Load-Optimization in Reconfigurable Networks: Algorithms and Complexity of Flow Routing*

Extended Abstract

Wenkai Dai Klaus-Tycho Foerster David Fuchssteiner Stefan Schmid
Faculty of Computer Science, University of Vienna, Austria
{wenkai.dai, klaus-tycho.foerster, david.alexander.fuchssteiner, stefan_schmid}@univie.ac.at

ABSTRACT

Emerging reconfigurable data centers introduce the unprecedented flexibility in how the physical layer can be programmed to adapt to current traffic demands. These reconfigurable topologies are commonly hybrid, consisting of static and reconfigurable links, enabled by e.g. an Optical Circuit Switch (OCS) connected to top-of-rack switches in Clos networks. Even though prior work has showcased the practical benefits of hybrid networks, several crucial performance aspects are not well understood.

In this paper, we study the algorithmic problem of how to jointly optimize topology and routing in reconfigurable data centers with a known traffic matrix, in order to optimize a most fundamental metric, maximum link load. We chart the corresponding algorithmic landscape by investigating both un-/splittable flows and (non-)segregated routing policies. We moreover prove that the problem is not submodular for all these routing policies, even in multi-layer trees, where a topological complexity classification of the problem reveals that already trees of depth two are intractable.

However, networks that can be abstracted by a single packet switch (e.g., nonblocking Fat-Tree topologies) can be optimized efficiently, and we present optimal polynomial-time algorithms accordingly. We complement our theoretical results with trace-driven simulation studies, where our algorithms can significantly improve the network load in comparison to the state of the art.

Keywords

reconfigurable networks, demand-awareness, matchings

1. INTRODUCTION

Data centers nowadays empower everyday life in aspects such as business, health, industry, but also science and social interactions. With the rise of related data-intensive workloads as generated by machine learning, artificial intelligence, and the distributed processing of big data in general, data center traffic is growing rapidly [29], evoking considerable interest in data center design [25].

Herein the emergence of a *programmable physical layer*, enabled by optical circuit switches [12, 33], free-space optics [8], or beamformed wireless connections [19], leads to intriguing new possibilities, as leveraging fully electrically packet switched networks "*is increasingly cost prohibitive and likely soon infeasible*" [24]. Extensive past work has already shown significant benefits of such reconfigurable data center networks [16], but the underlying complexity is not well understood [6]. For example, many works artificially restrict their flow routing policies to be segregated between programmable and static network parts, aiming to place elephant flows on reconfigurable links [14].

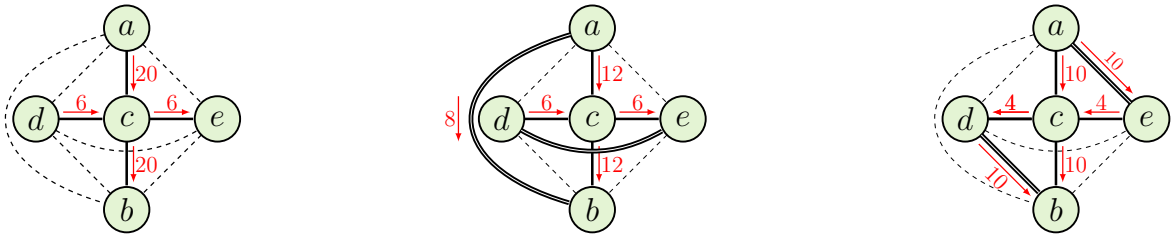
Whereas some general algorithmic results exist w.r.t. latency [7, 13, 15, 18, 21] or specific traffic patterns [32], complexity questions of network design for the objective of load-optimization are mostly uncharted. An exception is the work by Yang et al. [36], which focuses on the hardness induced by wireless interference, and the work by Zheng et al. [38], who study intractability on general graphs.

At the same time, link load is a most central performance metric [19], and flow routing in traditional networks has been investigated for decades already [2]. We are thus motivated by the desire to take the first steps towards fundamentally understanding the network design problem for load-optimization in data center networks, jointly considering flow routing and (interference-free) physical layer programmability enabled by, e.g., optical circuit switches.

Contributions. This paper investigates network design of load-optimization in reconfigurable networks with optical circuit switches, leveraging the flexibility of emerging programmable physical layers for flow routing. We study the problem of multiple dimensions, from splittable to unsplittable flows, to fully flexible (non-segregated) versus segregated routing policies. We summarize the results as follows:

1. *Complexity:* We prove strong NP-hardness for these four routing policies on trees of any height not less than two. Moreover, all four problem settings are not submodular w.r.t. load-optimization, preventing common approximation techniques.
2. *Algorithms:* In turn, we give polynomial-time optimal algorithms for the hybrid switch model [31], which applies to non-blocking data center interconnects as, e.g., Fat-Trees. To this end we leverage a combination of subset matching results and topology-specific insights.
3. *Evaluations:* Our workload-driven simulations (using real-world and synthetic traces) show that our algorithms significantly improve on state of the art methods, decreasing the maximum load by 1.6× to 2.0×.

*Supported by the European Research Council (ERC), grant agreement No. 864228 (project AdjustNet).



(a) A hybrid network N (b) Reconfiguration $M_1 = \{\{a,b\},\{d,e\}\}$ (c) Reconfiguration $M_2 = \{\{d,b\},\{a,e\}\}$

Figure 1: Illustration of the load-optimization reconfiguration problem with a hybrid network N of five nodes $\{a,b,c,d,e\}$, four static links (drawn solid) and four reconfigurable links (dashed).

2. MODEL AND PRELIMINARIES

Network model. Let $N = (V, E, \mathcal{E}, C)$ be a *hybrid* network [23, 31] connecting the n nodes $V = \{v_1, \ldots, v_n\}$ (e.g., top-of-the-rack switches), using static links E (usually connected by electrical packet switches), meanwhile N also contains a set of reconfigurable (usually optical) links $\mathcal{E}$. The graph $(V, E \cup \mathcal{E})$ is a bidirected[1] graph such that two directions of each bidirected link $\{v_i, v_j\} \in E$ (resp. $\{v_i, v_j\} \in \mathcal{E}$) work as two (anti-parallel) directed links (v_i, v_j) and (v_j, v_i) respectively. Let $\overrightarrow{E}$ (resp. $\overrightarrow{\mathcal{E}}$) denote the set of corresponding directed links of E (resp. $\mathcal{E}$). Moreover, a function $C : \overrightarrow{E} \cup \overrightarrow{\mathcal{E}} \mapsto \mathbb{R}^+$ defines capacities for both directions of each bidirected link in $E \cup \mathcal{E}$. Note that $(V, E \cup \mathcal{E})$ can be a multi-graph, e.g., when a reconfigurable link in $\mathcal{E}$ also connects two endpoints of a static link in E.

Reconfigured Network. We say that a hybrid network N is *reconfigured* by a reconfigurable switch if some reconfigurable links $M \subseteq \mathcal{E}$, which must induce *a matching*,[2] are *configured* (implemented) to enhance the static network (V, E). The enhanced network $N(M)$ obtained by integrating the configured links M with the static links E of the hybrid network N is called a *reconfigured network*, i.e., $N(M) = (V, E \cup M)$. The set of configured (bidirected) links M, i.e., a matching, is called *a reconfiguration* of N.

Topologies. Our network model does not place a restriction on the underlying static topology and hence can be applied generally. Notwithstanding, for our hardness results in §3, already simple trees suffice, whereas our positive algorithmic results cover many data center topologies (§4).

Traffic Demands. The resulting network should serve a certain communication pattern, represented as a $|V| \times |V|$ communication matrix $D := (d_{ij})_{|V|\times|V|}$ (*demands*) with non-negative real-valued entries. An entry $d_{ij} \in \mathbb{R}^+$ represents the traffic load (frequency) or a demand from the node v_i to the node v_j. With a slight abuse of notation, let $D(v_i, v_j) = d_{ij}$ hereafter.

Routing Models. In a reconfigured network, for each demand, *unsplittable* routing requires its flows being sent along a single (directed) path, while *splittable* routing does not restrict the number of paths used for sending its flows; Segregated routing requires flows being transmitted on either static links or configured links, but non-segregated routing admits to sending flows on a path consisting of both configured and static links [12, 33]. Hence, we have *four different routing models*: *Unsplittable & Segregated (US)*, *Unsplittable & Non-segregated (UN)*, *Splittable & Segregated (SS)*, and *Splittable & Non-segregated (SN)*.

2.1 Load Preliminaries

Load Optimization. Given a reconfigured network $N(M)$ and demands D, let $f : \overrightarrow{E} \cup \overrightarrow{M} \mapsto \mathbb{R}^+$ be *a feasible flow* serving demands D in $N(M)$ under a routing model $\tau \in \{\text{US}, \text{UN}, \text{SS}, \text{SN}\}$. *The load of each directed link* $e \in \overrightarrow{E} \cup \overrightarrow{M}$ induced by the flow f is defined as $L(f(e)) := f(e)/C(e)$, and then *the maximum load* is defined as $L_{\max}(f) := \max\left\{L(f(e)) : e \in \overrightarrow{E} \cup \overrightarrow{M}\right\}$. There must be *an optimal flow* f_{opt} such that its maximum load is minimized for all feasible flows serving D on $N(M)$. Such an optimal flow is called *a load-optimization flow* in $N(M)$.[3]

Load-Optimization Reconfiguration Problem. Given a hybrid network N, a routing model $\tau \in \{\text{US}, \text{UN}, \text{SS}, \text{SN}\}$, and demands D, the *τ-load-optimization reconfiguration problem* is to find *an optimal reconfiguration* $M \subseteq \mathcal{E}$, to generate *an optimally reconfigured network* $N(M)$, where the maximum load of load-optimization flow is minimized.

Example. To illustrate the τ-load-optimization reconfiguration problem, we give a small example in Figure 1. Fig. 1a depicts the network before any reconfiguration. We consider the routing model $\tau = \text{SN}$ and a capacity function $\forall e \in \overrightarrow{E} \cup \overrightarrow{\mathcal{E}} : C(e) = 20$, with the six demands: $D(a,b) = 8$, $D(a,c) = 6$, $D(c,b) = 6$, $D(d,b) = 6$ and $D(a,e) = 6$. In Fig. 1a, each flow can only be routed along static links, creating a link load of $20/20 = 1$ on, e.g., (a,c) with three demands of size $8, 6, 6$ from a. To improve the maximum link load, a greedy algorithm adds the links $\{a,b\}$ in Fig. 1b. Now, the demand $D(a,b) = 8$ is routed directly, reducing the maximum load to just 0.6, which cannot be improved further without undoing this first greedy choice.

Notwithstanding, we can improve the maximum load further. To this end, we select $\{a,e\}$ and $\{d,b\}$ to reconfiguration, as shown in Fig. 1c. At first, this might seem counterintuitive, as $D(a,e)$ and $D(d,b)$ are only of size 6 each, leaving a load of 0.7 on the links (a,c) and (c,b). However, the demand $D(a,b) = 8$ can be routed indirectly, via the path $\{a,e,c,d,e\}$, yielding an optimal link load of 0.5.

[1] Symmetrical connectivity is the standard industry assumption for static cabling, however for reconfigurable links as well. Outside highly experimental hardware, off-the-shelf products use full-duplex connections and this model assumption is hence prevalent, even in FSO [8].

[2] In other words, no two links in M are adjacent or share an endpoint, enforced by hardware constraints in practice (exclusive connections between ports).

[3] We note that in other works with analogous definitions, load might also be denoted by utilization, and load-optimization by load-balancing.

3. COMPLEXITY

In this section we consider the underlying complexity of the load-optimization problem in reconfigurable networks. We begin with the investigation of NP-hardness, and then discuss the submodularity for our objective functions.

3.1 Intractability

We state the conclusions of NP-hardness w.r.t. these four routing models in Theorem 1, where the strong NP-hardness is shown by reductions from the strongly NP-hard 3-PARTITION problem [17], and the weak NP-hardness is obtained from the weakly NP-hard 2-PARTITION problem [17].

THEOREM 1. *The τ-load-optimization reconfiguration problem, where $\tau \in \{US, SS, UN, SN\}$, is strongly NP-hard when the static network (V, E) of the given hybrid network N is a tree of height $h \geq 2$. Furthermore, for $\tau = UN$, the τ-load-optimization reconfiguration problem is weakly NP-hard if (V, E) is a star (a tree of height $h = 1$).*

3.2 Non-Submodularity

The submodularity of objective functions plays an important role in approximating optimization problems [30], as by Venkatakrishnan et al. [31] for hybrid switch networks. However, their objective function does not consider load-balancing and therefore does not apply in our setting.

Objective functions. We investigate the submodularity of the objective function Φ of a τ-reconfiguration problem, where for any matching $M \in 2^{\mathcal{E}}$, $\Phi(M) \in \mathbb{R}^+$ denotes the maximum load of a load-optimization flow in the reconfigured network $N(M)$. Moreover, we are also interested in the submodularity of the objective function Ω. For a matching $M \in 2^{\mathcal{E}}$, $\Omega(M) \in \mathbb{R}^+$ denotes the difference between the maximum load of a load-optimization flow serving D in the static network (V, E) and the maximum load of a load-optimization flow in the reconfigured network $N(M)$.

THEOREM 2. *For $\tau \in \{US, UN, SS, SN\}$-reconfiguration problems, objective functions Φ and Ω are not submodular.*

4. HYBRID SWITCH NETWORKS

As shown, already simple tree networks of height $h \geq 2$ are intractable, and optimizations leveraging submodularity are impossible. This raises the question if we can obtain *optimal* and *polynomial-time* algorithms for meaningful settings.

4.1 Non-Blocking Data Center Topologies

Common data center topologies have trees of height 2 as subgraphs or minors and hence seem like bad candidates for efficient algorithms at first glance. However, already early designs adapted from telecommunications such as Clos [10] topologies have a so-called non-blocking property, which we can use to our advantage. An interconnecting topology $\mathcal{C}$ is non-blocking, if the servability of a flow from v_1 to v_2 via $\mathcal{C}$ only depends on the utilization of the links $(v_1, \mathcal{C})$ and $(\mathcal{C}, v_2)$: "*such an interconnect behaves like a crossbar switch*" [37]. Non-blocking interconnects have become popular data center topologies [3] in particular in the form of folded Clos networks or Fat-Trees [22], depicted in Fig. 2a: the actual topology inside the interconnect (marked in a blue rectangle) is immaterial and we only need to consider the links incident to the nodes.[4]

[4]We note that the non-blocking property can also be restricted to keep distributed routing schemes in mind, we refer to Yuan [37] for an in-depth discussion.

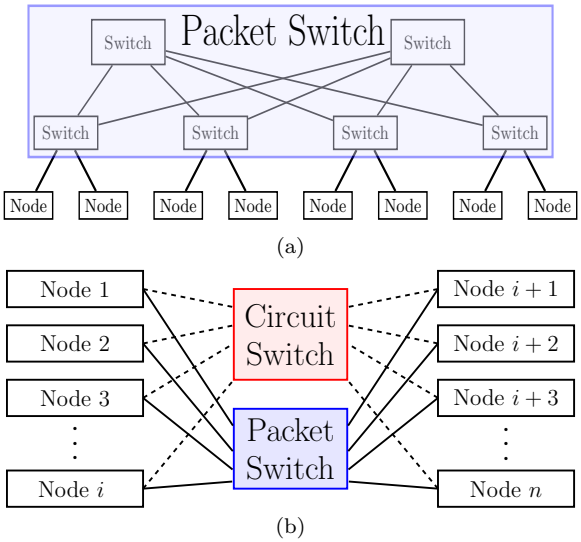


Figure 2: Illustration of a Fat-Tree network in 2a and a hybrid switch network in 2b, as in [29]. Due to the non-blocking property of network in 2a, we can abstract the interconnect enclosed by the blue box as a packet switch, as depicted in 2b. Additionally, a hybrid switch network also contains a reconfigurable circuit switch (e.g. an OCS) that provides a matching of the nodes, to be optimized for the demands D. Hence, augmenting a Fat-Tree by an OCS can be investigated from the viewpoint of hybrid switch networks.

As thus, for our purposes, we can abstract the data center interconnect $\mathcal{C}$ (which can be understood as a packet switch) by a single center node c, leaving our previous intractability considerations behind. We hence turn our attention to *hybrid switch networks* as considered by of Venkatakrishnan et al. [31], which are represented by a packet and a circuit switch connected to all nodes, see Fig. 2b.

Routing in hybrid switch networks is straightforward (only one path exists for each pair of nodes in the packet switched network), but the addition of a circuit switch adds a large degree of freedom: First, the number of possible matchings grows exponentially, and second, we have to decide for each flow which path to take as well. Notwithstanding, the special structure of hybrid switch networks allows us to solve reconfiguration and routing efficiently.

4.2 Optimization of Hybrid Switch Networks

We formalize our algorithm in Algorithm 1, which optimally solves the τ-reconfiguration problem on hybrid switch networks (HSNs). Due space constraints, we only provide an informal analysis for the following Theorem 3 in this paper.

THEOREM 3. *Algorithm 1 solves the $\{US, SS, SN\}$-reconfiguration problem optimally on HSNs in $O\left(n^4 \log n\right)$.*

Let a hybrid switch network $N = (V, E, \mathcal{E}, C)$ be reconfigured by an optimal matching $M \subseteq \mathcal{E}$, then each reconfigurable link $\{v_i, v_j\} \in M$ introduces a triangle G_{ij} that is a subgraph of $N(M)$ induced by three nodes $V_{ij} = \{c, v_i, v_j, \}$, where c is the central node (packet switch). Clearly, the center c is the unique cutting vertex to disconnect G_{ij} from $N(M)$. We can define local demands D_{ij} only for nodes of G_{ij} (lines 4-9 in Algorithm 1). Then, for each $\{v_i, v_j\} \in M$, we can compute a load-optimization flow f_{ij} serving D_{ij} in G_{ij} in a constant time, which also determines the lo-

cally maximum load $L_{\max}(f_{ij})$ for G_{ij}. Let f_{opt} be a load-optimization flow serving D in $N(M)$. It is easy to know that each edge e in G_{ij} has $f_{\text{opt}}(e) = f_{ij}(e)$. Let f_{old} be a flow serving D in the static network (V, E) of N before reconfiguration. We note that the minimized maximum load $L_{\max}(f_{\text{opt}})$ can occur at one edge inside a triangle or a static link not in any triangle. Thus, $L_{\max}(f_{\text{opt}})$ can be determined by a binary search for the locally maximum load of each triangle and each link load of f_{old}.

When $L_{\max}(f_{\text{opt}})$ is known, we know that a leaf node $v_i \in V$ must be included in a triangle in $N(M)$ if its static link $\{v_i, c\}$ has $f_{\text{old}}(\{v_i, c\}) > L_{\max}(f_{\text{opt}})$, otherwise the optimum $L_{\max}(f_{\text{opt}})$ cannot be satisfied. Therefore, we mark such nodes v_i as "red", which means that each node must be covered by a reconfigurable link. Any reconfigurable link $\{v_i, v_j\} \in \mathcal{E}$ would be selected into M if the locally maximum load of its triangle no larger than $L_{\max}(f_{\text{opt}})$. Now, the remaining task becomes solving the problem of Red-Target Matching (RTM) to find an optimal matching (reconfiguration) M, where we employ a maximum-weight matching.

Definition 1 (Red-Target Matching (RTM)). *Given a graph $G = (V, E)$ and a coloring $l : V \mapsto \{r, b\}$, to find a matching M of G such that each node $v \in V$ having the color $l(v) = r$ is contained in an edge of M.*

Improvement bounds. As we can see, in hybrid switch network N, for any two nodes v_i and v_j, there is a unique path before reconfiguration; and there are at most two edge-disjoint paths between v_i and v_j in any reconfigured network $N(M)$. Therefore, given a hybrid switch network N, demands D and a routing model $\tau \in \{\text{US, UN, SS, SN}\}$, by leveraging e.g. an OCS, the maximum load can at most be decreased by a factor of two after reconfiguration.

Competitivity of matching algorithms. We next investigate the theoretical performance of a maximum matching algorithm, as e.g. utilized in [33]. The heuristic idea based on a maximum matching is that for each reconfigurable link $\{u, v\} \in \mathcal{E}$, we send all flows of demands $D(u, v)$ and $D(v, u)$ on links (u, v) and (v, u) respectively, then to find a maximum matching to maximize total size of flows on a set of configured links M. As it turns out, such an optimization might yield nearly no benefit, even though an optimal algorithm could hit the theoretical lower bound of $\frac{1}{2}$.

Algorithm 1: Algorithm for Hybrid Switch Networks

Input : A hybrid switch network $N = (V, E, \mathcal{E}, C)$, where the static network (V, E) is a star with the center $c \in V$, demands D, and a routing model $\tau \in \{\text{US, SS, SN}\}$;

Output: A reconfiguration M and a load-optimization flow f_{opt} of $N(M)$ or "null";

let $f_{\text{old}} : \overrightarrow{E} \mapsto \mathbb{R}^+$ be a flow serving D on (V, E);
let $T := \left\{ L(f_{\text{old}}(e)) : e \in \overrightarrow{E} \right\}$ and $S = \emptyset$;
for *each reconfigurable link* $\{v_i, v_j\} \in \mathcal{E}$ **do**
  let G_{ij} be a triangle on nodes $V_{ij} = \{v_i, v_j, c\}$, where $\{c, v_i\}, \{c, v_j\} \in E$ and $\{v_i, v_j\} \in \mathcal{E}$;
  $\forall u \in \{v_i, v_j\}$ define local demands D_{ij} for G_{ij}:
    $D_{ij}(u, c) = \sum_{v \in V \setminus \{V_{ij}\}} D(u, v)$;
    $D_{ij}(c, u) = \sum_{v \in V \setminus \{V_{ij}\}} D(v, u)$;
    $D_{ij}(v_i, v_j) = D(v_i, v_j)$;
    $D_{ij}(v_j, v_i) = D(v_j, v_i)$;
  compute a load-optimization flow f_{ij} for D_{ij} and τ in the triangle G_{ij} in $O(1)$ time;
  $T = T \bigcup \{L_{\max}(f_{ij})\}$;
  $S = S \bigcup \{(\{v_i, v_j\}, L_{\max}(f_{ij}))\}$;
foreach *threshold* $t \in T$ **do**
  create a graph $G^\star = (V, E^\star)$, where $E^\star = \emptyset$;
  for *each static link* $e = \{v_i, c\} \in E$ **do**
    if *its load* $L(f_{\text{old}}(e)) > t$ **then**
      mark the node $v_i \in V$ "red" in $G^\star$;
  for *each pair* $(\{v_i, v_j\}, L_{\max}(f_{ij})) \in S$ **do**
    if *the value* $L_{\max}(f_{ij}) \leq t$ **then**
      add the edge $\{v_i, v_j\}$ into $G^\star$;
  if *a RTM matching M exists in $G^\star$* **then**
    obtain a load-optimization flow f_{final} on the reconfigured network $N(M)$ for D and τ;
    return a matching M and f_{opt};
return "null";

5. EVALUATIONS

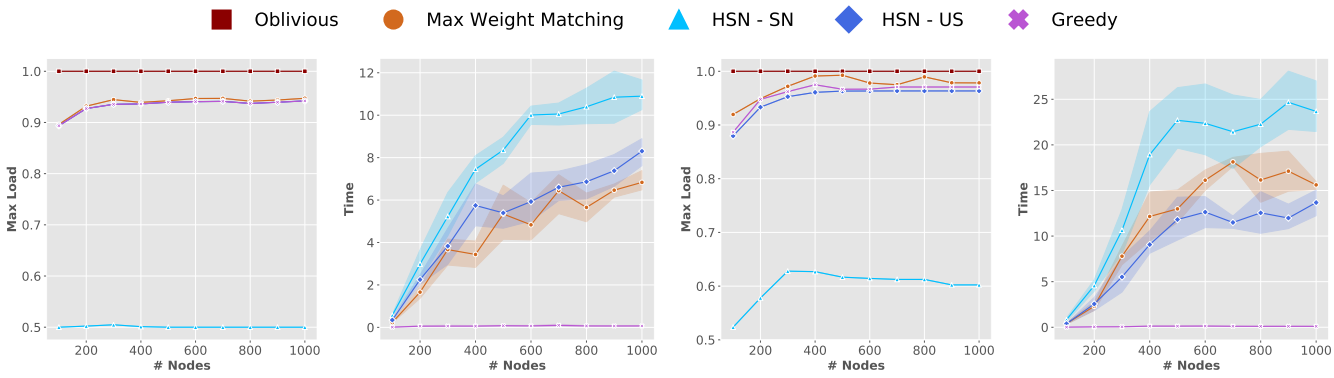
In order to study the performance of our algorithms under realistic workloads, we conducted extensive experiments with a simulator, which we will release together with this paper (as open source code). In particular, we benchmark our hybrid switch algorithms against several state of the art maximum matching and greedy baselines, considering a spectrum of packet traces on hybrid switch topologies as in Fig. 2. We first describe our methodology in §5.1 and then discuss our results in §5.2.

5.1 Methodology

Baselines. We consider the following baselines and implemented the corresponding algorithms for comparison. First, we compare our *h*ybrid *s*witch *n*etwork algorithms (denoted by *HSN-US/SN*) with a *Maximum Weight Matching* algorithm as a baseline, where routing occurs either on direct reconfigurable links or via the central packet switch. The matching algorithm is employed, e.g., by [12, 33] and is also optimal with respect to the average weighted path length [14] in this instance. Second, we also compare to a *Greedy* approach used by, e.g., Halperin et al. [19] and Zheng et al. [38]. For the link e that currently has the highest load, we check for the largest flow that can be rerouted on a direct connection, and offload it from the electrically switched network parts. This process is iterated until the load cannot be reduced further, where different links e can be chosen in each iteration. Lastly, we additionally plot the maximum load on the network before reconfiguration (labelled as *Oblivious*).

Traffic Workloads. Traffic traces in different networks and running different applications can differ significantly [5,9,18, 20,27]. We collected real-world and synthetic datasets from which we generate traffic matrices to evaluate and compare the performance of our algorithms. In particular, Datacenter traces from Facebook [27], HPC traces from the CESAR backbone [1], and synthetic pFabric traces [4].

Experimental Setup. All considered topologies, ranging from 40 to 3000 nodes, employ hybrid switch networks as



(a) Database Cluster: Max Load (b) Database Cluster: Runtime (c) Hadoop Cluster: Max Load (d) Hadoop Cluster: Runtime

Figure 3: Algorithmic comparison of the maximum load and runtime for different Facebook clusters

in Fig. 2b. We repeat each setting by running it 5 times and display the averaged results, normalizing the workload in the static topology. For the runtime, we plot the average values in seconds, but also display minimum and maximum values by shading. Our simulations were run on a machine with two Intel Xeons E5-2697V3 SR1XF with 2.6 GHz, 14 cores[5] each and a total of 128 GB RAM. The host machine was running Ubuntu 18.04.3 LTS. We implemented the algorithms in Python (3.7.3) leveraging the NetworkX library (2.3). For the implementation of the maximum matching algorithm we used the algorithm provided by NetworkX.

5.2 Results and Discussion

We discuss all results in text, but only plot the results for Facebook traces up to 1000 nodes, due to space constraints.

Potential for Load Optimization. All algorithms significantly improve the load over the Oblivious baseline and provide relatively stable benefits throughout all scenarios investigated. Among these algorithms, the HSN algorithms typically clearly outperform the others.

More specifically, for the database clusters, the reduction in the maximum load provided by the HSN-SN algorithm is almost a factor of two throughout the spectrum. For the Hadoop clusters, the performance of HSN-SN slightly decreases, but still achieves $\approx 60\%$ of the original Oblivious load up until a network size of 1000 and then stays stable at $\approx 70\%$ beyond. The three remaining algorithms (Greedy, Max. Weight Matching, and our HSN-US) achieve nearly identical values, with Greedy and HSN-US being slightly better. Above 1000 nodes, we observe that their capability to further reduce the load is quite restricted. Notwithstanding, they always perform significantly worse than HSN-SN, resulting in a comparatively load-increase of $\approx 60\%$.

Regarding the HPC traces, we can observe similar results as in the Database Cluster, in terms of maximum load reduction. Also for the pFabric traces, our HSN-US algorithm achieves a lower maximum load compared to the Greedy or Max. Weight Matching. Here, the variance is slightly higher than in the other experiments; this matches empirical observations on the complexity of the traces produced by these synthetic traces [5].

In regard to the maximum load reduction, we conclude that our HSN-SN algorithm is quite stable w.r.t. to the number of nodes in the network. In contrast to that, Max. Weight Matching and the Greedy algorithm asymptotically approach the maximum load of the unconfigured network.

Runtime Performance. The best runtime is generally achieved by the Greedy algorithm, due to its early termination when no link can be added anymore. Our experiments show that in the case of the Greedy algorithm, this is unfortunately happening very early on. Regarding the runtime of the the Max. Weight Matching, we want to emphasize that the algorithm is unaware of the underlying problem of reducing the maximum link load. Therefore, a lot of runtime is actually wasted without achieving any further load reduction. In comparison to Max. Weight Matching, our HSN-US has a similar runtime, while spending all of it searching for the best load reduction matching.

HSN-US is consistently faster than HSN-SN, and the latter features quite a high variance in runtime. Notwithstanding, HSN-US has the benefit of only routing along single paths, which can be beneficial for performance metrics beyond load [28, 36]. On the other hand, such issues can be alleviated with specialised multipath procotols [11, 26, 34].

Summary. While all algorithms provide load reductions, the extent of these optimizations and the required runtime differ significantly. Our results suggest that the load optimizations provided by HSN-US might prove beneficial over other segregated routing strategies, particularly because of its low runtime which is comparable to that of the Max. Weight Matching. We conclude that when considering both potential load reduction and runtime, HSN-SN provides a better tradeoff than HSN-US.

6. SELECTED RELATED WORK

While most related work on flow routing in data center networks focuses on non-reconfigurable topologies [25], reconfigurable topologies have recently received more attention. Algorithmic complexity aspects however are still not well understood, and we refer to [6, 16] for an overview. Performance guarantees for scheduling traffic matrices with specific skew were obtained by Venkatakrishnan et al. [32] leveraging submodularity, a condition that does not hold in our setting. Load-optimization in reconfigurable data centers was recently studied by Yang et al. [36], who investigated the impact of wireless interference on cross-layer optimization. We see our work as orthogonal, as we only consider

[5]However, each algorithm only utilized a single core.

inherently interference-free technologies.

Another interesting line of work is by Zheng et al. [38], who study how to enhance the design of Diamond, BCube and VL2 network topologies with small reconfigurable switches, inspired by Flat-Tree [35]. They target maximum link load as well, and present intractability results on general graphs, although these results do not transfer to specific data center topologies or trees, respectively. However, they do not analyze different routing models and their greedy algorithms not provide formal performance guarantees.

7. CONCLUSION

We investigated load minimization in reconfigurable hybrid networks, leveraging the flexibility of emerging programmable physical layers. We studied the underlying problem complexity, presented efficient optimizations for special cases, and complemented our theoretical results with simulations.

8. REFERENCES

[1] Characterization of the doe mini-apps. `portal.nersc.gov/project/CAL/doe-miniapps.htm`, 2016.

[2] R. K. Ahuja et al. *Network flows - theory, algorithms and applications*. Prentice Hall, 1993.

[3] M. Al-Fares et al. A scalable, commodity data center network architecture. In *SIGCOMM*, 2008.

[4] M. Alizadeh et al. pfabric: minimal near-optimal datacenter transport. In *SIGCOMM*, 2013.

[5] C. Avin et al. On the complexity of traffic traces and implications. In *SIGMETRICS*, 2020.

[6] C. Avin and S. Schmid. Toward demand-aware networking: a theory for self-adjusting networks. *Comput. Commun. Rev.*, 48(5):31–40, 2018.

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

[8] N. H. Azimi et al. Firefly: a reconfigurable wireless data center fabric using free-space optics. In *SIGCOMM*, 2014.

[9] T. Benson et al. Network traffic characteristics of data centers in the wild. In *Proc. IMC*, 2010.

[10] C. Clos. A study of non-blocking switching networks. *Bell System Technical Journal*, 32(2):406–424, 1953.

[11] A. A. Dixit et al. On the impact of packet spraying in data center networks. In *INFOCOM*, 2013.

[12] N. Farrington et al. Helios: a hybrid electrical/optical switch architecture for modular data centers. In *SIGCOMM*, 2010.

[13] T. Fenz et al. Efficient non-segregated routing for reconfigurable demand-aware networks. *Computer Communications*, 164:138 – 147, 2020.

[14] K.-T. Foerster et al. Characterizing the algorithmic complexity of reconfigurable data center architectures. In *ANCS*, 2018.

[15] K.-T. Foerster et al. On the complexity of non-segregated routing in reconfigurable data center architectures. *Comput. Comm. Rev.*, 49(2):2–8, 2019.

[16] K.-T. Foerster and S. Schmid. Survey of reconfigurable data center networks: Enablers, algorithms, complexity. *SIGACT News*, 50(2):62–79, 2019.

[17] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. 1979.

[18] M. Ghobadi et al. Projector: Agile reconfigurable data center interconnect. In *SIGCOMM*, 2016.

[19] D. Halperin et al. Augmenting data center networks with multi-gigabit wireless links. In *SIGCOMM*, 2011.

[20] S. Kandula et al. The nature of data center traffic: measurements & analysis. In *IMC*, 2009.

[21] J. Kulkarni, S. Schmid, and P. Schmidt. Scheduling opportunistic links in two-tiered reconfigurable datacenters. In *ArXiv Technical Report*, 2020.

[22] C. E. Leiserson. Fat-trees: Universal networks for hardware-efficient supercomputing. *IEEE Trans. Computers*, 34(10):892–901, 1985.

[23] H. Liu et al. Scheduling techniques for hybrid circuit/packet networks. In *CoNEXT*, 2015.

[24] W. M. Mellette et al. Expanding across time to deliver bandwidth efficiency and low latency. In *NSDI*, 2020.

[25] M. Noormohammadpour and C. S. Raghavendra. Datacenter traffic control: Understanding techniques and tradeoffs. *IEEE Communications Surveys and Tutorials*, 20(2):1492–1525, 2018.

[26] C. Raiciu et al. Improving datacenter performance and robustness with multipath TCP. In *SIGCOMM*, 2011.

[27] A. Roy et al. Inside the social network's (datacenter) network. In *Comput. Commun. Rev.*, volume 45, 2015.

[28] S. Sen, D. Shue, S. Ihm, and M. J. Freedman. Scalable, optimal flow routing in datacenters via local link balancing. In *CoNEXT*, 2013.

[29] A. Singh et al. Jupiter rising: a decade of clos topologies and centralized control in google's datacenter network. *Comm. ACM*, 59(9):88–97, 2016.

[30] V. V. Vazirani. *Approximation Algorithms*. Springer, Berlin, Heidelberg, 2001.

[31] S. B. Venkatakrishnan et al. Costly circuits, submodular schedules and approximate carathéodory theorems. In *SIGMETRICS*, 2016.

[32] S. B. Venkatakrishnan et al. Costly circuits, submodular schedules and approximate carathéodory theorems. *Queueing Syst.*, 88(3-4):311–347, 2018.

[33] G. Wang et al. c-through: part-time optics in data centers. In *SIGCOMM*, 2010.

[34] D. Wischik et al. Design, implementation and evaluation of congestion control for multipath TCP. In *NSDI*. USENIX Association, 2011.

[35] Y. Xia et al. A tale of two topologies: Exploring convertible data center network architectures with flat-tree. In *SIGCOMM*, 2017.

[36] Z. Yang et al. Achieving efficient routing in reconfigurable dcns. *ACM POMACS*, 3(3), 2019.

[37] X. Yuan. On nonblocking folded-clos networks in computer communication environments. In *IPDPS*, 2011.

[38] J. Zheng et al. Dynamic load balancing in hybrid switching data center networks with converters. In *ICPP*, 2019.