Raven

A Provisioning Service for GENI

John H. Hartman
Department of Computer Science
University of Arizona

Scott Baker
SB Software

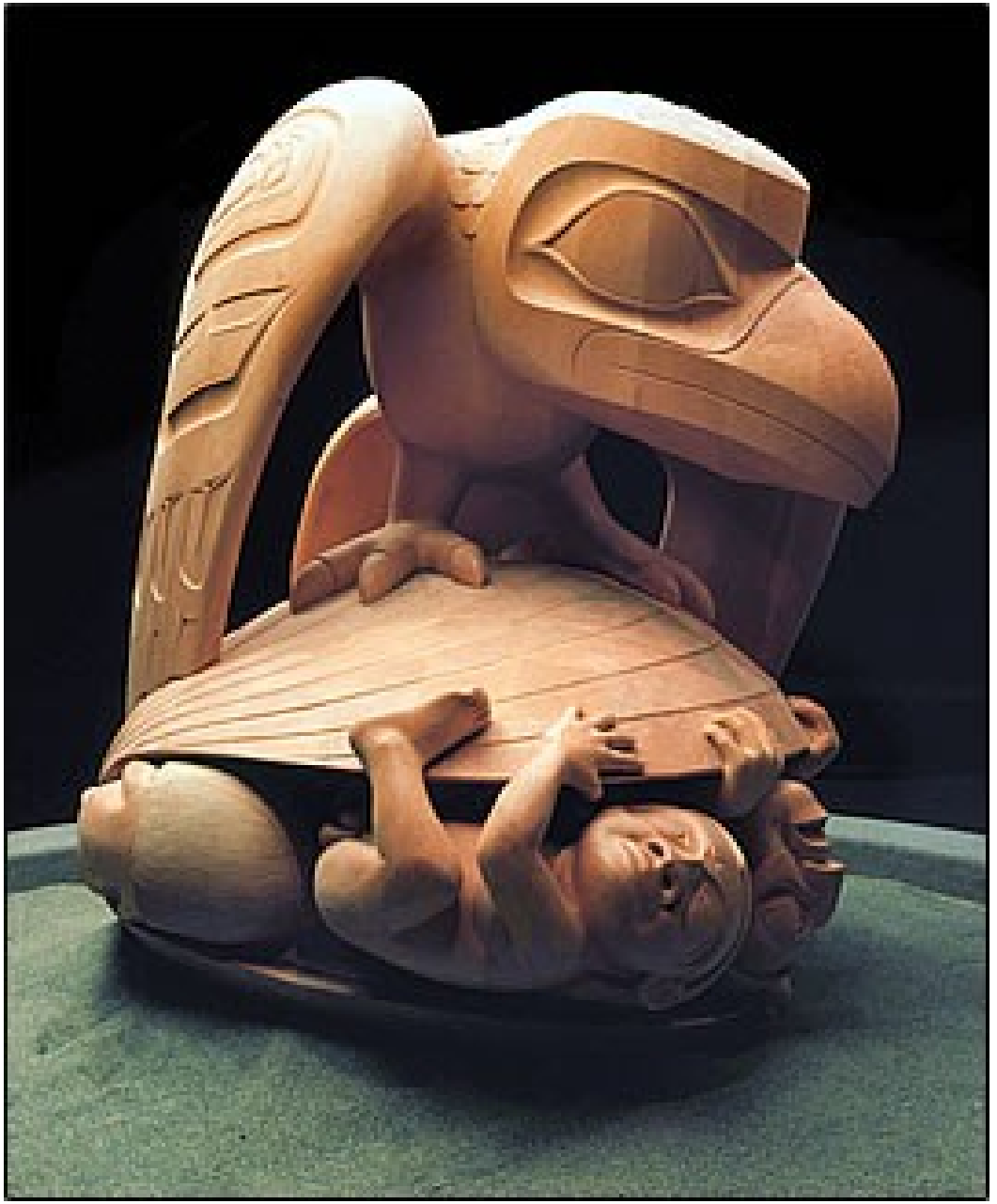
GENI Provisioning Service

- Provides what a GENI experiment needs to run:
  - Software
  - Runtime environment
  - Resources
- Slice management
- Configuration management
- Monitoring, data collection

Stork

- Distributed package management for PlanetLab
- Collective Package Management
  - Define groups of slivers
  - Specify package actions based on groups
- Secure installation
  - *Trusted packages file,* untrusted repository
- Efficient package transfers
  - FTP, HTTP, BitTorrent, CoBlitz, DOT/Set

Raven



Raven and the First Men, by Bill Reid
Museum of Anthropology, University of British Columbia

Cluster B PlanetLab

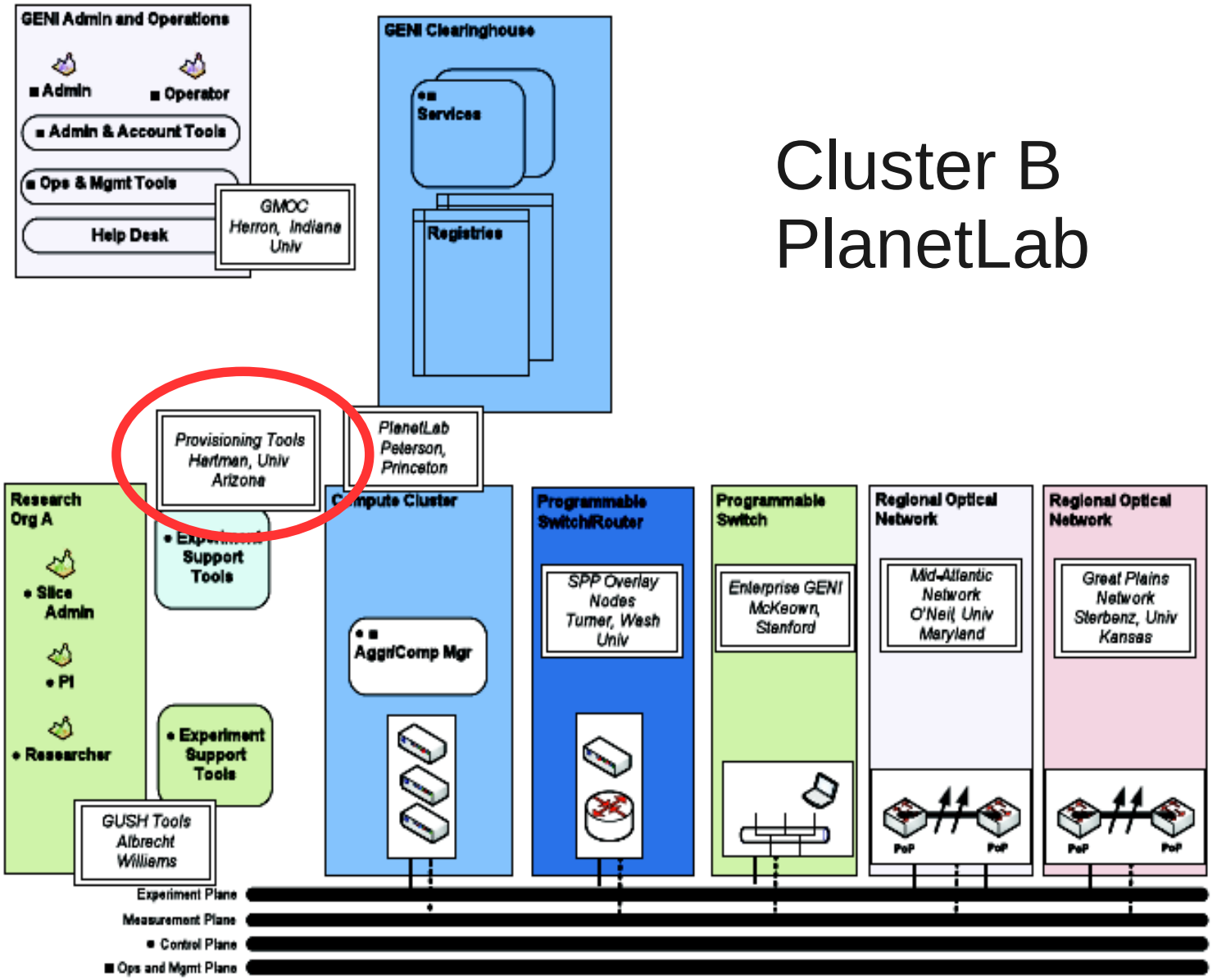


Figure 5-2. Cluster B utilizing PlanetLab control framework

Short-term Experiments

1. Develop software
2. Create slice(s)
3. Discover/select resources
4. Bind resources to slice(s)
5. Deploy software
6. Configure slivers
7. Start experiment
8. Monitor experiment, collect data

Long-term Experiments

- Develop software
- Create slice(s)
- Manage resources
- Manage software
- Configure slivers
- Monitor experiment
- Collect data

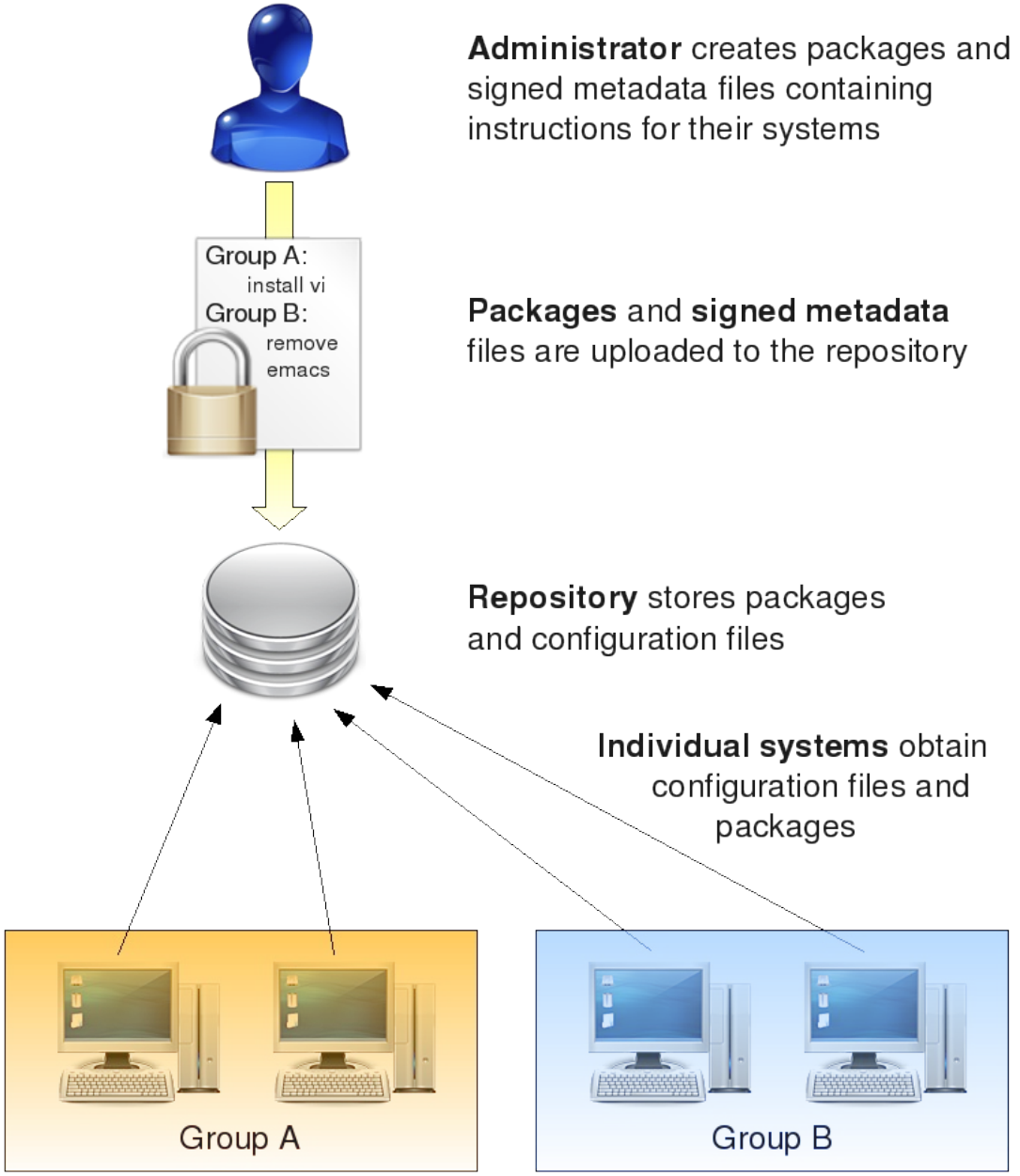
Raven Goals

- Software management (Stork)
  - Deploy software efficiently and securely
  - Collective package actions
  - Update software as experiment runs
- Slice management (Gush)
  - Dynamically select nodes
- More flexible security model

More Raven Goals

- Resource management
  - Resource allocation (Gacks)
  - Resource discovery (Sword?)
- Configuration management (?)
  - Configure slices of an experiment
  - Configure slivers of a slice
- Build environment == deployed environment
- Monitoring, collecting data (Gush/CoMon?)

Collective Package Actions



Administrator creates packages and signed metadata files containing instructions for their systems

Packages and **signed metadata** files are uploaded to the repository

Repository stores packages and configuration files

Individual systems obtain configuration files and packages

Individual systems **determine which instructions apply to them** and install or remove packages accordingly. Offline systems as well as new systems perform all actions when brought online

Specifying Package Actions

```
<PACKAGES>
<CONFIG GROUP="A">
<INSTALL PACKAGE="vi" VERSION="2.2"/>
</CONFIG>
<CONFIG GROUP="B">
<REMOVE PACKAGE="emacs"/>
</CONFIG>
<CONFIG>
<UPDATE PACKAGE = "firefox"/>
</CONFIG>
</PACKAGES>
```

Specifying Groups

```
<GROUPS>
<GROUP NAME="A">
<INCLUDE NAME="planetlab1.arizona.net"/>
<INCLUDE NAME="planetlab2.arizona.net"/>
</GROUP>
<GROUP NAME="B">
<INCLUDE SLICE="arizona_stork"/>
</GROUP>
<UNION NAME="Both" GROUP1="A" GROUP2="B"/>
</GROUPS>
```

Specifying Trusted Packages

Entities indicate trust in packages

```
<TRUSTEDPACKAGES>
<FILE PATTERN="stork-client-2.1.8-0.i386.rpm" HASH="9a1...ed" ACTION="ALLOW"/>
<FILE PATTERN="stork-config-2.1.8-0.i386.rpm" HASH="e4c...72" ACTION="ALLOW"/>
<FILE PATTERN="glibc-2.1.8-0.i386.rpm" HASH="179...a9" ACTION="ALLOW"/>
</TRUSTEDPACKAGES>
```

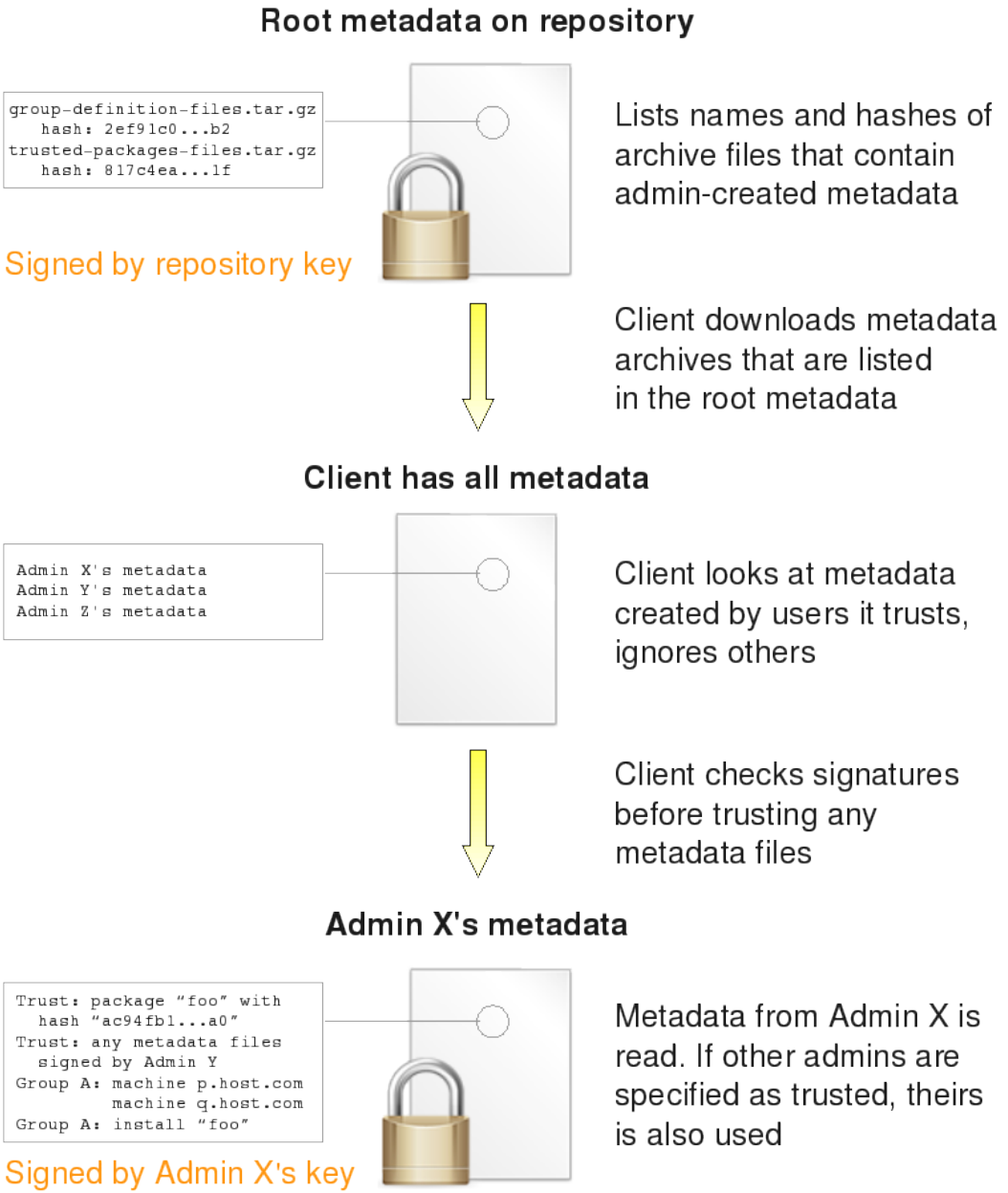
Entities delegate trust

```
<TRUSTEDPACKAGES>
<USER PATTERN="*" USERNAME="smbaker" PUBLICKEY="11S...nE" ACTION="ALLOW"/>
<USER PATTERN="*" USERNAME="justin" PUBLICKEY="szW...93" ACTION="ANY"/>
</TRUSTEDPACKAGES>
```

Delegation can restrict trust

```
<TRUSTEDPACKAGES>
<USER PATTERN="*" USERNAME="CERT" PUBLICKEY="ke+...3a" ACTION="DENY"/>
<USER PATTERN="stork*" USERNAME="stork" PUBLICKEY="lSe...2W" ACTION="ALLOW"/>
<USER PATTERN="apache*" USERNAME="apache" PUBLICKEY="SwA...CA" ACTION="ALLOW"/>
<FILE PATTERN="lynx-1.2.3-21.rpm" HASH="179...a9" ACTION="ALLOW"/>
</TRUSTEDPACKAGES>
```

Trusted Packages



Client builds a view of potentially installable packages based on the packages that it trusts according to the metadata from administrators it trusts. It then performs package management actions according to the actions specified and groups to which it belongs.

Gacks

- Resource allocation mechanism
- Prototyping on PlanetLab/geniwrapper
  - Privileged slice
- Each resource has chain of allocators and one consumer slice
- An allocator can replace itself or its successor
- Receipts provide audit trail

Thanks

- Justin Cappos
- Justin Samuel, Jude Nelson, Jeremy Plichta, Duy Nyugen, Jason Hardies, Matt Borgard, and Jeffry Johnston
- PlanetLab Consortium

Challenges

- Resource discovery/selection
  - Intrinsic characteristics (e.g. available CPU)
  - Extrinsic characteristics (e.g. forms clique)
  - Flexible resource types and characteristics (rspec)
    - e.g. port, currency, spectrum, software
- Security vs. ease of use
- Heterogeneous components