects source and destination of p. In the dispatching rules, we assume the transmission takes the whole step, even if the fraction of packet sent in that round is small. This will change the competitive ratio only by a constant factor.

More precisely, the dispatcher attempts to minimize the weighted latency increase caused by p (the latency of p, and the latencies of other packets). To this end, it needs to account for the different transmission times in the reconfigurable part of the network. The idea is to split packets into *chunks*, which can be transmitted in a single step (the size of a chunk depends on the delay of the assigned edge). The dispatcher is formally defined in Section 4.2. The scheduler then greedily chooses the subset of chunks to be transmitted at each step. The set of edges associated with each

chunk forms a stable matching. This process is presented in details in Section 4.3.

This scheduling policy of ALG might lead to packet starvation, that is, some packet p might be delayed forever. However, if this is the case, in each round, there must be some heavier packet that is being sent instead of p. All these packets must be also serviced by the optimal schedule, which makes its cost infinite as well.

4.1 Stable Matching and Blocking

A matching M is stable with respect to symmetric weights w if for any edge $e \notin M$, there exists edge $e' \in M$ adjacent to e such that $w_{e'} \geq w_e$. We say that edge e' blocks edge e. The scheduler at each time τ, transmits a set of packets whose assigned edges form a stable matching. We will say that a chunk c blocks another chunk c' when c is transmitted at time τ and $w_c \geq w_{c'}$, and edges assigned to c and c' share a transmitter or a receiver.

4.2 Dispatcher

At time τ, the dispatcher handles packets that arrived since time $\tau - 1$. The packets are processed one by one. We will say that a packet p' arrived before p if $a_{p'} \leq a_p$ or $a_{p'} = a_p$ and p' was already handled by the dispatcher before processing p. Each packet assigned to the reconfigurable network is split into chunks (parts of packet that can be transmitted in a single time step). For chunk c we denote by $p(c)$ the packet to which chunk c belongs. We say that a chunk is *pending* if it has not been transmitted through the reconfigurable network. For a set of chunks C we denote the total weight of chunks in C by $w(C)$.

For each packet p, let B_p be the set of chunks of packets that arrived before p in the input sequence and that are pending at the time when p is processed. We define the *impact* of p as the weighted latency of chunks from B_p that are blocked by (chunks of) p plus the weighted latency of (chunks of) p incurred in rounds when p was blocked by a chunk from B_p, or by its own chunks. In particular, if packet p is transmitted through the fixed network, its impact is just the weighted latency of p, that is $w_p \cdot d_\ell(p)$.

Ideally, when packet p arrives, we would like to minimize the impact of this packet. This is, however, impossible to compute online: when the stable matching changes as more packets arrive, the impact of packet p might change as well, although the set B_p does not change (it is a property of the input sequence, not the algorithm). An example of such a situation is depicted on Figure 2.

Instead, the algorithm minimizes the worst-case impact of p. Namely, for each edge $e \in E_p$, it computes, assuming that p is assigned to e, how many chunks from B_p might block p in the future and how many chunks from B_p might be blocked by p. Then, the algorithm minimizes the worst-cast impact by assigning p to either the edge from the reconfigurable network or to the direct fixed link between source and destination of p (only if such link exists).

Formally, when packet p is assigned to edge e, it is split into $d(e)$ (the delay of e) chunks, each of size $1/d(e)$ and weight $w_p/d(e)$. Let $C_p(e)$ be the set of these chunks, and let $A_p(e)$ be the set of chunks from B_p that are assigned to use the edge adjacent to e. We partition the set $A_p(e)$ into two disjoint subsets: $H_p(e)$ containing those chunks that may delay $C_p(e)$ (i.e., at least as heavy as $w_p/d(e)$) and

$L_p(e)$, which might be delayed by $C_p(e)$ (i.e., lighter than $w_p/d(e)$). Note that these definitions require that from two chunks of the same weight, the chunk of the earlier arriving packet is preferred. This will be preserved by the scheduler in the next section.

The worst-case impact of p assigned to e is then $\Delta_p(e) = w_p \cdot \left(d(u) + \frac{d(e)+1}{2} + d(v) \right) + w_p \cdot |H_p(e)| + d(e) \cdot w\left(L_p(e)\right)$. The first summand is the weighted latency of chunks of p (note that chunks $C_p(e)$ delay each other). The remaining two summands account for the latency increase coming from p interacting with other chunks: i.e., they count the number of chunks from $A_p(e)$ that might block p, and the weighted latency of packets from $A_p(e)$ that may be blocked by p (all $|C_p(e)| = d(e)$ chunks of p might block $L_p(e)$).

Let $e = \arg\min_{e' \in E_p} \Delta_p(e')$ be the edge that minimizes the worst-case impact of p among all edges of the reconfigurable network. If there exists a link $e_\ell = (\text{src}(p), \text{dest}(p))$ in E_ℓ, and if the weighted latency of sending p through e_ℓ is smaller than the worst-case impact of p assigned to e (i.e, $w_p \cdot d_\ell(p) \leq \Delta_p(e)$), packet p is assigned to edge e_ℓ; otherwise, packet p is assigned to edge e.

If packet p is not transmitted via a direct source-destination link, the edge e_p, which will eventually transmit packet p is fixed. In the remaining part of the paper we will use H_p and L_p to denote the corresponding terms for the edge e_p, that is, $H_p(e_p)$ and $L_p(e_p)$, respectively. The pseudocode of dispatcher is shown in Algorithm 1.


Algorithm 1 Dispatcher

---

for all pending packets p in order of their arrival **do**
    $e \leftarrow \arg\min_{e' \in E_p} \Delta_p(e')$
    $r \leftarrow \Delta_p(e)$ {worst-case impact of sending p via reconfigurable network}
    if $(\text{src}(p), \text{dest}(p)) \in E_\ell$ **then**
        $f \leftarrow w_p \cdot d_\ell(p)$ {cost of sending p via direct link}
    end if
    if $r < f$ or there is no link that can route p **then**
        p is assigned to edge e
    else
        p uses direct link
    end if
end for

4.3 Scheduler

We describe how at time τ packets released until time τ are transmitted through the reconfigurable network. To this end, we construct the set M_τ of chunks that will be transmitted in the interval $[\tau, \tau+1)$. The set of edges used by chunks from M_τ forms a stable matching. We assume that each packet p is already assigned to an edge e_p and split into chunks such that a chunk c assigned to edge e can be transmitted in a single step, i.e., $\text{size}(c) = 1/d(e)$. For a chunk c, by $p(c)$ we denote the packet whose part is c. The weight of c is then $w_c = w_p \cdot \text{size}(c)$.

The stable matching M_τ is constructed greedily. Initially, the set M_τ is empty. Then, for each pending chunk c, in order of decreasing weights and increasing arrival times, if both endpoints of e_c are free (i.e., selected chunks do not use edges adjacent to e_c), chunk c is selected for transmission and e_c becomes an element of M_τ. Otherwise, if at least one of the endpoints of e_c is already busy, then c
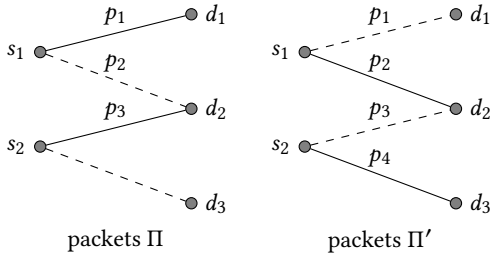
packet	path	weight	impact
p_1	$s_1 \to d_1$	1	$w_{p_1} = 1$
p_2	$s_1 \to d_2$	2	$w_{p_2} = 2$
p_3	$s_2 \to d_2$	3	$w_{p_2} + w_{p_3} = 5$

packet	path	weight	impact
p_1	$s_1 \to d_1$	1	$w_{p_1} = 1$
p_2	$s_1 \to d_2$	2	$w_{p_1} + w_{p_2} = 3$
p_3	$s_2 \to d_2$	3	$w_{p_3} = 3$
p_4	$s_2 \to d_3$	4	$w_{p_3} + w_{p_4} = 7$

packets Π packets Π'

Figure 2: The figure shows a graph and two inputs (sets of packets). In the graph, for each source, there is exactly one transmitter attached to it and for each destination, there is exactly one receiver (the transmitters and receivers are omitted on the picture). The label above each edge is the (only) packet that might use this edge. Solid edges mark the stable matching (assuming the weight of an edge is the weight of the packet it can transmit) that would be transmitted if no more packets arrived. Upon arrival of a new packet p_4, the stable matching changes. As a result, packet p_2 is not blocked by p_3 and p_2 blocks p_1. The columns of both tables contain the packet identifier, the only path (edge) that can transmit the packet, the weight of the packet, and its impact. Upon arrival of a new packet p_4, the impact of packet p_2 increases and the impact of packet p_3 decreases.

is not transmitted. Observe that, due to our ordering according to decreasing weights, c' blocks c as $w_{c'} \geq w_c$. When the algorithm processes all chunks, the selected chunks are transmitted.

The pseudocode of the scheduler is presented in Algorithm 2.

Algorithm 2 Scheduler (Greedy stable matching)

---

for all time τ **do**
    $C_\tau \leftarrow \emptyset$ {The set of transmitted chunks}
    $M_\tau \leftarrow \emptyset$ {Stable matching w.r.t. chunks' weights}
    for all pending chunks c **do**
        if e_c is not adjacent to M_τ **then**
            Add c to C_τ
            Add e_c to M_τ
        end if
    end for
    Transmit chunks of C_τ in the round τ
end for

---

5 COMPETITIVE ANALYSIS

In this section we prove that algorithm Alg is $O(\varepsilon^{-1})$-competitive given a $(2 + \varepsilon)$ speedup. In the analysis, instead of empowering the algorithm, we limit the capabilities of the optimum algorithm: For $\varepsilon \geq 0$ its transmission can take time at most $1/(2 + \varepsilon)$ in a single step. Although the schedule of the algorithm is non-migratory (a packet is assigned to and transmitted via exactly one path in G), the result holds against an optimal solution that is preemptive and migratory.

5.1 Linear Program Relaxation

For our analysis we will rely on the formulation of primal and dual linear programs (containing all feasible solutions transmitting packets with speed $1/(2 + \varepsilon)$), to which we will be able to charge the costs of our online algorithm. This will eventually allow us to upper bound the competitive ratio (how far our algorithm is off from a best possible offline solution).

For packet $p \in \Pi$, edge $e = (t, r) \in E_p$ and time $\tau \geq a_p$ we introduce a variable $x_{p,e,\tau}$ interpreted as a fraction of packet p that is sent through the path $\mathrm{src}(p) - t - r - \mathrm{dest}(p)$ at time τ. Note that, this transmission takes $\hat{d}(e) = d(\mathrm{src}(t), t) + d(e) + d(r, \mathrm{dest}(r))$ time steps, and incurs a weighted latency of $w_p \cdot x_{p,e,\tau} \cdot (\tau + \hat{d}(e) - a_p)$. To model fixed direct links between sources and destinations, for each packet p we introduce variable y_p, which is interpreted as the amount sent through this direct connection. The weighted latency of this transmission is then $w_p \cdot d_\ell(p) \cdot y_p$.

For time τ, let $P(\tau)$ be the set of packets released earlier than τ. The linear program $\mathcal{P}$ shown in Figure 3 contains the set of feasible solutions (in particular, the optimal solution in a resource augmentation model). Therefore, the objective value of the optimal solution to $\mathcal{P}$ is a lower bound on the weighted (integral) flow time of the optimal schedule.

The first and the second sets of constraints force transmitting all packets either through the reconfigurable or the fixed network (if the latter is possible). The remaining two sets of constraints are satisfied when the sets of edges corresponding to transmitted packets form a matching in the reconfigurable network and the transmission time is limited to $1/(2 + \varepsilon)$ (recall that sending amount s through edge e takes time $s \cdot d(e)$). This is satisfied by the optimal solution and possibly assignments that are not feasible schedules (e.g., the scaled average of two matchings). However, we require only that the minimal objective value of this linear program is a lower bound on the optimal transmission cost.

Note that the number of variables and constraints is potentially infinite, but it is sufficient to consider only τ smaller than $\max_{p \in \Pi} a_p + |\Pi| \cdot \max_{e \in E} \hat{d}(e)$. This is because if there is any pending packet, then any (reasonable) algorithm transmits at least one of them and transmitting packets one by one takes time at most $|\Pi| \cdot \max_{e \in E} \hat{d}(e)$.

We also note that our algorithm will not give feasible solution for the primal LP, as we are relying on resource augmentation. Hence, we next formulate the corresponding dual program. The program $\mathcal{D}$ dual to $\mathcal{P}$ is shown in Figure 4.

$$\text{min.} \quad \sum_{p \in \Pi} \sum_{e \in E_p} \sum_{\tau \geq a_p} w_p \cdot x_{p,e,\tau} \cdot \left(\tau + \hat{d}(e) - a_p \right) + \sum_{p \in \Pi_\ell} w_p \cdot y_p \cdot d_\ell(p)$$

$$\text{s.t.} \quad \sum_{e \in E_p} \sum_{\tau \geq a_p} x_{p,e,\tau} + y_p \geq 1 \qquad \qquad \text{for all } p \in \Pi_\ell$$

$$\sum_{e \in E_p} \sum_{\tau \geq a_p} x_{p,e,\tau} \geq 1 \qquad \qquad \text{for all } p \in \Pi \setminus \Pi_\ell$$

$$\sum_{r \in R(t)} \sum_{p \in P(\tau):(t,r) \in E_p} d(t,r) \cdot x_{p,(t,r),\tau} \leq \frac{1}{2+\varepsilon} \qquad \qquad \text{for all } \tau, \, t \in T$$

$$\sum_{t \in T(r)} \sum_{p \in P(\tau):(t,r) \in E_p} d(t,r) \cdot x_{p,(t,r),\tau} \leq \frac{1}{2+\varepsilon} \qquad \qquad \text{for all } \tau, \, r \in R$$

Figure 3: Linear program $\mathcal{P}$ containing all feasible solutions with reduced transmission speed in the reconfigurable network. We omit nonnegativity constraints of all variables.

$$\text{max.} \quad \sum_{p \in \Pi} \alpha_p - \frac{1}{2+\varepsilon} \left(\sum_{t \in T} \sum_{\tau} \beta_{t,\tau} + \sum_{r \in R} \sum_{\tau} \beta_{r,\tau} \right)$$

$$\text{s.t.} \quad \alpha_p - d(e) \cdot (\beta_{t,\tau} + \beta_{r,\tau}) \leq w_p \cdot (\tau + \hat{d}(e) - a_p) \quad \text{for all } p \in \Pi, e = (t,r) \in E_p, \tau \geq a_p$$

$$\alpha_p \leq w_p \cdot d_\ell(p) \qquad \qquad \text{for all } p \in \Pi_\ell$$

Figure 4: Linear program $\mathcal{D}$ dual to $\mathcal{P}$. We omit nonnegativity constraints of all variables.

5.2 Almost-feasible Solution to Dual Program

From the weak duality, the value of a feasible solution to program $\mathcal{D}$ is a lower bound on the cost of the optimal solution to $\mathcal{P}$. We utilize this to prove that algorithm Alg is competitive. For the sake of analysis, we construct an almost-feasible dual solution whose cost can be related to the cost of Alg.

In the solution to $\mathcal{D}$ used throughout the analysis, for packet $p \in \Pi$ we set the value of α_p to the worst-case impact of p estimated at the arrival of this packet: If packet p was transmitted through the direct source-destination link, we set $\alpha_p = d_\ell(p)$. Otherwise, if packet p was sent through edge e_p in the reconfigurable network, the value of α_p is set to $\alpha_p = \Delta_p(e_p)$.

For time τ, transmitter $t \in T$ and receiver $r \in R$, let $C_{t,\tau}$ and $C_{r,\tau}$ be the sets of all chunks assigned to use the edge adjacent to t and r, respectively, that have not reached their destination until time τ. We set $\beta_{t,\tau} = w(C_{t,\tau})$ and $\beta_{r,\tau} = w(C_{r,\tau})$ to the total weight of chunks from corresponding sets C.

By $\hat{\mathcal{D}}$ we denote the assignment of dual variables α and β defined in this section. The assignment $\hat{\mathcal{D}}$ is an almost-feasible solution to $\mathcal{D}$. In Section 5.4 we show that halving each variable α and β yields a feasible solution. Before that, we start from relating the cost of Alg to the objective value of $\hat{\mathcal{D}}$.

5.3 Alg-to-Dual Ratio

The goal of this section is to relate the cost of Alg to the objective value of the dual assignment $\hat{\mathcal{D}}$. First, we show that the latency accumulated in β variables is at most twice the cost of the algorithm. Second, we define a cost charging scheme from the weighted latency of Alg to variables α. Third, by jointly considering these two

relations, we get that the cost of the algorithm is at most $(2+\varepsilon)/\varepsilon$ times the value of the dual assignment $\hat{\mathcal{D}}$.

Lemma 5.1. $Alg \geq \sum_{t \in T} \sum_{\tau} \beta_{t,\tau} = \sum_{r \in R} \sum_{\tau} \beta_{r,\tau}$.

Proof. The packets that use direct connections incur (positive) latency, but are not counted by the β variables. Therefore, it is sufficient to prove that the sum of the transmitters' β variables as well as the sum of the receivers' β variables equals the weighted latency of packets (chunks) that are transmitted via the reconfigurable network.

Fix a chunk c of packet p that was transmitted via reconfigurable network. Let $A(c)$ denote the period when c was active, that is, all the times τ from a_p until the time c reaches $\text{dest}(p)$. For any time τ in $A(c)$, chunk c incurs cost w_c. Recall that chunk c is assigned to exactly one edge $e_{p(c)}$ from $E_{p(c)}$ and thus to exactly one transmitter t and receiver r (the endpoints of edge e). Hence, for each $\tau \in A(c)$, it holds that $c \in C_{t,\tau}$ and $c \in C_{r,\tau}$, so for each time $\tau \in A(c)$, w_c is counted towards $\beta_{t,\tau}$ and $\beta_{r,\tau}$. These are the only β variables that count the latency of c at transmission step τ. The lemma follows by summing over all packets and their chunks in the input. □

Alg-to-α's charging scheme. In this section we charge the cost of the algorithm (the weighted latencies) to packets. The goal is to show, that each packet is charged at most the value of the corresponding α variable. This will let us relate the cost of the algorithm to the objective value of $\hat{\mathcal{D}}$.

Fix packet p. If p was transmitted via the fixed network, then we simply charge its total latency to p itself. Otherwise, p was split into several chunks. The chunks of p might delay each other as

the edge e_p can transmit just one of them in a single transmission. Therefore, we will focus on a single chunk of packet p and charge its latency to other packets or packet p.

Let $c \in C_p(e_p)$ be a chunk of p. It incurs weighted latency of w_c for each time $\tau \in A(c)$ before it reached dest(p). When c is being transmitted through any edge of the graph, we simply charge w_c to p. It remains to charge the latency of c when the chunk was waiting in the transmitter's queue. For each such time τ, there exists another chunk that blocked c. Let B be the set of all chunks that blocked c.

For each chunk $c' \in B$, if c' and c are parts of the same packet p, the latency w_c is charged, again, to p. Otherwise, we charge w_c to p or $p' = p(c)$ depending on which of these two packets arrived later. If $a_p < a_{p'}$, the latency of w_c is charged to p'. Note that c' is heavier than c and hence $c \in L_{p'}$. If $a_{p'} < a_p$ we charge w_c to packet p. If this is the case, it holds that $c' \in H_p$.

In the following lemma we bound the charges received by each packet.

LEMMA 5.2. *For packet p, let c_p be the weighted latency charged to p. Then, it holds that $c_p \le \alpha_p$.*

PROOF. Fix packet p. If p is sent via the fixed network, the only charge it receives is $w_p \cdot d_\ell(p) = \alpha_p$. Otherwise, the charges received by p are threefold:

- First, for each of its chunks, packet p receives a charge of w_c for every time τ when c was not blocked (i.e., transmitted via any edge) or blocked by another chunk of p. The i-th (for $i \in \{1, 2, \ldots, d(e_p)\}$) delivered chunk of p charges $w_c \cdot (d(\text{src}(p), t) + i + d(r, \text{dest}(p)))$. In total the latency charged in this case is equal to

$$\sum_{i=1}^{d(e_p)} \frac{w_p}{d(e_p)} \cdot (d(\text{src}(p), t) + i + d(r, \text{dest}(p)))$$
$$= w_p \cdot \left(d(\text{src}(p), t) + \frac{d(e_p) + 1}{2} + d(r, \text{dest}(p)) \right).$$

- Second, when c blocked chunk c' of a packet that arrived earlier than p, it receives charge $w_{c'}$. This charge can be received only from packets in set L_p. Note that c' is delayed by all $d(e_p)$ chunks of p.
- Third, when c is blocked by some other chunk c' of a packet that arrived earlier than p, packet p receives a charge of w_c. In this case, $c' \in H_p$.

Combining all three cases, we obtain that the latency charged to p is at most

$$c_p \le w_p \cdot \left(d(\text{src}(p), t) + \frac{d(e_p) + 1}{2} + d(r, \text{dest}(p)) \right) +$$
$$w_p \cdot |H_p| + d(e) \cdot w(L_p) = \alpha_p$$

$\square$

In the next lemma we show that for any $\varepsilon > 0$, the fractional flow time of the algorithm is bounded by $O(\varepsilon^{-1})$ times the objective value of assignment $\hat{\mathcal{D}}$.

LEMMA 5.3. *For any $\varepsilon > 0$ it holds that $A\text{LG} \le (2 + \varepsilon)/\varepsilon \cdot \text{val}(\hat{\mathcal{D}})$.*

PROOF. Fix $\varepsilon > 0$. If we sum the guarantees from Lemma 5.2 over all packets $p \in \Pi$, we obtain $A\text{LG} \le \sum_p \alpha_p$. This combined with Lemma 5.1 immediately yields the lemma, as

$$\text{val}(\hat{\mathcal{D}}) = \sum_{p \in \Pi} \alpha_p - \frac{1}{2 + \varepsilon} \cdot \left(\sum_{t \in T} \sum_\tau \beta_{t,\tau} + \sum_{r \in R} \sum_\tau \beta_{r,\tau} \right)$$
$$\ge A\text{LG} - \frac{2}{2 + \varepsilon} \cdot A\text{LG} = \frac{\varepsilon}{2 + \varepsilon} \cdot A\text{LG}.$$

$\square$

5.4 Dual-to-OPT Ratio

By weak duality, the value of any feasible solution to $\mathcal{D}$ is a lower bound on the cost of OPT. Our assignment of α and β variables does not necessarily constitute a feasible solution to $\mathcal{D}$, that is, some constraints might be violated. However, these constraints are "almost feasible", i.e., the solution obtained by halving each variable is feasible. Therefore, the value of our dual solution is at most twice the optimum, which together with the results of the previous section, will lead to the bound on the competitive ratio of ALG.

The following lemma shows that the constraints in $\mathcal{D}$ corresponding to packet p and edge e are violated by a factor of 2, if instead of α_p we consider the precomputed impact of p assigned to edge e.

LEMMA 5.4. *For packet p, edge $e = (t, r) \in E_p$ and time $\tau \ge a_p$ it holds that*

$$\Delta_p(e) - d(e) \cdot (\beta_{t,\tau} + \beta_{r,\tau}) \le 2 \cdot w_p \cdot \left(t + \hat{d}(e) - a_p \right).$$

PROOF. Fix packet p, edge $e = (t, r) \in E_p$ and time $\tau \ge a_p$. We start with proving that

$$L := w_p \cdot |H_p(e)| + d(e) \cdot w(L_p(e)) \tag{1}$$
$$\le d(e) \cdot (\beta_{t,\tau} + \beta_{r,\tau}) + 2 \cdot w_p \cdot (\tau - a_p).$$

To this end observe that the contribution of a single chunk $c \in A_p(e)$ (i.e., chunk that uses edge adjacent to e) towards L is at most

$$\min (d(e) \cdot w_c, w_p). \tag{2}$$

Two nontrivial relations follow directly from the definitions of sets $H_p(e)$ and $L_p(e)$. If $c \in H_p(e)$, then $w_p \le w_c \cdot d(e)$ and when $c \in L_p(e)$, then $d(e) \cdot w_c \le w_p$.

Let P be the set of those chunks in $A_p(e)$ that have not reached their destination by time τ. By (2), the contribution of chunks from P towards L is at most $d(e) \cdot w(P) \le d(e) \cdot (\beta_{t,\tau} + \beta_{r,\tau})$.

It remains to bound the contribution of chunks in $Q = A_p(e) \setminus P$. Again, by (2), chunk $c \in Q$ contributes at most w_p towards L. The proof of (1) is concluded by observing that the set Q contains at most $2 \cdot (\tau - a_p)$ chunks as endpoints of edge e transmit at most one chunk in each transmission step.

The lemma follows by combining Inequality (1), the fact that $d(e) \geq 1$, and the definition of $\Delta_p(e)$:

$$\Delta_p(e) = w_p \cdot \left(d(t) + \frac{d(e)+1}{2} + d(r) \right)$$
$$+ w_p \cdot |H_p(e)| + d(e) \cdot w(L_p(e))$$
$$\leq w_p \cdot \hat{d}(e) + d(e) \cdot (\beta_{t,\tau} + \beta_{r,\tau}) + 2 \cdot w_p \cdot (\tau - a_p)$$
$$< 2 \cdot w_p \cdot \left(\tau + \hat{d}(e) - a_p \right) + d(e) \cdot (\beta_{t,\tau} + \beta_{r,\tau}).$$

□

In the next lemma we leverage weak duality to relate the value of our dual solution to the value of the optimum.

LEMMA 5.5. *The objective value of assignment $\hat{\mathcal{D}}$ is at most twice the cost of the optimal solution, i.e.,* $\mathrm{val}(\hat{\mathcal{D}}) \leq 2 \cdot \mathrm{Opt}$.

PROOF. We prove that the solution to $\mathcal{D}$ obtained by halving each variable α and β in $\hat{\mathcal{D}}$ is feasible. To this end we show that $\hat{\mathcal{D}}$ violates constraints by a factor at most 2.

First, for packet $p \in \Pi_\ell$, the constraint corresponding to primal variable y_p (i.e., routing through the fixed network) is violated by a factor of two as α_p minimizes the impact of routing p through any path, in particular, the direct source-destination link and hence $\alpha_p \leq w_p \cdot d_\ell(p)$.

Second, fix the dual constraint corresponding to primal variable $x_{p,e,\tau}$ for packet $p \in \Pi$, edge $e = (t,r) \in E_p$ and time $\tau \geq a_p$.

Applying the definition of α_p and Lemma 5.4 to the left-hand side of the dual constraint we obtain:

$$\alpha_p - d(e) \cdot (\beta_{t,\tau} + \beta_{r,\tau}) \leq \Delta_p(e) - d(e) \cdot (\beta_{t,\tau} + \beta_{r,\tau})$$
$$\leq 2 \cdot w_p \cdot \left(\tau + \hat{d}(e) - a_p \right).$$

Therefore, the dual solution created by halving each variable of $\hat{\mathcal{D}}$ is feasible. The lemma follows from weak duality. □

By combining Lemma 5.3 and Lemma 5.5, we bound the fractional flow time of ALG.

THEOREM 5.6. *For any input, and $\varepsilon > 0$, if ALG works $2 + \varepsilon$ times faster than the optimal offline algorithm OPT, the fractional flow time of ALG is bounded by* $\mathrm{ALG} \leq 2 \cdot (2/\varepsilon + 1) \cdot \mathrm{OPT}$.

Finally, we obtain the competitive ratio of ALG.

THEOREM 5.7. *ALG is $O(\varepsilon^{-2})$-competitive in a resource augmentation model with speedup $(2 + \varepsilon)$.*

6 RELATED WORK

Reconfigurable optical topologies [7, 8, 11, 13, 28–33] have recently received much attention in the literature as an alternative to traditional static datacenter topologies [2, 3, 6, 34–37]. It has been demonstrated that already demand-oblivious reconfigurable topologies can deliver unprecedented bandwidth efficiency [7, 8, 13]. By additionally exploiting the typical skewed and bursty structure of traffic workloads [17–19, 38–41], demand-aware reconfigurable topologies can be further optimized, e.g., toward elephant flows. To this end, existing demand-aware networks are based on traffic matrix predictions [42–44] or even support per-flow or "per-packet"

reconfigurations [9–11, 14, 31, 45, 46]. Our focus on this paper is on the latter, and we consider fine-grained scheduling algorithms.

While most existing systems are mainly evaluated empirically, some also come with formal performance guarantees. However, to the best of our knowledge, besides some notable exceptions which however focus on single-tier architectures [15, 22, 47–49], little is known about the competitive ratio which can be achieved by online packet scheduling algorithms in this context. In particular, our paper is motivated by the multi-tier ProjecToR architecture [11], to which our analysis also applies.

Our model and result generalizes existing work on competitive switch scheduling [20, 21]: In classic switch scheduling, packets arriving at a switch need to be moved from the input buffer to the output buffer, and in each time step, the input buffers and all their output buffers must form a bipartite matching. A striking result of Chuang, Goel, McKeown, and Prabhakar showed that a switch using input/output queueing with a speed-up of 2 can simulate a switch that uses pure output queueing [21]. Our model generalizes this problem to a multi-tier problem, and we use a novel primal-dual charging scheme.

Venkatakrishnan et al. [15] initiated the study of an offline scheduling variant of the circuit switch scheduling problem, motivated by reconfigurable datacenters. They consider a setting in which demand matrix entries are small, and analyze a greedy algorithm achieving an (almost) tight approximation guarantee. In particular, their model allows to account for reconfiguration delays, which are not captured by traditional crossbar switch scheduling algorithms, e.g., relying on centralized Birkhoff-von-Neumann decomposition schedulers [50]. Schwartz et al. [49] recently presented online greedy algorithms for this problem, achieving a provable competitive ratio over time. This line of research however is technically fairly different from ours: the authors consider a maximization problem, aiming to maximize the total data transmission for a certain time window, whereas in our model, we aim to minimize completion times (i.e., *all* data needs to be transmitted). Furthermore, while we consider a multi-tier network (inspired by architectures such as [11]), these works assume a complete bipartite graph. Last but not least, we also support a simple form of hybrid architectures in our model. Besides these differences in the model, our model differs significantly from [15, 49] in terms of the used techniques. While prior work relies, among other, on randomized rounding, we study an online primal-dual approach.

In general, online primal-dual approaches have received much attention recently, after the seminal work by Buchbinder and Naor [27]. Unlike much prior work in this area, we however do not use the online primal-dual approach for the design of an algorithm, but only for its analysis. In this regard, our approach is related to scheduling literature by Anand et al. [23], and the interesting work by Dinitz and Moseley [22] on reconfigurable networks. We generalize the analysis of [23] to a more general graph where we can have conflicts at receives and transmitters.

The paper [22] is parallel work to ours, and both papers build upon the theoretical result of [11] (the analysis so far only appeared in a technical report). In [22], a model is considered in which there is only one path between source and destination and each node can transmit a certain number of packets in one round. In this setting, the graph between transmitters and receivers is always

fully connected. In contrast, our setting allows for arbitrary graphs between transmitters and receivers as it is supported (and hence motivated) by existing optical technologies, both related to circuit switches [8, 13, 51] and free-space optics [11]. That said, the paper by Dinitz and Moseley and our paper use similar dual fitting techniques.

7 CONCLUSION

We presented a competitive scheduling algorithm for reconfigurable datacenter networks which generalizes classic switch scheduling problems and whose analysis relies on a dual-fitting approach. We understand our work as a first step and believe that it opens several interesting avenues for future research. In particular, it would be interesting to design fully decentralized algorithms and explore randomized scheduling algorithms. Our work also leaves open the question of optimality of bicriteria scheduling algorithms. More generally, it is important to generalize our model to also account for congestion in the fixed network.

REFERENCES

[1] 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," *ACM SIGCOMM computer communication review*, vol. 45, no. 4, pp. 183–197, 2015.

[2] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," in *ACM SIGCOMM Computer Communication Review*, vol. 38, pp. 63–74, ACM, 2008.

[3] 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," *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 4, pp. 63–74, 2009.

[4] M. Besta and T. Hoefler, "Slim fly: A cost effective low-diameter network topology," in *SC'14: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 348–359, IEEE, 2014.

[5] A. Singla, C.-Y. Hong, L. Popa, and P. B. Godfrey, "Jellyfish: Networking data centers randomly," in *Presented as part of the 9th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 12)*, pp. 225–238, 2012.

[6] S. Kassing, A. Valadarsky, G. Shahaf, M. Schapira, and A. Singla, "Beyond fat-trees without antennae, mirrors, and disco-balls," in *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, pp. 281–294, ACM, 2017.

[7] 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.

[8] 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 *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, pp. 267–280, ACM, 2017.

[9] 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. 41, no. 4, pp. 339–350, 2011.

[10] 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.

[11] 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 *Proceedings of the 2016 ACM SIGCOMM Conference*, pp. 216–229, ACM, 2016.

[12] C. Avin, K. Mondal, and S. Schmid, "Demand-aware network designs of bounded degree," *Distributed Computing*, pp. 1–15, 2017.

[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 *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication*, pp. 782–797, 2020.

[14] S. Schmid, C. Avin, C. Scheideler, M. Borokhovich, B. Haeupler, and Z. Lotker, "Splaynet: Towards locally self-adjusting networks," *IEEE/ACM Transactions on Networking (ToN)*, vol. 24, no. 3, pp. 1421–1433, 2016.

[15] S. Bojja Venkatakrishnan, M. Alizadeh, and P. Viswanath, "Costly circuits, submodular schedules and approximate carathéodory theorems," in *Proc. ACM SIGMETRICS*, pp. 75–88, 2016.

[16] M. Wang, Y. Cui, S. Xiao, X. Wang, D. Yang, K. Chen, and J. Zhu, "Neural network meets dcn: Traffic-driven topology adaptation with deep learning," *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, vol. 2, no. 2, p. 26, 2018.

[17] A. Roy, H. Zeng, J. Bagga, G. Porter, and A. C. Snoeren, "Inside the social network's (datacenter) network," in *ACM SIGCOMM Computer Communication Review*, vol. 45, pp. 123–137, ACM, 2015.

[18] T. Benson, A. Akella, and D. A. Maltz, "Network traffic characteristics of data centers in the wild," in *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, pp. 267–280, ACM, 2010.

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

[20] N. McKeown, "The islip scheduling algorithm for input-queued switches," *IEEE/ACM transactions on networking*, vol. 7, no. 2, pp. 188–201, 1999.

[21] S.-T. Chuang, A. Goel, N. McKeown, and B. Prabhakar, "Matching output queueing with a combined input/output-queued switch," *IEEE Journal on Selected Areas in Communications*, vol. 17, no. 6, pp. 1030–1039, 1999.

[22] M. Dinitz and B. Moseley, "Scheduling for weighted flow and completion times in reconfigurable networks," in *IEEE Conference on Computer Communications (INFOCOM)*, pp. 1043–1052, 2020.

[23] S. Anand, N. Garg, and A. Kumar, "Resource augmentation for weighted flow-time explained by dual fitting," in *Proceedings of the twenty-third annual ACM-SIAM symposium on Discrete Algorithms*, pp. 1228–1241, SIAM, 2012.

[24] J. S. Chadha, N. Garg, A. Kumar, and V. N. Muralidhara, "A competitive algorithm for minimizing weighted flow time on unrelatedmachines with speed augmentation," in *Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing*, STOC '09, p. 679–684, ACM, 2009.

[25] D. Gale and L. S. Shapley, "College admissions and the stability of marriage," *The American Mathematical Monthly*, vol. 69, no. 1, pp. 9–15, 1962.

[26] B. Kalyanasundaram and K. Pruhs, "Speed is as powerful as clairvoyance," *Journal of the ACM (JACM)*, vol. 47, no. 4, pp. 617–643, 2000.

[27] N. Buchbinder and J. Naor, *The design of competitive online algorithms via a primal-dual approach*. Now Publishers Inc, 2009.

[28] X. Zhou, Z. Zhang, Y. Zhu, Y. Li, S. Kumar, A. Vahdat, B. Y. Zhao, and H. Zheng, "Mirror mirror on the ceiling: Flexible wireless links for data centers," *Proc. ACM SIGCOMM Computer Communication Review (CCR)*, vol. 42, no. 4, pp. 443–454, 2012.

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

[30] 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, pp. 319–330, ACM, 2014.

[31] L. Chen, K. Chen, Z. Zhu, M. Yu, G. Porter, C. Qiao, and S. Zhong, "Enabling wide-spread communications on optical fabric with megaswitch," in *Proceedings of the 14th USENIX Conference on Networked Systems Design and Implementation*, NSDI'17, (USA), pp. 577–593, USENIX Association, 2017.

[32] Y. J. Liu, P. X. Gao, B. Wong, and S. Keshav, "Quartz: A new design element for low-latency dcns," *SIGCOMM Comput. Commun. Rev.*, vol. 44, pp. 283–294, Aug. 2014.

[33] 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.

[34] 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," *ACM SIGCOMM computer communication review*, vol. 45, no. 4, pp. 183–197, 2015.

[35] V. Liu, D. Halperin, A. Krishnamurthy, and T. Anderson, "F10: A fault-tolerant engineered network," in *Presented as part of the 10th {USENIX} Symposium on Networked Systems Design and Implementation (NSDI)*, pp. 399–412, 2013.

[36] H. Wu, G. Lu, D. Li, C. Guo, and Y. Zhang, "Mdcube: a high performance network structure for modular data center interconnection," in *Proceedings of the 5th international conference on Emerging networking experiments and technologies*, pp. 25–36, ACM, 2009.

[37] 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, pp. 17–17, 2012.

[38] S. Kandula, S. Sengupta, A. Greenberg, P. Patel, and R. Chaiken, "The nature of data center traffic: measurements & analysis," in *Proc. 9th ACM SIGCOMM conference on Internet measurement*, pp. 202–208, 2009.

[39] J. C. Mogul and L. Popa, "What we talk about when we talk about cloud network performance," *ACM SIGCOMM Computer Communication Review*, vol. 42, no. 5, pp. 44–48, 2012.

[40] S. Zou, X. Wen, K. Chen, S. Huang, Y. Chen, Y. Liu, Y. Xia, and C. Hu, "Virtualknotter: Online virtual machine shuffling for congestion resolving in virtualized datacenter," *Computer Networks*, vol. 67, pp. 141–153, 2014.

[41] Q. Zhang, V. Liu, H. Zeng, and A. Krishnamurthy, "High-resolution measurement of data center microbursts," in *Proceedings of the 2017 Internet Measurement Conference*, IMC '17, (New York, NY, USA), pp. 78–85, ACM, 2017.

[42] A. Singla, A. Singh, K. Ramachandran, L. Xu, and Y. Zhang, "Proteus: a topology malleable data center network," in *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, p. 8, ACM, 2010.

[43] 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*, vol. 22, pp. 498–511, April 2014.

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

[45] 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.

[46] 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," in *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM*, SIGCOMM '13, (New York, NY, USA), pp. 447–458, Association for Computing Machinery, 2013.

[47] M. Bienkowski, D. Fuchssteiner, J. Marcinkowski, and S. Schmid, "Online dynamic b-matching with applications to reconfigurable datacenter networks," in *38th International Symposium on Computer Performance, Modeling, Measurements and Evaluation (PERFORMANCE)*, 2014.

[48] C. Avin and S. Schmid, "Renets: Toward statically optimal self-adjusting networks," *arXiv preprint arXiv:1904.03263*, 2019.

[49] R. Schwartz, M. Singh, and S. Yazdanbod, "Online and offline greedy algorithms for routing with switching costs," *arXiv preprint arXiv:1905.02800*, 2019.

[50] N. McKeown, A. Mekkittikul, V. Anantharam, and J. Walrand, "Achieving 100% throughput in an input-queued switch," *IEEE Transactions on Communications*, vol. 47, no. 8, pp. 1260–1267, 1999.

[51] M. Y. Teh, Z. Wu, and K. Bergman, "Flexspander: augmenting expander networks in high-performance systems with optical bandwidth steering," *IEEE/OSA Journal of Optical Communications and Networking*, vol. 12, no. 4, pp. B44–B54, 2020.