Architecture for federating heterogeneous networking testbeds

R. Mahindra*, G. D. Bhanage*, G. Hadjichristofi*†, S. Ganu*‡, I. Seskar*, D. Raychaudhuri*
*WINLAB, Rutgers University, Piscataway, NJ 08854,USA
†University Of Cyprus, Cyprus
‡ARUBA Networks

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 has led to the Global Environment for Network Innovations (GENI) effort, supported by NSF, which aims at creating 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 the challenges involved in federating such diverse networking testbeds 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.**

I. INTRODUCTION

The GENI Project [1] aims to provide a flexible and programmable shared experimental infrastructure for investigation of future Internet protocols and software. As explained in the project development plan [2], GENI will consist of a global-scale wired network with programmable and virtualizable network elements (routers, switches, servers) 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, was highlighted in [4]. While PlanetLab [5] 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: device heterogeneity (e.g. wireless access points, ad hoc radios, sensors), 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 (e.g. experienced programmers needing little if any experimental support vs. protocol analysts who might prefer tools for higher-level programming and execution management). The GENI Architecture Working Group has initiated discussion of a general control and management framework that covers all these needs, but it is clear that a parallel thread of proof-of-concept prototyping would be extremely valuable to build a practical understanding of integration issues and provide guidance to the design. Based on these considerations, we believe that it would be productive to initiate collaborative prototyping activities aimed at integrating PlanetLab with other large scale wired and wireless testbeds such as Emulab [6], Xbone [10], DETER [11] and ORBIT [12]. In particular, a project aimed at integrating PlanetLab with a large-scale wireless testbed like ORBIT may be expected to yield important design insights on the issues of device heterogeneity and 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 virtualization. While significant progress has been made with virtualization of wired network elements (such as access routers), work on wireless virtualization still remains at an early stage [3] with much more work to be done for practical realization of techniques that can be used in GENI. It is recognized that strict virtualization of wireless network elements is fundamentally a difficult problem because of the fact that radios interfere with other radios in their immediate vicinity via interactions at RF spectrum, physical (PHY) and medium-access control (MAC) layers. These interactions make it more difficult to create orthogonal time slices on radio channels than on wired network links, but wireless deployments in large systems like GENI do have two additional dimensions (frequency and space) that can be exploited to isolate simultaneous experiments. This paper covers the integration framework that has been developed and demonstrates integrated experiments that use Frequency Division Multiplexing (FDMA) and Virtual MAC (VMAC) as forms of virtualization. The proof-of-concept prototyping presented in this paper would be extremely valuable in building a practical understanding of integration issues and providing guidance to the design of GENI. Results from the prototyping are expected to feed into ongoing system engineering work aimed at specifying key technology components that will constitute GENI. Contributions of this paper can be summarized as:

- Consider representative wired-wireless testbed and outline an integrated framework with two approaches

(PlanetLab driven and ORBIT driven) for rapid prototyping and experimental deployment over heterogeneous substrates.
- 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 II talks about the aspects to be considered while integrating heterogeneous wired and wireless testbeds. Section III talks about the representative wired and wireless testbeds considered for integration. Section IV discusses about the differences in the architectures of the two testbeds. Detailed discussion of the integration models is in Section V. Results from proof of concept experiments are shown in Section VI followed by conclusions.

II. 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 very flexible design that will enable a variety of network architectures, services, and applications to be evaluated; experiments to be conducted under real-world workloads to be deployed in an incremental manner over a network that spans a wide range of representative networking technologies. To accomplish these goals, we need to move far beyond existing testbeds and experimental facilities, yet the best way to do 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. (A slice refers to the distributed resources bound to a particular experiment.) As a result, GENI will need to support:

- A wide range of heterogeneous network resources (e.g., wireless subnets, customizable routers)
- An inclusive model to enable researchers to use the facility to run their experiments e.g., controlled or reproducible experiments and long−running deployment studies)
- The ability to federate across resources contributed by multiple autonomous organizations (e.g., from other countries and from other research communities).

In each of these dimensions, there is a lot to learn from other experimental testbeds, including Emulab [6], ORBIT [12], DETER [11], and WAIL [13]. This section addresses how the above-mentioned dimensions can be supported enabling GENI to meet these objectives. In particular, we propose to integrate PlanetLab with ORBIT, and in the process, better understand how virtualized slices can be extended to accommodate both heterogeneous wireless network technologies and multiple experimental modalities. Both ORBIT and Planetlab were designed to meet very different experimental research

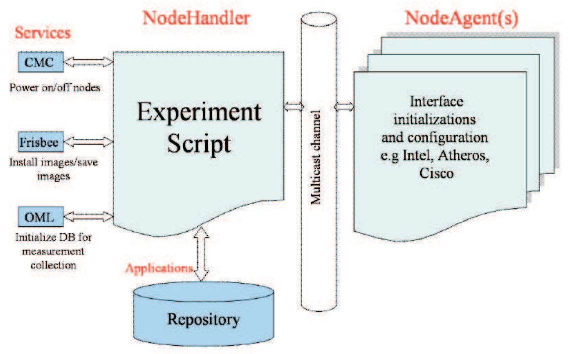

Services
NodeHandler
NodeAgent(s)
CMC
Power on/off nodes
Frisbee
Install images/save images
OML
Initialize DB for measurement collection
Experiment Script
Multicast channel
Interface initializations and configuration e.g Intel, Atheros, Cisco
Applications
Repository

Fig. 1. Software architecture overview with the ORBIT radio grid.

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.

III. Overview of the ORBIT and PlanetLAB testbeds

ORBIT: The overall architecture of the ORBIT testbed is influenced by the requirement to provide a multi-user wireless experimental facility. This presents some interesting challenges including routine ones such as user account maintenance, access control, user portal for experimenters as well as more complex ones related to optimizing the usage of the testbed by accommodating as many users as possible in a given time duration. As opposed to wired experimentation, where users can have access to the shared facility simultaneously and can be segregated either at the MAC layer using VLANs or IP layer using firewalls or a combination of both, wireless experimentation poses an interesting challenge due to the inherent broadcast nature of the medium, thereby affecting the other nodes in the vicinity. Partitioning a wireless grid in a controlled fashion for simultaneous experiments could be achieved either by introducing physical barriers that block radio propagations from one portion to the other or 'soft' walls introduced by using an array of noise generators. The former approach is difficult to reconfigure and involves physical movement of objects (RF shields) to the portion requiring isolation and hence not scalable. Nevertheless, until any of the above schemes are in place, a simple sequential scheduling allowing one set of experiments to use the entire shared facility at a time is currently being used. Thus, given the sequential nature of usage, it becomes necessary to accommodate as many users as possible and our software architecture and design is primarily influenced by this criterion. As identified earlier, in most of the experiments, setting up the experiment (using scripts or other control mechanisms) 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 simplifying data collection as much as possible. An experiment usually comprises the

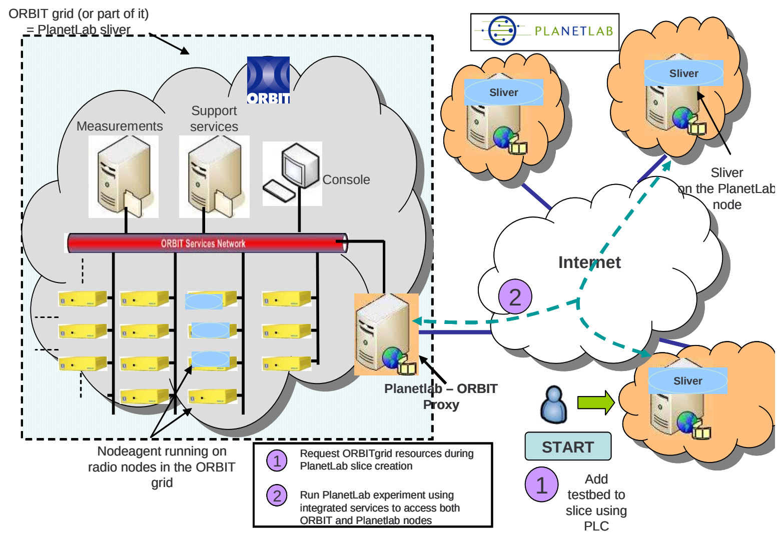

ORBIT grid (or part of it) = PlanetLab sliver
ORBIT
Measurements
Support services
Console
ORBIT Services Network
PLANETLAB
Sliver
Sliver
Sliver
Sliver on the PlanetLab node
Internet
2
Planetlab – ORBIT Proxy
START
1
Add testbed to slice using PLC
Nodeagent running on radio nodes in the ORBIT grid
1 Request ORBITgrid resources during PlanetLab slice creation
2 Run PlanetLab experiment using integrated services to access both ORBIT and Planetlab nodes

Fig. 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.

following steps: 1)Selection of nodes which will be a part of the experiment. 2)Selecting the roles played by each of these nodes in the experiment (sender, receiver, AP, relay etc). 3)Deploying necessary software on each node corresponding to the role they play each. 4)Configuration of wireless interfaces (Ad-hoc or Managed,Power levels, Channel settings etc). 5) Collecting results at run-time and collate them (statistical analysis or simple time plots)

In this 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). 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.

PlanetLab: Planetlab infrastructure consists of a globally distributed set of PL-nodes that support broad coverage services that benefit from having points of presence on the network. PlanetLab users either run short-term experiments or deploy continuously running services. These models are supported by PlanetLab's distributed virtualization - each experiment runs in a slice of the global resources. Multiple slices run concurrently on PlanetLab independent of each other. Each PL-node runs Linux-Vservers which implement both namespace and performance isolation among the various slices running on the node - defined as a sliver. A sliver is a set of allocated resources on a single PlanetLab node. Network virtualization is provided by use of VNET.

The first step for a user is to create a slice in the PlanetLab network that extends across several nodes. All slices are given best effort promises when they are created. Slices acquire and release network resources over time and there is no guarantee on the set of resources for a slice. PlanetLab Consortium (PLC) manages the creation and authentication of the slices. Once the slice has been created to the nodes, they are populated with the minimal Fedora Core Linux Installation.

The testbed does not offer any support for choreographing an experiment and controlling all the nodes using automated scripts. Popular tools like ssh, pssh [15], scp and pscp are used to gain access to the nodes, to control them and to populate the nodes with the required applications and software. In addition the results and measurements of the experiment have to be manually collected. Despite all this, there are third party softwares available for controlling and monitoring experiments in PlanetLab. Considering the above factors, the following two models are presented in the next section that have been developed to integrate these two testbeds and conduct joint experiments.

IV. Differences In Testbed Architectures

Both of these testbeds differ in their architecture significantly, based largely on their purpose. The major differences are as follows:

Experiment models - The duration of experiments on the ORBIT testbed is short-lived, on the order of a couple of hours, 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 and dynamic addition of slivers during run-time (relying on Planetlab's low-latency mechanism for sliver addition).

System state maintenance - Planetlab maintains information regarding the state of the individual nodes. ORBIT could either rely on existing Planetlab mechanisms for this information (as well as updates) or implement its own mechanism. In addition to per-node state, the ORBIT

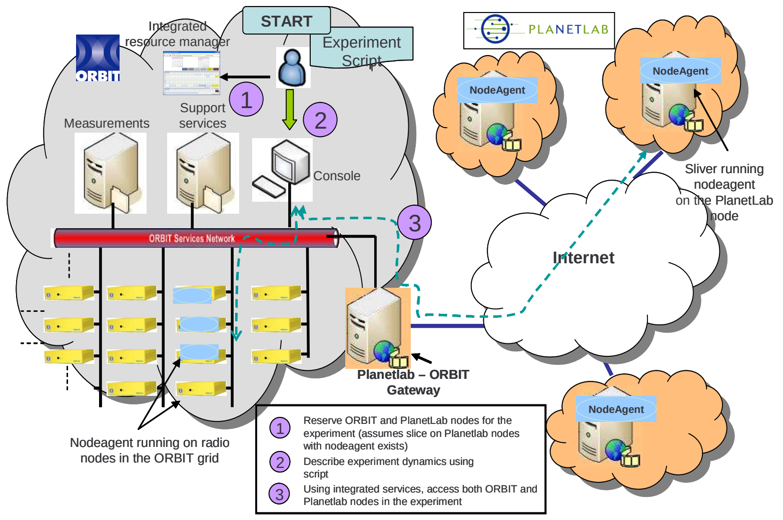

START
Experiment Script
Integrated resource manager
ORBIT
PLANETLAB
Measurements
Support services
Console
ORBIT Services Network
NodeAgent
Sliver running nodeagent on the PlanetLab node
Internet
Planetlab – ORBIT Gateway
Nodeagent running on radio nodes in the ORBIT grid
1 Reserve ORBIT and PlanetLab nodes for the experiment (assumes slice on Planetlab nodes with nodeagent exists)
2 Describe experiment dynamics using script
3 Using integrated services, access both ORBIT and Planetlab nodes in the experiment

Fig. 3. 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.

testbed will also need to maintain information on slice expiration and node membership at much shorter time scales. Given the potential possibility for the maintenance of state by both testbeds, it is imperative that they maintain a consistent view of the available resources.

Failure handling - Given the large-scale, distributed and evolving nature of both testbeds, failures of many different kinds are inevitable. These failures will need to be classified and strategies to mitigate them as well as corrective measures will need to be developed.

Transparent resource ownership and accountability - Users of the ORBIT testbed are affiliated to different organizations and utilize resources at much shorter time scales. In order to ensure accountability of user actions with regards to resource usage (especially on Planetlab), mechanisms will be required to track resource ownership information and share it between the two testbeds. Moreover, unlike the current ownership model on Planetlab where resource ownership mostly changes on time scales which are on the order of months, resource ownership in this model will change much faster, on the order of a couple of hours.

Experiment life cycle -While Planetlab is based on a long-running service oriented experimentation model based on the concept of virtualization of resources, the ORBIT testbed provides an infrastructure for repeatable wireless experiments. This experimental testbed runs on top of the Internet as an overlay thereby giving researchers access to: 1) a large set of geographically distributed machines, 2) a realistic network substrate that experiences congestion, failures, and diverse link behaviors, and 3) the potential for a realistic client workload. Planetlab uses a central slice creation and management utility, Planetlab Central (PLC) for adding, renewing and managing existing slices on different nodes.

On the other hand, 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. Exclusive access to the grid ensures no RF interference from other experiments. Experimental control software (NodeHandler) as well as an integrated measurement collection framework (OML) facilitate the definition, execution of experiments and also enable the collection of experimental results at run-time.

V. Integration Models

In this section we will present two approaches to integration of the virtualized testbeds. 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.

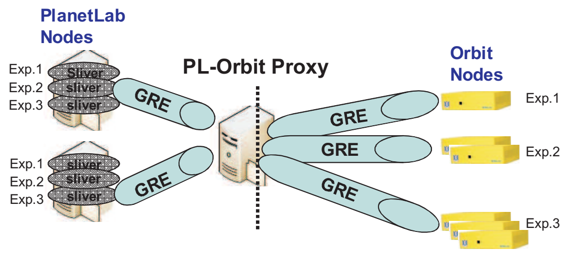


Fig. 4. A sample approach based on the PDIE model. The figure shows the use of a PlanetLab - ORBIT proxt for mapping PLanetLab slivers to ORBIT nodes.

A. 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 will provide abstractions for the setup, control and measurement on a specified topology using a modified version of the nodeHandler as the interface software. This approach does not involve major changes to either the PlanetLab or ORBIT testbeds but would require the development of a PlanetLab proxy module. PL users can use their own preferred tools, such as *pssh* [15] or the nodeHandler-nodeAgent framework.

Figure 4 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[7].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.

This design requires extensions to the Planetlab resource specification model to include topology and some other wireless-specific features. Note that this Planetlab-ORBIT gateway would also benefit from future virtualization capabilities to be developed for ORBIT, bridging the gap between the Planetlab slice model and ORBIT's single-user model.

Using the virtualized ORBIT testbed, a Planetlab slice could be extended to include individual ORBIT nodes. Currently, the entire ORBIT testbed resources have to be reserved and allocated to one user. To deploy the integration model, wireless virtualization solutions need to be investigated for ORBIT to allow long term concurrent integrated experiments. In this paper, we discuss a couple of proof-of-concept experiments on PlanetLab and ORBIT with the notion of virtualization of ORBIT testbed resources.

B. ORBIT Driven Integrated Experimentation (ODIE) Model

The ODIE approach allows users to include the long-running "ORBIT slice" in Planetlab nodes in their experiments with a single experimental script. This model is similar to that proposed in [14], where the authors describe the integration of the Emulab [6] and PlanetLab testbeds to provide Emulab users with access to PlanetLab resources. Figure 3 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 ORBIT NodeHandler was developed to support experiment definition, code download and execution. This NodeHandler communicates with modified NodeAgents running on the PlanetLab slivers. The following approaches can be used for topology selection and in our current implementation, we look at the manual addition approach

- Manual addition - experimenters can choose the PlanetLab nodes individually.
- Metric-based addition - experimenters can either prescribe the link-specific or node-specific characteristics they desire and NodeHandler will automatically assign the appropriate PlanetLab nodes to the experiment. If one of these nodes fails during the experiment, the NodeHandler shall dynamically switch to another node satisfying the experimenter's criteria.

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

- Extended addressing scheme to include PlanetLab nodes: We address the PlanetLab 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 (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. This modification eliminates the need to provide reliability in the application layer. 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 using the popular tool 'pssh' 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 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. This feature removed the need to manually

```
#-------------ACCESS POINT-----------#
defNodes('AccessPoint', [11,20])
{|node|
node.prototype("test:proto:mvlcrelay",
    {'duration' => prop.duration})
    #802.11 Master Mode
    node.net.w0.mode = "master"
    node.net.w0.type='a'
    node.net.w0.channel="48"
    node.net.w0.essid = "link1"
    node.net.w0.ip="192.168.7.1"
}
#---------------CLIENT--------------#
defNodes('Client', [19,2])
{|node|
node.prototype("test:proto:mvlcdest",
    {'duration' => prop.duration})
    node.net.w0.mode = "managed"
    #802.11 Managed Mode
    node.net.w0.type='a'
    node.net.w0.channel="48"
    node.net.w0.essid = "link1"
    node.net.w0.ip="192.168.7.7"
}
#----------- PlanetLAB nodes----------#
defPNodes('[21,3,[21,5]')
```

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

ssh each of the PlanetLab nodes in the experiment to start the applications. 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 that the nodeHandler parses to execute the experiment automatically.

- Caching Mechanisms: Caching mechanisms to aid measurements collection - if a PlanetLab node goes down during an experiment, mechanisms will be needed to extract measurements related to the previously running ORBIT experiment. The current ORBIT Measurement Library(OML) framework has to be extended to provide real time measurement collection for the PlanetLab nodes.

Experiment Scripting: The ODIE based experiment script is parsed and executed by the NodeHandler to choreograph the experiments. The NodeHandler runs on the console of the ORBIT grid. A single script for the ODIE models may be described in two sections:

- Node configuration section
- 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

```
#--Start applications on ORBIT nodes-#
whenAllInstalled() {|node|
    nodes('AccessPoint').startApplications
    wait 5
    nodes('Client').startApplications
    wait 195  # Experiment Duration
    allNodes.stopApplications
    Experiment.done
}
# MAPS to planetlab01.cs.washington.edu,
# and planet.cc.ga.atl.edu
# Start applications on PlanetLAB nodes
WhenPLReady(){
   defPApplication([21,3],'VIDEO1'){}
   defPApplication([21,5],'/VIDEO2'){}
   wait 195
   defPApplication([21,3],[21,5],'STOP'){}
PLexpdone() }
```

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

the nodes for the experiment. The first part of 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. The node is manually chosen by the user. However the PlanetLab nodes are addressed as extension to the ORBITs Node addressing scheme. 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].

Figure 6 describes the timing and execution section of a typical ODIE script. The *WhenAllInstalled()* module is responsible for checking if all the ORBIT nodes have been configured as per the specification in Figure 5. Typically when this is verified, individual applications like running a traffic analyzer, traffic generator or any custom defined applications can be executed based on the timing specified in the script. Typically the application script in Figure 6 starts the applications on the access point and waits for 5 seconds before starting the applications on the client nodes.

The module *WhenPlReady()* waits for the nodeAgents on the PlanetLab to report. There is a time-out for every PlanetLab node after which the NodeHandler ignores the nodes that fail to report and chooses other available nodes in the ”ORBIT slice” on PlanetLab. Once the desired number of PlanetLab nodes have reported, the applications on the respective nodes 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 should provide an

$Experiment setting$	$Max Jitter\ (ms)$	$Max Delta\ (ms)$	$Mean Jitter\ (ms)$
$Wireless\ One\ hop$	1.68	7.66	0.98
$Pl - Princeton$	1.69	7.69	0.98
$Pl - Washington$	1.88	8.76	1.05
$Pl - Washington\ (Loaded)$	35.76	415.8	3.62
$Pl - Japan$	52.18	779.02	7.38

Fig. 7. Jitter results observed with different PlanetLab nodes serving the same video over the internet to wireless clients in the ORBIT grid.

$Parameter$	$Value$
$Channel\ Rate$	24Mbps
$Offered\ Load$	Time Varying
$Experiment\ Duration$	2 Minutes
$Averaging\ Duration$	Per Second
$Operation\ Mode$	802.11a

Fig. 8. Experimental Parameters Used With ORBIT Nodes

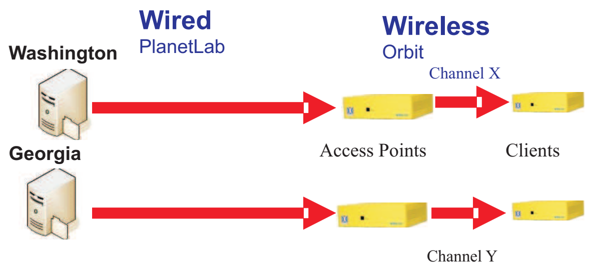


Fig. 9. Experiment layout where nodes are added from PlanetLab while frequency seperation (FDMA) is used with the ORBIT nodes.

interface for efficient resource teardown mechanism upon slice termination for each new experiment (every 1-2 hrs). This mechanism can then be leveraged by the NodeAgents running on the Planetlab nodes in the ORBIT slice.

VI. Proof-Of-Concept Integration With Wireless Virtualization

Virtualization requires efficient and systematic sharing of resources across the framework to enable maximum utilization. In this section we will evaluate some experimental scenarios with integrated tests of the PlanetLab and ORBIT testbed. Though these experiments performed with the ODIE model, they can be carried out with equal ease with a PDIE model. Throughout these experiments we show the ease of conducting experiments in an integrated fashion along with some approaches used for the virtualization of ORBITs wireless resources.

Typically, we are running the following experiments:

1) Integrated experiments with frequency division multiplexing of ORBIT Nodes
2) Integrated experiments with MAC layer virtualization of ORBIT Nodes

The experiments in this section do not aim at showing any important results with the example experiments but rather the dexterity of the framework to perform integrated experiments with maximum ease.

A. 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. It is important to note that we do not aim to do a comparative study of the video delivery algorithms themselves but rather demonstrate a way to evaluate these algorithms with our setup.

Topology: Figure 9 shows the topology for this experiment. We consider a typical scenario for streaming video delivery across a network path that includes a 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 such as channel rate, transmission power, injected noise, packet sizes can be varied to test the effect of the wireless link on the overall performance of the link. The video is streamed and played using the Video LAN (vlc) [8] player. The OML(ORBIT Measurement Library) framework [26] 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.

Jitter measurement: Figure 7 shows the results for the jitter values from the FDMA experiments. A video was delivered from PlanetLab nodes in Princeton (NJ), Washington and Japan. Results show relatively comparable jitter values for the Princeton and Washington PlanetLab nodes. Japan on the other hand sees a higher jitter for video delivery possibly due to higher traffic and geographical distance. In another case we simulated a heavily loaded server by adding traffic to the Washington node. The results show the increased delay for such a busy node. To include a few baseline experiments for comparison sake, we also tested the jitter across baseline scenarios such as a single hop wired and wireless link. All these readings are easily obtainable either through the ORBIT measurement library (OML) framework using the integrated tcpdump[9] tool. These experiments do not aim to specify performance of these algorithms with individual nodes but rather show the usefulness of using the integration models (ODIE/PDIE) to obtain results for

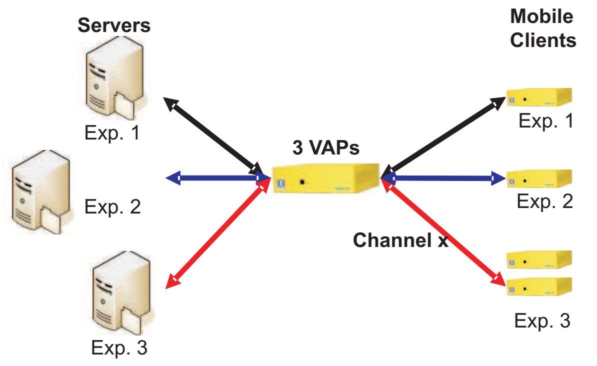


Fig. 10. Experiment layout where nodes are added from PlanetLab while the VAP support from the 802.11 linux drivers is exploited for running multiple networks from a physical AP.

averaging and testing across multiple nodes with repeated experiments.

Bit rate measurement: Figure 11 shows a plot of observed video bit rate at the client as a function of time. These bit rates are measured in the same experiments where we measured the jitter results shown in Figure 7. Experiment setup is the same as that shown in Figure 9. Figure 11 shows that the observed bit rate for the videos decreases with the PlanetLab node in Japan as compared to those in Princeton or Washington. Calibration tests performed over the one hop wired and wireless networks server as a baseline for comparison. Figure 12 shows the performance of the individual flows in terms of the average bit rate at the wireless client for the same video. Surprisingly the increased load on the Washington node only shows increased jitter in the video delivered with a comparable mean bit rate. The PlanetLab node in Japan has a significantly deteriorated bit rate at the client.

These results would help provide a deep understanding and evaluation of proposed video delivery algorithms. It would also help to better understand the limitation of media delivery over wired/wireless networks. Ongoing work on designing robust and reliable transport-layer protocols for achieving high quality video streaming would benefit from the results of such experiments. In addition, research on spectrum allocation, bandwidth management and inter-Access Point communication protocols would require the presence of a similar framework.

We would like to emphasize again, that the goal of these experiments is not to quantify individual test performance results (since tests over the internet are not entirely reproducible), but rather to show that the infrastructure based on our ODIE and PDIE models can be easily used for averaging and comparing performance across variety of nodes and experiment trials.

A disadvantage of using frequency division multiple access (FDMA) for virtualization on the ORBIT grid is that it does not scale well with the number of nodes and available orthogonal frequencies. In the next section we explain results based on a virtual access point based approach to virtualization on the ORBIT grid.

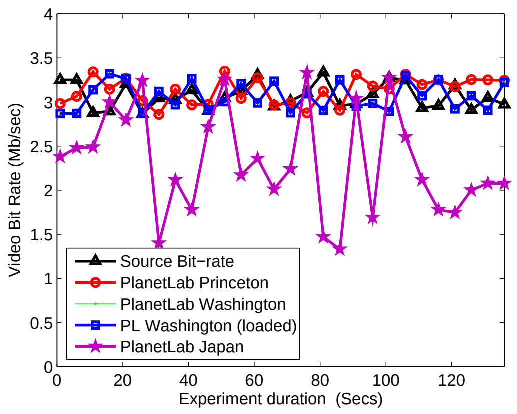


Fig. 11. Observed bit rate with the same test video being delivered from different PlanetLab nodes.

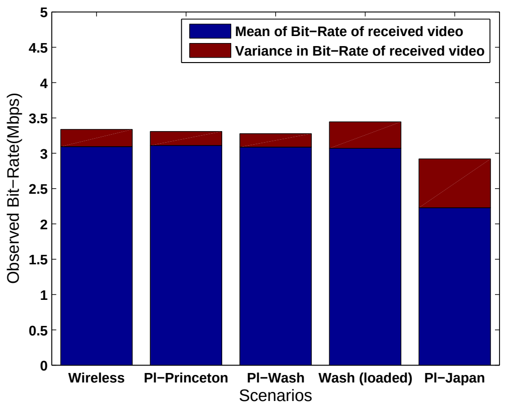


Fig. 12. Mean and variance statistics from the video bit rates observed at the client node.

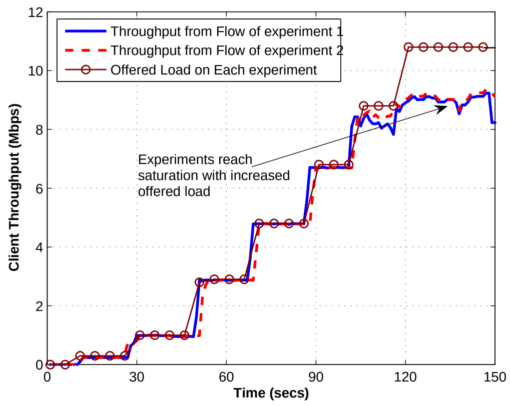
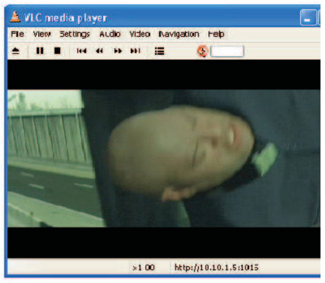
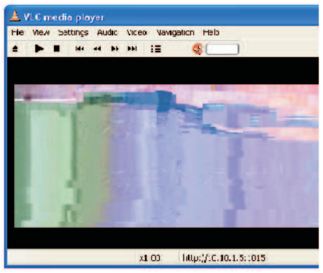


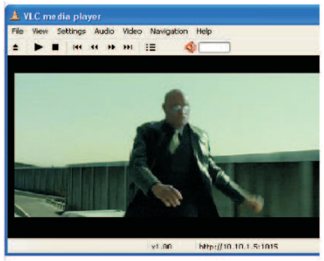
Fig. 13. Throughput (Mbps) seen at the wireless client without lack of traffic shaping or policy management for bandwidth control.


VLC media player
File View Settings Audio Video Navigation Help
x1.00 http://10.10.1.5:1015

(a) Below saturation video at the start of the experiment


VLC media player
File View Settings Audio Video Navigation Help
x1.00 http://10.10.1.5:1015

(b) The data flows saturate the channel, thereby resulting in deterioration of the video quality


VLC media player
File View Settings Audio Video Navigation Help
x1.00 http://10.10.1.5:1015

(c) Video performance after traffic shaping with manual intervention. Availability of sufficient bandwidth restores the video quality

Fig. 14. Video performance as the experiments progress with increasing offered loads. Video is observed by forwarding the traffic from the client node to an observation node in the grid.

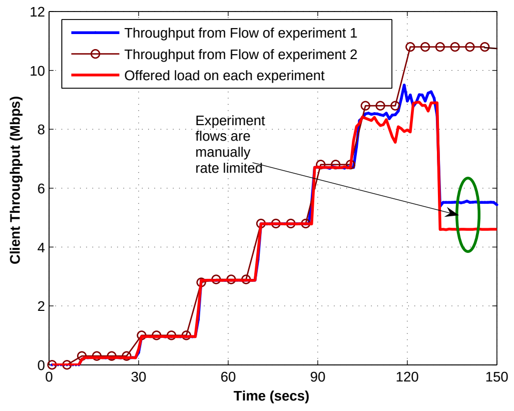


Fig. 15. Throughput (Mbps) seen at the wireless client where manual intervention rate limits flows to stop channel saturation.

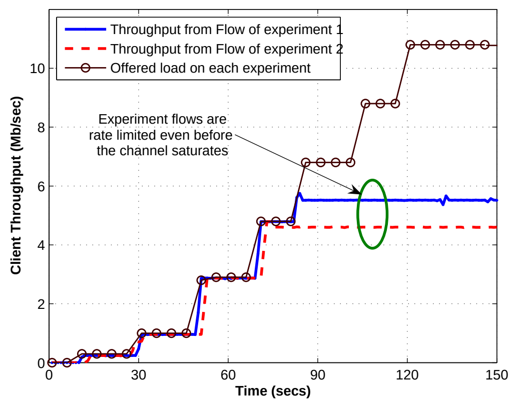


Fig. 16. Throughput (Mbps) seen at the wireless client where individual flow rates are pre-decided by the policy manager. Channel never saturates.

B. Virtual Access Point Based ORBIT slicing coupled with PlanetLAB

These proof-of-concept experiments use 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[27]. Its use is limited to fixed-star topologies to support long-running concurrent experiments. As compared to the FDMA based virtualization, this approach conserves the number of nodes and channels used by the experiments.

Aim : The goal of this experiment is to show some preliminary results that can be obtained with a typical ODIE/PDIE experiment. We also show that if a VAP based approach is used for virtualization on the grid, there are concerns with the performance of one experiment affecting the other. We follow this with a proposed solution to the interference problem.

Topology and setup: Figure 10 shows the setup for this test. This setup 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 to similar to the one shown in Figure 5 may be used.

Figure 13 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. Figure 14(a) shows the video seen at the client node which is represents the traffic of the third experiment in the setup. It can be observed that since there are very few packet drops and low congestion in both the wired and wireless network the video is clear.

As the offered load for each of the experiments is increased with time, the aggregate traffic on the wireless network reaches saturation. Figure 13 shows that in saturation the net throughput seen for both flows are comparable. However, the picture in Figure 14(b) shows that the video suffers considerably when the wireless channel is in saturation. The increased distortion in the video quality may be attributed to the increased levels of jitter and dropped packets with the video flow. To prevent such situations where the 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 15 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 Click Modular Router [32] 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, even before they are started. The policy managed 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 16 shows the results with a dummy policy manager. Both the experiments are rate limited to their assigned throughput values even before they reach channel saturation. Figure 14(c) shows a snapshot of the improved video seen at the client after traffic shaping is used at the VAP.

VII. Acknowledgment

The authors would like to acknowledge the contributions of Pandurang Kamat and Kishore Ramachandran from WINLAB for their contribution to the ORBIT-PL integration project.

VIII. Conclusions And Future Directions

The unified designs presented in this paper should 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.

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] C. Doerr A. Sheth M. Neufeld J. Fifield D. Grunwald SoftMac: Flexible Wireless Research Platform, in Hot Topics in Networking (HotNets-IV), 2005.
[4] D. Raychaudhuri and Editors M. Gerla, New architectures and disruptive technologies for the future internet: The wireless, mobile and sensor network perspective in Report of NSF Wireless Mobile Planning Group (WMPG) Workshop, August 2005.
[5] D. Culler L. Peterson, T. Anderson and T. Roscoe, "A blueprint for introducing disruptive technology into the internet," in First Workshop on Hot Topics in Networking (HotNets-I), 2002.
[6] L. Stoller R. Ricci S. Guruprasad M. Newbold M. Hibler C. Barb B. White J. Lepreau A. Joglekar "An integrated experimental environment for distributed systems and networks," in Proceedings of the 4th Symposium on Operating System Design and Implementation (OSDI 2002), 2002
[7] RFC3147, Generic Routing Encapsulation over CLNS Networks, IETF draft of the networking working group, July 2001
[8] Videolan Player in `http://www.videolan.org/vlc/`
[9] Manpage of tcpdump in `http://www.tcpdump.org/`
[10] XBONE in `http://www.isi.edu/xbone/`
[11] DETER TestBed in `http://www.isi.edu/deter`
[12] 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 Wireless Communications and Networking Conference,2005
[13] "Wisconsin advanced internet laboratory" in `http://wail.cs.wisc.edu`
[14] Implementing the emulab-planetlab portal :Experiences and lessons learned, in Proceedings of the First Workshop on Real, Large Distributed Systems (WORLDS 2004),2004.
[15] B. Chun PSSH/PSCP tool `http://www.theether.org/`
[16] E. Bugnion S. Devine and M. Rosenblum, Virtualization system including a virtual machine monitor for a computer with a segmented architecture in US Patent, 6397242, 1998.
[17] Connectix, product overview: Connectix virtual server, 2003 in `http://www.connectix.com/products/vs.html`
[18] K. Fraser S. Hand T. Harris A. Ho R. Neugebauer I. Pratt P. Barham, B. Dragovic and A. Warfield, Xen and the art of virtualization in SOSP, 2003
[19] M. Shaw A. Whitaker S.D. Gribble Scale and performance in the denali isolation kernel in Proceedings of the 5th Symposium on Operating Systems Design and Implementation, 2002
[20] S. Shenker L. Peterson and J. Turner Overcoming the internet impasse through virtualization in Third Workshop on Hot Topics in Networking (HotNets-III),2004.
[21] PlanetLab: An Open Platform For Developing, Deploying and Accessing Planetary-Scale Services in `https://www.planet-lab.org/`
[22] M. Huang, Vnet: Planetlab virtualized network access," in http://www.planet-lab.org/PDN/PDN-05-029.
[23] Ensim corp. ensim virtual private server," in in `http://www.ensim.com/products/materials/datasheetvps051003.pdf`
[24] J. Dike, User-mode linux in Proceedings of the 5th Annual Linux Showcase and Conference, 2001.
[25] Ieee standards for local and metropolitan area networks, ieee std 802.1q, virtual bridged local area networks," 2003.
[26] M. Singh, M. Ott, I. Seskar P. Kamat ORBIT Measurements Framework and Library (OML): Motivations, Design,Implementation, and Features in Proceedings of IEEE Tridentcom 2005, Trento, Italy, Feb 2005
[27] Creating virtual ap on madwifi `http://madwifi.org/wiki/UserDocs/MultipleInterfaces`
[28] B. Lim J. Sugerman, G. Venkitachalam Virtualizing i/o devices on vmware workstation's hosted virtual machine monitor" in Proceedings of the USENIX Annual Technical Conference, 2002.
[29] R. Siracusa M. Singh M. Ott, I. Seskar ORBIT testbed software architecture: Supporting experiments as a service in Proceedings of IEEE Tridentcom, 2005.
[30] C. Law, A Mehta, and K. Siu, A New Bluetooth Scatternet Formation Protocol, ACM Mobile Networks and Applications Journal, 2002.
[31] C. Law and K. Siu, A Bluetooth Scatternet Formation Algorithm, IEEE Symposium on Ad Hoc Wireless Networks, November 2001.
[32] Click Modular Router. `http://read.cs.ucla.edu/click/`
[33] IPERF, TCP/UDP Traffic Generation Tool, `http://dast.nlanr.net/Projects/Iperf/`
[34] P. Barford and M. Crovella, Generating Representative Web Workloads for Network and Server Performance Evaluation, In Proceedings of the ACM SIGMETRICS, pages 151-160, Madison WI, November 1998.