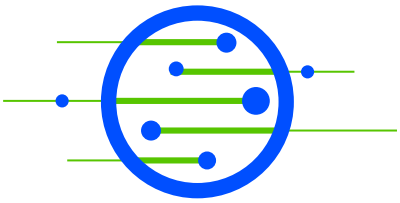

PlanetLab Phase 0: Technical Specification

The PlanetLab Architecture Team

Editor: Timothy Roscoe
Intel Research – Berkeley

PDN-02-002
August 2002

Status: Ongoing Draft.

PlanetLab Phase 0: Technical Specification

The PlanetLab Phase 0 Architecture Team

Version Control Information

CVS version information:

```
$Id: phase0.tex,v 1.2 2002/08/09 16:50:45 troscoe Exp $
```

This paper was run off August 9, 2002.

Contents

Chapter 1

Introduction

This evolving document tries to capture the design decisions made for Phase 0 of PlanetLab, together with the motivations for such decisions.

We intend this document to be a basis for discussion about the short-term design of PlanetLab. It will try to capture the reasons for the decisions we make as well as the nature of those decisions, and so hopefully move the discussion forward. Any feedback, criticism or flames the Architecture Team receive should in theory find a place here, either as a decision or a rejected course of action with a short passage saying why the road was not taken.

It has been initially modelled on Athena technical plan.

The initial version of this document is an attempt to synthesize ideas from a number of sources: the presentations from the March PlanetLab underground meeting, subsequent discussions on the PlanetLab mailing list, and discussions within the architecture team.

1.1 Notation

This document uses a few typographical conventions. Wherever possible, we've tried to separate out specifications of PlanetLab Phase 0 from discussions of why the spec is that way, but using annotations like this:

> **Motivation:** *Articulates the motivations for a decision: why we decided to do this, what alternatives occurred to us at the time, and why we rejected them in favour of the one in this document.*

Phase 0 of PlanetLab is a short-term project, to get something out there with a community of users who can collectively discuss its failings and areas from improvement. Many design issues for PlanetLab have been noted, but slated for Phase 1 or later. While this document describes what is *in* Phase 0, we use annotations like this for topics which are for possible later inclusion:

> **Out of scope for Phase 0:** *The topic is out of the scope of Phase 0 of PlanetLab. We'll try to justify leaving it out of the Phase 0 design, and possibly outline where it might fit in later and how we propose to address it.*

Some features of the design are not present in the initial Phase 0 design, but are expected to be added reasonably quickly. Hence:

> **Evolutionary:** *Not in the original Phase 0 release, but will be addressed well before the Phase 1 design is implemented. "Evolutionary" features will be treated in this document with varying degrees of thoroughness.*

There are many gaps in this document's coverage of the design of Phase 0. While the coverage will hopefully improve as the document evolves, there are cases right now where it is useful to flag a design issue as identified but still undecided. Hence:

> **Unresolved issue:** *This question must be decided for Phase 0, but has either yet to be considered fully by the Architecture Team, or is still the subject of controversy.*

Finally, there are a number of issues that this document might touch on but which we choose not to address here, because they are not primarily technical in nature. We use annotations like this to mention that we're aware of them:

> **Out of scope of this document:** *Topic is out of the scope of this document, because it's more of an administrative, financial, or political issue than a technical one. The topic is important for Phase 0, and we'll try and say where it is addressed elsewhere.*

1.2 Terminology

To facilitate discussion, this is a partial list of how terms are used in PlanetLab. Like the rest of this document, these terms and their definitions are up for grabs, but this section will hopefully serve as an (evolving) reference for the future.

Node: A server machine capable of running components of PlanetLab services.

Site: A physical (geographical) location where PlanetLab nodes are located, for example Princeton University, or Intel Research at Berkeley.

Cluster: The set of PlanetLab nodes located at a given site. Note that this is not necessarily very large: 2 and 3-node clusters are being deployed now, and single-machine clusters are conceivable, though not necessarily a good idea. Some clusters may be large, but in this document the term should not connote a large computing facility. We are not building the Grid.

User: An authorised human being wishing to deploy a service over PlanetLab. This is likely to be a member of a participating institution; a user will have the capability to request deployment of a service.

Client: While a user's code runs within a slice of the PlanetLab infrastructure, we use the term "client" to refer to a person or program which accesses a service which is implemented on PlanetLab. A client might be another PlanetLab service, but could also be a non-PlanetLab machine, program, or person.

Service: An application running over a set of PlanetLab nodes. Services include both parts of the PlanetLab infrastructure (for example, monitoring subsystems) and applications running on behalf of users.

Application: A PlanetLab service that is not part of PlanetLab's infrastructure.

Capsule: A component of a PlanetLab service that runs on a single node. The word "capsule" is not entirely satisfactory (for example, it has a different meaning in the Active Networks community), but it has a strong precedent in ISO RM-ODP and does capture the notion of a boundary around particular user's code running (eventually) in a VM slice of a PlanetLab node.

Slice: A distributed set of resources allocated to a service in PlanetLab. PlanetLab is about presenting users horizontal slices of the distributed platform as a whole, not simply dividing a node into virtual machines (which is a relatively well-understood problem).

Root server : In the short term (at least for Phase 0), PlanetLab will be centrally "managed". This term actually encompasses a variety of functions, including serving boot requests and requests for software updates, providing the account management interface to users, running a monitoring service, and running the backend database to support all these functions. Except where the design description needs to distinguish the machines running these, we refer to them in this document collectively as the "root server" for PlanetLab.

In later phases we can expect many of these functions to be decentralized and federated.

Chapter 2

Goals

The goals of Planet Lab Phase 0 are to start attracting researchers by providing a useful (if basic) platform for current wide-area networking and systems research, and to enable us to learn what is needed for later versions.

> **Out of scope of this document:** *A discussion of the wider goals for PlanetLab. These are described in the PlanetLab whitepaper and the position paper for Hot Networks, available from the PlanetLab web site.*

At a more detailed level, PlanetLab Phase 0 in fact has two conflicting goals.

The first is to provide an environment that early adopters among the research community will feel comfortable programming to (and porting their current work to with minimal effort). This entails a reasonably solid distributed infrastructure with a familiar execution environment and a robust and easy to use account management facility.

The second goal is to facilitate migration to a much more restricted API than Unix or Windows, and which will be designed on the basis of looking at the successful services implemented over this version of PlanetLab. This requires thorough instrumentation of the platform, as much for the benefit of PlanetLab's designers as for its users.

At the center of the contradiction is the definition of the API and Service Model that users can come to expect. The activity of defining and refining the service model will continue throughout the project.

One aspect of this evolution is the movement of functionality between the base system API, services provided by users, and services deemed to be part of the infrastructure. The core PlanetLab team will be soliciting user contributions in the form of services that can later become part of the infrastructure.

Chapter 3

Requirements

This section lays out the requirements for PlanetLab's functionality in Phase 0. It does not address requirements on components of PlanetLab which are implied by this functionality, for example the minimum specification of a PlanetLab node. Rather, this section is about describing the service PlanetLab provides to its users, and the constraints on the design imposed by what the implementers can achieve in the short time frame available for Phase 0.

3.1 Timescale

An overriding requirement for PlanetLab is that a working system is available for users on August 20th, 2002. Thus, the features and functionality in this document must be implementable by then (in some form). While we expect to continue to enhance Phase 0 PlanetLab after August 2002, most of the effort will be focussed on the design and eventual rollout for Phase 1. For this reason, many desirable PlanetLab features will be left for Phase 1.

3.2 Users

We need to support a small community of trusted users only for Phase 0. This will simply be those institutions represented at the March meeting, plus a small number of institutions which have been added since.

> **Motivation:** *This is a way of bootstrapping the requirements process: we start with a friendly, sophisticated user base who can provide the detailed feedback and suggestions for improvements which are essential to widening the user base in Phase 1*

This probably translates to a much larger number of "users", since we hope to attract a number of research projects from each institution. Thus our account management facility will need to handle on the order of a thousand users.

3.3 Sites and Nodes

The number of initial sites will be small - about 40. These will include all the institutions represented at the March meeting, and a few others that have been invited since. This means that we'll need to cope with less than 100 sites in the initial architecture.

Intel will initially be deploying about 100 nodes; the initial sites will have 2 or 3 nodes each.

> **Out of scope of this document:** *What the sites are, the deployment schedule, who gets what machines, and how the machines will be shipped, installed, etc. Initially this will be handled by Intel, but longer term we'll move to a more federated model.*

3.4 API and Execution Environment

The API (and ABI) provided to PlanetLab Phase 0 users must be something they are all familiar with. This implies a well-known network programming paradigm, such as Berkeley sockets, and realistically a Unix- or Posix-like environment in Phase 0.

PlanetLab users must not be able to gain root access to a PlanetLab node. By implication, they must not be able to install system-wide software of their own choice (such as what `rpm` does on a Linux system), change the operating system in any way, or load modules into a running kernel.

It is highly desirable to provide some kind of interface to raw sockets, to enable users to deploy network level gateways over PlanetLab (for example, RON).

Clearly this requirement is in tension with the need to deny all users root access to the machines.

There is no requirement for seamless migration of applications from Phase 0 to Phase 1.

> **Motivation:** *Part of our aim with Phase 0 is to find out what a reasonable restricted API looks like for Phase 1. Consequently, we expect to design the Phase 1 API to be useful and easy to move to for our Phase 0 users, but supporting absolutely everything they all do will be too constraining for us and defeat the object of narrowing the execution environment so it can be more precisely defined.*

There must be some kind of inter-capsule protection in the system. Services running on behalf of different users must not be allowed to unduly impact each other, either by unauthorised writing to (or reading from) each other's storage or by unilaterally exhausting one or more resources on the node.

> **Out of scope for Phase 0:** *Consideration of malicious services installed through the authorised deployment process. Our initial users will be well-known (particularly if we adopt Emulab's "principle investigator" model), and so social pressure can be applied. This will not be the case moving forward.*

Some form of controlled access to raw sockets is useful for Phase 0. This might not include the capability to implement a general IP router.

Users will receive a best-effort share of each node in Phase 0. However, we expect there will be a small number of initial users who will require very strong resource guarantees from the virtual machine for limited periods of time (for example, to perform timing-related experiments). We should try to accomodate such requests.

> **Out of scope for Phase 0:** *Novel operating systems issues for the nodes. With the exception of some kernel modifications for controlled access to raw sockets, we leave the operating systems issues for PlanetLab Phase 1. If good progress is made in implementing extra OS functionality, we might deploy it incrementally in Phase 0, but there are two pressures against doing this: (1) we need to collect both qualitative and quantitative feedback from users before making hard decisions regarding kernel facilities, and (2) we don't want to jeopardise the platform's usability, availability, or ease of use at this early stage by deploying operating system images that are not mature. As a consequence, strong resource isolation between capsules running on a*

node is deferred to Phase 1. Instead we rely on monitoring, management systems, and social pressure to prevent resource crosstalk from becoming a problem for services.

Unresolved issue: *The provision of semi-standard execution environments like a Java Runtime or Python interpreter by the infrastructure, usable by all services and supported as part of the PlanetLab Phase 0 infrastructure. Users will definitely request these, but can provide them themselves, at some cost in storage space. There is a danger that by providing them we give the illusion of a much richer (and therefore harder to narrow and specify) API than we would like to move to, and so by not providing these environments we encourage users to consider what they really need. On the other hand, we run the risk of discouraging users from adopting PlanetLab if the platform is too spartan.*

3.5 Hardware requirements

The nodes must all be capable of running a standard execution environment, such as a minimal Linux installation or some other operating system. By "standard", we mean that the facilities that a capsule can rely on being available on a node may be known in advance. This implies a standard processor architecture, but not necessarily the same processor implementation (or vendor) throughout.

Motivation: *At this stage, we cannot afford the resources to address heterogeneity within PlanetLab (such an effort would probably sink the project, as has happened in before). Standardizing machines to the extent proposed here makes the management problem tractable both for the PlanetLab infrastructure and application authors, and does not prevent the use of different models of machine (or different disk drives, networking interfaces, etc.).*

We must be able to remotely reboot any PlanetLab node, power-cycling it if necessary. This might involve human intervention in Phase 0.

Evolutionary: *The ability to completely power cycle any working PlanetLab node.*

3.6 Network connectivity

All PlanetLab clusters must have connectivity among themselves and to the Internet. Intra-cluster connectivity must be good (at least switched 100BaseT for all but the smallest clusters).

Connectivity between a cluster and the rest of the Internet is a much less of an issue; indeed long term we actually want a wide diversity of link characteristics from PlanetLab. For Phase 0, however, the principal requirement is that the nodes are reachable *most of the time* from management machines on the other side of the Internet.

In addition, connectivity must be good enough to make remote installation, monitoring, and management of the nodes feasible. This imposes a baseline constraint on both the availability and the available bandwidth of the nodes' Internet connection.

> **Out of scope for Phase 0:** *Wireless connections to PlanetLab nodes, or any low bandwidth links (less than 1Mb/s). Our aim initially is to have a manageable infrastructure, rather than to cope with everything the fringes of the Internet can throw at us. We are also interested primarily in planetary services and the software infrastructure providing them, rather than the design of clients that bind to them over a diverse set of access technologies.*

PlanetLab clusters should not be firewalled.

> **Motivation:** *PlanetLab nodes, even in Phase 0, may be running arbitrary services which listen on arbitrary ports. Anecdotal evidence, for instance from members of the Oceanstore group, suggests that even a well-understood home firewall owned by a skilled computer scientist can cause confusion when bringing up a new service behind it.*

Local site sysadmins should have access to a limited adminstrative interface to nodes at their site (possibly mediated through the root server). In particular, they should have the ability to limit outgoing (and incoming, if possible) bandwidth to the PlanetLab cluster at their site.

We expect PlanetLab to be used for a variety of networking and systems research activities, using highly experimental code. It seems inevitable that at some point an administrator at some site unrelated to PlanetLab will see packets or network behaviour which looks unusual (this has happened with many previous research projects). It is essential that such people can trace network phenomena back to

PlanetLab where appropriate, find out about the nature of PlanetLab and the packets in question, and if necessary can contact the relevant people in the PlanetLab team or the responsible user if there is a problem.

A related, even more serious scenario is when remote (non-PlanetLab) site administrator sends nastygram to University president complaining about spurious packets from a PlanetLab machine located institution. PlanetLab must provide mechanisms to be used as part of policy to both minimize the chance of this happening, and to take action to resolve such a situation.

> **Out of scope of this document:** *The political processes needed to prevent or resolve this kind of situation. This document instead focusses on technical mechanisms which can be used as part of the wider process of resolving the situation.*

3.7 User Accounts

Users should be given accounts on PlanetLab, which authorizes them to use the systems resources, request slices, etc. In Phase 0, account management may be highly simplified due to the limited user community (who generally all know each other).

> **Evolutionary:** *Users should be able to register for new accounts, manage their accounts (including their credentials), and create new slices online, for example through a web-based interface.*

As a baseline interface to the system, users should have shell access to a corresponding account on each node.

> **Motivation:** *PlanetLab is about computational services. To install, control, and diagnose problems with experimental services is a complex business. One approach is for PlanetLab to provide (build) a comprehensive environment for deploying, monitoring, and debugging such services, but this is a huge task. By providing for Phase 0 (and most likely Phase 1) a shell account, we give users a "baseline" interface which should allow them to perform most of the tasks they want to achieve on a node,* and also *the facilities for them to implement more sophisticated tools themselves. This latter point is very significant: this is how the PlanetLab community can learn what models work and what don't, and epitomizes the collaborative nature of the project.*

3.8 Management and configuration software

We must be able to reboot and completely reinstall all software on a PlanetLab node remotely and securely. Furthermore, we must be able to ensure that all software on the nodes is up-to-date, in the presence of intermittent and transient node failures or outages that may last days (for example, if a PlanetLab node in a University is powered down for 48 hours while being moved). Note that this involves each node both authenticating the server providing the software, and verifying the authenticity of the software package itself.

We must provide secure operational monitoring of the facility, to monitor for hardware failures, viruses, etc. The monitoring will be as close as possible to what we would provide for monitoring any collection of well-managed linux machines in a DMZ, plus whatever extra facilities may be required to PlanetLab.

To allow researchers to "restart after reboot" for ongoing services, we will report the status of each machine in the network on a centralized web page; researchers can periodically scan this page for changes, to determine when systems have been rebooted, and therefore when services on those machines need restarts.

3.9 Resource Usage

There must be some way for users to request a slice of PlanetLab, for some period of time, even though this might initially be a very simple mechanism (an email to the right person, for instance).

Similarly, there must be a way for resources to be reclaimed from users who are not using them. Experience would suggest that users are frequently disinclined to explicitly release resources that they no longer need.

3.10 Instrumentation

We are interested in how network service applications on PlanetLab use the physical machine resources and the facilities of the operating system. For this reason, we (the implementors of Phase 0) will need to collect detailed information about how applications use the nodes.

3.11 Storage

Users will require a storage area on each node for data and programs.

We will provide a set of standard scripts for managing the cluster of machines, for example, to push copies of files everywhere, to execute a script everywhere, etc. We will not provide a shared file server; rather, shared files will need to be explicitly downloaded before use. We expect overlay setup and maintenance to be the responsibility of the application.

> **Unresolved issue:** *How much storage does a node need? In some ways, this seems a harder problem than how much CPU power a node needs; the notion of graceful degradation under resource shortage is much harder with the spatial resource of disk space than the temporal resource of CPU, unless we treat local disk as a pure cache, which we're not yet in a position to do.*

Chapter 4

Service Model

This section describes the environment that users of PlanetLab will experience. While there is naturally considerable overlap between this *service model* and the implementation details in sections 5 and 6, it is still useful to make the distinction. For example, "use cases" belong here.

While the long-term plan for PlanetLab is to move to a thin virtual machine monitor specialized for network services, the initial "virtual machine" that we provide users is that provided by a Unix environment.

4.1 Accounts and authentication

Access to the machines will be via ssh with RSA authentication. Users will register their public keys via the account management interface on the PlanetLab web site, and these will be distributed to the nodes.

> **Motivation:** *RSA authentication provides a simple but familiar authentication interface, and with an authentication agent allows for considerable automation at the client end - experience with Emulab for instance suggests it works pretty well.*

Users will receive a standard Unix account on the machine, with no root access.

For the initial Phase 0 community, we will simply create 10 slices for each participating institution. Each slice will have a corresponding account. A Principal Investigator at each site will be responsible for the credentials for each account. Accounts will have simple quotas on file space and CPU usage.

Motivation: *In Phase 0 the user community is a set of known individuals at sites with nodes. This model is very simple (and so fits our time frame), but is still basically usable in the next few weeks.*

Evolutionary: *The precise relation between Unix UIDs on PlanetLab, human users, PlanetLab "projects" (analogous to Emulab "experiments"), will become more sophisticated. The hardwired accounts will go. Emulab has much valuable experience here, but PlanetLab also has unique challenges in this area - for example, we may have scaling problems for Unix UIDs which might entail their reuse. New users and institutions can be added when the account model has evolved to this stage.*

4.2 Scheduling of slices

Initial Phase 0 PlanetLab users will be allocated a fixed number of slices which are running "all the time".

Evolutionary: *A properly defined relation between slices, accounts, people, etc. Slices will have have leases. If a lease expires, the slice and/or the service running inside it is terminated.*

Out of scope of this document: *The policy for renewing leases.*

Evolutionary: *We expect that some applications will become sufficiently compelling for PlanetLab as a whole that they graduate to becoming an infrastructure slice with its own resources and a more permanent lease, and supported through platform mechanisms.*

Out of scope for Phase 0: *A full scheduler which can handle both the constraint-based scheduling of experiments to run at defined times, and the long-term deployment of infrastructure or application services. In other words, infrastructure support for the "dual-use" nature of PlanetLab.*

4.3 Per-node scheduling

Initially we offer mainly a best-effort CPU service (e.g. the basic Unix scheduler), perhaps augmented with the standard process quota facilities provided by the ker-

nel.

> **Out of scope for Phase 0:** *A full solution for scheduling services and experiments over slices. This includes the* policy *for resource allocation (apart from the simple ad hoc policy here). Also the software that embodies this management policy. PlanetLab will seek input from the user community to establish policy for overcommitment.*

It is likely that a small number of Phase 0 uses will request a dedicated machine (or fraction thereof) for a limited period of time to run experiments which are highly time-sensitive. It is possible that some of these applications really are dependent on using a dedicated machine to minimise response latency, for instance.

Rather than offering a "dedicated machine" service, we offer a service in Phase 0 with very strong guarantees for limited times. This service happens to be implemented by a dedicated machine in Phase 0. In later phases, it will be implemented by the slice isolation and scheduling mechanism.

> **Motivation:** *We believe there will always be a specialised need for resource guarantees. Furthermore, long term minimum guarantees of some kind (shares or reservations) will be a frequent requirement. Until we have a good slicing mechanism, this provides a very primitive way of guaranteeing resources for some applications which absolutely require them in the short term, for example for experimentation.*

4.4 Privilege level and execution environment

PlanetLab users cannot boot their own kernel, nor are they ever likely to be able to. Users don't get root access on their nodes.

> **Evolutionary:** *We will provide controlled access to raw sockets.*

4.5 Storage

Each PlanetLab user gets a Unix directory for storage of programs, data, and results. Every file the user needs to write to should be in this directory. There will be a quota on this file space.

Motivation: *Quotas act as primitive limits of resource usage so as to protect other users from log files filling up, but also give us a coarse-grained monitoring of file space usage by different slices.*

All local storage should be considered temporary, and may be lost at any time (in the same way that the node can fail at any time).

Unresolved issue: *How big the initial quota per node per slice is.*

Evolutionary: *Each service has a Janus like sandbox / virtual root*

We do not provide a global file system. Users may provide their own, implemented either over PlanetLab or externally to the PlanetLab nodes.

4.6 Supported environments and service

Our initial position on execution environments is this: we provide a well-defined, relatively raw Unix environment. Common utilities like text editors will be present; services must provide their own runtime environments beyond this (for exmaple, JVMs).

Motivation: *A straw poll of a few research groups revealed that all wanted the platform to provide a Java Virtual Machine, but all required a different version or vendor. In Phase 0, it's only feasible for use to support the platform if the users provide all the application-specific code. Furthermore, since our aim is to investigate the minimal protection interface over which most services can be implemented, it is not in our interest to maximise this interface early on by offering a variety of high-level execution environments.*

Unresolved issue: *Package managment for execution environments and ways of sharing code moving forward. The approach of md5 summing files and sharing them with copy-on-write has been suggested to reduce redundancy.*

4.7 Service control interface

The PlanetLab allocation service receives a *service description*, returns a *resource description*. A resource description includes a list of all the nodes allocated to the slice.

Unresolved issue: *Definitions of these.*

Group execution facility to run the same command line on each node allocated to a service. We provide one of these, but do not preclude users' reimplementations.

Unresolved issue: *Description of this service.*

Users can provide a subdirectory in their "home" directory on a PlanetLab node with a well-known name: `PlanetLab-control`. Scripts with particular names in this directory will be called by the node control software in response to various node wide conditions. Implementation of this directory by users is, of course, optional; they always have the opportunity to use ssh to manually start and stop services. However, this may be a useful facility.

Motivation: *The analogy here is with the `rc.d` directories in Unix, which provide great flexibility with a simple interface.*

configure is invoked when the software is first installed on a node. This provides basic file initialisation beyond untar-ing the archive.

start may be invoked when the machine reboots. This gives the service some ability to restart itself due to a transient failure of the infrastructure.

status may be run by the infrastructure and return a string which, initially, is simply the name of the service. If the service has recognizably failed, this script should fail.

Motivation: *Ultimately, this is only useful for demos and casual inspection of the service. However, it is still compelling, and relatively easy to do.*

stop may be invoked when the node is about to forcibly terminate a service, shortly before all processes belonging to the capsule are killed.

Note that the node reserves the right to kill a capsule at any time, but if possible the node will call stop some time in advance.

Unresolved issue: *Which other events are provided to the service interface by the infrastructure.*

Regardless, notification of node reboots and/or crashes is posted to a web page. This punts restart problems to users.

Out of scope for Phase 0: *A planetary scale event notification system.*

If any one of these scripts is not present when it would normally be invoked, the infrastructure continues. The output and return status of all of these scripts (except

possibly **status**) is discarded by the node. If users want the output of any of these scripts, it is their responsibility to save the output for later collection by them. As far as users are concerned, we discard the output of these scripts.

> **Motivation:** *It's clear on reflection that nothing useful at this stage can be done by the node in response to different outputs from these scripts, without imposing an unwarranted model of, for example, service restart on users. It's best to leave this undefined for the time being until it's clear what models make sense.*

4.8 Bulletin board and information discovery

PlanetLab will provide a way for a capsule to find out node-specific information such as the resource description for the service by writing the information into a file in the `PlanetLab-control` directory called `resources`. This is a slice description which includes the list of node addresses comprising the slice. At service creation time it is expected to be accurate.

> **Unresolved issue:** *The format of this file.*
>
> **Out of scope for Phase 0:** *A more general mechanism for dynamic changes to this information as the slice changes.*
>
> **Evolutionary:** *A general means of inter-node rendezvous for capsules in a specific service and across service (name service or tuple space). This is likely to be a service itself.*

4.9 File distribution

We will provide a means for users to distribute their software to all the nodes they intend to run a capsule on. Most likely, a user will submit a tarball through the PlanetLab web site, and this will be untar'ed into the service's home directory on each PlanetLab node.

We really want scripts to interactively copy files to and from a complete set of nodes (for logs, configuration data, etc.). Might want help building this.

Alternatively, users are always free to use scp to distribute their own files to PlanetLab nodes.

4.10 Network topology

Unresolved issue: *I'm not sure what this section means.*

4.11 Network address and port allocation

Each node will have a single IPv4 address.

Motivation: *We can't afford to allocate multiple IP addresses per node, given the deployment constraints for PlanetLab.*

We don't allow users to reserve ports, nor listen on privileged ports. For Phase 0, we will have central port number allocation authority (for PlanetLab) which is managed socially and assigns well-known ports to users.

Evolutionary: *The ability to listen on privileged ports may be provided by the same functionality that gives controlled access to raw sockets.*

Out of scope for Phase 0: *The general interface to the network seen by a slice, in terms of port numbers, addresses, VPNs allocated to slices, etc. There has been much useful discussion of these issues, and it's clear that much more technical discussion needs to occur before we are clear what it means to virtualise a network interface in PlanetLab.*

Chapter 5

Hardware

Nodes will be purchased centrally and dropshipped for Phase 0. Currently, all nodes are provided by Intel and supplied by Dell.

Two kinds of nodes are shipping: Dell PowerEdge 1650 1U Rackmount servers, each with 1Gb RAM and a 1.7GHz Intel Pentium III Xeon processor, and Dell "desktop" workstation machines, with an Intel Pentium IV processor and similar spec to the 1U machines. Both machines come with 1Gb/100Mb ethernet cards.

> **Out of scope of this document:** *Policy for appropriate use of nodes (exclusively PlanetLab, basically) and means for enforcing this policy (mainly social). Which sites receive which kinds of nodes is a matter for negotiation between PlanetLab and the site.*
>
> **Unresolved issue:** *Whether a front-end switch for each site, and the more general problem of network hardware configuration for the nodes.*
>
> **Evolutionary:** *Providing a networked power strip for each site.*

Each site will have a minimum of two machines.

> **Motivation:** *This allows us to (expensively) reserve an entire node for a user for a limited period of time if necessary.*
>
> **Out of scope for Phase 0:** *Provision of specialised peripherals for some of the nodes, such as GPS or CDMA2000 receivers, satellite uplinks, sensors, etc. Currently, all nodes are connected just to the network and are otherwise compute engines only.*

Chapter 6

Software Architecture

This section describes the details of the software architecture for PlanetLab Phase 0. The system as a whole will consist of software running on each node, together with a centralized group of services collectively referred to here as the PlanetLab root server - providing monitoring output, boot service, software distribution, etc.

While section 4 described PlanetLab Phase 0 from the point of view of users' experience with the system, this section addresses the software implementation necessary to provide this experience.

> **Out of scope of this document:** *The planned evolution of the PlanetLab node operating system towards a thin Virtual Machine Monitor (VMM) model.*

6.1 Node Operating System

The PlanetLab nodes will initially be configured with a basic standard Linux distribution.

> **Motivation:** *Our initial execution environment for PlanetLab must be familiar to all the potential users of the system, and readily available right now. We also require the ability to evolve both the interface (by narrowing it as requirements become clearer) and the operating system below it (as we add resource isolation, instrumentation, and eventually evolve to a thin virtual machine monitor). This basically means an open source Unix-like OS in widespread use. The feeling of*

the March meeting was rather more in favour of Linux than BSD, and while significant differences do exist, the facilities offered by the two are increasingly similar. We picked Linux.

Phase 0 will not initially allow users to acquire raw socket access to the network. However, during Phase 0 we will add functionality to the kernel to allow users to allocate (or perhaps lease) safe raw ports. Unlike raw sockets, these ports will be prevented from sourcing spoofed packets (e.g., they may not spoof the host IP source address or the dedicated port number), and will be bidirectional (e.g., will use packet filtering and software firewall to deliver incoming packets addressed to the port to the application). Kernel modifications are needed to accomplish this.

Evolutionary: *Adding support for safe raw sockets is the highest priority for the PlanetLab implementors. This functionality will be provided by the SILK kernel module.*

Evolutionary: *We will enforce outgoing network bandwidth limits using either iproute2 or the SILK module. This will be updated by the PlanetLab root server (including in response to local site administrative requests), and link utilisation will be monitored via ganglia. Bandwidth limits will be per-machine.*

Evolutionary: *We may move to some semi-virtualization technology over Linux in the short to medium term. User-Mode Linux and Linux Virtual Servers are both open-source candidates in this space.*

Out of scope for Phase 0: *Bandwidth limitations on a per-slice (per-capsule) basis; resource containers for CPU, memory and disk resources; and the corresponding scheduling and reservations mechanisms. It is possible that these mechanisms will be deployed incrementally during Phase 0, but they are likely to be considered as part of Phase 1.*

6.2 System Services

The node will provide very little in the way of always-running services. The basic services on the node will be `sshd` (TCP port 22), ganglia (TCP port 8649), `ntpd` as a client, and Rootstock's regular update service (which doesn't listen).

We provide no messaging services. Users will implement their own methods for capsules to send events off node.

6.3 Network connectivity and configuration

The IP addresses of all PlanetLab nodes will be registered with the PlanetLab root server, and will have names bound to them in the `planet-lab.org` domain which indicate which site the node is a part of. The name of a machine will be `PlanetLab`*n*`.`*site*`.planet-lab.org` , where *n* is the number of the node at the site (starting at 1), and *site* is a short name for the site, such as `princeton` or `irb`.

In addition, it is the responsibility of each PlanetLab site to ensure that IP addresses of its PlanetLab nodes map to recognizable PlanetLab domain names (either in one of the site's local domains or the `planet-lab.org` domain. We strongly suggest `PlanetLab`*n*`.`*localdomain* as the local name of a PlanetLab node, where *localdomain* is the local domain, such as `cs.princeton.edu` or `berkeley.intel-research.net`. These DNS names of the PlanetLab nodes are chosen so that reverse DNS lookup will clearly identify the "test lab/measurement" nature of packets reaching the outside world. It is in the interest of sites to do this, to reduce the enquiries they will recieve from admins who have no idea that the machine is linked to PlanetLab.

The PlanetLab root server will provide a dynamic DNS service for the `.planet-lab.org` domain, but note the IP addresses of PlanetLab nodes may still need to be statically assigned for some sites.

An email address will be clearly identified on the PlanetLab root server web page for where to complain about undesired packets (e.g., for Internet measurement studies).

In Phase 0, each node will have a single IP address, and well-known port numbers for services will be allocated by a central PlanetLab authority. Enforcement will be social.

Each node runs a web server which always returns a redirect to an appropriate page on `www.planet-lab.org`, either the home page or a per-node page. We'll log the request at the node. This page should contain enough information to mollify concerned network administrators. The web server should be as simple as possible.

> **Evolutionary:** *Allowing an "opt-out" policy where we block outgoing traffic from any PlanetLab node to an aggrieved person's subnet. We also need to know when people are doing things which generate unsolicited traffic. One option is to log all connections and the use raw sockets. Another is to collect netflow-like statistics. This is quite deep*

instrumentaion, and would be very useful for a number of reasons.

6.4 Configuration, Installation and Booting

Phase 0 of PlanetLab will use a modification of the Millenium Project's RootStock booting and configuration environment. The modifications consist of encrypting the install traffic, authenticating the boot server through a certificate, and digitally signing all packages to be installed on the nodes.

> **Motivation:** *There are a number of fine remote boot, installation, and management systems out there (Xenoboot, Emulab boot, etc.). Rootstock has the capability of constantly updating a running system and handling network connectivity failures well, and has a package-oriented approach to installation rather than the filing-system-at-once approach used by some other systems. This means that initial installs are slower (but are probably network b/w limited in both cases, and can be alleviated by CDROM caching), but the system is much simpler for incremental updates. Rootstock also provides an initial solution for distribution of application code, which may be a short term win.*

Server nodes will be purchased centrally and shipped directly to each remote site for installation. To install a new system, the site administrator accesses a web page to specify how the node receives its IP address (static or via DHCP), and this causes a floppy to be created and shipped to the remote site. The new node is booted off the floppy, causing it to retrieve the real OS image (from the PlanetLab root server) and to wipe its disk. If the system becomes corrupted and/or a new OS version needs to be installed, the node can be reconfigured from the floppy.

> **Evolutionary:** *The boot floppy will be replaced with a secure bootable CD which is permanently installed in the machine. As well as providing better security and adminstrative functionality, this means that a human being does not need to be present when the base system is installed (or reinstalled); currently someone must remove the floppy disk after booting.*

We will not allow nodes to join unless we purchased them (no donated nodes) and they must run our version of the Linux kernel. In particular, the only access we give to the local sysadmin is the password to the powerstrip (to allow them to disable the box in case it is hacked). The root password to each node is managed at the PlanetLab root server, and not provided to the local researcher or sysadmin.

Local sysadmins will, however, have a control interface to each node in order to limit outgoing network bandwidth. Local site administrator must be able to set the outgoing bandwidth limit for a node.

> **Unresolved issue:** *Changes to the standard RootStock configuration are approved by some body of people. Process issues to do with upgrading critical parts of a node (such as the kernel) need to be resolved: test areas or nodes, running multiple configurations side by side, etc.*

> **Evolutionary:** *Verification of server identity using SSL/server certificates, and signing of rootstock packages, or porting the rootstock update management scheme to a different boot monitor like Xenoboot.*

The PlanetLab root web server will have instructions on web site for removing the floppy disk after booting, so that when the machine reboots it actually runs the new installation. Xenoboot or a similar boot environment would remove this restriction.

6.5 System Health Monitoring and Instrumentation

PlanetLab will use the Millenium Ganglia system for monitoring and measurement of nodes in Phase 0. Ganglia works best if all the machines at a site are in the same multicast domain. The system documentation must mention this, on the installation web page, for instance.

> **Motivation:** *Ganglia's plugin architecture for monitoring allows us to extend measurement functionality over a 15-second granularity to the kinds of values we're interested in in the early stages of PlanetLab. Despite being designed for clusters, Ganglia's hierarchical architecture scales well to the "many tiny clusters" environment of PlanetLab, at least for the first couple of hundred sites, and is available right now. Beyond this, a new system may be needed.*

While Ganglia's information gathering and aggregation scales well for the moment, some of the data visualization tools it incorporates are not oriented to several hundred small clusters. More work will be required here.

> **Out of scope for Phase 0:** *A systematic approach to monitoring application-specific values, provided as a general service to users. Users are expected to provide their own monitoring infrastructure, and have complete flexibility in this. We hope to learn from their efforts what kind of*

measurement and monitoring models are appropriate for planetary-scale applications. We may still offer case-by-case usage of the initial Ganglia infrastructure, but it's not mandated.

Evolutionary: *Handling high-volume instrumentation data (such as the output from* `strace`*, netflow-like statistics, etc. will require a different approach. Circular buffer approaches and continuous monitoring allow us to monitor "after the fact" and also don't cause user performance expectation problems. The service for extracting snapshots of these circular buffers is a good candidate for running in a slice. Processing will have to be offline for the time being.*

6.6 Account Management

A central database at the PlanetLab root server will manage RSA ssh keys for all the nodes. Initially however, each site participating in PlanetLab will nominate a PI, who will be assigned the keys for 10 accounts, each of which will have a permanent slice. The account name, and the slice name, will be based on the short name for the site used in DNS registration above, leading to names like `princeton4` and `irb10`.

Evolutionary: *Emulabs's PI delegation model has to be the right way to go.* `ssh` *handles multiple keys per account, and we will log the key used to get into each account, allowing students to be given a key each. We'll need to give users advice on guarding their ssh keys.*

Evolutionary: *Machines will have numerous IDS mechanisms installed as soon as possible, via RootStock.*

6.7 User Software Deployment

As a baseline, each user will have `ssh` and `scp` access to each node on which their slice runs. The PlanetLab root web server will contain some sample scripts and ssh tutorials for help deployment software.

Evolutionary: *A better distribution mechanism such as* `rsync` *will be provided.*

Out of scope for Phase 0: *Full transactional installation capabilities across a slice.*

Chapter 7

Development Environment

PlanetLab will be, and must remain, and Open Source project. The licensing terms will be BSD, or the Intel Open Source licence (which is identical to BSD with the addition of an export regulation disclaimer).

> **Motivation:** *BSD provides the most flexibility in the use of PlanetLab code, and is the license already used or being considered for a variety of other related projects (Emulab, Oceanstore, etc.)*

All source code will be held in a CVS repository on SourceForge, which will also be used to host mailing lists from the project, and all the other nice facilities provided by SourceForge.

Chapter 8

Acknowledgements

Unresolved issue: *Comprehensive list of folks who directly contributed.*