

# Operating System Support for Planetary-Scale Network Services

Bavier, et al (Princeton and elsewhere), NSDI04

Context: System Structure

## Introduction

PlanetLab:

How to build a distributed infrastructure for evaluation and deployment of network services?

200 machines, 80 sites, 15 countries.

Supports a number of research projects.

<http://www.planet-lab.org/>

A number of potential aspects to look at the project. This paper looks at operating system support for network services in this environment.

## Relationships

1. Shared control of PlanetLab machines between PlanetLab administration and administration at local sites.
2. Users create a *slice*, which allows resource access on a distributed set of machines. Technique called *distributed virtualization*.
3. PlanetLab must support for researchers multiple, parallel services. Principle of *unbundled management*.
4. Rest of the Internet needs to feel safe from PlanetLab—potentially disruptive experiments could lead to site or more widespread problems.

Paper talks about the resulting “design” and “experiences” of PlanetLab resulting from these relationships.

## Distributed Virtualization

Slice—set of services on constituent nodes.

Setup `ssh` keys to allow access to each node.

There is a bootstrap slice on each node used to create other slices.

Slices must be isolated (protected) from one another.

Must partition resources between slices.

Must isolate PlanetLab from outside world:

- resource accounting
- audit resource and *action* use

From experience, security generally not a problem because of malicious use, but legitimate experimental uses can and have triggered outside concern.

## Unbundled Management

OS supports only local abstractions while global, network-wide abstractions are implemented by infrastructure services.

No application or service is privileged over another, save for a special control slice.

Practical expectations (*reasonable!*)

- developers do not want to port applications to a new API
- expect a full-featured Operating System, not minimalist one.

## Design Challenges

1. Virtual machine abstraction for slices. Full virtualization like VMware is too costly in terms of performance. Paravirtualization like Xen—“promising technology for PlanetLab.”  
Instead virtualize at the system-call level—“reasonable assurance of isolation”.
2. Isolate virtual machines. Want to avoid potential *crosstalk*, which causes problems between applications.
3. How to provide low-level access to virtual devices such as the network. PlanetLab uses a “safe” version of raw socket interface. Somewhat like Exokernel approach.
4. Distributed coordination of resources—is a finer grain problem than grid computing platform.
5. Monitoring and management of a large distributed infrastructure. It’s more than just network or host management. Home-brewed approach.

## PlanetLab Operating System Details

Linux 2.4 kernel.

Slice is a set of virtual machines, each VM is a vserver—*like a guest Operating System in Xen?*

Idea of *chroot()* to change the root of the file system—similar idea for other resources. Each vserver has a weaker form of root privilege.

Slices share port number and machine address space—*cannot have more than one service at a port across all vservers.*

Processes in a vserver inherit a security context, which is passed as part of a system call.

## More PlanetLab Operating System Details

Login name = slice name, use modified `bash` shell.

Lots of sharing of files between vservers—much more sharing than full virtualization approach.

Special vserver contexts for local admin and node mgmt

Point out weaker isolation and more QoS crosstalk than other VMM approaches.

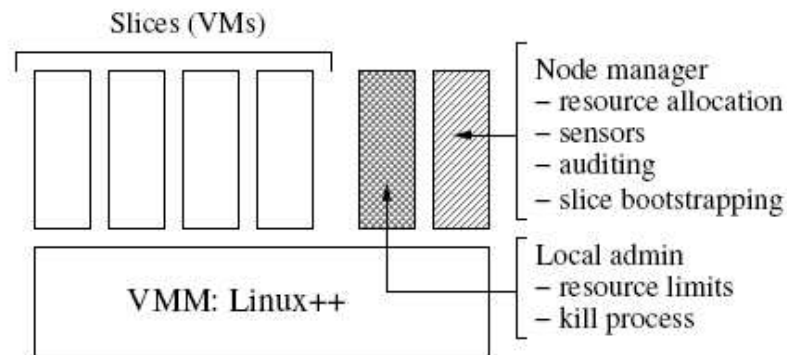


Figure 1: PlanetLab Node Architecture

## Resource Allocation

Resources have been used up by one slice at the exclusion of other slices—file descriptors, disk space, CPU.

Node manager generates resource capabilities (rcaps) to limit the total quantity of a resource that can be allocated. “Trusted services” can use *acquire()* operation to obtain an rcap.

Rcaps are bound to a vserver at slice creation time with *bind()* operation.

System calls have wrappers to intercept allocation requests.

## More on Resource Allocation

Hierarchical Token Bucket mechanism to cap total outgoing bandwidth of a node and vserver—*concern for impact on non-PlanetLab machines.*

CPU scheduling done with resource containers so each vserver has some number of shares. Provides CPU guarantees and fairness.

PlanetLab Central (PLC) is a trusted infrastructure service for globally bootstrapping slices. Updates a table that is downloaded by nodes every 10 minutes—*not immediate.* Will this centralized approach work long term??

## Safe Raw Sockets

Raw sockets allows for protocol testing. Safety ensured by binding to a TCP/UDP port and local addresses of outbound packets are verified.

A raw sniffer is available to capture packets on a safe socket. Can be passed to tcpdump.

Scriptroute service uses it to send a variety of packet types.

## Monitoring

Good monitoring is clearly needed.

Means of a low-level sensor interface to export data from OS and network. Two types: snapshot and streaming.

Sensor server is HTTP-compliant server with a URL to request data—*hierarchical like an SNMP MIB*.

Export wrappers around `/proc` file system.

Monitoring services to use this interface such as a rule-based approach.

Also traffic auditing.

Mixed experience with getting externally detected issues to the responsible node.

Also allow *actuators* to control rather than just read sensors.

## Evaluation

Vserver scalability due to a lot of sharing amongst vserver file systems.

Slice creation-0.15 seconds. Vserver creation takes 9.66 seconds on average—*takes time, but how often are these operations done??*

Timing on service initialization, but not clear on significance.

*Not a lot on performance evaluation here.*

## Summary

Not a lot on evaluation, but more a design and experience report.

Large distributed platform.

Used for many network experiments.

WPI looking to join.