# Integration of OMF-Based Testbeds in a Global-Scale Networking Facility

Giovanni Di Stasi, Stefano Avallone, and Roberto Canonico

Università di Napoli Federico II, Dipartimento di Informatica e Sistemistica,
Via Claudio 21, 80125 Naples, Italy
{giovanni.distasi,stefano.avallone,roberto.canonico}@unina.it

**Abstract.** PlanetLab is a global scale platform for experimentation of new networking applications in a real environment. It consists of several nodes, offered by academic institutions or companies spread all over the world, that can be shared by the networking community for its tests. The main drawback of PlanetLab is its scarce heterogeneity in terms of the access technologies it offers. In this paper we discuss the efforts we made in order to alleviate this problem. We first developed a tool that allowed us to integrate a WiFi testbed controllable by OMF (Orbit Management Framework) [16] in PlanetLab by means of a multi-homed PlanetLab node. OMF is a set of tools that make it easy to automatically execute experiments and collect measurements on a WiFi testbed. The tool we developed allows, more generally, to solve the issues that arises with multi-homed PlanetLab nodes (i.e. PlanetLab nodes having more than a network interface). In order to be able to fully exploit the potential of such PlanetLab nodes, there is the need for the users to add routing rules (e.g. rules to reach a destination through the WiFi interface, instead of the Ethernet interface). Such operation cannot be performed in a PlanetLab environment, as the rules a user adds would also affect other users' traffic. Therefore it arises the necessity of user-specific routing tables, i.e. routing tables whose rules are only valid for traffic belonging to that user. In this way the user is able to route his traffic through the WiFi interface, and make it traverse the OMF-controllable WiFi testbed, while other users' traffic continues to get routed through the default primary interface. We also had to support the integration of the OMF facilities (e.g. the OMF controller) into the user environment, which is called slice, in order to allow for the customization of the testbed (e.g. loading a specific disk image on each node) and the automatical execution of experiments. The software we developed to achieve such integration is in the process of being integrated in the code base of PlanetLab, so that anyone is able to integrate its wireless infrastructure in PlanetLab.

**Keywords:** heterogeoneus networks, network design, experimentation, measurement, performance.

## 1 Introduction

In recent times various attempts to enhance the heterogeneity of PlanetLab have been made. Above all, we mention OneLab, an European Project funded by

N. Bartolini et al. (Eds.): QShine/AAA-IDEA 2009, LNICST 22, pp. 545–555, 2009.

the European Commission in its Sixth Framework Programme [2]. The project started in September 2006 with “two overarching objectives: (1) to extend the current PlanetLab infrastructure and (2) to create an autonomous PlanetLab Europe”. PlanetLab Europe is a European-wide research testbed that is linked to the global PlanetLab through a peer-to-peer federation [4]. During this project different kinds of access technologies (such as UMTS [5], WiMax [9] and WiFi) were integrated, allowing the installation of new kinds of multi-homed PlanetLab nodes (e.g. nodes with an Ethernet plus a interface). The users, however, had not been provided with a tool that allowed them to set the kernel routes required to use these new access technologies. Hence the user could not choose which interface to use to reach a given destination, limiting his experimental possibilities (e.g. the possibility to route one flow using the WiMax interface, while using the Ethernet interface for the remaining traffic). UMTS interfaces in PlanetaLab did not suffer from this limitation, as the software that manages them has the ability to set for which destinations the UMTS interface is required [12]. One of the contribution of this work has been to generalize that software, allowing it work with any kind of network interface and support different operations. Another attempt to add more heterogeneity in PlanetLab is in [15]. In order to integrate an OMF-testbed in PlanetLab, the authors propose the use of a gateway PlanetLab node, whose function is to open tunnels between itself and the selected nodes in the OMF testbed. Differently from our approach, the gateway node is not a client of the OMF testbed, but creates the tunnels. A similar approach was taken in [13]. The authors aimed at integrating the VINI virtual network infrastructure [11] with OMF-based testbeds.

The rest of the paper is structured as follows. Section 2 discusses the integration of PlanetLab nodes and an OMF testbed. Section 3 provides details of our implementation, with particular emphasis on the sliceip tool. Section 4 illustrates the testbed we used for our experimentations, while section 5 presents the results of some experiments we carried out to test our solution. Finally, Section 6 concludes this paper.

## 2 Integration Scenario

The PlanetLab node in our integration scenario is a multi-homed node featuring an Ethernet and a WiFi interface. The WiFi interface is used to access the OMF testbed, which is set to work as an access network (i.e. it is connected to the Internet). The Ethernet interface is mainly used for control traffic (e.g. the traffic for accessing the node), but can also be used for experiments. The PlanetLab node is equipped with the normal PlanetLab software, with the following additions: 1) the tool we developed, called *sliceip*; 2) the *OMF NodeHandler* and 3) a locking script, i.e. a script used to lock the OMF testbed. The tool *sliceip* has the function to allow the user add slice-specific routing rules. By adding these rules, the user is able to select which access network to use for his experiment. The rules are slice-specific in the sense that they are valid only for the traffic belonging to user’s slice. The *OMF NodeHandler* is a component of OMF that is used to setup the OMF testbed. It takes as input an experiment description and executes the required operations (e.g. loading a given disk image on a node, starting a given application on a node, etc.) by contacting the *OMF NodeHandler Agents* (simply OMF NodeAgents) installed on every node of the OMF testbed. As previously stated, the OMF testbed has to work as access network, so it has to have a node connected to the Internet and set as gateway. This gateway node, in addition to routing functions, has to do natting, as the ip addresses used in the OMF testbed are private. The OMF testbed at this stage of development can be accessed on an exclusive way. The user, in order to be able to use the OMF testbed, has to acquire a lock, by using the locking script we provide. This lock ensures that only that user can setup the OMF testbed and route his traffic through it. After acquiring the lock, the user can setup the OMF testbed by means of the *OMF NodeHandler*, and then perform the experiment. The locking mechanism is accomplished by allowing only packets belonging to the slice that locked the testbed reach the OMF control network and the WiFi interface. In future, the limitation of one user at a time will be removed[1]. This will be achieved by allowing different users work on two different subsets of the OMF testbed nodes and by employing different subset of non-interfering channels.

## 3 Implementations Details

As we have seen in the previous section, three main components are required to enable integration between PlanetLab and OMF-based testbed. The first two, *sliceip* and the *locking script* are installed in the root context of the node (i.e. the privileged context) and have counterparts, called frontends, in the slice-context (i.e. the unprivileged user context). The frontends communicate with the tools installed in the root context by means of named pipes created by a component of PlanetLab called vsys [8]. The frontend writes in one of the two pipes, and what it has written is received by the backend, which checks the input to see if it is valid and starts the requested operation. Once the operation is completed, the backend writes in the second pipe the results, which are received by the frontend and written to the standard output for the user. The *OMF NodeHandler* does not require root privileges to run, so it can be installed by the user and run in the unprivileged slice context.

### 3.1 The Sliceip Tool

*sliceip* is the tool we developed in order to enable slice-specific routing tables in PlanetLab nodes. Using this tool, the user is able to define routing rules which apply only to traffic of his slice. This is required for the user to be able to choose which interface to use for his experiment. In particular, *sliceip* enables slice-specific routing tables by leveraging a feature of the Linux kernel and a feature of the VNET+ subsystem [6] of PlanetLab. The Linux kernel has the

[1] OMF developers are already working on removing this limitation.

ability to define up to 255 routing tables. To have some traffic routed following the rules of a particular routing table, it is necessary to associate that traffic to it by means of rules applied with *iproute2*. The rules specify packets in terms of the destination address, the netfilter mark, etc. In our case, we set the netfilter mark of packets belonging to the user's slice (i.e. the packets that are generated or are going to be received by an application running on that slice) by exploiting a feature of the VNET+ subsystem of PlanetLab. We ask this subsystem, by means of an *iptables*[1] rule[2], to set the netfilter mark equal to the id of the slice (i.e. a numeric value that identifies the slice on that node) to which they belong. We then add a rule with *iproute2* to associate the packets which belong to the slice with the routing table allocated for that slice. We also set an *iptables* SNAT rule (Source Network Address Translation) in order to set the source IP addresses of packets that are going out through a non-primary interface (the primary interface is the one the default routing rule points to). This rule is required because the source ip address of packets is set after the *first routing process* happens. In fact, in case multiple routing tables are used, the routing process follows the following steps: 1) the interface for sending the packets is selected following the rules of the main routing table and the source IP address is set accordingly (this is the first routing process); 2) if the user changes the netfilter mark of the packets in the *mangle chain* of *iptables* and a rule is defined for routing those packets with a different routing table, a *rerouting process* is triggered. This rerouting process follows the rules of the selected (i.e. the slice-specific) routing table and the interface to be used is set accordingly; 3) the packet is sent out using the selected interface. During the step 2, the source IP address is left unchanged, so we need to change it explicitly before the packets are sent during the step 3.

The user interacts with *sliceip* by means of a frontend that resides in the slice. This frontend extends the syntax of the *ip* command of the *iproute2* suite with the following two commands:

- *enable <interface>*: initialize the routing table for the user's slice, add the rule to set the netfilter mark of packets to the user's slice id, add a rule to associate those packets with the routing table of the slice and add the SNAT rule for interface;
- *disable <interface>*: remove the SNAT rule for interface, remove the rule to associate the packets of the slice to the respective routing table, and remove the rule that sets the netfilter mark of packets.

## 4 WILE-E Testbed

We deployed a prototype testbed called WILE-E (WireLEss Experimental) in order to show a real case of our integration strategy (see Figure 1). It consists of: i) 3 Soekris net4826-48 Single Board Computers; ii) 3 business-class access point Netgear WG302Uv1; iii) 1 Linux machine acting as gateway towards the

[2] The *–copy-xid* PlanetLab extension to *iptables*.

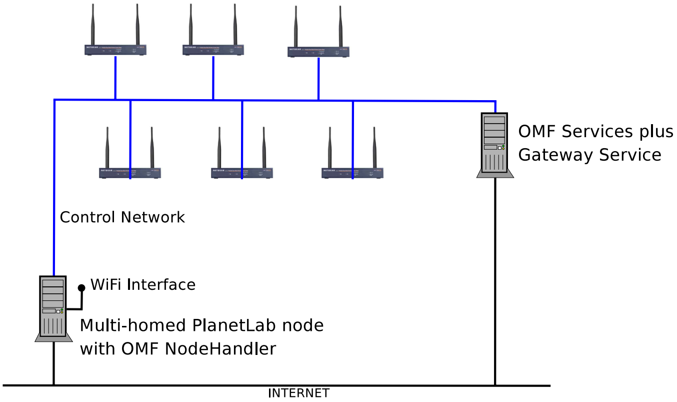

**Fig. 1.** WILE-E architecture

Internet for the testbed; iv) 1 PlanetLab node, through which researchers can access the Wireless Mesh testbed to run their own experiments. The PlanetLab node is equipped with two Ethernet interfaces (one with a public IP address and another connected to the internal OMF control network) and with a WiFi 802.11a/g interface to perform experiments that involve the WILE-E testbed. This PlanetLab node is associated to a private PlanetLab deployment, so we have root access for it (needed for some experiments we will show afterwards).

The Soekris net4826-50 SBC is based on an AMD Geode SC1100 CPU (at 266Mhz), has 128Mbyte DRAM memory, a 128Mbyte Flash disk, a FastEthernet interface and two 802.11a/g Atheros wireless cards. The Netgear WG302Uv1 access point is based on an Intel XScale IXP422B network processor (at 266Mhz), has 32Mbyte DRAM memory, a 16Mbyte Flash disk, a FastEthernet interface and two 802.11a/g Atheros wireless cards. We had to create two baseline images, one for each kind of device. Baseline images are the disk images the user can load on the nodes. The first image is meant to be used with the Soekris SBC. It is based on the Voyage Linux distribution [7] (v. 0.5), a distribution for embedded devices derived from Debian. This image provides two kernel images: a 2.6.20 vanilla Linux kernel and a 2.6.19 Linux kernel patched to support Click Modular Router [14]. The latter kernel can be useful for experimenters that need to run software routers constructed with the Click framework. An OMF baseline image needs to have some OMF software components which provide an interface to the OMF NodeHandler and to the OMF services for controlling the node. These components are: i) *the OMF NodeAgent*, that is the software entity, which performs local operations, such as setting the channel of the wireless interface channel, starting an application, and so on; ii) the *OMF Traffic Generator* (OTG), that is used to generate traffic for experiments; iii) the *OMF Measurement Library* (OML), a shared library used by OTG and, optionally, other user's application to send traffic traces to the *OML server daemon*, i.e. the OMF service whose function is to store the traffic traces. The second baseline image is for the Netgear access points. It is based on OpenWrt [3] (Kamikaze version), a Linux distribution for embedded devices. OpenWrt supports a large number of devices and has a great community of users. In order to install the OMF components on this image, we had to create the port of some of the OMF components to OpenWrt: the NodeAgent, OTG and the OML library. We plan to submit these packages to the OpenWrt repository in order to ease for other users the process of deploying OMF testbeds that comprise nodes supported by OpenWrt.

## 5 Proof-of-Concept Experiments

In the following we describe some proof-of-concept experiments in order to practically demonstrate the usage of the WILE-E testbed and to characterize the behavior of the sliceip tool.

The first experiment involves the WILE-E testbed and a remote PlanetLab Europe node, located at INRIA (Figure 2. In this experiment, two end-to-end flows are generated from the multi-homed PlanetLab node that gives access to the WILE-E testbed to the remote host.

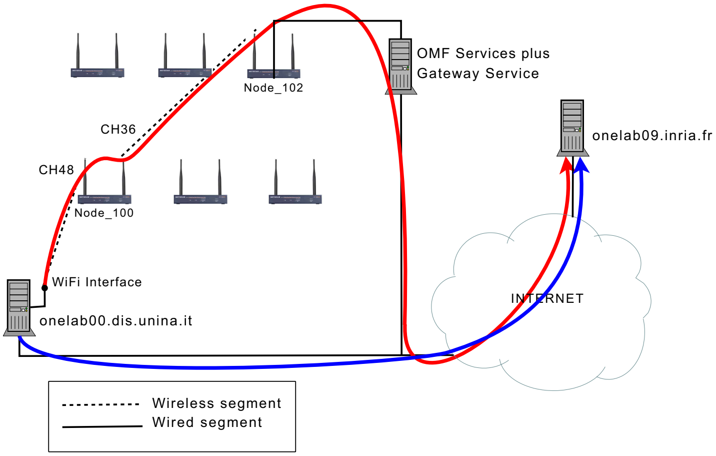

**Fig. 2.** An experiment with the WILE-E testbed

The experiment went through the following steps:

1. nodes onelab00.dis.unina.it and onelab09.inria.fr were added to slice
2. the OMF testbed was set up using the *OMF Nodehandler*; in particular, one of the interfaces of Node_100 was set in *Master mode* to make this node behave as an access point for the PlanetLab node;
3. a first flow was generated as a CBR flow of fixed size UDP packets; as the default routing rule of the PlanetLab node pointed to the Ethernet interface,

the packets went out through the Ethernet interface, and followed the blue path;

4. a slice-specific routing rule was added to onelab00.dis.unina.it to forward packets with destination onelab09.inria.fr through Node_100 of the Wireless Testbed;
5. a second flow was generated (with the same parameters of the previous flow); this traffic traversed the Wireless Mesh testbed and reached the Internet through the *Gateway Service* (along the red path);
6. in order to verify experiment isolation (i.e. if users of other slices were affected by the routing rule we added with `sliceip`), we generated the same flow in a second slice, and verified that this latter flow followed the blue path.

We used D-ITG (Distributed Internet Traffic Generator) [10], a platform capable to accurately generate traffic flows specified through two random processes: packet Inter Departure Time (IDT) – the time between the transmission of two consecutive packets – and Packet Size (PS) – the amount of data being transferred by the packets. Both processes are modeled as i.i.d. series of random variables, whose distribution can be selected by the user among a rich set of supported ones (constant, uniform, exponential, pareto, normal, cauchy, etc). D-ITG also incorporates some of the models proposed in the literature for the IDT an PS of the most well-known application protocols. D-ITG enables to evaluate a set of QoS performance metrics such as throughput, packet loss, delay (One Way Delay and Round Trip Time) and jitter.

The setup of the WILE-E testbed was performed by using the OMF facilities. The setup is described by the following script (interpreted by the OMF NodeHandler).

```
defGroup('gateway', [8,102]) {|node|
  node.net.w0.mode="adhoc"
  node.net.w0.type='a'
  node.net.w0.channel="36"
  node.net.w0.essid="meshnet"
  node.net.w0.ip="192.168.6.3"
  node.net.w0.netmask="255.255.255.0"
}

defGroup('ap', [8,100]) {|node|
  node.net.w0.mode="master"
  node.net.w0.type='a'
  node.net.w0.channel="48"
  node.net.w0.essid="meshnet-ap"
  node.net.w0.ip="192.168.7.1"
  node.net.w0.netmask="255.255.255.0"

  node.net.w1.mode="adhoc"
  node.net.w1.type='a'
  node.net.w1.channel="36"
  node.net.w1.essid="meshnet"
  node.net.w1.ip="192.168.6.2"
  node.net.w1.netmask="255.255.255.0"
}

group("ap").exec('dnsmasq',['--dhcp-range\
=192.168.7.2,192.168.7.254,255.255.255.0,infinite'])

group("ap").exec('route',['add','-host',\
'143.225.229.236','gw','192.168.6.3','metric','1000',\
'ath1'])

group("gateway").exec('ifconfig',['eth0:1',\
'192.168.10.102'])

group("gateway").exec('route',['add','-net',\
'192.168.7.0','netmask','255.255.255.0','gw',\
'192.168.6.2','metric','100','ath0'])

group("gateway").exec('route',['add','-host',\
'143.225.229.236','gw','192.168.10.200','metric',\
'1000','eth0'])

whenAllInstalled() {|node|
        wait 60
        allGroups.startApplications()
        wait 30
        STDIN.gets
        Experiment.done

}
```

### 5.1 Overhead Analysis

The aim of this experiment is to analyze the overhead introduced by our slice-specific routing mechanism.

The slice-specific routing mechanism requires, in respect to the standard routing mechanism, the execution of some extra steps: as we previously mentioned, the netfilter mark of packets has to be set to the slice-id value and different routing table (i.e. one for each slice that requests it) need to be handled.

To evaluate the overhead, we generated two end-to-end flows between two PlanetLab hosts. The source node belongs to a private PlanetLab environment (onelab00.dis.unina.it) and is equipped with two Ethernet interfaces; the sink node to PlanetLab Europe (planetlab01.dis.unina.it). The flows were two CBR (constant bitrare) TCP packet flows at 30Mbit/s and were generated both in the slice context of the source node.

Before generating the first flow, we inserted a rule in the root context, in order to make the flow go out through the second Ethernet interface. We generated the flow, collected the log file of the transmission on the sink node and removed the routing rule. Then we inserted the same routing rule in the slice context, by using sliceip. After the flow ended we collected the log on the sink node. In Figure 3, we compare the bitrate obtained for the two flows. The results for both flows are in average very close, showing that the overhead introduced is negligible.

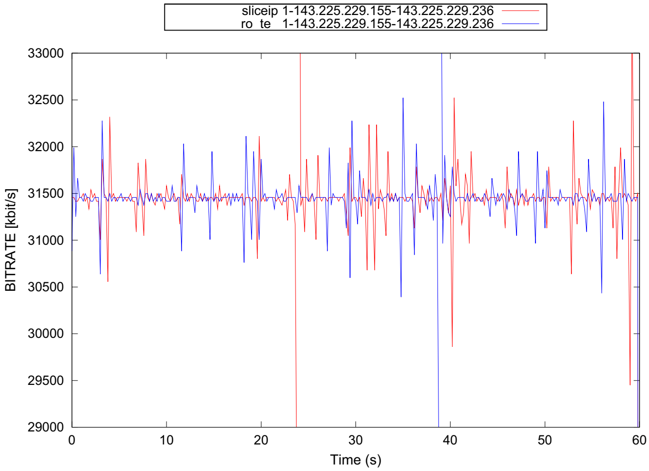

**Fig. 3.** Sliceip - route comparison

## 5.2 Redundant Routing in a Slice

This experiment shows how to exploit sliceip and the OMF WiFi testbed to perform experiments on routing redundancy with PlanetLab nodes. In the experiment we exploit the additional WiFi interface of the node to provide Internet coverage through the OMF testbed when the primary Ethernet interface becomes unavailable. This can be done by using sliceip, with the addition of a script needed to monitor the state of the Internet connection on the primary Ethernet interface. All of this is done inside the user's slice, without affecting experiments in other slices.

The redundant routing mechanism is implemented in this way: a monitoring script continuously checks if a "test host"', i.e. an Internet host chosen by the user for testing the connection, is reachable; as soon as it notices that it is not reachable anymore, the script adds a routing rule to reach the destinations selected by the user - i.e. the destinations for which the user wants to guarantee the redundant routing - through the WiFi interface. The WiFi interface is then used to reach those destinations until the script notices that the main interface is working again.

When the primary Ethernet interface fails, users with open shells on the slice will lose control of the node. This is a problem our mechanism does not address (as the control traffic is bound to the main interface). Nonetheless, the batch experiments running towards the selected destinations will continue to work.

In Figure 4 the results of an experiment of routing redundancy is shown. The graph shows the bitrate of two end-to-end transmissions between two PlanetLab hosts (the same nodes of the previous experiment). The first transmission is performed in a normal PlanetLab slice, while the second in a slice where *sliceip* is installed and the monitoring script is run. During both transmission a fault of the Ethernet interface is simulated (by shutting down the interface). As you can see, the first transmission, the one in the slice where no routing redundancy is implemented, stops working, while the second only stops for a short amount of time, after which it shows a lower bitrate. The short interruption is due to the fact that it takes some time for the script to recognize the fail of the Internet connection and some time for the new routing rule to become active. The lower bitrate is then due to the fact that the flow is traversing the OMF testbed.

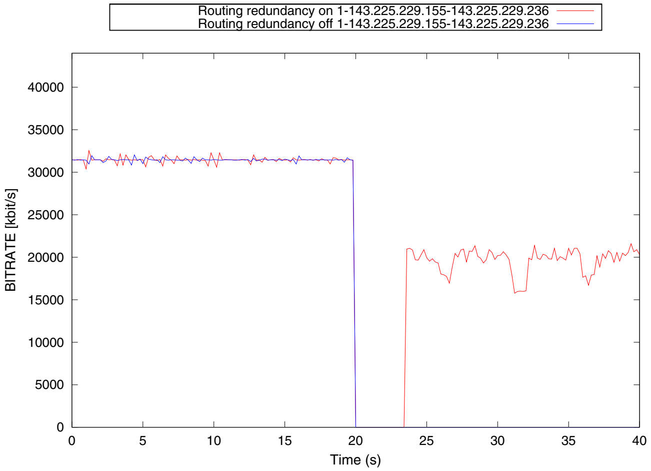

**Fig. 4.** Redundant routing experiment

## 6 Conclusions

In this paper we showed how we managed to integrate an OMF-controllable WiFi testbed in Planetlab.

This was achieved thanks to the development of two tools: sliceip, which allows PlanetLab users define slice-specific routing table, and the locking script, which allows PlanetLab users lock the OMF testbed for exclusive use. The main scope of the slice-specific routing table is to allow the user route his traffic through the OMF testbed, using it as an access network. It is, however, not only useful in this circumstance, but for all the cases where the user needs to control its own routing table. We showed, on this regard, the use of sliceip to perform a redundant routing experiment.

The locking script has the scope to discipline the access of the OMF testbed among the multiple concurrent experiments running in a PlanetLab node.

## Acknowledgments

Research outlined in this paper has been partially supported by the European Union under the ONELAB2 Project FP7-224263.

## References

1. Netfilter/IPtables, http://www.netfilter.org
2. Onelab, http://www.onelab.eu/
3. Openwrt, http://openwrt.org/
4. Planetlab europe - federation, http://www.planet-lab.eu/federation
5. UMTS Forum, http://www.umts-forum.org/
6. VNET+ subsystem of PlanetLab, http://www.cs.princeton.edu/~sapanb/vnet/
7. Voyage Linux, http://linux.voyage.hk/
8. vsys, http://www.cs.princeton.edu/sapanb/vsys/
9. WiMax, http://ieee802.org/16/
10. Avallone, S., Emma, D., Pescapè, A., Ventre, G.: High performance internet traffic generators. The Journal of Supercomputing 35(1), 5–26 (2006)
11. Bavier, A., Feamster, N., Huang, M., Peterson, L., Rexford, J.: In VINI veritas: realistic and controlled network experimentation. In: Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications, pp. 3–14. ACM, New York (2006)
12. Botta, A., Canonico, R., Stasi, G.D., Pescapé, A., Ventre, G.: Providing UMTS connectivity to PlanetLab nodes. In: Proceedings of the 3rd International Workshop on Real Overlays & Distributed Systems, ROADS 2008, Madrid, Spain (December 2008)
13. Hadjichristofi, G., Brender, A., Gruteser, M., Mahindra, R., Seskar, I.: A wired-wireless testbed architecture for network layer experimentation based on ORBIT and VINI. In: Proceedings of the second ACM international workshop on Wireless network testbeds, experimental evaluation and characterization, pp. 83–90. ACM, New York (2007)
14. Kohler, E., Morris, R., Chen, B., Jannotti, J., Kaashoek, M.: The click modular router. ACM Transactions on Computer Systems (TOCS) 18(3), 263–297 (2000)
15. Mahindra, R., Bhanage, G., Hadjichristofi, G., Ganu, S., Kamat, P., Seskar, I., Raychaudhuri, D.: Integration of heterogeneous networking testbeds (2008)
16. Ott, M., Seskar, I., Siraccusa, R., Singh, M.: Orbit testbed software architecture: Supporting experiments as a service. In: TRIDENTCOM 2005: Proceedings of the First International Conference on Testbeds and Research Infrastructures for the DEvelopment of NeTworks and COMmunities (TRIDENTCOM 2005), Washington, DC, USA, pp. 136–145. IEEE Computer Society, Los Alamitos (2005)