Auction-based Scheduling of Wireless Testbed Resources

Harris Niavis, Kostas Choumas, George Iosifidis, Thanasis Korakis and Leandros Tassiulas

***Abstract*—Experimentation in testbeds is gaining increasing ground as a necessary validation step for every theoretical study in communication networks. However, the first-come-first-served policy employed today by most testbeds does not ensure the fair and efficient utilization of their resources, which often lie idle. Ideally, every testbed should be utilized as much as possible and serve the most important requests. In this paper we introduce a novel resource scheduling mechanism for the wireless testbed NITOS. The proposed scheme is based on VCG auctions and includes an allocation and a pricing rule which induce the users to judiciously submit experiment requests. We prove theoretically and demonstrate numerically that this scheme ensures that the testbed resources (nodes and channels) are assigned to the users with the highest needs. Our mechanism can be incorporated in the next generation resource management systems for NITOS and similar testbeds.**

I. Introduction

Today there is growing consensus that a complete and detailed theoretical study of a wireless network problem must be followed by an extensive validation of the results. Therefore, it is not surprising that an increasing number of researchers run experiments on testbeds. This tendency is best exemplified by the recent deployment of many wireless testbeds around the globe, such as ORBIT [1], Emulab [2] and NITOS [3]. These facilities are open to the research community which means that anyone can access their resources - sometimes even remotely. At the same time, there are currently many ongoing efforts for federating different testbeds. Representative examples are the EU projects Fed4FIRE (fed4fire.eu) and OpenLab (www.ict-openlab.eu). The ultimate goal of these initiatives is to create a large-scale shared facility that will offer a vast set of open-access experimentation tools.

A. Motivation and Methodology

In this new era, it is crucial to derive methods for managing efficiently the resources of each testbed (or, federation of testbeds). Clearly, as the number of experiment requests increases [4], the typical "send-an-email" method for reserving the entire testbed is not efficient. In NITOS testbed, we use OMF, a cOntrol and Management Framework [5] to allocate subsets of the testbed's resources (known as *slices*) to multiple users on a first-come-first-served (FCFS) fashion. Although this policy increases the number of experiments that can be concurrently executed, it does not answer the following question: *when two or more users ask for the same wireless node who should be able to reserve it*? The testbed manager would prefer to allocate it to the user with the highest needs. However, the manager is not currently aware of the importance of each experiment.

The authors are with the Department of Electrical and Computer Engineering, University of Thessaly, Volos, Greece. Emails: {haniavis, kohoumas, giosifid, korakis, leandros} @uth.gr

In this work, we employ auction theory [6] to address this issue. Auction mechanisms manage to allocate resources efficiently, i.e., to the users (*bidders*) with the largest needs, even when the latter is private information known only to each user. The last few years auctions have been extensively used in network resource allocation problems [7]. Typically, every auction includes an allocation rule according to which the auctioned items are assigned to the bidders, and a pricing rule that determines how much each bidder will pay. Here, our objective is to induce the testbed users to reveal their actual needs, i.e., declare truthfully how important their experiments are, and assign accordingly the NITOS resources to users with the highest needs.

To achieve this goal, we use the VCG auction mechanism [6] which guarantees truthful bidding. This is ensured due to the VCG pricing rule that charges each user a price equal to the externality it induces to other users. Intuitively, a user who asks and receives a NITOS node that is requested by many other users, pays a high price since her request impacts multiple users (the other requesters). In order to enable payments, we incorporate a closed virtual economy where users are initially endowed with a virtual currency (*points*) budget which can spend to pay for their experiments. In this context, each submitted request consists of the description of the slice and the points the user is willing to pay for it. The testbed manager (or *scheduler*) collects the requests and runs periodically the auction to determine which users will be satisfied and what prices will be charged.

One particular aspect of the testbed reservation problem is that often a user is not interested in reserving a specific resource, i.e., a certain node or frequency, but can be actually satisfied with any resource of the testbed (or any resource of a specific type). In NITOS for example, a user may want to use a wireless node without being interested on whether it is mobile or fixed. *Flexibility* in requests is very important both for users, who can more easily reserve resources, and for the testbed which can satisfy more users. In the proposed mechanism, we explicitly take into account this aspect. We classify NITOS resources in different groups based on their characteristics, and allow each user to ask for a specific resource (non-flexible request) or for any resource belonging to a particular group (flexible request). Accordingly, we design additional constraints to ensure that the resources allocation will be feasible. This is a not a trivial task since each resource may belong to many different groups, e.g., a node belongs both to the general group of wireless nodes and to the group of mobile nodes.

B. Related work and Contributions

Resource allocation problems for shared infrastructures, like NITOS, appear in several networking areas. Grid computing and more recently cloud computing are two prominent examples where users generate several job (task) requests that need to be executed in the (virtualized) servers. Auction-based methods for resolving request conflicts have been often used for these systems, e.g., see [9], [10], [11] and references therein. However, NITOS testbed scheduling differs since the resources cannot be shared by many users, i.e., they are not virtualized (unlike processing power and storage capacity in clouds). Furthermore, the NITOS scheduler, in contrast to cloud managers, cannot migrate or postpone the execution of requested experiments (non-preemptive scheduling).

The various wireless testbeds around the globe do not currently employ sophisticated resource allocation schemes. For example, in ORBIT [1] each user reserves exclusively the entire testbed (400 nodes) for several hours. Even worse, this is accomplished with a first-come-first-served (FCFS) policy, which does not take into account the importance of each experiment. On the other hand, in PlanetLab [12] and VINI [13] all requests are admitted and executed in a best effort basis, i.e., without performance guarantees. Obviously, this method is also oblivious to users' needs.

To the best of the authors' knowledge, the only suggestion for a wireless testbed scheduling scheme where users (may) have different priorities is Mirage [14]. In this scheme, the testbed manager runs a first-price auction in order to determine the user requests that will be implemented. However, this mechanism does not always allocate the resources to the most important requests since first price auctions are often untruthful and hence inefficient [6]. Here, we propose a different scheme that overcomes this barrier. Therefore, there is no need for additional mechanisms such as the tax and usage control systems proposed in [14]. Besides, our mechanism models and takes explicitly into account the possibility of flexible user requests, which is an approach with increasingly interest from the research community, regarding the provisioning in wireless testbeds [15] .

To this end, our main technical contributions are:

- *Testbed Modeling*. We provide a systematic method for modeling the resources of NITOS which allows the design of auction-based mechanisms for allocating its resources. Our model takes explicity into account the flexibility of user requests. This method is generic and can be used in other wireless testbeds as well.
- *Auction Design*. We propose an auction scheme that enables the efficient usage of NITOS resources under high demand and unknown user valuations, for flexible or non-flexible requests. The mechanism is carefully designed based on our experience with NITOS, and takes into account practical considerations. Hence, it can be incorporated in the forthcoming OMF versions.
- *Performance Evaluation*. Using real data statistics from NITOS [16], we compare the proposed mechanism with the FCFS policy which is currently used in NITOS and other similar testbeds. Our findings indicate a substantial performance improvement which increases with the load and the size of the testbed, even when only a small subset of users submit flexible requests.

TABLE I
NITOS Utilization Statistics

Time Period	Number of Users	Number of Requests
1st Semester 2011	93	3038
2nd Semester 2011	134	4213
1st Semester 2012	211	5122
2nd Semester 2012	234	5403
1st Semester 2013	289	5968

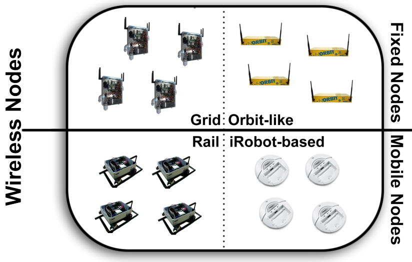


Fig. 1. The 50 NITOS nodes can be classified in 7 groups. Namely, there is the largest group of all nodes (50) which are separated in the group of fixed nodes (30) and the group of mobile nodes (20). The fixed nodes can be further classified as powerful Grid (20) and non-powerful Orbit-like nodes (10), while the mobile nodes are characterized based on their mobility pattern to Rail (10) and iRobot nodes (10).

The rest of this paper is organized as follows. Section II provides some necessary details about NITOS, presents its model abstraction (system model), and formaly introduces the NITOS scheduling problem. Section III introduces the auction mechanism, i.e., the scheduling and pricing rules, that handles both flexible and non-flexible requests. In Section IV we evaluate the performance of the auction schemes under a variety of scenarios, based on NITOS statistics we have collected. Finally, Section V concludes our study.

II. System Model and Problem Statement

Based on the aforementioned description, in this section we model NITOS resources and describe in detail the task of the NITOS auction-based *scheduler*.

NITOS Background and Model. The NITOS wireless testbed, which is deployed at the premises of University of Thessaly, comprises 50 nodes that have a variety of wireless interfaces such as WiFi, LTE, Bluetooth, etc. Every experiment can be executed using any subset of the 32 different frequency channels that are available. NITOS is open to the research community (accessible through Internet), and has been recently federated with other testbeds around the globe. This has resulted in a substantial increase of the experiment requests, as can be inferred from Table I. These requests are currently served on a FCFS basis while we also employ the concept of spectrum slicing [17] where different users can concurrently reserve different subsets of nodes and frequencies and experiment without interfering with each other. However, this slicing does not currently take into account user valuations nor it allows for flexible requests.

The latter is very important as the various NITOS resources share some common operational features and capabilities with each other, thus can be classified in different groups. Each group contains resources that, in a specific level of abstraction, are identical and hence can be used interchangeably in certain experiments (the ones that need only that level of abstraction). More specifically, the NITOS nodes can be classified in 7 overlapping groups (forming an hierarchy) as explained in Figure 1, and the frequencies in 3 groups, where the largest contains all the channels and the other 2 disjoint groups with 12 and 20 channels correspond to the 2.4GHz and 5GHz spectrum bands respectively.

More formally we model NITOS as follows[1]. We assume that there is a set $\mathcal{N}$ of $N = |\mathcal{N}| = 50$ wireless nodes and a set $\mathcal{F}$ of $F = |\mathcal{F}| = 32$ different frequency channels. The scheduler allocates each of these resources, for a minimum time duration of 2 hours. In other words, the system resource management and operation is time slotted, $t = 1, 2, \ldots, T$. We assume that the allocation of the resources is determined by the scheduler in the beginning of each time epoch which consists of $T = 12$ slots (one-day period). This means that the resources are reserved in the beginning of each day (set at 8am GMT hours).

Modeling a real system like a wireless testbed is a quite challenging task. Here, we introduce the concept of the *NITOS Resource Unit* (NRU) which is defined as a pair of node - time slot (n, t), or a pair of frequency - time slot (f, t). In other words, each of these pairs is considered a distinct resource unit of the testbed that can be allocated to at most one user during each epoch. We denote with $\mathcal{M}$ the set of all NRUs:

$$\mathcal{M} = \{(n, t), (f, t) : \forall n \in \mathcal{N}, \forall f \in \mathcal{F}, t = 1, 2, \ldots, T\} \quad (1)$$

Clearly, some of these NRUs share common features (i.e., can be used for running the same experiments) and belong to the same group as it was explained above.

Namely, we assume that there is a set $\mathcal{G}$ of NRU groups[2] where each group $g \in \mathcal{G}$ contains a certain number of $c_g = |g|$ NRUs. For example, the NRU group of the Rail NITOS nodes and a specified time slot, contains 10 NRUs. In NITOS there are 7 groups for the nodes and 3 groups for the channels which are accoringly extended to include the time dimension (and become NRU groups). The groups are overlapping and each NRU may belong to multiple different groups. In particular, for a specified time slot, a Rail node belongs to the corresponding group of Rail nodes, to the group of mobile nodes and to the largest group of all NITOS nodes. It is worth mentioning that there are groups containing only one NRU, enabling this way users to request very specific resource units.

User Requests. We assume that there is a set $\mathcal{I}$ of testbed users who are interested in running experiments during the current epoch. Each user $i \in \mathcal{I}$ submits a set of requests $\mathcal{R}_i$ to the scheduler, as depicted in Fig. 2. In every request $r \in \mathcal{R}_i$, the users asks specific NRUs or any NRUs belonging in certain groups, and provides the respective *declared valuation* $\hat{v}_r \geq 0$. Notice that the actual valuations v_r are private information, known only to each user, and in general can be different that those declared, i.e., $v_r \neq \hat{v}_r$. For example, a user may declare

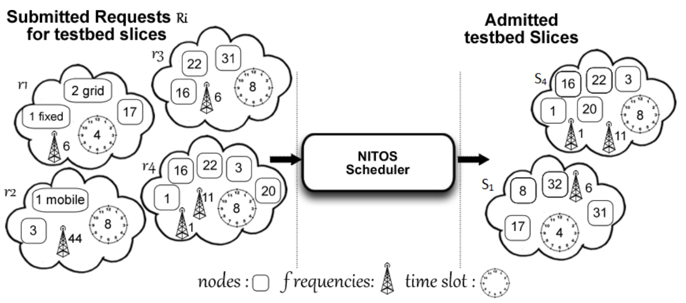


Fig. 2. Schematic diagram of the NITOS resource scheduling process.

a higher valuation $\hat{v}_r > v_r$ believing that this will increase the probability to acquire the requested resource units. We define the vector of all submitted valuations:

$$\hat{\boldsymbol{v}} = (\hat{v}_r \geq 0 : \forall r \in \mathcal{R}) \quad (2)$$

Note that every request may ask NRUs belonging to different groups, more general or more specific. For example, consider a user with a general (i.e., simple) experiment request who asks for a specific frequency and any couple of wireless nodes. He can be satisfied by any couple of nodes that belong to the largest group of all NITOS nodes and the specific requested channel. We need to stress here that a user is satisfied only if he receives all the requested NRUs. The subset of the resource units that user reserves due to a specific request $r \in \mathcal{R}$ is called slice S_r.

Charging Scheme. We assume that there is in place a pricing and charging mechanism. This can be any type of closed virtual economy or point system that can be incorporated in the new version of the NITOS Scheduler. Each user is endowed with $B \geq 0$ virtual currency units (hereafter referred to as points) in the beginning of a large time period which consists of Q time epochs[3]. This is the budget of each user that she can use to pay for the slices she needs during these Q epochs.

For each user $i \in \mathcal{I}$, the scheduler determines the subset of her requests $\tilde{\mathcal{R}}_i \subseteq \mathcal{R}_i$ that will be admitted and the aggregate points $p_i \geq 0$ that she will pay for all the admitted requests. The prices are determined according to the *payment rule* the scheduler employs. Clearly, as it will be shortly discussed, the price for a user $i \in \mathcal{I}$ depends both on the requested resource units and the declared valuation $\hat{v}_r$ for each of her admitted requests $r \in \tilde{\mathcal{R}}_i$. Obviously, it is most likely that a user will succeed in getting cheaper a resource when she submits a request like *give me any of your fixed nodes*, instead of submiting a request for a specific fixed node like *give me node 29*.

Scheduling Problem. The objective of the scheduler is to maximize the efficiency of NITOS by admitting the most important requests. Let us introduce the $0-1$ discrete variable $x_r = \{0, 1\}$ which determines whether the request r will be admitted or not. We define the respective allocation vector:

$$\boldsymbol{x} = (x_r \in \{0, 1\} : \forall r \in \mathcal{R}) \quad (3)$$

Similarly, we introduce the pricing vector for all users in $\mathcal{I}$:

$$\boldsymbol{p} = (p_i \geq 0 : \forall i \in \mathcal{I}) \quad (4)$$

Formally, the problem of finding the optimal allocation-pricing policy $(\boldsymbol{x}^*, \boldsymbol{p}^*)$ is defined as follows:

[1]Although the auction is designed for NITOS, we provide a parameterized formulation, i.e., using variables instead of specific numbers, so as to have a general formulation applicable for any similar testbed infrastructure.

[2]Hereafter we use the term "group" and "set" interchangeably.

[3]The exact value of Q can be defined using statistic data from NITOS. Our preliminary results indicate that a month is a suitable time period.

NITOS Scheduling Problem: Given a set $\mathcal{R}$ of user requests for experiments, within a certain time epoch, and a set $\mathcal{M}$ of NITOS Resource Units (NRUs), the scheduler must devise the allocation $\boldsymbol{x}^*$ and pricing $\boldsymbol{p}^*$ policy which ensures the efficient usage of NITOS resources.

In the sequel we present an auction scheme that address this problem while takes into account the particular characteristics of testbed scheduling. That is, user requests are, in general, flexible but at the same time inelastic since, in order to satisfy a user, the scheduler should allocate to her all the requested resources (i.e., the entire slice).

III. Scheduling Policy

In this section we use auction theory and design an allocation and pricing mechanism that successfully addresses the above challenges and solves the NITOS Scheduling Problem for flexible and non-flexible user requests. One of the prevalent auction mechanisms that ensures truthful bidding under a variety of settings and assumptions is the celebrated Vickrey - Clarke - Grooves (VCG) mechanism [6], which we also employ in this work.

A. Allocation Rule

Let us begin with the allocation rule. In the beginning of each epoch, the scheduler collects all the requests $\mathcal{R}$ and constructs the NRU request matrix $\boldsymbol{A}$:

$$\boldsymbol{A} = \big(a_{rg} \in \{0, ..., c_g\} : \forall r \in \mathcal{R}, \forall g \in \mathcal{G}\big) \tag{5}$$

where a_{rg} is the number of NRUs from group $g \in \mathcal{G}$ that request $r \in \mathcal{R}$ asks. The goal of the NITOS scheduler is to maximize the total sum of the valuations of the users whose requests are admitted, i.e., to maximize $\sum_{r\in\mathcal{R}} x_r\hat{v}_r$. To achieve this, the scheduler should resolve the conflicts created by the different requests.

One unique aspect of the proposed scheme is that it allows conflict resolution for requests of different flexibility. Note that the creation of the different NRU groups implies that we actually have less physical resource units than the union of these sets (groups). For example, if a certain mobile node is assigned to a user with a flexible request (i.e., for any node), then this node cannot be assigned to another user who has requested it (in the same slot). To resolve such conflicts, and ensure that the admitted requests can be actually implemented, we need to carefully design the constraint set of the NRU allocation problem.

First, notice that for any two groups $g_1, g_2 \in \mathcal{G}$ it holds $g_1 \supseteq g_2$, $g_1 \subseteq g_2$, or $g_1 \cap g_2 = \emptyset$. In other words, based on the definition of NITOS groups, there is an hierarchy among the NRU groups. The lowest level groups are the singletons consisting of the specific nodes and frequencies, while the highest level groups have no supersets in $\mathcal{G}$ (e.g., the group of all frequencies). A set of different requests can be concurrently admitted only if they are not violating the size (or, *capacity*) constraint for any group $g \in \mathcal{G}$ that the requested NRUs belong to. Hence, we need to ensure for every group $g \in \mathcal{G}$, and all its subgroups $s \subseteq g$, with $s \in \mathcal{G}$, that their size constraint is not violated. Namely, that the scheduler does not assign more NRUs (to specific or more flexible requests) than those that are available in the respective groups.

Summarizing the above, the scheduling problem (SP) can be written as follows:

$$\text{SP:}\quad \max_{\boldsymbol{x}} \sum_{r\in\mathcal{R}} x_r\hat{v}_r \tag{6}$$

$$\text{s.t.}\quad \sum_{s\in\mathcal{G}:s\subseteq g}\left(\sum_{r\in\mathcal{R}} a_{rs}x_r\right) \leq c_g, \forall g \in \mathcal{G} \tag{7}$$

$$x_r = \{0,1\}, \forall r \in \mathcal{R} \tag{8}$$

where constraints (7) ensures that the admitted requests can be implemented, i.e., no more NRUs than the available are assigned under any level of abstraction/flexibility. This is a discrete optimization problem, due to (8), based on the weighted set packing problem which is NP-hard [8]. However, as it will be shown in the sequel, the small dimension of the NITOS scheduling problem renders the complexity of its solution numerically affordable for offline calculations as those we need in testbeds (running once a day or a week).

The solution $\boldsymbol{x}^*$ of the (SP) problem indicates which requests should be admitted, yet it does not determine which specific NRUs should be used to satisfy each one of the admitted requests. The process of mapping the admitted requests to specific available (i.e., not allocated to other requests) resources is the following: We start from the lowest and continue to the highest level groups, mapping randomly selected NRUs of these groups to the admitted requests. The following example will illustrate sufficiently the whole process.

Numerical Example. At this point, let us provide a simple yet indicative example to further explain the allocation scheme. Consider a group $g_M = \{a_1, a_2, a_3, b_1, b_2, b_3\}$ of testbed nodes, that is separated into two subgroups $g_a = \{a_1, a_2, a_3\}$ and $g_b = \{b_1, b_2, b_3\}$ of a-type and b-type nodes respectively (for simplicity, we assume that $T = 1$). Also, for each node there is an extra singleton group, i.e., $g_{a_1} = \{a_1\}$, $g_{a_2} = \{a_2\}$, $g_{a_3} = \{a_3\}$, $g_{b_1} = \{b_1\}$, $g_{b_2} = \{b_2\}$ and $g_{b_3} = \{b_3\}$. Assume that there are 3 user requests r_1, r_2 and r_3, as follows: (i) r_1 asks for 2 nodes of a-type (from g_a), (ii) r_2 asks simply for any 2 nodes (from g_M), and (iii) r_3 asks for 1 a-type node (from g_a) and the b_3 node (from g_{b_3}). The requests are summarized in Table II.

In order to understand whether these requests can be concurrently admitted we need to carefully check the capacity constraints for each group and for all its subgroups. Starting from the smaller groups (which do not have subgroups) we see that there aren't more than one requests asking for the same specific node (singleton group). Accordingly, we check the a-type and b-type groups. Again we see that the requested items of a-type (or a_1, a_2, a_3 type) do not exceed in total the size of group a which is 3. Similarly we can verify for the b-type nodes. Finally, all the requests do not ask in total more than 6 nodes, which are the actual nodes the testbed has (the size of g_M). Hence, constrain (7) is satisfied $\forall g \in \mathcal{G}$ if $x_r = 1$, $\forall r \in \mathcal{R}$.

After finding which requests will be admitted, we can determine how they will be implemented. This resource allocation process starts from the lowest level singleton groups. Subsequently, the user of request r_3 reserves the b_3 resource and the a_2 resource. Then, the user of request r_1 reserves the two leftover resources of a-type and at last, the user of request

TABLE II
AN EXAMPLE OF A FLEXIBLE REQUEST MATRIX $\bar{\boldsymbol{A}}$

$\bar{\boldsymbol{A}}$	g_M	g_a	g_b	g_{a_1}	g_{a_2}	g_{a_3}	g_{b_1}	g_{b_2}	g_{b_3}
c_g	6	3	3	1	1	1	1	1	1
r_1	0	2	0	0	0	0	0	0	0
r_2	2	0	0	0	0	0	0	0	0
r_3	0	0	0	0	1	0	0	0	1

r_2 reserves the remaining two b_1 and b_2 resources. Notice that this greedy approach satisfies first the more specific requests and accordingly the more general ones. Due to (7), it is ensured that the method will satisfy all the admited requests.

The solution of the SP problem is the optimal (i.e. efficient) request admission policy $\boldsymbol{x}^*$ which yields an aggregate system valuation (or welfare):

$$SW\big(\boldsymbol{x}^*, \hat{\boldsymbol{v}}\big) = \sum_{r \in \mathcal{R}} \hat{v}_r x_r^* \tag{9}$$

Notice though that this amount represents the testbed efficiency only if the declared values $\hat{v}_r$ are equal to the actual values of the user requests v_r, $\forall\, r \in \mathcal{R}$. This is ensured by the proper selection of the payment rule.

B. Pricing Rule

The choice of the payment rule is of crucial importance. In VCG mechanisms, each bidder (user) is charged with a price that is equal to the externality she induces to the system. For example, a user who asks and receives a large slice (e.g., consisting of many and popular NRUs), will have to pay a high price since her request results in dissatisfying many other users who are rejected by the scheduler and will not be able to run their experiments.

This payment rule for each user $i \in \mathcal{I}$ can be formally defined as follows:

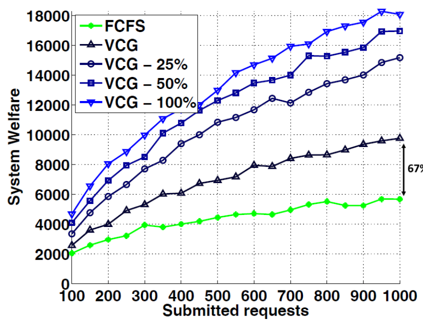
$$p_i = SW\big(\boldsymbol{x}^*_{-i}, \hat{\boldsymbol{v}}_{-i}\big) - SW\big(\boldsymbol{x}^*, \hat{\boldsymbol{v}}_{-i}\big) \tag{10}$$

where $\boldsymbol{x}^*_{-i}$ is the solution of (SP) that we obtain if we change the valuation vector to $\hat{\boldsymbol{v}}_{-i}$, setting $\hat{v}_r = 0$, $\forall r \in \mathcal{R}_i$. In other words, the first term is the aggregate valuation of the admitted experiments of the users, other than i, when i does not participate in the auction. The second term is the aggregate valuation of the admitted experiments, of users other than i, when i does submit her requests (and hence impacts the other users).

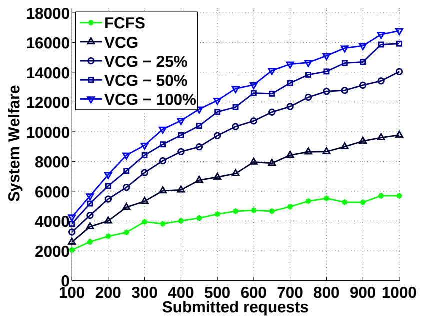
The VCG payment induces all the users to reveal their actual experiment valuations. Namely, each user $i \in \mathcal{I}$ declares valuations $\hat{v}_r = v_r$, $\forall r \in \mathcal{R}_i$ (incentive compatibility property). Therefore, the solution of the SP succeeds in allocating the slices to the users with the highest needs. Notice that this guarantees that the testbed resources will be utilized as much as possible, which is one of the main considerations of the testbed managers. Clearly, a user with high needs is expected to better utilize the reserved slice. Finally, we need to stress that VCG mechanisms posses the so-called individual rationality property, i.e. $p_i \leq \sum_{r \in \mathcal{R}_i} v_r$, $\forall i \in \mathcal{I}$, which means that each user will not have to pay a higher price than the actual valuation of her requests.

IV. PERFORMANCE EVALUATION

We compare the proposed VCG-based scheduling with the current NITOS FCFS scheduling policy, using as performance metrics the number of admitted requests and the produced welfare (efficiency).



(a) Flexible requests ask for wireless nodes and frequency channels in general.



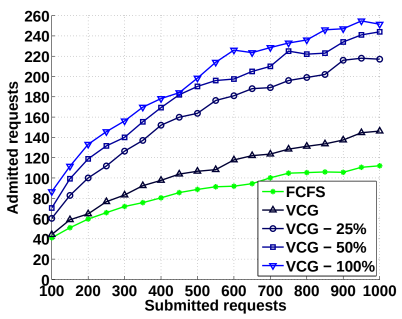
(b) Flexible requests ask for fixed or mobile nodes and 2.4GHz or 5GHz frequency channels.

Fig. 3. System Welfare of the proposed VCG-based scheduling policy with the current FCFS policy, with and without flexibility. The labels of the VCG plots indicate the percentage of the flexible requests.

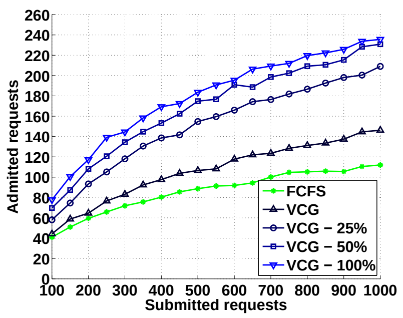
Simulation setup and methodology. We used statistics from NITOS experiment requests that are collected over the last two years [16]. Based on their analysis, we assume that each requested slice includes on average 5.3 nodes and 1.5 frequencies. Moreover, we have measured the request probabilities for each one of the NRUs and accordingly calculated the respective probabilities for the higher-level (more generic) NRU groups. Clearly, the probabilities do not follow a uniform distribution as some NRUs are more advanced and hence more popular to the users (for example, Grid nodes are requested more often than Orbit nodes). User valuations are randomly drawn from a uniform distribution in the interval $v_r \in [1, 100]$.

We investigate the algorithm performance in 3 different scenarios regarding the flexibility of user requests. Namely, we consider the cases that 25%, 50% or 100% of user requests are flexible. We want to see how the system performance is affected as more users become flexible. Additionally, we study the impact of the type of flexibility. That is, we first consider the highest degree of flexibility, assuming that flexible requests ask for any type of node and channel. Next, we consider less flexible requests which ask for fixed or mobile nodes and channels in 2.4GHz or 5 GHz bands. This reveals how the system is affected by the type (or degree) of request flexibility. For all the plots we have run the simulation for different number of submitted requests (ranging from 100 to 1000), while each of these simulations is repeated 20 times.

Numerical Results. In Figure 3(a), we compare the Sys-



(a) Flexible requests ask for wireless nodes and frequency channels in general.



(b) Flexible requests ask for fixed or mobile nodes and 2.4GHz or 5GHz frequency channels.

Fig. 4. Number of admitted requests under the proposed VCG-based scheduling policy and the current FCFS policy, with or without flexibility. The labels of the VCG plots indicate the percentage of the flexible requests.

tem Welfare (SW) of the proposed VCG-based scheduling policy with the respective SW of the existing FCFS policy. Interestingly, the efficiency improvement due to the proposed scheduling policy can be up to 67%, under 1000 non-flexible requests, which motivates its adoption by NITOS and other similar testbeds. Notice that the simulations (with exhaustive search) required less than 15 minutes on average to calculate the scheduling policy for one epoch. Moreover, the admitted requests of our policy exceed the corresponding ones of the existing policy, as we can see in Figure 4(a). This means that not only we admit the most important requests (those with the highest valuations) but also we increase the number of served users. The performance improvement of the scheme increases with the number of submitted requests.

Accordingly, we study the performance of the proposed policy when users submit flexible requests. In the same Figures 3(a)-4(a), there are some plots that correspond to different percentages of flexible requests (percentages of users who submit flexible requests), whereas each flexible request asks for the highest level groups of resources. Obviously, as the percentage of flexible requests (wrt the total submitted requests) increases, the scheduling performance improves also. *It is important to notice here that even when only* 25% *of users submit flexible requests, the proposed scheme yields a significant performance improvement.*

In addition, we run one more set of simulations, where the flexible requests are asking now for lower level groups of resources (less flexible). Notice that these requests are still flexible, since they are not demanding specific resources, but less flexible than the previous ones, since they are asking resources from lower level (i.e., smaller) groups. As it is depicted in Figures 3(b) and 4(b), the performance of the scheduling policy is degraded when the flexible requests are asking for lower level groups, compared to the previous results depicted in the figures above.

V. Conclusions

This work is motivated by our every day experience with NITOS, and constitutes the first step in our effort to design and implement an auction scheme for the practical and important problem of efficient resource allocation in wireless testbeds. We have proposed a combinatorial VCG multi-item auction which ensures the efficient utilizatoin of the testbed's resources while takes into account the flexibility that some user requests may have. Our real data-driven numerical analysis showed significant performance gains compared to the schemes currently employed by NITOS and other testbeds.

There are many interesting directions for future work which we will pursue in order to complete our study and implement this new scheduling - pricing policy. For example, the selection of the values for the system parameters such as the time period T and the budget B, can be made using more detailed statistics about user activities which we are currently collecting. Even more challenging is to study the users behavior. Namely, we expect that users will encounter difficulties in deciding how many points they should allocate to each of their experiments within the periods Q. Assisting user decisions, is another intriguing direction that actually falls into the context of behavioral economics. Finally, we are planning to extend our study for users that are flexible in the time dimension as well.

References

[1] ORBIT Wireless Testbed, www.orbit-lab.org.
[2] Emulab: Network Emulation Testbed, www.emulab.net.
[3] NITOS Wireless Testbed, nitlab.inf.uth.gr/NITlab/index.php/testbed.
[4] Openlab Newsletter, www.ict-openlab.eu/fileadmin/documents/newsletters/OpenLab_News_03-2013.pdf.
[5] T. Rakotoarivelo, M. Ott, G. Jourjon, and I. Seskar, "OMF: a control and management framework for networking testbeds", *SIGOPS Oper. Syst. Rev.*, vol. 43, no. 4, Jan 2010.
[6] V. Krishna, "Auction Theory", *Academic Press*, 2010.
[7] I. Koutsopoulos, and G. Iosifidis, "Auction Mechanisms for Network Resource Allocation", *in Proc. of WiOpt/RawNet*, 2010.
[8] S. Vries, and R. V. Vohra, "Combinatorial Auctions: A Survey", *INFORMS Journal on Computing*, vol.15, no.3, pp.284-309, 2003.
[9] R. Buyya, D. Abramson, J. Giddy, and H. Stockinger, "Economic models for resource management and scheduling in Grid computing", *Concurrency and Computation: Practice and Experience*, vol. 14, no. 13-15, 2002.
[10] Q. Zhang, Q. Zhu, and R. Boutaba, "Dynamic Resource Allocation for Spot Markets in Cloud Computing Environments", *in Proc. of IEEE UCC*, 2011.
[11] H. Zhang, B. Li, H. Jiang, F. Liu, A. V. Vasilakos, J. Liu, "A Framework for Truthful Online Auctions in Cloud Computing with Heterogeneous User Demands", *in Proc. of IEEE Infocom*, 2013.
[12] PlanetLab, planet-lab.org.
[13] VINI, "A Virtual Network Infrastructure", http://www.vini-veritas.net/.
[14] B.N. Chun, *et al.*, "Mirage: a microeconomic resource allocation system for sensornet testbeds", *in Proc. of IEEE Workshop Embedded Networked Sensors*, 2005.
[15] G. Coulson, *et al.*, "Flexible experimentation in wireless sensor networks", *Communications of the ACM* , vol.55, no. 1, pp. 82-90, 2012.
[16] NITOS Usage Statistics, available online at volos.iti.gr/~gitsis/.
[17] A.C. Anadiotis, *et al.*, "Towards maximizing wireless testbed utilization using spectrum slicing", *in Proc. of Tridentcom*, 2010.