

# Lightweight Operating-System support for a scalable and robust virtual network infrastructure

Sapan Bhatia

Marc Fiuczynski

Andy Bavier

Larry Peterson

PlanetLab [3] is a platform for planetary-scale services that has been used very successfully by distributed systems researchers. Multiple simultaneous experiments are contained within separate “slices”, which on each machine is implemented as a lightweight, Linux-VServer based VM [4]. Linux-VServer isolates services with respect to the filesystem, memory, CPU and bandwidth. However, it does not provide complete virtualization of the network; rather, all slices on a node share the same IP address and port space.

VINI [1] is a virtual network infrastructure that lets researchers evaluate new protocols, routing schemes, and overlay services in a realistic, controlled environment. When complete, the infrastructure will support multiple concurrent experiments connecting to multi-Gbps links, and give researchers full control over network topologies and link characteristics, packet forwarding, and routing protocols. VINI is a small step towards GENI, which will allow experimentation with new network architectures to overcome the Internet Impasse [2]. VINI currently consists of 36 nodes deployed at 22 sites in National LambdaRail and Internet2 networks.

The starting point for VINI was the MyPLC software distribution. MyPLC enables anyone to create their own “private” PlanetLab installation, running the same software as the public PlanetLab but distinct from it. A key drawback of this approach to building VINI was the lack of network virtualization in PlanetLab; VINI must support simultaneous experiments running on different virtual network topologies. We propose writing a work-in-progress report describing our work on new OS-level network virtualization and virtual topology management techniques for VINI.

Implementing network virtualization in the context of Linux-VServer presents several challenges. First, we need to fully virtualize the OS networking stack to give each experiment full control over its own route table and a set of virtual devices, and allow it to insert traffic shapers and packet filters without affecting other traffic. Second, the overhead of such virtualization must be lightweight to support high throughput, as well as to scale to dozens of concurrent “virtual-router” experiments on a single node—a typical PlanetLab node has 40-50 active experiments and we expect similar utilization for VINI. Finally, the capabilities of a virtual router must be policed to restrict its control over the physical infrastructure to the scope of the experiment. For example, one virtual router should not hijack/sniff packets

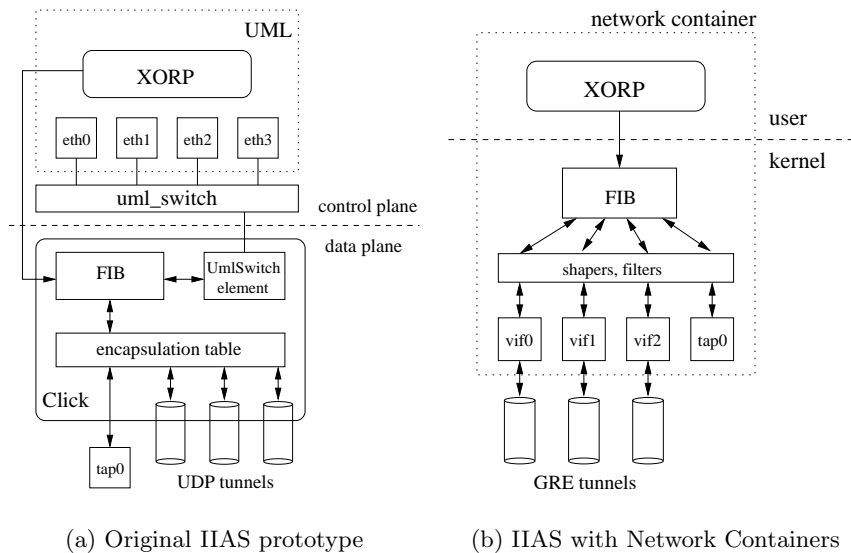
belonging to another virtual router.

Our approach to OS-level virtualization integrates the Linux community’s NetNS work with the Linux-VServer virtual machines currently in use by PlanetLab, to provide “Network Containers” within slices. Our paper will discuss how this approach overcomes the challenges listed above while highlighting three contributions.

Our first contribution is to *make a case for full fledged Network Containers implemented as virtualized network stacks within Linux*. The current approach to virtualize the network in Linux-VServer are based on IP, UDP, and TCP multiplexing and demultiplexing. Although this arrangement lets clients and servers function natively without requiring intermediate address translation, it does not let services change packet route in the kernel, send control messages to the NIC device driver, or add new virtual interfaces and packet filters. To virtualize the OS network stack involves contextualizing global variables and data structures used to represent the route table, packet filters, and queuing disciplines such that they are instantiated separately for each Linux-VServer VM. We will present the performance impact of our virtual network stack implementation and compare it with other approaches. Our results show negligible degradation in throughput, and a modest increase in jitter relative to vanilla Linux that may be reduced or eliminated by tuning the CPU scheduler.

Our second contribution is *a model for using Network Containers to create virtual topologies on VINI*. In our model, a VINI slice has one or more Network Containers, each holding a set of virtual devices and a copy of the kernel’s IPv4 forwarding table that can route packets between these devices. At a lower layer each virtual device in a Network Container maps onto a GRE tunnel endpoint to implement the virtual topology. The full range of traffic shapers and packet filters implemented by Linux can be used to alter aspects of the virtual links. Software running in the slice can add or remove routes from the container’s in-kernel forwarding table.

Figure 1 shows how our model simplifies the “Internet In A Slice” (IIAS), a prototype VINI experiment we described in [1]. Figure 1(a) shows the original design, with all data plane forwarding done in user space by Click. The virtual topology for the experiment was implemented by UDP sockets. However, routing software like XORP operates on network interfaces; for this reason we ran XORP inside UML and used Click to map network interfaces inside UML to the UDP sockets that



**Figure 1: Internet In A Slice Evolution**

form the virtual links. Click modules also implement bandwidth shaping, and can introduce latency and loss on the virtual links. This design allowed us to implement an entire virtual network inside a slice on the current PlanetLab. However, performance was poor since every packet was forwarded in user space.

In contrast, Figure 1(b) shows how leveraging Network Containers vastly simplifies our IIAS experiment architecture. Now XORP can run directly inside the VServer. XORP sees a set of network interfaces and writes entries into a forwarding table in the Linux kernel. Virtual link characteristics like bandwidth, latency, and loss are emulated using the Linux kernel’s built-in capabilities, e.g., a Hierarchical Token Bucket for traffic shaping. Packets sent to a virtual interface in the Network Container are forwarded to the appropriate GRE tunnel; likewise, packets received from a tunnel arrive in the container on the corresponding virtual interface.

Our third contribution is *an extensible method of instantiating and modifying virtual topologies*. We provide VINI users with a simple language for specifying topologies. Specifications written in this language are fed to a compiler, that in turn issues requests to a virtual network manager daemon. VINI developers can easily extend the language, compiler and daemon by plugging Python scripts into our framework. This allows easy prototyping of new models for building virtual topologies using Network Containers, for example by introducing new tunneling mechanisms (e.g., VLANs or MPLS circuits instead of GRE) or new interactions between containers (e.g., layering containers to do multi-level routing). Compiling a specification causes FIFO files to appear in the slice that can be used to instan-

tiate or invoke specific actions on the topology. These FIFOs, and the operations that they perform, are only visible in this slice; no other slices can modify or otherwise interact with this topology.

In conclusion, we would like to present the current state of the implementation of a VINI node, emphasizing on the underlying OS support, and bringing out design issues that we have resolved or are currently resolving.

## 1. REFERENCES

- [1] A. Bavier, N. Feamster, M. Huang, L. Peterson, and J. Rexford. In *vini veritas: Realistic and controlled network experimentation*. In *SIGCOMM’06: Proceedings of the ACM SIGCOMM 2006 Conference*, September 2006.
- [2] L. Peterson. Overcoming the internet impasse through virtualization. In *HotNets-III: Proceedings of the 3rd Workshop on Hot Topics in Networks*, November 2004.
- [3] L. Peterson, A. Bavier, M. Fiuczynski, and S. Muir. Experiences building PlanetLab. In *OSDI’06: Proceedings of the 7th USENIX Symposium on Operating System Design and Implementation*, November 2006.
- [4] S. Soltesz, H. Poltz, M. Fiuczynski, A. Bavier, and L. Peterson. Container-based operating system virtualization: A scalable, high-performance alternative to hypervisors. In *EuroSys’07: Proceedings of the 2nd ACM EuroSys Conference*, March 2007.