Fifty-third Annual Allerton Conference
Allerton House, UIUC, Illinois, USA
September 29 - October 2, 2015

Enabling Service Function Chaining through Routing Optimization in Software Defined Networks

Andrey Gushchin
Cornell University
Ithaca, New York 14850
avg36@conell.edu

Anwar Walid
Bell Labs, Alcatel-Lucent
New Providence, NJ 07974
anwar.walid@alcatel-lucent.com

Ao Tang
Cornell University
Ithaca, New York 14850
atang@ece.cornell.edu

***Abstract*—Routing in SDN-enabled networks with middleboxes enables high level of control over network flows so that desired level of security, efficiency and performance can be achieved. However, the new capabilities place constraints on available resources such as switch memory and middlebox processing capacity. Additionally, flow traversal constraints may also arise where each network flow has to visit certain middleboxes in a specific order. In this work, we investigate the problem of achieving global routing objectives while satisfying necessary constraints. We formulate the routing problem as an integer linear optimization problem, and propose an efficient randomized algorithm for solving it. We provide both theoretical and practical (using simulations) analysis for our algorithm. The simulations show that while other routing approaches (including greedy and previously proposed one) violate the switch memory capacity constraints (requiring 10 times more capacity in some cases), and could attain a low objective value, our algorithm satisfies all traversal and network resource constraints and achieves a high objective value.**

I. Introduction

Middleboxes (firewalls, proxies, IDS, WAN optimizers) are widely deployed [14] special devices that perform functional processing of traffic to facilitate achieving a desired level of performance and security. Traditional middlebox is a standalone physical device that is usually able to perform only one network function. Different network flows may require different functional processing, and the order in which the functions are applied may also be important, which complicates routing in the networks under limited resource constraints. Mechanism of controlling routing through the specified functional sequence is called Service Function Chaining (SFC). Software-Defined Networking (SDN) approach enables a fine-grained control of the network flows, which can alleviate routing of traffic demands in networks with middleboxes. Several relevant solutions are offered in the recent works (e. g. [1], [2], [3]).

Besides the traditional resource constraints such as link capacities, several new constraints have to be taken into account in routing design for SDN-enabled networks with middleboxes. For example, the number of routing rules that can be installed in a single switch is limited by the switch memory capacity. Ternary Content-Addressable Memory (TCAM) used in SDN switches is an expensive resource both in terms of cost and power consumption. A TCAM switch can generally support only a few thousand rules (for instance, 1500 rules in 5406zl switch [3]). Switch flow table overflow is a serious problem that can significantly degrade network performance and, therefore, should be avoided. Another constraint requires that the load on each middlebox does not exceed its processing capacity, which is an important factor in middlebox deployments ([4], [3]). In addition to the above resource constraints, routing must satisfy traversal constraints: required network functions have to be applied to a flow in a specific order. For example, a desired traversal sequence for a flow can be defined as firewall $\rightarrow$ IDS $\rightarrow$ proxy.

The goal of this work is to design an algorithm that routes the required demands (when possible), while satisfying network resource and traversal constraints. A routing obtained by the algorithm can be translated into a set of routing rules that are further installed in the switches. Besides satisfying the constraints, a routing should achieve a desired level of the objective function which may be defined, for example, in terms of the total network throughput, maximum link, middlebox, or switch utilization. Although our results are applicable to various choices of the objective function, in this work, for demonstrative purposes we concentrate on one particular objective which is the total throughput maximization under the fairness requirement that all traffic demands have to be satisfied up to the same scaling factor.

In this paper, to tackle the complexity of the routing problem, we utilize a probabilistic approach and propose a three-step randomization-based routing algorithm that satisfies traversal constraints and has the following features.

- Low computational complexity: the algorithm requires solving only two linear optimizations.
- Using careful manipulation of the Chernoff bound, we show (Theorem 1) that a routing obtained by the algorithm satisfies switch memory constraints with high probability. If violations exist, routing is modified so that all switch constraints are satisfied.
- Routing obtained by the algorithm tries to satisfy as many demands as possible, while satisfying link and middlebox capacity constraints (Theorem 2). It also allows to achieve a high value of the desired objective function.

The article is organized as follows: in section II we introduce necessary notations, define the problem setup, describe the network resource and traversal constraints. In addition, we formulate a general integer linear optimization problem whose optimal solution is a routing that satisfies all required constraints. Because this general problem belongs to the class of NP-hard problems, alternative approaches are

978-1-5090-1824-6/15/$31.00 ©2015 IEEE

required to obtain a suitable routing solution in an efficient way. In section III we describe our algorithm and provide a theoretical justification for it. In section IV we evaluate its performance and compare it with several other routing approaches. We further discuss related works in section V, and conclude in VI.

II. Problem Formulation

A. Network Resources and Commodities

We assume that the network topology is defined by a directed graph $G = (V, E)$, where V is a set of nodes and E is a set of its edges. Each node is either a switch or a middlebox. We will denote the set of switch nodes by V_S, and the set of nodes corresponding to middleboxes by V_M, then $V_S \cup V_M = V$. If n_S and n_M are the numbers of switches and middleboxes in the network, respectively, then $n_S + n_M = n$, where n is the total number of nodes in graph G.

An SDN switch possesses a memory capacity that can be used to install a certain number of routing rules in a switch. Let $b(v)$ denote the memory capacity of switch $v \in V_S$, i.e. the maximum number of routing rules that it can accommodate. Further, $c(e)$ is the capacity of link $e \in E$, and $g(m)$ is a processing capacity of middlebox $m \in V_M$, i.e. the maximum amount of traffic that it can process per unit of time.

There are several types of network functions (WAN optimization, proxy, firewall, etc.) that can be applied to routed traffic, and each middlebox can perform one of these functions. In the rest of the article we will use terms middlebox and network function interchangeably. Our results, however, can be also applied to the case of consolidated middleboxes when each device can perform all or a set of network functions ([5], [6], [7]). It is assumed that several middleboxes performing same network function may exist in the network. Although middleboxes may transform traffic which could sometimes result in an increase or a decrease of the traffic volume, such transformations are not considered in this work.

In addition, a set of K traffic demands or "commodities" $a_i =< s_i, t_i, d_i, z_i >$, $i = 1, \ldots, K$ is defined for the network. Here s_i and t_i are, respectively, the source node and the destination node, d_i is the required demand, i.e. an amount of flow that has to be routed for commodity a_i, and it is assumed that all demands d_i, $i = 1, \ldots, K$, are known in advance or can be estimated. Finally, z_i is a class of the commodity a_i and is defined by an ordered set of required network functions (for example, firewall→ IDS→proxy) that have to be applied to the commodity's traffic. The main notations are summarized in Table I.

B. Routing via Integer Linear Optimization

Flow for each commodity $a_i =< s_i, t_i, d_i, z_i >$ is routed via a set of its admissible paths. A path p is called an admissible path for commodity a_i, if it starts at the source node s_i, ends at the destination t_i, and traverses all middleboxes required by the commodity's class z_i, in a correct order. In that case we will write $p \in a_i$. Using only admissible paths for routing traffic enables satisfying the traversal requirements. Therefore, a single commodity's traffic can be split into several admissible paths. The splitting, however, is performed at the IP flow level and not at the packet level, so that all packets of a flow follow the same path. This is similar to Equal Cost MultiPath [8] in data centers, where hashing is utilized to split traffic at the IP level to route on multiple paths. As in [9], we assume that for every commodity a set of admissible paths is provided or can be found using for example, "k-shortest" paths algorithm. The set of admissible paths for all commodities is denoted by $\mathcal{P}$.

TABLE I: Main notations.

$b(v)$	capacity of switch $v \in V_S$
$c(e)$	capacity of link $e \in E$
$g(m)$	capacity of middlebox $m \in V_M$
$c(p)$	capacity of path $p \in \mathcal{P}$
$v(p)$	number of rules required in switch $v \in V_S$ for path $p \in \mathcal{P}$
a_i $< s_i, t_i, d_i, z_i >$	commodity (traffic demand) a_i with source s_i, destination t_i, demand d_i, and class z_i
$V_S \subset V$	set of switch nodes
$V_M \subset V$	set of middlebox nodes

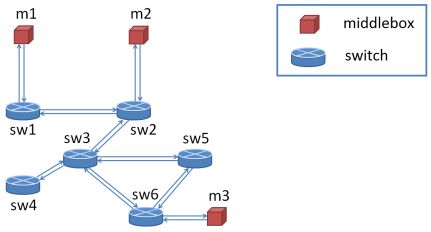


Fig. 1: Example of a network topology with $n_S = 6$ switches and $n_M = 3$ middleboxes. Path $sw1 \rightarrow m1 \rightarrow sw1 \rightarrow sw2 \rightarrow m2 \rightarrow sw2 \rightarrow sw3 \rightarrow sw5$ requires two routing rules to be installed in each of the switches $sw1$ and $sw2$.

We will write $v \in p$ to designate that switch $v \in V_S$ belongs to the path p, $e \in p$ means that link $e \in E$ is part of the path p, and finally, notation $m \in p$ means that middlebox $m \in V_M$ belongs to the path p. It is possible that an arbitrary path p traverses a switch node $v \in p$ more than once. For example, if $a_i =< sw1, sw5, d_i, z_i >$ is a commodity in a network from Fig. 1, where $z_i = \{m1, m2\}$, and path $p \in a_i$ is defined as $sw1 \rightarrow m1 \rightarrow sw1 \rightarrow sw2 \rightarrow m2 \rightarrow sw2 \rightarrow sw3 \rightarrow sw5$, then flow along the path p traverses each of switches $sw1$ and $sw2$ twice. Let $v(p)$ denote the number of times the path p visits node $v \in V_S$. Then, $v(p)$ is also equal to the number of routing rules that must be installed in switch v if path p is used for routing. For simplicity it will be assumed that each path p cannot traverse the same directed link more than once, and only paths going through any link at most once will be used for routing traffic demands. It is possible, however, to generalize our work to the case when a path is allowed to traverse a link more than once, in a similar way as we handle the paths traversing a switch several times. We suppose additionally that no path can visit the same middlebox more than once. Under our assumptions, any path p traverses each directed link at most once, but can visit a switch $v \in p$ more than once. Then, the switch v can use the incoming interface to distinguish traffic on the path p that visits v for the first time, second time, etc. In this paper, a routing scheme or a routing solution is defined as a map that assigns to each path $p \in \mathcal{P}$ a nonnegative value corresponding to the amount of flow that is sent along the path p.

Notice, that due to the resource constraints, it may not

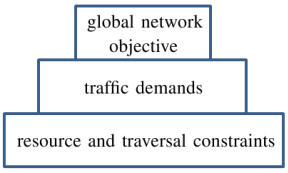


Fig. 2: Components of the routing problem.

be possible to satisfy all required demands [1], and in that case we try to satisfy as many demands as possible. In addition to satisfying the demands, network resource and traversal constraints, a routing solution should achieve a certain objective. Therefore, the process of designing a routing solution should take into account three components that constitute a pyramid in Fig. 2 that we use for illustration. The first component corresponding to the bottom of the pyramid, is the resource and traversal conditions that must be always met by any routing solution. Next component is the demands that sometimes may not be satisfied all together due to the resource constraints. A desired routing solution, however, satisfies as many demands as possible. Finally, on top of the pyramid is the global network objective, and an appropriate routing solution tries to achieve its optimal value by efficiently routing traffic for the commodities whose demands can be satisfied. The global objective can be defined as maximizing the total network throughput without the fairness requirement, or as minimizing the throughput while satisfying the demands (if the goal is to route all demands with the minimum possible amount of network resources). Another possible objective is to route all traffic demands so that the maximum switch (middlebox, link) utilization is minimized.

Our routing algorithm is suitable for various choices of the global network objective function. For illustrative purposes, we select (as our objective), the total throughput maximization under a fairness requirement (also known as maximization of fractional demands) which is considered in the rest of the article. Using this network objective, a corresponding integer linear optimization problem (1) can be formulated that utilizes the precomputed set of admissible paths $\mathcal{P}$. This optimization problem maximizes the demand scaling factor D such that for each commodity its demand is satisfied with a factor of D. Therefore, the objective function takes into account demands of the commodities, and an amount of flow routed for each commodity in a solution to this optimization is proportional to its demand. This choice of the objective function allows to find routing solutions that achieve fairness among the commodities in terms of the amount of routed flow. Moreover, an optimal solution of (1) satisfies all resource and routing constraints. Variable $f(p) \geq 0$ of the optimization problem defines an amount of flow that is sent along path p.

$$\textbf{max } D,$$

$$\forall v \in V_S : \sum_{p:\ v \in p} \delta(f(p)) \cdot v(p) \leq b(v), \tag{1a}$$

$$\forall e \in E : \sum_{p:\ e \in p} f(p) \leq c(e), \tag{1b}$$

$$\forall m \in V_M : \sum_{p:\ m \in p} f(p) \leq g(m), \tag{1c}$$

$$\forall a_i : \sum_{p:\ p \in a_i} f(p) \geq D \cdot d_i, \tag{1d}$$

$$\forall p \in \mathcal{P} : f(p) \geq 0. \tag{1e}$$

In this optimization problem constraint (1a) is a switch memory constraint. Here $\delta(f(p))$ is a binary-valued indicator function for path p defined as follows:

$$\delta(f(p)) = \begin{cases} 1, & \text{if } f(p) > 0; \\ 0, & \text{if } f(p) = 0. \end{cases} \tag{2}$$

Therefore, if path p is used for routing (i.e. if $f(p) > 0$), then corresponding routing rules have to be installed in switch $v \in p$, and the number of required rules in switch v for path p is $v(p)$. Next, (1b) is a link capacity constraint, (1c) is a middlebox processing capacity constraint, and (1d) is the demand constraint. By (1e) we require that all flow variables $f(p)$ take nonnegative values. As implied by constraints (1c), it is assumed for simplicity that for each middlebox, the per-packet processing cost does not depend on a particular flow. This assumption can be easily generalized, however, by multiplying each variable $f(p)$ in (1c) by the corresponding (to path p) per-packet processing cost.

Optimization problem (1) belongs to the class of integer linear programs (ILP) due to the integer-type switch memory constraint (1a). Therefore, the problem is NP-hard [9], and it is generally not easy to obtain an optimal solution to (1) efficiently. Notice, that the integer-type switch memory constraints do not depend on the global network objective and, thus, the routing problem is NP-hard regardless of the chosen objective. Another drawback of the ILP (1) formulation is that its optimal solution may not satisfy as many demands as possible. For example, when the switch memory capacities are not sufficient, it may not be possible to provide even a single path for each commodity and thus, to obtain a solution to (1) with a positive value of D. To overcome these difficulties, we propose a low complexity randomization-based algorithm that allows us to find a routing solution satisfying switch memory constraints (1a) and resource requirements (1b) and (1c). The algorithm also tries to satisfy as many demands as possible, and achieves a high value of the objective function.

III. Randomized Routing Algorithm

In this section we describe and analyze our routing approach. The algorithm contains three main steps. At the first step a fairness-aware multicommodity routing optimization problem (4) is solved which is similar to problem (1), but has different variables and modified switch memory constraints. We emphasize that a fairness-aware problem (4) is solved at the first step of the algorithm regardless of the choice of the global objective function. This optimization problem belongs to the class of linear optimization problems and therefore, can be efficiently solved. Similarly to (1), the objective of this optimization problem is to maximize the total flow under the fairness requirement. After a solution to this linear program is obtained, the second, randomization step is initiated. At this step, using a solution to (4), the paths for routing traffic are chosen with certain probabilities. We analyze the randomization step and prove in Theorem 1 that a routing solution utilizing only the chosen paths satisfies switch memory constraints with high probability. Moreover, in Theorem 2 we show that it is possible to obtain a routing using only these paths such that all network resource constraints are satisfied with high probabilities, and provide a lower bound on the expected value of routed flow for each commodity. At the third step of the algorithm, using paths chosen at the randomization step, an additional

[1] A routing solution satisfies the demand of a commodity a_i if at least d_i units of flow is sent for this commodity in the routing solution.

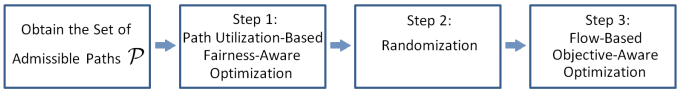


Fig. 3: Schematic of our routing algorithm.

multicommodity flow linear optimization problem is solved. In contrast to the optimization (4) of the first step, the objective of this optimization is the same as the global network objective. However, because we chose the fairness-aware throughput maximization as the global network objective, in this case optimization (8) solved at the third step has the same objective function as in problems (1) and (4). Therefore, the original integer linear routing problem (1) is decomposed into two parts: we first choose a suitable set of admissible paths at the first two steps via randomization, and then achieve a high value of the network objective by solving optimization (8). The schematic of our routing algorithm is shown in Fig. 3.

A. Step 1: Path Utilization-Based Fairness-Aware Optimization

At the first step of our approach we solve a fairness-aware linear optimization problem (4) in which integer-type switch memory constraints are replaced by their linear alternatives. The resulting linear relaxation is a multicommodity flow problem in its path-flow formulation. To facilitate a further application of the randomization step, the paths' utilizations variables $x(p)$ taking values in the interval $[0,1]$ are used in (4), and thus, this problem is a path utilization-based optimization problem. In addition to the utilization of a path, we need to define the path's capacity. Let $c(p)$ denote the capacity of a path $p \in a_i$ which is defined as

$$c(p) = \min\{\min_{e\in p} c(e), \min_{m\in p} g(m), d_i\}, \tag{3}$$

i.e. it is the minimum between the bottleneck link capacity belonging to path p, the bottleneck middlebox capacity on path p, and demand of the commodity a_i associated with path p. Bounding the path's capacity by the demand allows to obtain a set of utilization variables (a solution to (4)) that take into account values of the demands, and it helps to improve the choice of paths at the randomization step. The linear optimization problem that is solved at the first step of our algorithm is formulated as follows:

$$\textbf{max } D,$$

$$\forall v \in V_S : \sum_{p:\ v\in p} x(p)\cdot v(p) \le b(v), \tag{4a}$$

$$\forall e \in E : \sum_{p:\ e\in p} x(p)\cdot c(p) \le c(e), \tag{4b}$$

$$\forall m \in V_M : \sum_{p:\ m\in p} x(p)\cdot c(p) \le g(m), \tag{4c}$$

$$\forall a_i : \sum_{p:\ p\in a_i} x(p)\cdot c(p) \ge D\cdot d_i, \tag{4d}$$

$$\forall p \in \mathcal{P} : x(p) \in [0,1]. \tag{4e}$$

Constraint (4a) is a modified switch memory constraint, (4b) is a link capacity constraint, (4c) is a middlebox processing capacity constraint, and (4d) is the demand constraint. By (4e) we require that all utilization variables $x(p)$ take values from the interval $[0,1]$. Similarly to (1), this optimization problem is fairness-aware: amount of flow routed for every commodity in its optimal solution is proportional to the commodity's demand, and the coefficient of proportionality is the same for all commodities.

Optimization problem (4) is an instance of linear programming, and its solution can be efficiently found. An optimal solution, however, while satisfying modified switch memory constraints (4a), may violate the original integer-type constraints (1a). Therefore, to guarantee satisfaction of constraints (1a), an additional processing of an optimal routing solution to (4) is required. This processing is performed via randomization and constitutes the second step of our algorithm.

B. Step 2: Randomization

The second step of the algorithm uses a solution to (4) obtained at the previous step and defined by the values of utilization variables $x(p)$ and variable D. In particular, at the second step each path p is chosen independently with probability $x(p)$. This operation is well-defined since utilization variables $x(p)$ take values from the interval [0,1]. We justify such choice of paths in Theorem 1, where it is proved that any routing solution that utilizes only chosen paths, satisfies switch memory constraints (1a) with high probability. Furthermore, in Theorem 2 we demonstrate that it is possible to obtain a routing solution using only chosen paths such that all network resource constraints are satisfied with high probabilities, and provide a lower bound on the expected value of routed flow for each commodity in this routing solution. This lower bound for each commodity is proportional to the commodity's demand and, therefore, fairness is implied in this routing solution. The implied fairness facilitates satisfying as many demands as possible in contrast to sending as excessive amount of flow for one commodity at the expense of not satisfying demands of the other commodities. In our analysis we will use the Chernoff bound in the following form.

If independent random variables x_i ($i = 1,\dots,n$), take values in $[0,1]$, $X = \sum_{i=1}^{n} x_i$, and $\mathrm{E}[X] \le \mu$, then:

$$\mathrm{P}\Big[X \ge (1+\epsilon)\mu\Big] \le e^{-\frac{\epsilon^2}{2+\epsilon}\mu}. \tag{5}$$

We additionally define for each switch node $v \in V_S$

$$\bar{v} \triangleq \max_{p:v\in p} v(p)$$

to be the maximum number of rules required to be installed in switch v among all paths going through v.

Theorem 1 *Let $\big(x(p), D\big)$ be a solution to optimization* (4)*. If each path $p \in \mathcal{P}$ is independently chosen with probability $x(p)$, then any routing solution utilizing only chosen paths can violate switch memory constraint* (1a) *by a factor more than $(1+\epsilon_0)$ with probability not greater than $\frac{1}{n}$, where $\epsilon_0 > 0$ satisfies*

$$b(v) \ge \frac{\bar{v}\cdot(4+2\epsilon_0)\cdot\log n}{\epsilon_0^2} \tag{6}$$

for each switch v.

Proof: We will show that if condition (6) holds, then switch memory constraints (1a) are satisfied with high probabilities. We consider an arbitrary switch node $v \in V_S$ and associate with each path $p : v \in p$ a random variable y_p defined as follows:

$$y_p = \begin{cases} v(p), & \text{if } p \text{ is selected (with probability } x(p)); \\ 0, & \text{otherwise.} \end{cases}$$

Let X be a random variable that corresponds to the number of routing rules in switch v after the randomization step is applied, then

$$X = \sum_{p:\ v \in p} y_p.$$

Moreover, because each path p is independently selected with probability $x(p)$, and since constraints (4a) are satisfied, the expected number of routing rules in switch v is:

$$\mathrm{E}[X] = \sum_{p:\ v \in p} x(p) \cdot v(p) \leq b(v).$$

We now consider additional variables $\hat{y}_p$ that are defined as

$$\hat{y}_p = \begin{cases} \frac{v(p)}{\bar{v}}, & \text{if } p \text{ is selected (with probability } x(p)); \\ 0, & \text{otherwise,} \end{cases}$$

and let $\hat{X} = \sum_{p:\ v \in p} \hat{y}_p = \frac{1}{\bar{v}} \cdot X$, then

$$\mathrm{E}[\hat{X}] = \sum_{p:\ v \in p} \frac{x(p) \cdot v(p)}{\bar{v}} \leq \frac{b(v)}{\bar{v}}.$$

Since $v(p) \leq \bar{v}$ for any path p and switch $v \in p$, each random variable $\hat{y}_p$ takes values from the interval $[0, 1]$. Because each path p is selected independently, variables $\hat{y}_p$ are independent, and we are ready to apply the Chernoff bound (5) with $\epsilon = \epsilon_0$ and $\mu = \frac{b(v)}{\bar{v}}$:

$$\mathrm{P}\Big[\hat{X} \geq (1+\epsilon_0) \cdot \frac{b(v)}{\bar{v}}\Big] \leq e^{-\frac{\epsilon_0^2}{2+\epsilon_0} \cdot \frac{b(v)}{\bar{v}}},$$

and if condition (6) holds, then

$$e^{-\frac{\epsilon_0^2}{2+\epsilon_0} \cdot \frac{b(v)}{\bar{v}}} \leq \frac{1}{n^2},$$

and thus

$$\mathrm{P}\Big[X \geq (1+\epsilon_0) \cdot b(v)\Big] \leq \frac{1}{n^2}.$$

The number of switch memory constraints is equal to the number of switches n_S, and $n_S \leq n$. Therefore, using the union bound, probability that the number of routing rules in an arbitrary switch v exceeds this switch's memory capacity by a factor more than $(1+\epsilon_0)$ is not greater than $\frac{1}{n}$. ■

Let $\hat{\mathcal{P}} \subseteq \mathcal{P}$ be the set of the admissible paths chosen by the randomization step.

Theorem 2 *If $\frac{c(p)}{(1+\hat{\epsilon})}$ units of traffic is sent along each chosen path $p \in \hat{\mathcal{P}}$, where $\hat{\epsilon}$ is the unique positive root of a quadratic equation*

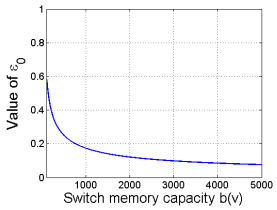
$$\epsilon^2 - 3\log n \cdot \epsilon - 6\log n = 0, \tag{7}$$

then the resulting routing solution:

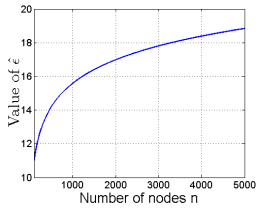
- *can violate link capacity constraint* (1b) *by a factor more than* 1 *with probability not greater than $\frac{1}{n}$;*
- *can violate middlebox processing capacity constraint* (1c) *by a factor more than* 1 *with probability not greater than $\frac{1}{n^2}$;*
- *routes for each commodity a_i a flow of expected value greater or equal to $\frac{D}{(1+\hat{\epsilon})} \cdot d_i$.*

The proof of the Theorem 2 is also based on the Chernoff bound and can be found in the extended version of this article [13].

To illustrate the relationship between the switch memory capacity $b(v)$ and parameter ϵ_0 satisfying inequality (6), in Fig. 4a we plotted the minimal value of ϵ_0 that satisfies (6) for different values of switch memory capacities $b(v)$, and for $n = 1000$, $\bar{v} = 1$. In Fig. 4b we show the dependence of the unique positive root of the quadratic equation (7) on the number of nodes n.



(a) Minimal value of ϵ_0 satisfying (6) as a function of $b(v)$ for $\bar{v} = 1$ and $n = 1000$.



(b) Dependence of the unique positive root $\hat{\epsilon}$ of (7) on the number of nodes n.

Fig. 4: Dependence of ϵ_0 on $b(v)$ and of $\hat{\epsilon}$ on n.

Routing solution considered in the Theorem 2 satisfies all network resource constraints (1a), (1b) and (1c) with high probabilities. Although the expected value of flow routed for each commodity a_i, $i = 1, \ldots, K$, is bounded below by $\frac{D}{(1+\hat{\epsilon})} \cdot d_i$, demand constraints (1d) may be violated. In addition, the total network throughput may suffer from a possibly non-optimal choice of flow amount $\frac{c(p)}{(1+\hat{\epsilon})}$ routed through each path p. To overcome this problem, we suggest the third step described in the following subsection: using only paths that were chosen by randomization, we achieve a high value of the global network objective by solving a multicommodity flow problem in a path-flow formulation, whose variables are the amounts of flow along each path.

C. Step 3: Flow-Based Objective-Aware Optimization

At the core of the third step is a global network objective-aware linear optimization problem (8) that utilizes paths $\hat{\mathcal{P}}$ chosen at the randomization step. Let $\hat{\mathcal{P}}_i$ be the set of chosen admissible paths for commodity a_i, where $i = 1, \ldots, K$. Similarly to optimization problem (1), variable $f(p)$ in optimization (8) of the third step corresponds to a path $p \in \hat{\mathcal{P}}$ and defines an amount of flow that is sent along this path. Variables $f(p)$ are allowed to take arbitrary nonnegative values. Objective function of the throughput maximization problem (8) is the same as the global network objective i.e. the throughput is maximized under the fairness constraint (8c).

$$\textbf{max } D,$$

$$\forall e \in E: \sum_{p:\ e \in p} f(p) \leq c(e), \tag{8a}$$

$$\forall m \in V_M: \sum_{p:\ m \in p} f(p) \leq g(m), \tag{8b}$$

$$\forall a_i: \hat{\mathcal{P}}_i \neq \varnothing: \sum_{p:\ p \in \hat{\mathcal{P}}_i} f(p) \geq D \cdot d_i, \tag{8c}$$

$$\forall p \in \hat{\mathcal{P}}: f(p) \geq 0. \tag{8d}$$

This optimization problem uses the paths that were chosen by the randomization step, and does not contain switch memory constraints. Notice that constraint (8c) must be satisfied only for commodities that have at least one path chosen by the randomization step. For commodities without

any chosen paths this constraint can be satisfied only with $D \leq 0$, and including such commodities into constraint (8c) would lead to the optimal objective function value equal to zero. Therefore, when the switch memory capacities are too low to satisfy all traffic demands, our algorithm tries to maximize routed flow for a subset of commodities. In this work we focus on a problem setup when the switch memory is the most scarce network resource, i.e. it may not be possible to satisfy demands for all commodities due to the switch memory constraints, and not because of link or middlebox capacity constraints, and in that case $D > 1$ in the optimal solution to (8). Because the probability of switch memory constraints violations in a routing solution obtained by optimization (8) is nonzero, we additionally remove from a solution the paths that violate the constraints in a following way. We sort paths in $\hat{\mathcal{P}}$ by the value of $f(p)$ in the solution to (8) in the ascending order, and then remove paths that traverse overloaded switches one by one, until all switch memory constraints are satisfied.

Algorithm 1: Randomized Routing Algorithm

Input : Set of admissible paths $\mathcal{P}$ created using, for example, "k-shortest" paths algorithm.

1 Find a solution $\big(x(p), D\big)$, where $p \in \mathcal{P}$, to the Path Utilization-Based Fairness-Aware optimization problem (4).

2 Obtain the path-set $\hat{\mathcal{P}}$ by choosing each path $p \in \mathcal{P}$ with probability $x(p)$.

3 Find a solution $\big(f(p), D\big)$, where $p \in \hat{\mathcal{P}}$, to the Flow-Based Objective-Aware optimization problem (8). If not all switch memory constraints are satisfied, sort paths in $\hat{\mathcal{P}}$ by value $f(p)$ in ascending order, and remove paths going through overloaded switches one by one, until all switch memory constraints are satisfied.

The complete three-step algorithm is listed in Algorithm 1. In this algorithm the original integer linear routing problem (1) is decomposed into two parts: we first choose a suitable set of admissible paths $\hat{\mathcal{P}}$ at the first two steps via randomization, and then achieve a high value of the objective by solving optimization (8). As shown in the Theorems 1 and 2, the paths for routing provided by randomization step are chosen in a special way. In particular, it was proved that using these selected paths, the network traffic can be routed so that the switch memory constraints are satisfied with high probability, and implied fairness among the commodities in the path selection process of Step 2 facilitates satisfying as many traffic demands as possible.

Different choices of the objective function can be handled by our randomized algorithm. In particular, factor D, the objective function of optimization (8) can be replaced by a different objective. Regardless of the objective function, however, we suggest using path utilization-based fairness-aware optimization (4) at the first step of our algorithm, which allows to obtain a suitable set of admissible paths $\hat{\mathcal{P}}$.

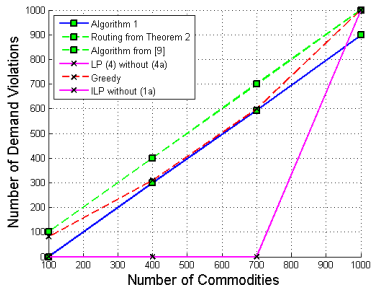
IV. Evaluation

A. Simulation Setup

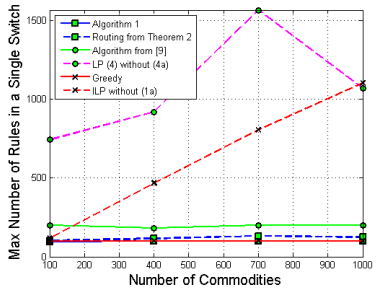
In this section we provide the results of simulations implemented to analyze the performance of our algorithm, and to compare it with several other routing approaches. Similarly to the previous sections, the global network objective used in evaluation is the total throughput maximization under the fairness requirement. We explore routing solutions obtained by our routing approach and other algorithms; in particular, in each experiment: 1) we find average and maximum numbers of routing rules in switches; 2) for every commodity we calculate the value of D showing in what proportion the commodity's demand is satisfied; 3) estimate number of commodities whose demands are satisfied, i. e. commodities with $D \geq 1$; 4) calculate an average value of D among all commodities, it allows to evaluate the total network throughput.

Simulations are performed for various setups: we consider three different network topologies: a "toy" network from Fig. 1 with $n_S = 6$ and $n_M = 3$, Geant network with $n_S = 41$, $n_M = 9$, and Fat Tree topology with $n_S = 84$, $n_M = 4$. Additionally, we vary the number of commodities and memory capacities of the switches. In every network topology under consideration, each middlebox is connected to a single switch via bi-directional links as shown in the example from Fig. 1. The middlebox processing capacity constraints are combined with link capacity constraints: if we define the capacity of a link connecting a switch with a middlebox as the minimum between its real capacity and the corresponding middlebox capacity, then the resulting link capacity constraint will guarantee that the total flow to the middlebox does not exceed its processing capacity. In the Geant network, middleboxes are connected to the switch nodes with the highest nodal degree in the graph defining the Geant topology. The Fat Tree topology consists of 4 core, 16 aggregate and 64 edge switches, where each aggregate switch is connected to 4 edge switches, and each core switch is connected to a single middlebox. For simplicity it is assumed in the simulations that there exists only one network function that has to be applied to all network flows. Therefore, all commodities belong to the same class defined by this network function. The set of admissible paths for each commodity is obtained using "k-shortest" paths approach based on Yen's algorithm [10]. In particular, k shortest paths are found from each source switch to each middlebox, and then k shortest paths from each middlebox to the destination switch. We used values $k = 3$ for the "toy" topology, and $k = 1$ for the Geant and Fat Tree networks. Link capacities were equal to 300, 300 and 500 for the "toy", Geant and Fat Tree topologies, respectively.

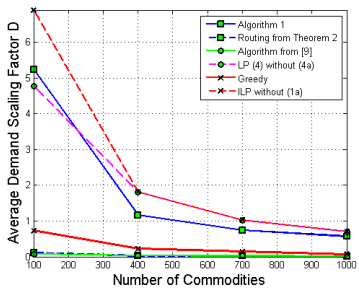
We analyzed and compared the following routing algorithms: 1) Algorithm 1; 2) routing solution defined in Theorem 2; 3) randomized algorithm from [9]; 4) routing solution obtained by solving optimization (4) without switch memory constraints; 5) greedy approach that uses the previous routing solution and removes from it paths with smallest amount of flow to satisfy switch capacity constraints; 6) routing solution obtained by solving optimization (1) without switch memory constraints. Notice that although in Algorithm 1 some paths can be removed if the switch memory constraints are violated, in the routing solution from Theorem 2 all paths obtained at the randomization step are utilized for routing, and violation of switch memory constraints is possible. Therefore, the simulation results allow us to assess how likely the path selection at the randomization step leads to a switch constraints violation. The fourth routing solution in our analysis is obtained by solving optimization (4) without switch memory constraints. Comparison of this solution with Algorithm 1 helps to estimate an amount of throughput that is lost due to the switch memory constraints and to the choice of paths at the randomization step. This routing solution is likely to violate the switch capacity constraints, and the greedy algorithm (fifth algorithm in the comparison analysis) further deletes the paths traversing overloaded switches until all switch memory constraints are satisfied.



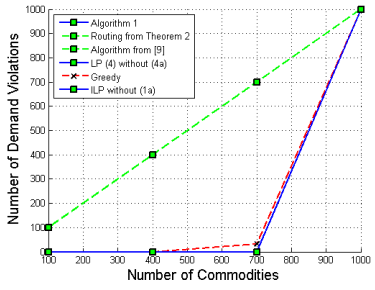
(a) Number of demand violations, $b(v) = 100$.



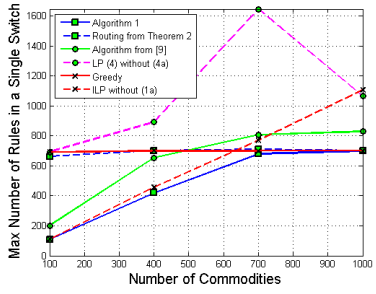
(b) Maximum number of required rules in a single switch, $b(v) = 100$.



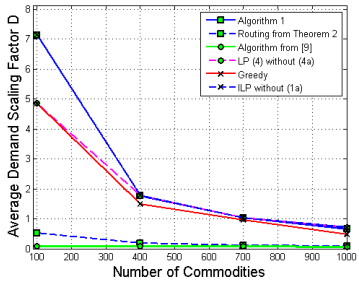
(c) Average demand scaling factor D, $b(v) = 100$.



(d) Number of demand violations, $b(v) = 700$.



(e) Maximum number of required rules in a single switch, $b(v) = 700$.



(f) Average demand scaling factor D, $b(v) = 700$.

Fig. 5: Simulation results for "toy" topology and switch memory capacities $b(v) = 100$ and $b(v) = 700$. Same line color and style is used for algorithms if their graphs coincide (e.g. Routing from Theorem 2 and Algorithm from [9] in Figs. 5a and 5d).

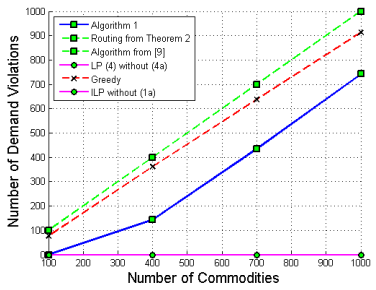
Similarly to Algorithm 1, the paths are removed in an ascending order of flow routed through them. We compare greedy algorithm with Algorithm 1 to evaluate if the path selection by randomization outperforms the simple path choice of the greedy algorithm. In addition, to estimate the value of the maximum total network throughput that can be achieved, a routing scheme obtained by solving optimization (1) without integer switch memory constraints is included into the comparison analysis.

B. Simulation Results

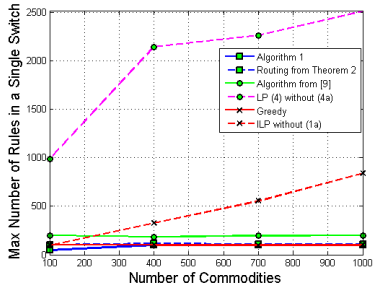
Figures 5, 6 and 7 illustrate the results of simulations for the "toy", Geant and Fat Tree topologies, respectively. Top three subfigures (a, b and c) of each figure correspond to the switch memory capacity $b(v) = 100$, and bottom subfigures (d,e and f) correspond to the switch memory capacity $b(v) = 700$. In each experiment all switches in the network have equal memory capacities. For a fixed network topology and switch memory capacity, we vary the number of commodities (x-axis in each subfigure) from 100 to 1000. Source node, destination node and demand for each commodity were chosen randomly in each experiment, and each demand was sampled uniformly from the interval [1, 1.5]. The figures show how the number of demand violations (subfigures a and d), maximum number of required rules in a single switch (subfigures b and e), and average demand scaling factor D (subfigures c and f) change with the number of commodities for each routing algorithm. In the figures we used the same line color and style for algorithms if their graphs coincided. For example, routing solution from Theorem 2 and algorithm from [9] violate all demands in each experiment, and we used green dashed line for both of them in subfigures a and d.

It can be observed that when the switch memory capacities are too low ($b(v) = 100$) for the given amount of commodities, Algorithm 1 is not capable to satisfy all demand constraints. However, it performs better than the greedy approach, the routing solution considered in the Theorem 2, and the algorithm from [9]. When the switch memory capacities are sufficiently large ($b(v) = 700$), Algorithm 1 performs as well as optimizations (1) and (4) without switch memory constraints. Additionally, it generally outperforms other routing approaches under consideration. Fig. 5a and 5d illustrate that for a large number of commodities (1000) even optimizations (1) and (4) without switch memory constraints violate all demands. This happens due to the link capacity constraints: value of D achieved by these optimizations is less than one.

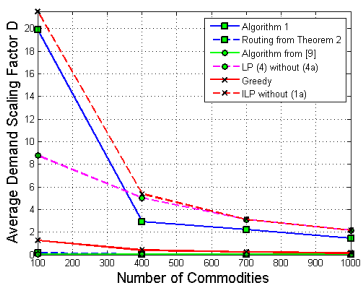
While Algorithm 1 and greedy algorithm do not violate switch memory constraints (maximum number of rules in any switch is not greater than the switch memory capacity in every experiment), other routing solutions can violate these constraints. In particular, routing solutions corresponding to optimizations (1) and (4) without switch memory constraints violate these constraints by a factor of 10-25 in some cases, as can be observed, for example, in Fig. 5b and Fig. 6b. It can also be seen that the routing solution from Theorem 2 violates the switch memory capacities in some experiments, but the amounts of violations are relatively small. Therefore, only a small number of paths has to be removed at the last step of Algorithm 1.



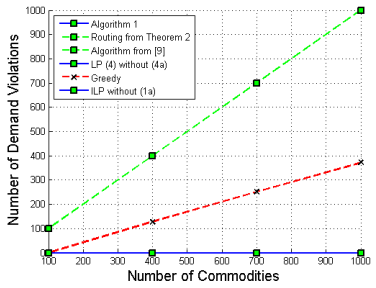
(a) Number of demand violations, $b(v) = 100$.



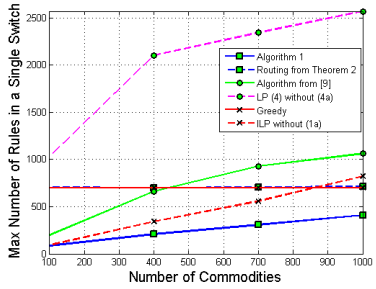
(b) Maximum number of required rules in a single switch, $b(v) = 100$.



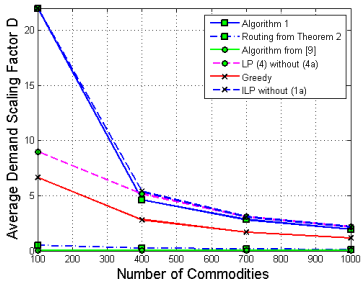
(c) Average demand scaling factor D, $b(v) = 100$.



(d) Number of demand violations, $b(v) = 700$.



(e) Maximum number of required rules in a single switch, $b(v) = 700$.



(f) Average demand scaling factor D, $b(v) = 700$.

Fig. 6: Simulation results for Geant topology and switch memory capacities $b(v) = 100$ and $b(v) = 700$. Same line color and style is used for algorithms if their graphs coincide.

From subfigures c and f of each of the three figures it can be observed that the performance of Algorithm 1 is close to the performance of optimization (1) without switch memory constraints, and therefore, Algorithm 1 allows to achieve the value of total network throughput that is close to the maximum possible value. Such high performance is possible due to the third step of Algorithm 1, where the flow-based objective-aware optimization problem (8) is solved.

Therefore, our simulations demonstrate that routing solutions obtained by Algorithm 1 satisfy a large portion of demands depending on the amounts of available network resources, and achieve a high value of the total network throughput, while satisfying all resource and routing constraints. Although in some experiments the routing solutions corresponding to optimizations (1) and (4) without switch memory constraints demonstrate a better performance compared to Algorithm 1 (in terms of demand violations and average D), these solutions in most cases violate the switch memory constraints, and thus, cannot be implemented in the network.

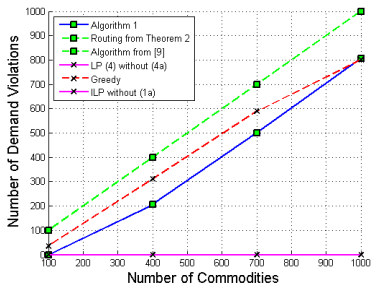
V. Related Work

A randomized algorithm for routing in SDN-enabled networks with integer-type switch memory constraints is presented in [9]. Although this algorithm and our work utilize similar randomization-based techniques, there are several important differences between the algorithms. One of the main differences is that networks considered in [9] do not contain middleboxes, and traffic for each commodity has to be routed from its source to destination without visiting special middlebox nodes, which significantly simplifies the routing problem. For example, routing optimization problems considered in [9] do not contain middlebox capacity constraints. In addition, any admissible path used for routing does not visit a single switch more than once. As shown by simulations, algorithm from [9] is not suitable for fairness-aware routing in SDN networks with middleboxes: it can significantly violate the switch capacity constraints, does not satisfy demands and achieves low average value of D, i.e. low total throughput.

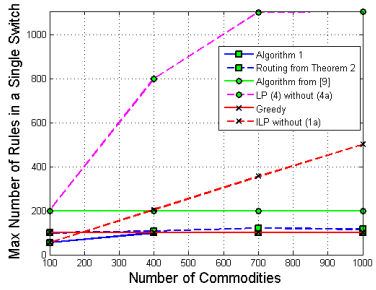
Routing approach in [3], similarly to our work, is designed for routing in networks with middleboxes. The original integer linear optimization problem is decomposed into two smaller subproblems, one of which is also of an integer linear type, and another belongs to the class of linear problems. The first (offline) subproblem allows to find a set of admissible paths for the commodities, while the objective of the second (online) subproblem is to route all traffic demands so that the maximum middlebox utilization is minimized.

Besides [3], there is a line of works ([11], [5]) that study the problem of routing in networks with middleboxes and aim to achieve a fair load balancing among middleboxes or other types of network resources. For instance, in [11] an optimization model for Network Intrusion Detection Systems (NIDS) is presented. However, most of these works do not solve a problem of finding suitable paths for routing, and assume that a set of paths is provided. In other words, the integer-type switch memory capacity constraints are not explicitly incorporated and discussed in [11] and [5].

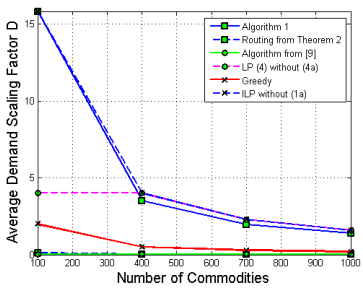
In [12] middleboxes run as virtual machines at the physical machines, and a flow-maximization routing linear optimization problem is proposed. The integer switch memory constraints are not incorporated into the routing problem,



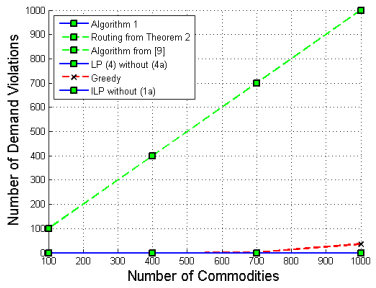
(a) Number of demand violations, $b(v) = 100$.



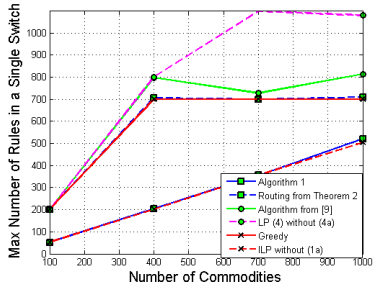
(b) Maximum number of required rules in a single switch, $b(v) = 100$.



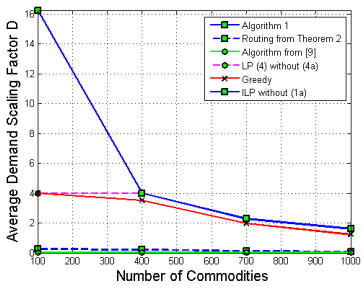
(c) Average demand scaling factor D, $b(v) = 100$.



(d) Number of demand violations, $b(v) = 700$.



(e) Maximum number of required rules in a single switch, $b(v) = 700$.



(f) Average demand scaling factor D, $b(v) = 700$.

Fig. 7: Simulation results for Fat Tree topology and switch memory capacities $b(v) = 100$ and $b(v) = 700$. Same line color and style is used for algorithms if their graphs coincide.

however, what makes it easier to find a routing solution. The authors also explore the problem of an optimal placement of middleboxes in the network.

VI. Conclusion

In this work we explored the problem of routing in SDN-enabled networks with middleboxes and switch memory constraints. Middleboxes are widely deployed network devices, and the number of them in a network can be comparable to the number of routers [14]. We formulated an integer linear optimization problem whose optimal solution satisfies all network resource (switch memory, link, middlebox processing capacities) and routing (correct sequence of middleboxes for each commodity) constraints, and can achieve high objective value. However, solution to this optimization problem cannot be efficiently obtained due to the integer-type switch memory constraints. Moreover, if it is not possible to satisfy all demands due to the resource constraints, the number of satisfied demands in this solution may be much lower than the maximum possible number. We then presented our three-step ramdomization-based routing algorithm with relatively low computational complexity, and provided a theoretical justification for it. We implemented simulations for various setups, and the results of simulations demonstrate advantages of our routing algorithm. In particular, it allows to achieve a high value of the network objective, while satisfying traffic demands (when possible) and network resource constraints.

References

[1] S. K. Fayazbakhsh, V. Sekar, M. Yu and J. C. Mogul. FlowTags: enforcing network-wide policies in the presence of dynamic middlebox actions. In *Proc. HotSDN*, 2013.

[2] A. Gember, R. Grandl, A. Anand, T. Benso and A. Akella. (2012). Stratos: Virtual middleboxes as first-class entities. UW-Madison TR1771.

[3] Z. Qazi, C. Tu, L. Chiang, R. Miao, V. Sekar and M. Yu. SIMPLE-fying middlebox policy enforcement using SDN. In *Proc. ACM SIGCOMM*, 2013.

[4] J. Sherry, S. Hasan, C. Scott, A. Krishnamurthy, S. Ratnasamy and V. Sekar. Making middleboxes someone else's problem: network processing as a cloud service. In *Proc. ACM SIGCOMM*, 2012.

[5] V. Sekar, N. Egi, S. Ratnasamy, M. Reiter and G. Shi. Design and implementation of a consolidated middlebox architecture. In *Proc. NSDI*, 2012.

[6] A. Greenhalgh, F. Huici, M. Hoerdt, P. Papadimitriou, M. Handley and L. Mathy. Flow processing and the rise of commodity network hardware. In *Proc. ACM SIGCOMM CCR*, 2009.

[7] J. W. Anderson, R. Braud, R. Kapoor, G. Porter and A. Vahdat. xOMB: extensible open middleboxes with commodity servers. In *Proc. ANCS*, 2012.

[8] M. Chiesa, G. Kindler and M. Schapira. Traffic Engineering with ECMP: An Algorithmic Perspective.

[9] R. Cohen, L. Lewin-Eytan, J. Naor and D. Raz. On the effect of forwarding table size on SDN network utilization. In *Proc. IEEE INFOCOM*, 2014.

[10] J. Y. Yen. An algorithm for finding shortest routes from all source nodes to a given destination in general networks. In *Quarterly of Applied Mathematics 27.4 (1970): 526.*

[11] V. Heorhiadi, M. K. Reiter and V. Sekar. New opportunities for load balancing in network-wide intrusion detection systems. In *Proc. CoNEXT*, 2012.

[12] M. Charikar, Y. Naamad, J. Rexford and X. K. Zou. Multi-Commodity Flow with In-Network Processing.

[13] https://www.dropbox.com/s/j2y9hc0ge02iei0/Infocom5extend.pdf?dl=0

[14] J. Sherry, S. Ratnasamy, and J. Sherry At. "A survey of enterprise middlebox deployments." (2012).