On the effect of forwarding table size on SDN network utilization

Rami Cohen
IBM Haifa Research Lab.
ramic@il.ibm.com

Liane Lewin-Eytan
IBM Haifa Research Lab.
lianel@il.ibm.com

Joseph (Seffi) Naor
Computer Science Dept., Technion
naor@cs.technion.ac.il

Danny Raz
Computer Science Dept., Technion
danny@cs.technion.ac.il

***Abstract*—Software Defined Networks (SDNs) are becoming the leading technology behind many traffic engineering solutions, both for backbone and data-center networks, since it allows a central controller to globally plan the path of the flows according to the operator's objective. Nevertheless, networking devices' forwarding table is a limited and expensive resource (e.g., TCAM-based switches) which should thus be considered upon configuring the network. In this paper, we concentrate on satisfying global network objectives, such as maximum flow, in environments where the size of the forwarding table in network devices is limited. We formulate this problem as an (NP-hard) optimization problem and present approximation algorithms for it. We show through extensive simulations that practical use of our algorithms (both in Data Center and backbone scenarios) result in a significant reduction (factor 3) in forwarding table size, while having a small effect on the global objective (maximum flow).**

I. Introduction

Software Defined Networks (SDNs) is a network architecture where the data forwarding behavior of network elements is determined by a centralized controller. This separation of the control plane from the data plane allows network operators to gain a fine grain control over the actual way packets are forwarded, and thus better utilize their network. For this reason SDN is becoming the leading technology behind many traffic engineering solutions both for backbone and data-center networks[1].

In the SDN paradigm, network devices are becoming simpler and cheaper, since they do not have to support complex logics of distributed routing protocols. However, in most cases, these devices should support general forwarding rules (e.g., OpenFlow [14]). Considering the growing demands for a fast and efficient data plane, these rules are implemented using expensive technology (in terms of cost and power) such as TCAM (Trenary Content Aware Memory), which implies that the number of forwarding rules, or the effective size of forwarding tables in these devices, is limited (see [16], [15]). This, in turn, imposes constraints on the control plane, since the realization of a certain global objective may require more local forwarding rules. Typically, each entry in the forwarding table is dedicated to a different flow (characterized by several parameters, according to specific header fields), and contains instructions for routing the flow over its pre-defined path. We note that even in small data centers, comprising of several dozens of physical servers, the number of different flows can easily grow to several orders of magnitude above this number, when taking into account potential pairs of connecting entities which might also correspond to VMs residing within the physical hosts. Thus, limits on the number of flows (or paths) that can pass through the forwarding devices constitute an important restriction.

In many practical scenarios, constraints on the routing paths are already deployed. Examples of such are policy-based rules such as "do not route internal US traffic through Europe", or operational-based rules such as "only use paths with latency bounded by 20 milliseconds". Thus, the goal of the control plane algorithm is to select the right set of allowed paths, such that local constraints on the size of the forwarding table in network elements are met in a way that maximizes the global objective under consideration (e.g., maximum network utilization). However, the specific parameters of the problem under hand may vary, depending on the specific technology and the specific use case (e.g., backbone TE or data center networking). We note that having local constraints on the forwarding table size is not unique to SDNs. In optical switches for example, implementing a forwarding rule requires very expensive resources, and thus the number of rules allowed for each device is very limited.

Our goal in this paper is to provide a profound understanding of the problems related to satisfying global network objectives, such as maximum flow, in environments where the size of the forwarding table in network devices is limited. To this end we define and study a general model capturing the most important aspects of SDNs, and evaluate the advantages of using our results in different practical use-cases.

In general, a flow can be sent over multiple paths, even when adhering to a particular policy, as described above. The total size of the forwarding table (or the number of forwarding rules) in a network device is then the total number of flow paths that can go through it (since each flow traversing the device requires a forwarding rule). We thus model the limited size of forwarding tables by bounding the maximum number of paths that can pass through each node in the network. This bound is called the *path-degree* or *forwarding table size* of the node. We consider a bandwidth-constrained practical setting, where the links in the network have limited bandwidth capacity.

We thus investigate the *bounded path-degree max flow problem*, where given different traffic demands in the network, the goal is to maximize the overall feasible traffic that can be routed, while satisfying the limited bandwidth capacities of the links, as well as the path-degrees of the nodes. The bounded path-degree max flow problem captures several well-

[1]See e.g., http://www-03.ibm.com/systems/networking/solutions/sdn.html, http://www.opendaylight.org/, http://www.wired.com/wiredenterprise/2012/04/going-with-the-flow-google/

978-1-4799-3360-0/14/$31.00 ©2014 IEEE

known NP-hard problems. The *disjoint paths* problem, where the goal is to determine whether a set of source-destination pairs can be connected by disjoint paths, is a special case of our problem. This can readily be verified by letting all links have unit capacity, all source-destination pairs have unit demand, and setting the path-degrees of all nodes to be equal to 1.

We provide an in-depth theoretical analysis of the bounded path-degree max flow problem, both in the case where the allowed paths are explicitly provided, and in the general case where flow can be routed on any (arbitrary) path. Our solution is derived from rounding a fractional solution to a linear formulation of the problem. We note that our linear formulation is somewhat unusual, in the sense that a feasible integral solution to the problem does not necessarily define a $\{0,1\}$ solution to the linear relaxation.

In particular, we present a bicriteria approximation algorithm guaranteed to achieve an expected total flow equal to $OPT/O(\log n)$, where OPT is a bound on the maximum flow obtained by any solution and n denotes the number of nodes in the network. When adding a *fairness* requirement, according to which the flow over each path is not allowed to consume than a $1/\log n$ fraction of the path capacity, we can prove that the bicriteria approximation algorithm achieves an expected total flow equal to $OPT/O(1)$. Our model and results are also valid for the case of weighted flow demands, where the goal is to maximize the overall weighted traffic that can be routed.

The practical importance of restricting the size of forwarding tables has recently been acknowledged by the ASIC industry[2]. Our approach is much cheaper and faster to deploy, as it uses the software capabilities of the controller, so as to overcome the hardware limitations of the network devices. The theoretical part of our work provides an in-depth understanding of these important issues, as well as several provable effective algorithms. In order to demonstrate how it can be used to address the actual problems arising in practice, we conducted a thorough simulation-based evaluation of the performance of our algorithms. We used several types of networking topologies, representing both backbone, current data centers, and future data center networks. For each of the latter topologies we generated a set of $200,000$ flows[3] and measured the size of the forwarding tables needed to route this traffic using oblivious protocols (i.e. max-flow) that do not consider the amount of available forwarding rules in each device. We then deployed a practical algorithm which is based on the theoretical algorithms we developed and measured the expected performance. Clearly, the results depend on the network topology; for backbone based topology we are able show that table size aware protocols can increase the total flow in the network by a factor of 2, and for typical data center topologies by more than 50%.

The paper is organized as follows. Section II presents related works. The theoretical analysis of the bounded path-degree max flow problem is presented in Section III. The model is formally defined in Section III-A, Sections III-B & III-C contain our approximation algorithms, and Section III-D presents the solvability of the general case of our problem. The simulation study is presented in Section IV. Finally, Section V contains the conclusions.

[2]See e.g., http://pica8.org/blogs/?p=201, http://www.mellanox.com/blog/

[3]In many practical scenarios we expect many more flows, and our results also apply to these cases.

II. Related Work

To the best of our knowledge, the path-degree max flow problem has not been investigated before. Our problem is related to different versions of flow decompositions, where the goal is to compute a flow that can be partitioned with respect to various constraints. The maximum unsplittable flow problem was introduced by Kleinberg [11], where the goal is to decide whether demands from a single source to different terminals can be each routed on a single path, so that capacity constraints are satisfied. Many variations of the problem were introduced subsequently, including unsplittable multicommodity flow, variants in which we allow more than one path per commodity, and varying optimization criteria (for recent work see [3], [12] and references therein). Another related version of this problem is presented in [10], which considers the problem of decomposing a flow into a small number of paths.

Degree-constrained flows is another set of problems related to ours, in which node bounds apply to the outdegrees (number of outgoing edges) rather than to transient paths. A special case is confluent flow, where at any node all the flow departs along a single edge. Confluent flows were investigated in [4], [5], focusing on the problem of determining confluent flows with minimum congestion. A generalization of this problem was investigated in [6], considering d-furcated flows, which are flows with a support graph of maximum outdegree d.

Finally, a traffic engineering optimization problem is presented in [1], whose motivation arises from the SDN paradigm as well. The authors of [1] consider traffic engineering in the case where a SDN controller controls only a few SDN forwarding elements in the network, and the rest of the network uses a standard routing protocol like OSPF. Their model and problem are thus completely different from ours, however our SDN-traffic engineering motivation is closely related.

III. The Bounded Path-Degree Max Flow Problem

The path-degree max flow problem is NP-hard and therefore we provide in this section approximate solutions. Our starting point for obtaining an approximate solution is a linear programming formulation of the problem (Section III-A). Our formulation is somewhat unusual, in the sense that a feasible solution to the problem does not necessarily define an integral solution to the linear formulation. This happens since the path-degree constraints are essentially non-linear; a path carrying *positive* flow contributes to the path-degree constraints of all the nodes belonging to it, independently of the flow value. We show that the integrality gap of our linear program is at least $\Omega(\sqrt{n})$, where n denotes the number of nodes in the network. Thus, circumventing the integrality gap and obtaining better approximation factors requires a further relaxation. In our solution, constraints are violated, however, the extent to which they are violated is bounded.

Our approximate solution is derived from a randomized rounding of a fractional solution to the linear program. Specifically, we present a bicriteria approximation algorithm (Section III-B) which is guaranteed to achieve an expected total flow equal to at least $OPT/O(\log n)$, where OPT is an upper

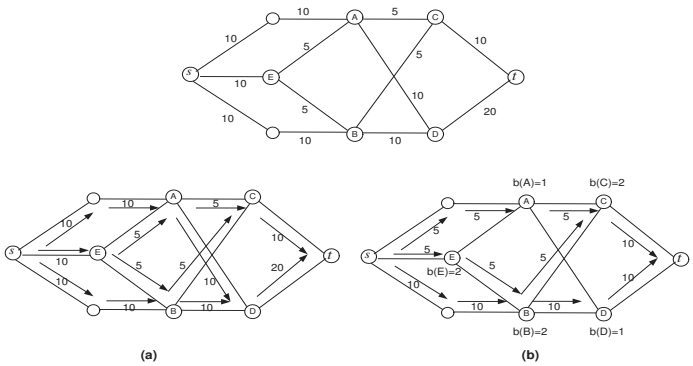

Fig. 1. (a) max flow between s and t without path-degree constraints; (b) max flow in the presence of path-degree bounds on the nodes.

bound on the maximum flow value obtained by any solution, and is derived from the linear program. Our flow solution satisfies the link capacity constraints. However, the path-degree constraints might be violated by our solution. The violation of each path degree constraint is by at most a factor of $O(\log n)$ (with very high probability).

We further consider the case where *fairness* is required (Section III-C). This means that flow over each path p is not allowed to consume more than a $1/\log n$ fraction of the capacity of any of the links belonging to p. We can prove that in this case the bicriteria approximation algorithm achieves an expected total flow equal to $OPT/O(1)$.

In the general case, where flow can be routed on any path in the network, our linear program can have an exponential number of variables, corresponding to the number of distinct paths in the network. An efficient solution of the linear program in this case requires an efficient separation oracle for the dual program. We develop such a separation oracle and show how to implement it efficiently (Section III-D).

A. General model

We are given a directed graph (or network) $G = (V, E)$ with link capacities $c(e)$ for each link $e \in E$. There are pairs of sources and destinations (s_i, t_i) wishing to transmit traffic between them. Each pair (s_i, t_i) is associated with a *flow demand* d_i. Flow is routed between the sources and the destinations via *flow paths*. The number of flow paths that can be routed through a particular node is bounded and called the *path degree* of the node. For node v, the path degree is denoted by $b(v)$. The *bounded path-degree max flow problem* asks for the total maximum feasible flow between all pairs (s_i, t_i) satisfying: (i) for each i, total flow between s_i and t_i does not exceed d_i; (ii) for each link e, the total flow through e does not exceed $c(e)$; and (iii) for each node v, the number of (positive) flow paths going through it does not exceed $b(v)$.

A simple example is depicted in Figure 1, where the link capacities are given in the upper graph. There is a single pair (s, t) with a large flow demand. Without path-degree constraints, the maximum flow is equal to 30 and is given in 1(a). In 1(b), path-degree bounds are added and the maximum feasible flow drops down to 20.

We formulate the bounded path-degree max flow problem as a linear program. In this linear program, we need to represent both capacity and path degree constraints. To this end, for a path p connecting a pair (s_i, t_i), define, $c(p)$, the *capacity* of p, as the minimum between the bottleneck link capacity of p and the demand d_i, i.e., $c(p_{(s_i,t_i)}) = \min(\min_{e\in p}\{c(e)\}, d_i)$. Let $x(p)$, $0 \le x(p) \le 1$, denote the fraction of path p used by the linear program. Thus, the amount of flow routed through p is defined to be $f(p) = c(p) \cdot x(p)$. We now discuss the path-degree constraints. Each flow path p, for which $f(p) > 0$, should contribute a unit towards the path-degree constraints of the nodes belonging to it. This means that the contribution of $f(p)$ to the path-degree constraints is a "step function" which is difficult to capture by linear constraints. In our formulation, the contribution of path p to the path-degree constraints of the nodes belonging to p is defined to be $x(p)$. Thus, even if we are given a feasible flow solution to the bounded path-degree max flow problem, the values of the variables $x(p)$ may still be required to be fractional. Hence, our fractional formulation is not a relaxation of an integral solution to the problem, even though the value of our formulation is still an upper bound on any feasible flow solution. In this sense our formulation uses the integral/fractional linear programming framework in a somewhat uncommon way.

Our formulation of the bounded path-degree max flow problem is as follows.

$$\text{Max} \quad \sum_p x(p) \cdot c(p) \qquad s.t. \qquad \text{(Flow-LP)}$$

$$\text{for each edge } e: \quad \sum_{\{p|e\in p\}} x(p) \cdot c(p) \le c(e), \quad (1)$$

$$\text{for each node } v: \quad \sum_{\{p|v\in p\}} x(p) \le b(v), \quad (2)$$

$$\text{for each pair } (s_i, t_i): \quad \sum_{\{p|p\in(s_i,t_i)\}} x(p) \cdot c(p) \le d_i. \quad (3)$$

The first constraint (1) is the link capacity constraint, stating that the total flow passing through an edge e is at most its capacity $c(e)$. The second constraint (2) is the node path-degree constraint, stating that the sum of the path fractions passing through a node v is at most its path-degree bound $b(v)$. The third constraint (3) states that the sum of flows over all paths connecting a pair (s_i, t_i) is at most the demand d_i. Note that there is no need to add a constraint bounding $x(p) \le 1$, as it is implied by constraint (1).

The number of variables $x(p)$ in (Flow-LP) depends on the number of paths connecting the sources and destinations, and thus can be of exponential size in general. However, the number of constraints is polynomial. We show in Section III-D that (Flow-LP) can be efficiently solved using a combinatorial framework. Furthermore, the number of paths carrying positive flow in a fractional optimal solution can be shown to be polynomial. However, the running time of this algorithm includes several high value constants that might be problematic in large scale settings. Moreover, in practice, it is often the case that potential paths for routing the traffic are limited to a specific set and provided ahead of operation time. We thus consider a more practical variation of our problem, where we are given a

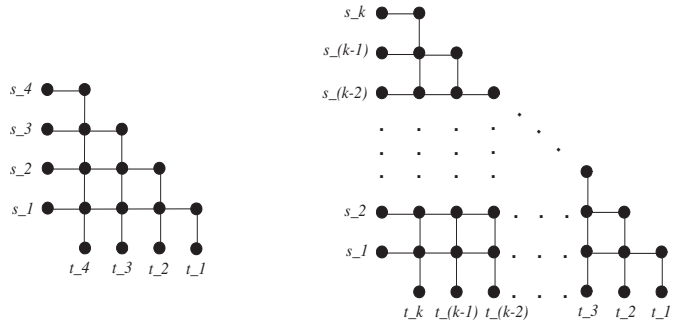

Fig. 2. Integrality gap of the bounded path-degree max flow problem.

set of pre-defined paths $\mathcal{P}$ (typically of polynomial-size) over which the flow between the pairs (s_i, t_i) can be routed. In this case, the linear program remains the same as (Flow-LP), but has only a small number of variables $x(p)$ (for all $p \in \mathcal{P}$), and thus can be solved more efficiently.

We note that our model is also valid for the case where each flow demand d_i between (s_i, t_i) has weight w_i. These weights can induce, for example, a prioritization of the demands. In this case, the objective function is maximization of the total weighted flow $\sum_i \sum_{p \in (s_i,t_i)} x(p) \cdot c(p) \cdot w_i$. All the results presented in this section can be directly applied to the weighted case as well.

Integrality Gap. We present an integrality gap of $\Omega(\sqrt{n})$ for linear program (Flow-LP). The integrality gap is demonstrated by the instance in Figure 2, comprising of a grid with k pairs of sources and destinations (s_i, t_i), $1 \leq i \leq k$, all with demands equal to 1. The left side of the figure presents the grid for $k = 4$, and the right side presents the grid for general k. The capacities of all links are 1, and the path-degree bounds of all nodes are 1. Any two paths p_i and p_j, connecting pairs (s_i, t_i) and (s_j, t_j) respectively, share at least one common node. In addition, any path p_i connecting (s_i, t_i), intersects all the paths connecting all other pairs. Thus, the integral solution can connect only a single pair and route a flow of 1 over its path, achieving a revenue of $OPT_{int} = 1$. In the fractional solution, we can choose k paths connecting the pairs (s_i, t_i) with the property that at most two paths go through each internal node. Thus, by routing a flow of $1/2$ over each path ($x(p) = 1/2$ for each path p), we adhere to the path-degrees of the nodes, and achieve a revenue of $OPT_{frac} = k/2$. As the number of nodes is $n = O(k^2)$, we get an integrality gap of $\Omega(\sqrt{n})$.

B. An $(O(\log n), O(\log n))$ bicriteria approximation

We present a bicriteria approximation algorithm for the bounded path-degree max flow problem. Our algorithm first solves (Flow-LP), the fractional LP relaxation of the problem. Given an optimal fractional solution, the algorithm applies randomized rounding to the solution simply by choosing each path $p \in \mathcal{P}$ (independently) with probability equal to $x(p)$, and then routing over p a flow value of $c(p)/O(\log n)$. The algorithm achieves an expected total flow equal to $OPT/O(\log n)$, where OPT is the maximum flow obtained by the fractional solution (for which flow is also restricted to $\mathcal{P}$), and n denotes the number of nodes in G. Our algorithm satisfies the link capacity constraints, but may violate the path-degree bounds of the nodes by at most a factor of $O(\log n)$.

It follows from the integrality gap example above that any algorithm satisfying the path-degree bounds can only achieve an $\Omega(\sqrt{n})$-approximation factor. Thus, relaxing the path-degree bounds is motivated by this example, so as to improve the approximation factor.

Algorithm 1 Randomized $(O(\log n), O(\log n))$ bicriteria approximation

1: Solve (Flow-LP) for the bounded path-degree max flow problem.
2: Independently, for each path $p \in \mathcal{P}$, choose it with probability $x(p)$.
3: For each chosen path p, route over p a flow value of $c(p)/(6 \log n)$.

In the sequel we analyze the algorithm. We compute the expected flow value and bound the probability that link capacities and path-degrees are violated, resulting from the randomized rounding. The following version of the Chernoff bound will be very useful in our analysis. Given are independent random variables $x_1, \ldots, x_n$, where for all i, $x_i \in [0, 1]$. Let $\mu = \mathbb{E}[\sum_{i=1}^{n} x_i]$. Then,

$$\mathbf{Pr}\left[\sum_{i=1}^{n} x_i \geq (1+\epsilon)\mu\right] \leq e^{\frac{-\epsilon^2 \mu}{2+\epsilon}}. \tag{4}$$

Link capacity constraints. We first bound the extent to which link capacities are violated. Given a link e, we define a set of random independent variables $y_{p,e}$ corresponding to the paths going through e:

$$y_{p,e} = \begin{cases} c(p) & \text{with probability } x(p) \\ 0 & \text{otherwise} \end{cases}$$

Random variables $y_{p,e}$ are independent. Note that their sum $f(e) = \sum_{\{p|e \in p\}} y_{p,e}$ is exactly the flow over link e following our randomized rounding procedure. The expected flow over e is:

$$\hat{f}(e) = \mathbb{E}\left[\sum_{\{p|e\in p\}} y_{p,e}\right] = \sum_{\{p|e\in p\}} x(p) \cdot c(p) \leq c(e), \tag{5}$$

where the last inequality follows from the link capacity constraint (1) of (Flow-LP).

Lemma 1: The probability of violating the capacity of link e by more than a factor of $(1 + 6 \log n)$ is at most $1/n^4$, i.e.,

$$\mathbf{Pr}\left[\sum_{\{p|e\in p\}} y_{p,e} \geq (1 + 6\log n) n \cdot c(e)\right] < \frac{1}{n^4},$$

Proof: We normalize the random variables $y_{p,e}$ to the range $[0, 1]$ by considering $\frac{y_{p,e}}{c(e)}$. By inequality (5), $\mathbb{E}[\sum_{\{p|e\in p\}} \frac{y_{p,e}}{c(e)}] \leq 1$. Applying the Chernoff bound (4) to the variables $\frac{y_{p,e}}{c(e)}$ and choosing $\epsilon = 6 \log n$ we get:

$$\mathbf{Pr}\left[\sum_{\{p|e\in p\}} \frac{y_{p,e}}{c(e)} \geq (1 + 6\log n)\right] \leq e^{\frac{-36 \log^2 n}{2+6\log n}} < e^{-4 \log n} = \frac{1}{n^4}.$$

■

The number of links in G is at most n^2. By the union bound, $\sum_e (\mathbf{Pr}[f(e) \geq (1 + 6 \log n)c(e)]) < \frac{1}{n^2}$, meaning that the probability that the capacity of any link (among all links in G) is violated by a factor higher than $O(\log n)$ is less than $1/n^2$, i.e., negligible.

Demand constraints. We turn to consider the flow demands and bound the probability that the flow between (s_i, t_i) exceeds the demand d_i. We define random variables $y_{p,i}$ corresponding to the paths P connecting (s_i, t_i):

$$y_{p,i} = \begin{cases} c(p) & \text{with probability } x(p) \\ 0 & \text{otherwise} \end{cases}$$

Random variables $y_{p,i}$ are independent. Note that their sum $f_i = \sum_{\{p|p\in(s_i,t_i)\}} y_{p,i}$ is exactly the amount of flow between s_i and t_i following our randomized rounding procedure. The expected flow over all paths connecting (s_i, t_i) is:

$$\hat{f}_i = \mathbb{E}\left[\sum_{\{p|p\in(s_i,t_i)\}} y_{p,i}\right] = \sum_{\{p|p\in(s_i,t_i)\}} x(p)\cdot c(p) \le d_i, \quad (6)$$

where the inequality follows from the constraint (3) of (Flow-LP).

Lemma 2: The probability of violating demand i by more than a factor of $(1 + 6\log n)$ is at most $1/n^4$, i.e.,

$$\mathbf{Pr}\left[\sum_{\{p|p\in(s_i,t_i)\}} y_{p,i} \ge (1+6\log n)\log n \cdot d_i\right] < \frac{1}{n^4}.$$

Proof: We normalize the random variables $y_{p,i}$ to the range $[0, 1]$ by considering $\frac{y_{p,i}}{d_i}$. By inequality (6), $\mathbb{E}[\sum_{\{p|p\in(s_i,t_i)\}} \frac{y_{p,i}}{d_i}] \le 1$. Applying the Chernoff bound (4) to the variables $\frac{y_{p,i}}{d_i}$ and choosing $\epsilon = 6\log n$, we get:

$$\mathbf{Pr}\left[\sum_{\{p|p\in(s_i,t_i)\}} y_{p,i} \ge (1+6\log n) d_i\right] < \frac{1}{n^4}.$$

■

The number of demands in G is at most n^2. By the union bound, $\sum_i (\mathbf{Pr}[f_i \ge (1+6\log n)d_i]) < \frac{1}{n^2}$, meaning that the probability that the flow of any pair (s_i, t_i) exceeds the demand value d_i by more than a factor of $O(\log n)$ is less than $1/n^2$, thus negligible.

Path-degree constraints. We now turn to consider path-degree constraints and bound the probability that the flow going through node v exceeds the path-degree constraint $b(v)$. We can assume that for each node v, $b(v) \ge 1$, that is, at least one path can pass through each node. Given a node v, we define a set of random independent variables $y_{p,v}$:

$$y_{p,v} = \begin{cases} 1 & \text{with probability } x(p) \\ 0 & \text{otherwise} \end{cases}$$

Random variables $y_{p,v}$ are independent. Note that their sum $\beta(v) = \sum_{\{p|v\in p\}} y(p, v)$ is exactly the number of flow paths going through v following our randomized rounding procedure. The expectation of $\beta(v)$ is:

$$\hat{\beta}(v) = \sum_{\{p|v\in p\}} y(p,v) = \sum_{\{p|v\in p\}} x(p) \le b(v), \quad (7)$$

where the last inequality follows from the path-degree constraint (2) of (Flow-LP).

Lemma 3: The probability of violating the path-degree constraint of node v by more than a factor of $(1 + 5\log n)$ is at most $1/n^3$, i.e.,

$$\mathbf{Pr}\left[\sum_{\{p|v\in p\}} y_{p,v} \ge (1+5\log n)\log n \cdot b(v)\right] < \frac{1}{n^3}.$$

Proof: We normalize the random variables $y_{p,v}$ to the range $[0, 1]$ by considering $\frac{y_{p,v}}{b(v)}$. By inequality (7), $\mathbb{E}[\sum_{\{p|v\in p\}} \frac{y_{p,v}}{b(v)}] \le 1$. Applying the Chernoff bound (4) to the variables $\{\frac{z_{p,v}}{b(v)}\}_{\{p|v\in p\}}$ and choosing $\epsilon = 5\log n$ we get:

$$\mathbf{Pr}\left[\sum_{\{p|v\in p\}} \frac{y_{p,v}}{b(v)} \ge (1+5\log n)\right] \le e^{\frac{-25\log^2 n}{2+5\log n}} < e^{-3\log n} = \frac{1}{n^3}.$$

■

The number of nodes in G is n. By the union bound, $\sum_v (\mathbf{Pr}[\beta(v) \ge (1+5\log n)b(v)]) < \frac{1}{n^2}$, meaning that the probability that the number of flow paths going through a node exceeds the path-degree constraint by more than a factor of $O(\log n)$ is less than $1/n^2$, thus negligible.

Approximation factor. It follows from our analysis that by scaling down the flow on each chosen path by a factor of $O(\log n)$, link capacities are only violated with negligible probability. The path-degree constraints will not be violated by more than a factor of $O(\log n)$ with very high probability.

Contribution of flow to the objective function should only come from flows satisfying the demand constraints. Since we scaled down the flow on each chosen path by a factor of $O(\log n)$, demand constraints are only violated with negligible probability (by our analysis), and thus all flow paths that are chosen by the algorithm indeed contribute their flow. Consequently, we achieve a flow solution of total value $OPT/O(\log n)$, satisfying link capacities, demand constraints, and violating path-degree constraints by at most a factor of $O(\log n)$.

C. An $(O(1), O(\log n))$ bicriteria approximation

We present an $(O(1), O(\log n))$ bicriteria approximation under the assumption that the flow over each path p is not allowed to consume more than a $1/\log n$ fraction of the path capacity. This restriction can easily be incorporated into linear program (Flow-LP) by requiring that for each path p, $x(p) \le \frac{1}{\log n}$. For this case, our randomized algorithm achieves an expected total flow of $OPT/O(1)$, while satisfying the link capacity constraints, but potentially violating the path-degree bounds of the nodes by at most a factor of $O(\log n)$, as before.

Algorithm 2 Randomized $(O(1), O(\log n))$ bicriteria approximation

1: Solve (Flow-LP) for the bounded path-degree max flow problem with the additional constraint $x(p) \le \frac{1}{\log n}$ for each path p.
2: Independently, for each path $p \in \mathcal{P}$, choose it with probability $x(p)\log n$.
3: For each chosen path p, route over p flow of value $c(p)/(7\log n)$.

Our algorithm applies randomized rounding to the fractional optimal solution by choosing each path $p \in \mathcal{P}$ with probability equal to $x(p)\log n$. Note that $x(p)\log n \le 1$. We then route over p flow of value $\frac{c(p)}{7\log n}$. factor.

The analysis of the algorithm goes along similar lines as in Section III-B. The proofs are thus presented concisely, focusing on the particularities of this case.

Link capacity constraints. We define the set of random independent variables $y_{p,e}$ as follows:

$$y_{p,e} = \begin{cases} c(p)\frac{1}{\log n} & \text{with probability } x(p)\log n \\ 0 & \text{otherwise} \end{cases}$$

Lemma 4: The probability of violating the capacity of link e by more than a constant factor of 7 is at most $1/n^4$, i.e.,

$$\mathbf{Pr}\left[\sum_{\{p|e\in p\}} y_{p,e} \ge 7c(e)\right] < \frac{1}{n^4}.$$

Proof: We normalize the random variables $y_{p,e}$ to the range $[0, 1/\log n]$ by considering $\frac{y_{p,e}}{c(e)}$. From the link capacity constraint (1), the expected value

$$\mathbb{E}\left[\sum_{\{p|e\in p\}} \frac{y_{p,e}}{c(e)}\right] = \sum_{\{p|e\in p\}} \frac{c(p)}{c(e)} \cdot \frac{1}{\log n} \cdot x(p) \cdot \log n \leq 1.$$

Applying the Chernoff bound (4) to the variables $\frac{y_{p,e}}{c(e)} \cdot \log n$ (where now $\mu \leq \log n$) and choosing $\epsilon = 6$, yields the desired result. ■

Note that the sum of the random independent variables $\sum_{\{p|e\in p\}} y_{p,e}$ is exactly the value of the flow over link e following our randomized rounding procedure. By the union bound, the probability that the capacity of a link is violated by more than a factor of 7 is less than $1/n^2$, thus negligible.

Demand constraints. The analysis of the demand constraints is similar to the link capacity constraints. Defining a set of random independent variables $y_{p,i}$ corresponding to the paths connecting (s_i, t_i), it can be shown that the probability that the flow of any pair (s_i, t_i) exceeds the demand value d_i by more than a factor of 7 is less than $1/n^2$, thus negligible.

Path-degree constraints. The path-degree of a node is not changed by scaling down the value of positive flows over the different paths, as any positive value has the same contribution to the degree. Intuitively, this explains why the additional constraint on the value of $x(p)$ does not improve the violation of the path-degree constraints.

We define random independent variables $z_{p,v}$ for each node v:

$$z_{p,v} = \begin{cases} 1 & \text{with probability } x(p)\log n \\ 0 & \text{otherwise} \end{cases}$$

Applying the Chernoff bound (4) to the variables $\{\frac{z_{p,v}}{b(v)}\}_{\{p|v\in p\}}$ (similarly to Section III-B), yields that the probability that the number of flow paths going through a node exceeds the path-degree constraint by more than a factor of $O(\log n)$ is less than $1/n^2$, thus negligible.

Approximation factor. It follows from our analysis that by scaling down the flow on each chosen path p by a factor of $7\log n$, link capacities and demand constraints are only violated with negligible probability. Consequently, we achieve a flow solution of total value $OPT/7$, satisfying link capacities, demand constraints, and violating path-degree constraints by at most a factor of $O(\log n)$.

D. Solvability of the general problem

In the general bounded path-degree max flow problem, there is no (predefined) restricted set of allowed flow paths $\mathcal{P}$, and in general traffic can use any path in the network. Thus, the LP relaxation of this problem (see (Flow-LP)) may have an exponential number of variables $x(p)$, corresponding to all possible paths p connecting sources and destinations. Nevertheless, we show that by applying the Ellipsoid algorithm to the dual linear program, (Flow-LP) can be solved efficiently (i.e., in polynomial time). Moreover, the randomized rounding algorithms and analysis presented in Sections III-B and III-C still apply to the general case of our problem.

In the dual linear program, there is a dual variable y_e for each capacity constraint on edge e (primal constraint (1)), a dual variable z_v for each path-degree constraint on node v (primal constraint (2)), and a dual variable α_i for each demand constraint i (primal constraint (3)). The dual problem is as follows.

$$\text{Min} \sum_{e\in E} c(e)\cdot y_e + \sum_{v\in V} b(v)\cdot z_v + \sum_i d_i \cdot \alpha_i \qquad \text{s.t.} \qquad (8)$$

$$\forall p \in (s_i,t_i): \sum_{\{e\in p\}} c(p)\cdot y_e + \sum_{\{v\in p\}} z_v + c(p)\cdot\alpha_i \geq c(p) \quad (9)$$

$$\forall\ e,\ v, \text{ and } i\text{: } y_e,\ z_v,\ \alpha_i \geq 0. \quad (10)$$

As already mentioned, in the dual problem the number of variables is polynomial while the number of constraints might be exponential (all possible paths). Such a pair of primal-dual linear programs can be solved by the Ellipsoid algorithm if a separation oracle can be found for the dual LP. A separation oracle is a polynomial-time algorithm, that given an assignment to the dual variables, either answers that the assignment is feasible for the dual linear program, or finds a violated constraint. Moreover, if the separation oracle can find the most violated constraint in polynomial time, the LPs can be solved efficiently using a combinatorial algorithm (see e.g., [8].

The separation oracle. Given an assignment to the dual variables, the separation oracle checks whether there is a violated constraint. That is, for all i, does there exist a path $p \in (s_i, t_i)$ such that:

$$\sum_{\{e\in p\}} y_e + \frac{1}{c(p)} \sum_{\{v\in p\}} z_v < 1 - \alpha_i. \qquad (11)$$

The separation oracle can be implemented by computing the path p minimizing the value of the expression in the LHS of (11). Path p either defines a violated constraint or provides a certificate that the dual solution is indeed feasible. (We note that if p defines a violated constraint, it also defines the most violated constraint.) The separation oracle thus reduces to the problem of computing:

$$\min_{p\in(s_i,t_i)} \left\{ \sum_{\{e\in p\}} y_e + \frac{1}{c(p)} \sum_{\{v\in p\}} z_v \right\}. \qquad (12)$$

We note that the coefficient $\frac{1}{c(p)}$ (which is specific to each path p) is the source of the difficulty of solving(12). Otherwise, (12) is easily reducible to the problem of finding a shortest path in a graph (via Dijkstra's algorithm), with both edge and node lengths (y_e, z_v).

We solve the problem of (12) efficiently as follows. Let us restrict our attention to the paths connecting s_i and t_i. (We do this for each i separately.) First, for each link e, $c(e)$ is assumed to be at most d_i, otherwise $c(e)$ can be truncated to d_i, since $c(p) \leq d_i$ for all $p \in (s_i, t_i)$. Next, we define a set of graphs $\{G_w = (V, E_w)\}$ with respect to parameter w, where $E_w = \{e \in E | c(e) \geq w\}$. The graphs in the set $\{G_w\}$ are defined for each value $w = c(e)$ of edge capacities in E, and thus $|\{G_w\}| \leq |E|$.

Claim 3.1: Each path $p \in (s_i, t_i)$ is a path in at least one graph in $\{G_w\}$. Conversely, each path $p \in G_w$ (for all w) is a path in G.

The proof of the claim is immediate, and follows from the fact that each edge in G_w is also an edge in G, and a path p in G is a path in all graphs G_w for which $w \leq c(p)$.

We define $\ell(p) = \sum_{\{e\in p\}} y_e + \frac{1}{c(p)} \sum_{\{v\in p\}} z_v$ as the *length* of path p, with distance values y_e on the edges and $\frac{z_v}{c(p)}$ on the nodes. We define $\ell_w(p) = \sum_{\{e\in p\}} y_e + \frac{1}{w} \sum_{\{v\in p\}} z_v$ as the *length* of path p, with distance values y_e on the edges and $\frac{z_v}{w}$ on the nodes. As already mentioned, we note that minimizing $\ell_w(p)$ in G_w can be solved efficiently by a variant of Dijkstra's algorithm.

The algorithm computing $\min_p\{\ell(p)\}$ (thus solving the minimization problem stated in (12)) runs as follows. For each graph G_w, we compute the shortest path with respect to $\ell_w(p)$. The algorithm picks the path with minimum length $\ell_w(p)$ among all the shortest paths computed for the graphs in $\{G_w\}$. Denote by p' the path with minimum such length, being the shortest path in $G_{w'}$. The output of the algorithm is the (true) length of p', i.e., $\ell(p') = \sum_{\{e\in p'\}} y_e + \frac{1}{c(p')} \sum_{\{v\in p'\}} z_v$.

Let p^* be the path minimizing (12). We show that $\ell(p') \leq \ell(p^*)$, thus proving the correctness of the algorithm. Let $w^* = c(p^*)$. Thus, $p^* \in G_{w^*}$ and $\ell(p^*) = \ell_{w^*}(p^*)$. Since the algorithm returned path p', $\ell_{w'}(p') \leq \ell_{w^*}(p^*)$. However,

$$\begin{aligned} \ell(p') &= \sum_{\{e \in p'\}} y_e + \frac{1}{c(p')} \sum_{\{v \in p'\}} z_v \\ &\leq \sum_{\{e \in p'\}} y_e + \frac{1}{w'} \sum_{\{v \in p^*\}} z_v = \ell_{w'}(p'), \end{aligned} \tag{13}$$

since $p' \in G_{w'}$ and $c(p') \geq w'$. Hence, $\ell(p') \leq \ell(p^*)$, completing the proof of correctness.

IV. Experimental Study

In this section, we present extensive simulations, demonstrating the practical use of our algorithm for the bounded path-degree max flow problem, both in data centers and backbone scenarios. Our experiments were performed over the practical setting comprising of two main phases. In the first phase, we compute a set comprising of 10 different paths for each pair (s_i, t_i), over which the traffic can be routed (traffic can be split between the paths).

In the second phase, we compute the maximum flow for this setting, and provide as output the flow value routed over each path. Given the pre-defined set of potential paths $\mathcal{P}$, the second phase is performed by solving the fractional path-degree max flow LP over the set of paths $\mathcal{P}$, and rounding the solution, as described in Section III-B. The complete algorithm used in our experiments is described by Algorithm 3.

Algorithm 3 complete algorithm for the bounded path-degree max flow problem

1: For each pair (s_i, t_i), compute a set $\mathcal{P}_i$ of k different paths over which the traffic can be routed. ▷ "k-shortest" or "almost disjoint" paths
2: Apply algorithm 1 (or 2) using the paths in the sets $\mathcal{P}_i$.
3: For each node v with path-degree violation:
- Sort the paths with positive flow passing through v in decreasing order according to their amount of flow.
- Remove paths according to this order until path-degree $b(v)$ is satisfied.

A. Simulated instances

We performed our experiments over three types of graphs, representing different practical cases of networking infrastructures.

- The Barabasi-Albert (BA) model [2], used for generating random scale-free networks with power-law degree distributions. Scale free networks are observed in many systems, including the Internet backbone network [7]. The BA graphs were created with 1000 nodes. During the creation of the graph, each new added node has an initial degree of 2, and is connected to an existing node v_i with probability $\frac{\delta_i}{\sum_j \delta_j}$, where δ_i is the degree of node v_i and $\sum_j \delta_j$ is the sum of degrees over all pre-existing nodes. The nodes in these graphs represent forwarding devices, to which dozens of physical hosts are attached using point-to-point connectivity. Assuming different demands can be associated with different virtual machines, or different applications running on a physical host, we obtain an infrastructure where we can easily accommodate a large set of demands.
- The BCube network architecture [9], specifically designed for modular data centers supporting bandwidth-intensive applications. At the core of the BCube architecture is its server-centric network structure, where servers act not only as end hosts but also as relay nodes for each other. BCube is a recursively defined structure, where a BCube_0 comprises of n servers connected to an n-port switch, and a BCube_k ($k \geq 1$) is constructed from n BCube_{k-1} and n^k n-port switches. We generate a 2-level structure, with $n = 30$. We chose to simulate the BCube topology as it is much more complex than other common Data Center topologies (e.g., Fat-Tree), allowing the flow to be routed on diverse paths.
- A mesh topology, with $n = 40 \cdot 12$ nodes, each with degree 4. Such a topology can either model a general core network (Internet backbone), or a cold storage data center topology, used to store data that is written once and accessed infrequently. We model an infrastructure of 12 racks, each rack comprising of 40 UBoxes, each may contain dozens of virtual or physical servers. In this type of data center, top-of-rack (ToR) switches are omitted due to power and budget constraints, and the connectivity between servers is obtained by the use of mini switches (typically with four or six 10 Gigabit ports) located of each UBox. In this case the mini-switches are connected one to one another using a mesh-like topology. This topology is highly relevant in our case since such mini-switches are usually limited devices with low resources in terms of forwarding flow capacities.

For all topologies, we simulated traffic of 200,000 different demands. Commonly, forwarding tables size may range from thousands to dozen of thousands for on-die devices (with resources on-chip), while they may grow to hundreds of thousands for external devices. In each graph instance used for our experiments, this size was set uniformly between the range of 500 and 10,000. This value is set with respect to the number of flow demands (200,000). We note that these values can characterize much larger instances, where the number of flows and the forwarding table sizes increase accordingly (see Figure 7 and respective discussion on *normalized forwarding table size*, end of Section IV-B).

In all topologies, link capacities are uniform (typically representing 10 Gigabit links), while demands are generated randomly using uniform distributions, and are normalized with respect to the maximum flow that can be supported by our infrastructure. Specifically, we denote by $F_{\max}$ the maximum flow that can be routed for all demands in case there are no path degree constraints on the nodes. Then, each randomly generated demand d_i is normalized to $\frac{d_i}{\sum_i d_i} \cdot F_{\max}$.

We considered two heuristics for selecting the set of paths during the first phase of our algorithm. As the first heuristic, we used a generalization of Dijkstra's algorithm for computing the k shortest paths between each pair (s_i, t_i) (in our instances, all links are weighted equally). The second heuristic was used for selecting "almost disjoint" paths, that is, paths with a large number of distinct links. For each pair (s_i, t_i), the "almost disjoint paths" heuristic first picks the shortest path. Then, it significantly increases the weights of its links, picking the shortest path of the new instance, and continuing similarly, each time increasing the weights of the links of the chosen path. The simulations are presented for the best set of paths computed during the first phase, which is achieved by the "almost disjoint paths" heuristic. This heuristic achieves a better performance, as it tries to distribute the usage of forwarding tables resources between distinct paths. Conversely, considering our instances, the k shortest paths heuristic picks paths having a significant overlap, thus condensing flows and creating a set of loaded nodes.

B. Simulation results

Figure 3 presents the performance of our algorithm with respect to the maximum flow that can be achieved (considered as 100%), that is, the flow achieved in case there are no bounds on the forwarding tables size. Results are presented for all three types of topology, namely, BA, BCube and mesh topologies. The performance (percentage of maximal flow achieved) is computed for different

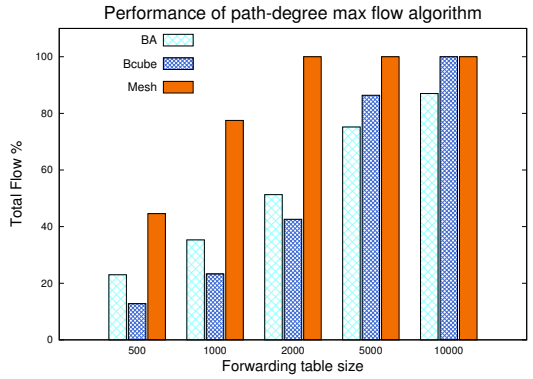

Fig. 3. Percentage of the total flow achieved by Alg. (3) with respect to the maximum flow that can be routed in the network.

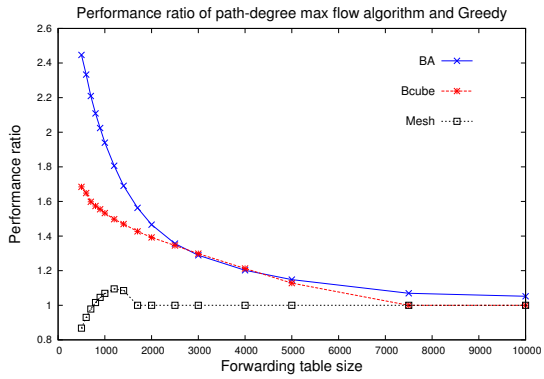

Fig. 4. Performance ratio of Alg. (3) and the greedy algorithm.

values of forwarding tables size (path-degree of the nodes), ranging from 500 to 10,000 entries. Clearly, the smaller the flow table size of the nodes, the lower the total flow that can be achieved. For forwarding table sizes of 5000 entries, the flow achieved is above 75% of the optimum (BA topology), while for sizes of 1000 entries it decreases to be above 25% of the optimum (BCube topology). We note that when computing the maximum flow, where no bounds are provided on the forwarding tables size, the actual usage of forwarding table entries was measured to be up to 30,000 (the highest values were measured for the BA topology). As can be seen from the graph, using our technology, one needs to use only 10,000 entries (a factor of 3) while reducing the total flow by 15%.

However, it could be the case that only few devices actually need big tables, and the extra flows carry only a small fraction of the total flow. To check it, we tested our algorithm against a greedy algorithm (Greedy) that calculates a maximum (fractional) flow without considering forwarding tables size constraints, and then removes small flows passing through overloaded nodes, so as to adhere to the path-degree constraints. The ratio between the total flow achieved by our algorithm and Greedy is depicted in Figure 4, for all three types of topologies.

It can be seen that the performance of our algorithm exceeds the performance of the greedy algorithm in almost all cases (except a single case for the mesh topology). The lower the flow table size of the nodes, the better the performance of our algorithm compared to greedy. This is a direct outcome of the fact that our algorithm has a global view of the network, and balances the utilization of the forwarding tables resources among all nodes, which is more significant as the resources become more limited (i.e. small forwarding table sizes). The larger difference is observed for the BA topology, where our algorithm outperforms the greedy algorithm by up to a factor of 2.5. For the BCube topology, a difference of up to a factor of 1.7 can be observed between the performance of the algorithms. Note that all values were measured for the case of 200,000 flow demands, and are thus expected to be even more significant in settings of larger scale, with a higher number of flow demands.

While the theoretical approximation factor of our randomized rounding algorithm is only guaranteed to be at most $O(\log n)$, in all the simulations performed, we obtained a difference of less than 1% between the flow achieved by the fractional optimal solution and our randomized rounded solution. Consequently, almost no violation of the constraints occurred during the randomized rounding procedure of the algorithm, and the solution obtained is almost optimal.

Figure 5 presents the distribution of the flows between the different paths provided by the first phase of the algorithm. A number of 10 different paths were computed for each flow demand as potential paths for routing the traffic. It can be seen that in all topologies, up to 45% of the demands use the first path (shortest path), and the rest are distributed between the other paths. This result focuses on the importance of the first phase of our algorithm, and emphasizes the need of providing a set of diverse paths for each flow demand.

An interesting practical question is the distribution of the forwarding table resources between the nodes. Clearly, such a distribution highly depends (among other things) on the network topology. Figure 6 presents the utilization and average load of the forwarding tables of the nodes within each of the three topologies. The maximal table size in this simulation was set to 1000 entries, however, we note that a similar usage was observed for larger values as well. It can be seen that in the mesh topology, which is a flat network, all nodes are equally loaded and reach a very high percentage of their size limit. In the BA topology, the forwarding tables load nearly follows the power-law nature of the graph, where only 10% of the nodes utilize more than 80% of their forwarding table size. In the BCube topology, the distribution follows the hierarchical structure of the graph, where most of the nodes utilize less than 15% of the resources while 7% of the nodes utilize more than 90% of their forwarding table size.

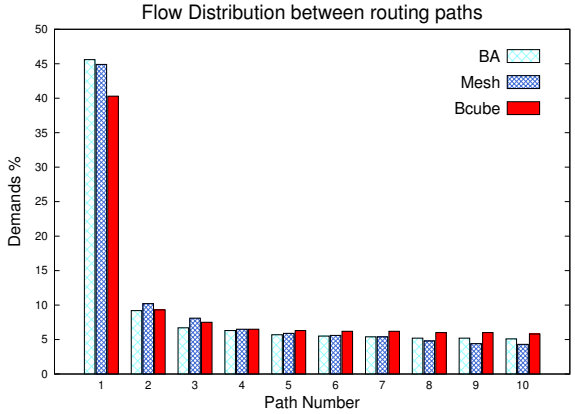

Fig. 5. Flow distribution of the demands between the different paths provided as input by the first phase of Alg. (3).

In order to further understand the capabilities of our algorithm in different settings, Figure 7 presents its performance with respect to a *normalized forwarding table size* (NS). This normalization is done with respect to $\rho = \frac{m \cdot \delta}{n}$ where m is the number of flow demands, δ is the average routing path length, and n is the number of devices in the network. This number represents the forwarding table size needed for the "ideal" case where all flows are distrusted evenly among all network devices. For example, in order to satisfy 200,000 demands in a network with a total of 1,000 nodes, where the average routing path comprises of six nodes, the forwarding table size of each node should be at least $\frac{200000 \cdot 6}{1000}$ (if the forwarding table sizes are not equal, then the total number of entries should be considered instead).

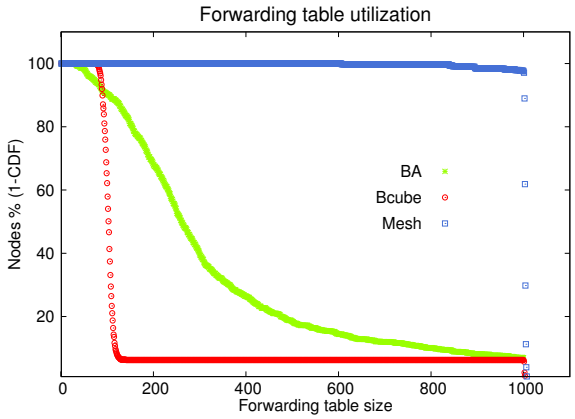

Fig. 6. The x-axis represents the number of forwarding table entries used. The y-axis represents the complement of the cumulative distribution function, i.e., percentage of nodes with usage higher than the respective x-axis value.

So when we have b entries in each device of the network, the normalized forwarding table size is $\frac{b}{\rho} = \frac{b \cdot n}{m \cdot \delta}$. In Figure 7, the performance of our algorithm is evaluated with respect to different NS values, providing a good indication of its performance on different settings and scales. It can be observed that in hierarchical network topologies (such as BA and BCube), where most of the routing paths traverse a small set of nodes, a larger NS value is required in order to satisfy the set of demands, as opposed to uniform graphs (e.g., mesh) where routing path are uniformly distributed between the nodes [4].

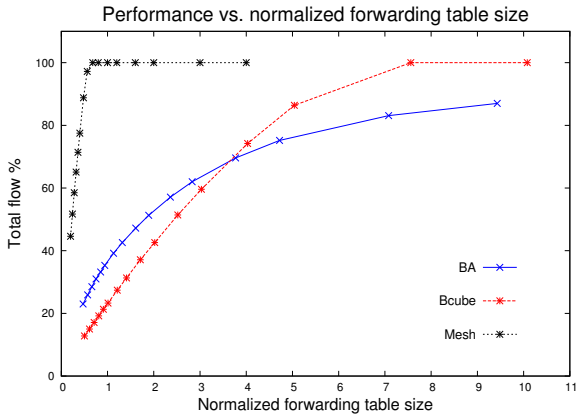

Fig. 7. Performance of Alg. 3 with respect to different NS values. The y-axis represents the percentage of the total flow achieved wr to the maximum flow that can be routed in the network.

V. Discussion

The vast interest and the fast adoption of SDN solutions introduce new interesting challenges. In this paper we address one of the important limitations of SDN devices - the limited sizes of their TCAM based forwarding tables. We conducted a thorough theoretical study of the bounded path-degree max flow problem, for maximizing network utilization in environments where the size of the forwarding table of the network devices is limited. We considered both the general version of the problem, as well as a practical setting in which there is a set of pre-defined paths over which the traffic can be routed, and developed bi-criteria randomized approximation algorithms that can be implemented in the practical setting with high efficiency.

[4]The discussion is valid only for the case of flow demands with uniform distribution. The load of other demand distributions would not be uniformly distributed over the network, and thus a different computation would be needed for different parts of the network topology.

We demonstrated the practical usefulness of these results by evaluating the expected performance gain in several typical networking scenarios. The results indicate that our forwarding table size aware algorithms can increase the total network flow by a factor of 2 in the backbone setting and by more than 50% in the data center setting, compared to current table size oblivious algorithms.

The setting we study is one of the first examples where the power of the SDN paradigm is used to overcome hardware related deployment issues. From the theoretical point of view, to the best of our knowledge, our work is the first to tackle network utilization and flow configuration under capacity constraints, as well as limited forwarding table sizes of network devices. Many more problems and future research directions arise in the SDN-traffic engineering context. Examples of such are the online version of our setting, the investigation of this framework with respect to other types of device constraints, or different network design problems for efficiently locating SDN-compliant devices within different networking infrastructures.

References

[1] S. Agarwal, M. Kodialam, and T.V. Lakshman. Traffic engineering in Software Defined Networks. *Proceedings of IEEE Infocom 2013*.

[2] R. Albert and A.L. Barabasi. Statistical mechanics of complex networks. *Reviews of Modern Physics*, Vol. 74, pp. 47-97, 2002.

[3] G. Baier, E. Kohler, and M. Skutella. The k-splittable fow problem. *Algorithmica*, Vol. 42, pp. 231178, 2005.

[4] J. Chen, R. Rajaraman, and R. Sundaram. Meet and merge: approximation algorithms for confluent flow. *Proceedings of the 35th ACM Symposium on Theory of Computing (STOC)*, pp. 373-382, 2003.

[5] J. Chen, R.D. Kleinberg, L. Lovasz, R. Rajaraman, R. Sundaram, and A. Vetta. (Almost) Tight bounds and existence theorems for single-commodity confluent flows. *Journal of the ACM*, Vol. 54(4), Article 16, 2007.

[6] P. Donovan, B. Shepherd, A. Vetta, and G. Wilfong. Degree-Constrained Network Flows. *Proceedings of the 39th annual ACM symposium on theory of computing (STOC)*, pp. 681-688, 2007.

[7] M. Faloutsos, P. Faloutsos, and C. Faloutsos. On power-law relationships of the Internet topology. *ACM SIGCOMM Comput. Commun. Rev.*, Vol. 29(4), pp. 251-262, 1999.

[8] N. Garg, and J. Könemann. Faster and simpler algorithms for multicommodity flow and other fractional packing problems. *SIAM J. Comput.*, Vol. 37(2), pp. 630-652, (2007)

[9] C. Guo, G. Lu, D. Li, H. Wu, X. Zhang, Y. Shi, C. Tian, Y. Zhang, and Songwu Lu. BCube: A high performance, server-centric network architecture for modular data centers. *Proceedings of the ACM SIGCOMM 2009 conference on Data communication*, 2009.

[10] T. Hartman, A. Hassidim, H. Kaplan, D. Raz, and M. Segalov. How to split a flow? *Proceedings of IEEE Infocom 2012*.

[11] J. M. Kleinberg. Single-source unsplittable flow. *Proceedings of FOCS 1996*, pp. 6817.

[12] S.G. Kolliopoulos. Edge-disjoint paths and unsplittable flow. *Handbook of Approximation Algorithms and Metaheuristics*, ser. Chapman and Hall/CRC, T.F. Gonzalez, Ed., 2007.

[13] C.E. Leiserson. Fat-trees: Universal networks for hardware-efficient supercomputing. *IEEE Transactions on Computers*, 1985.

[14] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. OpenFlow: enabling innovation in campus networks. *SIGCOMM Comput. Commun. Rev.*, Vol. 38(2), 2008.

[15] R. Narayanan, S. Kotha, G. Lin, A. Khan, S. Rizvi, W. Javed, H. Khan, and S.A. Khayam. Macroflows and Microflows: Enabling Rapid Network Innovation through a Split SDN Data Plane. *2012 European Workshop on Software Defined Networking (EWSDN)*, pp. 79-84, 2012.

[16] B. Stephens, A. Cox, W. Felter, C. Dixon, and J. Carter. PAST: scalable ethernet for data centers. *Proceedings of the 8th international conference on Emerging networking experiments and technologies (CoNEXT '12)*, pp. 49-60, 2012.