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

Mehrnoosh Shafiee, *Student Member, IEEE*, and Javad Ghaderi, *Member, IEEE*

***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 an equal-cost multi path (ECMP) 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 the 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 congestions. 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 consider low-complexity congestion-aware algorithms that assign the flows to the available paths in an *online fashion* and *without splitting*. Specifically, we propose a myopic algorithm that assigns every arriving flow to an available path with the minimum marginal cost (i.e., the path which yields the minimum increase in the network cost after assignment) and prove that it asymptotically minimizes the total network cost. Extensive simulation results are presented to verify the performance of the myopic algorithm under a wide range of traffic conditions and under different datacenter architectures. Furthermore, we propose randomized versions of our myopic algorithm, which have much lower complexity and empirically show that they can still perform very well in symmetric network topologies.**

***Index Terms*— Markov chains, load balancing, online algorithms, routing algorithms, datacenter network.**

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., [2]–[6]) or designing new networks with better topological features (e.g., FatTree [2], VL2 [7], hypercube [8], hypergrid [9], random graphs such as JellyFish [10], 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 (rate) and a duration. Once a flow is assigned to a path in the network, it consumes resource (bandwidth) equal to its size (rate) 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 multi flow routing considered in this paper has two key distinguishing objectives:

1) *it does not allow flow splitting* because splitting the flow is undesirable due to TCP reordering effect [11]. Resolving packet reordering requires modification of protocol stack [12], which might be costly. Without splitting, many versions of multi flow routing in networks become hard combinatorial problems [13], [14]. In fact, the static version of the problem considered in this paper (i.e., given a static list of flows, assigning flows to paths without splitting so as to balance the load in the network) is known to be NP-hard, through its connection to the Partition problem [15].[1]
2) *it allows dynamic routing* because it considers the current utilization of links in the network when making the routing decisions for newly arrived flows unlike static solutions where the mapping of flows to the paths is fixed and requires the knowledge of the traffic matrix.

Manuscript received December 2, 2016; revised July 15, 2017; accepted September 3, 2017; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor A. Ferragut. Date of publication September 27, 2017; date of current version December 15, 2017. This work was supported by the NSF under Grant CNS-1652115 and Grant CNS-1565774. An earlier version of this paper appeared in the INFOCOM 2016 conference [1]. *(Corresponding author: Mehrnoosh Shafiee.)*

The authors are with the Department of Electrical Engineering, Columbia University, New York, NY 10027 USA (e-mail: s.mehrnoosh@columbia.edu; jghaderi@ee.columbia.edu).

Digital Object Identifier 10.1109/TNET.2017.2751251

[1]In the Partition problem, given a set of numbers, we are asked to divide them into two subsets such that the maximum of the sum of the numbers in the sets is minimized. This can be reformulated as the load balancing in a simple two-node network with two parallel edges.

A. Related Work

Seminal solutions for flow routing in datacenters (e.g. [7], [16]) 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 [3]–[6], [17] 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 [18] or in recently proposed datacenter architectures [10]. Theoretical performance of ECMP in Clos networks under a static flow model has been studied in [19]. There have been recent efforts to address the shortcomings of ECMP. The proposed algorithms range from centralized solutions (e.g., [3], [4]), where a centralized scheduler makes routing decisions based on global view of the network, to distributed solutions (e.g., [6], [20]) 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., [5]) where the routing decisions are made by the end-host transport protocol rather than by the network operator; however, they require significant changes to Transport layer which might not be feasible in public cloud platforms [12]. Authors in [21] investigated a more general problem based on a Gibbs sampling technique and proposed 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 [12], [20], [22]. As explained, dealing with packet reordering involves overhead and modification of protocol stack.

Our work is also related to a large body of literature on traffic engineering and congestion control. For brevity, we only highlight the most relevant work. The first line of work, e.g. [23]–[25], studies the problem of minimizing the cost of carrying traffic in a static multi-commodity flow model and under a convex cost function for the link rates. *Given the knowledge of the traffic matrix* (commodities) among the nodes, routing algorithms are proposed that iteratively update *the fraction of traffic of each flow* that should be sent on each outgoing link in the network. They rely on splitting flows among the least weighted paths where the weight of each link is defined by its marginal link cost.

The second line of work is atomic and non-atomic congestion games in game theory [26]–[29]. In the context of routing, players are the commodities, strategy sets are the set of directed source-destination paths for the commodities, the edge cost $c_e(f_e)$ is a function of the amount of congestion f_e over edge e, and the path cost $c_p(f)$ is the sum of the cost of the links along the path p. A player i incurs a cost $c_p(f)f_p^{(i)}$ for sending $f_p^{(i)}$ amount of traffic over the path p. In the atomic games, each player must choose a single path to route its commodity, while in non-atomic games, player can distribute its commodity fractionally over the set of paths. The two versions are fundamentally different. While the atomic game in general does not admit a Nash equilibrium, the nonatomic game always has a Nash equilibrium (Wardrop equilibrium) [30]. In Wardrop equilibrium, all the paths used by a given commodity have equal cost. Moreover, it's known in non-atomic games that selfish best response moves (selfish routing) by the players iteratively converge to the Wardrop equilibrium, which is a local minimum of a potential function (network cost) $\sum_e \int_0^{f_e} c_e(x)dx$.

The third line of work is oblivious routing [31]–[33] in which routes are computed to optimize the worst-case performance over the set of traffic matrices. This ensures that the computed routes are prepared for changes in traffic demands without the need to update the routes, however this is a pessimistic point of view and may be far from optimal in relatively stable periods of traffic or stable networks [32].

While the proposed myopic algorithm in this paper is reminiscent of prior algorithms under flow splitting and non-atomic games (e.g. [23]–[25], [28]–[30]), the results in this paper are not trivially drawn from these prior work. First, unlike [23]–[25], [28]–[30] that rely on splitting flows in any granularity and rerouting them continuously to find the optimal routing, we *do not allow flow splitting and migrations*. Second, unlike [23]–[25], [28]–[30] that consider a static set of flows with known traffic demand, we are dealing with a *dynamic* version of the problem when flows arrive and depart dynamically over time and the traffic demand *is not* known. Such constraints arise in practice due to the varying nature of the traffic over time and space in datacenters as well as undesirability of packet reordering in flow splitting. Our technical approach relies on a careful analysis of the fluid limits of the system under the myopic policy (without flow splitting) and proof of convergence to an invariant set which is the set of optimal flow assignments in steady state. Under unsplittable flows, the fluid limits are not continuously differentiable which poses a significant technical challenge. Intuitively, as the number of flows in the system grows, the difference between the optimal expected network cost under unsplittable flow assignment and that under splittable flow assignment should vanish in the performance ratio. We rigorously establish this intuition, and further, present deterministic and randomized algorithms with low complexity which perform very well in practice.

Finally, 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 [2], [34]–[36]. The weight construct in the algorithms proposed in this paper can provide an approach to optimally accommodate dynamic variations in datacenter network traffic in centralized control platforms such as OpenFlow [34].

B. Contributions

The main contributions of this paper can be summarized as follows.

- **Asymptotic optimality of a myopic algorithm.** We propose and analyze a simple flow scheduling algorithm to minimize the average network cost (the sum of convex functions of link utilizations). Specifically, we propose a myopic algorithm that assigns every arriving flow to an available path with the minimum marginal cost (i.e., the path which yields the minimum increase in the network cost after assignment). We prove that this simple myopic algorithm is asymptotically optimal in *any* network topology, in the sense that *the performance ratio between the average network cost under the myopic algorithm and the optimal cost approaches* 1 *as the mean number of flows in the system increases*. The myopic 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.

- **A low complexity randomized algorithm.** We also propose randomized versions of our myopic algorithm which have much lower complexity. In the randomized algorithm with parameter $k \geq 2$, instead of considering all the available paths upon arriving of a flow, k paths are chosen at random and then the flow is assigned to the path with the minimum marginal cost among these k paths. Similar to the myopic algorithm, randomized versions do not rely on flow splitting, flow migration/rerouting, or the knowledge of the traffic pattern. We empirically investigate the effect of parameter k on the algorithm performance.
- **Empirical evaluation of the algorithms.** We evaluate our myopic algorithm and its randomized versions under various workload and network topologies. For the flow generation, we consider two traffic models: (i) Poisson arrival of flows with exponentially distributed durations, and (ii) based on data from empirical studies of datacenter traffic. For the network topology, we consider FatTree (a highly structured topology), and JellyFish (a random topology). Our empirical results show that the myopic algorithm in fact performs very well under a wide range of traffic conditions in both datacenter topologies. Further, the randomized algorithms can perform very well by choosing the proper parameter k (the number of randomly chosen paths), in particular in symmetric network topologies (like FatTree) small values of k will suffice.

C. Notations

Given a sequence of random variables $\{X_n\}$, $X_n \Rightarrow X$ indicates convergence in distribution, and $X_n \rightarrow X$ 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.

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, in FatTree [2] (Figure 1) only the edge (top-of-the-rack) switches are connected to servers, while in JellyFish [10] (Figure 2) 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.

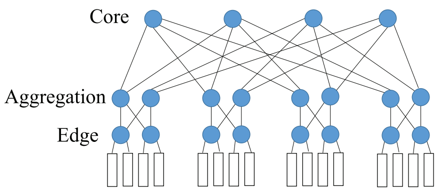

Fig. 1. FatTree connecting 16 servers (rectangles) using 4-port switches (circles).

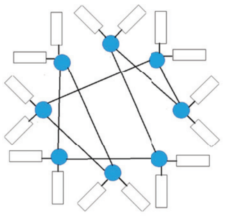
Fig. 2. JellyFish (random graph) connecting 16 servers (rectangles) using 4-port switches (circles).

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 $1/\mu_j$. It is possible to extend our results to a more general model of flow arrival and service time, e.g., when the arrival process is a "renewal" process and service time distribution has lower bounded "hazard rate", using a similar approach as in [37]. We will also report simulation results in Section V that show that our myopic algorithm indeed 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 available 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). Note that R_j could be the set of all possible paths from a_j to d_j or a subset of them as desired by the network operator. We 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

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

The online (Markov) scheduling algorithm determines the path where an arriving flow at time t is placed, as a function of the current network state $Y(t)$.

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,

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

where by $l \in i$ we mean 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.

Note that under any Markov scheduling algorithm, the network state $\{Y(t)\}_{t \geq 0}$ is a continuous-time, irreducible Markov chain. It is also positive recurrent, because the total number of type-j flows $X^{(j)}(t)$ in the system is a Markov chain independent of the scheduling algorithm, and its stationary distribution is Poisson with mean ρ_j. Therefore, the process $\{Y(t)\}_{t \geq 0}$ has a unique stationary distribution as $t \to \infty$.

C. Problem Formulation

For the purpose of load balancing, the network can attempt to optimize different objectives [38] such as minimizing the maximum link congestion in the network or minimizing the sum of link costs where each link cost is a convex function of the link congestion (e.g. this could be a link latency measure [39]). Under both objectives, the traffic needs to be distributed and balanced among the feasible paths in the network, which is essential for maintaining low end-to-end delay for different flows. 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 $g(Z_l)$ to be the cost of link 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, $F(Y(t)) = \sum_{l \in E} g(Z_l(t))$. We consider polynomial cost functions of the form

$$g(x) = \frac{x^{1+\alpha}}{1+\alpha}, \ \alpha > 0, \tag{4}$$

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 congestion in the network.[2]

III. Algorithm Description

In this section, we describe our myopic 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).

Algorithm 1 Myopic 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:
         $$w_i^{(j)}(Y(t)) = \sum_{l \in i} \Delta_l^{(j)}(Y(t)),$$        (5)
     • Differential form:
         $$w_i^{(j)}(Y(t)) = \sum_{l \in i} \delta_l^{(j)}(Y(t)).$$        (6)
2: Place the flow on a path $i$ such that
         $$i = \arg\min_{k \in R_j} w_k^{(j)}(Y(t)).$$                     (7)
     Break ties in (7) uniformly at random.
```

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:*

$$\Delta_l^{(j)}(Y(t)) = g\Big(Z_l(t) + s_j\Big) - g\Big(Z_l(t)\Big). \tag{8}$$

- *Differential form:*

$$\delta_l^{(j)}(Y(t)) = s_j g'\big(Z_l(t)\big). \tag{9}$$

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

[2]Here we have considered identical links for simplicity but the analysis is easily extendable to the case that $g(\cdot)$ is a function of x/c_l where c_l is the link capacity, or the case that each link has a weight and the goal is to minimize the weighted summation of the link costs.

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

$$F(Y(t^+)) - F(Y(t)) = \sum_{l \in i} \left[g\big(Z_l(t) + s_j\big) - g\big(Z_l(t)\big) \right] = \sum_{l \in i} \Delta_l^{(j)}(Y(t)). \tag{10}$$

Similarly, based on the differential marginal costs, we have

$$\frac{\partial F(Y(t))}{\partial Y_i^{(j)}(t)} = \sum_{l \in i} s_j g'\big(Z_l(t)\big) = \sum_{l \in i} \delta_l^{(j)}(Y(t)). \tag{11}$$

Algorithm 1 describes our myopic flow assignment algorithm that places the newly generated flow on a path that minimizes the increase in the network cost based on either forms (10) or (11). 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 (5) and (6) are essentially identical in our asymptotic performance analysis in the next section, however it seems slightly easier to work with the differential form (6). 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.

Remark 1: Note that in Algorithm 1 the flow is assigned to a path with the minimum path marginal cost. The path with the minimum path marginal cost is not necessarily the same as the path with the minimum end-to-end congestion (sum of link congestions in the path).

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 Algorithm 1. 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 Algorithm 1 to the optimal cost. We deliberately defer the rigorous claims and proofs about the fluid limits to Section VII 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 $r\lambda_j$, $j \in \mathcal{J}$ (therefore, a system with a larger r would experience heavier traffic), 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, and might not be continuously differentiable; nevertheless, they satisfy the following set of differential equations. We state the result as the following lemma whose proof can be found in Section VII.

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

$$\frac{\mathrm{d}}{\mathrm{d}t} y_i^{(j)}(t) = \lambda_j p_i^{(j)}(y(t)) - \mu_j y_i^{(j)}(t) \tag{12a}$$

$$p_i^{(j)}(y(t)) = 0 \textit{ if } i \notin \arg\min_{k \in R_j} w_k^{(j)}(y(t)) \tag{12b}$$

$$p_i^{(j)}(y(t)) \geq 0, \quad \sum_{i \in R_j} p_i^{(j)}(y(t)) = 1 \tag{12c}$$

$$w_i^{(j)}(y(t)) = \sum_{l \in i} s_j g'(z_l(t)). \tag{12d}$$

Equation (12a) 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 (12b) follows from (7) 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 (12a) and (12c) 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,

$$\frac{\mathrm{d}}{\mathrm{d}t} x^{(j)}(t) = \lambda_j - \mu_j x^{(j)}(t), \quad \forall j \in \mathcal{J}, \tag{13}$$

which clearly implies that

$$x^{(j)}(t) = \rho_j + (x^{(j)}(0) - \rho_j) e^{-\mu_j t} \; \forall j \in \mathcal{J}. \tag{14}$$

Consequently at steady state,

$$x^{(j)}(\infty) = \rho_j, \quad \forall j \in \mathcal{J}, \tag{15}$$

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 myopic algorithm. First note that by (15), the values of $y(\infty)$ are confined to a convex compact set Υ defined below

$$\Upsilon \equiv \{y = (y_i^{(j)}) : y_i^{(j)} \geq 0, \sum_{i \in R_j} y_i^{(j)} = \rho_j, \forall j \in \mathcal{J}\}. \tag{16}$$

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

$$\begin{aligned} \min \quad & F(y) \\ \text{s. t.} \quad & y \in \Upsilon \end{aligned} \tag{17}$$

Denote by $\Upsilon^\star \subseteq \Upsilon$ the set of optimal solutions to the optimization (17). The following proposition states that the fluid limits of Algorithm 1 indeed converge to an optimal solution of the optimization (17).

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

$$d(y(t), \Upsilon^\star) \to 0. \tag{18}$$

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,

$$\lim_{r\to\infty} \frac{\mathbb{E}\Big[F(Y^r(\infty))\Big]}{\mathbb{E}\Big[F(Y^r_{opt}(\infty))\Big]} = 1. \tag{19}$$

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 myopic algorithm and the optimal one), the mean number of flows in the system in steady state is $O(r)$. 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 VII.

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

$$w_i^{(j)}(y) = \eta_j,\ \forall i \in \Gamma_j, \tag{20a}$$

$$w_i^{(j)}(y) \geq \eta_j,\ \forall i \in R_j \setminus \Gamma_j, \tag{20b}$$

where $w_i^{(j)}(\cdot)$ defined in (12d).

Proof of Lemma 2: Consider the following optimization problem,

$$\min \quad F(y) \tag{21a}$$

$$\text{s.t.} \quad \sum_{i\in R_j} y_i^{(j)} \geq \rho_j, \quad \forall j \in \mathcal{J} \tag{21b}$$

$$y_i^{(j)} \geq 0, \quad \forall j \in \mathcal{J},\ \forall i \in R_j. \tag{21c}$$

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 (17) has the same set of optimal solutions as the optimization (21). 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.

$$L(\eta, \nu, y) = F(y) + \sum_{j\in\mathcal{J}} \eta_j(\rho_j - \sum_{i;i\in R_j} y_i^{(j)}) - \sum_{j\in\mathcal{J}}\sum_{i;i\in R_j} \nu_i^{(j)} y_i^{(j)}. \tag{22}$$

From KKT conditions [40], $y \in \Upsilon^\star$, if and only if there exist vectors η and ν such that the following holds. Feasibility:

$$y \in \Upsilon, \tag{23a}$$

$$\eta_j \geq 0, \quad \nu_i^{(j)} \geq 0\ \forall j \in \mathcal{J},\ i \in R_j, \tag{23b}$$

Complementary slackness:

$$\eta_j(\rho_j - \sum_{i;i\in R_j} y_i^{(j)}) = 0, \quad \forall j \in \mathcal{J}, \tag{24a}$$

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

Stationarity:

$$\frac{\partial L(\eta, \nu, y)}{\partial y_i^{(j)}} = 0.\ \forall j \in \mathcal{J}, \quad i \in R_j. \tag{25a}$$

Note that (23a) implies (24a). It follows from (25a) that

$$\frac{\partial F(y)}{\partial y_i^{(j)}} = \eta_j + \nu_i^{(j)}, \quad \forall j \in \mathcal{J},\ i \in R_j. \tag{26}$$

Define Γ_j as in the statement of the lemma. Note that Γ_j is nonempty for all $j \in \mathcal{J}$ by (23a). Then combining (24b) and (26), $\forall j \in \mathcal{J}$, and noting that $\frac{\partial F(y)}{\partial y_i^{(j)}} = w_i^{(j)}(y)$ by definition, yields (20a)-(20b). □

Next, we show that the set of optimal solutions $\Upsilon^\star$ is an invariant set of the fluid limits, using the fluid limit equations (12a)-(12d), 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 (12b), therefore

$$\frac{\mathrm{d}}{\mathrm{d}t}\Big(\sum_{i\in I_i^{(j)}(t)} y_i^{(j)}(t)\Big) = \lambda_j - \Big(\sum_{i\in I^{(j)}(t)} y_i^{(j)}(t)\Big)\mu_j. \tag{27}$$

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 (27) has a unique solution for $\sum_{i\in I^{(j)}(t)} y_i^{(j)}(t)$ which is

$$\sum_{i\in I^{(j)}(t)} y_i^{(j)}(t) = \rho_j,\ t \geq 0. \tag{28}$$

On the other hand, since $x^{(j)}(0) = \rho_j$, by (14),

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

Equations (28) and 29 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

$$d(y(t), \Upsilon^\star) \to 0. \tag{30}$$

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

Proof of Lemma 4: Starting from $y(0) \in \Upsilon$, (14) implies that

$$x^{(j)}(t) = \sum_{i \in R_j} y_i^{(j)}(t) = \rho_j \ \forall j \in \mathcal{J}, \tag{31}$$

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

$$\begin{aligned}
&(\mathrm{d}/\mathrm{d}t)F(y(t)) \\
&= \sum_{j \in \mathcal{J}} \sum_{i \in R_j} \frac{\partial F(y)}{\partial y_i^{(j)}} \frac{\mathrm{d}y_i^{(j)}(t)}{\mathrm{d}t} \\
&= \sum_{j \in \mathcal{J}} \mu_j \big[\rho_j \sum_{i \in R_j} w_i^{(j)}(y(t)) p_i^{(j)}(t) - \sum_{i \in R_j} w_i^{(j)}(y(t)) y_i^{(j)}(t)\big] \\
&\overset{(a)}{=} \sum_{j \in \mathcal{J}} \mu_j \big[\rho_j \eta_j(y(t) - \sum_{i \in R_j} w_i^{(j)}(y(t)) y_i^{(j)}(t)\big] \\
&\overset{(b)}{<} \sum_{j \in \mathcal{J}} \mu_j \big[\rho_j \eta_j(y(t)) - \eta_j(y(t)) \sum_{i \in R_j} y_i^{(j)}(t)\big] \overset{(c)}{=} 0.
\end{aligned} \tag{32}$$

Equality (a) follows from the fact that $p_i^{(j)}(t) = 0$ if $w_i^{(j)}(t) > \eta_j(t)$, and $\sum_{i \in I^{(j)}(t)} p_i^{(j)}(t) = 1$, $t \geq 0$, by (12b) and (12c). 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 (31). □

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 $(\mathrm{d}/\mathrm{d}t)F(y(t))$ (as given by (32)) 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,

$$(\mathrm{d}/\mathrm{d}t)F(y(t))\big|_{y(t)=a} \leq -\epsilon_2 \tag{33}$$

By the continuity of $(\mathrm{d}/\mathrm{d}t)F(y(t))$ in $y(t)$, there exists a $\delta > 0$ such that $\|y(t) - a\| \leq \delta$ implies,

$$|(\mathrm{d}/\mathrm{d}t)F(y(t)) - (d/dt)F(a)| \leq \epsilon_2/2 \tag{34}$$

Combining (33) and (34), for all $y(t)$ such that $\|y(t) - a\| \leq \delta$,

$$(\mathrm{d}/\mathrm{d}t)F(y(t)) \leq -\epsilon_2/2.$$

By (14), 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 $(\mathrm{d}/\mathrm{d}t)F(y(t)) \leq -\epsilon_2/2 < 0$. Applying Lyapunov argument with $F(.)$ as Lyapunov function completes the proof of Proposition 1. □

V. Simulation Results

In this section, we provide simulation results and evaluate the performance of Algorithm 1 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 1 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 2 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).

For the 128-server FatTree, when source and destination switches are located in different (same) racks, our myopic algorithm considers 16 (4) equal length candidate paths. For the case of d-regular random graphs (where each node has d edges), the number of paths between 2 switches can be very large which could significantly increase the computational complexity of the algorithm. To reduce the computation overhead, we can neglect the long paths since such paths will naturally have large marginal costs and will not be used by Algorithm 1. In our simulations, for the case of JellyFish, we consider (at most) the first 20 shortest paths (in terms of the number of links) for each pairs of 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 Algorithm 1 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 to simulate a more dynamic range of flow durations. The flow sizes are chosen according to a log-normal distribution.

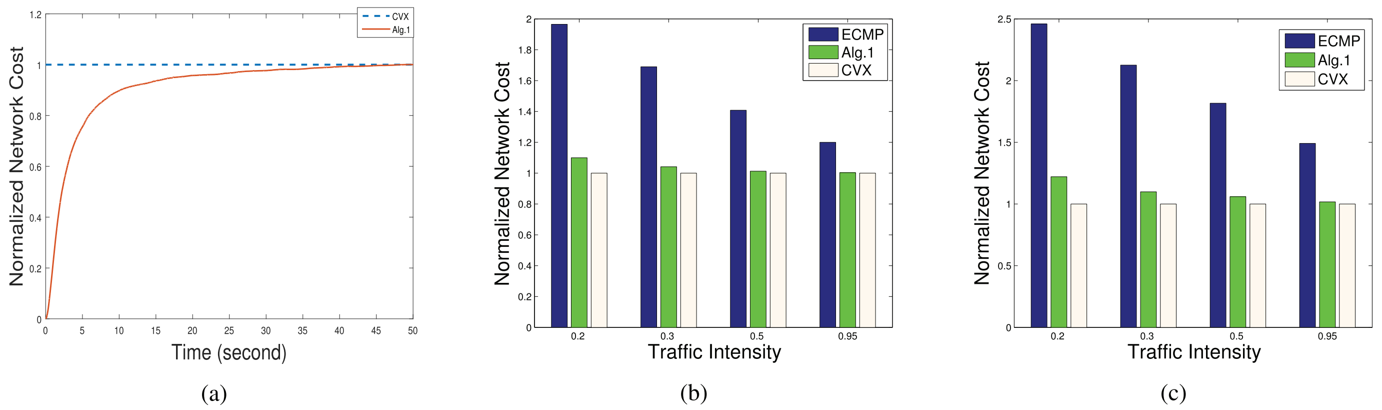

Fig. 3. Experimental Results for FatTree. **(a)**: Convergence of the network cost under Algorithm 1, normalized with the lower-bound on the optimal solution (CVX), to 1. The scaling parameter r is 100 here. **(b)** and **(c)**: Performance ratio of Algorithm 1 and ECMP in FatTree, normalized with the lower-bound (CVX) for exponential and empirical traffic models.

- *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 [41], service times based on the empirical result in [11], and log-normal flow sizes [41]. 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 [11].

In both models, the flow sizes are log-normal with mean 1.2 and standard deviation 0.4. This generates flow sizes ranging from 1% to 40% of link capacity with high probability to capture 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 these 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 Algorithm 1 and a benchmark algorithm (similar to (19)). Since the optimal algorithm (e.g. 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) is hard to implement (and even unknown), instead we use a convex relaxation method to find a lower-bound on the optimal cost at each time. We note that, for FatTree topology, equal splitting of every flow among its candidate paths is optimal. For JellyFish topology, every time a flow arrives or departs, we use CVX [42], 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 Algorithm 1 and traditional ECMP (which statically assigns flows to the shortest paths (in number of links) via flow hashing.), 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 3a 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 3b and 3c 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 myopic algorithm is very close to the lower-bound on the optimal value (CVX) for light, medium, and high traffic intensities. As it is shown, the performance improves at higher traffic intensities which correspond to larger values of r in Theorem 1. They also suggest that Theorem 1 holds under more general arrival and service time processes. In this simulations, Algorithm 1 gave a performance improvement ranging form 50% to more than 100%, compared to ECMP, depending on the traffic intensity, under the empirical traffic model. The standard deviation (SD) of performance ratio for 30 different runs ranges from 0.14 to 0.01 for Algorithm 1, and from 0.3 to 0.03 for ECMP as traffic intensity grows.

B. Experimental Results for JellyFish

Figure 4a shows that the aggregate cost under Algorithm 1 indeed converges to the optimal solution which again verifies Theorem 1. Figures 4b and 4c 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 [43] and [44] for regular random graphs. Again we see that our myopic algorithm performs very well in all light, medium, and high traffics. In JellyFish, Algorithm 1 yields performance gains ranging from 60% to 70%, compared to ECMP, under the empirical traffic model. Corresponding SD for 30 different runs ranges from 0.04 to 0.01 for Algorithm 1, and from 0.1 to 0.05 for ECMP as traffic intensity grows.

VI. Randomized Myopic Algorithms

Algorithm 1 needs to consider all the available paths for an arriving flow and finds the shortest path based on the

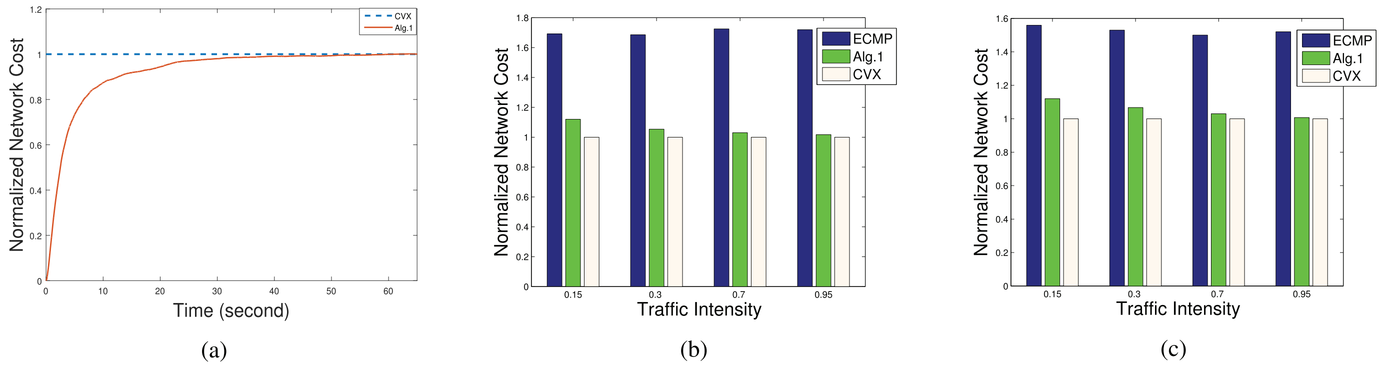

Fig. 4. Experimental Results for JellyFish. **(a)**: Convergence of the network cost under Algorithm 1 in JellyFish, normalized with the lower-bound on the optimal solution (CVX), to 1. The scaling parameter r is 100 here. **(b)** and **(c)**: Performance ratio of Algorithm 1 and ECMP in JellyFish, normalized with the lower-bound (CVX) for exponential and empirical traffic models.

(integral (5) or differential (6)) marginal cost of paths. In this section, we describe and empirically evaluate randomized versions of our myopic algorithm which have less complexity than Algorithm 1, while can effectively provide a large fraction of the performance gain obtained by Algorithm 1. Our approach is motivated by the literature on *randomized load balancing* for scheduling jobs in servers, where a widely used idea is that, instead of considering all the servers and assigning the arriving job to the least-loaded server, k servers are first chosen at random (for some $k \geq 2$) and then the job is assigned to the least-loaded server among them. This idea was originally proposed in [45], where it was shown that having $k = 2$ leads to exponential improvement in the expected time a job spends in the system over $k = 1$ which is basically the totally random assignment.

In our setting, a counterpart of this approach can be used for scheduling of flows in paths as follows. Fix k, when a flow is generated, the algorithm chooses k paths at random out of the available paths for the flow, then calculates the marginal costs of these k paths according to the integral or the differential form formulas, and assigns the flow to the path with the minimum path marginal cost among these k paths. See Algorithm 2 for the full description.

We notice that ECMP in structured topologies like FatTree, where all candidate paths for an arriving flow have the same number of links (same length), is basically the random assignment of flows to the paths which is identical to setting $k = 1$ in Algorithm 2.

Next, we empirically evaluate the performance of Algorithm 2 for different values of k. We present the results for two different topologies and two traffic model as in Section V. For JellyFish, we consider (at most) the first 20 shortest paths (in terms of the number of links) for each pairs of switches to be consistent with Section V.

Algorithm 2 Randomized Myopic Algorithm with Parameter k

```
Suppose a type-$j$ flow arrives at time $t$ when the system is in state $\mathbf{Y}(t)$. Then,
1: Choose $k$ paths from the set $|R_j|$, uniformly at random, let
     $R_j^{(k)}$ denotes this subset of paths.
2: Compute the path marginal costs $w_i^{(j)}(Y(t))$, $i \in R_j^{(k)}$,
     in either of the forms below:
     • Integral form:
         $$w_i^{(j)}(Y(t)) = \sum_{l \in i} \Delta_l^{(j)}(Y(t)), \quad (35)$$
     • Differential form:
         $$w_i^{(j)}(Y(t)) = \sum_{l \in i} \delta_l^{(j)}(Y(t)). \quad (36)$$
3: Place the flow on a path $i$ such that
         $$i = \arg \min_{k \in R_j^{(k)}} w_k^{(j)}(Y(t)). \quad (37)$$
     Break ties in (37) uniformly at random.
```

A. Experimental Results for FatTree

Figures 5 and 6 show the cost performance under Algorithm 2 with different values of k, normalized by the cost of Algorithm 1, under the exponential and the empirical traffic models respectively. Note that Algorithm 2 with $k = 16$ is equivalent to Algorithm 1, as there are at most 16 available paths for an arriving flow in the FatTree topology we described in Section V. Error bars in all plots correspond to standard deviation of normalized mean network cost computed from results of 30 runs.

In these two plots, we can see that the maximum improvement in network cost we get by increasing k happens at $k = 2$ compared with random assignment of flows, $k = 1$. Furthermore, as we increase value of k we get smaller improvement in performance. For instance, normalized cost improves about 0.4 by increasing k from 1 to 2, while the improvement from $k = 2$ to $k = 4$ is about 0.1, for traffic intensity equal to 0.3 under exponential model (Figure 5). This behavior is seen in both figures, and is more profound for higher traffic intensity.

B. Experimental Results for JellyFish

Figures 7 and 8 show the network cost under Algorithm 2 with different values of k, normalized by the cost

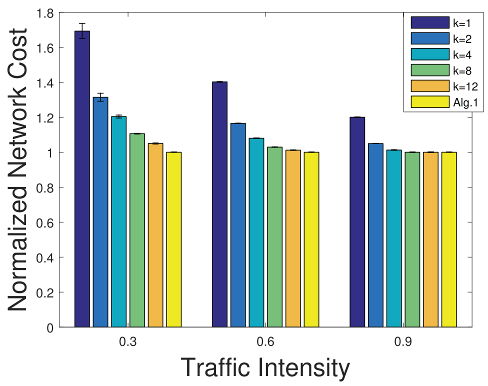

Fig. 5. Performance of Algorithm 2 with diffrent values of k, in FatTree under the exponential traffic model, normalized with the Algorithm 1.

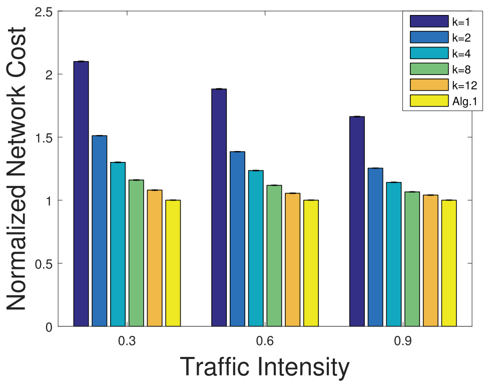

Fig. 6. Performance of Algorithm 2 with diffrent values of k, in FatTree under the empirical traffic model, normalized with the Algorithm 1.

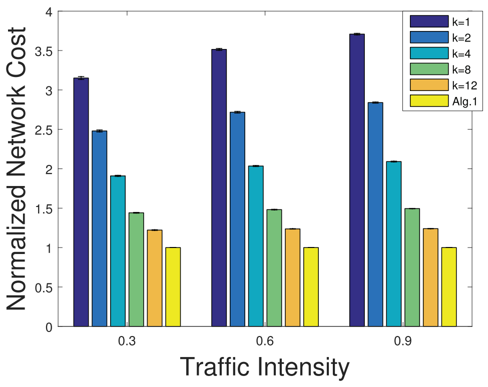

Fig. 7. Performance of Algorithm 2 with different values of k in JellyFish under the exponential traffic model, normalized with the Algorithm 1.

of Algorithm 1, under the exponential and the empirical traffic models respectively. Note that Algorithm 2 with $k = 20$ is equivalent to Algorithm 1, as there are at most 20 available paths considered between any two switches in the JellyFish topology we described in Section V.

In these figures, we observe the same behavior as what discussed for FatTree: the performance improvement obtained by increasing k by one is larger for smaller k. Also, comparing Figures 7 and 8 with Figures 4b and 4c, in order for Algorithm 2 to beat ECMP–which only considers shortest paths (in the terms of the number of links)–we need to choose $k \geq 12$.

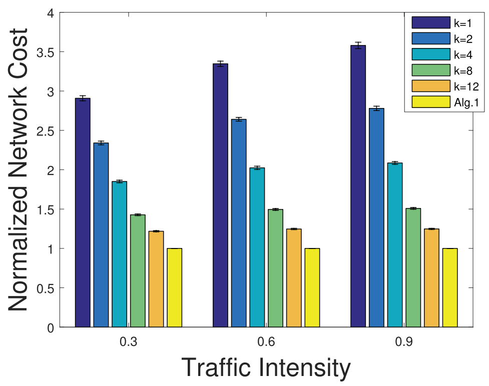

Fig. 8. Performance of Algorithm 2 with different values of k in JellyFish under the empirical traffic model, normalized with the Algorithm 1.

We also note that in JellyFish, for small k (e.g., $k = 1, 2$), the normalized cost under the randomized algorithm increases as traffic intensity grows, unlike the results for FatTree. This can be justified by noting that the symmetric structure of FatTree allows random assignment of flows to balance the load better as traffic intensity increases (higher flow arrival rates) because the number of flow-to-path assignment decisions increases. However, in JellyFish the structure is asymmetric and long paths are used more frequently by the randomized algorithm as traffic intensity increases. As a result, the convexity of the link cost function, and the fact that the network cost is the summation of all links' costs, will cause a larger network cost in higher traffic intensities.

Based on the simulations, we conclude that to get a reasonably good performance, we need smaller values of k in FatTree compared to JellyFish. This can be attributed to the fact that all the candidate paths for a flow in the FatTree topology have the same number of links, while in the JellyFish topology, paths can be very different in terms of their number of links. So selection of k paths completely at random, as used in Algorithm 2, might lead to using long paths which contribute more to the network cost. Thus, uniform sampling seems more suitable for symmetric topologies like FatTree. We postpone the exact analysis of the randomized myopic policy to a future work.

VII. Formal Proofs of Fluid Limits and Theorem 1

A. Proof of Fluid Limits

We prove the existence of fluid limits under Algorithm 1 and derive the corresponding fluid equations (12a)-(12d). Arguments in this section are quite standard [37], [46], [47]. 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

$$Y_i^{(j)^r}(t) = Y_i^{(j)^r}(0) + \Pi_{i,j}^a\Big(\int_0^t P_i^{(j)}(Y^r(s)) r\lambda_j ds\Big) - \Pi_{i,j}^d\Big(\int_0^t \mu_j Y_i^{(j)^r}(s)ds\Big), \quad \forall j \in \mathcal{J},\ i \in R_j \tag{38}$$

where $\Pi^a_{i,j}(.)$ and $\Pi^d_{i,j}(.)$ are independent unit-rate Poisson processes, and $P^{(j)}_i(Y^r(t))$ 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 [48], almost surely,

$$\frac{1}{r}\Pi^a_{i,j}(rt) \to t, \text{ u.o.c.}; \ \frac{1}{r}\Pi^d_{i,j}(rt) \to t, \text{ u.o.c.} \tag{39}$$

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

$$a^r_{i,j}(t) = \frac{1}{r}\Pi^a_{i,j}\Big(\int_0^t P^{(j)}_i(Y^r(s))r\lambda_j ds\Big),$$
$$d^r_{i,j}(t) = \frac{1}{r}\Pi^d_{i,j}\Big(\int_0^t \mu_j Y^{(j)^r}_i(s)ds\Big). \tag{40}$$

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 of Lemma 5: The proof is standard and follows from the fact that $a^r_{i,j}(.)$ and $d^r_{i,j}(.)$ are asymptotically Lipschitz continuous (see e.g., [37], [46], [49] for similar arguments), namely, there exists a constant $C > 0$ such that for $0 \le t_1 \le t_2 < \infty$,

$$\limsup_r (a^r_{i,j}(t_2) - a^r_{i,j}(t_1)) \le C(t_2 - t_1), \tag{41}$$

and similarly for $d^r_{i,j}(.)$. More precisely, for arrival process $a^r_{i,j}(.)$, we argue that,

$$\begin{aligned}&\limsup_r (a^r_{i,j}(t_2) - a^r_{i,j}(t_1))\\ &= \limsup_r \frac{1}{r}\Pi^a_{i,j}\Big(\int_{t_1}^{t_2} P^{(j)}_i(Y^r(s))r\lambda_j ds\Big)\\ &\overset{(a)}{\le} \limsup_r \frac{1}{r}\Pi^a_{i,j}\Big(\int_{t_1}^{t_2} r\lambda_j ds\Big)\\ &= \limsup_r \Big(\frac{1}{r}\Pi^a_{i,j}(r\lambda_j(t_2 - t_1))\Big)\end{aligned}$$

where inequality (a) follows from the fact that $P^{(j)}_i(Y^r(s)) \le 1$. Using (39), we obtain (41). The argument is similar for $d^r_{i,j}(.)$, 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 (38), (40), (39), and the existence of the fluid limits (Lemma 5), that

$$y^{(j)}_i(t) = y^{(j)}_i(0) + a^{(j)}_i(t) - d^{(j)}_i(t),$$

where $d^{(j)}_i(t) = \int_0^t y^{(j)}_i(s)\mu_j ds$, and $\sum_{i\in R_j} a^{(j)}_i(t) = \lambda_j t$, $a^{(j)}_i(t)$ is nondecreasing. The fluid equations (12a) and (12c) are the diffrential form of these equations (the fluid sample paths are Lipschitz continuous so the derivatives exist almost everywhere), where

$$p^{(j)}_i(t) := \frac{1}{\lambda_j}\frac{\mathrm{d}a^{(j)}_i(t)}{\mathrm{d}t}. \tag{42}$$

For any type j, and for $w^{(j)}_i(y(t))$ defined in (12d), let

$$w^\star_j(y(t)) = \min_{i\in R_j} w^{(j)}_i(y(t)).$$

Consider any regular time t and a path $i \notin \arg\min_{i\in R_j} w^{(j)}_i(y(t))$. By the continuity of $w^{(j)}_i(y(t))$, there must exist a small time interval (t_1, t_2) containing t such that

$$w^{(j)}_i(y(\tau)) > w^\star_j(\tau) \ \ \forall \tau \in (t_1, t_2).$$

Consequently, for all r large enough along the subsequence,

$$w^{(j)}_i(y^r(\tau)) > w^\star_j(y^r(\tau)) \ \ \forall \tau \in (t_1, t_2).$$

Multiplying both sides by r^α, it follows that

$$w^{(j)}_i(Y^r(\tau)) > w^\star_j(Y^r(\tau)), \ \ \forall \tau \in (t_1, t_2).$$

Hence $P^{(j)}_i(Y^r(\tau)) = 0$, $\tau \in (t_1, t_2)$, and $a^{r(j)}_i(t_1, t_2) = 0$, for all r large enough along the subsequence. Therefore $a^{(j)}_i(t_1, t_2) = 0$ which shows that $(\mathrm{d}/\mathrm{d}t)a^{(j)}_i(t) = 0$ at $t \in (t_1, t_2)$. This establishes (12b). □

B. Proof of Theorem 1

We first show that

$$F(y^r(\infty)) \Longrightarrow F^\star, \tag{43}$$

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 $\epsilon_1 > 0$, $|F(y(t_{\epsilon_1})) - F^\star| \le \epsilon_1$. With probability 1, every subsequence y^{r_n} has a further subsequence $y^{r_{n_k}}$ such that $y^{r_{n_k}}(t) \to y(t)$ u.o.c. (see Lemma 5), hence, by the continuous mapping theorem [48], we also have $F(y^{r_{n_k}}(t)) \to F(y(t))$, u.o.c. For any $\epsilon_2 > 0$, for r_{n_k} large enough, we can choose an $\epsilon_3 > 0$ such that, uniformly over all initial states $y^{r_{n_k}}(0)$ such that $\|y^{r_{n_k}}(0) - y(0)\| \le \epsilon_3$,

$$\mathbb{P}\{|F(y^{r_{n_k}}(t_{\epsilon_1}) - F(y(t_{\epsilon_1}))| < \epsilon_1\} > 1 - \epsilon_2 \tag{44}$$

This claim is true, since otherwise for a sequence of initial states $y^{r_{n_k}}(0) \to y(0)$ we have

$$\mathbb{P}\{|F(y^{r_{n_k}}(t_{\epsilon_1}) - F(y(t_{\epsilon_1}))| < \epsilon_1\} \le 1 - \epsilon_2,$$

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

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

which in particular implies

$$F(y^{r_{n_k}}(\infty)) \Longrightarrow F^\star,$$

because ϵ_1 and ϵ_2 can be made arbitrarily small. Hence, we have shown that every sequence $F(y^{r_n}(\infty))$ has a further subsequence $F(y^{r_{n_k}}(\infty))$ that converges to the same limit $F^\star$

(the unique optimal cost). Therefore in view of [48, Th. 2.6], we can conclude that $F(y^r(\infty)) \Longrightarrow F^\star$.

Next, we show (19). Under any algorithm (including Algorithm 1 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 $|E|g\big(\bar{s}X^r(\infty)/r\big)$, 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 (43), by [48, Th. 3.5], under our Algorithm 1.

$$\mathbb{E}\Big[F(Y^r(\infty)/r)\Big] \to F^\star. \tag{45}$$

Now consider any optimal algorithm for the optimization (3). It holds that

$$F(\mathbb{E}\Big[y_{\text{opt}}^r(\infty)\Big]) \le \mathbb{E}\Big[F(y_{\text{opt}}^r(\infty))\Big] \le \mathbb{E}\Big[F(y^r(\infty))\Big],$$

where the first inequality is by Jensen's inequality, and the second follows from definition of optimality. Taking the limit as $r \to \infty$, it follows by an squeeze argument that

$$\mathbb{E}\Big[F(Y_{\text{opt}}^r(\infty)/r)\Big] \to F^\star. \tag{46}$$

Finally, (45) and (46) will imply (19) in view of the polynomial structure of F.

VIII. Concluding Remarks

This paper presents a simple myopic algorithm that dynamically adjusts the link weights as a function of the link congestions 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 myopic 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 time-scale 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. Another possibility is to use the randomized versions of our myopic algorithm with an optimized parameter k which only takes a small random subset of available paths into account and finds the shortest path among them. While this algorithm has much lower complexity, it performs very well in structured topologies such as FatTree for small k. We leave theoretical analysis of the randomized versions as an open problem for future work. Finally, we would like to note that our myopic algorithm and its randomized versions can be directly applied to scheduling flowlets instead of scheduling flows, which can give higher rate/granularity of flows [6], [36].

References

[1] M. Shafiee and J. Ghaderi, "A simple congestion-aware algorithm for load balancing in datacenter networks," in *Proc. 35th Annu. IEEE Int. Conf. Comput. Commun. (INFOCOM)*, Apr. 2016, pp. 1–9.

[2] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 4, pp. 63–74, 2008.

[3] T. Benson, A. Anand, A. Akella, and M. Zhang, "MicroTE: Fine grained traffic engineering for data centers," in *Proc. 7th Conf. Emerg. Netw. Experim. Technol.*, 2011, Art. no. 8.

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

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

[6] S. Kandula, D. Katabi, S. Sinha, and A. Berger, "Dynamic load balancing without packet reordering," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 37, no. 2, pp. 51–62, 2007.

[7] A. Greenberg *et al.*, "VL2: A scalable and flexible data center network," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 4, pp. 51–62, 2009.

[8] C. Guo *et al.*, "BCube: A high performance, server-centric network architecture for modular data centers," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 4, pp. 63–74, 2009.

[9] M. Bradonjić, I. Saniee, and I. Widjaja, "Scaling of capacity and reliability in data center networks," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 42, no. 2, pp. 46–48, 2014.

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

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

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

[13] S. Even, A. Itai, and A. Shamir, "On the complexity of time table and multi-commodity flow problems," in *Proc. 16th Annu. Symp. Found. Comput. Sci.*, Oct. 1975, pp. 184–193.

[14] G. M. Guisewite and P. M. Pardalos, "Minimum concave-cost network flow problems: Applications, complexity, and algorithms," *Ann. Oper. Res.*, vol. 25, no. 1, pp. 75–99, 1990.

[15] Y. Dinitz, N. Garg, and M. X. Goemans, "On the single-source unsplittable flow problem," in *Proc. 39th Annu. Symp. Found. Comput. Sci.*, 1998, pp. 290–299.

[16] R. N. Mysore *et al.*, "PortLand: A scalable fault-tolerant layer 2 data center network fabric," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 4, pp. 39–50, 2009.

[17] J. Cao *et al.*, "Per-packet load-balanced, low-latency routing for clos-based data center networks," in *Proc. 9th ACM Conf. Emerg. Netw. Experim. Technol.*, 2013, pp. 49–60.

[18] P. Gill, N. Jain, and N. Nagappan, "Understanding network failures in data centers: Measurement, analysis, and implications," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 41, no. 4, pp. 350–361, Aug. 2011.

[19] M. Chiesa, G. Kindler, and M. Schapira, "Traffic engineering with Equal-Cost-MultiPath: An algorithmic perspective," *IEEE/ACM Trans. Netw.*, vol. 25, no. 2, pp. 779–792, Apr. 2017.

[20] S. Sen, D. Shue, S. Ihm, and M. J. Freedman, "Scalable, optimal flow routing in datacenters via local link balancing," in *Proc. 9th ACM Conf. Emerg. Netw. Experim. Technol.*, 2013, pp. 151–162.
[21] J. W. Jiang, T. Lan, S. Ha, M. Chen, and M. Chiang, "Joint VM placement and routing for data center traffic engineering," in *Proc. IEEE INFOCOM*, Mar. 2012, pp. 2876–2880.
[22] K. He *et al.*, "Presto: Edge-based load balancing for fast datacenter networks," in *Proc. ACM Conf. Special Interest Group Data Commun.*, 2015, pp. 465–478.
[23] R. Gallager, "A minimum delay routing algorithm using distributed computation," *IEEE Trans. Commun.*, vol. COM-25, no. 1, pp. 73–85, Jan. 1977.
[24] N. Michael and A. Tang, "HALO: Hop-by-hop adaptive link-state optimal routing," *IEEE/ACM Trans. Netw.*, vol. 23, no. 6, pp. 1862–1875, Dec. 2015.
[25] D. Xu, M. Chiang, and J. Rexford, "Link-state routing with hop-by-hop forwarding can achieve optimal traffic engineering," *IEEE/ACM Trans. Netw.*, vol. 19, no. 6, pp. 1717–1730, Dec. 2011.
[26] R. W. Rosenthal, "A class of games possessing pure-strategy Nash equilibria," *Int. J. Game Theory*, vol. 2, no. 1, pp. 65–67, 1973.
[27] N. Nisan, T. Roughgarden, E. Tardos, and V. V. Vazirani, *Algorithmic Game Theory*, vol. 1. Cambridge, U.K.: Cambridge Univ. Press, 2007.
[28] T. Roughgarden, *Selfish Routing and the Price of Anarchy*, vol. 174. Cambridge, MA, USA: MIT Press, 2005.
[29] P. Key, L. Massoulié, and D. Towsley, "Path selection and multipath congestion control," in *Proc. IEEE 26th IEEE Int. Conf. Comput. Commun. (INFOCOM)*, May 2007, pp. 143–151.
[30] J. G. Wardrop, "Road paper. Some theoretical aspects of road traffic research," *Proc. Inst. Civil Eng.*, vol. 1, no. 3, pp. 325–362, 1952.
[31] D. Applegate and E. Cohen, "Making intra-domain routing robust to changing and uncertain traffic demands: Understanding fundamental tradeoffs," in *Proc. Conf. Appl., Technol., Architectures, Protocols Comput. Commun.*, 2003, pp. 313–324.
[32] H. Wang *et al.*, "COPE: Traffic engineering in dynamic networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 36, no. 4, pp. 99–110, 2006.
[33] M. Bienkowski, M. Korzeniowski, and H. Räcke, "A practical algorithm for constructing oblivious routing schemes," in *Proc. 15th Annu. ACM Symp. Parallel Algorithms Architectures*, 2003, pp. 24–33.
[34] N. McKeown *et al.*, "OpenFlow: Enabling innovation in campus networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, Apr. 2008.
[35] M. Casado *et al.*, "Rethinking enterprise network control," *IEEE/ACM Trans. Netw.*, vol. 17, no. 4, pp. 1270–1283, Aug. 2009.
[36] M. Alizadeh *et al.*, "CONGA: Distributed congestion-aware load balancing for datacenters," in *Proc. ACM Conf. SIGCOMM*, 2014, pp. 503–514.
[37] A. L. Stolyar, "An infinite server system with general packing constraints," *Oper. Res.*, vol. 61, no. 5, pp. 1200–1217, 2013.
[38] M. Chiesa, G. Kindler, and M. Schapira, "Traffic engineering with Equal-Cost-MultiPath: An algorithmic perspective," in *Proc. IEEE INFOCOM*, Apr. 2014, pp. 1590–1598.
[39] B. Fortz and M. Thorup, "Internet traffic engineering by optimizing OSPF weights," in *Proc. 19th Annu. Joint Conf. IEEE Comput. Commun. Soc. (INFOCOM)*, vol. 2. Mar. 2000, pp. 519–528.
[40] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.
[41] D. Ersoz, M. S. Yousif, and C. R. Das, "Characterizing network traffic in a cluster-based, multi-tier data center," in *Proc. 27th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, 2007, p. 59.
[42] M. Grant and S. Boyd. (Mar. 2014). *CVX: MATLAB Software for Disciplined Convex Programming, Version 2.1*. [Online]. Available: http://cvxr.com/cvx
[43] J. Díaz, M. J. Serna, and N. C. Wormald, "Bounds on the bisection width for random d-regular graphs," *Theor. Comput. Sci.*, vol. 382, no. 2, pp. 120–130, 2007.
[44] B. Bollobás, "Random graphs," in *Modern Graph Theory*. New York, NY, USA: Springer, 1998, pp. 215–252.
[45] M. Mitzenmacher, "The power of two choices in randomized load balancing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 12, no. 10, pp. 1094–1104, Oct. 2001.
[46] A. L. Stolyar and Y. Zhong, "Asymptotic optimality of a greedy randomized algorithm in a large-scale service system with general packing constraints," *Queueing Syst.*, vol. 79, no. 2, pp. 117–143, 2015.
[47] J. Ghaderi, Y. Zhong, and R. Srikant, "Asymptotic optimality of BestFit for stochastic bin packing," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 42, no. 2, pp. 64–66, 2014.
[48] P. Billingsley, *Convergence of Probability Measures*, 2nd ed. New York, NY, USA: Wiley, 1999.
[49] S. N. Ethier and T. G. Kurtz, *Markov Processes: Characterization and Convergence*, vol. 282. Hoboken, NJ, USA: Wiley, 2009.

Mehrnoosh Shafiee received the B.Sc. degree from the EE Department, Sharif University of Technology, Tehran, Iran. She is currently pursuing the joint M.Sc. and Ph.D. degrees with the Department of Electrical Engineering, Columbia University, since 2014. She is interested in the analysis and design of resource allocation algorithms for large-scale distributed systems.

Javad Ghaderi received the B.Sc. degree from the University of Tehran, Iran, in 2006, the M.Sc. degree from the University of Waterloo, Canada, in 2008, and the Ph.D. degree from the University of Illinois at Urbana–Champaign (UIUC) in 2013, all in electrical and computer engineering. He spent a one-year Simons Postdoctoral Fellowship with The University of Texas at Austin. He joined the Department of Electrical Engineering, Columbia University, in 2014. His research interests include network algorithms and network control and optimization. He was recipient of the Mac Van Valkenburg Graduate Research Award, UIUC, the Best Student Paper Finalist at the 2013 American Control Conference, the Best Paper Award at the CoNEXT 2016, and the NSF CAREER award in 2017.