PlanetLab Architecture
(PlanetLab OS)
Key Ideas

• Shared infrastructure
  – distributed virtualization
  – slice abstraction
    ▪ set of virtual machines
    ▪ initialized with boot state

• Many groups contributing to its definition
  – infrastructure services
  – unbundled management
Requirements

• Underspecified slice abstraction
  – bootstrap slice creation service
  – minimal programming environment (no tunnels, no Java)

• Slice isolation
  – allocate/schedule node resources, w/ hard guarantees
  – partition shared address spaces
  – stable programming environment (no root access)

• Isolate PlanetLab
  – limits on resource consumption
  – audit resource usage
Requirements (cont)

- **Unbundled management**
  - OS defines only local (per-node) behavior
    - global (network-wide) behavior implemented by services
  - multiple competing services in parallel
    - shared, unprivileged interfaces
  - what privileged services are required?

- **Get it running yesterday with familiar API**
  - start with Linux and incrementally transform
Design Challenges

• Virtualization on each node: at what level?
  – hypervisors (e.g., VMWare)
    ▪ don’t scale well
    ▪ don’t need multi-OS functionality
  – paravirtualization (e.g., Xen, Denali)
    ▪ not yet mature
  – virtualize at system call interface (e.g., Jail, Vservers)
    ▪ reasonable compromise
    ▪ doesn’t provide the isolation that hypervisors do

• Isolating virtual machines
  – borrow scheduling mechanisms from MM systems
  – control: global/competing vs local/cooperative
Design Challenges (cont)

• Access to devices (e.g., Exokernel, Nemesis)
  – must support shared access
  – global services more important than local control

• Distributed coordination of resources
  – batch jobs vs continuous running services

• Management
  – existing tools targeted at single-organization
Virtual Machine

• Vserver: virtualizes at system call interface
  – each vservisor runs in its own security context
    ▪ private UID/GID name space
    ▪ limited superuser capabilities (e.g., no CAP_NET_RAW)
  – uses chroot for file system isolation
  – scales to 1000 of vservers per node (29MB each)

• Node Manager
  – privileged security context
  – interface for creating virtual machines
  – performs admission control

• Local admin context
  – set site limits (e.g., bandwidth)
Resource Allocation

• Interface (node manager)
  – rcap ← acquire(rspec)
  – bind(rcap, sliceid)

• Implementation (kernel)
  – link
    ▪ per-node cap
    ▪ fair allocation
    ▪ hard guarantees
    ▪ rate-control specific packets (e.g., ICMP)
  – processor
    ▪ proportional share scheduler

• Bootstrap slice creation service
  – trusted slice
Slice Creation

• PlanetLab Central (GUI)
  – users
    ▪ establish ssh keys
  – institution’s PI
    ▪ select slice name; e.g., princeton_597a
    ▪ assign users to slice
    ▪ instantiate slice
      – select set of machines (resource discovery)
      – set per-node rspec

• On each node
  – call acquire/bind
  – 10sec to create empty slice
Safe Raw Sockets

• Standard
  – privileged operation
  – access to all packets to/from host

• Safe version
  – bound to a specific UDP/TCP port
  – ensure that outgoing packets do not spoof
  – related ICMP packets

• Uses
  – ScriptRoute
  – user-level protocol stacks
Monitoring Services

• Serve several purposes
  – discover/select resources for a slice
  – monitor node/network health
  – measure/monitor Internet activity

• Exploit sensors
  – local state (/proc) + local view of the network (ping)
  – http://localhost:33080/nodes/ip/name

• Multiple services being built
  – Sophia: distributed Prolog engine
  – PIER: distributed SQL query processor
  – IrisNet: XML-based queries