Semi-distributed Traffic Engineering for Elastic Flows in Software Defined Networks

Emmanuele Benedetto[*+], Ilario Filippini[+], Jocelyne Elias[†], Fabio Martignon[◇], and Yao Shen[*]

[*]Shanghai Jiao Tong University, Shanghai, China, email: *yshen@cs.sjtu.edu.cn*
[+]Politecnico di Milano, Milan, Italy, email: *emmanuele.benedetto@mail.polimi.it, ilario.filippini@polimi.it*
[†]University of Bologna, Bologna, Italy, email: *jocelyne.elias@unibo.it*
[◇]University of Bergamo, Bergamo, Italy, email: *fabio.martignon@unibg.it*

Abstract—**Software-Defined Networking (SDN) is becoming the reference paradigm to provide advanced Traffic Engineering (TE) solutions for future networks. However, taking all TE decisions at the controller, in a centralized fashion, may require long delays to react to network changes. With the most recent advancements in SDN programmability some decisions can (and should indeed) be offloaded to switches.**

In this paper we present a model to route *elastic demands* in a general network topology adopting a semi-distributed approach of the control plane to deal with path congestion. Specifically, we envision a Stackelberg approach where the SDN controller takes the role of *Leader*, choosing the most appropriate subset of routing paths for the selfish users (network switches), which behave as *Followers*, making local routing decisions based on path congestion. To overcome the complexity of the problem and meet the time requirements of real-life settings, we propose effective heuristic procedures which take into accurate account traffic dynamics, considering a stochastic scenario where both the number and size of flows change over time. We test our framework with a custom-developed simulator in different network topologies and instance sizes. Numerical results show how our model and heuristics achieve the desired balance between making global decisions and reacting rapidly to congestion events.

Index Terms—**SDN, Traffic Engineering, Stackelberg Game, Model and Heuristics, Elastic demands**

I. INTRODUCTION

Software-Defined Networking (SDN) has become the reference paradigm in both the industrial and academic worlds due to its potential to provide advanced Traffic Engineering (TE) solutions for current and future networks. It decouples control and data planes: the first is responsible for signaling traffic and routing, while the latter involves packet forwarding.

Traffic Engineering studies the optimization of network performance, reducing operational costs, and balancing the utilization of network resources. Several TE solutions performing load balancing in an SDN context have been proposed in the literature [1–4]. Nowadays, traffic is characterized by unpredictable events and fluctuations which require *dynamic* network reconfigurations. The use of a centralized controller to coordinate a distributed architecture permits to determine and enforce solutions which are very effective; however, taking all TE decisions at the controller, in a centralized fashion, may require excessively long delays to react to network changes.

Much of current Internet traffic is *elastic* by nature, since it is handled by the Transmission Control Protocol (TCP), which permits to enforce end-to-end congestion control while, at the same time, ensuring a fair sharing for connections that experience similar network parameters (in particular, the Round Trip Time of the connection). According to a Sandvine report [5], due to its ubiquitous use in video streaming, web browsing, file transfers, communications, and other application types, TCP typically represents 85% to 90% of fixed access Internet traffic and as much as 96% of mobile Internet traffic. Furthermore (see for example [6]) it has been shown that the resource allocation that is carried out by TCP flows is well approximated by a well-known fairness paradigm, the Max-Min-Fairness (MMF) [7]: each TCP connection uses, in average, the whole available capacity until it is bottlenecked at some link, on which it equally shares its capacity with the other concurrent flows. This holds also for "TCP-like" protocols that are gaining momentum, like QUIC, for example, which exhibit a similar behavior.

Previous works in this context ([6]) consider as known/given the number of connections in the network and further assume a single-path routing. Bottleneck routing games have been introduced in [8], in a MMF context, trying to move traffic to the less utilized parts of the network, avoiding congestion by minimizing the utilization of the most used links. However, due to the increasing number of services and the users mobility, the number of active flows changes over time, and, in addition, a single-path routing does not allow nodes to make optimal resource decisions based on the real-time network congestion and flows intensities.

For these reasons, in this paper we present a mathematical model (*Adaptive Multi-Path*, AMP) and two effective heuristics that perform routing of elastic (TCP-like) demands in general network topologies, adopting a hybrid framework based on a Stackelberg (Leader-Follower) approach, as in [9], that we call "semi-distributed": the controller (Leader) provides a set of paths to each active origin-destination pair, adapting its cardinality based on heterogeneous demands intensities (we will show numerically that such approach provides consistent advantages if compared to providing a fixed number of paths, or a unique path). We also set an upper bound to the maximum cardinality of paths to simplify switches' route management and reduce the usage of the Ternary Content Addressable (TCAM) memory [10]. The switches (Followers) then choose among these paths and adapt their local forwarding behavior to the real-time congestion information obtained in a distributed fashion. This distinguishes our contribution from previous works that separately consider SDN-based load balancing

([2–4]), elastic traffic ([6–8]), and semi-distributed resource allocation ([11, 12]) approaches. In addition, as a further novel and important contribution, we explicitly consider and model a stochastic scenario where both the number and size of flows change over time, which leads to radically different choices compared to those that typically apply to traditional static scenarios.

We numerically assess the performance of our proposed model and heuristics with a custom-built python-based network simulator under stochastic traffic arrivals to show how our multi-path routing enables switches to choose their best current path based on real-time congestion information. Moreover, we observe that our designed heuristics are able to find very good solutions in a short time (few tens of seconds) also in relatively large network scenarios, which is a necessary requirement for an effective operation in a real environment.

The paper is structured as follows: Section II discusses the system model. Section III and Section IV present, respectively, a mathematical formulation of our problem and two effective heuristics. Numerical results are discussed in Section V for general and realistic network scenarios. Finally, conclusions and directions for future research are outlined in Section VI.

II. SYSTEM MODEL

We focus on an SDN scenario where programmable switches implement the network user-plane and are equipped with processing capabilities (e.g., P4 or OpenState-compliant switches). The semi-distributed approach we present aims at leveraging the advantages of both centralized and distributed control-plane solutions. In particular, it can quickly balance path loads to follow real-time traffic changes, based on live congestion information and independent decisions made by each switch, and, at the same time, rely on a global view, provided by the SDN controller, to steer switches' behavior towards an optimal performance.

The framework we propose is shown in Fig. 1. The network supports several elastic demands from a source switch (src) to a destination switch (dst). Each demand can be routed via multiple paths and split among them on a per-flowlet basis [2].

In this context, three are the main TE choices we have to make to cope with traffic dynamics: (i) the number of paths to be assigned to each source-destination demand, (ii) the routing paths, namely the links crossed by each path, and (iii) the demand split ratio across multiple paths. Our framework assigns the first two to the controller, while the split ratio is decided by each source switch. Given the paths provided by the controller, each switch maintains a local congestion measure for each of them. The congestion can be periodically estimated by using probe packets [3] or piggybacking techniques [2] measuring the available capacity. At the arrival of the first packet of a flowlet, the switch will route the entire flowlet through the path experiencing the lowest congestion. If another flowlet starts while the previous is still active, it will be routed again through the path exhibiting the lowest congestion, considering the contribution of ongoing flowlets.
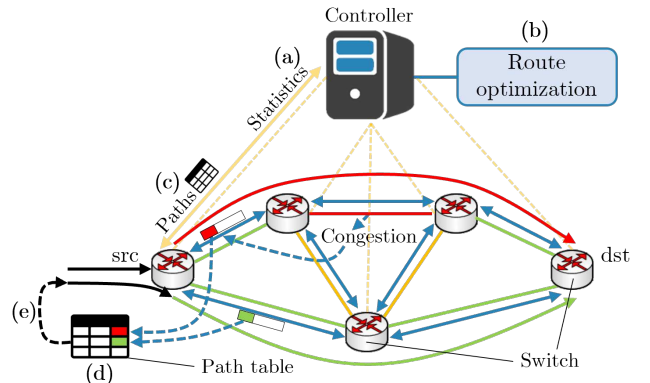

Fig. 1: Proposed semi-distributed framework.

Arguably, the network performance provided by the switches' adaptive split will strongly depend on the available paths and their number. In addition, the design of the best paths must consider the Max-Min-Fairness resource allocation originated by the interaction of elastic flows. Although TCP flows may deviate from the MMF model when in the presence of different Round-Trip-Times (RTT), other commonly-implemented mechanisms to limit the congestion, like Random Early Detection (RED), allow to practically limit this effect [13]. Computing optimal routing paths considering a different resource allocation model (f.i., considering fixed traffic demands or arbitrary sharing mechanisms) would clearly lead to a sub-optimal, often remarkably worse, performance when the network is subject to elastic traffic, which is dominant in the Internet [5].

Each switch chooses the best path according to its own congestion measurements. Therefore, from a network perspective, we can describe them as competitive players that try to selfishly minimize the congestion seen by respective flows. The interactions among these players routing flowlets through available paths will eventually converge to a stable split condition [8], namely the game-theoretical concept of equilibrium, where no player has an incentive to unilaterally deviate. A routing model that arbitrarily assigns split ratios to switches would not be representing the real resource allocation in stable conditions, so it would produce inefficient routing.

In light of the above aspects, the framework we propose works as follows (see Fig. 1): (a) Switches monitor the flowlet arrival intensities of the demands for which they are the source node and periodically send this information statistics to the SDN controller. (b) The controller determines the number and the routing paths to assign to each demand according to the received traffic intensities, considering the aspects related to MMF and equilibrium resource allocation. It can use the model in Sec. III or the much faster heuristic algorithms proposed in Sec. IV. (c) Path entries are then sent to each switch. (d) Each switch keeps measuring the congestion perceived on assigned paths and stores this information in a path table. (e) When a new flowlet of a demand arrives, the switch looks up the path table and selects the routing entry for the demand with the current minimum congestion.

This combination of real-time congestion-based switch reactions and periodical measurement-based routing updates from the controller provides an efficient trade-off between *fast* and *optimal* network reconfiguration to follow traffic dynamics.

III. PROBLEM FORMULATION

Traditional MMF-based capacity allocations [6] rely on the assumption that the number of concurrent flows on each link is known, and they equally share the link capacity. However, real communication networks behave like a *stochastic* scenario, where elastic traffic consists of connections (flowlets) that arrive at arbitrary instants and last until the connection is closed, e.g., the file transfer is completed. We can characterize this traffic assuming Poisson arrivals for each demand with rate Λ^d and file size drawn from a general distribution of mean σ. However, the actual number of flowlets along a link depends not only on flowlets' arrival rates, but also on their service times, which depend, in turn, on the resources allocated to those flowlets and the file size distribution. Since the resource allocation depends, in turn, on the number of per-link concurrent flowlets, modeling this interaction under MMF assumptions is in general intractable, and, even for a given routing, it is not possible to obtain a closed form solution for the distribution of the number of concurrent flowlets through a link [14]. For this reason, rather than focusing on the exact bandwidth allocation for each flow, our approach aims at finding a set of paths that adapts to the network topology and to the heterogeneous intensity distribution of the origin-destination pairs. Therefore, instead of considering the exact number of flowlets for each demand, we consider its *intensity*, that is the arrival rate Λ^d. Therefore, what we call *throughput* (bandwidth) of a demand x is in fact an approximation of it. However, this does not harm the goal of the model, which is to provide good routing paths to allow switches to perform the best possible throughput-oriented choices. Consistently with this definition, we introduce the concept of *intensity-normalized bandwidth* of a demand as its total available bandwidth normalized by its traffic intensity.

The routing design model we propose, named *Adaptive Multi-Path (AMP)*, considers as inputs a network graph $G(\mathcal{V}, \mathcal{A})$, $\mathcal{V}$ and $\mathcal{A}$ being the set of nodes and arcs in the graph, a set of demands $\mathcal{D}$ with their traffic intensities $\{\Lambda^d, d \in \mathcal{D} : \Lambda^d \geq 1\}$, the set of all available paths $\mathcal{P}^d$ from the source to the destination of each demand d, and a maximum number of allowed paths N. It provides as output a variable number of routing paths for each demand such that the network throughput is maximized.

The model includes four set of variables: $x^{d,p}$ and $y^{d,p}$ are, respectively, a real variable expressing the total throughput (bandwidth) for demand d along the path $p \in \mathcal{P}^d$, and a binary variable indicating if demand d uses path $p \in \mathcal{P}^d$. Then, binary variable $b_{i,j}^{d,p}$ indicates if arc (i,j) is a bottleneck for demand d along path p. Finally, the real variable $z_{i,j}$ expresses the largest intensity-normalized bandwidth among those of the demands traversing arc (i,j).

The objective function of the AMP model is:

$$\Omega = \max \sum_{d \in \mathcal{D}} w^d \sum_{p \in \mathcal{P}^d} x^{d,p} \tag{1}$$

which maximizes the weighted sum of the total bandwidth available to each demand. To enforce the model to give higher weights to demands with higher intensity, we set $w^d = \Lambda^d$. For the sake of clarity, we divide the description of model constraints in the following three sets:

a) Multi-path constraints: These constraints define the multi-path behavior:

$$\sum_{p \in \mathcal{P}^d} y^{d,p} \leq N \qquad \forall d \in \mathcal{D} \tag{2}$$

$$\sum_{p \in \mathcal{P}^d} y^{d,p} \geq 1 \qquad \forall d \in \mathcal{D} \tag{3}$$

$$y^{d,p} \leq \frac{x^{d,p}}{M_1} \qquad \forall d \in \mathcal{D}, p \in \mathcal{P}^d \tag{4}$$

$$y^{d,p} \geq \frac{x^{d,p}}{M_2} \qquad \forall d \in \mathcal{D}, p \in \mathcal{P}^d \tag{5}$$

Constraints (2) and (3) enforce, respectively, the maximum and minimum number of paths per demand. Due to constraints (4) and (5), the binary variable $y^{d,p}$ takes value 1 if and only if path $p \in \mathcal{P}^d$ is used by demand d. The values of $M_1 = \min_{(i,j) \in \mathcal{A}} \{c_{i,j}\} / (N \cdot \sum_{d \in \mathcal{D}} \Lambda^d)$ and $M_2 = \max_{(i,j) \in \mathcal{A}} \{c_{i,j}\}$ are, respectively, the lower and upper bound[1].

b) Max-Min Fairness constraints: In a scenario characterized by multiple simultaneous flowlets and multiple paths for each demand, MMF conditions in Sec. II must be properly adapted. Since each flowlet behaves as an independent elastic traffic connection, when multiple flowlets of the same demand share the same bottleneck, they equally share the total available bandwidth. We refer to this fraction of bandwidth as the *per-flowlet bandwidth*. Similarly, if more demands share the same bottleneck, all their flowlets equally share its available bandwidth, experiencing the same per-flowlet bandwidth.

The bandwidth allocation among multiple flowlets of different demands routed through multiple paths deserves further comments, which are related to the way each switch independently routes incoming flowlets according to the congestion along its available paths. Recalling some notions of [8], it is said to be a (Nash) Equilibrium a routing profile set for which every switch considers its chosen strategy to be the best under the given choices of other nodes. However, each switch is selfish and tries to improve its own bottleneck, which is the performance of the worst link under the given strategy (paths) it is using. A stable condition is the one where each switch balances the per-flowlet bandwidth of a demand over its available paths so that all its bottlenecks have the same value. Indeed, if no balance is achieved, the switch would have incentive to deviate some of demand's flowlets to the less congested path, i.e., the one with the largest per-flowlet bandwidth. Therefore, the per-flowlet bandwidth of a demand

---

[1]The upper bound considers the case where the demand is the only competitor in its bottleneck link, which has the maximum capacity in the network. The lower bound is achieved when the N paths of all demands are routed through the same arc $(i,j) \in \mathcal{A}$, which has the smallest capacity in the network. Note that the lower bound holds in case $x^{d,p} > 0$.

is the same through all the bottleneck links of its paths. Extending the above reasoning from flowlets of the same demand passing through different bottleneck links to flowlets of different demands passing through the same bottleneck link, we can conclude that the per-flowlet bandwidth (and hence the intensity-normalized bandwidth) of a demand must be greater than or equal to the one of any other demand routed via the same bottleneck link. We enforce this behavior with the following constraints, where $L_{i,j}^{d,p}$ is a parameter which takes value 1 if arc (i,j) belongs to the path p of demand d.

$$\sum_{(i,j)\in\mathcal{A}} b_{i,j}^{d,p}\cdot L_{i,j}^{d,p} \geq y^{d,p} \qquad \forall d\in\mathcal{D}, p\in\mathcal{P}^d \quad (6)$$

$$\sum_{d\in\mathcal{D}}\sum_{p\in\mathcal{P}^d} x^{d,p}\cdot L_{i,j}^{d,p} \leq c_{i,j} \qquad \forall (i,j)\in\mathcal{A} \quad (7)$$

$\forall (i,j)\in\mathcal{A}, d\in\mathcal{D}, p\in\mathcal{P}^d :$

$$\sum_{d'\in\mathcal{D}}\sum_{p'\in\mathcal{P}^d} x^{d',p'}\cdot L_{i,j}^{d',p'} \geq c_{i,j}\cdot b_{i,j}^{d,p} \quad (8)$$

$$z_{i,j} \geq \frac{\sum_{p'} x^{d,p'}}{\Lambda^d}\cdot L_{i,j}^{d,p} - M_3(1-y^{d,p}) \quad (9)$$

$$\frac{\sum_{p'} x^{d,p'}}{\Lambda^d}\cdot L_{i,j}^{d,p} \geq z_{i,j} - M_3(1-b_{i,j}^{d,p}) \quad (10)$$

Constraints (6) ensure that there is at least one bottleneck arc for each demand, constraints (7) are link capacity constraints, constraints (8) guarantee that bottleneck arcs are saturated, and constraints (9) assign to the variable $z_{i,j}$ the value of the largest intensity-normalized bandwidth on the arc $(i,j)\in A$. Finally, constraints (10) impose that all demands for which (i,j) is bottleneck must have an intensity-normalized bandwidth at least as large as the one of any other demand routed through it. The value of $M_3 = (\max_{(i,j)\in A}\{c_{i,j}\}\cdot N)/\min_{d\in D}\{\Lambda^d\}$ is the upper bound of the intensity-normalized bandwidth achieved in the best-case of a demand routed with no competitors over N paths whose bottlenecks are links of maximum capacity.

c) Other inequalities: The last set of constraints are additional cutting planes that help the MILP solution process by improving the quality of the linear relaxation:

$$x^{d,p} \leq M_1 \qquad \forall d\in\mathcal{D}, p\in\mathcal{P}^d \quad (11)$$

$$\sum_{(i,j)\in\mathcal{A}} b_{i,j}^{d,p}\cdot L_{i,j}^{d,p} \leq \frac{x^{d,p}}{M_1} \qquad \forall d\in\mathcal{D}, p\in\mathcal{P}^d \quad (12)$$

$$b_{i,j}^{d,p} \leq L_{i,j}^{d,p}\cdot y^{d,p} \qquad \forall d\in\mathcal{D}, p\in\mathcal{P}^d \quad (13)$$

Constraints (11) and (12) are upper bounds for the bandwidth available to demand d on path p and for the $b_{i,j}^{d,p}$ variable, respectively. Constraints (13) impose that arc (i,j) can be bottleneck for $d\in\mathcal{D}$ only if path $p\in\mathcal{P}^d$ is indeed used.

IV. HEURISTICS

To allow our model to follow the network dynamics and let it achieve a prompt (re-)configuration of traffic routing, we propose in this section two heuristic routing optimization algorithms. The first one, *heur-AMP*, overcomes the complexity of the original AMP model by working on a restricted set of elementary paths. The second one, *stochastic-AMP*, is designed for a dynamic environment, providing more degrees of freedom to avoid congestion issues during live operations.

A. Heur-AMP

Since the number of available paths for each demand grows exponentially with the size of the network, we can remarkably reduce the complexity of the AMP problem by considering a reduced set of "good" candidate paths. To obtain such subset, we rely on the continuous relaxation of some of the integer variables in the AMP formulation. Integer variables are $y^{d,p}$, specifying if demand d uses path p, and $b_{i,j}^{d,p}$, indicating the bottleneck link of each path p. Namely, we solve the AMP model by relaxing only variables $b_{i,j}^{d,p}$ (*relaxed-AMP*), and then consider every path p for demand d such that $y^{d,p} = 1$. This has the twofold advantage of providing hints about few paths per-demand with a potentially high performance and obtaining such an indication in short computation times.

Since the set of paths selected by relaxing variables $b_{i,j}^{d,p}$ may cause feasibility issues in the constraints of the original AMP formulation, we cannot guarantee that the number of per-demand paths resulting from relaxed-AMP provides a feasible solution for the original AMP model. Therefore, we solve the original AMP model again considering now the reduced set of paths indicated by relaxed-AMP.

B. Stochastic-AMP

AMP model and heur-AMP heuristic implement a routing path optimization that provides high throughput in static conditions, i.e., considering average traffic intensities Λ^d, as indicated by the values of their objective functions. However, the real network behavior is characterized by a stochastic scenario with random flowlet arrivals, each with a random duration. This may considerably and unpredictably [14] shift the network working point from the one that was optimal in average conditions. In this case, even if it is essential to give relevance to more intense demands, it is also crucial to set a bound to the minimum offered bandwidth to guarantee the best possible worst-case. Since the worst per-flowlet bandwidth is the one experienced in the most congested link of the network, maximizing the minimum per-flowlet bandwidth will implicitly reduce the congestion in all network bottlenecks.

Therefore, in the first step of this new heuristic approach, we modify heur-AMP by changing its objective function into:

$$\max\min \nu^d = \sum_{p\in\mathcal{P}^d} x^{d,p}/\Lambda^d \quad (14)$$

which we linearize with standard additional constraints [7].

However, as for any max-min optimization, it can happen that network resources are not fully exploited. To preserve an optimal worst-case throughput and target at a good overall resource allocation, we consider as a second step of the approach another modified version of heur-AMP, where we add lower-bound constraints to the minimum per-flowlet bandwidth:

$$\sum_{p\in\mathcal{P}^d} x^{d,p} \geq \Lambda^d \nu^d \qquad d\in\mathcal{D} \quad (15)$$

and we plug in a different objective function. Indeed, we argue that multiple per-demand paths to forward flowlets can better cope with traffic dynamics in a stochastic scenario. Indeed, they provide better opportunities to network switches to avoid

TABLE I: MILP results

			AMP		fixed-AMP	
			2	3	2	3
$|\mathcal{V}|$	$|\mathcal{A}|$	$|\mathcal{D}|$	Ω'	Ω'	Ω'	Ω'
5	7	10	1.10	1.11	1.03	0.81
6	8	10	1.09	1.10	1.01	**0.78**
6	9	10	1.11	1.11	1.07	0.98
8	12	10	1.21	**1.22**	1.17	1.13
8	12	15	**1.09**	1.11	1.04	0.99
avg			1.12	1.13	**1.06**	0.94

TABLE II: Heuristics and MILP results in the *abilene* and *polska* network scenarios.

		AMP				heur-AMP						stochastic-AMP			
		2		3		1		2		3		2		3	
net	$|\mathcal{D}|$	Ω'	γ'	Ω'	γ'	Ω'	γ'	Ω'	γ'	Ω'	γ'	Ω'	γ'	Ω'	γ'
abilene	10	1.15	0.98	1.15	0.93	1.00	1.04	1.13	1.11	1.12	1.15	1.11	1.62	1.12	1.62
	20	1.06	1.03	1.06	0.94	0.96	1.31	1.01	1.32	1.02	1.36	1.01	1.73	1.03	1.73
	30	1.04	0.97	1.04	1.08	0.96	1.51	0.98	1.67	0.98	1.67	0.94	2.91	0.94	2.91
	40	1.02	0.93	1.01	0.94	0.95	1.24	0.94	1.46	0.94	1.46	0.94	1.96	0.95	1.96
	avg	1.07	0.98	1.07	0.97	0.97	1.28	1.02	1.39	1.02	1.41	1.00	2.06	1.01	2.06
polska	10	1.29	0.85	**1.38**	0.73	0.99	0.93	1.23	0.99	1.31	0.88	1.21	1.44	1.26	1.49
	20	1.06	1.03	1.06	0.94	0.96	1.31	1.01	1.32	1.02	1.36	1.01	1.73	1.03	1.73
	30	1.10	0.73	1.13	0.80	0.99	1.15	1.06	1.17	1.05	1.15	1.02	2.03	1.04	2.03
	40	1.04	0.84	1.04	0.88	0.95	1.39	0.95	1.23	0.95	1.25	0.91	3.53	0.91	**3.53**
	avg	1.12	0.86	1.15	0.84	0.97	1.20	1.06	1.18	1.08	1.16	1.04	2.18	1.06	2.20

congestion. Hence, to incentivize the solution to provide more paths for each demand, we modify the objective function into:

$$\max \sum_{d \in \mathcal{D}} w^d \sum_{p \in \mathcal{P}^d} x^{d,p} + \alpha \frac{\sum_{d \in \mathcal{D}, p \in \mathcal{P}^d} y^{d,p}}{|\mathcal{D}|}. \quad (16)$$

The normalization with demand cardinality $|\mathcal{D}|$ ensures the second term to be at most N, so we tune parameter α such that the second term assumes a value about 0.1% of the first one.

Stochastic-AMP then consists in a *first step*, in which we use heur-AMP with expression (14), while, in the *second step*, we use again heur-AMP with (15) and (16). Notice that, as shown in Sec. V, applying heur-AMP for both the first and second step remarkably reduces the solution time, thus leading to an overall time-efficient heuristic.

V. NUMERICAL RESULTS

To evaluate the performance of our models and heuristics, in this Section we compare them in terms of objective function value, computation time, and throughput statistics obtained with our network simulator.

a) Network instances: We consider three network topologies from the SNDlib library [15]: *abilene* ($|\mathcal{V}| = 12$, $|\mathcal{A}| = 30$), *polska* ($|\mathcal{V}| = 12$, $|\mathcal{A}| = 36$) and *nobel-germany* ($|\mathcal{V}| = 17$, $|\mathcal{A}| = 52$). We assign a random capacity to each arc choosing the values between 2, 2.4, and 8 Gbps. We consider randomly generated sets of origin-destination pairs with different demands cardinality $|\mathcal{D}|$ in the range [10,70]. We consider for each instance an intensity Λ^d computed by sampling from a uniform distribution over [1,10].

We solve the proposed models and heuristics using the CPLEX solver (ver. 12.8) on a PC with two quad-core Intel Xeon E5620 processors and 90 GBytes of RAM, imposing a time limit of 1 hour for each run.

b) MILP results: Table I shows the results of the AMP and *fixed-AMP* models, where in *fixed-AMP* we impose exactly N paths per demand by setting equality in constraint (2). The results of each row are averaged over 10 instances. For each instance we randomize the link capacities, the combination of origin-destination pairs, and their intensities. Columns are grouped into model type and number of paths $N \in [1, 2, 3]$. For each of them, we report the value of Ω', which is the objective function (1) normalized w.r.t. the value obtained with AMP for $N = 1$. Since the *fixed-AMP* variant has proved computationally hard to solve, we use different subsets in terms of nodes and arcs of *nobel-germany* only for this case.

We can see that by introducing multiple paths (for $N \geq 2$) the performance improves, from 9% up to 22%. However, while the AMP model exploits multi-path routing as an opportunity to obtain performance gains, for fixed-AMP this can be a harmful limitation, being forced to split traffic over exactly N paths for each demand. Indeed, even if we can observe an improvement of 6% on average with $N = 2$, a decrease occurs with $N = 3$, which can be as large as 22%.

Table II reports the results of the AMP model and two proposed heuristics, *heur-AMP* and *stochastic-AMP*. Column γ' is the smallest intensity-normalized capacity assigned to a demand, normalized again w.r.t. AMP for $N = 1$. A multi-path solution exhibits the best performance when the network is highly meshed and not too congested, as more room is available to apply smart TE decisions. Indeed, a maximum improvement of 38% (in bold in Table II) is observed for the polska network with $|\mathcal{D}| = 10$.

The optimality gap between the heuristics and AMP over the same instances never goes beyond 9% on average. However, we have a remarkable computation time reduction with the heuristics as results are available in few seconds: up to 14s for heur-AMP and 60s for stochastic-AMP, on average, rather than almost 1 hour (time limit) for AMP model. Furthermore, γ' shows that stochastic-AMP can provide a higher worst-case bandwidth, up to 353% (in bold in Table II) better than AMP with 1 path. This aspect of stochastic-AMP will become more evident when we analyze the simulation results hereafter.

c) Dynamic Scenario: We developed a python-based network simulator that generates flowlets over time and computes their bandwidth allocation at each time instant according to the TCP waterfilling algorithm [16]. For each couple of nodes, flowlet arrivals are generated proportionally to traffic intensity Λ^d and routed according to the paths provided by the controller and local switch decisions. Flowlets' file sizes are drawn from an exponential distribution. The arrival intensity and file-size mean are tuned in such a way that network stability is preserved for all examined models. When a flowlet arrives, the source switch can perform the path selection in two ways:

- *Predefined static* (denoted by letter P in Figure 2). This path selection method models the case in which switches have no forwarding decision capabilities. The switch sends a number of flowlets on each path that is proportional to the (fixed) ratio $x^{d,p}/\sum_p x^{d,p}$, enforced by the controller on the basis of AMP model or heuristics.
- *Oracle dynamic* (denoted by O). We assume the switch is perfectly aware, in real-time, of the congestion level of

Fig. 2: Average normalized throughput. Heuristics simulation results.
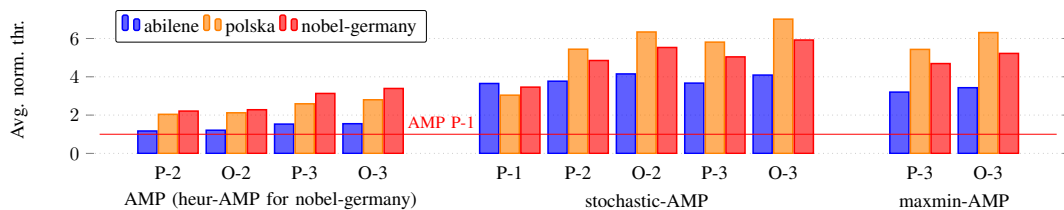
available paths and sends each incoming flowlet through the path with the currently minimum congestion. This provides an upper bound to the performance achievable by monitoring solutions, where congestion is estimated with packet probing or piggybacking techniques.

Figure 2 shows the average throughput achieved by these two path selection methods, computed as the inverse of the average completion time, and normalized by the one obtained by AMP for $N = 1$. Results correspond to the average values among all the instances listed in Table II. We further consider the *nobel-germany* network, for which the demands cardinality is extended to a maximum of $|\mathcal{D}| = 70$ to match its size. Since for this large topology we could not find a solution with AMP model in reasonable time, we run *heur-AMP* heuristic. Reported results are grouped by the model type, the path selection method (P and O) and the value of N ($\in [1, 2, 3]$).

We can observe the following: *1)* Independently of the model type/heuristic and the network topology, the model performs better when we increase the number of available paths N (up to 7 times), and the O path selection method outperforms the P method (as expected) as real-time congestion information permits to better follow traffic variations and avoid congestion. *2)* Stochastic-AMP always outperforms AMP in terms of average normalized throughput. Indeed, it is interesting to see that stochastic-AMP with a single path routing solution shows better performance (up to 238%) than AMP model with 3 available paths under both path selection methods (AMP-P-3 and AMP-O-3). *3)* Finally, we analyze the solution obtained by halting stochastic-AMP at its first step (*maxmin-AMP*), thus focusing only on (14). It shows better performance than AMP, but the gap with stochastic-AMP is evident, as it omits the second step aiming to improve network resource utilization.

VI. CONCLUSION

In this paper we proposed a mathematical model, as well as two effective heuristics, that route elastic demands in general network topologies. We envisioned a semi-distributed approach to deal with path congestion, where the SDN controller provides a limited set of paths to each active switch, adapting their number to congestion conditions and estimated demands intensities. The switches choose within this set of paths, adapting their forwarding behavior to the real-time congestion information.

Numerical results, obtained in different network scenarios with an increasing number of nodes and demands, show that our proposed heuristics can find very good routing solutions

taking only few tens of seconds as computing time, while solving exact models becomes rapidly infeasible with increasing instance sizes. Furthermore, they are able to accurately track dynamic changes in the arriving flows that naturally occur in real networks. We also provided, through oracle-dynamic path selection, an upper bound to the performance achievable by any on-line monitoring solution, where congestion is estimated with packet probing or piggybacking methods.

Future work will include a more detailed study of the game's features, including the quality of the equilibria reached by the followers (switches) in a fully distributed way, as well as the convergence properties of the procedures necessary to reach such equilibria in real-life scenarios.

REFERENCES

[1] C. E. Hopps, "Analysis of an equal-cost multi-path algorithm," *RFC 2992*, pp. 1–8, 2000.

[2] M. Alizadeh, T. Edsall, S. Dharmapurikar, R. Vaidyanathan, K. Chu, A. Fingerhut, V. T. Lam, F. Matus, R. Pan, N. Yadav, and G. Varghese, "Conga: Distributed congestion-aware load balancing for datacenters," *SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 4, pp. 503–514, 2014.

[3] N. Katta, M. Hira, C. Kim, A. Sivaraman, and J. Rexford, "HULA: Scalable Load Balancing Using Programmable Data Planes," in *Proc. of the Symposium on SDN Research*, Santa Clara, CA, USA, 2016.

[4] W. Wang, W. He, and J. Su, "Enhancing the effectiveness of traffic engineering in hybrid SDN," in *2017 IEEE International Conference on Communications (ICC)*, 2017, pp. 1–6.

[5] "TCP Optimization: Opportunities, KPIs, and Considerations," *Sandvine White Paper*, 2016.

[6] S. Coniglio, L. Gianoli, E. Amaldi, and A. Capone, "Elastic Traffic Engineering Subject to a Fair Bandwidth Allocation via Bilevel Programming," *IEEE/ACM Trans. Netw.*, vol. 28(6), pp. 2407–2420, 2020.

[7] D. Nace and M. Pioro, "Max-min fairness and its applications to routing and load-balancing in communication networks: A tutorial," *IEEE Communications Surveys Tutorials*, vol. 10, no. 4, pp. 5–17, 2008.

[8] R. Banner and A. Orda, "Bottleneck routing games in communication networks," *IEEE JSAC*, vol. 25, no. 6, pp. 1173–1179, 2007.

[9] Y. Korilis, A. Lazar, and A. Orda, "Achieving network optima using stackelberg routing strategies," *IEEE/ACM Transactions on Networking*, vol. 5, no. 1, pp. 161–173, 1997.

[10] Y. Guo, H. Luo, Z. Wang, X. Yin, and J. Wu, "Routing optimization with path cardinality constraints in a hybrid SDN," *Computer Communications*, vol. 165, pp. 112–121, 2020.

[11] Y. Chen, Z. Li, B. Yang, K. Nai, and K. Li, "A stackelberg game approach to multiple resources allocation and pricing in mobile edge computing," *Future Gen. Comp. Sys.*, vol. 108, pp. 273–287, 2020.

[12] L. Zhong, M. Li, Y. Cao, and T. Jiang, "Stable user association and resource allocation based on stackelberg game in backhaul-constrained hetnets," *IEEE TVT*, vol. 68, no. 10, pp. 10 239–251, 2019.

[13] E. Altman, C. Barakat, E. Laborde, P. Brown, and D. Collange, "Fairness analysis of TCP/IP," in *Proceedings of the 39th IEEE conference on decision and control*, IEEE, vol. 1, 2000, pp. 61–66.

[14] T. Bonald and A. Proutière, "A queueing analysis of data networks," in *Queueing Networks*. Boston, MA: Springer US, 2011, pp. 729–765.

[15] S. Orlowski *et al.*, "SNDlib 1.0–Survivable Network Design Library," *Netw.*, vol. 55, no. 3, pp. 276–286, May 2010.

[16] D. Bertsekas and R. Gallager, *Data Networks (2nd Ed.)* USA: Prentice-Hall, Inc., 1992.