lecture 16:
virtualization
5590: software defined networking

anduo wang, Temple University

TTLMAN 401B, R 17:30-20:00

virtual testbed

Thomas Anderson., et al. "Overcoming the Internet Impasse through Virtualization"

architectural barnacles

multi-provider nature — (lack of) consensus

- difficult to achieve consensus
- (achieving consensus) remove competitive advantage from architectural innovation

as result …

- ad hoc workarounds
- impair the Internet's long-term *flexibility, reliability, and manageability*

overcoming the impasse

requirements

- experiment easily with new architectures on live traffic
- plausible deployment path
- comprehensive, supporting broad range of architectural problems

limitation of existing approaches

testbeds

- lease lines connecting a limited set of locations
  - production testbed (real traffic/user): conservative in experiments
  - production testbed (no real traffic): adventurous experiments, less viability

overlay

- creating and maintaining overlay is straightforward
  - narrow fix to isolated solutions/functions
  - architectural tame

PlantLab

geographically distributed computing platform

- services and applications run in a slice of the platform
- slice: a set of nodes, exposing a fraction of its resources (VMs)

technical contribution

- distributed virtualization
  - acquisition of distributed set of VMs, forming a single, compound entity
- isolation

virtual testbed

an overlay substrate

- a set of dedicated but multiplexed nodes

a client-proxy mechanism

- a host uses the proxy to opt in to a particular experiment
- treats a nearby overlay node as the first-hop router

virtual testbed, recap

uses virtualization in two crucial ways

- client proxy + virtual link = a native network
- multiplexing overlay nodes creates many virtual testbeds that operate simultaneously

resolve …

- barrier-to-entry
- architectural limitation

PlanetLab

Larry Peterson., et al. “The design principles of PlanetLab”

Andy Bavier. , et al. “Operating Systems Support for Planetary-Scale Network Services”

PlanetLab design principles

a geographically distributed platform for deploying, evaluating, and accessing planetary-scale network services

deployed, started before fully understand what the architecture would be

↓

requirement: being able to evolve the system

↓

PlanetLab principles: co-evolve with the architecture itself

goals

early experiment with new ideas	platform to experiment with planetary-scale services
deployment of new services	platform for novel services to be deployed and serve a real user community
availability of a rich set of services	catalyze the evolution of the Internet into a service-oriented architecture

goals — subtle tensions

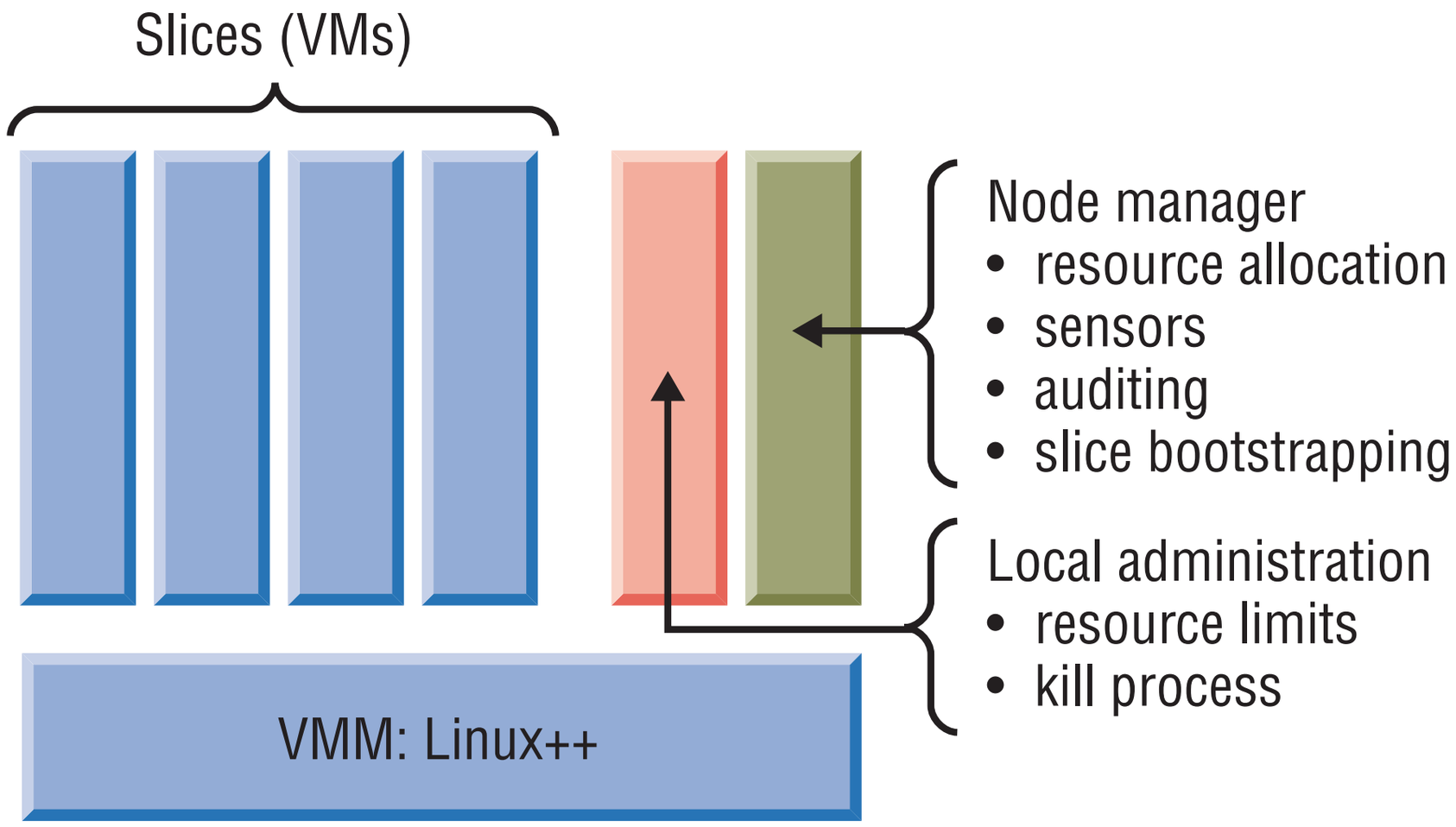
experiments with new ideas

- short-term experiments VS. continuously running services
- service by research community VS. production network
  - scalability, security, autonomy

balance point

- push the scalability, security, robustness, and decentralization
- ensure the evolution of the architecture is driven by the requirements of the running system

PlanetLab overview



PlanetLab services and applications run in a slice of the platform: a set of nodes on which the service receives a fraction of each node's resources, in the form of virtual machine (VM)

PlanetLab overview

slice

- a collection of VMs spread around the world
- VMs are implemented by virtual machine monitor (VMM)
- VMs are controlled by node manager

sliver

- the instantiation of a slice on a given node

control plane

- node manager + infrastructure service
  - example infrastructure service offers a interface for local site admin

PlanetLab principles

distributed virtualization

- *acquisition of a distributed set of VM, treated as a single, compound entity (slice)*
- slices are underspecified, minimizing the extent to which future users are constrained
  - how a slice's constituent VMs are connected (overlay via tunneling?)
  - what language or runtime (JVM software package)

unbundled management

- allow parallel infrastructure services to run their slices, evolving independently

isolation — isolating slices

allocate and schedule resources

- cycles, bandwidth, memory, storage …

partition and contextualize the available name space

- the network address, file names

provide a stable programming base

- prevent a slice (code running in that slice) from negatively affecting another slice

isolation — isolating PlanetLab

interaction between PlanetLab and the rest of the network must be attributable to a PlanetLab user	thoroughly account and limit resource usage and consumption
explicit trust relationship between node users, authorities, and slice user	audit resource usage

recap —

OpenFlow, FlowVisor, and PlanetLab