Auction-based Resource Allocation in OpenFlow Multi-tenant Networks

Salvatore D'Oro, Laura Galluccio, Panayotis Mertikopoulos, Giacomo Morabito and Sergio Palazzo

***Abstract*—In this paper, we investigate the allocation of network resources (such as Flow Table entries and bandwidth) in *multi-tenant* Software-Defined Networks (SDNs) that are managed by a FlowVisor. This resource allocation problem is modeled as an auction where the FlowVisor acts as the *auctioneer* and the network Controllers act as the *bidders*. The problem is analyzed by means of *non-cooperative* game theory, and it is shown that the auction admits a unique Nash Equilibrium (NE) under suitable conditions. Furthermore, a novel distributed learning procedure is provided that allows each Controller to reach the game's unique NE in a few iterations by exploiting only locally available information. An implementation in OpenFlow-compliant SDNs is also proposed in a way that exploits native procedures already offered by OpenFlow. Finally, simulation results show that the proposed auction-based resource management scheme leads to significant improvements in network performance (for instance, achieving gains of up to 5× reduction in transmission delays).**

***Index Terms*—Software Defined Networking, OpenFlow, auction, resource allocation.**

I. Introduction

Multitenancy is a concept referring to the possibility for several customers to share certain resources such as physical network elements and links and use them as they were the sole users of those resources. In the last years, increased attention has been paid to the application of the multitenancy concept in the networking domain. Even if most efforts have been focused on the application of multitenancy concepts to the datacenter domain [1, 2], in many other scenarios such concepts can be exploited beneficially. Two relevant scenarios, for example, are that of *virtual network operators* in which several companies sell network access services using the network infrastructure owned by a third party [3], and that of *virtualized network functions* where hardware network elements, such as switches and firewalls, are substituted by software middleboxes that execute networking procedures on one or more virtual machines (VMs) deployed on the cloud [4].

In multi-tenant scenarios, the owner of the network infrastructure has two major needs:

- Maximize the quality of service experienced by its customers, that is, render its customers *satisfied*;
- Maximize its revenue.

In order to meet both of them, efficient resource management mechanisms should be considered. Recently, thanks to their capability to provide dynamic network management [5–8], software-defined networks (SDNs) have attracted much interest in the literature as a reliable framework to provide support for multitenancy [9–13] and energy-efficiency [14, 15]. In SDNs, control and data planes are decoupled. Network control and management are centralized and implemented in software, while the data/forwarding plane consists of an underlying physical network composed by several SDN-compliant switches and links. Although there are several ways to implement SDNs, in this paper we consider OpenFlow [16] as the most popular implementation thereof. In fact, as we show later, OpenFlow specifications already provide procedures to support dynamic resource allocation in multi-tenant networks.

In a SDN, multiple networks can coexist; thus, to properly manage their interactions, OpenFlow provides a FlowVisor [17], which is a high-level controller that is designed to act as a proxy between the physical network and multiple customers. By exploiting FlowVisor protocols, OpenFlow fully supports the multitenancy principle. In fact, FlowVisor and OpenFlow together allow the network owner to divide the network resources into *slices* and give full control of each slice to one customer that, to this purpose, runs a software program referred to as *Controller*. OpenFlow and FlowVisor ensure isolation between slices and therefore, each Controller can use its share of the network resources as if it were the sole controller doing so. In the following, we will identify the network owner with the FlowVisor and its customers with the corresponding Controllers.

In this paper we address the case where the FlowVisor reserves a portion of the network resources and divides it among the Controllers that compete with each other to obtain such resources. Our problem formulation is general and can be applied to several resource allocation problems in SDN scenarios. However, for illustration purposes we focus on two relevant resource allocation problems where each Controller competes to obtain either additional space in OpenFlow routing tables, i.e., Flow Tables, to store its routing policies, or bandwidth on a certain network link to improve its achievable throughput. It is worth noting that both Flow Tables and bandwidth are scarse resources in many OpenFlow applications. On the one hand, Flow Tables are implemented in finite capacity memories. On the other hand, bandwidth is well-known to be limited on the network links. Accordingly, efficient assignment of such resources is of extreme importance.

To this end, in line with a large body of literature on the design of distributed resource management techniques we consider *auctions* as the allocation instrument.

In this perspective, the FlowVisor acts as the *auctioneer* while the Controllers act as the *bidders*. Periodically the

S. D'Oro, L. Galluccio, G. Morabito and S. Palazzo are with the CNIT Research Unit at University of Catania, Italy;

P. Mertikopoulos is with the French National Center for Scientific Research (CNRS) and the Laboratoire d'Informatique de Grenoble, France;

This work has been supported by the European Commission in the framework of the FP7 Network of Excellence in Wireless COMmunications NEWCOM# (Grant agreement no. 318306), and by the French National Research Agency (ANR) grant NETLEARN (contract no. ANR-13-INFR-004).s

FlowVisor starts a new auction to sell a certain amount of network resources and each Controller makes a bid. Controllers determine their bids based on their interest in the network resources, i.e. the object of the auction. The FlowVisor then assigns each Controller a portion/share of the network resources which is proportional to the submitted bid.

In this context, Controllers behave selfishly: each of them aims at maximizing a utility function which is the difference between the *benefit* from the obtained resource and the effective *cost* (monetary and otherwise) for obtaining it. Obviously,

- The Controller's *benefit* increases with the amount of the obtained share;
- The Controller's *cost*, in general, increases commensurately with the placed bid.

Given that Controllers have conflicting interests, the above scenario can be modeled using *non-cooperative game theory*. In this paper we provide such a game-theoretic formulation and show that the resulting game admits a unique Nash equilibrium (NE) for a wide class of benefit and cost functions which are specific to the SDN scenarios. Such states represent a stable operation point of the system where no Controller has an incentive to deviate –and thus disrupting the system equilibrium. Also, we provide a novel exponential learning scheme which converges to the NE and we show that such convergence occurs within a few iterations of the auction. We also provide extensive numerical results that provide significant insight into the considered scenario.

Motivated by this analysis, we provide an implementation of the auction-based resource allocation mechanism which is based on procedures and commands that are already provided by OpenFlow, and we show that the proposed mechanism leads to significant improvement in network performance. Specifically, in the Flow Table auction which is specifically addressed in SDN scenarios because of the working mechanism at the Controller, our approach increases the probability to find a matching rule in the Flow Table by a constant term equal to 0.2 in many cases. Instead, in the bandwidth auction, the proposed auction-based resource allocation mechanism reduces transmission delays up to 5 times if compared to those experienced when static resource allocation policies are considered.

The rest of this paper is organized as follows. An overview of the related work is provided in Section II. In Section III we introduce the proposed auction game in a multi-tenant OpenFlow scenario. The auction game is described and studied in Section IV, where a distributed learning scheme to reach the unique NE of the auction game is also presented. In Section V we illustrate numerical results that provide useful insights on both the auction game dynamics and the learning scheme. In Section VI a discussion on implementation issues of the proposed auction-based resource allocation scheme in OpenFlow networks is presented; also, an implementation of the auction game is provided with extensive simulation results which show the effectiveness of the proposed approach. Finally, Section VII concludes the paper. All relevant system parameters and variables are summarized in Table I.

II. Related work

The problem of resource allocation and management is crucial in any networking environment and a large body of literature exists on the subject spanning all types of networks, e.g., [18–20]. Obviously, resource allocation is a major issue in software defined networks as well [21]. In this context, the possibility to support multi-tenancy raises a new set of issues regarding resource management [22, 23]. In fact, the necessity emerges to identify a tradeoff between the needs of tenants, each interested in obtaining the largest possible amount of resources at the lowest cost, and the needs of the owner of the network infrastructure [24, 25]. This kind of problem has been stated in the context of data centers, where, however, most of the focus was in processing and storage resources rather than in networking resources [26].

In this context, several centralized approaches have been proposed in the literature to provide optimal resource allocation schemes [27–29]. However, such solutions often lead to NP-hard problems that can be implemented in real systems only if sub-optimal solutions and performance losses are tolerable [28, 29]. Also, centralized solutions require perfect information and cooperation among the centralized entity and the other agents in the network, i.e., the Controllers. Unfortunately, Controllers are unlikely to share their private information, and are more likely to act selfishly, seeking to unilaterally maximize their individual profits. Therefore, it is hard to implement centralized solutions in competitive scenarios where information is neither revealed nor shared among parties. On the contrary, distributed resource allocation mechanisms are well-known to be able to provide low-complexity, efficient and privacy-preserving solutions to a variety of resource allocation problems in telecommunication networks. Accordingly, and in line with a large body of literature on the design of distributed resource management techniques spanning most networking domains [30–33], we consider *auctions* as the allocation instrument

The idea of exploiting auction-based mechanisms to regulate resource allocation between different tenants has been proposed in the context of SDN and OpenFlow as well [34, 35]. Our work in this paper goes far beyond [34, 35] because we develop here a game-theoretic framework capturing the Controllers' selfish interactions, and we show that, under suitable conditions, the resulting game admits a unique Nash equilibrium. Furthermore, a distributed learning procedure is provided that allows to reach such equilibrium in a small number of iterations.

III. System Model

Our model consists of a physical network of switches operating under an OpenFlow framework that enables software-defined networking. The behavior of OpenFlow switches is completely determined by the contents of a Flow Table (or several), whose entries specify flows and how packets of these flows must be treated. More specifically, each entry contains three sections:

- *Rules:* this is used to match incoming packets to existing flows. In fact, this entry contains a set of header values against which incoming packets are matched to determine which flow they belong to.

- *Actions:* this entry specifies how packets belonging to the flow should be treated, e.g., where a packet should be forwarded, whether it should be dropped, whether and how they should be modified, etc.
- *Stats:* this entry contains statistical information which can be used by the elements in the control plane to tune their policies.

The FlowVisor manages a set of OpenFlow-compliant network *slices* (logical sub-networks composed of switches and links that are isolated from one another, even if they share the same underlying physical resources), each of which is assigned to a single Controller who is able to fully control decisions based on its own requirements and forwarding policies. The Controllers enforce their policies by inserting appropriate entries in the Flow Table. In *multi-tenant* SDNs, multiple slices and their corresponding controllers coexist. To manage the interactions between several Controllers, OpenFlow provides a *FlowVisor* that acts as a proxy between the physical network and the co-existing Controllers. Furthermore, the FlowVisor is able to control both the network topology and the network status by monitoring parameters such as delay, network load and link utilization.

In this context, we consider an auction-based resource management scheme for SDNs where a portion of the available resources is leased to interested Controllers $\mathcal{N} = \{1, \dots, N\}$. Let $\hat{R}$ denote the total amount of the shared resource. Without loss of generality, we assume that $\hat{R}$ is finite. i.e, $\hat{R} < +\infty$. However, note that we do not make any assumption on the type of the shared resource. In fact, in our context network resources could be identified with bandwidth (like in traditional networks) as well as Flow Table entries. It follows that our model is able to capture a wide variety of scenarios. Furthermore, let $R \equiv \xi\hat{R}$ denote the fraction of the total resource that the FlowVisor decides to lease via the auction mechanism to the Controllers that want to improve the capability of their slices.

In the auction, the FlowVisor acts as the auctioneer and the participating Controllers act as independent bidders that do not communicate with each other.[1] Each bidder $i \in \mathcal{N}$ submits a bid $b_i \in \mathcal{B}_i \equiv [0, B_i]$ where B_i denotes the maximum admissible bid of the i-th Controller. Note that since B_i is a function of i, our model can also capture the case in which Controllers are heterogeneous w.r.t. their economic resources. When the auction is over, the FlowVisor collects all bids and assigns to each bidder $i \in \mathcal{N}$ a fraction ϱ_i of the available resource proportional to the corresponding bid, i.e.

$$\varrho_i(b) = R \frac{b_i}{\sum_{j \in \mathcal{N}} b_j} \tag{1}$$

In Eq. (1), ϱ_i is the portion of the available resource that is assigned to Controller i as a function of the bid b_i. The allocation of said portion of available resource is temporary and resource leasing expires as soon as the auction is repeated [31] – typically in scenario-specific time intervals of fixed or variable width according to contractual agreements between the FlowVisor and Controllers which are defined and enforced by Service-Level Agreements (SLAs). To the best of our knowledge, even though this proportional allocation auction scheme has been exploited in other relevant application scenarios [36], its use in this context is novel. For continuous good allocation problems of this type, we could alternatively consider an optimal auction in the sense of Myerson [37] or a continuous second-price sealed-bid auction (Vickrey auction) for multiple continuous goods [38, 39]. The advantage of using Eq. (1) is that it is much simpler to implement, so it can be deployed in the context of SDNs with minimal computational overhead.

It is evident that each Controller may have a different interest on a particular resource depending on several factors, such as the amount of traffic that flows through a link or a switch, the resource already assigned to the Controller, the number of network users, etc. To account for this, let $\theta_i \in [0, 1]$ denote the *interest factor* of bidder i representing the benefit that it expects to receive for a unit of resource (specifically, $\theta_i = 0$ implies that the i-th bidder has no interest in the auction)[2]. On the other hand, from a benefit perspective and in tune with traditional economic assumptions, we assume that the user's benefit from a resource can be modeled as an increasing concave function $\omega_i(b)\colon \mathcal{B}_i \to \mathbb{R}$ in the submitted bid, i.e., Controllers experience diminishing returns when increasing the amount of the submitted bid[3]. Specifically, in this paper we focus on the case where the Controllers' benefit functions are affine functions of the fraction of the obtained resource, i.e., $\omega_i(b) = a_i + c_i\varrho_i$ for some suitable constants a_i and c_i. From (1), it can be verified that $\omega_i(b) = a_i + c_i\varrho_i$ is also a concave function in the submitted bid, as we have beforehand assumed.

Furthermore, when submitting a bid, each Controller faces an incurred cost (e.g., a monetary cost) which can vary from one Controller to another. The effective cost to each Controller might be different than the actual monetary value of the Controller's bid. The reasons for this are diverse: for instance, each bid could be subject to a value added tax (VAT) or other form of taxation which would increase the actual cost to the user. When the auction ends, additional resources are allocated to participating Controllers. Such resources need to be properly managed and, in general, lead to an additional cost in the network management process. Therefore, Controllers can be charged for this additional management cost. Furthermore, the same monetary amount may have a vastly different impact to Controllers with different budgets, and this impact does not have to be linear in the monetary value of each Controller's bid. For example, when Controllers have a finite budget, doubling a large bid is much costlier (in relative terms) than doubling a small bid [40]. Thus, in tune with standard economic assumptions [40], we posit in what follows that the *effective cost* to the i-th Controller is given by an increasing convex cost function $\Gamma_i\colon \mathcal{B}_i \to \mathbb{R}$, called the *cost function* of Controller i.

In view of this cost-benefit analysis, the *utility* of bidder i can be expressed as the difference between the bidder's expected

[1]For simplicity, in what follows, we will use the terms "bidder", "Controller" and "player" interchangeably.

[2]In principle, θ_i changes over time, however, we assume that auctions take place often enough so that the evolution of θ_i can be neglected. This assumption is justified by the rapid convergence to the equilibrium as discussed in the following Section V.

[3]The concavity assumption has been widely used in economics theory and can also be referred to the concept of risk-aversion behavior of rational decision-makers [40].

TABLE I
NOTATION

Variable	Description
$\mathcal{N}$	Set of Controllers
R	The amount of auctioneed resource
b_i	Bid submitted by Controller $i \in \mathcal{N}$, $b_i \in \mathcal{B}_i \equiv [0, B_i]$
ϱ_i	Fraction of the available resource obtained by i when the auction is over
$\theta_i \in [0, 1]$	Interest factor of i
$\omega_i(b), \Gamma_i(b_i), u_i(b)$	Benefit, Cost and Utility functions for i
f	Space needed to store a flow table entry
T_H, T_R, T_i	Matching time, request and forwarding delays
$P_i^H(b)$	Hit probability of i
L_i, N_i	Number of stored flow table entries and number of flows w.r.t. Controller i
α	The Zipf distribution parameter
λ	Non-negative cost parameter
γ_n	Step-size of the learning procedure

benefit and the cost to achieve it. More specifically:

$$u_i(b) = \omega_i(b) - \Gamma_i(b_i), \tag{2}$$

where $\omega_i(b)$ is the benefit function for Controller i (which also incorporates the dependency on ϱ_i), and $b = (b_1, \ldots, b_N) \in \mathcal{B} \equiv \prod_i \mathcal{B}_i$ denotes the *bid profile* of the Controllers participating in the auction.

In reality, different scenarios have different features and issues. Therefore, no unique model to represent users' benefits and costs in SDNs exists. On the contrary, the definitions of $\omega_i(b)$ and $\Gamma_i(b_i)$ in Eq. (2) depend on the considered application scenario. For this reason, in Section III-A we consider two relevant SDNs scenarios and we propose different benefit function models that are representative of a wide variety of SDNs scenarios. Finally, in Section III-B we propose several cost function models that reflect realistic costs experienced in SDN scenarios.

A. Benefit Function Models

1) Flow Table Auction: This is the case where the auctioneer, i.e., the FlowVisor, decides to sell a fraction (or all) the available space of a given flow table stored inside a specific switch. We assume that each switch stores a single flow table, and each flow table is stored inside a cache memory. It is worth noting that OpenFlow ensures isolation among different slices and allows to create non-overlapping virtual private flow tables which are stored in the switch's flow table. Therefore, even though the same flow table is shared among all Controllers that have access to the same switch, Controllers cannot add/modify flow entries which correspond to other network slices. Instead, each Controller can buy additional flow entries and add them to its own non-overlapping private virtual flow table.

To process a flow, the corresponding flow entry has to be defined in the flow table. We assume that each flow entry requires the same amount of space in the flow table. Let f denote this amount of space. Upon receiving each packet, the receiving switch first tries to find a matching rule, or matching entry, for the flow which the packet belongs to. Then, if a matching rule exists, the corresponding action is performed. Instead, if no matching entries are found, the switch asks the corresponding Controller a rule for that flow through a `PacketIn` message [41]. Then, a `FlowMod` message [41] which contains the rule and the corresponding action to be executed for all packets belonging to the same flow is sent by the Controller to the requesting switch. Finally, if the flow table has some available storage space, the requesting switch stores the new flow entry in its flow table.

It is worth noting that, according to OpenFlow and FlowVisor specifications, each switch is able to identify which slice a given packet belongs to. Therefore, even though a matching rule for a given packet does not exist in the flow table, pre-configured flow table entries are exploited to forward the `PacketIn` message to the corresponding Controller. Instead, if those pre-configured flow table entries are not installed on a given switch, in general, actions to drop the packet or forward it to a default Controller are taken.

It is important to focus on the impact of the above operations on network performance. Clearly, if a matching rule is found, the search operation introduces a small delay, say T_H. On the contrary, when no matches exist, the transmission of both `PacketIn` and `FlowMod` messages causes a requesting delay T_R that depends on the distance between the requesting switch and the corresponding Controller. For example, the requesting delay T_R can grow up to approximately 100 ms in many cases [42].

In realistic scenarios, we have that $T_R >> T_H$.[4] Also, note that cache memories, i.e., the storage space available for each flow table, are finite. Accordingly, the number of flow entries that can be stored in flow tables is limited. Thus, all Controllers compete with each other to get as much space as possible to store their own entries and reduce the expected forwarding delay:

$$T_i = T_H P_i^H + (1 - P_i^H) T_R \tag{3}$$

[4]For example, the time complexity of most search algorithms on Random Access Memories is well known to be either $\mathcal{O}(F)$ or $\mathcal{O}(\log F)$, where F is the number of entries in the flow table. Also, since $T_R >> T_H$ despite the value of F, in this paper we assume that T_H is $\mathcal{O}(F)$.

where P_i^H is the *hit probability* of the i-th Controller, i.e., the probability that a rule for the considered flow already exists in the table. It can be easily shown that minimizing the expected delay is equivalent to maximizing the hit probability P_i^H. Also, one can show that the hit probability depends on the amount ϱ_i of available resources to store the additional rules in the flow table that the Controller obtains at the end of the auction.

Accordingly, in the flow table auction scenario the benefit function $\omega_i(b)$ of the i-th Controller is given by:

$$\omega_i(b) = \theta_i P_i^H(b) \tag{4}$$

However, the hit probability for a given flow depends on the probability distribution of that flow. Therefore, we consider two relevant scenarios depending on the flows' distribution:

- *Uniform distribution:* in this case, the Controllers' flows are uniformly distributed. Accordingly, their benefit function can be written as

$$\omega_i(b) = \theta_i \frac{L_i + \varrho_i/f}{N_i} \tag{5}$$

  where L_i is the number of rules already stored by Controller i in the flow table, N_i is the number of flows of Controller i and ϱ_i is defined in Eq. (1) and represents the space in the flow table, i.e., the cache size, obtained at the end of the auction by Controller i. Accordingly, ϱ_i/f represents the number of additional flow entries (and thus, rules) that can be stored in the flow table by the i-th Controller normalized by the cache size. Clearly, this is the worst case scenario where all flows are expected to arrive with the same probability and the uncertainty in predicting flow arrivals is maximized.
- *Zipf distribution:* in this case, we assume that the probability that a packet arriving at a certain switch belongs to a given flow is distributed according to a Zipf's law [43]. Consequently, there are some *popular* rules that are expected to be satisfied with higher probability than *unpopular* ones. Under the finite cache and infinite request stream assumptions, it has been show in [44] that the hit probability is asymptotically $\approx \mathcal{O}(R_{\text{cache}}^{1-\alpha})$, where $\alpha \in [0,1]$ is the Zipf distribution parameter and R_{cache} is the available storage space in the cache. Accordingly, we have $P_i^H(b) \approx \mathcal{O}(L_i + \varrho_i/f)^{1-\alpha}$. Finally, by exploiting the Bernoulli's inequality for binomial series, the benefit function $\omega_i(b)$ can be approximated and linearized as follows:

$$\omega_i(b) = \theta_i L_i^{-\alpha}(L_i + (1-\alpha)\varrho_i/f) \tag{6}$$

It is worth noting that the number of flow table entries which can be assigned to each Controller is expressed as an integer value. However, the value of ϱ_i/f is, in general, not an integer. Therefore, to realistically represent an OpenFlow scenario where only an integer number of table entries can be assigned to bidders, we assume that the number ϱ_i/f of entries that are assigned to each bidder at the end of the auction is rounded to the nearest integer.

2) Bandwidth Auction: The FlowVisor is able to continuously monitor bandwidth and link utilization on each physical link in the network. Therefore, to avoid inefficient resource allocation and to improve link utilization, it is possible to sell the available unused bandwidth on the network link in question.

In this scenario, each Controller seeks to maximize the amount of bandwidth achieved on the considered link at the end of the auction. Accordingly, and in line with traditional auction-based bandwidth allocation schemes, we define the following benefit function:

$$\omega_i(b) = \theta_i \varrho_i \tag{7}$$

B. Cost Function Models

When the auction mechanism is over, additional network resources are allocated to SDN Controllers. To manage such resources, additional network management procedures have to be performed by the network operator, leading to extra costs in the network management process. For example, at the end of the flow table auction new flow entries are created and inserted in the flow table. For each incoming flow the switch accesses the flow table and searches for a matching entry.

In general, flow table entries are stored in physical memories inside SDN switches, and the lookup procedure to find a matching rule introduces a cost which is expected to increase with the number of stored entries [45]. Intuitively, highly populated flow tables lead to larger search times. On the contrary, when the number of stored flow entries is small, the time needed to access them is small. Thus, since the auction increases the number of entries stored in the flow table, it also increases the cost to search and access such entries.

Similarly, the bandwidth auction also causes an increase in the amount of available bandwidth on each considered link. Thus, Controllers are likely to increase the transmission rate and the amount of data flowing through the network. As a consequence, the utilization factor of the network is expected to increase as well. Thus, to guarantee the correct operation of the network and manage this additional data traffic, additional resource management and control is required (e.g., congestion control, traffic balancing and other similar procedures).

Accordingly, we consider two different models for the cost function $\Gamma_i(b_i)$ in Eq. (2): linear cost (LC), and non-linear cost (NLC). More in detail:

- in the LC scheme we define $\Gamma_i(b_i) = \lambda b_i/B_i$;
- in the NLC scheme we consider an exponential-based cost scheme [46], i.e., $\Gamma_i(b_i) = \lambda(e^{b_i/B_i} - 1)$.

where $\lambda \geq 0$ is a non-negative cost imposed by the auctioneer to any bidder that buys some resources.

With all this in mind, we can define the following non-cooperative auction game $\mathfrak{G} = \mathfrak{G}(\mathcal{N}, \mathcal{B}, u)$:

- The set of *players* (or *bidders*) is $\mathcal{N} = \{1, \dots, N\}$.
- The *action space* of each bidder $i \in \mathcal{N}$ is $\mathcal{B}_i$.
- The players' *utility functions* $u_i \colon \mathcal{B} \equiv \prod_i \mathcal{B}_i \to \mathbb{R}$, given as in Eq. (2).

Accordingly, we will say that a bid profile $b \in \mathcal{B}$ is a NE of $\mathfrak{G}$ when no bidder has an incentive to change its bid unilaterally. In a more formal way, this can be stated as:

Definition 1. *A bid profile $b^* = (b_1^*, \dots, b_N^*) \in \mathcal{B}$ is called a Nash equilibrium (NE) of the auction game $\mathfrak{G}$ if*

$$u_i(b_i^*, b_{-i}^*) \geq u_i(b_i, b_{-i}^*) \tag{8}$$

for every bid $b_i \in \mathcal{B}_i$ of player i and for all $i \in \mathcal{N}$, where (b_i^*, b_{-i}^*) is shorthand for $b^* = (b_1^*, \dots, b_i^*, \dots, b_N^*)$.

The Nash equilibria of $\mathfrak{G}$ represent stable resource management policies where each Controller is satisfied with respect to its individual cost-benefit characteristics and with the existing resource allocation scheme. As such, the analysis in the following sections will focus on the NE of $\mathfrak{G}$ and we will show that the system admits a unique stable state which can be reached by the players in a distributed fashion, i.e., without requiring any coordination among Controllers or the transmission of private information (for example related to each Controller's bids).

It is worth noting that the considered model has been designed to deal with several SDN-specific issues, such as the benefit function model used in the Flow Table Auction. Even though the specifics of this model are inherently tied to the underlying SDN model, the proposed auction mechanism can be adapted to a general framework to deal with a wide variety of resource allocation problems in the networking domain.

IV. System Stability and Learning

In this section, we examine the existence and uniqueness of the Nash equilibrium and propose an exponential learning mechanism through which the bidders can converge to the equilibrium.

Theorem 1 (NE existence and uniqueness)**.** *The game $\mathfrak{G}$ always admits a unique NE.*

Proof: Note first that each player's payoff function u_i is concave in b_i being it defined as the difference between a concave and a convex function; thus, existence of NE follows from the general theory of [47]. To prove the uniqueness of the NE, we define the affine benefit function $\omega_i(b) = a_i + c_i\varrho_i(b)$, where a_i and c_i are two non-negative real numbers, and $\varrho_i(b)$ is defined in Eq. (1) and represents the resource obtained by Controller i at the end of the auction. Likewise, we prove uniqueness by establishing the diagonal strict concavity (DSC) condition [47], i.e.

$$\sum\nolimits_{i\in\mathcal{N}} r_i v_i(b) \cdot [b_i - b_i^*] < 0 \quad \text{for all } b \in \mathcal{B}, \tag{9}$$

for some $r_1, \dots, r_N > 0$ and for all Nash equilibria b^* of $\mathfrak{G}$, where $v_i(b)$ is the so called players' *marginal utility* that we introduce in Eq. (10). For that, by using the analysis in [48], it suffices to show that *a*) each utility function u_i is *strictly* concave in b_i and convex in b_{-i}; and *b*) the function $\sigma(b, r) = \sum_{i\in\mathcal{N}} r_i u_i(b)$ is concave in b for some $r = (r_1, \dots, r_N) \in \mathbb{R}^N$ with $r_i > 0$.

For the first condition, strict concavity is ensured by the fact that $u_i(b_i, b_{-i})$ is a sum of a strictly concave and a concave function (in b_i); convexity in b_{-i} is also straightforward. For the second condition, if $r_i = c_i^{-1}$, then we can rewrite $\sigma(b, r) = 1 + \sum_{i\in\mathcal{N}} r_i a_i - \sum_{i\in\mathcal{N}} r_i \Gamma_i(b_i)$; given that the cost functions Γ_i are convex, $\sigma(b, r)$ is concave in b. Therefore, the claim of the theorem follows from [48]. ■

Remark 1. *From Theorem 1, we have that the game $\mathfrak{G}$ with benefit and cost functions being defined as in Sections III-A and III-B admits a unique NE.*

A. Learning strategy

Theorem 1 shows that the auction initiated by the FlowVisor admits a single stable state where each Controller is unilaterally satisfied with respect to its individual assigned resource and cost-benefit characteristics. However, we still have to prove whether this stable policy can be reached in a distributed fashion. To that end, we propose below a simple learning scheme that allows the bidders to converge to game's unique NE.

The key quantity in what follows will be the players' *marginal utility functions*

$$v_i(b) = \frac{\partial u_i}{\partial b_i}, \tag{10}$$

i.e., the marginal increase (or decrease) in the utility of player i with respect to its own bid $b_i \in \mathcal{B}_i$. In particular, an easy differentiation yields:

$$v_i(b) = \theta_i R \frac{\sum_j b_j - b_i}{\left[\sum_j b_j\right]^2} - \Gamma_i'(b_i) = \theta_i \frac{\varrho_i(R - \varrho_i)}{b_i R} - \Gamma_i'(b_i), \tag{11}$$

where ϱ_i is the amount of resource obtained by player i in the auction as defined in Eq. (1). Accordingly, the marginal utility of player i can be calculated in a fully distributed way with local information only (knowledge of the amount of resource obtained and the player's own cost function Γ_i), so any learning scheme that relies only on v_i will also cope with auction privacy requirements.

According to this, in order to increase their utilities, players simply need to pursue the direction of marginal utility increase while maintaining their bids b_i at an admissible level – i.e. between 0 and B_i. To this end, we propose the *exponential learning* scheme (XL):

$$\begin{cases} z_i(n+1) &= z_i(n) + \gamma_n v_i(b(n)), \\ b_i(n+1) &= \frac{B_i}{1+\exp(-z_i(n+1))}, \end{cases} \tag{12}$$

where γ_n is a decreasing step-size sequence and $z_i(n)$ is an auxiliary variable. Importantly, this learning scheme is *reinforcing* because the update step of (XL) increases with v_i; it is also *fully distributed* and *privacy-preserving* because the marginal utilities v_i can be obtained directly from (11) without any exchange of private information among bidders. From Algorithm 1, each bidder has to update the bid b_i, measure v_i, and update their scores z_i. All the above operations have time complexity $\mathcal{O}(1)$. Therefore, since such operations have to be executed N times, the proposed learning mechanism has complexity $\mathcal{O}(N)$ per iteration, and $\mathcal{O}(1)$ per iteration and per Controller. If Z resources are put up for auction, then Z parallel auctions are executed, and the complexity of the learning mechanism is $\mathcal{O}(NZ)$ – or $\mathcal{O}(Z)$ per Controller. That is, the computational complexity of each iteration of the proposed mechanism is linear with respect to the numbers N and Z of Controllers in the network and of auctioneed resources.

The main result concerning the algorithmic implementation of (12) (cf. Algorithm 1) is that it converges to the unique Nash equilibrium of the auction game $\mathfrak{G}$. More formally, we have:

Theorem 2. *If the algorithm's step-size sequence γ_n satisfies $\sum_n \gamma_n = +\infty$ and $\sum_n \gamma_n^2 < +\infty$, then Algorithm 1 converges to the unique equilibrium of the auction game $\mathfrak{G}$.*

Algorithm 1 Exponential Reinforcement Learning (XL)

```
Parameter: step-size sequence $\gamma_n$ (default: $\gamma_n = 1/n$).
Initialize: $n \leftarrow 0$; $z_i \leftarrow 0$ for all $i \in \mathcal{N}$.
Repeat
    $n \leftarrow n + 1$;
    for each bidder $i \in \mathcal{N}$ do simultaneously
        bid $b_i \leftarrow B_i \left[1 + \exp(-z_i)\right]^{-1}$;
        measure marginal utility $v_i$ from (11);
        update scores: $z_i \leftarrow z_i + \gamma_n v_i$;
    until termination criterion is reached.
```

Proof: The main steps of the proof consist in i) deriving the mean dynamics of (12) in continuous-time, ii) establishing their convergence to equilibrium using the DSC condition given in Eq. (9), iii) obtaining the corresponding discrete-time results using the theory of stochastic approximation [49]. For simplicity, we take $B_i = 1$; the general case follows by rescaling.

Note first that the continuous-time equivalent of (12) is

$$\dot{z}_i = v_i, \qquad b_i = \left[1 + \exp(-z_i)\right]^{-1}, \tag{13}$$

or, after decoupling b_i and z_i:

$$\dot{b}_i = b_i(1 - b_i)v_i. \tag{14}$$

Hence, let $r_i = 1/(\theta_i R)$, as in the proof of Theorem 1, and set

$$V(b) = \sum\nolimits_i r_i \left[b_i^* \log \frac{b_i^*}{b_i} + (1 - b_i^*) \log \frac{1 - b_i^*}{1 - b_i} \right], \tag{15}$$

where b^* is the unique NE of $\mathfrak{G}$. Some algebra then yields

$$\dot{V} = \sum\nolimits_i r_i v_i(b) \cdot \left[b_i - b_i^*\right], \tag{16}$$

so, from (9) we have $\dot{V} \le 0$ with equality iff $b = b^*$. This shows that V is a strict Lyapunov function for (13), i.e., $b(t) \to b^*$.

For the discrete-time analysis, let $V_n = V(b(n))$ where $b(n)$ is the n-th iterate of Algorithm 1. Then, writing V in terms of z as

$$V = \sum\nolimits_i r_i \left[\log(1 + e^{z_i}) - z_i b_i^*\right], \tag{17}$$

a first-order Taylor expansion yields:

$$V_{n+1} \le V_n + \gamma_n \sum\nolimits_i r_i v_i(b(n)) \cdot \left[b_i(n) - b_i^*\right] + \mathcal{O}(\gamma_n^2), \tag{18}$$

where the $\mathcal{O}(\gamma_n^2)$ remainder is uniformly bounded by $\frac{1}{2}M\gamma_n^2$ for some $M > 0$ (simply note that the marginal utilities v_i are bounded on $\mathcal{B}$). Now, if the iterates $b(n)$ stay a bounded distance away from b^*, (9) shows that we have $V_{n+1} \le V_n - \gamma_n c + \mathcal{O}(\gamma_n^2)$ for some $c > 0$. Letting $n \to \infty$ and telescoping, we obtain the contradiction $V_n \to -\infty$ (note that $V \ge 0$ by construction and $\sum_n \gamma_n^2 < \sum_n \gamma_n = +\infty$ by assumption), so $b(n)$ must come arbitrarily close to b^* infinitely often. Since $b(n)$ is a stochastic approximation of the mean dynamics (13) in the sense of [49], convergence to the unique equilibrium follows from Theorem 6.9 in [49]. ■

V. Numerical Examples

In this section, we provide some relevant numerical results that illustrate the dynamics of our auction-based resource management scheme under different scenarios and cost functions. We consider $N = 30$ Controllers whose interest factors are uniformly distributed at random in the interval $[0, 1]$ so as to consider the worst-case scenario where the entropy is maximized, and thus also the unpredictability. Also, unless explicitly stated otherwise, we set the maximum admissible bid of each Controller to $B_i = B = 1$ for all $i \in \mathcal{N}$. The results have been obtained by executing a number of simulations such that all statistical results are stated at a 95% confidence level [50]. Confidence intervals are not shown for the sake of illustration.

A. Flow Table Auction

At the beginning of the auction, we assume that each Controller is provided with $L_i = 200$ flow table entries (and thus, rules) that can be stored. Also, the Zipf distribution parameter is set by a value of $\alpha = 0.7$ as frequently employed in the literature, e.g., in [44], and we assume that each flow entry occupies a unitary cache size, i.e., $f = 1$.

In Fig. 1 we show the per-user average hit probability as a function of the amount R of resources, i.e., the cache size, sold by the auctioneer when different flow distributions and cost function models are considered. Since we have assumed $f = 1$, R is expressed in number of flow entries. When the amount of the available cache size R is high, Controllers are likely to obtain a large number of flow table entries at the end of the auction, which results in high values of the hit probability. Therefore, upon increasing the cache size, Fig. 1 shows that the hit probability increases as well. Also, if the cost parameter λ is high ($\lambda = 5$), Controllers that are not interested in the resource experience high costs and low benefits. Thus, only Controllers that are interested in the resource submit their individual bids. Accordingly, interested Controllers improve their hit probability, while non interested Controllers do not obtain any additional flow entries and their hit probability is the same as the one they had at the beginning of the auction. Therefore, the system's average hit probability is low.

On the contrary, when λ is low ($\lambda = 1$) or equal to zero, even non interested Controllers are likely to submit their bids, which allows them to share the available resources among all participating Controllers. All Controllers obtain additional space to store their flow entries. Thus, the average hit probability in the two latter cases, is higher than the one achieved in the case $\lambda = 5$. Furthermore, Fig. 1 also shows that the average hit probability in the LC case is higher than (or equal to) that achieved in the NLC case. Finally, a higher average hit probability is achieved when flows are Zipf distributed. Zipf distribution allows to prioritize flows that are expected to arrive with a higher probability. Accordingly, popular flows are likely to have a matching rule in the flow tables. On the contrary, the arrival rate of uniformly distributed flows is hard to be predicted and the only way to achieve high hit probability consists in increasing the cache size put up for auction.

Fig. 2 shows the average hit probability as a function of the ratio L_i/N_i when different cost function models are considered and $\lambda = 1$. First we note that the hit probability at the end of the auction is always higher than that obtained when no auction mechanisms are considered and the resource is statically allocated to Controllers. It follows that by putting up for auction

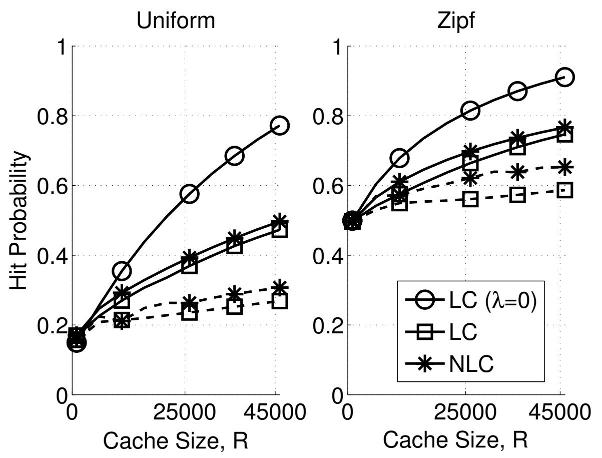

Fig. 1. Average hit probability as a function of the cache size R for different cost function models (Solid lines: $\lambda = 1$; Dashed lines: $\lambda = 5$).

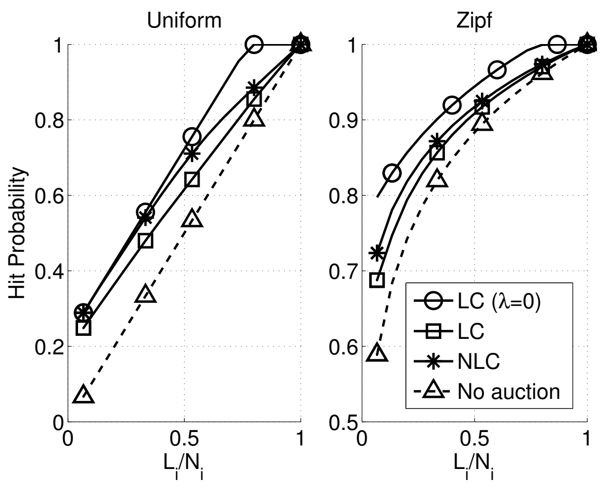

Fig. 2. Average hit probability as a function of the ratio L_i/N_i for different cost function models and $\lambda = 1$.

unallocated cache resources, it is possible to improve SDNs performance. Recall that L_i is the number of flow table entries that are already assigned at the beginning of the auction, and N_i is the number of flows for each Controller. Accordingly, by increasing the ratio L_i/N_i, the hit probability increases as well. As already shown in Fig. 1, higher average hit probabilities are achieved when flows are distributed according to a Zipf law.

Finally, to analyze on the optimality gap between our distributed solution and optimal ones, we compare the proposed approach with two centralized approaches. In more detail, we consider a centralized solution which is designed to find a social optimum, i.e., a solution that maximizes the social welfare function $\sum_{i\in\mathcal{N}} u_i(b)$; and a centralized solution that maximizes the total hit probability $\sum_{i\in\mathcal{N}} P_i^H$ of the system. Accordingly, in Fig. 3 we show the average hit probabilty as a function of the cache size R when $N = 15$ and $\lambda = 5$. It is worth noting that the proposed solution is near-optimal in almost all cases. Also, in the NLC case, it has better hit probability than that achieved at the social optimum. In fact, the socially optimal solution maximizes the overal utility of the system defined as $\sum_{i\in\mathcal{N}} u_i(b)$,

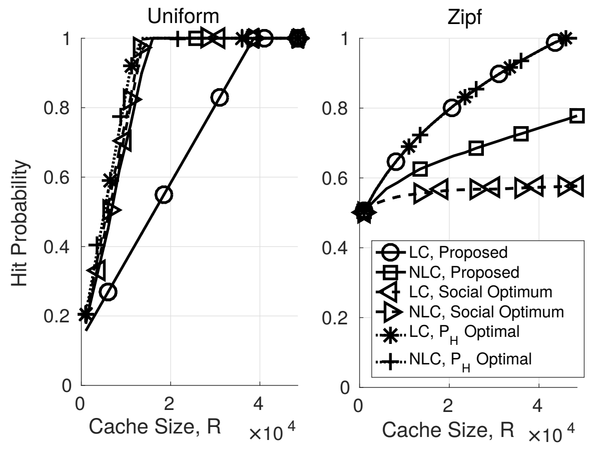

Fig. 3. Comparison between our distributed solution and two optimal centralized solutions.

TABLE II
Simulation parameter settings.

Bidder	B_i	θ_i
C_1	0.5013	0.4923
C_3	0.9976	0.9727
C_6	0.8944	0.7391
C_8	0.39	0.0319
C_{13}	0.3433	0.7111
C_{23}	0.5523	0.2815
C_{24}	0.9791	0.7311

which does not necessarily means that the hit probability is also maximized.

B. Bandwidth Auction

Fig. 4 illustrates the impact of the cost function on the Controllers' bidding strategies. Specifically, we show how the average normalized bid strategy at the NE defined as $1/N \cdot \sum_{i\in\mathcal{N}} b_i^*/B_i$ varies as a function of the cost parameter λ under different cost function models. As expected, when $\lambda = 0$ each Controller submits a bid whose value is equal to its own maximum admissible bid, i.e., $b_i = B_i$ for all $i \in \mathcal{N}$. On the other hand, when LC and NLC cost models are considered, an increase in the value of λ causes an increase in the costs experienced by each Controller. Accordingly, bids submitted by Controllers decrease as λ increases. Also, if the amount R of resources that are leased out is small (solid lines), submitted bids are smaller than those submitted when larger values of R are considered (dashed lines). This latter result is caused by the fact that when R is small, only interested Controllers submit high bids. On the contrary, a larger amount of available resources R leads non interested Controllers to submit non-zero bids to obtain a small fraction of the considered resource.

To illustrate the bidders' behavior and dynamics, in Fig. 5 we show the Controllers' bidding strategies for different cost function models. For simplicity, we only show the behavior of a subset of the participating Controllers. More specifically we consider Controllers C_i with $i \in \{1, 3, 6, 8, 13, 23, 24\}$ and we assume $\lambda = 0.2$. For illustration purposes, we assume that the maximum admissible bid B_i and the interest factor θ_i

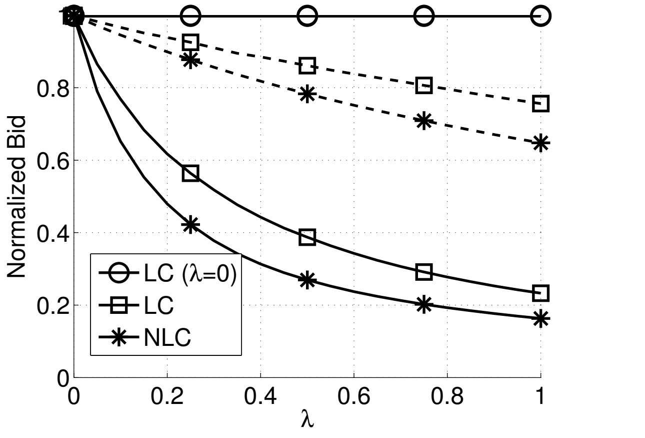

Fig. 4. Average normalized bid as a function of the cost parameter λ for different cost function models (Solid lines: $R = 10$; Dashed lines: $R = 100$).

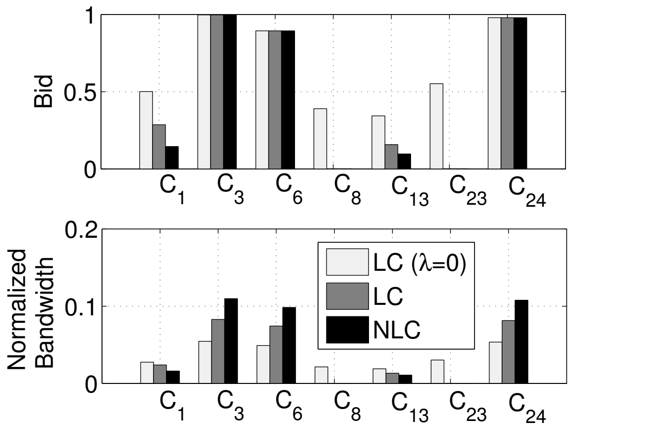

Fig. 5. Bid strategies and normalized allocated bandwidth for different cost schemes.

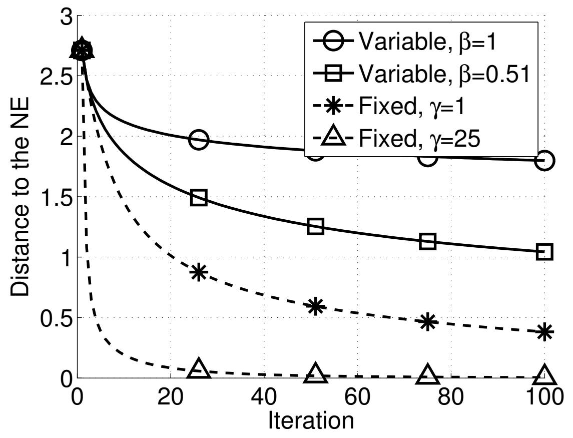

Fig. 6. Distance to the NE for different step-size schemes.

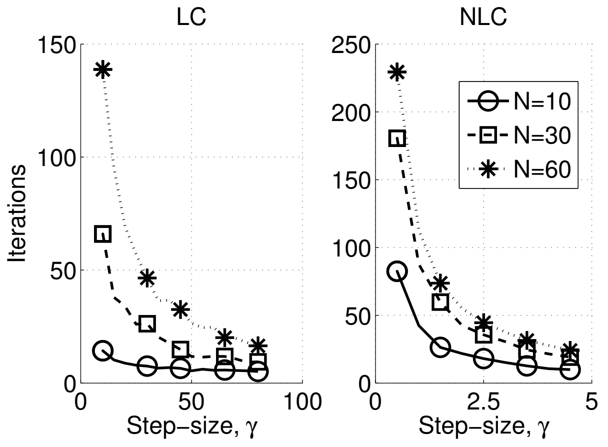

Fig. 7. Number of iterations needed to reach the NE as a function of the step-size γ_n for different network configurations and cost functions.

are uniformly distributed in the interval $[0, 1]$. The considered simulation setup is reported in Table II. Fig. 5 shows that when no cost is charged to bidders (i.e., $\lambda = 0$ case), all bidders submit non-zero bids. On the contrary, no bids are submitted by C_8 and C_{23} in the LC and NLC cases. In fact, such controllers are not interested in buying additional space in the flow table, i.e., θ_8 and θ_{23} are small. Controllers such as C_3, C_6 and C_{24} are interested in participating in the auction and have high budgets. Therefore, they submit $b_i = B_i$ independently of the actual cost model. On the contrary, even though C_1 and C_{13} are interested in buying additional bandwidth, their budget is low. Accordingly, their submitted bids decrease when they experience costs as in the LC and NLC cases.

C. Convergence Evaluation

As stated in Theorem 2, the convergence to the unique NE of Algorithm 1 is guaranteed in the case a variable step-size (e.g., $\gamma_n = 1/n^\beta$ with $\beta \in (0.5, 1]$) is used. However, in the following we show that the learning process still converges to the NE even if a fixed step-size γ_n is used. For illustrative purposes, in the following we only consider the bandwidth auction case, but the same results hold for the flow table auction case as well.

Figure 6 illustrates how fast the game converges to the unique NE by means of the proposed distributed learning procedure. Specifically, the figure shows the Euclidian distance between the actual bidding profile at each iteration and the NE in case of LC and NLC schemes. As expected, the convergence rate is slow when a variable step-size is considered. Also, by increasing the value of the parameter β, it is possible to improve the convergence rate of the proposed learning procedure under a varible step-size scheme. Otherwise, if a fixed step-size is considered, the convergence rate increases. More specifically, the higher the step-size, the faster the convergence rate is.

In Figure 7 we investigate the scalability of the proposed learning procedure as the number of bidders participating in the auction increases, and we also show how fast the learning process converges to the unique NE as a function of the fixed step-size γ_n for different cost schemes. Figure 7 shows that an increase in the step-size causes an improvement in the convergence rate of the learning process in both the LC and NLC scheme. For low values of the step-size γ_n, if a small number of bidders is considered ($N = 10$), the Algorithm 1 converges fast to the equilibrium point, while the convergence rate is slower when the number of bidders increases. By increasing the value of γ_n, the convergence rate of the learning process is fast even in highly populated auctions with a large number of participants. Therefore, an increase in the step-size helps in improving the scalability of the proposed approach; in fact, with large values of γ_n the number of iterations needed to reach the equilibrium in scarcely populated auctions ($N = 10$) and highly populated auctions ($N = 60$) are comparable and anyway small.

VI. The OpenFlow Auction in Action

In this section, we focus on a practical implementation of the auction framework in an OpenFlow-based SDN where the auctioned resource is assumed to be the available bandwidth on a given link. More in detail, Section VI-A summarizes the main procedures and operations needed to properly implement the auction framework by exploiting OpenFlow features and mechanisms; in Section VI-B we present extensive simulation results that show how auction-based bandwidth allocation can effectively improve the achievable performance in SDNs.

A. Implementation aspects

By exploiting its software-based management, OpenFlow can be considered as the enabling technology for a variety of different services such as auction-based bandwidth allocation. In fact, OpenFlow specifications already provide an important tool, namely *network slicing*. The slicing feature allows the FlowVisor to create several independent virtual networks lying on the same physical network. Therefore, by exploiting such mechanism it is possible to perform several bandwidth allocation operations.

In particular, OpenFlow specifications provide the `fvctl add-slice [options] <slicename> <controller-url> <admin-email>` command that can be easily exploited to create/modify slices. Specifically, it is used at the high-level by the FlowVisor during the start-up phase to create a new slice whose name is `<slicename>`, and assign it to a given Controller at a specific URL (i.e., `<controller-url>` could be `tcp:hostname:port`). It is also possible to specify some additional options and the email address of the Controller.

In order to limit the maximum bandwidth that each Controller can use, we exploit the OpenFlow `Rate Limiter` [41]. `Rate Limiter` is a software element in the switch that continuously monitors and measures the rate of packets on a given link. When the rate of the traffic belonging to the slices of a given Controller exceeds the maximum threshold, `Rate Limiter` executes a `drop` action on those packets that are not allowed to flow through the network.

In the following, we show the basic operations performed by both the FlowVisor and the Controllers before, during and after the auction. The communication link between each Controller and the FlowVisor can exploit either TCP or UDP protocols. For the sake of simplicity, let us focus on the simple case when the FlowVisor monitors only a single link connecting two different OpenFlow-compliant switches. Let us denote as l the monitored link. In our model we assume that $R = \xi\hat{R}$, where $\xi \in [0, 1]$ is the fraction of the available bandwidth that the FlowVisor is willing to sell[5]. As discussed in Section V, the FlowVisor also fixes a cost λ that each Controller has to pay.

In Figure 8 we provide a flowchart of the relevant operations executed by the both the FlowVisor and the i-th Controller during the auction.

A `notifyAuction()` message is sent to each Controller by indicating the link l, the amount of bandwidth R to be sold and the cost λ. Each Controller evaluates its interest θ_i in buying some additional bandwidth on the link l, determines its bid according to Algorithm 1, and sends a `sendBid()` message including the bid b_i to the FlowVisor.

By simply using a timeout, the FlowVisor is able to collect the bidding vector b containing all the bids submitted by the interested Controllers and modify the bandwidth allocation policy by modifying the `Rate Limiter`'s settings on each switch connected to the corresponding link. Specifically, the FlowVisor assigns an additional bandwidth ϱ_i to each Controller according to Eq. (1).

Hence, each Controller re-executes the learning algorithm, the bidding profile is updated and sent to the FlowVisor until the convergence criterion is reached and a `notifyAuctionEnd()` message is sent by the FlowVisor to the Controllers.

When the auction is over, each Controller that has successfully submitted a bid, sends the payment notification that depends on the cost parameter λ.

B. Simulation Results

In order to evaluate the improvement in performance that can be achieved by exploiting the proposed auction-based bandwidth allocation mechanism, we have run extensive simulations over an OpenFlow-based SDN framework.

The underlying physical network and the different slices are illustrated in Figure 9. We consider a 1970's ARPANET-like topology in which nine OpenFlow switches are deployed and provide several users with the access to the core network. We further assume that there is a FlowVisor managing five different slices each of them assigned to one of the five Controllers.

To simulate a real auction, we let the network evolve and perform OpenFlow procedures. The FlowVisor continuously monitors the utilization of the network. Thus, as described in Section VI-A, if there is some available bandwidth on the link l, the auction is notified to all Controllers.

For the sake of simplicity, in our simulations we assume that all bidders have the same maximum admissible bid, i.e., $B_1 = B_2 = \cdots = B_N$, and that the auction involves the link between switches S_4 and S_8 only[6] which is shared among four of the five slices[7].

To evaluate the impact of the interest factor on the network performance, in the following we consider two illustrative use

[5]For example, the FlowVisor could be interested in keeping unused some bandwidth to provide additional best-effort services or avoid congestion caused by a sudden increase in the traffic rate on the considered link.

[6]However, note that the proposed approach also works in the general case where Controllers have different values of B_i and any of the links in the physical network is involved in the auction.

[7]Note that OpenFlow guarantees the independence between different slices, thus several flows can coexist on the same link without interfering with each other.

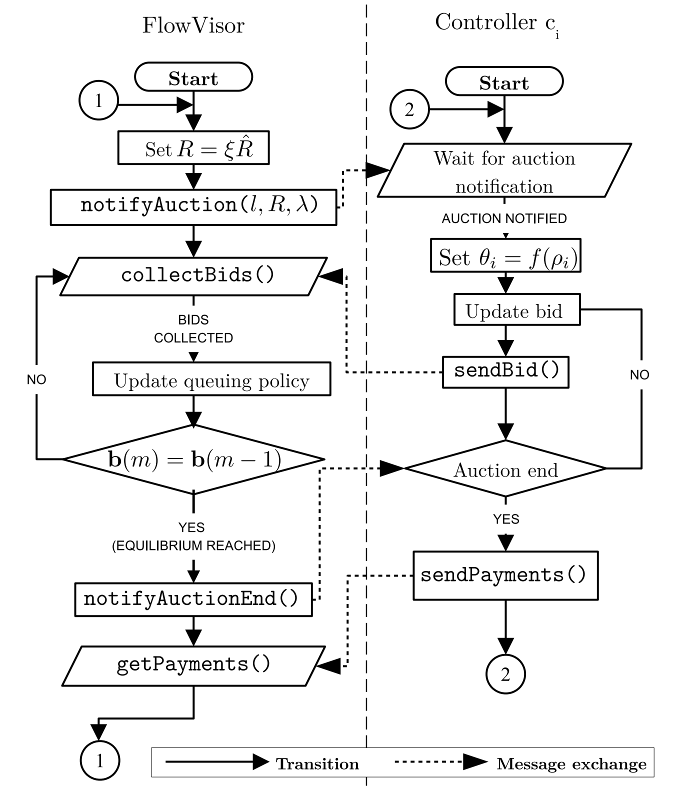

Fig. 8. Flowchart of the operations performed by the FlowVisor and the generic Controller C_i.

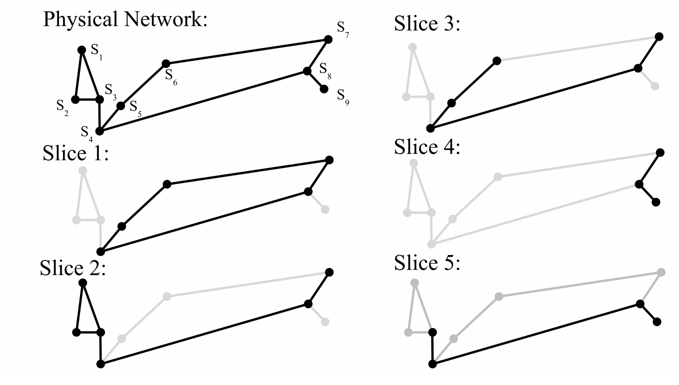

Fig. 9. Underlying physical network and logical slices. Black circles and lines indicate the switches and links belonging to each slice; gray lines indicate the network links and switches that are not part of a slice.

cases. Specifically, we focus on the case where each Controller generates its interest factor θ_i according to two different schemes:

- Usage-based Scheme (XS): where θ_i only depends on the usage factor ρ_i of the link related to the i-th bidder, i.e., $\theta_i^{XS} = \rho_i$;
- Delay-based Scheme (DS): where θ_i is a function of the expected reduction in the transmission delay caused by the aquisition of some additional bandwidth, i.e., $\theta_i^{DS} = \beta R / \left[\tilde{R}_i^2 (1-\rho_i)^2\right]$, where $\tilde{R}_i$ is the bandwidth that is already allocated to a Controller before the beginning of the auction and β is a positive real valued number[8].

TABLE III
Simulation parameter settings.

Bidder	$\tilde{R}_i$	ρ_i	θ_i^{DS}	θ_i^{XS}
C_1	3.4 Mbits/s	0.29	0.29	$8 \cdot 10^{-6}$
C_2	800 Kbits/s	0.99	0.99	0.92
C_3	1.2 Mbits/s	0.89	0.89	0.003
C_4	0 Mbits/s	0	0	0
C_5	2.6 Mbits/s	0.51	0.51	$3 \cdot 10^{-6}$

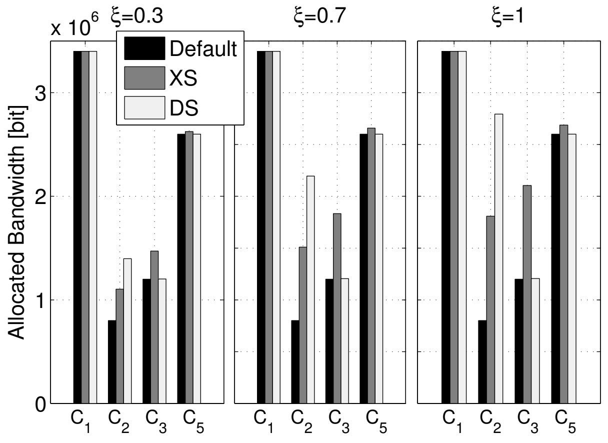

Fig. 10. Allocated bandwidth at the end of the auction as a function of the parameter ξ for different interest factor schemes.

To model realistic traffic patterns, we consider that each Controller manages several LANs that generate random traffic such as web browsing, file transfer, email and video streaming. However, without losing in generality, we can assume that the traffic flowing through each slice is similar in average.

In Figure 10 we show how the bandwidth is allocated to each Controller before the start ("Default") and after the end of the auction as a function of the parameter ξ for different interest factor schemes[9]. In the XS scheme, each bidder has a high interest in the link; therefore, all Controllers are willing to participate in the auction to buy some additional bandwidth. Differently, as shown in Table III, in the DS scheme only bidders with the lowest available bandwidth are characterized by high interest. In fact, when the available bandwidth is low, such as for Controller C_2, buying additional bandwidth is much more fruitful in terms of delay reduction than when the available bandwidth is already high. Consequently, as shown in Figure 10, only the Controllers that experience high delay participate in the auction to obtain additional bandwidth. Note that Controller C_4 does not participate in the auction as it has no interest in the considered link.

[8]The expression of θ_i in the DS scheme has been obtained as the derivative of the delay with respect to an increase of the available bandwidth assuming that the link can be modeled as an M/M/1.

[9]Note that confidence intervals are not shown for the sake of readability as the allocated bandwidth exhibits really small fluctuation around the average value.

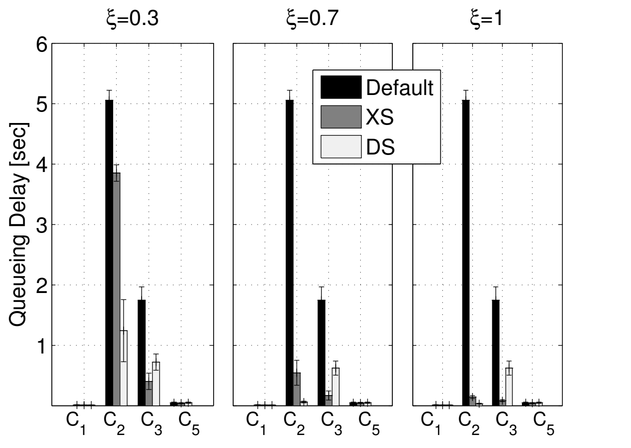

Fig. 11. Expected queuing delay as a function of the parameter ξ for different interest factor schemes.

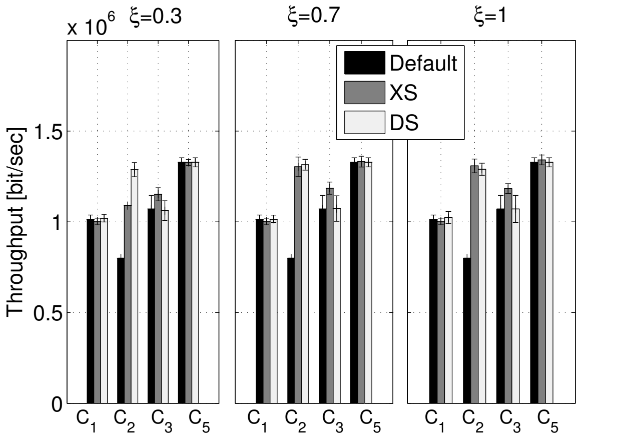

Fig. 12. Throughput at the end of the auction as a function of the parameter ξ for different interest factor schemes.

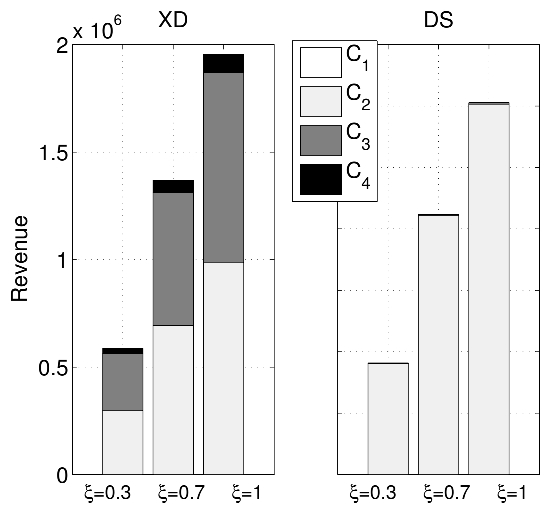

Fig. 13. Revenue of the FlowVisor at the end of the auction as a function of the parameter ξ for different interest factor schemes.

Figure 11 shows the comparison between the experienced queuing delay before the start and after the end of the auction as a function of the fraction of the available bandwidth that the auctioneer decides to sell for the two different interest-factor schemes. As expected, by increasing the fraction of the available bandwidth that each Controller can buy, the experienced transmission delay decreases. Actually, the most interesting result is that the reduction in the experienced queuing delay for both C_2 and C_3 is significant even when only a small fraction (e.g. $\xi = 0.3$) of the available bandwidth is sold. Furthermore, if the interest factor of each Controller is generated according to a DS scheme, only Controllers that are expected to considerably reduce queuing delay participate in the auction. Therefore, from Figure 11 it stems that the DS scheme is especially suitable to delay-sensitive traffic.

The improvement in the achievable throughput is shown in Figure 12. As expected, Controllers which obtain more additional bandwidth on link l are those that experience higher performance improvement. As an example, both C_2 and C_3 acquire a considerable amount of bandwidth and thus the achievable throughput, after the auction, is higher as compared to the one achieved before the start of the auction. Otherwise, Controllers that obtain less bandwidth do not experience any considerable improvement as the queuing policy remains almost the same.

Finally, the revenue of the auctioneer, calculated as $\sum_{i \in \mathcal{N}} \lambda b_i$, is illustrated in Figure 13 as a function of different interest factor schemes. It is shown that in the XS scheme, as all Controllers are more or less interested in the considered link l, they participate in the auction and the submitted bids are high, thus the revenue obtained by the FlowVisor is high as well. On the other hand, if a DS scheme is considered, only Controllers that are expected to improve the queuing delay participate in the auction, the amount of received bids is low and thus, the total revenue of the FlowVisor is also low. Note that, even in this case, the outcome of the auction is still a win-win solution as the FlowVisor receives a revenue anyway, while Controllers are able to improve their performance in terms of queuing delay and achievable throughput. Such a result shows that in an auction both the auctioneer and the bidders gain something. Thus, auction-based bandwidth allocation in OpenFlow multitenant networks can actually be regarded as a new challenging market where each player is enabled to improve its own utility.

VII. Conclusions

In this paper, we have analysed an auction-based scheme for the allocation of network resources in multi-tenant Software Defined Networks. In particular, it is assumed that a FlowVisor manages the resource allocation mechanism by executing an auction of fractions of network resources. The bidders are the Controllers who compete to gain additional network resources at a cost which is related to their interest in getting a particular network resource. The resulting scenario is modeled and studied as a non cooperative auction game, and uniqueness and convergence to the Nash equilibrium are investigated. Furthermore, a learning procedure that allows the Controllers to converge to this equilibrium in a distributed way is introduced and its convergence to the Nash equilibrium is demonstrated. Also, it is discussed how the proposed auction mechanism can be supported by using procedures already offered by OpenFlow. Simulation results show that the proposed auction-based resource management scheme allows to obtain significant performance improvement.

References

[1] B. P. Rimal, E. Choi, and I. Lumb, "A taxonomy and survey of cloud computing systems," in *Proceedings of the Fifth International Joint Conference on INC, IMS and IDC*, Aug. 2009.

[2] J. Mudigonda, P. Yalagandula, J. Mogul, B. Stiekes, and Y. Pouffary, "NetLord: A scalable multi-tenant network architecture for virtualized datacenters," in *Proceedings of the ACM SIGCOMM 2011 Conference*, Aug. 2011.

[3] C. S. Mower, M. A. Palmer, and S. C. Mayhew, "Networking as a service: delivering network services using remote appliances controlled via a hosted, multi-tenant management system," Jan. 1 2013, US Patent 8,347,355.

[4] R. Yu, G. Xue, V. Kilari, and X. Zhang, "Network function virtualization in the multi-tenant cloud," *IEEE Network*, vol. 29, no. 3, May 2015.

[5] S. Shenker, M. Casado, T. Koponen, and N. McKeown, "The future of networking, and the past of protocols," *Open Networking Summit*, 2011.

[6] H. Jin, D. Pan, J. Liu, and N. Pissinou, "OpenFlow-based flow-level bandwidth provisioning for CICQ switches," *IEEE Transactions on Computers*, vol. 62, no. 9, Sep. 2013.

[7] N. Handigol, S. Seetharaman, M. Flajslik, N. McKeown, and R. Johari, "Plug-n-Serve: Load-balancing web traffic using OpenFlow," *ACM SIGCOMM Demo*, Aug. 2009.

[8] D. Tuncer, M. Charalambides, S. Clayman, and G. Pavlou, "Adaptive resource management and control in software defined networks," *Network and Service Management, IEEE Transactions on*, vol. 12, no. 1, pp. 18–33, March 2015.

[9] L. Sun, K. Suzuki, C. Yasunobu, Y. Hatano, and H. Shimonishi, "A network management solution based on OpenFlow towards new challenges of multitenant data center," in *Proceedings of the 9th Asia-Pacific Symposium on Information and Telecommunication Technologies (APSITT)*, Nov. 2012.

[10] A. TaheriMonfared and C. Rong, "Multi-tenant network monitoring based on software defined networking," in *On the Move to Meaningful Internet Systems: OTM 2013 Conferences*, ser. Lecture Notes in Computer Science, vol. 8185, 2013.

[11] Y. Shin, S. Kang, J. Kwak, B. Lee, and S. Hyang, "The study on configuration of multi-tenant networks in SDN controller," in *Proceedings of the 16th International Conference on Advanced Communication Technology (ICACT)*, Feb. 2014.

[12] Z. Bozakov and P. Papadimitriou, "AutoSlice: Automated and scalable slicing for software-defined networks," in *Proceedings of the 2012 ACM CoNEXT Student Workshop*, Dec. 2012.

[13] M. Banikazemi, D. Olshefski, A. Shaikh, J. Tracey, and G. Wang, "Meridian: an SDN platform for cloud network services," *IEEE Communications Magazine*, vol. 51, no. 2, 2013.

[14] R. Bolla, R. Bruschi, F. Davoli, and C. Lombardo, "Fine-grained energy-efficient consolidation in sdn networks and devices," *Network and Service Management, IEEE Transactions on*, vol. 12, no. 2, pp. 132–145, June 2015.

[15] G. Faraci and G. Schembra, "An analytical model to design and manage a green sdn/nfv cpe node," *Network and Service Management, IEEE Transactions on*, vol. 12, no. 3, pp. 435–450, Sept 2015.

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

[17] R. Sherwood, G. Gibb, K.-K. Yap, G. Appenzeller, M. Casado, N. McKeown, and G. Parulkar, "FlowVisor: A network virtualization layer," *OpenFlow Switch Consortium, Tech. Rep*, 2009.

[18] Z. Dziong, *ATM Network Resource Management*, 1997.

[19] I. Akyildiz, D. Goodman, and L. Kleinrock, "Mobility and resource management in next generation wireless systems," *Selected Areas in Communications, IEEE Journal on*, vol. 19, no. 10, pp. 1825–1831, October 2001.

[20] W. Li, F. C. Delicato, P. F. Pires, Y. C. Lee, A. Y. Zomaya, C. Miceli, and L. Pirmez, "Efficient allocation of resources in multiple heterogeneous wireless sensor networks," *Parallel and Distributed Computing, Journal of*, vol. 74, no. 1, pp. 1775–1788, January 2014.

[21] I. Akyildiz, A. Lee, P. Wang, M. Luo, and W. Chou, "Research challenges for traffic engineering in software defined networks," *Network, IEEE*, vol. 30, no. 3, pp. 52–58, May-June 2016.

[22] M. Casado, T. Koponen, R. Ramanathan, and S. Shenker, "Virtualizing the network forwarding plane," in *Proceedings of ACM PRESTO 2010*. ACM, 2010.

[23] P. Skldstrm and K. Yedavalli, "Network virtualization and resource allocation in openflow-based wide area networks," in *Proceedings of IEEE ICC 2012*. IEEE, 2012.

[24] G. Schaffrath, A. Feldmann, A. Wundsam, C. Werle, R. Bless, M. Kind, L. Mathy, P. Papadimitriou, A. Greenhald, and O. Maennel, "Network virtualization architecture: proposal and initial prototype," in *Proceedings of ACM VISA 2009*. ACM, 2009.

[25] A. Fischer, J. F. Botero, M. T. Beck, H. de Meer, and X. Hesselbach, "Virtual network embedding: A survey," *Communications Surveys & Tutorials, IEEE*, vol. 15, no. 4, Fourth Quarter 2013.

[26] F. Bari, R. Boutaba, R. Esteves, L. Zambenedetti Granville, M. Podlesny, G. Rabbani, Q. Zhang, and M. F. Zhani, "Data center network virtualization: A survey," *Communications Surveys & Tutorials, IEEE*, vol. 15, no. 2, Second Quarter 2012.

[27] D. Shue, M. J. Freedman, and A. Shaikh, "Performance isolation and fairness for multi-tenant cloud storage," in *Symposium on Operating Systems Design and Implementation (OSDI 12)*, vol. 12, 2012, pp. 349–362.

[28] H. Ballani, P. Costa, T. Karagiannis, and A. Rowstron, "Towards predictable datacenter networks," in *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 4. ACM, 2011, pp. 242–253.

[29] A. Rai, R. Bhagwan, and S. Guha, "Generalized resource allocation for the cloud," in *Proceedings of the Third ACM Symposium on Cloud Computing*. ACM, 2012, p. 15.

[30] J. Huang, R. A. Berry, and M. L. Honig, "Auction-based spectrum sharing," *Mobile Networks and Applications*, vol. 11, no. 3, 2006.

[31] M. P. Wellman, W. E. Walsh, P. R. Wurman, and J. K. MacKie-Mason, "Auction protocols for decentralized scheduling," *Games and Economic Behavior*, vol. 35, no. 1, 2001.

[32] F. Martignon, S. Paris, I. Filippini, L. Chen, and A. Capone, "Efficient and truthful bandwidth allocation in wireless mesh community networks," *IEEE/ACM Transactions on Networking*, vol. 23, no. 1, Feb. 2015.

[33] P. Maillé and B. Tuffin, "Pricing the internet with multibid auctions," *IEEE/ACM Transactions on Networking*, vol. 14, no. 5, 2006.

[34] T. V. Ganesh. (2012, Jan.) Towards an auction-based internet. *Available at:* https://gigadom.wordpress.com/2011/12/26/towards-an-auction-based-internet.

[35] ——. (2013, Feb.) A method for optimal bandwidth usage by auctioning available bandwidth using the OpenFlow protocol. *Available at:* https://gigadom.wordpress.com/2013/02/05/a-method-for-optimal-bandwidth-usage-by-auctioning-available-bandwidth-using-the-openflow-protocol.

[36] A. Sahasrabudhe and K. Kar, "Bandwidth allocation games under budget and access constraints," in *Information Sciences and Systems, 2008. CISS 2008. 42nd Annual Conference on*. IEEE, 2008, pp. 761–769.

[37] R. B. Myerson, "Optimal auction design," *Mathematics of operations research*, vol. 6, no. 1, pp. 58–73, 1981.

[38] P. Cramton, Y. Shoham, and R. Steinberg, "Combinatorial auctions," 2006.

[39] R. Jain and J. Walrand, "An efficient mechanism for network bandwidth auction," in *Network Operations and Management Symposium Workshops, 2008. NOMS Workshops 2008. IEEE*. IEEE, 2008, pp. 227–234.

[40] A. Mas-Colell, M. Whinston, and J. Green, *Microeconomic Theory*. Oxford University Press, 1995.

[41] OpenFlow switch specification version 1.3.1. *Available at:* https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.3.1.pdf.

[42] K. Phemius and M. Bouet, "Openflow: Why latency does matter," in *Proceedings of the 2013 IFIP/IEEE International Symposium on Integrated Network Management*, May 2013.

[43] G. K. Zipf, "Human behavior and the principle of least effort." Addison-Wesley, 1949.

[44] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web caching and Zipf-like distributions: Evidence and implications," in *Proceedings of IEEE INFOCOM*, Mar. 1999.

[45] A. Lazaris, D. Tahara, X. Huang, E. Li, A. Voellmy, Y. R. Yang, and M. Yu, "Tango: Simplifying SDN control with automatic switch property inference, abstraction, and optimization," in *Proceedings of the 10th ACM International Conference on emerging Networking Experiments and Technologies*, Dec. 2014.

[46] S. D'Oro, P. Mertikopoulos, A. L. Moustakas, and S. Palazzo, "Adaptive

transmit policies for cost-efficient power allocation in multi-carrier systems," in *Proceedings of the 12th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt)*, 2014.

[47] J. B. Rosen, "Existence and uniqueness of equilibrium points for concave n-person games," *Econometrica: Journal of the Econometric Society*, 1965.

[48] C. Goodman, "Note on existence and uniqueness of equilibrium points for concave n-person games," *Econometrica: Journal of the Econometric Society*, 1980.

[49] M. Benaïm, "Dynamics of stochastic approximation algorithms," in *Seminaire de probabilites XXXIII*. Springer, 1999.

[50] J. Rice, *Mathematical statistics and data analysis*, 2006.

Salvatore D'Oro (S'12) received the B.S. degree in Computer Engineering and the M.S. degree in Telecommunications Engineering degree both at the University of Catania. In 2013 and 2015, he was a Visiting Researcher at Université Paris-Sud 11, Paris, France and at Ohio State University, Ohio, USA. He is currently a PhD Student at the University of Catania and his main interests are next generation communication networks, optimization, security and game theory. In 2013, he served on the Technical Program Committee (TPC) of the 20th European Wireless Conference (EW2014).

Laura Galluccio Laura Galluccio (M02) received the Laurea Degree in electrical engineering in 2001 and the Ph.D. degree in electrical, computer, and telecommunications engineering in 2005 from the University of Catania, Italy. Since 2002 she was with the Italian National Consortium of Telecommunications (CNIT), working as a Research Fellow in the FIRB VICOM and NoE SATNEX projects. Since 2010, she has been an Assistant Professor with the University of Catania. In 2005 she was Visiting Scholar with the COMET Group, Columbia University, New York. Her research interests include unconventional communication networks, software defined networks, and network performance analysis. She serves in the editorial boards of Elsevier Ad Hoc Networks and Wiley Wireless Communications and Mobile Computing journals.

Panayotis Mertikopoulos (M'11) graduated valedictorian from the Physics Department of the University of Athens in 2003. He obtained the M.Sc. and M.Phil. degrees in Mathematics from Brown University, USA, in 2005 and 2006 respectively, and his Ph.D. degree from the University of Athens in November 2010. During 2010–2011, he held a postdoctoral fellowship at École Polytechnique, Paris, France. Since 2011, he has been with the French National Center for Scientific Research (CNRS) and the Laboratoire d'Informatique de Grenoble. PM received the best paper award at NETGCOOP 2012 and was an Embeirikeion Foundation Fellow between 2003 and 2007. He has also served as TPC co-chair for WiOpt '14, general co-chair of AlgoGT '13, publications chair for WiOpt '13 and ValueTools '12, and is on the TPC of numerous conferences on game theory and wireless networks. Since 2014, the PI is also a member of the steering committee of the Optimization and Decision Theory branch of the Société de Mathématiques Appliquées et Industrielles (SMAI). His main interests lie in optimization, game theory, dynamical systems, and their applications to telecommunication networks.

Giacomo Morabito received the laurea degree and the PhD in Electrical, Computer and Telecommunications Engineering from the Istituto di Informatica e Telecomunicazioni, University of Catania in 1996 and 2000, respectively. From November 1999 to April 2001, he was with the Broadband and Wireless Networking Laboratory of the Georgia Institute of Technology as a Research Engineer. Since April 2001 he is with the Dipartimento di Ingegneria Informatica e delle Telecomunicazioni of the University of Catania where he is currently Associate Professor. His research interests focus on analysis and solutions for wireless networks and Internet of Things.

Sergio Palazzo (M'92−SM'99) received the degree in electrical engineering from the University of Catania, Catania, Italy, in 1977. Since 1987, he has been with the University of Catania, where is now a Professor of Telecommunications Networks. In 1994, he spent the summer at the International Computer Science Institute (ICSI), Berkeley, as a Senior Visitor. In 2003, he was at the University of Canterbury, Christchurch, New Zealand, as a recipient of the Visiting Erskine Fellowship. His current research interests are in modelling, optimization, and control of wireless networks, with applications to cognitive and cooperative networking, SDN, and sensor networks. Prof. Palazzo has been serving on the Technical Program Committee of INFOCOM, the IEEE Conference on Computer Communications, since 1992. He has been the General Chair of some ACM conferences (MobiHoc 2006, MobiOpp 2010), and currently is a member of the MobiHoc Steering Committee. He has also been the TPC Co-Chair of some other conferences, including IFIP Networking 2011, IWCMC 2013, and European Wireless 2014. He also served on the Editorial Board of several journals, including IEEE/ACM Transactions on Networking, IEEE Transactions on Mobile Computing, IEEE Wireless Communications Magazine, Computer Networks, Ad Hoc Networks, and Wireless Communications and Mobile Computing.