Integration Of Heterogeneous Networking Testbeds

R. Mahindra[†‡] G. Bhanage[†] G. Hadjichristofi[†§] S. Ganu[†¶]
P. Kamat[†‖] I. Seskar[†] D. Raychaudhuri[†]
[†]WINLAB, Rutgers University, NJ Technology Center, North Brunswick, NJ, USA
[‡]NEC Laboratories America, NJ, USA
[§]Cyprus University,Cyprus
[¶]Aruba Networks, CA, USA [‖]ASK.com, NJ, USA

ABSTRACT

As networking research expands into new frontiers, the research community has felt a need for a heterogeneous networking research infrastructure to experiment with the interaction and integration of different types of networks, and to test the performance of various networking protocols in realistic environments. This requirement has led to the Global Environment for Network Innovations (GENI) initiative to create a global infrastructure for conducting networking experiments across diverse substrates such as wired (local and wide-area), wireless, sensor and cellular networks. In this paper, we discuss and present two models for building such an experimental infrastructure. The first model enables a wired testbed to link with wireless edge nodes during an experiment, whereas the second model enables a wireless testbed to link to wired testbeds. Proof-of-concept experiments are also presented reinforcing the usefulness of the models in terms of facilitating experiments over the integrated heterogeneous infrastructure.

1. INTRODUCTION

The GENI Project [1] aims to provide a flexible and programmable shared experimental infrastructure for the investigation of future internet protocols and software. As explained in the project development plan [2], GENI will consist of a global-scale wired network along with several wireless access network deployments intended to support experimentation with mobile computing devices, embedded sensors, radio routers, etc. This research finds solutions for an important technical issue related to the integration of wireless networks into GENI, namely the integration of control and management across wired and wireless networks, through the provision of a single programming interface and experimental methodology.

The importance of the integration of control and management across wired and wireless experimental networks, is highlighted in [3]. While PlanetLab [4] serves as the baseline model for programming and virtualization in wired GENI, the model needs to be significantly extended to accommodate the full range of envisioned usage. Specific extensions to be considered include: a broader range of experiment types (e.g., short-term network performance experiments running on selected network nodes vs. long-term slices used in PlanetLab), and alternative end-user support requirements Our research, aimed at integrating PlanetLab with ORBIT [8], a large-scale wireless testbed, may yield to important design insights on the issues of necessary extensions to control and management protocols for effective support of wireless networks as an integral part of the experimental system.

A related aspect of this integration is the ability to carry multiple concurrent experiments within the integrated platform. This capability can be achieved through network virtualization. Unlike the wired network, virtualization of wireless network elements is fundamentally a difficult problem because of the broadcast nature of the wireless medium. This paper covers the integration framework that has been developed and also demonstrates proof-of-type integrated experiments that use Frequency Division Multiplexing (FDMA) and Virtual MAC (VMAC) as forms of virtualization on the ORBIT side. Contributions of this paper are to:
1. Consider representative wired-wireless testbed and outline an integrated framework with two approaches (PlanetLab driven and ORBIT driven) supporting different flavors of end-user requirements and
2. Address the problem of supporting multiple concurrent experiments over these substrates and provide proof of concept experiments conducted using the framework.

The rest of the paper is structured as follows. Section 2 describes the aspects to be considered while integrating heterogeneous wired and wireless testbeds. Section 3 talks about the representative wired and wireless testbeds considered for integration. Detailed discussion of the integration models is in Section 4. Results from proof of concept experiments are shown in Section 5 followed by our conclusions.

2. INTEGRATION OF WIRED AND WIRELESS EXPERIMENTATION NETWORKS

To support realistic and large-scale experimentation with new network architectures and distributed systems an integrated testbed framework would have to be based on a very flexible design that will enable a variety of network architectures, services, and applications to be evaluated. To accomplish these goals, we may need to move far beyond existing testbeds and experimental facilities, yet the best way to do

Tridentcom 2008, March 18, 2008, Innsbruck, Austria.

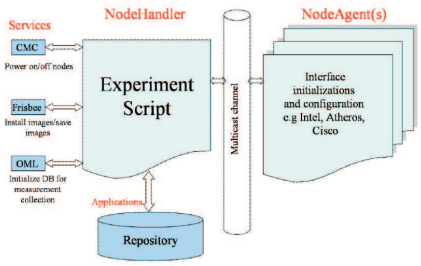


Figure 1: Software architecture overview with the ORBIT radio grid.

this would be to leverage (and synthesize) a wide range of ideas and techniques that have been developed in isolation on individual testbeds. Specifically, GENI will adopt PlanetLab's model of virtualizing the available resources, thereby allowing multiple experiments to run in isolated slices. In addition, GENI will need to support the following:
1. An inclusive model to enable researchers to use the facility to run their experiments e.g., controlled short-term reproducible experiments and long−running deployment studies
2. Wireless network virtualization to support multiple concurrent experiments on the integrated testbed.

In each of these dimensions, there is a lot to learn from other experimental testbeds, including Emulab [5], ORBIT [8], and DETER [7]. In particular, we propose to integrate PlanetLab with ORBIT. Both ORBIT and Planetlab were designed to meet very different experimental research requirements. The next section will walk through some of the fundamental differences in the design of these testbeds that should be considered while building the integration models.

3. OVERVIEW OF THE ORBIT AND PLANETLAB TESTBEDS

The major differences between ORBIT and PlanetLab testbeds considered in this paper are as follows:

Experiment life cycle -While Planetlab is based on a long-running service oriented experimentation model based on the concept of distributed virtualization of resources, the ORBIT Radio Grid is a multi-user wireless experimental research testbed that allows "sequential" short-term access to the radio grid resources for repeatable experimentation. Scheduling is done so that users have exclusive access to the grid during their assigned time slot. We propose virtualization solutions for the ORBIT testbed to facilitate the integrated framework to support multiple concurrent experiments.

Experiment models - The duration of experiments on the ORBIT testbed is short-lived, as opposed to Planetlab's services-oriented model, which supports experiment durations on the order of months. To resolve this issue, we propose the usage of a long-running ORBIT slice on the PlanetLab nodes.

Experimental Control and Management Framework - The overall architecture of the ORBIT testbed is to provide a multi-user wireless experimental facility and it is designed to accommodate as many users as possible. In most of the experiments, setting up the experiment and collecting results of the experiments and collating them usually is a significant contributor to the overall experiment time. Hence, the design goal is to reduce this setup time and to simplify data collection as much as possible. In the ORBIT framework, the experiment controller is called the *nodeHandler* and the corresponding client side software residing on the nodes that responds to commands from the *nodeHandler* is the *nodeAgent*. The experiment is specified in the form of a Ruby script and is disseminated over multicast to the nodes involved in the experiment (see Figure 1). The *nodeHandler* also interacts with other support services to initialize the environment prior to the actual experiment. These tasks may include powering up the relevant nodes, installing custom images on the nodes, if needed, and setting up the databases for measurement collection.
Even though PlanetLab provides a globally distributed substrate for conducting geographically diverse experiments, it does not provide an official experimental software framework to facilitate the deployment. The testbed does not offer any built-in support for choreographing an experiment and controlling all the nodes using automated scripts.

Considering the above factors, the following two models are presented in the next section that have been developed to integrate these testbeds and conduct joint experiments.

4. INTEGRATION MODELS

The first model (PDIE) is intended to serve PlanetLab users who want to extend their experiments to include wireless networks at the edge without changing the PlanetLab interface, while the second model (ODIE) is intended to serve ORBIT wireless network experimenters who want to augment their experiments by adding wired network features without major changes to their code.

4.1 PlanetLab Driven Integrated Experimentation (PDIE) Model

The *PDIE* model shown in Figure 2 provides a PlanetLab-ORBIT gateway as a node that PlanetLab users can access when they want to include emulated wireless edge networks in their experiments. The PlanetLab-ORBIT gateway provides abstractions for the setup, control and measurement on a specified topology using a modified version of the *nodeHandler* as the interface software. This approach did not involve major changes to either the PlanetLab or ORBIT testbeds, but requires the development of a PlanetLab proxy module.

Figure 3 describes the current design with the PDIE integration model. The PlanetLab-ORBIT gateway machine, which acts as a proxy, is running as a part of the ORBIT framework. This gateway communicates with PlanetLab nodes via GRE tunnels[6]. One GRE tunnel is setup for every selected PlanetLab node. Packets received from different experiments or slices on PlanetLab nodes are redirected to the corresponding ORBIT nodes. This functionality is achieved by having GRE tunnels from the gateway to the ORBIT nodes. On the PlanetLab side, experiments 1, 2, and 3 are running in different slivers on each PlanetLab node. On the ORBIT side one or more ORBIT nodes are linked to the PlanetLab slivers of each experiment.

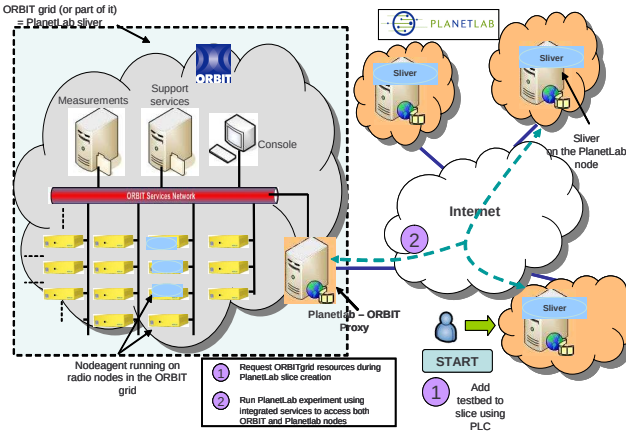


Figure 2: Outline of the PlanetLab driven integrated experimentation(PDIE) model where PlanetLAB users get scheduled access to chosen nodes on the ORBIT grid using the concept of an ORBIT sliver.

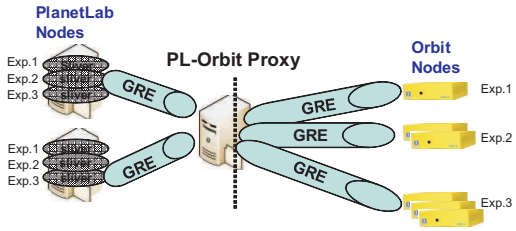


Figure 3: A sample approach based on the PDIE model. The figure shows the use of a PlanetLab - ORBIT proxy for mapping PLanetLab slivers to ORBIT nodes.

4.2 ORBIT Driven Integrated Experimentation (ODIE) Model

The ODIE approach allows users to include a long-running "ORBIT slice" in Planetlab nodes in their experiments with a single experimental script. This model is similar to that proposed in [9], where the authors described the integration of the Emulab [5] and PlanetLab testbeds to provide Emulab users with an access to PlanetLab resources. Figure 4 presents a conceptual view of the ODIE model.

The ODIE model has been implemented by extending the ORBIT *nodeAgent* functionality to work in the PlanetLab node "ORBIT Slivers". A modified version of the ORBIT *nodeHandler* was developed to support experiment definition, code download, and execution. This *nodeHandler* communicates with modified *nodeAgents* running on the PlanetLab slivers. The topology selection for the experiment is done by manual addition, where experimenters choose the PlanetLab nodes individually.

In order to communicate with nodes on the local subnet (ORBIT nodes) as well as remote PlanetLab nodes, we extend the naming/addressing scheme and communication protocol for the *nodeHandler-nodeAgent* Framework to allow access to geographically diverse nodes.

Extended addressing scheme: We address the Planet-

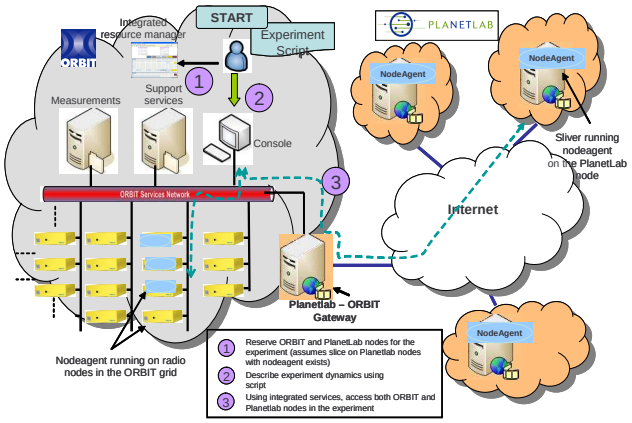


Figure 4: Outline of the ORBIT Driven Integrated Experimentation (ODIE) model where ORBIT users can add PlanetLab nodes to their experiments using the concept of an ORBIT slice.

Lab nodes as though they were part of the ORBIT network and have the local DNS map requests for Planetlab nodes to their respective public domain names. In ORBIT, all nodes are addressed as x,y where x is the row number (1..20) and y is the column number (1..20). Presently we have 20 PlanetLab nodes in the ORBIT Slice addressed as [21,1..20].(for e.g. *node21-3.orbit-lab.org* will map to *planetlab-01.cs.washington.edu*).

Extended communication layer: Currently in an ORBIT experiment, commands sent to the ORBIT nodes from the *nodeHandler* use reliable multicast. For Planetlab nodes on the Internet, these commands needed to be tunneled using reliable unicast since multicast support on the routers in a path cannot be assumed. The *nodeHandler* has been modified to communicate with the *nodeAgents* on the PlanetLab nodes over unicast-TCP. The *nodeHandler* communicates with the PlanetLab nodes in each experiment that requires wired networking resources.

The sequence of communications during an experiment is as follows: When an experiment is started, the *nodeHandler* starts the *nodeAgents*on the specified PlanetLab nodes and waits for them to report back. After a timeout it records all the PlanetLab nodes that have successfully reported back. The nodes that fail to report during the timeout period are replaced by other PlanetLab nodes in the ORBIT Slice. This procedure is repeated till the desired number of PlanetLab nodes have reported back. (A failure could result from node failure, node maintenance, slice clean-up, link failure etc.). The next step for the *nodeHandler* is to send commands to the *nodeAgents* to start the necessary applications on the PlanetLab nodes. The *nodeAgents* then report success or error messages back to the *nodeHandler* indicating the status of the nodes. After setting up the PlanetLab nodes, the *nodeHandler* configures and sets up the ORBIT nodes. The user simply provides a unified experiment script including both PlanetLab and ORBIT nodes and the application definition which the *nodeHandler* parses to execute the experiment automatically.

Experiment Scripting: The ODIE based experiment

```
#-------------ORBIT nodes-----------#
defNodes('AccessPoint',[11,20]){|node|
node.prototype("test:proto:mvlcrelay",
    {'duration' => prop.duration})
    node.net.w0.mode = "master"
    node.net.w0.essid = "link1"
    node.net.w0.ip="192.168.7.1"
}
defNodes('Client', [19,2]) {|node|
node.prototype("test:proto:mvlcdest",
    {'duration' => prop.duration})
    node.net.w0.essid = "link1"
    node.net.w0.ip="192.168.7.7"
}#----------PlanetLab nodes---------#
defPNodes('[21,3],[21,5]')
```

Figure 5: Node configuration section of a sample script (ODIE model).

```
#--Start applications on ORBIT nodes-#
whenAllInstalled() {|node|
    nodes('AccessPoint').startApplications
    nodes('Client').startApplications
    wait 195  # Experiment Duration
    allNodes.stopApplications
    Experiment.done
}
#--Start applications on PlanetLAB nodes--#
WhenPLReady(){
   defPApplication([21,3],[21,5],'VIDEO'){}
   wait 195
   defPApplication([21,3],[21,5],'STOP'){}
PLexpdone() }
```

Figure 6: Experiment execution section of a sample script (ODIE model).

script is parsed and executed by the *nodeHandler* to choreograph the experiments. A single script for the ODIE models may be described in two sections: (A) Node configuration section (B) Experiment timing and execution section.

The node configuration section is responsible for setting up all the nodes being used as a part of the integrated experiment while the timing and execution section of the ODIE script describes the execution sequence of the experiment. Figure 5 shows the section of the script that configures the nodes for the experiment. The first part of the code configures the wireless interfaces of two ORBIT nodes; one as an access point and the other as a client. The configuration part also defines the PlanetLab nodes in Washington and Georgia to include in the experiment. Figure 6 describes the timing and execution section of a typical ODIE script. The module *WhenPlReady()* waits for the *nodeAgents* on the PlanetLab nodes to report. Once the desired number of PlanetLab nodes have reported, the applications defined in *defPApplication()* are initiated. The *Plexpdone()* module ensures the slice is cleaned up at the end of the experiment.

Since the *nodeHandler*/*nodeAgent* framework is based on Ruby (a highly portable, scripting language) and since both PlanetLab and ORBIT run different flavors of the same OS (Linux), the porting of *nodeAgent* on Planetlab slivers was relatively easy. PlanetLab does not provide an interface for efficient resource teardown mechanism upon experiment termination for each new experiment (every 1-2 hrs). This mechanism has been leveraged by the *nodeAgents* running on the PlanetLab nodes in the ORBIT slice.

4.3 Distinction between the PDIE and ODIE Models

The major motivation behind the two models is to give the users a similar interface to the integrated testbed as the individual testbeds. The PlanetLab users would find the PDIE model easier to conduct their integrated experiments while the ODIE model would suit the current ORBIT users. Besides this difference, there are two major features that set these models apart:

4.3.1 PlanetLab to ORBIT connectivity

The ORBIT nodes do not have public IPs. Hence, in the ODIE model, any sort of connectivity between ORBIT and PlanetLab has to be initiated in ORBIT. Probable solutions to this problem include setting up tunneling, VPN etc. However, in the PDIE model, the PlanetLab-ORBIT gateway provides the proxy for connectivity between the testbeds. Note that this gateway would also benefit from virtualization solutions that are developed for ORBIT, bridging the gap between the PlanetLab slice model and ORBIT's current single-user model.

4.3.2 Control and Management Framework

The ODIE model is an extension of the current ORBIT framework to the PlanetLab nodes. The PDIE model is based on the current PlanetLab working model and lacks an integrated framework. Therefore, the ODIE model will be preferred by protocol analysts who prefer tools for convenient and easy experimentation.

5. PROOF-OF-CONCEPT INTEGRATION WITH WIRELESS VIRTUALIZATION

In this section, we evaluate some experimental scenarios with integrated tests of the PlanetLab and ORBIT testbed. We also investigate the following virtualization approaches in our experiments.

1. Integrated experiments with frequency division multiplexing of ORBIT nodes.
2. Integrated experiments with MAC layer virtualization of ORBIT nodes.

Using the virtualized ORBIT testbed, a Planetlab slice could be extended to include individual ORBIT nodes. Additionally, virtualization improves the utilization of the resources and provides a scalable integrated framework to support multiple users and experiments concurrently on the limited resources. The experiments in this section do not aim at showing important research results, but rather the dexterity of the framework to perform integrated experiments.

5.1 Frequency based ORBIT slicing coupled with PlanetLab

Aim: In this proof-of-concept experiment we show the use of our architecture in testing the performance of video delivery algorithms.

Topology: Figure 7 shows the topology for this experiment. We consider a typical scenario for streaming video delivery across a network path that includes an edge wireless link. The experimentation includes two flows from PlanetLab nodes to two Access Points configured within ORBIT. The Access Points relay traffic to their respective clients over orthogonal channels. Isolating experiments on different and possibly orthogonal frequencies is one of the easiest approaches to wireless virtualization (FDMA). The video streamed from PlanetLab goes over the internet giving experimenters the characteristics of a realistic network. Physical and MAC layer parameters of the wireless link can also be varied. The video is streamed and played using the Video LAN (vlc) [14] player. The ORBIT Measurement Library (OML) framework of ORBIT provides means for recording the bit rate and jitter in the video received at the ORBIT clients. We record measurements for different PlanetLab nodes in terms of their geographical distance from ORBIT.

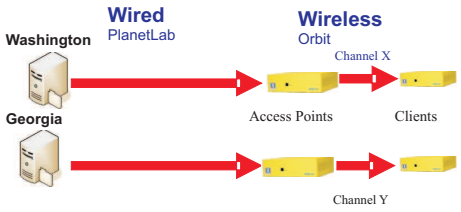


Figure 7: Experiment layout where nodes are added from PlanetLab while frequency seperation (FDMA) is used with the ORBIT nodes.

Jitter measurement: Figure 8 shows the results for the jitter values from the FDMA experiments for PlanetLab nodes in Princeton (NJ), Washington and Japan. Results show relatively comparable jitter values for the Princeton and Washington nodes. Japan on the other hand sees a higher jitter for video delivery possibly due to higher traffic and geographical distance. We also show the results of a heavily loaded server by adding traffic to the Washington node. We recorded the jitter values across certain baseline scenarios such as a single hop wired and wireless link for a comparison study.

Bit rate measurement: Figure 9 shows a plot of the video bit rate observed at the client as a function of time. The observed bit rate for the videos is lower for the PlanetLab node in Japan as compared to those in Princeton or Washington. Surprisingly the increased load on the Washington node only showed increased jitter in the video delivered with a comparable mean bit rate. However, the PlanetLab node in Japan

Experiment Setting	*Max Jitter (ms)*	*Mean Jitter (ms)*
Wireless One hop	1.68	0.98
Pl – Princeton	1.69	0.98
Pl – Washington	1.88	1.05
Pl – Washington (Loaded)	35.76	3.62
Pl – Japan	52.18	7.38

Figure 8: Jitter results observed with different PlanetLab nodes serving the same video over the internet to wireless clients in the ORBIT grid.

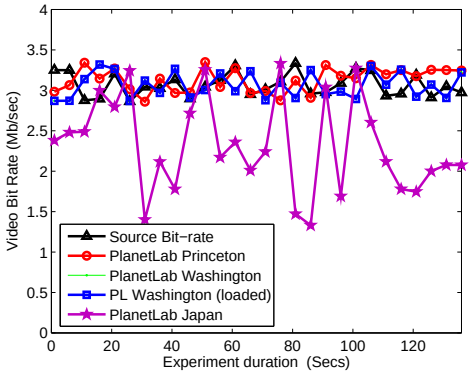


Figure 9: Observed bit rate with the same test video being delivered from different PlanetLab nodes.

has a deteriorated bit rate at the client due to the greater geographical distance.

These results could help evaluate proposed video delivery algorithms providing a deeper understanding of their operation. In addition, research on spectrum allocation, bandwidth management and inter- Access Point communication protocols would require the presence of a similar framework.

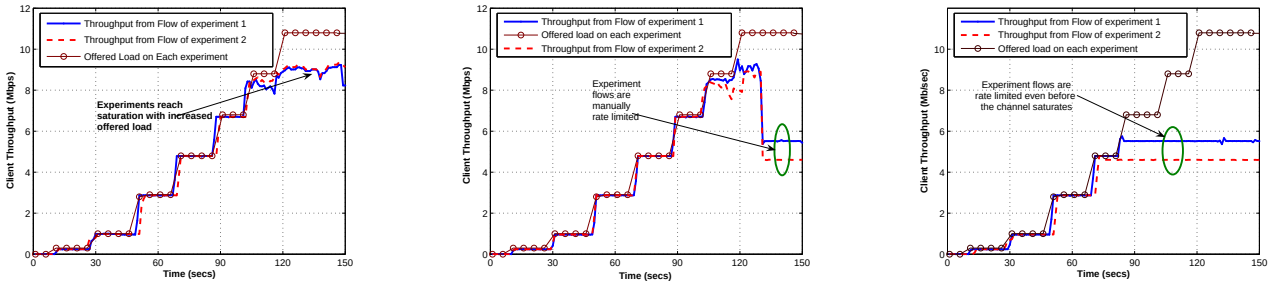
5.2 Virtual Access Point Based ORBIT slicing coupled with PlanetLAB

This proof-of-concept experiment uses virtualization at the MAC layer with the introduction of a VAP (Virtual Access Point). The VAP provides logical partitioning among the experiments based on ESSIDs [12]. In [10], the authors discuss in detail the feasibility of VAPs for wireless virtualization.

Aim : The goal of this experiment is to demonstrate a VAP-based approach used for virtualization. We also show degradation in performance with the use of VAP and possible solutions to mitigate them.

Topology and setup: Figure 10 shows our experimental setup. It consists of two UDP-CBR traffic flows belonging to two independent experiments which are being sent by servers running on PlanetLab nodes to their respective clients running on the ORBIT grid. The third flow is a video streaming from one of the Planetlab nodes to the clients running on the ORBIT grid. To setup this configuration of nodes an experiment script similar to the one shown in Figure 5 was used.

Figure 11(a) shows a plot of the two UDP traffic flows as seen at the receiver on the wireless client node. The offered load for each of these experiments is increased as a function of time for each of these experiments. As long as the aggregate offered load is below saturation, both flows have a fair share of the throughput. The video quality of the third experiment was found to be good as it produced a clear picture. As the offered load for each of the experiments is increased with time, the aggregate traffic on the wireless network reaches saturation. Figure 11(a) shows that in saturation the net throughput seen for both flows are comparable. However, the video flow suffers considerably when the wireless channel is in saturation. To prevent such situations where the



(a) No traffic shaping or policy management for bandwidth control.

(b) Manual intervention rate limits flows to stop channel saturation.

(c) Individual flow rates are predecided by the policy manager.

Figure 11: Throughput (Mbps) seen at the wireless client.

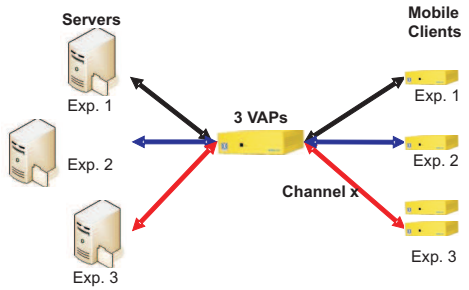


Figure 10: Experiment layout where nodes are added from PlanetLab while the VAP support from the 802.11 linux drivers is exploited for running multiple networks from a physical AP.

performance of one experiment affects the other we incorporate the use of traffic control with the experiments. Possible approaches to bandwidth control are: (1) Manual Intervention (2) Policy Manager based interference control.

Figure 11(b) shows a typical scenario with manual intervention in an experiment. Initially as the aggregate offered load is increased, the experiments are pushed into saturation. However, with manual intervention it is possible to rate limit the traffic flows. We make use of the Click Modular Router [13] as a tool to implement bandwidth shaping. It is also possible to have a policy manager to decide the maximum share of throughputs of these experiments before they are started. The policy manager could be integrated with the experiment scheduling and resource tracking mechanisms to ensure that each of the experiments get a fair share of the resources. Figure 11(c) shows the results with a policy manager. Both the experiments are rate limited to their assigned throughput values even before they reach channel saturation. The video quality was also found to be restored.

6. CONCLUSIONS

The unified designs presented in this paper could serve as a practical foundation for wired/wireless integration in future heterogeneous testbeds. We believe that the common control and management model, presented here, for a globally distributed networking infrastructure will lead to easier and faster experimentation and more efficient use of testbed resources. Our proof of concept experiments demonstrate the effectiveness in terms of the ease of experimental deployment and the overall usefulness of this integrated framework.

7. REFERENCES

[1] NSF Global Environment For Network Innovations (GENI). `http://www.geni.net/`

[2] GENI Design Principles in `http://www.geni.net/designprinciples.pdf`

[3] D. Raychaudhuri and M. Gerla, New architectures and disruptive technologies for the future internet in Report of NSF WMPG Workshop, August 2005.

[4] D. Culler L. Peterson, T. Anderson and T. Roscoe, "A blueprint for introducing disruptive technology into the internet," in HotNets-I 2002.

[5] L. Stoller R. Ricci S. Guruprasad M. Newbold "An integrated experimental environment for distributed systems and networks," in Proceedings of the 4th Symposium on Operating System Design and Implementation (OSDI 2002), 2002

[6] RFC3147, Generic Routing Encapsulation over CLNS Networks, IETF draft of the networking working group, July 2001

[7] DETER TestBed in `http://www.isi.edu/deter`

[8] D. Raychaudhuri, I. Seskar, M. Ott, S. Ganu, K. Ramachandran, H. Kremo, R. Siracusa, H. Liu, and M. Singh, Overview of the orbit radio grid testbed for evaluation of next-generation wireless network protocols in WCNC,2005

[9] Emulab-PlanetLab portal in Proceedings of the First Workshop on Real, Large Distributed Systems (WORLDS 2004),2004.

[10] R. Mahindra, G. Bhanage, G. Hadjicristofi, I. Seskar, D. Raychaudhuri, YY. Zhang Space Versus Time Separation For Wireless Virtualization On An Indoor Grid in proceedings for IEEE NGI 2008

[11] PlanetLab: An Open Platform For Developing, Deploying and Accessing Planetary-Scale Services in `https://www.planet-lab.org/`

[12] Creating virtual ap on madwifi `http://madwifi.org/wiki/UserDocs/MultipleInterfaces`

[13] Click Router `http://read.cs.ucla.edu/click/`

[14] Videolan Player in `http://www.videolan.org/vlc/`

[15] Manpage of tcpdump in `http://www.tcpdump.org/`

[16] IPERF, TCP/UDP Traffic Generation Tool, `http://dast.nlanr.net/Projects/Iperf/`