Towards Maximizing Wireless Testbed Utilization using Spectrum Slicing *

Angelos-Christos Anadiotis[1], Apostolos Apostolaras[2], Dimitris Syrivelis[2], Thanasis Korakis[2], Leandros Tassiulas[2], Luis Rodriguez[3], Ivan Seskar[4], and Maximilian Ott[5]

[1] School of Electrical and Computer Engineering
National Technical University of Athens
Athens, Greece
[2] Department of Computer and Communication Engineering
University of Thessaly
Centre for Research & Technology Hellas (CERTH)
Volos, Greece
[3] Atheros Communications
[4] WINLAB, Rutgers University
Technology Center of New Jersey, USA
[5] National ICT Australia(NICTA)
Alexandria, NSW 1435, Australia

Abstract. As experimentation becomes one of the de-facto approaches for benchmarking, researchers are turning to testbeds to test, review and verify their work. As a result, several research laboratories build wireless testbeds, in order to offer their researchers a real environment to test their algorithms. As testbeds become more and more popular, the need for a managerial tool that will not only provide a unified way for defining and executing an experiment and collecting experimental results, but that will also serve as many users as possible maximizing the utilization of its resources, is growing. In this spirit, we propose a scheme that exploits wireless testbeds functionality by introducing *spectrum slicing* of the testbed resources. This scheme can be incorporated inside OMF, an already existing wireless testbeds managerial framework, which is widely used by many researchers.

1 Introduction

The theoretical analysis and the simulation of a new wireless protocol or technique can give us important information about its performance in terms of throughput, delay, power consumption, etc. However, in order to have analytically tractable models, several simplifications of the real world have to be made. While the simulations have the ability to incorporate more general models, we are

* The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement n°224263

still limited by the complexity of the simulation software and our limited knowledge of the wireless environment. Some specific limitations of the simulation approach in depicting a real wireless network include inaccurate respresentation of the wireless medium, simplification of synchronization issues that occur in wireless terminals and ignorance of several aspects such as the computational overhead.

Due to the above limitations, researchers have focused in the last few years on the studying of wireless schemes through implementing them on real platforms. Most of the implementation is done on open source platforms, such as software defined radios or open source wireless drivers. This new trend in wireless networks has triggered the birth and evolution of several wireless testbeds around the globe. Researchers may reserve a testbed for a specified time and execute their experiments there. But, how is that reservation made? Until now, the experimenter reserved the whole testbed (or a very large part of it if we are talking for a really big testbed such as ORBIT) even if he actually needed only a few nodes and frequency channels. This reservation policy prohibits other users from using the testbed at the same time, since the experiments may interfere with each other. Moreover, most of the times the reservation is made after an oral agreement between the potential users.

An answer to these issues would be the dynamic, on-demand partition of the testbed to smaller parts, based on the available resources and the experimenters demands. So, we need to build a managerial mechanism that will be able to both handle multiple requests from the testbed users and partition the testbed efficiently by creating virtual slices and assigning them to the respective users. We intend to build such a mechanism using spectrum slicing techniques.

Currently, one of the most used testbeds is ORBIT [11] in WINLAB [6]. ORBIT consists of 400 nodes, available to the registered users. It has a very well organized management system which allows users to book the testbed at available time slots. Although ORBIT's reservation framework is very useful since it allows a large amount of users to remotely access the testbed, it has a significant drawback: It does not allow for efficient use of the testbed resources. In most of the experiments, only a small amount of nodes are being used, while the rest are staying idle. Usually, a researcher reserves the whole testbed (400 nodes) for a couple of hours and he only uses no more than 10 nodes, leaving the rest 390 nodes idle. With slicing, these nodes could serve the needs of other users.

ORBIT's example shows the need to develop a tool that will maximize the utility of a wireless testbed. In this paper, we are proposing a scheme based on spectrum slicing, which takes advantage of the large availability of a particular resource -that is spectrum- and, through that, increases the whole testbed's availability to experimenters. Of course slicing can refer to other resources too, such as power (adjust the power that each slice will transmit to create a "safe" area for each user), network cards (a node that has many network cards could assign subgroups of them to different experimenters) and nodes (many users could use the same node using virtualization techniques), however in this paper we

focus on spectrum slicing. This scheme is developed as a part of a more generic managerial framework that is being designed in the concept of OneLab2 [4]. OneLab2 intends to federate heterogeneous testbeds located in different places under a unified system. As we are illustrating in later sections, our new managerial mechanism allocates a particular group of channels to a group of nodes that is assigned to one user. In this way, we optimize the resources usage of the testbed by allowing multiple users to operate on the testbed simultaneously, without interfeering with each other.

The challenge in wireless testbeds slicing is the isolation of experiments, as there are inter-dependencies among the resources. In contrast to a wired interface where all we need to do is to manage the sharing of a specified resource on a single node, sharing a wireless interface may also affect the sharing of interfaces on other nodes. What correlates them are things like spectrum, location and power which are also correlated. Power and location for instance, are two factors that could affect each other.

The remainder of the paper is structured as follows: In Section 2 we give related and previous work made in the domain of wireless testbeds resource allocation. In Section 3 we give a short description of the OMF framework that we used for developping our scheme. In Section 4 we present our spectrum slicing scheme from a top-down approach. In Section 5 we present our testbed, an example usage scenario for our scheme and statistics which give us useful feedback on the improvement of testbed utilization. Finally, in Section 6 we give the conclusions that he have reached through our work in this field and in Section 7 our future plans.

2 Related Work

Several work has been made on efficient resource allocation on wireless testbeds. However, most of this work is focused on virtualization techniques, which implies more complex implementation and operating system dependence. Next, we are giving two representative examples of such systems:

Emulab. Emulab is a network testbed, giving researchers a wide range of environments in which to develop, debug, and evaluate their systems. In Emulab, there has been developed a system which virtualizes hosts, routers and networks, while retaining near total application transparency. This system is based on FreeBSD Jails, which provides filesystem and network namespace isolation and some degree of superuser privilege restriction.[13]

Mirage. Mirage is a resource allocation system, which was designed for sensor networks testbeds and it is based on an auction scheme. The experimenters are bidders, who argue for resources, using a virtual currency issued by the central system. So, if a user uses the testbed in a way that matches the system's criteria, then he has more credits to claim resources for a next experiment.[10]

NITLab. In NITLab [5], we have implemented a spectrum slicing scheme, which however had some significant drawbacks and we decided to change it to this one we are describing here. Specifically, we had focused on the new framework of Linux wireless drivers, provided by cfg80211 [1]. This packet, which is meant to replace Wireless Extensions [7, 15], can support Central Regulatory Domain Agent (CRDA) [2] which controls the channels to be set on the system, based on the regulations of each country. By making some changes on this, we managed to succeed spectrum slicing on our testbed. However, this scheme limited us in terms of the available drivers that could be used with it and the Linux kernel versions that could enhance CRDA. Moreover, this scheme's implementation was tricky and very much system dependable. [8, 9]

Our work here is independent from the related works described above and can be used in cooperation to them, as it schedules resource utilization from a higher level. Furthermore, we are moving our implementation on to a more abstract level, that is the one of the management framework, in order to set it platform independent, since OMF is intended to cover more platforms that just Linux. An analysis on virtualization schemes can be also found in [14], however in this paper, we are actually implementing the spectrum slicing scheme, which as shown in Section 5 has a very good performance on our testbed, while with its extensions that we are planning (see Section 7), we are expecting to scale for even more large and complex testbeds.

3 Wireless Testbed Managerial Framework

We are using cOntrol and Management Framework (OMF) [3] for managerial framework. Currently OMF is deployed on several testbeds around the globe, including ORBIT. Using a ruby-like experiment definition language, the experimenter writes an abstract description of the experiment, stating which nodes to use and what for, uses traffic generators, sinkers and other utilities which are being constantly updated and integrated inside OMF. Providing full transparency to users, OMF is responsible for loading their images to the testbed nodes that they have asked for, for configuring the nodes based on the experiment description and for gathering the results. Currently OMF consists of three basic components: *Gridservices*, *Nodehandler* and *Nodeagent*. Next, we give a short description of each one of them:

Gridservices. Gridservices consist of a set of web services, which are responsible for both executing system actions, such as turning a node on or off, rebooting nodes, loading images, etc. and getting information about the testbed as they have access to two databases: one for the testbed and its configuration and one for the scheduler where we keep information critical for slicing. Gridservices are residing on the testbed server.

Nodehandler. Nodehandler does the actual testbed management using Gridservices and other operating system applications. The user interacts with the Nodehandler to load an image to the nodes and to execute an experiment. Based on the experiment definition, which Nodehandler is responsible to interpret, this component is sending the respective commands to the nodes in order to configure them and trigger the applications needed for the experiment. Like Gridservices, Nodehandler runs on the testbed server too.

Nodeagent. Contrarily to Gridservices and Nodehandler, Nodeagent runs on the client-side of the testbed; that is the nodes. Previously we said that Nodehandler is responsible for sending commands to the nodes, based on the experiment definition. Here comes Nodeagent, which is responsible for receiving these commands them, understanding them and then trigger the respective applications. These applications could refer to the node configuration, a traffic generator, a traffic sink, etc.

We had to extend all the three components above to integrate spectrum slicing support inside OMF. In the next section, we will show our basic idea for achieving spectrum slicing, the dilemmas and the decisions we had to make when implementing our scheme in OMF.

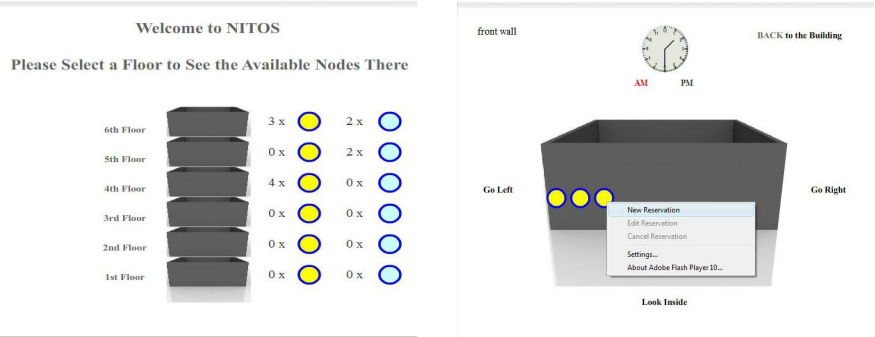
4 Scheduling Experiments on Wireless Testbeds

Currently OMF does not include any scheduling algorithms that would synchronize the experiments execution. Its implementation does not include any permissions checking for access to the testbed resources. However, in a public, multiuser environment, we need a system that will be able to assign resources only to the users that have the right to use them, while offering the experimenters a way to declare the resources that they need for their experiments. In our work, resources are divided in two categories: *nodes* and *spectrum*. So, we are providing a tool which is used by the experimenters to reserve nodes and spectrum for some time (which should not exceed some limit). Using spectrum slicing, our tool makes the testbed available to users who would like to use different resources at the same time.

4.1 Spectrum Slicing

By slicing, we mean the partitioning of the testbed based on some criteria. With spectrum slicing, we aim to partition the testbed into smaller, virtual, testbeds which are using different spectrum and, hence, they do not interfere with each other. The spectrum that each virtual testbed will use could be either defined by the experimenter at scheduling or dynamically assigned, if the experimenter does not care about the channel that he uses (for example he could ask for any channel of 802.11g modulation).

Spectrum slicing can be combined with any other resource allocation scheme, since it does not require any "negotiations" with other system resources. Furthermore, spectrum is always associated with a wireless testbed experiment and, hence, there will always be a chance to slice the testbed based on the wireless channels each experiment needs.



(a) Testbed deployment overview. (b) Selection of particular testbed node.

Fig. 1. NITOS Scheduler Node Selection

4.2 Allocating Resources - Slices

Slices are created dynamically, upon the user reservation procedure. As we have already mentioned, we discriminate resources in two categories: nodes and spectrum. Resource allocation can be made statically or dynamically. Currently, we have developed a static scheme, but we working on extending it based on Topology and Link Quality Assessment Protocol (TLQAP) [12]. In this scheme, the experimenter selects the nodes and the channels he would like to use during reservation, while at the same time, he also declares the time slots that he will be using those resources. Next, we are illustrating the basic idea of our resource allocation scheme, based on spectrum slicing. Finally, we are making a brief report on how dynamic resource allocation would be succeeded by extending our already existing tools.

Let us consider a testbed with OMF as its management system. As we have already mentioned, OMF does not include a scheduler, hence we need to develop one as a separate component of our system. In NITLab, we have developed a scheduler, whose User Interface is available to public, through our web site. This User Interface is responsible for guiding the user through the reservation process and is designed in such a manner that the experimenter may have a very specific view of the testbed topology. Providing outside and inside view of our six-floor building, we aim to give the experimenters the best perspective of the nodes that they are reserving for their experiments.

Now consider an experimenter who would like to use the testbed. Assuming that he has already registered, he may log in to the scheduler's web site and gain access to its User Interface. From there, he first chooses the date that he would like to run his experiments. Then, the actual scheduling process begins, with the experimenter seeing our testbed building with an indication beside each floor on the number of each node type that reside on that floor, as shown in Figure 1(a).

Based on these data, the user can choose a floor and guide around it from both an outside and inside view. Having an exact view of the position of each node, he makes his choice by selecting to reserve one, as shown in Figure 1(b). The clock that appears on this frame can be clicked by the user on the time he would like to check the nodes status. By clicking there, the frame is automatically refreshed and the nodes are colored according to their status, while at the same time, the new user loses permission to request a new reservation on that node at that time. So, at this phase, the demand for reservation overlap prevention is satisfied; the experimenter chooses a node available at the time he needs it and proceeds to the next step.

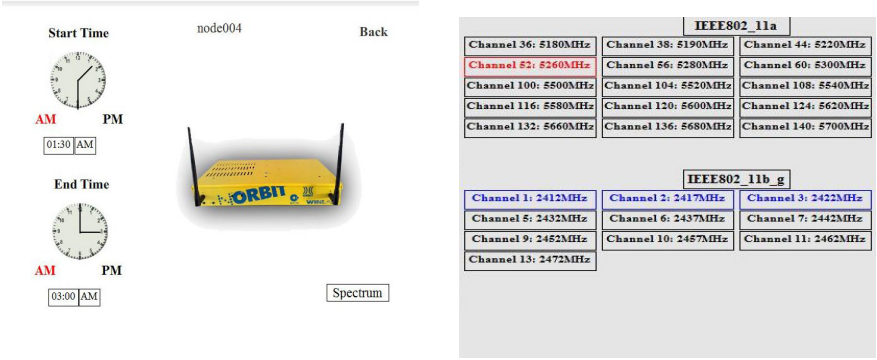
At this point, the user has selected his node and he is about to reserve it for some time. For this end, we give him two clocks, one for choosing the start time for his experiment and one for the end time (see Figure 2(a)). Giving the user another clock here may seem to have a security gap since the user may try (either willingly or not) to "trick" the scheduler. However, this cannot happen. First of all, the clock of the previous step is used for checking and not for reserving, as such a thing would not be very practical for the user. Then we need to perform the same check for availability of the current node here too. So, when the time duration that the experimenter chooses at this step, includes time of another user on this node, the scheduler does not allow him to move on to the next step and, hence, reserve the node.

Guided by the scheduler, the experimenter has successfully chosen a node and some time to use it. The last thing he has to do is to choose the spectrum he would like to use; that is a group of channels that will be reserved for him during his time (see Figure 2(b)). Again, the scheduler does not allow the experimenter to choose a channel that is reserved by another one during that time. This is the final step; the experimenter submits his choice and the system reserves the node and the spectrum for him. After that, he goes to the first step, getting the picture of the whole bulding to continue with his reservations.

The scheduler identifies the user and lets him edit or delete his reservations at any time. It also keeps track of the last choices that he made on reservation time and spectrum, providing them to him as default choices for the current session. Finally, the scheduler provides the experimenter with the option to check out all his reservations, grouped by the reservation time. So, at any time, he may login and checkout the nodes and the channels he has reserved for some time.

4.3 Implementation

The implementation of our spectrum slicing scheme is done on two levels: (a) the user interface which guides the user through the reservation process and does not



(a) Time reservation of particular node. (b) Spectrum Selection.

Fig. 2. NITOS Scheduler Resource Reservation

allow him to reserve an already reserved resource and (b) the OMF components, where we have added new and extended old ones to succeed the monitoring and control of the slices that are created at reservation. Next, we are examining in more depth the implementation details of each one of these two levels.

4.3.1 User Interface

The scheduler's user interface is designed to be available through a web site, so that any users may have access to it. Its goal is to allow the experimenters reserve the resources they need (in terms of nodes and spectrum) in an efficient way for the testbed usage. So, we need to reassure two things: on the first hand an easy to use environment and, on the other hand, an application that does not allow their choices to mess with other experimenters ones. Next, we are giving the reservation procedure procedure giving all the details of what happens underneath.

First the user has to log in and choose a date for his experiment. After that, we create session for that user where we hold the details of his account. From this point on, the scheduler knows who that user is and, based on that and the date, it manages permissions to resources that the user might need to access on the next steps. The main scheduling application is now deployed. This application consists of a flash animation which uses multiple PHP scripts and XML files to give the experimenter the information he needs, as we explain next.

The scheduler gives the user a perspective of the testbed topology. On our testbed, NITOS, which is located on a six floor building, the scheduler shows the number of each node type, residing on each floor. The topology view is loaded dynamically by using XML files. The scheduler application reads the respective configuration file and loads the topology that the experimenter will be able to use during his session. We have reached the choice of XML files because we are aiming to develop a tool that would easily support any other similar wireless testbed, without having to do any major changes on the code. Moreover, the testbeds

themselves are not static and it would not be convenient for the administrators to change the code each time a node falls down, or a new one is added to the testbed.

The user clicks on a floor and gets its perspective. Our user interface provides full inside and outside view of the floor. Along with this view, scheduler provides the user a clock where he clicks on the time he would like to check for reservations. Each click triggers a PHP script, which checks a database that resides on the testbed web server (for greater speed) and colors the nodes respectively. So, if the node is free at that time, it is colored with its native color and the user can make a new reservation. If the node is reserved by another user, it is colored red and the current user cannot do anything on it. Finally, if the node is reserved by this user, it is colored purple and the user can edit his reservation. What we actually do here is to grant permissions on each node (resource) based on the user that claims it.

The next step, is using similar tools with the experimenter clicking on start and end time for his experiment and the scheduler checking if those are available. At each step, the scheduler does not allow the user to move on without making a right choice. When this step has finished, the experimenter has selected a node and a time duration for his experiment.

Moving on to the final step, the experimenter has to declare the set of channels that he needs to perform his experiments. Again, using an XML file, scheduler loads all channels available, based on the laws of each country. After that, it checks the database to see which of these channels are reserved by other users and which ones are reserved by this user at a previous step. Keeping the same template as before, we mark with red the frequencies that cannot be chosen, with purple the user's previous choices on this node (in case the user had selected to edit his reservation) and with blue the previous user's choices on this session, so that he does not have to select the same channels again and again for each node he reserves.

After that step, the user's choices are committed to the scheduler's database. This database contains information about the testbed topology, the available spectrum and, of course, all users reservations. Using PHP scripts and XML configuration files, this database can be automatically updated through the web site by the scheduler's administrators. Furthermore, the scheduler's web site can provide information to each user of the reservations he has made until now, so that he may see older preferences which fitted, check the exact reservation details when the time has come to execute the experiment, or anything else.

Finally, since the scheduler's user interface resides on the web server, while the testbed has another server, we have setup a secure communication channel, which is used by the scheduler to inform the testbed server's cron daemon to schedule necessary tasks for each experiment. Such tasks are unlocking the users accounts when the reservation starts and locking them when it ends and setting up firewall rules that prevent the users from trying to access nodes that are not assigned to them, by using applications others than OMF (for example secure shell).

4.3.2 OMF Components

Until now we have described the part of the scheduler which is focused to the experimenter and his choices at reservation. This, however, is not always enough. Mistakes can be made some times willingly, some times not; in any case, we need to ensure that the experimenters will stick on their choices and, even if they try, the system will not allow them to use any resources that they have not reserved.

In order to do that, we have chosen to extend OMF, which is a very popular managerial framework for wireless tesbeds. In Section 3, we gave a short description on OMF, how it is structured and the role of its components. Here, we give a detailed description of the additions and the extensions we had to make inside this framework to integrate spectrum slicing support in it.

Before anything else, we need a way for OMF and the scheduler's database to communicate. For this purpose, we have added one more service group to Gridservices named scheduler and we have added one more service to the inventory service group. Next, we are showing what these services are responsible for. First of all, the inventory service group is developed inside OMF and provides a set of webservices that provide general information about the testbed (such as node names, IP addresses, etc). This information is stored in a database residing on the testbed server and the inventory service group reads this database to return the proper response. Our addition here is a service which gets a node location (that is its coordinates) based on its IP address. Note here that the node location is a piece of information that is the same on both the scheduler's and the testbed's database and, thus, we can use it to do the matching. We have added this service, because when an experiment is executed, OMF does not know a node's location; only its IP address.

Now that scheduler knows the exact location of the node, it can use the scheduler service group to get any information needed from the scheduler's database. Namely, the services provided by this group provide functionality to get a node reservations based on its coordinates, the spectrum that this reservation contains and the user that owns it. Furthermore, it provides services that can do the matching between a channel or a frequency number and the respective spectrum identification number as it is stored in the database. All this information will be used by Nodeagent, which decides whether to allow the user use the channel or not.

So, Nodeagent is responsible for deciding whether the resources declared in the experiment should be allocated to the experimenter. In order to decide, the Nodeagent has to ask the scheduler's database if the specified resources have been reserved by the experimenter. So, when the experiment sets the wireless card channel, this information is passed to the Nodeagent, which now knows the channel along with its own IP address. All he needs is the user identification to check with the scheduler's database if this channel (and, of course, node) should be allocated to that user.

However, this is not straightforward, since the user usually logs into the node as root (keep in mind that the experiment loads his own image to the nodes, so he has full privileges on them). So, we need to track where did he use the

username that he also used for registering. The scheduler is designed in such a manner that, when a user registers to the system, then an account with the same username and password is automatically created to the testbed's server. The user uses this account to both access the user interface and the testbed server (using secure shell connection). This can solve our problem, since we can say for sure that the user that is running the experiment is logged into the console with the same username that he has made his reservation.

This information, though, relies on the testbed server, while the Nodeagent runs on the client side; that is the nodes. We need to pass that information from the server to the clients. This is done by the Nodehandler, the OMF service that is running on the server side and is responsible for controlling the experiment execution. Using its built-in message passing mechanism, Nodehandler tells the Nodeagent the username of the experimenter and now the last one has almost everything he needs to do the matching, except the date. The system should not rely on the experimenter to keep the clock of his clients coordinated with the testbed. This is why, Nodehandler sends, along with the username, the date at that time and the Nodeagent adjusts its clock to match the server's.

At this point, Nodeagent has all the information needed to check with the scheduler if the requested resources should be allocated to the experimenter. Using the web services we described above, the Nodeagent checks if there is a reservation at that time for that user and if the spectrum reserved at this reservation matches the channel that the experimenter has requested to assign to the network card through his experiment.

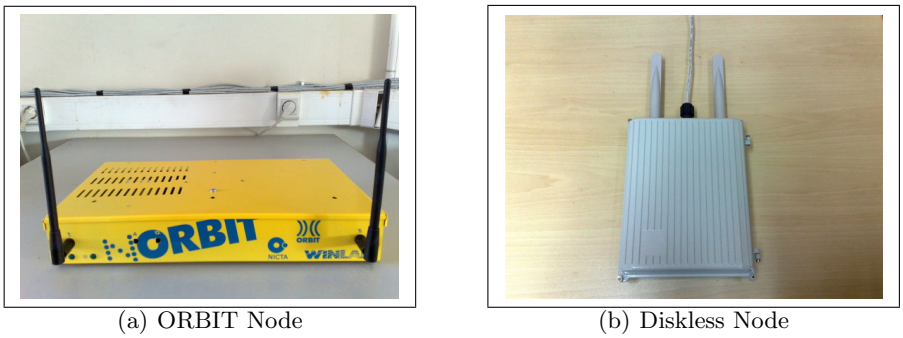
If all data match, then the Nodeagent lets the experiment execution move on. Otherwise, it notifies the Nodehandler that a resource violation has taken place and stops its execution (without assigning the channel to the node's network card). When the Nodehandler receives that message, the execution is terminated immediately and an ERROR message is thrown back to the experimenter describing the resource violation.

5 Slicing in Action - Usage Statistics

5.1 Testbed Description

The testbed that we used for design and deployment of our scheme is consisted of 10 ORBIT-like nodes, as depicted in Figure 3(a) and 5 Diskless nodes, as shown in Figure 3(b). An ORBIT-line node consists of a 1GHz VIA C3 processor, 512MB of RAM, 40GB of hard disk, two ethernet ports and two miniPCI slots which are used to host two Atheros WiFi cards. Our diskless nodes consist of a 500 MHz AMD Geode LX800 CPU, 256MB of RAM, a 1GB Flash Memory Card, two ethernet LAN ports and two Atheros wireless cards.

All the nodes are connected through wired Ethernet with the testbed's server - *console*. In console we have all the required testbed services running. These services are both network services, such as Dynamic Host Configuration Protocol (DHCP) server which gives IP address to the nodes, Domain Name System



(a) ORBIT Node (b) Diskless Node

Fig. 3. NITOS Nodes

(DNS) server which gives names to the nodes, Network File System (NFS) server for experiment results and slicing support as we are going to see next, and testbed services which are combined to the functionality of OMF.

We also maintain a web server where we keep the web interface of our system's scheduler. On this server, we also keep some scripts mandatory for remotely booking the nodes and a MySQL server for keeping records of the testbed status at each slot. Finally, we have set up a secure communication line, using Secure Shell (SSH) and a RSA key between the web server and console so that the scripts on the web server trigger the respective scripts on console.

After the user books some nodes at a specific time on the testbed, he logs into console at that time and from there, he can start using the testbed. The image is loaded on each node from console through the wired Ethernet interface. More information about our testbed's architecture can be found at our web site.

Although we have built this scheme on our testbed, it could also be applied to any wireless testbed which is using OMF as its management framework.

5.2 Experiment Execution Scenario

Here, we give an experiment execution scenario on our testbed, in order to illustrate the scheduler's usage and importance. In this scenario, we have Bob and Alice to be our experimenters, who have made their reservations and want to use our testbed, NITOS.

In Figure 4 we are showing a snapshot of NITOS execution on October 10, 2010 at 11:05 AM. At that time, there are two users whose experiments are about to be loaded: Bob and Alice. As we can see, both our users have reserved resources through NITOS Web Server and these reservations are kept into the Scheduler DB. Bob's reservation begins at 10:00:00 AM, ends at 12:00:00 AM and includes channels 7 and 8. Similarly, Alice's reservation begins at 10:30:00 AM, ends at 11:30:00 AM and includes channels 2 and 3.

Now, based on these reservations, Bob and Alice are trying to execute an experiment, so they log into NITOS server. At this point, we should mention, that since we allow experimenters to log into our testbed server, we need to be

very careful with security. This is why, we alter our firewall rules, which are tailor made on each user, so that he would not be able to access any of the testbed resources he has not already reserved (e.g. in our case, Bob cannot use a node that belongs to Alice or to neither of two).

After they have logged into the server, Bob and Alice ask OMF to execute their experiments. In his experiment description, Bob is setting the channel of his nodes to 7 and Alice to 8. Based on each experiment, the Nodehandler notifies accordingly the Nodeagent that is running on the nodes, which rely on a University building. When the Nodeagent is asked to set the channel of Bob's nodes, it checks Bob's reservation on Scheduler DB to see if channel 7 is included. Indeed, channel 7 is included in his reservation and the experiment execution continues normally. Now, when the Nodeagent is asked to set the channel of Alice's nodes, it checks Alice's reservation on Scheduler DB and sees that Alice has reserved channels 2 and 3 and not channel 8. In this case, the Nodeagent does not set the channel, terminates this session and sends an ERROR code back to the Nodehandler which indicates that there has been a channel allocation problem with Alice's experiment. The Nodehandler terminates the experiment execution and notifies Alice of the problem.

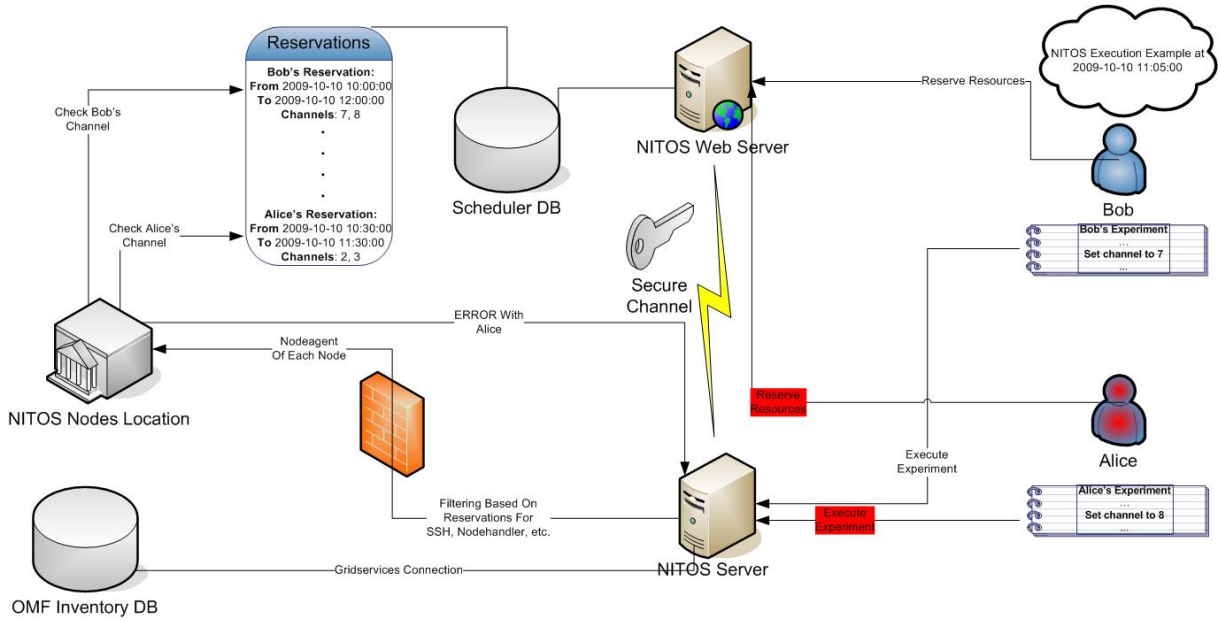


Fig. 4. Bob and Alice Experiments Execution

5.3 Usage Statistics

To outline slicing benefits, we have monitored testbed usage for a period of 5 months. We have added logging support to our scheduler as follows: The user is firstly prompted to enter the number of nodes that are needed by his/her experiment without being able to see which testbed nodes are occupied. If the system has enough resources to satisfy the request, the user may continue with the standard allocation procedure. Otherwise, the scheduler informs the user that the required amount of nodes cannot be allocated. With this approach we were able to log the allocation requests which were denied. Note that in the standard scheduler interface, the user is provided with enough visual information to determine whether the required number of nodes is available or not and avoids issuing requests which would be denied.

During these 5 months, the testbed use was approximately 500 hours. During these hours a total of 1008 requests were issued. To determine overall testbed utilization, we logged for each hour during which the testbed was in use and allocation requests were denied, the number of nodes that were not occupied. More specifically, we regard as the testbed idle time unit the idle hour of a single node. If for example 5 nodes are idle during a testbed usage hour this amounts to 5 idle hours. Since we have 15 nodes, the available usage time units of the logging period were 7500 ($TestbedUsageHours * NumberOfNodes$). To compare slicing with the simple allocation scheme that cannot allocate wireless frequencies, we developed a simulator. We should note here, that our testbed topology, in terms of physical wireless range, forms 2 independent neighborhoods. The first neighborhood has 7 nodes and the second 8. Therefore,the simple node allocation scheme can assign each node neighborhood independently and host up to two testbed users concurrently. Our simulator implements this policy, replays the allocation requests that have been logged for the 5-month period, determines the number of denied requests and testbed utilization time of the simple allocation scheme. In Figure 5 we depict the number of denied requests along with the testbed total idle time that we logged for slicing and simulated for simple allocation scheme.

6 Conclusions

As wireless networking research emerges, the respective testbed infrastructures and management systems should employ more sophisticated approaches to distribute available resources. While many of the management concepts that have been introduced for wired testbeds, have been extended and reused by wireless testbed management frameworks, the latter face an additional important challenge: the distribution and management of the wireless bandwidth in terms of frequency channels, which along with the node topology and connectivity range can become a very complicated task. In this work we attempted to address this issues.

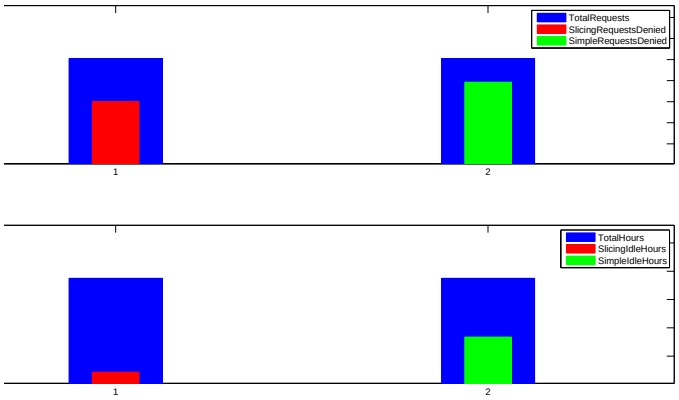


Fig. 5. Slicing performance for a 5-month period, compared to the simple allocation policy simulator results for exactly the same workload.

7 Future Work

In the domain of scheduling experiments on wireless testbeds, there still is much work that has to be done. Since we are expecting a growth to the testbeds usage by the researchers in the near future, we should put our efforts in developing a scheduling scheme, which will be able to allocate resources to experimenters efficiently, while, in the same time, it will be providing a transparent mechanism for executing the experiments.

First of all, we are working on integrating TLQAP in our scheduler. The scheduler can use TLQAP to extract information about the status of the testbed resources at any time needed. With this information, it is in place to match the experimenters demands on power, spectrum, location, etc. with the resources and schedule the experiment whenever they are available, without having the experimenter himself to check for their availability at each slot.

Furthermore, we are working on implementing other slicing schemes, which will depend on the transmission power control, the sharing of wireless network cards and the sharing of nodes themselves. For instance, we may adjust the power that a node will transmit based on the experiment characteristics and, thus, create a smaller neighborhood where the experiment will take place, while the rest of the testbed will stay available to other users. With network cards sharing, we plan to let two users make use of a node which has two wireless cards on, by assigning one card to each user. Finally, nodes sharing will give us the power to run multiple experiments on different images on the same for multiple users. We are expecting that this last scheme, in combination with spectrum and power slicing and wireless cards sharing, will give us a large scale improvement to the testbed utilization.

Wireless testbeds federation is another step that we are planning to take. The additions made for spectrum slicing in OMF services, provide us a very good tool for this end. We are thinking federation in two aspects: that of experiment execution and of experiment scheduling. Our work focuses on satisfying both these aspects using web services, which can be used as the tool to remotely invoke resource, get information, etc. However, with federation, we have other issues arising too, such as security, since we are dealing with resource allocation through a public network (Internet).

References

1. cfg80211 - Linux Wireless. `http://wireless.kernel.org/en/developers/Documentation/cfg80211`.
2. CRDA - Linux Wireless. `http://wireless.kernel.org/en/developers/Regulatory/CRDA`.
3. OMF Developer Portal. `http://omf.mytestbed.net`.
4. OneLab2. `http://www.onelab.eu/`.
5. UTH NITLab. `http://nitlab.inf.uth.gr`.
6. WINLAB - Rutgers University. `http://winlab.rutgers.edu`.
7. Wireless-Extensions - Linux Wireless. `http://wireless.kernel.org/en/developers/Documentation/Wireless-Extensions`.
8. Angelos-Christos Anadiotis, Apostolos Apostolaras, Dimitris Syrivelis, Thanasis Korakis, Leandros Tassiulas, Luis R. Rodriguez, Ivan Seskar, Maximilian Ott. A Demonstration of a Slicing Scheme for Efficient Use of Testbed's Resources. Demo - Mobicom 2009, September 2009.
9. Angelos-Christos Anadiotis, Apostolos Apostolaras, Dimitris Syrivelis, Thanasis Korakis, Leandros Tassiulas, Luis R. Rodriguez, Maximilian Ott. A New Slicing Scheme for Efficient Use of Wireless Testbeds. In *Proceedings of the 4th ACM international workshop on Experimental evaluation and characterization*, 2009.
10. Brent N. Chun, Philip Buonadona, Alvin AuYoung, Chaki Ng, David C. Parkes, Jeffrey Schneidman, Alex C. Snoeren, Amin Vehdat. Mirage: A Microeconomic Resource Allocation System for Sensornet Testbeds. In *Proceedings of the 2nd IEEE workshop on Embedded Networked Sensors*, 2005.
11. D. Raychaudhuri, I. Seskar, M. Ott, S. Ganu, K. Ramachandran, H. Kremo, R. Siracusa, H. Liu, M. Singh. "Overview of the ORBIT Radio Grid Testbed for Evaluation of Next-Generation Wireless Network Protocols".
12. Dimitris Syrivelis, Angelos-Christos Anadiotis, Apostolos Apostolaras, Thanasis Korakis, Leandros Tassiulas. Tlqap: A topology and link quality assessment protocol for efficient node allocation on wireless testbeds. In *Proceedings of the 4th ACM international workshop on Experimental evaluation and characterization*, 2009.
13. Mike Hibler, Robert Ricci, Leigh Stoller, Jonathon Duerig, Shashi Guruprasad, Tim Stack, Kirk Webb, Jay Lepreau. Large-scale Virtualization in the Emulab Network Testbed. In *USENIX 2008 Annual Technical Conference on Annual Technical Conference*, 2008.
14. R. Mahindra, G. D. Bhanage, G. Hadjichristofi, I. Seskar, D. Raychaudhuri, Y.Y. Zhang. Space Versus Time Separation For Wireless Virtualization On An Indoor Grid. In *Next Generation Internet Networks*, 2008.
15. J. Tourrilhes. Wireless Extensions for Linux. `http://www.hpl.hp.com/personal/Jean_Tourrilhes/Linux/Linux.Wireless.Extensions.html`.