PlanetLab: Evolution vs Intelligent Design in Global Network Infrastructure

Larry Peterson
Princeton University
PlanetLab

- 780 machines spanning 360 sites and 40 countries
- Supports *distributed virtualization*
  each of 600+ network services running in their own *slice*
Slices
Slices
Slices
Per-Node View

Virtual Machine Monitor (VMM)
Global View

PLC
Long-Running Services

• Content Distribution
  – CoDeeN: Princeton
  – Coral: NYU, Stanford
  – Cobweb: Cornell

• Storage & Large File Transfer
  – LOCI: Tennessee
  – CoBlitz: Princeton

• Information Plane
  – PIER: Berkeley, Intel
  – PlanetSeer: Princeton
  – iPlane: Washington

• DHT
  – Bamboo (OpenDHT): Berkeley, Intel
  – Chord (DHash): MIT
Services (cont)

• Routing / Mobile Access
  – i3: Berkeley
  – DHARMA: UIUC
  – VINI: Princeton

• DNS
  – CoDNS: Princeton
  – CoDoNs: Cornell

• Multicast
  – End System Multicast: CMU
  – Tmesh: Michigan

• Anycast / Location Service
  – Meridian: Cornell
  – Oasis: NYU
Services (cont)

- Internet Measurement
  - ScriptRoute: Washington, Maryland
- Pub-Sub
  - Corona: Cornell
- Email
  - ePost: Rice
- Management Services
  - Stork (environment service): Arizona
  - Emulab (provisioning service): Utah
  - Sirius (brokerage service): Georgia
  - CoMon (monitoring service): Princeton
  - PlanetFlow (auditing service): Princeton
  - SWORD (discovery service): Berkeley, UCSD
Usage Stats

• Slices: 600+
• Users: 2500+
• Bytes-per-day: 4 TB
• IP-flows-per-day: 190M
• Unique IP-addrs-per-day: 1M
Two Views of PlanetLab

• Useful research instrument
• Prototype of a new network architecture
  – programmability and virtualization deep in the network

• This talk…
  – insights into the design process
  – technical lessons
  – operational lessons
Requirements

1) It must provide a global platform that supports both short-term experiments and long-running services.
   – services must be isolated from each other
   – multiple services must run concurrently
   – must support real client workloads
Requirements

2) It must be available now, even though no one knows for sure what “it” is. 
   – deploy what we have today, and evolve over time
   – make the system as familiar as possible (e.g., Linux)
   – accommodate third-party management services
Requirements

3) We must convince sites to host nodes running code written by unknown researchers from other organizations.
   - protect the Internet from PlanetLab traffic
   - must get the trust relationships right
4) Sustaining growth depends on support for site autonomy and decentralized control.
   – sites have final say over the nodes they host
   – must minimize (eliminate) centralized control
Requirements

5) It must scale to support many users with minimal resources available.
   – expect under-provisioned state to be the norm
   – shortage of logical resources too (e.g., IP addresses)
Design Challenges

• Minimize centralized control without violating trust assumptions.

• Balance the need for isolation with the reality of scarce resources.

• Maintain a stable and usable system while continuously evolving it.
Trust Relationships

Princeton  
Berkeley  
Washington  
MIT  
Brown  
CMU  
NYU  
EPFL  
Harvard  
HP Labs  
Intel  
NEC Labs  
Purdue  
UCSD  
SICS  
Cambridge  
Cornell  
...

Trusted Intermediary (PLC)

princeton_codeen  
nyu_d  
cornell_beehive  
att_mcash  
cmu_esm  
harvard_ice  
hplabs_donutlab  
idsl_psepr  
irb_phi  
paris6_landmarks  
mit_dht  
mcgill_card  
huji_ender  
arizona_stork  
ucb_bamboo  
ucsd_share  
umd_scriptroute  
...

...
1) PLC expresses trust in a user by issuing it credentials to access a slice
2) Users trust PLC to create slices on their behalf and inspect credentials
3) Owner trusts PLC to vet users and map network activity to right user
4) PLC trusts owner to keep nodes physically secure
Decentralized Control

- **Owner autonomy**
  - owners allocate resources to favored slices
  - owners selectively disallow un-favored slices

- **Delegation**
  - PLC grants tickets that are redeemed at nodes
  - enables third-party management services

- **Federation**
  - create “private” PlanetLabs
    - now distribute MyPLC software package
  - establish peering agreements
Virtualization

- Virtual Machine Monitor (VMM)
- Node Mgr
- Owner VM
- VM1
- VM2
- ... VMn
- Auditing service
- Monitoring services
- Brokerage services
- Provisioning services
- Linux kernel (Fedora Core)
  + Vservers (namespace isolation)
  + Schedulers (performance isolation)
  + VNET (network virtualization)
Active Slices

![Graph of Active Slices]

- **Slices with a process**
- **X-axis:** 05/May to 06/Mar
- **Y-axis:** 0 to 100

Legend:
- Min
- 1st Q
- Median
- 3rd Q
- Max
Resource Allocation

• Decouple slice creation and resource allocation
  – given a “fair share” (1/Nth) by default when created
  – acquire/release additional resources over time
    • including resource guarantees
• Protect against thrashing and over-use
  – link bandwidth
    • upper bound on sustained rate (protect campus bandwidth)
  – memory
    • kill largest user of physical memory when swap at 85%
Evolution vs Intelligent Design

- Favor design principles over a fixed architecture
- Let experience dictate what problems to solve
- Tactically…
  - leverage existing software and interfaces
  - keep VMM and control plane orthogonal
  - exploit virtualization
    - vertical: management services run in slices
    - horizontal: stacks of VMs
  - give no one root (least privilege + level playing field)
  - support federation (divergent code paths going forward)
    - minimize universal interface
“Top 10” Lessons

1) Work fast, before anyone cares
2) If you don’t talk to your university’s general counsel, you aren’t doing network research
3) From universal connectivity to gated communities
4) PlanetLab: We debug your network
5) Overlays are not networks
6) Critical mass comes first, then you can worry about scale
7) Build it and they (research papers) will come
8) Empower the user: yum
9) The first million, you have to steal
10) Inferior tracks lead to superior locomotives
Collaborators

• Andy Bavier
• Marc Fiuczynski
• Mark Huang
• Scott Karlin
• Aaron Klingaman
• Martin Makowiecki
• Reid Moran
• Steve Muir
• Stephen Soltesz
• Mike Wawrzoniak

• David Culler, Berkeley
• Tom Anderson, UW
• Timothy Roscoe, Intel
• Mic Bowman, Intel
• John Hartman, Arizona
• David Lowenthal, UGA
• Vivek Pai, Princeton
• Neil Spring, Maryland
• Amin Vahdat, UCSD
• Rick McGeer, HP Labs
CoDeeN (cont)

- Open proxies are abused
  - Series of security measures [Usenix 04, Usenix 06]
- DNS fails more frequently than expected
  - CoDNS: leverage peers [OSDI 04]
- Doesn’t scale for large files
  - CoBlitz: replicate “chunks” [NSDI 06]
- Internet routes fail
  - PlanetSeer: triangulate failures [OSDI 04]

Vivek Pai & KyoungSoo Park
CPU Availability

[Graph showing CPU availability over time with different quartiles and min/max values]
Scheduling Jitter

![Graph showing scheduling jitter with different RTT values and percentages.]

- Network
- Overlay w/SCHED_RR
- Overlay
Node Availability

A: Runup to NSDI ’05 deadline
B: After NSDI ’05 deadline
C: 3.0 rollout begins
D: 3.0 stable release
E: 3.1 stable release
F: 3.2 rollout begins
G: 3.2 stable release

Stable nodes (up > 30 days) ——
Active in last 30 days ——
Registered nodes ————

![Graph showing node availability and different milestones with timelines and counts.](image)
Memory Availability

![Graph showing Memory Availability](image-url)
Live Slices
Memory Availability

![Graph showing memory availability over time with lines indicating minimum, first quartile, median, third quartile, and maximum values.](image)
Bandwidth Out
Bandwidth In

![Bandwidth Graph]

- 1st Q
- Median
- 3rd Q
- Max
Disk Usage
1) PLC expresses trust in a user by issuing it credentials to access a slice
2) Users trust to create slices on their behalf and inspect credentials
3) Owner trusts PLC to vet users and map network activity to right user
4) PLC trusts owner to keep nodes physically secure

MA = Management Authority  |  SA = Slice Authority
Slice Creation

PI
SliceCreate( )
SliceUsersAdd( )

User/Agent
GetTicket( )

PLC (SA)

VM
NM

CreateVM(slice)

(redeem ticket with plc.scs)
Brokerage Service

User
BuyResources( )

Broker

PLC
(SA)

VMM

NM VM VM VM...

Bind(slice, pool)

(broker contacts relevant nodes)