A Simple Congestion-Aware Algorithm for Load Balancing in Datacenter Networks

Mehrnoosh Shafiee, and Javad Ghaderi, Columbia University

***Abstract*—We study the problem of load balancing in datacenter networks, namely, assigning the end-to-end data flows among the available paths in order to efficiently balance the load in the network. The solutions used today rely typically on ECMP (Equal Cost Multi Path) mechanism which essentially attempts to balance the load in the network by hashing the flows to the available shortest paths. However, it is well known that ECMP performs poorly when there is asymmetry either in the network topology or the flow sizes, and thus there has been much interest recently in alternative mechanisms to address these shortcomings. In this paper, we consider a general network topology where each link has a cost which is a convex function of the link utilization. Flows among the various source-destination pairs are generated dynamically over time, each with a size (bandwidth requirement) and a duration. Once a flow is assigned to a path in the network, it consumes bandwidth equal to its size from all the links along its path for its duration. We propose a low-complexity congestion-aware algorithm that assigns the flows to the available paths in an *online fashion* and *without splitting*, and prove that it asymptotically minimizes the total network cost. Extensive simulation results are presented to verify the performance of our algorithm under a wide range of traffic conditions and under different datacenter architectures.**

I. Introduction

There has been a dramatic shift over the recent decades with search, storage, and computing moving into large-scale datacenters. Today's datacenters can contain thousands of servers and typically use a multi-tier switch network to provide connectivity among the servers. To maintain efficiency and quality of service, it is essential that the data flows among the servers are mapped to the available paths in the network properly in order to balance the load and minimize the cost (e.g., delay, congestion, etc.). For example when a large flow is routed poorly, collision with the other flows can cause some links to become congested, while other less utilized paths are available.

The datacenter networks rely on path multiplicity to provide scalability, flexibility, and cost efficiency. Consequently, there has been much research on flow scheduling algorithms that make better use of the path multiplicity (e.g., [1]–[5]) or designing new networks with better topological features (e.g., FatTree [1], VL2 [6], hypercube [7], hypergrid [8], random graphs such as JellyFish [9], etc.).

In this paper, we consider a general network topology where each link is associated with a cost which is a convex function of the link utilization (e.g., this could be a latency function). The network cost is defined as the sum of the link costs. Flows among the various source-destination pairs are generated dynamically over time where each flow is associated with a size and a duration. Once a flow is assigned to a path in the network, it consumes resource (bandwidth) equal to its size from all the links along its path for its duration. The main question that we ask is the following. Is it possible to design a low-complexity algorithm, that assigns the flows to the available paths in an *online fashion* and *without splitting*, so as to minimize the average network cost?

In general, multi flow routing in networks has been extensively studied from both networking systems and theoretical perspective, however the problem considered here has two key distinguishing objectives. First, it does not allow flow splitting because splitting the flow is undesirable due to TCP reordering effect [10]. Without splitting, many versions of multi flow routing in networks become hard combinatorial problems [11], [12]. Second, it allows dynamic routing because it considers the current utilization of links in the network when making the routing decisions unlike static solutions where the mapping of flows to the paths is fixed and requires the knowledge of the traffic matrix.

A. Related Work

Seminal solutions for flow scheduling (e.g. [6], [13]) rely on Equal Cost Multi Path (ECMP) load balancing which statically splits the traffic among available shortest paths (via flow hashing). However, it is well known [2]–[5], [14] that ECMP can balance load poorly since it may map large long-lived flows to the same path, thus causing significant load imbalance. Further, ECMP is suited for symmetric architectures such as FatTree and performs poorly in presence of asymmetry either due to link failures [15] or in recently proposed datacenter architectures [9].

There have been recent efforts to address the shortcomings of ECMP however they are mostly heuristics with no performance guarantees. The proposed algorithms range from centralized solutions (e.g., [2], [3]), where a centralized scheduler makes routing decisions based on global view of the network, to distributed solutions (e.g., [5], [16]) where routing decisions are made in a distributed manner by the switches. There are also host-based protocols based on Multi Path TCP (e.g., [4]) where the routing decisions are made by the end-host transport protocol rather than by the network operator. [17] investigates a more general problem based on a Gibbs sampling technique and proposes a plausible heuristic that requires re-routing and interruption of flows (which is operationally expensive). There are also algorithms that allow flow splitting and try to resolve the packet reordering effect in symmetric network topologies [5], [16], [18].

Software Defined Networking (SDN) has enabled network control with quicker and more flexible adaptation to changes in the network topology or the traffic pattern and can be leveraged to implement centralized or hybrid algorithms in datacenters [1], [19], [20].

B. Contribution

We propose and analyze a simple flow scheduling algorithm to minimize the average network cost (the sum of convex functions of link utilizations). Our main contributions can be summarized as below.

- We prove that our simple algorithm is asymptotically optimal in any network topology, in the sense that the performance ratio between our algorithm and the optimal cost approaches 1 as the mean number of flows in the system increases.
- Our algorithm does not rely on flow splitting, hence packets of the same flow will travel along the same path without reordering. Further, it does not require migration/rerouting of the flows or the knowledge of the traffic pattern.
- Our experimental results show that our algorithm in fact performs very well under a wide range of traffic conditions and datacenter network topologies.

For practical implementations, the weight construct in our algorithm can provide an approach to optimally accommodate dynamic variations in datacenter network traffic in centralized control platforms such as OpenFlow [19].

C. Notations

Given a sequence of random variables $\{X_n\}$, $X_n \Rightarrow X$ indicates convergence in distribution, and $X_n \to X$ indicates the almost sure convergence. Given a Markov process $\{X(t)\}$, $X(\infty)$ denotes a random variable whose distribution is the same as the steady-state distribution of $X(t)$ (when it exists). $\|\cdot\|$ is the Euclidian norm in $\mathbb{R}^n$. $d(x,S) = \min_{s\in S}\|s-x\|$ is the distance of x from the set S. 'u.o.c.' means uniformly over compact sets.

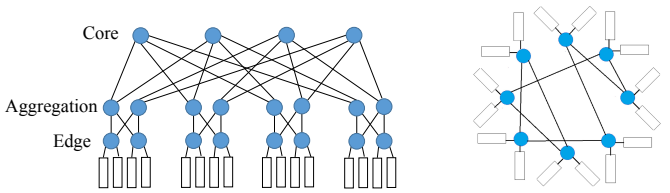
D. Organization

The remainder of the paper is organized as follows. In Section II, we introduce the datacenter network and traffic model. Our algorithm is presented in Section III. Section IV is devoted to the main results and performance analysis using fluid limits. Section V contains our simulation results to verify the performance of our algorithm under a wide range of traffic conditions and various datacenter architectures. The rigorous proofs of some of the results are provided in Section VI. Section VII contains our concluding remarks.

II. Model and Problem Statement

A. Datacenter Network Model

We consider a datacenter (DC) consisting of a set of servers (host machines) connected by a collection of switches and links. Depending on the DC network topology, all or a subset of the switches are directly connected to servers; for example,



(a) FatTree

(b) JellyFish (random graph)

Fig. 1: Two datacenter networks connecting 16 servers (rectangles) using 4-port switches (circles).

in FatTree (Figure 1a) only the edge (top-of-the-rack) switches are connected to servers, while in JellyFish (Figure 1b) all the switches have some ports connected to servers. Nevertheless, we can model any general DC network topology (FatTree, JellyFish, etc.) by a graph $G(V,E)$ where V is the set of switches and E is the set of communication links. A path between two switches is defined as a set of links that connects the switches and does not intersect itself. The paths between the same pair of source-destination switches may intersect with each other or with other paths in DC.

B. Traffic Model

Each server can generate a flow destined to some other server. We assume that each flow belongs to a set of flow types $\mathcal{J}$. A flow of type $j \in \mathcal{J}$ is a triple (a_j, d_j, s_j) where $a_j \in V$ is its source switch (i.e., the switch connected to the source server), $d_j \in V$ is its destination switch (i.e., the switch connected to its destination server), and s_j is its size (bandwidth requirement). Note that based on this definition, we only need to find the routing of flows in the switch network $G(V,E)$ since the routing from the source server to the source switch or from the destination switch to the destination server is trivial (follows the direct link from the server to the switch). Further, two switches can have *more than one* flow type with different sizes. We assume that type-j flows are generated according to a Poisson process with rate λ_j, and each flow remains in the system for an exponentially distributed amount of time with mean $1/\mu_j$ (we will see in Sections V that our algorithm actually performs very well under much more general arrival and service time processes.).

For any $j \in \mathcal{J}$, let R_j denote the set of all paths from a_j to d_j, then each type-j flow must be accommodated by using only one of the paths from R_j (i.e., the flow cannot be split among multiple paths). Assume that R_j is nonempty for each $j \in \mathcal{J}$. Define $Y_i^{(j)}(t)$ to be the number of type-j flows routed along the path $i \in R_j$ at time t. The network state is defined as

$$Y(t) = \left(Y_i^{(j)}(t); i \in R_j, j \in \mathcal{J}\right). \tag{1}$$

Under any online (Markov) flow scheduling algorithm, $\{Y(t)\}_{t\geq 0}$ evolves as a Markov chain. We also define $X^{(j)}(t) = \sum_{i\in R_j} Y_i^{(j)}(t)$ which is the total number of type-j

flows in the network at time t. Let $Z_l(t)$ be the total amount of traffic (congestion) over link $l \in E$. Based on our notations,

$$Z_l(t) = \sum_{j \in \mathcal{J}} \sum_{i: i \in R_j, l \in i} s_j Y_i^{(j)}(t), \tag{2}$$

(here $l \in i$ means that link l belongs to path i). We also define $\rho_j = \lambda_j/\mu_j$ which is the mean offered load by type-j flows.

C. Problem Formulation

For the purpose of load balancing, the network can attempt to optimize different objectives [21] such as minimizing the maximum link utilization in the network or minimizing the sum of link costs where each link cost is a convex function of the link utilization (e.g. this could be a link latency measure [22]). In this paper, we use the latter objective but by choosing proper cost functions, an optimal solution to the later objective can be used to also approximate the former objective as we see below.

We define $g(Z_l/C_l)$ to be the cost of link l with capacity C_l when its congestion is Z_l. Our goal is to find a flow scheduling algorithm that assigns each flow to a single path in the network so as to minimize the mean network cost in the long run, specifically,

$$\begin{aligned} &\text{minimize } \lim_{t\to\infty} \mathbb{E}\left[F(Y(t))\right] \\ &\text{subject to: serving each flow using one path,} \end{aligned} \tag{3}$$

where,

$$F(Y(t)) = \sum_{l \in E} g(Z_l(t)/C_l). \tag{4}$$

We consider polynomial cost functions of the form

$$g(x) = \frac{x^{1+\alpha}}{1+\alpha},\ \alpha > 0, \tag{5}$$

where $\alpha > 0$ is a constant. Thus g is increasing and strictly convex in x. As $\alpha \to \infty$, the optimal solution to (3) approaches the optimal solution of the optimization problem whose objective is to minimize the maximum link utilization in the network.

III. Algorithm Description

In this section, we describe our algorithm for flow assignment where each flow is assigned to one path in the network (no splitting) without interrupting/migrating the ongoing flows in the network. Recall that $Y(t) = (Y_i^{(j)}(t))$ is the network state, $Y_i^{(j)}(t)$ is the number of type-j flows on path $i \in R_j$, and $Z_l(t)$ is the total traffic on link l given by (2)

First, we define two forms of *link marginal cost* that measure the increase in the link cost if an arriving type-j flow at time t is routed using a path that uses link l.

Definition 1. *(Link marginal cost) For each link l and flow-type j, the link marginal cost is defined in either of the forms below.*

- *Integral form:*

$$\Delta_l^{(j)}(Y(t)) = g\Big(\frac{Z_l(t)+s_j}{C_l}\Big) - g\Big(\frac{Z_l(t)}{C_l}\Big). \tag{6}$$

- *Differential form:*

$$\delta_l^{(j)}(Y(t)) = \frac{s_j}{C_l} g'\big(\frac{Z_l(t)}{C_l}\big). \tag{7}$$

Based on the link marginal costs, we can characterize the increase in the network cost if an arriving type-j flow at time t is routed using path $i \in R_j$. Specifically, let $Y(t^+) = Y(t) + e_i^{(j)}$, where $e_i^{(j)}$ denotes a vector whose corresponding entity to path i and flow type j is one, and its other entities are zero. Then $F(Y(t))$ is the network cost before the type-j flow arrival, and $F(Y(t^+))$ is the network cost after assigning the type-j flow to path i. Then, it is easy to see that

$$\begin{aligned} F(Y(t^+)) - F(Y(t)) &= \sum_{l \in i} \Big[g\Big(\frac{Z_l(t)+s_j}{C_l}\Big) - g\Big(\frac{Z_l(t)}{C_l}\Big)\Big] \\ &= \sum_{l \in i} \Delta_l^{(j)}(Y(t)). \end{aligned} \tag{8}$$

Similarly, based on the differential marginal costs, we have

$$\begin{aligned} \frac{\partial F(Y(t))}{\partial Y_i^{(j)}(t)} &= \sum_{l \in i} \frac{s_j}{C_l} g'\big(\frac{Z_l(t)}{C_l}\big) \\ &= \sum_{l \in i} \delta_l^{(j)}(Y(t)). \end{aligned} \tag{9}$$

Algorithm 1 describes our flow assignment algorithm that essentially places the newly generated flow on a path that minimizes the increase in the network cost based on either forms (8) or (9).

Algorithm 1 Flow Scheduling Algorithm

Suppose a type-j flow arrives at time t when the system is in state $\mathbf{Y}(t)$. Then,

1: Compute the path marginal costs $w_i^{(j)}(Y(t))$, $i \in R_j$, in either of the forms below:

- Integral form:

$$w_i^{(j)}(Y(t)) = \sum_{l \in i} \Delta_l^{(j)}(Y(t)), \tag{10}$$

- Differential form:

$$w_i^{(j)}(Y(t)) = \sum_{l \in i} \delta_l^{(j)}(Y(t)). \tag{11}$$

2: Place the flow on a path i such that

$$i = \arg\min_{k \in R_j} w_k^{(j)}(Y(t)). \tag{12}$$

Break ties in (12) uniformly at random.

Upon arrival of a flow, Algorithm 1 takes the corresponding feasible paths and their link congestions into the account for computing the path marginal costs $w_i^{(j)}(t)$ but it does not require to know any information about the other links in the network. The two forms (10) and (11) are essentially identical in our asymptotic performance analysis in the next section, however the differential form (11) seems slightly easier to work with. Algorithm 1 can be implemented either centrally

or in a distributed manner using a distributed shortest path algorithm that uses the link marginal costs, $\Delta_l^{(j)}(t)$ or $\delta_l^{(j)}(t)$, as link weights.

IV. Performance Analysis via Fluid Limits

The system state $\{Y(t)\}_{t\geq 0}$ is a stochastic process which is not easy to analyze, therefore we analyze the fluid limits of the system instead. Fluid limits can be interpreted as the first order approximation to the original process $\{Y(t)\}_{t\geq 0}$ and provide valuable qualitative insight into the operation of the algorithm. In this section, we introduce the fluid limits of the process $\{Y(t)\}_{t\geq 0}$ and present our main result regarding the convergence of our algorithm to the optimal cost. We deliberately defer the rigorous claims and proofs about the fluid limits to Section VI and for now mainly focus on the convergence analysis to the optimal cost which is the main contribution of this paper.

A. Informal Description of Fluid Limit Process

In order to obtain the fluid limits, we scale the process in rate and space. Specifically, consider a sequence of systems $\{Y^r(t)\}_{t\geq 0}$ indexed by a sequence of positive numbers r, each governed by the same statistical laws as the original system with the flow arrival rates $r\lambda_j$, $j\in\mathcal{J}$, and initial state $Y^r(0)$ such that $Y^r(0)/r\to y(0)$ as $r\to\infty$ for some fixed $y(0)$. The fluid-scale process is defined as $y^r(t)=Y^r(t)/r$, $t\geq 0$. We also define $y^r(\infty)=Y^r(\infty)/r$, the random state of the fluid-scale process in steady state. If the sequence of processes $\{y^r(t)\}_{t\geq 0}$ converges to a process $\{y(t)\}_{t\geq 0}$ (uniformly over compact time intervals, with probability 1 as $r\to\infty$), the process $\{y(t)\}_{t\geq 0}$ is called the fluid limit. Then, $y_i^{(j)}(t)$ is the fluid limit number of type-j flows routed through path i. Accordingly, we define $z_l^r(t)=Z_l^r(t)/r$ and $x^{(j)^r}(t)=X^{(j)^r}(t)/r$ and their corresponding limits as $z_l(t)$ and $x^{(j)}(t)$ as $r\to\infty$.

The fluid limits under Algorithm 1 follow possibly random trajectories but they satisfy the following set of differential equations. We state the result as the following lemma whose proof can be found in Section VI.

Lemma 1. *(Fluid equations) Any fluid limit $y(t)$ satisfies the following equations. For any $j\in\mathcal{J}$, and $i\in R_j$,*

$$\frac{\mathrm{d}}{\mathrm{d}t}y_i^{(j)}(t)=\lambda_j p_i^{(j)}(y(t))-\mu_j y_i^{(j)}(t) \tag{13a}$$

$$p_i^{(j)}(y(t))=0 \text{ if } i\notin \arg\min_{k\in R_j} w_k^{(j)}(y(t)) \tag{13b}$$

$$p_i^{(j)}(y(t))\geq 0,\ \sum_{i\in R_j} p_i^{(j)}(y(t))=1 \tag{13c}$$

$$w_i^{(j)}(y(t))=\sum_{l\in i}\frac{s_j}{C_l}g'(z_l(t)/C_l). \tag{13d}$$

Equation (13a) is simply an accounting identity for $y_i^{(j)}(t)$ stating that, on the fluid-scale, the number of type-j flows over path $i\in R_j$ increases at rate $\lambda_j p_i^{(j)}(y(t))$, and decreases at rate $y_i^{(j)}\mu_j$ due to departures of type-j flows on path i. $p_i^{(j)}(y(t))$ is the fraction of type-j flow arrivals placed on path i. $w_i^{(j)}(y(t))$ is the fluid-limit marginal cost of routing type-j flows in path i when the system is in state $y(t)$. Equation (13b) follows from (12) and states that the flows can only be placed on the paths which have the minimum marginal cost $\min_{k\in R_j} w_k^{(j)}(y(t))$.

It follows from (13a) and (13c) that the total number of type-j flows in the system, i.e., $x^{(j)}(t)=\sum_{i\in R_j} y_i^{(j)}(t)$, follows a deterministic trajectory described by the following equation,

$$\frac{\mathrm{d}}{\mathrm{d}t}x^{(j)}(t)=\lambda_j-\mu_j x^{(j)}(t),\ \forall j\in\mathcal{J}, \tag{14}$$

which clearly implies that

$$x^{(j)}(t)=\rho_j+(x^{(j)}(0)-\rho_j)e^{-\mu_j t}\ \forall j\in\mathcal{J}. \tag{15}$$

Consequently at steady state,

$$x^{(j)}(\infty)=\rho_j,\ \forall j\in\mathcal{J}, \tag{16}$$

which means that, in steady state, there is a total of ρ_j type-j flows on the fluid scale.

B. Main Result and Asymptotic Optimality

In this section, we state our main result regarding the asymptotic optimality of our algorithm. First note that by (16), the values of $y(\infty)$ are confined to a convex compact set Υ defined below

$$\Upsilon\equiv\{y=(y_i^{(j)}): y_i^{(j)}\geq 0,\ \sum_{i\in R_j} y_i^{(j)}=\rho_j,\ \forall j\in\mathcal{J}\}. \tag{17}$$

Consider the problem of minimizing the network cost in steady state on the fluid scale (the counterpart of the optimization (3)),

$$\min\ \ F(y) \tag{18a}$$

$$\text{s.t.}\ \ y\in\Upsilon. \tag{18b}$$

Denote by $\Upsilon^\star\subseteq\Upsilon$ the set of optimal solutions to the optimization (18). The following proposition states that the fluid limits of our algorithm indeed converge to an optimal solution of the optimization (18).

Proposition 1. *Consider the fluid limits of the system under Algorithm 1 with initial condition $y(0)$, then as $t\to\infty$*

$$d(y(t),\Upsilon^\star)\to 0. \tag{19}$$

Convergence is uniform over initial conditions chosen from a compact set.

The theorem below makes the connection between the fluid limits and the original optimization problem (3). It states the main result of this paper which is the asymptotic optimality of Algorithm 1.

Theorem 1. *Let $Y^r(t)$ and $Y^r_{opt}(t)$ be respectively the system trajectories under Algorithm 1 and any optimal algorithm for the optimization (3). Then in steady state,*

$$\lim_{r\to\infty}\frac{\mathbb{E}\Big[F(Y^r(\infty))\Big]}{\mathbb{E}\Big[F(Y^r_{opt}(\infty))\Big]}=1. \tag{20}$$

For example, one optimal algorithm that solves (3) is the one that every time a flow arrives or departs, it re-routes the existing flows in the network in order to minimize the network cost at all times. Of course this requires solving a complex combinatorial problem every time a flow arrives/departs and further it interrupts/migrates the existing flows. Under any algorithm (including our algorithm and the optimal one), the mean number of flows in the system in steady state is $\mathrm{O}(r)$. Thus by Theorem 1, Algorithm 1 has roughly the same cost as the optimal cost when the number of flows in the system is large, but at much lower complexity and with no migrations/interruptions.

The rest of this section is devoted to the proof of Proposition 1. The proof of Theorem 1 relies on Proposition 1 and is provided in Section VI.

C. Proof of Proposition 1

We first characterize the set of optimal solutions $\Upsilon^\star$ using KKT conditions in the lemma below.

Lemma 2. *Let $\Gamma_j = \{i \in R_j : y_i^{(j)} > 0\} \subseteq R_j,\ j \in \mathcal{J}$. A vector $y \in \Upsilon^\star$ iff $y \in \Upsilon$ and there exists a vector $\eta \geq 0$ such that*

$$w_i^{(j)}(y) = \eta_j,\ \forall i \in \Gamma_j, \tag{21a}$$

$$w_i^{(j)}(y) \geq \eta_j,\ \forall i \in R_j \setminus \Gamma_j, \tag{21b}$$

where $w_i^{(j)}(\cdot)$ defined in (13d).

Proof of Lemma 2. Consider the following optimization problem,

$$\min \quad F(y) \tag{22a}$$

$$\text{s.t.} \quad \sum_{i \in R_j} y_i^{(j)} \geq \rho_j,\ \forall j \in \mathcal{J} \tag{22b}$$

$$y_i^{(j)} \geq 0,\ \forall j \in \mathcal{J},\ \forall i \in R_j. \tag{22c}$$

Since $F(y)$ is an strictly increasing function with respect to $y_i^{(j)}$, for all $j \in \mathcal{J}, i \in R_j$, it is easy to check that the optimization (18) has the same set of optimal solutions as the optimization (22). Moreover, both optimizations have the same optimal value. Hence we can use the Lagrange multipliers $\eta_j \geq 0$ and $\nu_i^{(j)} \geq 0$ to characterize the Lagrangian as follows.

$$\begin{aligned} L(\eta, \nu, y) =& F(y) + \sum_{j \in \mathcal{J}} \eta_j (\rho_j - \sum_{i; i \in R_j} y_i^{(j)}) \\ & - \sum_{j \in \mathcal{J}} \sum_{i; i \in R_j} \nu_i^{(j)} y_i^{(j)} \\ =& \sum_l g_l \big(\sum_{j \in \mathcal{J}} s_j \sum_{i \in R_j, l \in i} y_i^{(j)} \big) \\ & + \sum_{j \in \mathcal{J}} \big[\eta_j \rho_j - \sum_{i; i \in R_j} (\eta_j + \nu_i^{(j)}) y_i^{(j)} \big]. \end{aligned} \tag{23}$$

From KKT conditions [23], $y \in \Upsilon^\star$, if and only if there exist vectors η and ν such that the following holds.

Feasibility:

$$y \in \Upsilon, \tag{24a}$$

$$\eta_j \geq 0, \quad \forall j \in \mathcal{J}, \tag{24b}$$

$$\nu_i^{(j)} \geq 0, \quad \forall j \in \mathcal{J},\ i \in R_j, \tag{24c}$$

Complementary slackness:

$$\eta_j (\rho_j - \sum_{i; i \in R_j} y_i^{(j)}) = 0, \quad \forall j \in \mathcal{J}, \tag{25a}$$

$$\nu_i^{(j)} y_i^{(j)} = 0, \quad \forall j \in \mathcal{J},\ i \in R_j, \tag{25b}$$

Stationarity:

$$\frac{\partial L(\eta, \nu, y)}{\partial y_j^{(i)}} = 0.\ \ \forall j \in \mathcal{J},\ i \in R_j. \tag{26a}$$

Note that (24a) implies (25a). It follows from (26a) that

$$\frac{\partial F(y)}{\partial y_j^{(i)}} = \eta_j + \nu_i^{(j)},\ \forall j \in \mathcal{J}, i \in R_j. \tag{27}$$

Define Γ_j as in the statement of the lemma. Note that Γ_j is nonempty for all $j \in \mathcal{J}$ by (24a). Then combining (25b) and (27), $\forall j \in \mathcal{J}$, and noting that $\frac{\partial F(y)}{\partial y_i^{(j)}} = w_i^{(j)}(y)$ by definition, yields (21a)-(21b).

□

Next, we show that the set of optimal solutions $\Upsilon^\star$ is an invariant set of the fluid limits, using the fluid limit equations (13a)-(13d), and Lemma 2.

Lemma 3. *$\Upsilon^\star$ is an invariant set for the fluid limits, i.e., starting from any initial condition $y(0) \in \Upsilon^\star$, $y(t) \in \Upsilon^\star$ for all $t \geq 0$.*

Proof of Lemma 3. Consider a type-j flow and let

$$I^{(j)}(t) = \arg\min_{i \in R_j} w_i^{(j)}(y(t))$$

be the set of paths with the minimum path marginal cost. Note that $\sum_{i \in I^{(j)}(t)} p_i^{(j)}(t) = 1$, $t \geq 0$, by (13b), therefore

$$\frac{\mathrm{d}}{\mathrm{d}t} \Big(\sum_{i \in I_i^{(j)}(t)} y_i^{(j)}(t) \Big) = \lambda_j - \Big(\sum_{i \in I^{(j)}(t)} y_i^{(j)}(t) \Big) \mu_j. \tag{28}$$

Since $y(0) \in \Upsilon^\star$, it follows from Lemma 2 that $\sum_{i \in I^{(j)}(0)} y_i^{(j)}(0) = \rho_j$. Hence, Equation (28) has a unique solution for $\sum_{i \in I^{(j)}(t)} y_i^{(j)}(t)$ which is

$$\sum_{i \in I^{(j)}(t)} y_i^{(j)}(t) = \rho_j,\ t \geq 0. \tag{29}$$

On the other hand, since $x^{(j)}(0) = \rho_j$, by (15),

$$x^{(j)}(t) = \sum_{i \in R_j} y_i^{(j)}(t) = \rho_j,\ t \geq 0. \tag{30}$$

Equations (29) and 30 imply that, at any time $t \geq 0$, $y_i^{(j)}(t) = 0$ for $i \notin I^{(j)}(t)$, and $y_i^{(j)}(t) \geq 0$ for $i \in I^{(j)}(t)$ such that

$\sum_{i\in I^{(j)}(t)} y_i^{(j)}(t) = \rho_j$. Hence, $y(t) = \left(y_i^{(j)}(t)\right) \in \Upsilon^\star$ by using $\eta_j(t) = \min_{k\in R_j} w_k^{(j)}(y(t))$ in Lemma 2. □

Next, we show that the fluid limits indeed converge to the invariant set $\Upsilon^\star$ starting from an initial condition in Υ.

Lemma 4. *(Convergence to the invariant set) Consider the fluid limits of the system under Algorithm 1 with initial condition $y(0) \in \Upsilon$, then*

$$d(y(t), \Upsilon^\star) \to 0. \tag{31}$$

Also convergence is uniform over the set of initial conditions Υ.

Proof of Lemma 4. Starting from $y(0) \in \Upsilon$, (15) implies that

$$x^{(j)}(t) = \sum_{i\in R_j} y_i^{(j)}(t) = \rho_j \quad \forall j \in \mathcal{J}, \tag{32}$$

at any time $t \geq 0$. To show convergence of $y(t)$ to the set $\Upsilon^\star$, we use a Lyapunov argument. Specifically, we choose $F(.)$ as the Lyapunov function and show that $(\mathrm{d}/\mathrm{d}t)F(y(t)) < 0$ if $y(t) \notin \Upsilon^\star$. Let $\eta_j(y(t)) = \min_{k\in R_j} w_k^{(j)}(y(t))$. Then

$$\begin{aligned}
(\mathrm{d}/\mathrm{d}t)F(y(t)) &= \sum_{j\in\mathcal{J}} \sum_{i\in R_j} \frac{\partial F(y)}{\partial y_i^{(j)}} \frac{\mathrm{d}y_i^{(j)}(t)}{\mathrm{d}t} \\
&= \sum_{j\in\mathcal{J}} \sum_{i\in R_j} w_i^{(j)}(y(t)) \big[\lambda_j p_i^{(j)}(t) - \mu_j y_i^{(j)(t)}\big] \\
&= \sum_{j\in\mathcal{J}} \mu_j \big[\rho_j \sum_{i\in R_j} w_i^{(j)}(y(t)) p_i^{(j)}(t) - \sum_{i\in R_j} w_i^{(j)}(y(t)) y_i^{(j)}(t)\big] \\
&\overset{(a)}{=} \sum_{j\in\mathcal{J}} \mu_j \big[\rho_j \eta_j(y(t)) - \sum_{i\in R_j} w_i^{(j)}(y(t)) y_i^{(j)}(t)\big] \qquad (33) \\
&\overset{(b)}{<} \sum_{j\in\mathcal{J}} \mu_j \big[\rho_j \eta_j(y(t)) - \eta_j(y(t)) \sum_{i\in R_j} y_i^{(j)}(t)\big] \\
&\overset{(c)}{=} 0.
\end{aligned}$$

Equality (a) follows from the fact that $p_i^{(j)}(t) = 0$ if $w_i^{(j)}(t) > \eta_j(t)$ by (13b). Inequality (b) follows from the fact that $y(t) \notin \Upsilon^\star$, so by Lemma 2, there exists an $i \in R_j$ such that $y_i^{(j)}(t) > 0$ but $w_i^{(j)}(y(t)) > \eta_j(y(t))$. Equality (c) holds because of (32). □

Now we are ready to complete the proof of Proposition 1, i.e., to show that starting from any initial condition in a compact set, uniform convergence to the invariant set $\Upsilon^\star$ holds.

Proof of Proposition 1. First note that $(d/dt)F(y(t))$ (as given by (33)) is a continuous function with respect to $y(t) = (y_i^{(j)}(t) \geq 0)$. This is because the path marginal costs $w_i^{(j)}(y(t))$ are continuous functions of $y(t)$ and so is their minimum $\eta_j(y(t)) = \min_{i\in R_j} w_i^{(j)}(y(t)))$.

Next, note that by Lemma 4, for any $\epsilon_1 > 0$, and $a \in \Upsilon$, there exists an $\epsilon_2 > 0$ such that if $F(a) - F(\Upsilon^\star) \geq \epsilon_1$ then $(d/dt)F(y(t))\big|_{y(t)=a} \leq -\epsilon_2$. By the continuity of $(d/dt)F(y(t))$ in $y(t)$, there exists a $\delta > 0$ such that $\|y(t) - a\| \leq \delta$ implies $|(d/dt)F(y(t)) - (d/dt)F(a)| \leq \epsilon_2/2$. Therefore, for all $y(t)$ such that $\|y(t) - a\| \leq \delta$,

$$(d/dt)F(y(t)) \leq -\epsilon_2/2.$$

By (15), for any $\delta > 0$, we can find t_δ large enough such that for all $t > t_\delta$, $\|y(t) - a\| \leq \delta$ for some $a \in \Upsilon$. Putting everything together, for any $\epsilon_1 > 0$, there exists $\epsilon_2 > 0$ such that if $F(y(t)) - F(\Upsilon^\star) \geq \epsilon_1$ then $(d/dt)F(y(t)) \leq -\epsilon_2/2$. This completes the proof of Proposition 1. □

V. Simulation Results

In this section, we provide simulation results and evaluate the performance of our algorithm under a wide range of traffic conditions in the following datacenter architectures:

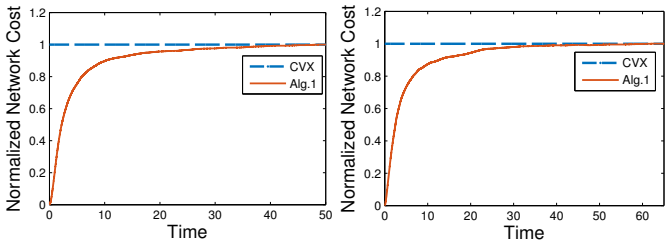
- *FatTree* which consists of a collection of edge, aggregation, and core switches and offers equal length path between the edge switches. Figure 1a shows a FatTree with 16 servers and 8 4-port edge switches. For simulations, we consider a FatTree with 128 servers and 32 8-port edge switches.
- *JellyFish* which is a random graph in which each switch i has k_i ports out of which r_i ports are used for connection to other switches and the remaining $k_i - r_i$ ports are used for connection to servers. Figure 1b shows a JellyFish with 4-port switches, and $k_i = 4$, $r_i = 2$ for all the switches. For simulations, we consider a JellyFish constructed using 20 8-port switches and 100 servers. Each 8-port switch is connected to 5 servers and 3 remaining links are randomly connected to other switches (this corresponds to $k_i = 8$, $r_i = 3$ for all the switches).

Our rationale for selecting these architectures stems from the fact that they are on two opposing sides of the spectrum of topologies: while FatTree is a highly structured topology, JellyFish is a random topology; hence they should provide a good estimate for the robustness of our algorithm to different network topologies and possible link failures.

We generate the flows under two different traffic models to which we refer to as *exponential model* and *empirical model*:

- *Exponential model*: Flows are generated per Poisson processes and exponentially distributed durations. The parameters of duration distribution is chosen uniformly at random from 0.5 to 1.5 for different flows. The flow sizes are chosen according to a log-normal distribution.
- *Empirical model*: Flows are generated based on recent empirical studies on characterization of datacenter traffic. As suggested by these studies, we consider log-normal inter-arrival times [24], service times based on the empirical result in [10], and log-normal flow sizes [24]. Particularly, the most periods of congestion tend to be short lived, namely, more than 90% of the flows that are more than 1 second long, are no longer than 2 seconds [10].

In both models, the flow sizes are log-normal with mean 0.1 and standard deviation 1. This generates flow sizes ranging



(a) Convergence in FatTree. (b) Convergence in JellyFish.

Fig. 2: Convergence of the network cost under Algorithm 1, normalized with the the lower-bound on the optimal solution (CVX), to 1. The scaling parameter r is 100 here.

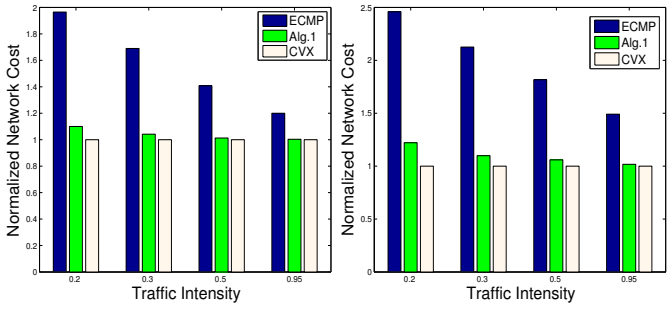
from 0.1% to 40% of link capacity which captures the nature of flow sizes in terms of "mice" and "elephant" flows. Furthermore, we consider a random traffic pattern, i.e., source and destination of flows are chosen uniformly at random. The link cost parameter α is chosen to be 1 in this simulations.

Under both models, to change the traffic intensity, we keep the other parameters fixed and scale the arrival rates (with parameter r).

We report the simulation results in terms of the performance ratio between our algorithm and a benchmark algorithm (similar to (20)). Since the optimal algorithm is hard to implement, instead we use a convex relaxation method to find a lower-bound on the optimal cost at each time. Specifically, every time a flow arrives or departs, we use CVX [25], to minimize $F(Y(t))$, by relaxing the combinatorial constraints, i.e., allowing splitting of flows among multiple paths and re-routing the existing flows. We compare the network cost under our algorithm (Algorithm 1) and traditional ECMP, normalized by the lower-bound on the optimal solution (to which we refer to as CVX in the plots).

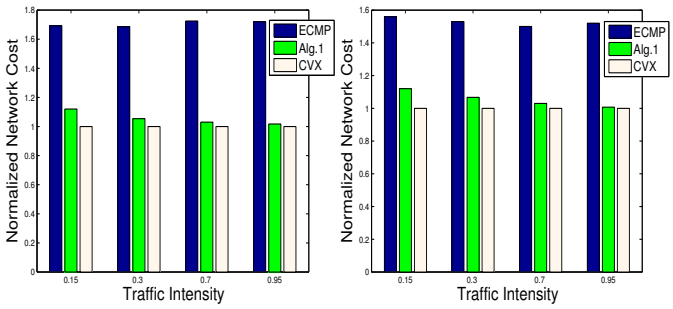
A. Experimental Results for FatTree

Figure 2a shows that the aggregate cost under Algorithm 1 indeed converges to the optimal solution (normalized cost ratio goes to 1) which verifies Theorem 1. Figures 3a and 3b show the cost performance under Algorithm1 and ECMP, normalized by the CVX lower-bound, under the exponential and the empirical traffic models respectively. The traffic intensity is measured in terms of the ratio between the steady state offered load and the bisection bandwidth. For FatTree, the bisection bandwidth depends on the number of core switches and their number of ports. As we can see, our algorithm is very close to the lower-bound on the optimal value (CVX) for light, medium, and high traffic intensities. They also suggest that Theorem 1 indeed holds under more general arrival and service time processes. In this simulations, our algorithm gave a performance improvement ranging form 50% to more than 100%, compared to ECMP, depending on the traffic intensity, under the empirical traffic model.



(a) Exponential traffic model. (b) Empirical traffic model

Fig. 3: Performance ratio of Algorithm 1 and ECMP in FatTree, normalized with the lower-bound (CVX).



(a) Exponential traffic model. (b) Empirical traffic model.

Fig. 4: Performance ratio of Algorithm 1 and ECMP in JellyFish, normalized with the lower-bound (CVX).

B. Experimental Results for JellyFish

Figure 2b shows that the aggregate cost under Algorithm 1 indeed converges to the optimal solution which again verifies Theorem 1. Figures 4a and 4b compare the performance of Algorithm 1 and ECMP, normalized with the lower-boud on the optimal solution (CVX), under both the exponential and empirical traffic models. As before, the traffic intensity is measured by the ratio between the steady state offered load and the bisection bandwidth. To determine the bisection bandwidth, we have used the bounds reported in [26], [27] for regular random graphs. Again we see that our algorithm performs very well in all light, medium, and high traffics. In JellyFish, our algorithm yields performance gains ranging from 60% to 70%, compared to ECMP, under the empirical traffic model.

VI. Formal Proofs of Fluid Limits and Theorem 1

A. Proof of Fluid Limits

We prove the existence of fluid limits under our algorithm and derive the corresponding fluid equations (13a)-(13d). Arguments in this section are quite standard [28], [29], [30]. Recall that $Y^r(t)$ is the system state with the flow arrival rate $r\lambda_j$, $j \in \mathcal{J}$, and initial state $Y^r(0)$. The fluid-scale process is $y^r(t) = Y^r(t)/r$, $t \in [0, \infty)$. Similarly, $z_l^r(t) = Z_l^r(t)/r$ and $x^{(j)^r}(t) = X^{(j)^r}(t)/r$ are defined. We assume that $y^r(0) \to y(0)$ as $r \to \infty$ for some fixed $y(0)$.

We first show that, under Algorithm 1, the limit of the process $\{y^r(t)\}_{t\geq 0}$ exists along a subsequence of r as we show next. The process $Y^r(t)$ can be constructed as follows

$$\begin{aligned} Y_i^{(j)^r}(t) =& Y_i^{(j)^r}(0) + \Pi_{i,j}^a\Big(\int_0^t P_i^{(j)}(Y^r(s))r\lambda_j ds\Big) \\ &- \Pi_{i,j}^d\Big(\int_0^t \mu_j Y_i^{(j)^r}(s)ds\Big) \quad \forall j\in\mathcal{J}, i\in R_j \end{aligned} \tag{34}$$

where $\Pi_{i,j}^a(.)$ and $\Pi_{i,j}^d(.)$ are independent unit-rate Poisson processes, and $P_i^{(j)}(Y^r(t))$ is the probability of assigning a type-j flow to path i when the system state is $Y^r(t)$. Note that by the Functional Strong Law of Large Numbers [31], almost surely,

$$\frac{1}{r}\Pi_{i,j}^a(rt)\to t,\ \text{u.o.c.};\ \frac{1}{r}\Pi_{i,j}^d(rt)\to t,\ \text{u.o.c.} \tag{35}$$

where u.o.c. means uniformly over compact time intervals. Define the fluid-scale arrival and departure processes as

$$\begin{aligned} a_{i,j}^r(t) &= \frac{1}{r}\Pi_{i,j}^a\Big(\int_0^t P_i^{(j)}(Y^r(s))r\lambda_j ds\Big), \\ d_{i,j}^r(t) &= \frac{1}{r}\Pi_{i,j}^d\Big(\int_0^t \mu_j Y_i^{(j)^r}(s)ds\Big). \end{aligned} \tag{36}$$

Lemma 5. *(Convergence to fluid limit sample paths) If $y^r(0)\to y(0)$, then almost surely, every subsequence $(y^{r_n}, a^{r_n}, d^{r_n})$ has a further subsequence $(y^{r_{n_k}}, a^{r_{n_k}}, d^{r_{n_k}})$ such that $(y^{r_{n_k}}, a^{r_{n_k}}, d^{r_{n_k}})\to(y,a,d)$. The sample paths y, a, d are Lipschitz continuous and the convergence is u.o.c.*

Proof Sketch of Lemma 5. The proof is standard and follows from the fact that $a_{i,j}^r(.)$ and $d_{i,j}^r(.)$ are asymptotically Lipschitz continuous (see e.g., [28], [29], [32] for similar arguments), namely, there exists a constant $C>0$ such that for $0\leq t_1\leq t_2<\infty$,

$$\limsup_r (a_{i,j}^r(t_2) - a_{i,j}^r(t_1)) \leq C(t_2-t_1),$$

and similarly for $d_{i,j}^r(.)$. The above inequality follows from (35) and noting that $(y^r(.))$ is uniformly bounded over any finite time interval for large r. So the limit (y,a,d) exists along the subsequence. □

Proof of Lemma 1. It follows from (34), (36), (35), and the existence of the fluid limits (Lemma 5), that

$$y_i^{(j)}(t) = y_i^{(j)}(0) + a_i^{(j)}(t) - d_i^{(j)}(t),$$

where

$$d_i^{(j)}(t) = \int_0^t y_i^{(j)}(s)\mu_j ds,$$

$$\sum_{i\in R_j} a_i^{(j)}(t) = \lambda_j t,\ a_i^{(j)}(t) \text{ is nondecreasing.}$$

The fluid equations (13a) and (13c) are the diffrential form of these equations (the fluid sample paths are Lipschitz continuous so the derivatives exist almost everywhere), where

$$p_i^{(j)}(t) := \frac{1}{\lambda_j}\frac{\mathrm{d}a_i^{(j)}(t)}{\mathrm{d}t}. \tag{37}$$

For any type j, let $w_j^\star(y(t)) = \min_{i\in R_j} w_i^{(j)}(y(t))$, for $w_i^{(j)}(y(t))$ defined in (13d). Consider any regular time t and a path $i\notin \arg\min_{i\in R_j} w_i^{(j)}(y(t))$. By the continuity of $w_i^{(j)}(y(t))$, there must exist a small time interval (t_1,t_2) around t such that $w_i^{(j)}(y(\tau)) > w_j^\star(\tau)$ for all $\tau\in(t_1,t_2)$. Consequently, for all r large enough along the subsequence, $w_i^{(j)}(y^r(\tau)) > w_j^\star(y^r(\tau))$, $\tau\in(t_1,t_2)$. Multiplying both sides by r^α, it follows that $w_i^{(j)}(Y^r(\tau)) > w_j^\star(Y^r(\tau))$, $\tau\in(t_1,t_2)$. Hence $P_i^{(j)}(Y^r(\tau))=0$, $\tau\in(t_1,t_2)$, and $a_i^{r(j)}(t_1,t_2)=0$, for all r large enough along the subsequence. Therefore $a_i^{(j)}(t_1,t_2)=0$ which shows that $(\mathrm{d}/\mathrm{d}t)a_i^{(j)}(t)=0$ at $t\in(t_1,t_2)$. This establishes (13b). □

B. Proof of Theorem 1

We first show that

$$F(y^r(\infty)) \Longrightarrow F^\star, \tag{38}$$

where $F^\star = F(\Upsilon^\star)$ is the optimal cost. By Proposition 1 and the continuity of $F(\cdot)$, for any fluid sample path $y(t)$ with initial condition $y(0)$, we can choose t_{ϵ_1} large enough such that given any small $\epsilon_1>0$, $|F(y(t_{\epsilon_1}))-F^\star|\leq\epsilon_1$. With probability 1, $y^r(t)\to y(t)$ u.o.c. (see Lemma 5), hence, by the continuous mapping theorem [31], we also have $F(y^r(t))\to F(y(t))$, u.o.c. For any $\epsilon_2>0$, for r large enough, we can choose an $\epsilon_3>0$ such that, uniformly over all initial states $y^r(0)$ such that $\|y^r(0)-y(0)\|\leq\epsilon_3$,

$$\mathbb{P}\{|F(y^r(t_{\epsilon_1}) - F(y(t_{\epsilon_1}))| < \epsilon_1\} > 1-\epsilon_2 \tag{39}$$

This claim is true, since otherwise for a sequence of initial states $y^r(t)\to y(0)$ we have

$$\mathbb{P}\{|F(y^r(t_{\epsilon_1}) - F(y(t_{\epsilon_1}))| < \epsilon_1\} \leq 1-\epsilon_2,$$

which is impossible because, almost surely, we can choose a subsequence of r along which uniform convergence $F(y^r(t))\to F(y(t))$, with initial condition $y(0)$ holds. Hence,

$$\begin{aligned} &\mathbb{P}\{|F(y^r(t_{\epsilon_1}))-F^\star|<2\epsilon_1\} \\ &\geq \mathbb{P}\{|F(y^r(t_{\epsilon_1}) - F(y(t_{\epsilon_1}))| + |F(y(t_{\epsilon_1}))-F^\star| < 2\epsilon_1\} \\ &\geq \mathbb{P}\{|F(y^r(t_{\epsilon_1}) - F(y(t_{\epsilon_1}))| < \epsilon_1\} > 1-\epsilon_2 \end{aligned}$$

which in particular implies (38) because ϵ_1 and ϵ_2 can be made arbitrarily small.

Next, we show (20). Under any algorithm (including our algorithm and the optimal one), $\sum_{i\in R_j} Y_i^{(j)^r}(\infty)/r = X^{(j)^r}(\infty)/r$, where $X^{(j)^r}(\infty)$ has Poisson distribution with mean $r\rho_j$, and $X^{(j)^r}(\infty)$, $j\in\mathcal{J}$, are independent. Let $\bar{s}=\max_{j\in\mathcal{J}} s_j<\infty$. The traffic over each link l is clearly bounded as $Z_l^r/r < \bar{s}\sum_j X^{(j)^r}(\infty)/r = \bar{s}X^r(\infty)/r$ where $X^r(\infty)$ has Poisson distribution with mean $r\sum_j\rho_j$. Hence, $F(y^r(\infty))$ is stochastically dominated by $|E|g\Big(\frac{\bar{s}X^r(\infty)/r}{C_l}\Big)$, and g is polynomial. It then follows that the sequence of random variables $\{F(y^r(\infty))\}$ (and also $\{y^r(\infty)\}$) are uniformly integrable under any algorithm. Then, in view of (38), by Theorem 3.5 of [31], under our algorithm.

$$\mathbb{E}\Big[F(Y^r(\infty)/r)\Big]\to F^\star. \tag{40}$$

Now consider any optimal algorithm for the optimization (3). It holds that $F(\mathbb{E}\Big[y^r_{\text{opt}}(\infty)\Big]) \leq \mathbb{E}\Big[F(y^r_{\text{opt}}(\infty))\Big] \leq \mathbb{E}\Big[F(y^r(\infty))\Big]$ where the first inequality is by Jensen's inequality. Taking the limit as $r \to \infty$, it follows by an squeeze argument that

$$\mathbb{E}\Big[F(Y^r_{\text{opt}}(\infty)/r)\Big] \to F^\star. \tag{41}$$

(40) and (41) will imply (20) in view of the polynomial structure of F.

VII. Concluding Remarks

This paper presents a simple algorithm that dynamically adjusts the link weights as a function of the link utilizations and places any newly generated flow on a least weight path in the network, with no splitting/migration of existing flows. We demonstrate both theoretically and experimentally that this algorithm has a good load balancing performance. In particular, we prove that the algorithm asymptotically minimizes a network cost and establish the relationship between the network cost and the corresponding weight construct. Although our theoretical result is an asymptotic result, our experimental results show that the algorithm in fact performs very well under a wide range of traffic conditions and different datacenter networks.

While the algorithm has low complexity, the real implementation depends on how fast the weight updates and least weight paths can be computed in practical datacenters (e.g., based on SDN). One possible way to improve the computation timescale is to perform the computation periodically or only for long flows, while using the previously computed least weight paths for short flows or between the periodic updates.

References

[1] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 4, pp. 63–74, 2008.

[2] T. Benson, A. Anand, A. Akella, and M. Zhang, "MicroTE: Fine grained traffic engineering for data centers," in *Proceedings of the 7th Conference on Emerging Networking Experiments and Technologies*. ACM, 2011, p. 8.

[3] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat, "Hedera: Dynamic flow scheduling for data center networks." in *NSDI*, vol. 10, 2010, pp. 19–19.

[4] C. Raiciu, S. Barre, C. Pluntke, A. Greenhalgh, D. Wischik, and M. Handley, "Improving datacenter performance and robustness with multipath TCP," *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 4, pp. 266–277, 2011.

[5] S. Kandula, D. Katabi, S. Sinha, and A. Berger, "Dynamic load balancing without packet reordering," *ACM SIGCOMM Computer Communication Review*, vol. 37, no. 2, pp. 51–62, 2007.

[6] A. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta, "VL2: A scalable and flexible data center network," in *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 4, 2009, pp. 51–62.

[7] 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.

[8] M. Bradonjić, I. Saniee, and I. Widjaja, "Scaling of capacity and reliability in data center networks," *ACM SIGMETRICS Performance Evaluation Review*, vol. 42, no. 2, pp. 46–48, 2014.

[9] A. Singla, C.-Y. Hong, L. Popa, and P. B. Godfrey, "Jellyfish: Networking data centers randomly." in *NSDI*, vol. 12, 2012, pp. 17–17.

[10] S. Kandula, S. Sengupta, A. Greenberg, P. Patel, and R. Chaiken, "The nature of data center traffic: Measurements & analysis," in *Proceedings of the 9th ACM SIGCOMM Conference On Internet Measurement Conference*, 2009, pp. 202–208.

[11] S. Even, A. Itai, and A. Shamir, "On the complexity of time table and multi-commodity flow problems," in *16th Annual Symposium on Foundation of Computer Science*. IEEE, 1975, pp. 184–193.

[12] G. M. Guisewite and P. M. Pardalos, "Minimum concave-cost network flow problems: Applications, complexity, and algorithms," *Annals of Operations Research*, vol. 25, no. 1, pp. 75–99, 1990.

[13] R. Niranjan Mysore, A. Pamboris, N. Farrington, N. Huang, P. Miri, S. Radhakrishnan, V. Subramanya, and A. Vahdat, "Portland: A scalable fault-tolerant layer 2 data center network fabric," in *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 4, 2009, pp. 39–50.

[14] J. Cao, R. Xia, P. Yang, C. Guo, G. Lu, L. Yuan, Y. Zheng, H. Wu, Y. Xiong, and D. Maltz, "Per-packet load-balanced, low-latency routing for clos-based data center networks," in *Proceedings of the 9th ACM Conference on Emerging Networking Experiments and Technologies*. ACM, 2013, pp. 49–60.

[15] P. Gill, N. Jain, and N. Nagappan, "Understanding network failures in data centers: Measurement, analysis, and implications," in *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 4, 2011, pp. 350–361.

[16] S. Sen, D. Shue, S. Ihm, and M. J. Freedman, "Scalable, optimal flow routing in datacenters via local link balancing," in *Proceedings of the 9th ACM Conference on Emerging Networking Experiments and Technologies*, 2013, pp. 151–162.

[17] J. W. Jiang, T. Lan, S. Ha, M. Chen, and M. Chiang, "Joint VM placement and routing for data center traffic engineering," in *Proceedings of IEEE, INFOCOM, 2012*, pp. 2876–2880.

[18] A. Dixit, P. Prakash, Y. C. Hu, and R. R. Kompella, "On the impact of packet spraying in data center networks," in *Proceedings of IEEE, INFOCOM, 2013*, pp. 2130–2138.

[19] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: Enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008.

[20] M. Casado, M. J. Freedman, J. Pettit, J. Luo, N. Gude, N. McKeown, and S. Shenker, "Rethinking enterprise network control," *IEEE/ACM Transactions on Networking (TON)*, vol. 17, no. 4, pp. 1270–1283, 2009.

[21] M. Chiesa, G. Kindler, and M. Schapira, "Traffic engineering with equal-cost-multipath: An algorithmic perspective," in *Proceedings of IEEE, INFOCOM, 2014*, pp. 1590–1598.

[22] B. Fortz and M. Thorup, "Internet traffic engineering by optimizing OSPF weights," in *Proceeding of 19th annual joint conference of the IEEE computer and communications societies. INFOCOM 2000*, vol. 2, pp. 519–528.

[23] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[24] D. Ersoz, M. S. Yousif, and C. R. Das, "Characterizing network traffic in a cluster-based, multi-tier data center," in *ICDCS'07. 27th International Conference on Distributed Computing Systems, 2007.* IEEE, pp. 59–59.

[25] M. Grant and S. Boyd, "CVX: Matlab software for disciplined convex programming, version 2.1," http://cvxr.com/cvx, Mar. 2014.

[26] J. Díaz, M. J. Serna, and N. C. Wormald, "Bounds on the bisection width for random d-regular graphs," *Theoretical Computer Science*, vol. 382, no. 2, pp. 120–130, 2007.

[27] B. Bollobás, *Random graphs*. Springer, 1998.

[28] A. L. Stolyar, "An infinite server system with general packing constraints," *Operations Research*, vol. 61, no. 5, pp. 1200–1217, 2013.

[29] A. L. Stolyar and Y. Zhong, "Asymptotic optimality of a greedy randomized algorithm in a large-scale service system with general packing constraints," *Queueing Systems*, vol. 79, no. 2, pp. 117–143, 2015.

[30] J. Ghaderi, Y. Zhong, and R. Srikant, "Asymptotic optimality of BestFit for stochastic bin packing," *ACM SIGMETRICS Performance Evaluation Review*, vol. 42, no. 2, pp. 64–66, 2014.

[31] P. Billingsley, *Convergence of probability measures*. John Wiley & Sons, 2013.

[32] S. N. Ethier and T. G. Kurtz, *Markov processes: Characterization and convergence*. John Wiley & Sons, 2009, vol. 282.