

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

Sapan Bhatia

Marc Fiuczynski

Andy Bavier

Larry Peterson

We are building VINI [1] to be a virtual network infrastructure that lets network researchers evaluate new protocols, routing schemes, and overlay services in a realistic, controlled environment. The infrastructure is intended to support multiple concurrent experiments connecting to multi-Gbps links, which presents several challenges at the OS-level. First, we need to fully virtualize the OS networking stack giving researchers full control over route tables, traffic shapers, packet filters, etc.. In this way they can explore building new control planes and fast data planes in order to overcome the Internet Impasse [2]. Second, the overhead of such virtualization must be lightweight to a) support high throughput and b) scale to dozens of concurrent 'virtual-router' experiments on a single node (as is the expected workload on GENI). In this way virtual routers can perform in a way consistent with their real analogs. 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. We propose to illustrate an implementation of VINI designed with these challenges in mind, focusing on its building blocks. Although this implementation is work-in-progress, a working prototype is operational at Princeton and we plan to deploy it on multiple public VINI nodes connected to the National Lambda Rail (see [www.vini-veritas.net](http://www.vini-veritas.net))

PlanetLab [3] is a platform for planetary-scale services that has been used very successfully by distributed systems researchers. Multiple simultaneous experiments are each contained within separate 'slices', which on each machine are represented as a lightweight, Linux-VServer based VM [4]. Although Linux-VServer isolates services with respect to the filesystem, memory, CPU and bandwidth, it does not provide complete virtualization of the network, as required by VINI. The OS constructs and mechanisms to bridge this gap are the proposed focus of this poster. In the following paragraphs, we present the contributions we would like to illustrate.

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 go as far as changing packet routes, sending control messages to the NIC device driver, adding 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 have evaluated the performance impact of two virtual network stack implementa-

tions for client/server and routing workloads. Our results show negligible degradation in throughput and an increase in jitter relative to vanilla Linux, which can be reduced or eliminated by tuning the CPU scheduler.

Our second contribution is the *integration of virtualized network stacks with Linux-VServer and deployment of this OS onto VINI*. It is possible to create virtual topologies in VINI within which one can test new L3 protocols on virtual interfaces, leverage the kernels packet filters and bandwidth-shaping functionality, pass control messages to devices (e.g., iwconfig), and bring virtual devices up and down. Additionally, they can request virtual links to have a rich set of vetted link disciplines such as random packet drops, delays, etc.

Our third contribution is a *secure bootstrap implementation* of the VINI control plane: *i.e.*, the mechanism that lets virtual topologies be instantiated and managed. The use of a virtual network stack raises several security concerns, as services now need to be granted privileges to administer interfaces, open raw sockets etc. While *POSIX capabilities* is a strong candidate for a long term solution, their current implementation in Linux is immature. We have implemented an interface that lets services running in a VINI "slice" to request changes in the virtual network. For example, functions like the definition of a virtual link, its embedding on the physical infrastructure and the definition of encapsulation strategies (such as tunneling or bridging) are requested through a virtual network manager daemon. Requests for a virtual topology are defined in a simple declarative language that we have developed for this purpose. A specification written in this language is fed to a tool that issues the corresponding request to the daemon.

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

- [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.