A Wired-Wireless Testbed Architecture for Network Layer Experimentation Based on ORBIT and VINI

George C. Hadjichristofi, Avi Brender, Marco Gruteser, Rajesh Mahindra, Ivan Seskar
WINLAB, Rutgers University
671 Route 1 South
North Brunswick, NJ 07310, USA
gh@winlab.rutgers.edu, abrender@eden.rutgers.edu, {gruteser, rajesh, seskar}@winlab.rutgers.edu

ABSTRACT
Renewed interest has emerged in novel internet architectures that among other aspects can better address the specific requirements of wireless networks, such as increased host mobility, router mobility, or flow control over fast-changing links. Most existing testbeds, however, focus either on wired or wireless networks and provide inadequate support for experimentation with novel network architectures spanning the wired and wireless domain.

This paper presents an initial design of a global-scale wired-wireless testbed and its prototyping based on the existing ORBIT and VINI testbeds. It allows researchers to define custom network topologies comprising both wired and wireless nodes and experimenting with new network and transport layer protocols in these networks. The testbed relies on virtualization on wired nodes, trading a slight performance penalty for the ability to allow simultaneous usage for multiple long-running experiments.

Categories and Subject Descriptors: D.4.7 [Operating Systems]:Organization and Design---Distributed systems; C.2.3 [Computer-Communication Networks]: Network Operations---Network management; D.4.8 [Operating Systems]: Performance---Operational analysis

General Terms: Design, Experimentation

Keywords: VINI, ORBIT, PlanetLab, Integration, Architecture, Experiments

1. INTRODUCTION
As the continued expansion of the Internet is exposing security and host mobility limitations, among others, in the original design, renewed interest has emerged in a fundamental redesign of the internet architecture. This is accompanied by the GENI effort to provide a testbed for experimentation with novel network architecture concepts.

Most existing large-scale network testbed designs, however, concentrate either on wired networks (e.g., PlanetLab[1] or VINI [2][3], Emulab[4][5]) or on controlled wireless experimental platforms such as the Open Access Research Testbed for Next-Generation Wireless Networks (ORBIT) testbed [6], the Ad Hoc Protocol Evaluation Testbed (APE) [7], MIT Roofnet, Transit Access Points (TAPs) [8], and WHYNET [9]. We are only aware of the expansion of Emulab with wireless nodes as a first integrated design, providing emulation capabilities to test novel network concepts.

In this paper, we design and prototype a testbed that provides global-scale experimentation with custom topologies involving wired and wireless nodes based on the existing ORBIT and VINI testbeds. Both testbeds are in daily heavy use, but have only allowed exclusively wired or wireless experiments. Using the integrated platform, researchers will be able to emulate real network topologies with wired nodes at PlanetLab sites around the world and wireless nodes at the ORBIT facility. The contribution of this paper is a framework for integrating wired and wireless testbeds to support Layer 3 experimentation. In our design, the wired and wireless testbeds may be physically disjoint and may belong to different organizations or parties (i.e., do not share the same control and management framework, and network architecture [4][5]). To the best of our knowledge, this is the first attempt for such an integration.

In section 2, we describe the ORBIT and VINI testbeds. Section 3 specifies our design goals and requirements for this integration and section 4 provides the design aspects and implementation details of the integrated solution. Section 5 demonstrates a proof-of-concept Layer 3 experiment deployed over the integrated wired and wireless testbed. Section 6 discusses our solution.

2. BACKGROUND
ORBIT is a wireless network of 400 nodes arranged in a two-dimensional grid [6]. The project is currently being developed and operated by WINLAB at Rutgers University. Researchers must use an online scheduler to reserve time slices on the grid, during which time they have full access to every node. During a time slot an ORBIT user can image all of the radio nodes with an operating system of his choice and carry out various experimental scenarios.

VINI is a virtual network infrastructure that allows network researchers to evaluate their protocols and services and runs on the PlanetLab wired testbed [1][2][3]. PlanetLab is a global research network operated by Princeton University. Currently there are nearly 800 nodes available for use at 379 locations

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
WiNTECH'07, September 10, 2007, Montréal, Québec, Canada.
Copyright 2007 ACM 978-1-59593-738-4/07/0009...$5.00.

around the world. Each physical PlanetLab node runs virtualization software known as Linux VServer. Each experimenter is given a sliver, or a virtualized section of a physical node. Each sliver is created on every PlanetLab node that is part of the experiment.

2.1 VINI and ORBIT Architectures

The VINI virtual network infrastructure creates virtual nodes that run on existing PlanetLab nodes. Each virtual node has access to one or more virtual network resources that can be used during experiments. VINI leverages a number of technologies, which have been integrated together [3]. Figure 1 shows the basic components that are utilized in a VINI virtual node.

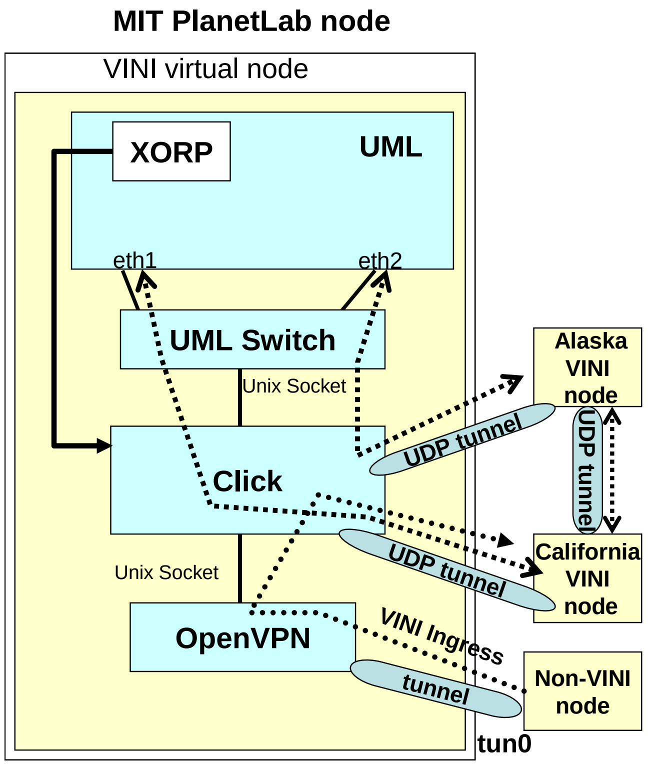


Figure 1. VINI software architecture.

User Mode Linux (UML) runs as a user-space process and is a full-featured Linux kernel [10]. It creates a virtual environment complete with network devices and a file system. The UML instance is started from within the experimenter's slice on PlanetLab. Each virtual environment on a PlanetLab-VINI node communicates with other virtual environments by creating an overlay topology with UML interfaces as the end points.

XORP, running within UML, is an open source routing protocol suite [11]. XORP implements a number of routing protocols, including BGP, OSPF, RIP, PIM-SM, IGMP, and MLD. It manipulates routes in the data plane through forwarding engines, such as the Linux kernel routing table and the Click modular software router discussed later on. XORP generally assumes that each link to a neighboring router is associated with a physical interface. Within UML, it has the impression that the Ethernet interfaces are directly connected to neighbors. For example, if an MIT sliver is to be connected to Alaska and California, within UML packets sent out via eth1 would be sent to California and packets sent out via eth2 would be sent to Alaska (see Figure 1). In reality, all packets going in and out of UML are sent first through the UML switch and Click modules at the lower layers. The IP addressing scheme used in UML provides each virtual link with its own class C IP address.

The UML switch is a virtual switch used to connect UML and Click through Unix sockets. Click is a modular routing software created and maintained by MIT [12][13]. All packets leaving UML are sent through the UML Switch to Click for inspection. Click is highly configurable and can inspect, modify, and route any type of packet. All the information exchanged between Click and UML must be Ethernet encapsulated. Click creates UDP sockets on a specified port for VINI related packets and maps the packets sent on these UML interfaces to the appropriate UDP tunnel.

OpenVPN [14] was introduced into the VINI software architecture with the main scope of connecting an outside machine to a VINI experiment, as an edge node, and allowing for IP traffic injection into VINI. It is ran on every VINI virtual node and is linked with the Click router trough Unix sockets allowing for appropriate packet framing of ingress traffic as Ethernet packets. In the reverse direction, Click must strip Ethernet headers on packets from UML and deliver them to OpenVPN. OpenVPN on VINI nodes uses the PlanetLab slice's tun0/24 and is configured as a server that pushes IP addresses in 10.0.0.0/8 range.

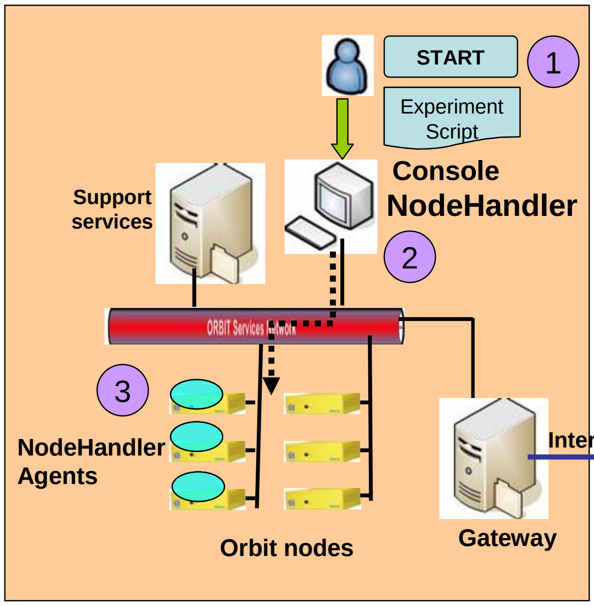


Figure 2. Setting up an experiment using the ORBIT framework.

Contrary to VINI, a user of the wireless ORBIT testbed is not restricted by any software architecture constraints. An ORBIT user has complete freedom to install and run any operating system and related software on ORBIT nodes and carry out wireless experiments during a reserved time slot. More specifically, if an

ORBIT user needs to run an experiment he/she typically logs onto the console and uses the NodeHandler to communicate with the NodeHandler Agents running on the ORBIT nodes, as shown in Figure 2. The NodeHandler can deploy the experiment for the user based on a given experimental script. A series of operating system images are already available and preconfigured with the NodeHandler and any other required communication software that enable the dynamic execution and control of experiments on all of the 400 ORBIT nodes. The usage model of ORBIT is different compared to VINI's. ORBIT experiments mainly have a short duration whereas PlanetLab-VINI experiments tend to be long-running.

3. DESIGN GOALS AND REQUIREMENTS

The integration between wireless and wired networks for network layer experiments needs to satisfy the following goals:

• No packet type restrictions: Any type of Ethernet encapsulated packet should be able to propagate between the two networks. These packet types include both IP and non-IP packets.

• Arbitrary topology creation: The solution should provide researchers with the capability to connect any wireless node to any wired node in different combinations and carry network layer and above experiments. Figure 3 shows a sample network configuration where wireless nodes (e.g., ORBIT nodes) are connected to wired (e.g., PlanetLab-VINI) nodes. One or more wireless nodes are connected to multiple wired nodes in the integrated overlay network.

• Transparency: While a software architecture on a testbed may provide a platform to deploy controlled network experiments it should not impose a high learning curve on users to setup these experiments. In addition, it should not require the users to make extensive modifications to the existing software architecture to deploy an experiment.

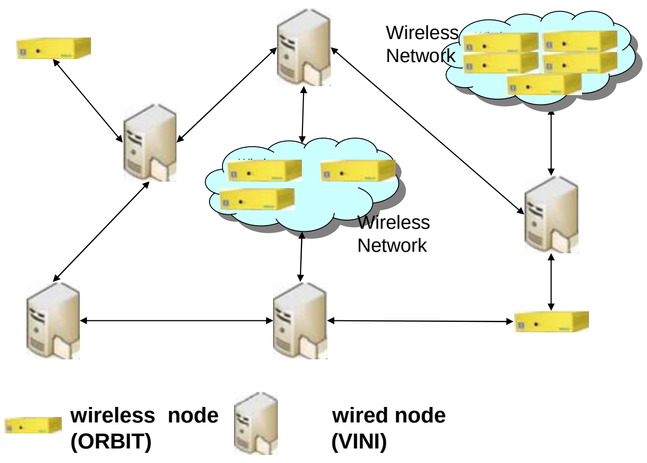


Figure 3. Sample configuration of the integrated testbed.

Reflecting back to the goals of this design, there are certain requirements or restrictions that are imposed for a successful integration of wireless and wired testbeds. Currently, most of the traffic carried out in the Internet uses IP. Experiments that aim to investigate non-IP traffic would have to be executed in an isolated testbed that may not properly reproduce a real-life network environment. Since the focus of this paper is on large scale multi-user testbeds that may be deployed across the globe then the notion of non-IP traffic needs to be addressed within the context of overlay networks. Overlay networks typically utilize IP tunnels and each overlay may represent an experiment that slices the resources of both wired and wireless testbeds. It is therefore important that the new integrated design accommodates the entire non-IP packet within the overlays' tunnels.

Another aspect that needs to be addressed is virtualization. Virtualization based on solutions such as VMware, Xen, or UML allow the creation of slices across a testbed and accommodate multiple users. Experiments on wired networks have been based on Layer 3 of the OSI stack, or above. However, typical experiments on wireless testbeds are based on the entire OSI stack and thus impose limitations in terms of virtualizing the wireless interface at the physical and MAC layers. In order to allow a user the flexibility to experiment with these lower layers, we chose not to virtualize the wireless nodes for this integration, but simply spatially segment the wireless testbed and allow each network slice full access to a specific segment of nodes.

4. INTEGRATION OF ORBIT AND VINI

In this section, we identify the various aspects of the architecture of the two testbeds that need to be modified or augmented to facilitate their integration and meet the goals mentioned in Section 3. The solution presented below represents the architecture that proved to provide the most flexibility and functionality and the least complexity in deployment.

4.1 VINI Architecture Components

OpenVPN IP tunnels allow the injection of an outside computer into a VINI experiment, but with limited capabilities. The current configuration allows a single host running an OpenVPN client to establish a VPN tunnel into VINI and inject IP traffic across a virtual network topology. This limitation has to be removed to allow a whole network of computers to be connected to VINI at multiple connection points.

In addition, OpenVPN in VINI uses IP tunnels (i.e., IP packets within IP) rather than Ethernet tunnels (i.e., Ethernet packets within IP). Modifying OpenVPN to use Ethernet links has many advantages over IP links, including but not limited to:

• Non-IP Protocols: An Ethernet-based OpenVPN link allows experimenters to run any protocol (e.g., NetBIOS or IPX) that is framed in Ethernet packets.

• Dynamic IP addresses: For security and routing reasons, when using an IP OpenVPN link in the existing VINI architecture, all IP addresses and ranges for which the link is responsible must be configured in advance and cannot be changed without breaking the VPN tunnel. Ethernet tunnels do not suffer from this limitation because the Ethernet tunnels simply pass all Ethernet frames through the tunnel without any IP requirements or routing complications.

• Broadcast Packets: OpenVPN IP links support unicast packets only and do not support broadcast/multicast packets. This is problematic because routing protocols operating on Layer 3 of the OSI stack may rely on broadcast packets to discover neighbors or peers. OpenVPN Ethernet links allow broadcast packets (Ethernet address: FF-FF-FF-FF-FF-FF or IP address range 240.0.0.0/4) to flow between VINI and ORBIT.

Another VINI architecture aspect that needs modifications is the Click modular router. Click performs forwarding of packets based on IP addresses, which restricts users into having predefined IP ranges that should be modified in Click as the experiment changes to reflect the proper IP routing scheme. In order to remove the need to modify Click on a per experiment basis and make experiments independent of IP protocols, changes are needed on Click so that Ethernet packets are delivered to the user space (i.e., UML) instead of IP packets. Within UML, an experimenter can then work in the familiar Linux environment and create the proper network drivers/modules to handle IP or non-IP packets.

4.2 ORBIT Architecture Components

ORBIT nodes do not have a predefined software architecture as compared to VINI nodes. Therefore, the integration issue as it relates to ORBIT deals more with deriving a solution that provides a desirable integrated topology rather than modifying an existing software architecture. On the ORBIT side, the integrated solution should enable a VINI node to communicate to one or more ORBIT nodes or groups of nodes. Two possible candidates that provide different topology characteristics are the Router configuration and the Bridge configuration.

In the Router configuration, an ORBIT node is set up as a router and is connected to the VINI network as seen in Figure 4. An OpenVPN tunnel is used to provide a point-to-point link between the two networks. The ORBIT Router A node supports the protocol of the packets received from the OpenVPN link through proper configuration within Linux.

It should be emphasized that although the example uses IPv4 routing, this design can utilize non-IP routing as well. In addition, this configuration enables Router A to run any of the routing protocol supported through XORP in the VINI environment and therefore exchange routes automatically with VINI nodes. In terms of integration, this set up can be visualized as adding nodes (e.g., Router A) and extending the existing VINI core network while providing access to a wireless networks. Typically, this mode can be utilized to enable integration of multi-hop wireless networks with a wired testbed.

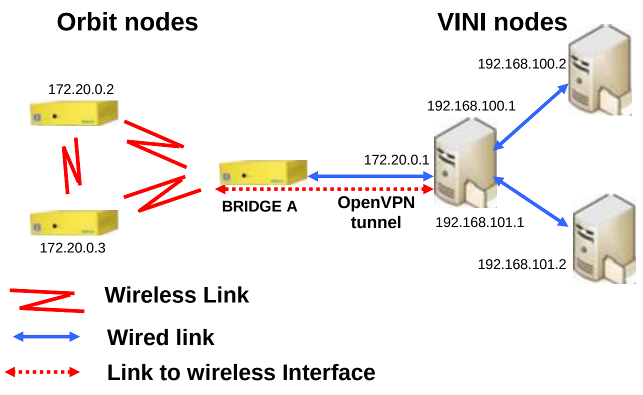


Figure 4. Integration using the router configuration.

In the Ethernet Bridge configuration, the ORBIT node bridges the OpenVPN interface with the wireless interfaces and removes the need to carry routing. Such a setup allows for experiments where multiple wireless end nodes are attached to VINI nodes and can be visualized as adding a wireless interface to a VINI node that is physically disjoint.

An example of a bridge configuration is shown in Figure 5. The VINI interface (172.20.0.1) is virtually connected to the Bridge A's wireless interface, which essentially makes that VINI node a wireless node attached to the ORBIT network or retrospectively provides the VINI node access to wireless ORBIT nodes (e.g., 172.20.0.2 and 172.20.0.3). Once again, IPv4 is used here as an example, but this framework will allow non-IP and broadcast packets. Typically, this mode can be utilized to enable the integration of access point functionality on the wired testbed nodes (i.e., one hop wireless connectivity).

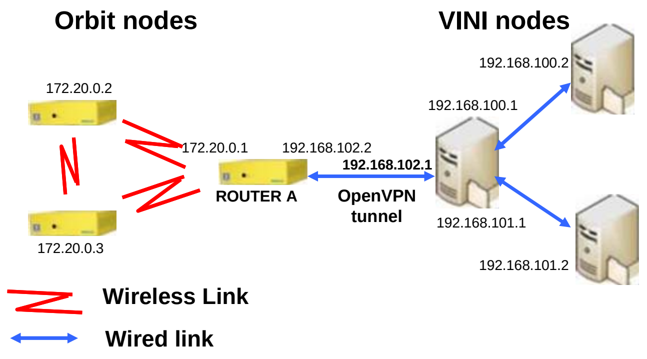


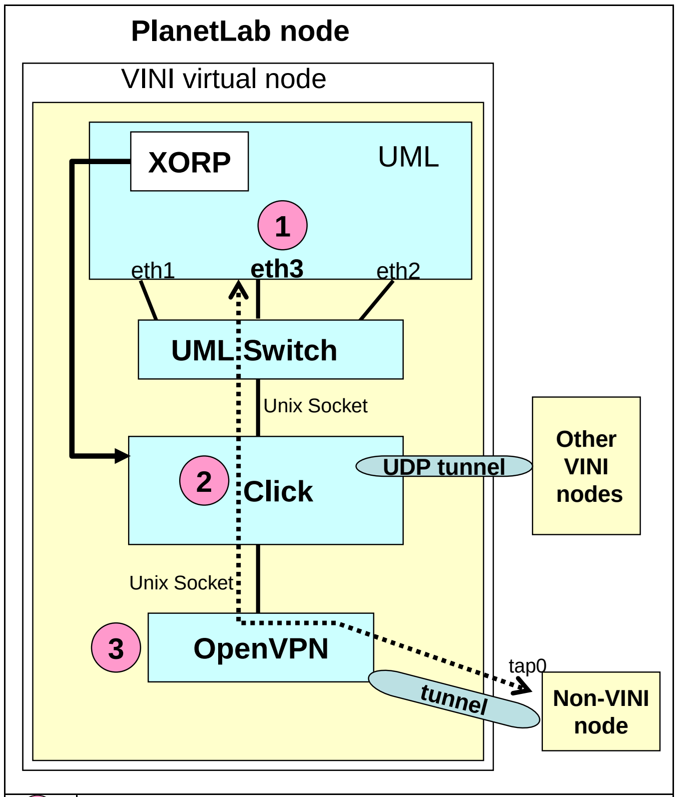
Figure 5. Integration using the bridge configuration.

4.3 IMPLEMENTATION

The modifications applied to ORBIT and VINI testbeds are presented in the following subsections.

4.3.1 VINI Modifications

For ingress traffic we pass Ethernet packets from OpenVPN to the UML instead of IP packets as currently utilized by VINI. We achieve this feature by modifying both OpenVPN and Click configurations. Ethernet tunnels are enabled by changing OpenVPN links between ORBIT and VINI to use Linux Tap devices instead of Tun devices. Therefore, any traffic from ORBIT to VINI is now delivered via OpenVPN Ethernet tunnels instead of OpenVPN IP tunnels. We then elect to send packets from OpenVPN directly to UML with no modifications. Since packets coming in from OpenVPN are already Ethernet encapsulated they do not need to be encapsulated again by Click. The Click forwarding mechanism is disabled and packets are sent directly to UML without inspecting the packet contents with a directive such as: "openvpn :: Socket(UNIX_DGRAM, "/tmp/click.sock") -> openvpn." Therefore, the UML instance handles all the packet routing decisions. The standard Linux 'route' command can be used within UML to direct packets. The advantage of this configuration is the support of broadcast traffic and non-IP based protocols. Non-IP routing protocols can be implemented and tested within UML. In the current VINI architecture, Click routes packets based on IP addresses, which restricts experiments to IP protocols. An additional benefit of this configuration is that it removes the need to modify the Click configuration for each experiment through the VINI setup files.



1	a) New interface binds to OpenVPN b) Enable forwarding and arp
2	a) Forward ingress traffic to UML b) Forward egress traffic to OpenVPN based on UML Ethernet address
3	a) Changed addressing scheme b) Utilized Tap instead of Tun interfaces to bind to OpenVPN and use Ethernet tunnels c) Changed from server to client setup to point-to-point setup

Figure 6. Key modifications of VINI architecture.

The method for handling egress traffic from the VINI UML instance to ORBIT in the modified architecture involves creating a new virtual interface in each UML instance, eth3 (see Figure 6). The VINI configuration scripts assign each eth3 interface a unique MAC address and its own IP address in the 192.168.0.0/16 range. Packets sent out from UML through this eth3 interface are assigned this known MAC address. Click is again modified to process Ethernet packets coming from the source MAC address of eth3 and send matching packets through OpenVPN links. Within the UML instance, eth3 interface is seen as a regular Ethernet interface and is expected to act as a normal interface. The interface can be assigned multiple IP addresses (aliases) on different subnets and can be used for any Linux routing implementation.

Another issue that we had to address during the integration of PlanetLab-VINI and ORBIT was the dynamic allocation of IP addresses. Both testbeds utilize the private class A reserved IP range 10.0.0.0/8, which lead to IP address conflicts. More specifically, each VINI instance uses the PlanetLab slices tap0 interface, which is assigned a unique class C address space within the 10.0.0.0/8 address space. OpenVPN server in VINI attempts to push routes for 10.0.0.0/8 to OpenVPN clients that need to connect to VINI. On the other hand, ORBIT nodes use class A addresses to communicate between the management console and to access other ORBIT services. Thus, connectivity to the ORBIT nodes running OpenVPN clients (i.e., ORBIT nodes) during integration was lost. This problem was alleviated by assigning address from 172.16.0.0/12 and 192.168.0.0/16 to OpenVPN servers and clients, which are private IP ranges and will not conflict with standard ORBIT or PlanetLab IP ranges. Some other modifications that were made included enabling IP packet forwarding and ARP responses on UML instances, both of which were disabled in VINI by default.

Overall, the aforementioned modifications accommodate both ingress and egress flows and create a virtual Ethernet link between the UML eth3 interface and the interface on ORBIT nodes without Click having to know the details of the traffic flowing between the two points. Figure 6 summarizes the key modifications on the VINI architecture.

4.3.2 ORBIT Modifications

An ORBIT baseline image was used as a foundation for building the ORBIT node with router or bridge functionality. The baseline image was Debian GNU with Linux kernel 2.6.12. OpenVPN was compiled and installed along with OpenSSL [15]. The Linux kernel was recompiled with the Tunneling and Bridging options to enable the creation of Tap interfaces for OpenVPN and to allow the operation in bridging mode. Bridgeutils was also compiled and installed to provide the node with the proper tools to create, modify and delete bridges.

The Bridge configuration (described in Section 4.2) is accomplished using the Linux bridge tool, brctl (see Figure 7). A TAP interface, tap0 is created using the Linux 'mknod' command so that OpenVPN can use that device to pass packets. A bridge interface, br0 is then created and linked to the tap0 interface forming a virtual Ethernet bridge. An IP address is assigned to br0 to provide the OpenVPN link endpoint with an IP addresses. Thus, the UML eth3 interface on VINI (shown in Figure 6) is linked to tap0 on ORBIT, which is bridged to ath0, the wireless interface. Packets sent out of the eth3 interface will go onto the ORBIT wireless network.

In the Router configuration, a tap0 device is again created and linked to a bridge, br0 device (see Figure 8). An IP address (and optional multiple aliases) are assigned to br0 and traffic is routed between br0 and other interfaces (e.g., wireless interface ath0).

For both Bridge and Router configurations, the VINI scripts are set to automatically generate commands for the ORBIT control framework to image, power on, and configure the ORBIT Bridge and Router nodes. Thus, automatic topology creation during experiments is facilitated.

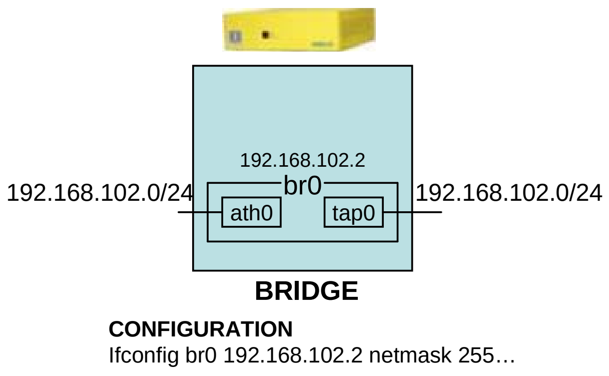
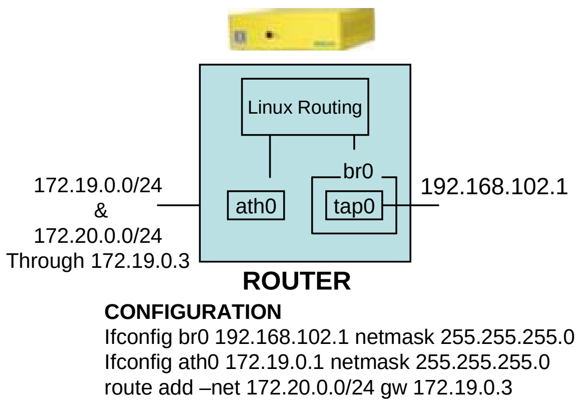


Figure 7. Example of Bridge IP configuration.



```
Ifconfig br0 192.168.102.1 netmask 255.255.255.0
Ifconfig ath0 172.19.0.1 netmask 255.255.255.0
route add –net 172.20.0.0/24 gw 172.19.0.3
```

Figure 8. Example of Router IP configuration.

5. INTEGRATED MOBILITY SCENARIO

In this section we provide proof-of concept of the integrated architecture by deploying a Layer 3 experiment over PlanetLab-VINI and ORBIT. The objective of this experiment is to investigate hand-off issues between access points that may belong to different Internet Service Providers (ISPs). Although mobility support is a requirement in the Internet, IP does not properly addresses its specific needs. Designs that have been proposed to better support mobility use some form of triangular routing (e.g., Mobile IP). In this experiment, we investigate ways to simplify hand-off by removing triangular routing.

5.1 Experiment Description

Figure 9 represents the topology that is used in the experiment. Three VINI nodes are physically located in Berkeley, California Tech, and MIT, and communicate with each other via UDP tunnels. A Video Server is linked to the California Tech VINI node by using an ORBIT node with the Bridge configuration, as previously described. We then attach two access points, A and B, to the other VINI nodes and configure them in the Router mode.

The ath0 wireless interfaces on the four ORBIT nodes are configured with 172.16.X.X IP addresses and are assigned proper channel, essid, and frequency. The tap0/br0 interfaces are configured with 192.168.X.X IP addresses. OSPF is utilized to automatically set up connectivity between nodes.

The overlay connectivity of the integrated testbed can be shown by performing *traceroute* from the Mobile client (node1-2), to Access Point B (node1-3):

```
node1-2.sb2.orbit-lab.org:~# traceroute 192.168.103.2

traceroute to 172.16.1.2 (172.16.1.2), 30 hops max, 40 byte packets

 1 node1-1. (172.16.0.1)  0.521 ms  0.495 ms  0.451 ms

 2 eth2.berkeley (192.168.107.1)  73.075 ms  103.113 ms  74.466 ms

 3 eth3.caltech (192.168.100.3)  86.964 ms  86.009 ms  103.015 ms

 4 eth3.mit (192.168.101.3)  167.395 ms  184.267 ms  170.087 ms

 5 tap0.node1-3 (192.168.103.2)  192.127 ms  177.791 ms 189.877 ms
```

The path between the Mobile Client and Access Point B is through access point A (node 1-1), Berkeley, California Tech, MIT, and onto the tap0 interface of Access Point B, which is the expected path. Thus, it can be seen that in addition to ORBIT-to-ORBIT node connectivity in the 172.16.0.0/12 range, the ORBIT nodes have access to the VINI nodes on the 192.168.0.0/16 network.

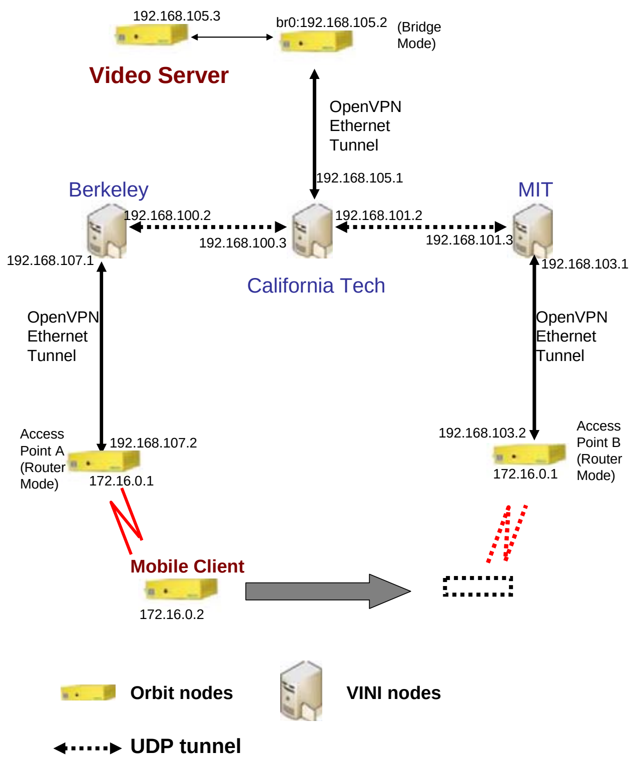


Figure 9. Integrated architecture to test hand-off in a mobility scenario.

The execution steps of the experiment are as follows; initially, video is streamed from the Video Server to the Mobile Client through access point A. As the Mobile Client moves away from access point A connectivity breaks and the video freezes. Access point A senses that the link is broken and through OSPF advertises that it is no longer in the routing path of the Mobile Client. That update is propagated and reflected in all the nodes. Meanwhile, the Mobile Client establishes connectivity to the network via access point B, which in turn advertises the new link to the Mobile Client. Once the new routing information is propagated in the network the video is restored.

It is important to note that this investigation is different from Mobile IP because in the case of mobile IP access point A would relay the packets of the Mobile Client to access point B after the hand-off. In this scenario, the packets do not traverse the Berkleley and access point A overlay nodes and do not get redirected to access point B. They instead go through California Tech, MIT, and access point B and are delivered directly to the Mobile Client.

5.2 Experimental Results

Initial results of our measurements with the Iperf tool have shown that the packet delay with Mobile IP is 443 msecs as compared to 225 msecs with our setup. Furthermore, by manually triggering the change in the routing paths when the connectivity of the Mobile Client between access point A and B changed, we have found that it takes approximately 3.5 seconds for the new routes to propagate in the network and for the video to get restored.

Figure 10 shows the throughput from the Video Server to the Mobile Client with different offered loads. With an offered load of 50 Mbps, the deviation in throughput is higher. However, the average throughput for both offered loads tends to be approximately the same. These values can be justified after looking at the throughput characteristics of the OpenVPN links in Table 1.

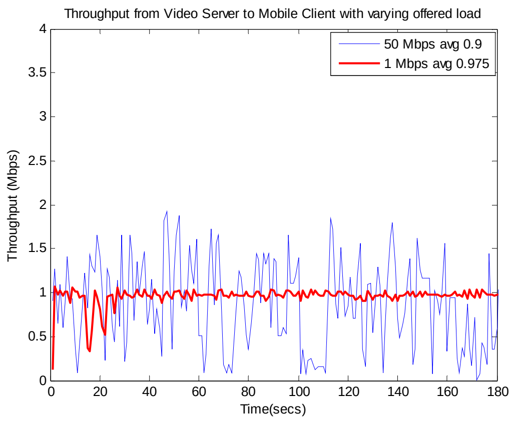


Figure 10. Throughput characteristics of received video.

The OpenVPN link between the Video Server and California Tech has a throughput of 1.25 Mbps inferring that it is the bottleneck of the video connection. It is therefore worthwhile to note that the selection of links during integration is important as it may dictate the results of an experiment. In addition, the throughput variation indicates the need for QoS guarantees over virtual overlay links to accommodate experiments that are delay sensitive. Currently, there are no standard Quality of Services (QoS) guarantees on the PlanetLab nodes for bandwidth, IO or CPU time. The GENI effort aims to address this issue [17].

Table 1. Throughput, delay, and jitter on OpenVPN links

OpenVPN LINKs	**THROUGHPUT Mpbs**	**DELAY usec**	**JITTER Usec**
Access point A to Berkeley	4.01	83	3.0
Access point B to MIT	4.25	99	2.9
Server bridge to California Tech	1.25	19	7.5

6. DISCUSSION

The integration lessons from ORBIT and VINI can be generalized and applied to the integration of other wired and wireless testbeds. The utilization of Ethernet tunnels instead of IP tunnels has allowed the support of non-IP protocols and broadcast packets for network layer experimentation.

While VINI provides a powerful platform to create controlled network topologies, it's automatically generated underlying configuration files require intimate knowledge of VINI's inner workings. As previously described, VINI links together Click modular software router, UML, UML Switch, XORP BGP/OSPF routing software and OpenVPN using a system of Unix sockets, UDP sockets and Linux Tap/Tun interfaces. By carrying Ethernet traffic to the user space, we allow users with basic knowledge in Linux and Linux networking to use the integrated testbed without knowing the details of the lower layers of virtualization on the nodes (e.g., Click) or having to modify the underlying system. Therefore, transparency aids the user to expedite the deployment of an experiment.

PlanetLab nodes use virtualization to accommodate multiple users. The cost of providing Layer 3 experiments through virtual internetworking at the UML level is, however, a lower performance since forwarding data packets in the UML kernel incurs nearly 15% additional overhead [16].

As previously discussed in Section 3, it is not preferred to virtualize wireless nodes because users typically require access to the entire protocol stack to investigate lower layer issues (assuming a single wireless card per node). However, it is important to note that ORBIT nodes have 2 wireless interfaces, and we could potentially use two instances of UML and support two concurrent experiments by attaching an interface to each virtual machine. We chose not to provide concurrent support for two experiments per node as there is currently no control mechanism within UML to provide access to the lower layer stack. Access to the physical and MAC layers is typically needed in experiments that involve wireless nodes. We instead spatially divide the ORBIT grid such that each VINI-ORBIT experiment gets a portion of the wireless nodes. Experiments use orthogonal frequencies, which limits the number of concurrent experiments based on the number of orthogonal frequencies available on the 802.11a/b/g interfaces.

Eventhough the aforementioned proof-of-concept experiment was based on 802.11a/b/g, the ORBIT grid can also provide support for experiments involving heterogeneous wireless networks. In addition to 802.11a/b/g, a subset of ORBIT nodes are equipped with Zigbee, Bluetooth, and GNU radios. A researcher can therefore choose to integrate a wired testbed with groups of ORBIT nodes that use different wireless technologies.

Another aspect of integration is the unification of the control and management architectures of both wired and wireless testbeds. An ORBIT user wanting to run an integrated experiment can communicate with a centralized control and management infrastructure that allocates to the user the Orbit grid for a particular time slot, while at the same time communicates with VINI nodes and set ups the required authentication information needed for OpenVPN tunnel creation. In addition, it can authorize the ORBIT user to use specific slivers on the VINI nodes. On the other hand, a VINI user wanting to run an integrated experiment can communicate with a control and management plane that allocates ORBIT nodes for the user as well as copy the required authentication keys to enable OpenVPN tunnel creation on the ORBIT nodes. Therefore, there is a need for an identity-management federation across both testbeds. In order to allow for this integration from the perspective of an ORBIT user, we extended the ORBIT control framework to include VINI nodes through the use of a single programming script and experimental methodology. This unified control and management architecture is currently missing from the perspective of a VINI user.

7. CONCLUSION

In this paper, we presented an integrated architecture solution that enables network layer experiments over wired and wireless networks on existing Internet links with realistic background traffic. Our solution provides an abstraction of the underlying software architecture that simplifies the configuration complexity of setting up experiments. In addition, it supports non-IP traffic and broadcast traffic, as well as any- to-any host connectivity. The usefulness of this integrated architecture was demonstrated through the investigation a video hand-off design as an alternative to Mobile IP.

The results of the research discussed in this paper provide a valuable basis on which researchers can build and prototype mixed wired/wireless experiments. Such a platform will assist in developing next generation protocols that address many core issues such as mobility and location-directed broadcasts over extensive network architectures. The results of these experiments may one day power the next generation of network devices designed for the wireless information age.

8. ACKNOWLEDGEMENT

We would like to thank Andy Bavier and Jennifer Rexford at Princeton University for their help with VINI. This material is based upon work supported by the National Science Foundation under Grants No. CNS 0335244 and CNS 0627032.

9. REFERENCES

[1] "PlanetLab: An open platform for developing, deploying, and accessing planetary-scale services," https://www.planet-lab.org/, available May 15, 2007.

[2] "Understanding VINI" , https://www.vini-veritas.net/documentation/pl-vini/user/understand, available May 10, 2007.

[3] A. Bavier, N. Feamster, M. Huang, L. Peterson, and J. Rexford, "In VINI Veritas: Realistic and Controlled Network Experimentation," in *ACM SIGCOMM*, Vol. 36, No. 4, pp. 3-14, October 2006.

[4] "Emulab – Network Emulation Testbed," http://www.emulab.net/, available May 20, 2007.

[5] B. White, J. Lepreau, and S. Guruprasad, "Lowering the barrier to wireless and mobile experimentation," *ACM SIGCOMM Computer Communications Review*, Vol. 33, pp. 47-52, 2003.

[6] D. Raychaudhuri, et al., "Overview of the ORBIT radio grid testbed for evaluation of next-generation wireless network protocols," in *IEEE WCNC*, vol. 3, pp. 1664-1669, 2005.

[7] H. Lundgren, D. Lundberg, J. Nielsen, E. Nordstrom, and C. Tschudin, "A large-scale testbed for reproducible ad hoc protocol evaluations," in *IEEE WCNC*, vol. 1, pp. 412-418, 2002.

[8] R. Karrer, A. Sabharwal, and E. Knightly, " Papers from Hotnets-II: Enabling large-scale wireless broadband: the case for TAPs," in *ACM SIGCOMM Computer Communication Review*, Vol. 34, pp. 27-32, 2004.

[9] *WHYNET*, http://pcl.cs.ucla.edu/projects/whynet/, accessed on 12/05/2005.

[10] "User-Mode Linux," http://user-mode-linux.sourceforge.net/, available May 5, 2007.

[11] "XORP: Open Source IP Router," http://www.xorp.org/, available May 17, 2007.

[12] "Click Modular Router," http://pdos.csail.mit.edu/click/, available May 20, 2007.

[13] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek, "The Click modular router," ACM Transactions on Computer Systems, vol. 18, pp. 263–297, August 2000.

[14] "OpenVPN: An open source SSL VPN solution," http://openvpn.net/.

[15] 'OpenSSL: The Open Source toolkit for SSL/TLS," http://www.openssl.org/, available May 19, 2007.

[16] X. Jiang and D. Xu, "Violin: Virtual internetworking on overlay infrastructure," in *Proc. International Symposium on Parallel and Distributed Processing and Applications*, pp. 937–946, 2004.