Achieving Efficient Resource Allocation on Non-RF Isolated Wireless Testbed Deployments

Apostolos Apostolaras[1], Dimitris Syrivelis[1], Thanasis Korakis[1], Leandros Tassiulas[1]
[1]*Centre for Research & Technology, Hellas*
Email:{ apaposto, jsyr, korakis, leandros}@inf.uth.gr

Abstract:
The building of functional wireless networking system prototypes and the deployment of the respective large scale experiments have become a standard research practice. To that end, wireless testbed support has a significant role: to provide users with an efficient environment that will simplify development and evaluation of observed results. Contrary to wired network media testbeds, wireless deployments face very important challenges, that derive from the unpredicted behaviour of the wireless connectivity which can affect seriously experimental results. In this paper we describe our approach towards the dissemination of wireless bandwidth on our non-RF isolated testbed. Our system is composed of two basic subsystems: i) An existing OSI layer 2 protocol that assesses Topology and Link Quality and ii) A scheduling service that performs fine-grained distribution of wireless bandwidth and prevents user experiments from interfering with each other.

Keywords: Wireless Testbed, Management framework

1. Introduction

Managing and distributing a collection of wire interconnected computers to multiple users with guaranteed resource availability, like the number of dedicated processors, memory and network bandwidth, has a number of challenges which have already been addressed by wired media managing frameworks. To deploy an experiment, the wireless testbed user will need to allocate nodes that satisfy certain topology and link quality requirements. Moreover, the knowledge of the maximum possible throughput that each connection can achieve, allows the user to properly evaluate the observed experiment performance. In some wireless testbed deployments [1] were all the nodes are in an RF isolated room and are tightly located to avoid connectivity range problems, the allocation of frequency channel, also known as slicing, can be performed in a static manner. In these setups, wireless channels and bandwidth can be distributed as any other resource, like processors or memory and the user must not be able to change them.

While the described wireless testbed deployments are appropriate for a wide range of networking system experiments, there is also a need for testbed deployments that are closer to an end user setup. In such deployments there might be indoor and outdoor nodes with wireless links of varying quality, where some of them might not be able to directly communicate. Moreover, the testbed might not be RF isolated, e.g deployed on a campus site, where several other wireless terminals, out of the testbed context, compete for channel use. Nevertheless, for a certain range of experiments, that usually belong in wireless mesh network research, the user can take advantage of the described testbed setups to evaluate networking systems (e.g. adhoc routing protocols, hidden terminal solutions).

Regarding resource reservation, until now, the user 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. 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 user demands. To address these issues we have build a mechanism that uses spectrum slicing techniques but, still, not all allocation problems have been solved. The user does not apriori know which nodes will satisfy the experiment needs. Most experiments will require certain topologies and link qualities between the nodes. A complete system will interact with the user, gather experiment requirements, find the nodes that satisfy them and proceed with the slicing allocation. In this work we have merged our scheduler with Topology and Link Quality Assessment Protocol (TLQAP) that is described in [2] to provide this support.

The proposed allocation scheme has been developed as a part of a more generic managerial framework that is being designed in the concept of OneLab2 [3]. OneLab2 intends to federate heterogeneous testbeds located in different places under a unified system.

2. Related Work

Several work has been made on efficient resource allocation on wireless testbeds. For instance, Emulab [4] is a network testbed, that provides researchers with a wide range of environments to develop, debug, and evaluate their systems. Emulab virtualizes hosts, routers and networks, while retaining near total application transparency.

Mirage [5] is a resource allocation system, which was designed for sensor network testbeds and it is based on an auction scheme. The users are bidders, who argue for resources, using a virtual currency issued by the central system.

Our approach is more generic, because as a part of the OMF framework it is platform independent. Moreover, an analysis on virtualization schemes can be also found in [6], however in this paper, we are actually implementing the spectrum slicing scheme, which has proven to be useful on our testbed.

3. Wireless Testbed Managerial Framework

We are using cOntrol and Management Framework (OMF) [7] as the managerial framework. Currently OMF is deployed on several testbeds around the globe, including ORBIT. Using a ruby-like experiment definition language, the user writes an abstract description of the experiment, stating which nodes to use and what to deploy on them. Providing full transparency to users, OMF is responsible for loading system images to the designated testbed nodes. Currently OMF consists of three basic components: *Gridservices*, *Nodehandler* and *Nodeagent*, where the first two run on the testbed server, while the third runs on each node. Next, we give a short description of each one of these components:

Gridservices. Gridservices consist of a set of web services, which are responsible for both executing system actions 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. Like Gridservices, Nodehandler runs on the testbed server too.

Nodeagent. Nodeagent runs on the testbed 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 executing received commands.

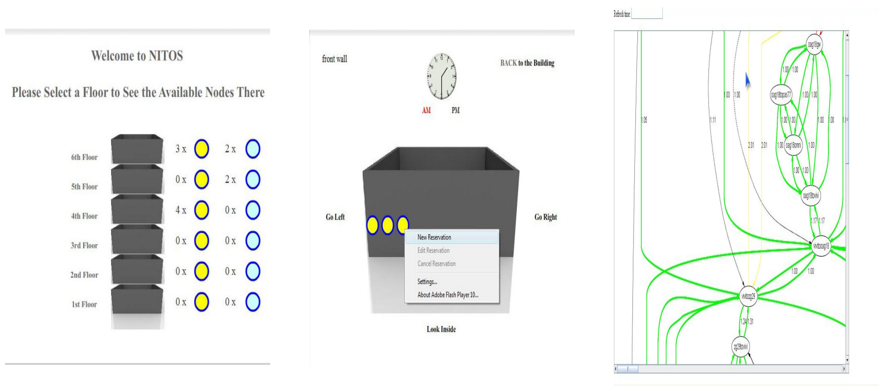
We had to extend all the three components above to integrate spectrum slicing support inside OMF. In the next section, we will show our basic idea for achieving spectrum slicing, the dilemmas, the decisions we had to make and the integration with TLQAP.

4. Scheduling Experiments on Wireless Testbeds

Currently OMF does not include any scheduling algorithms that could facilitate concurrent experiment execution. 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 users 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 users to reserve nodes and spectrum for some time (which should not exceed a specified limit). Using spectrum slicing, our tool makes the testbed available to users who would like to use different resources at the same time. Moreover the users may observe the available connectivity between the nodes, before allocating them.

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 user during scheduling or dynamically assigned. Spectrum slicing is combined with TLQAP connectivity scheme. The user can see, before proceeding with the reservation of the current node, the connectivity graph from the perspective of this node. This graph has been dynamically generated from the TLQAP database using the DOT tools.



(a) Testbed deployment overview. (b) Selection of particular testbed node. (c) TLQAP node connectivity from the perspective of the current node .

Figure 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. The user 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.

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 user 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 user a location-based perspective of the nodes that they are reserving for their experiments. This provides the users with an initial hint about node connectivity. Obviously, co-located nodes are expected to have good link qualities while this is highly unlikely for distant nodes. Of course TLQAP data will clarify the connectivity attributes.

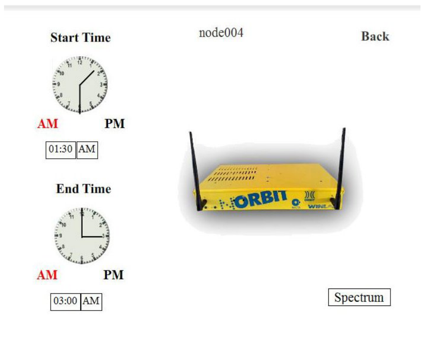
The registered user logs in to the scheduler's web site and interacts with the User Interface. Firstly he chooses the required date. Then, the actual scheduling process begins. Our testbed building is depicted with an indication beside each floor on the number of each node type that reside on that floor, as shown in Figure 1(a).

Based on these location picture, 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). Immediately then the user is provided with the TLQAP connectivity graph 1(c) which is constructed from the perspective of the current node. At this point, the user may decide which other nodes he may need to allocate based on their connectivity with the current node. The TLQAP graph depicts node connectivity as well as the largest rate which achieved a Packet Delivery Ratio equal to 1 during the TLQAP experiments.

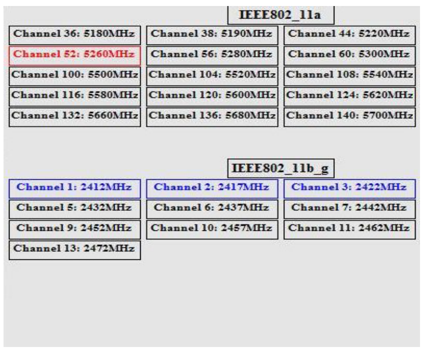
To avoid having users attempting to reserve concurrently the same nodes we have a clock on each frame, which can be clicked by the user on the time he would like to check the nodes status. By clicking there, the frame is automatically refreshed and the nodes are colored according to their status, while at the same time, the new user cannot request a new reservation on that node for that time slot.

After the user has selected his node, and seen connectivity with others, he can proceed with the actual reservation. Our front-end provides user with two clocks, one for choosing the start time for his experiment and one for the end time (see Figure 2(a)).

Guided by the scheduler, the user 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 user to choose a channel that is reserved by another one during that time. This is the final step; The user may repeat the process to reserve other nodes. Of course, the user may come back later and check out all his reservations, grouped by the reservation time.



(a) Time reservation of particular node.



IEEE802_11a

Channel 36: 5180MHz	Channel 38: 5190MHz	Channel 44: 5220MHz
Channel 52: 5260MHz	Channel 56: 5280MHz	Channel 60: 5300MHz
Channel 100: 5500MHz	Channel 104: 5520MHz	Channel 108: 5540MHz
Channel 116: 5580MHz	Channel 120: 5600MHz	Channel 124: 5620MHz
Channel 132: 5660MHz	Channel 136: 5680MHz	Channel 140: 5700MHz

IEEE802_11b_g

Channel 1: 2412MHz	Channel 2: 2417MHz	Channel 3: 2422MHz
Channel 5: 2432MHz	Channel 6: 2437MHz	Channel 7: 2442MHz
Channel 9: 2452MHz	Channel 10: 2457MHz	Channel 11: 2462MHz
Channel 13: 2472MHz		

(b) Spectrum Selection.

Figure 2: NITOS Scheduler Resource Reservation

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 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.

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 users reserve the resources they need (in terms of nodes and spectrum) in an efficient way for the testbed usage. The main scheduling application consists of a flash animation which uses multiple PHP scripts and XML files to give the user the required information. The scheduler depicts the testbed topology dynamically by reading XML files. When the testbed deployment changes the XML files are informed and all changes are propagated to the scheduler front-end. The scheduler subsystem that needs to periodically perform necessary tasks for each experiment relies on Cron daemon. 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.

The OMF part of our scheduler system aims at confining the user experiment deployment to the slicing selections which were made during reservation time. The user may make wireless configuration mistakes, so we need to ensure that the users 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 Section 3., we gave a short description on OMF, how it is structured and the role of its components. Here, we provide a detailed description of the extensions we had to make inside this framework to integrate spectrum slicing support.

Firstly OMF and the scheduler's database must communicate. For this purpose, we have added one more service group to Gridservices named scheduler and we have added the respective service to the inventory service group. First of all, the inventory service group is developed inside OMF and provides a set of webservices that give general information about the testbed (node names, IP addresses, etc). This information is stored in a database which the inventory service group reads to respond. Our extension here is a service which gets a node location based on its IP address. We have added this service, because when an experiment is executed, OMF does not know a nodes' location; only its IP address.

Now that scheduler can use the service group to get any information needed from the its 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 perform matching between a channel or a frequency number and the respective stored spectrum identification number.

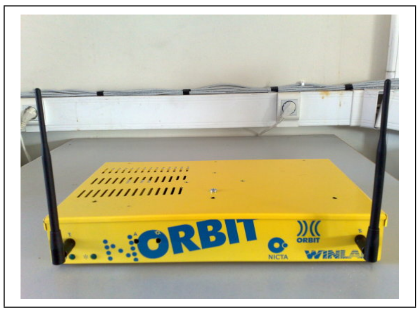
Nodeagent is responsible for deciding whether the resources declared in the experiment should be allocated to the user. In order to decide, the Nodeagent has to ask the scheduler's database if the specified resources have been reserved by the user. Firstly, Nodeagent will have to wait until the wireless network card on the node is configured. At that time, the experiment defines a channel to be used by the network card. So, at this point, Nodeagent knows the channel and its own IP address. All he needs is the user identification to check with the scheduler's database if this channel 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.

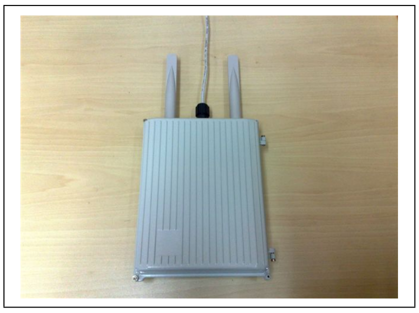
This information, though, relies on the testbed server, while the Nodeagent runs on the nodes. We need to pass that information from the server to the clients. This is done by the Nodehandler, the OMF service that is running on the server side and is responsible for controlling the experiment execution. Using its built-in message passing mechanism, Nodehandler tells the Nodeagent the username. For security reasons, Nodehandler sends, along with the username, the date at that time and the Nodeagent adjusts its clock to match the servers'. Nodeagent has all the information needed to check with the scheduler if the requested resources should be allocated to the user. 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 requested channel. 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

5. Overview of NITOS Testbed

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. Our diskless nodes consist of a 500 MHz AMD Geode LX800 CPU, 256MB of RAM, a 1GB Flash Memory Card, two ethernet LAN ports. Both types host 2 Atheros 5213 WiFi cards (a total of 30 interfaces is available).



(a) ORBIT Node



(b) Diskless Node

Figure 3: NITOS Nodes

All the nodes are connected through wired Ethernet with the testbed's server - *console*. In console we have all the required testbed services running. We also maintain a web services where we keep the web interface of our system's scheduler. Finally, we run a MySQL server for keeping records of the testbed status and TLQAP data. 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.

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 these issues.

References

[1] "WINLAB - Rutgers University." http://winlab.rutgers.edu.

[2] 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.

[3] "OneLab2." http://www.onelab.eu/.

[4] Mike Hibler, Robert Ricci, Leigh Stoller, Jonathon Duerig, Shashi Guruprasad, Tim Stack, Kirk Webb, Jay Lepreau, "Large-scale Virtualization in the Emulab Network Testbed."

[5] 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."

[6] 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."

[7] "OMF Developer Portal." http://omf.mytestbed.net.