PLANETLAB

Distributed System Management: PlanetLab Incidents and Management Tools

Robert Adams
Intel Corporation

PDN–03–015
November 2003

Status: Ongoing Draft.

Distributed System Management: PlanetLab Incidents and Management Tools†

Robert Adams
Intel Corporation

4 November 2003

Abstract

PlanetLab is an open, global, distributed test bed for developing, deploying and accessing planetary-scale network services. Its goal is to be the infrastructure for a new generation of applications and services in the Internet. Supporting this new generation of distributed applications and services in the open Internet presents new challenges in support, maintenance and administration.

This paper describes some of the management incidents that occurred in the first year of PlanetLab operation and describes some of the technologies that have been developed to cope with these problems with an eye to exploring the requirements for scalable management of an open distributed computer system.

1 PlanetLab Operation

PlanetLab is an open, global, distributed test bed for developing, deploying and accessing planetary-scale network services [PLANETLAB]. As of October 2003, there are more than 200 machines at 90 sites worldwide[1] available to support both short-term experiments and long-running network services. Over the next few years, PlanetLab will grow to over 1000 nodes and host some of the newest and innovative services available.

A small support team monitors the operation of all the PlanetLab nodes and fields support emails from users. The support team fulfills several tasks: keeping PlanetLab running; supporting the installation and operation of new PlanetLab nodes; fielding support questions and problems; developing basic PlanetLab user features; and developing tools for management and tracking of PlanetLab operation.

To better understand how PlanetLab is run, what follows is a brief description of how PlanetLab operates.

PlanetLab is a collection of computers distributed around the Internet. The individual computers are called *nodes*. PlanetLab nodes run a standard version of Linux with some enhancements for supporting multiple applications and users.

Applications share a PlanetLab node by residing in separate virtual servers. To each application, the running environment looks like a private Linux computer – a user has root access to the system files and can install Linux packages (RPMs, etc) as required. The system kernel has been modified to use vservers [VSERVER] for each user of the node. Vservers are akin to the "BSD jail" code and gives each user of a node its own separate copy of the system files and certain special privileges that make it look like the user has complete control of the computer.

PlanetLab is not about running an application on one node. It's about running distributed, decentralized applications and services across many nodes. In PlanetLab lingo, a *slice* is an application that cuts across several nodes and a *sliver* is the part of a slice that runs on one node. To support this ability for a slice (application or service) to have access to multiple nodes, PlanetLab implements a mechanism to create and destroy *slices* across the testbed and to distribute SSH keys that give users access to slivers on each node.

Institutions and corporations join PlanetLab by donating computers and becoming a *site*. Each site has a *principal investigator* responsible for that site's use of PlanetLab. Individual *researchers* sign up for PlanetLab accounts by supplying their name, site and email address and additionally agreeing to the PlanetLab Acceptable Use Policy [AUP].

At each of the sites, there is a *local site administrator* responsible as the technical contact for the nodes at the site. PlanetLab nodes are connected to an institutions network and receive their IP address and DNS entries from the hosting site.

The principal investigator controls access to slices. At a university, for instance, it is common for the principal investigator to be a faculty member associated with several research programs. The principal investigator assigns slices and their associated PlanetLab resources to individual users by associating the PlanetLab accounts with the slices. The authorized users then use PlanetLab for their experiments, research and development.

† PlanetLab PDN: PDN-03-015. http://www.planet-lab.org/pdn/pdn-03.015.pdf .

[1] Status and size of PlanetLab is available at http://www.planet-lab.org/.

PlanetLab's initial implementation has the account registration and slice assignment occurring at a central location called *PlanetLab Support*. PlanetLab Support is a web server and database that holds the information on the accounts, slices and nodes. Additionally, there are a small number of people who maintain PlanetLab, keep it running and respond to any problems that might arise.

Some observations from this organization:

- The applications, research and services run on PlanetLab are not controlled or certified by any central authority;
- Because most of the initial work will be research, network traffic from PlanetLab nodes will be pushing the boundaries of Internet operation in both pattern and volume;
- The individual nodes are at sites and thus appear to be under the administration of that site (who know even less about what is run on the nodes);
- A very small team at PlanetLab Central has the job of keeping PlanetLab running.

Running hundreds of nodes scattered across multiple locations means that there will always be some nodes that are unavailable. Over time, tools have been developed and adapted to monitor the operation of the nodes (these will be discussed later in the paper), but outside of simple machine failures, most of the support is around "incidents" that are initiated by emails in the support email mailing list.

In 2003, the number of PlanetLab nodes has grown from 100 to over 200. As seen in Figure 1, the total available nodes (nodes that are accessible to researchers) has risen steadily while the number of unavailable nodes has stayed at a more or less steady number of less than 20. Some of the notable spikes in unavailable nodes are described later in this paper.

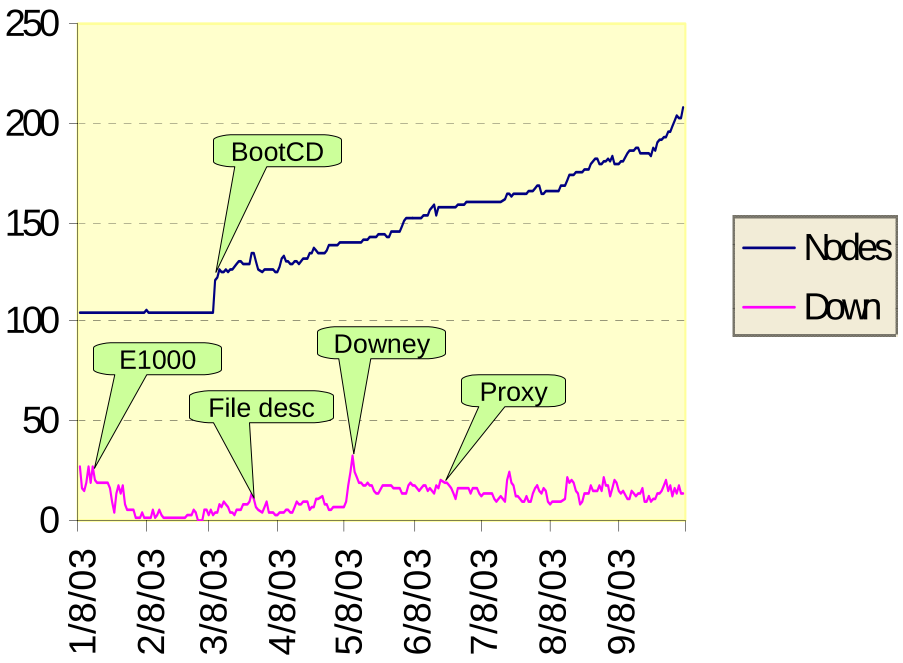

Figure 1: PlanetLab nodes and node availability for 2003[2]

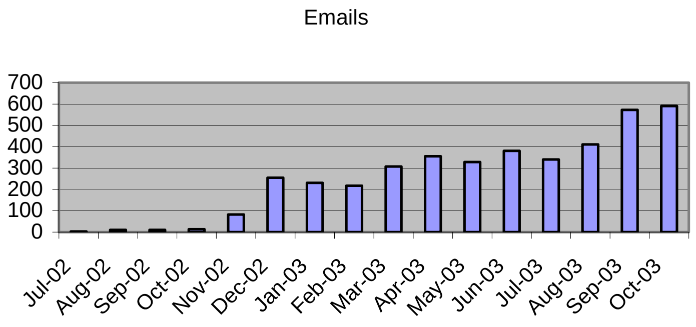

Figure 2: Number of support emails[3]

[2] PlanetLab node availability information published at http://www.planet-lab.org/logs/scout-monitor/
[3] Support emails available at http://sourceforge.net/projects/planetlab/

2 Incidents

We use the term incident to describe an event that requires support time to resolve an operational problem -- getting a node operational or repair a software problem.

The following sections describe some of the incidents that have occurred. The incidents are described and then some lessons learned from that particular incident are given. The names given to the incidents correspond roughly to the subject line of the PlanetLab support mailing list. A more complete list of incidents is given in Appendix A.

From January 2003 to October 2003 incidents are classified into four broad categories:

1. broken hardware and driver problems,
2. broken infrastructure software problems,
3. networking problems: bandwidth problems and traffic type problems
4. problems with the applications or services

Appendix A lists the major events by category, date and short description. The following section of this paper describes each of these incident classes

Broken Hardware

PlanetLab, of course, has the usual problems with hardware, peripherals and the associated driver software. Additionally, the systems are remote and not easy to diagnose or maintain.

"E1000": On December 19, 2002. On this date, someone noticed that one node at each site was unavailable. The support team set about rebooting nodes, which eventually returned all of nodes to service, but nodes kept crashing thereafter. For several weeks, PlanetLab Support was spending time identifying crashed systems and local technical support people had to reboot systems several times a week. At one point, nearly 30% of the PlanetLab nodes were not operational. Local technical contacts reported that the console showed kernel panics mostly from the "SILK" module. PlanetLab kernels have special patches and modules that support the virtualization environment and these panics suggested the problem was in one of these.

Examination of the slice network activity logs pointed to one slice that usually was logged in when the node crashed with a panic. The experimenter was asked to hold off their experiments and this significantly reduced the number of system panics. The developer of the SILK module examined the code and trickles of information from the consoles of panics systems lead people to suspect the queuing mechanism of the E1000 Ethernet driver. It was finally noted that the PlanetLab kernel included an old revision of the driver for this card. A new kernel was built and deployed and these kernel panics ceased.

Lessons: stress testing of the whole system would have alleviated this problem. The testing must include exercising the PlanetLab special modules and the type of load PlanetLab experiments generate. It was hypothesized that the fact that mapping experiments make many short term connections to many disparate systems – something "normal" program don't usually do – is what caused the connection queuing mechanism in the driver to fail.

Another observation was the amount of people time it took to manage, research, identify and fix this problem. The local technical support people had to physically access the nodes, read the console output and reboot the systems. This identified the need for better remote control of the systems.

"Memory Parity": Twice nodes have been made unavailable by memory parity errors. Memory parity errors are reported on the console so they are easy to detect if you have console access. This was true in both cases (Jan 27, 2003 "UCLA memory" and Mar 10, 2003 "Stanford memory") and in each of these cases, local technical contacts manually read and reported the console output.

Lessons: Sometimes the hardware fails. Capturing console output for panics is important for diagnosis. Some sites had the consoles wired in a loop (one system's console out is wired to the next system's serial in) but this has proved unwieldy because it was hard to get cables, hard to wire correctly and never captured the offending events. Eventually a special kernel module was added that captured console output and saved it across reboots.

Broken Software

PlanetLab runs a modified version of Linux that supports the isolation of environments between the slices on a node, provide operational logging information, and implement the login and access mechanisms that support PlanetLab accounts. As with any software, there are bugs.

"TCP connection problem", February 27, 2003. The PlanetLab kernel contains patches that tracks and supports network usage by each of the virtual servers (slices). The "SILK" module interfaces to the networking stack and collects statistics and provides special access to "raw sockets" (a subset of true raw sockets that allows the mapping and protocol experiments that are common on PlanetLab). At the end of February, a new version of the SILK module was deployed in PlanetLab. In a day or two, researchers started reporting problems with sockets accessed from Java. The developer of the SILK module was able to quickly identify a problem and a new version of the module was distributed to PlanetLab

Lesson: thorough testing of new kernels and kernel features is an obvious need. And the testing must cover the types of application environments the researchers are using – in this case, stress testing was done but that didn't show the problems with the Java environment. Also, "rolling updates" are a good strategy for updating the PlanetLab infrastructure – update only a portion of PlanetLab, wait to see if there problem, update to some more nodes, evaluate success, etc.

"Cron hangage": December 30, 2002. Someone noticed that some of the maintenance cron jobs look hung – they seem to have been running for more than 5 hours. The day before, all of the PlanetLab nodes had been rebooted for a software update. This caused every node to connect to PlanetLab Support and to download the updates. But 130 nodes, all trying to contact one update server, caused long delays. The nodes eventually came up and the cron jobs exited.

Lesson: even the PlanetLab infrastructure must comprehend the problems of distributed systems. This led to an evaluation of the boot and update mechanism and of an eventual redesign. The current implementation still relies on a central location for update but future plans are for a totally decentralized system as the supporting services become available.

Network

PlanetLab nodes are connected to the everyday Internet from within their hosting institutions. The interplay of experiments on PlanetLab and their network environment causes incident of two types: bandwidth (too much bandwidth used) and traffic (the traffic pattern is outside normal profiles).

2.A.1 Excessive Bandwidth

"Bandwidth spikes": November 26, 2002. A university network administrator complained about bandwidth spikes from three hosted PlanetLab nodes. The high volume of network traffic was strictly outbound and to destinations that did not exist. This caused the border routers to do extra work for each packet and the overall effect was to make the university's Internet connection unusable. The nodes were disconnected.

Lesson: Experiments can exceed the usual bandwidth limits and can generate traffic that causes special problems. Per node bandwidth caps were identified as a tool to manage this type of problem.

"Seattle meltdown": December 6, 2002. A site administrator took their PlanetLab nodes offline because they saturated the local network. This was traced to an experimenter measuring maximum throughput of a networked application.

Lesson: Again, experimenters can exceed local patterns. In this case, in addition to needing bandwidth limiters, some education of the experimenters was needed so they understood the PlanetLab acceptable use policy and some of the limitations of the environment.

"Canterbury bandwidth": January 12, 2003. The University of Canterbury, New Zealand reported that their hosted PlanetLab nodes passed six gigabytes of traffic in two weeks. At this site, network bandwidth is charged for by the megabyte so large traffic usage can run up a very large bill. This problem has also occurred at some domestic universities where bandwidth is charged by the gigabyte. Some PlanetLab experiments are specifically working on the problem of distributed storage of large files and they tend to move large amounts of data for sustained periods of time.

Lesson: A system of capping, limiting and allocating network bandwidth is needed. The solution should apply to balancing the usage of limited bandwidth between slices on a node as well as controlling the total node bandwidth and bytes used.

"ucb5 traffic": June 2, 2003. The PlanetLab support group sent a message to an experimenter asking if the more than one terabyte of traffic they had generated across PlanetLab was an expected feature of their experiment.

Lesson: This is one of the first occurrences of the PlanetLab monitoring tools making possible problems visible and making proactive actions possible.

2.A.2 Inappropriate Traffic

"port 0": November 27, 2002. Some PlanetLab nodes were sending large numbers of ICMP packets to port 0 in order to do mapping and timing experiments. Several of these nodes were sending the packets to one machine. The additive effect eventually took down the DMZ router at that site. Using the packet count logs, it was possible to identify the experimenter generating the traffic and ask them to stop and then redesign their experiment.

Lesson: It is important to trace network traffic back to the person generating it. At this time, the SILK module counted packets transmitted by active slices and returned that information in the `/proc` directory. Since the experiment was still running, it was possible to identify which slice was generating the traffic.

"Planetlab Attach": December 18. 2003. An ISP reported that a PlanetLab node was "attacking" a system on their network by exceeding the inbound packet flow limits. The "attack" happened on December 18th and the email asking for correction was sent to administrative contact for the second level domain name owner (the university). On December 19th, the report of the attack was forwarded to the local technical owner of the machine who, on December

20th, forwarded the report to the PlanetLab support list. The traffic was traced to an experiment and the experiment operation was modified.

Lesson: Two lessons here: 1) mapping experiments set off network monitoring alarms easily and 2) there is a long chain of people and a long period of time between the alarm going off and the experimenter creating the traffic.

The number of people involved is a problem because each has a limited understanding of the actual problem and they all have a job of dealing with "attacks". Many of the people in the middle of the chain take claims of network attacks very seriously but their only possible response is to disconnect the offending computer. This makes the PlanetLab node unavailable for all its users and makes it hard to diagnose the problem. In particular, the logs on the computer cannot be analyzed to see who created the reported traffic. The problem of eliminating the "people in the middle" is talked about in the later section "Disintermediation".

That the report of this "attack" took two days to progress from the attacked to the people who could analyze the problem means that records of traffic must be archived and kept for at least several days. Short experiments could generate the traffic for only a few hours and experiment could be long gone by the time someone looks at the actual source computer to see who is generating the traffic. This pointed to a need for audit logs of all network traffic sent from PlanetLab nodes and this information must be kept for a minimum of several days.

"Gnutella1": December 31. 2002. A university's network monitoring tool detected Gnutella traffic from a PlanetLab node. As policy, peer-to-peer file sharing is not allowed at this particular university. The traffic was traced to an experimenter who was using the Limewire application to collect peer-to-peer query and topology information and to not actually share any copyrighted material. The experimenter was asked to not perform the experiments at this particular institution.

Lesson: Some sites have specific policy requirements for application traffic. These have legal ramifications. Also, network-monitoring tools detect the use of an application (in this case, the use of a particular peer-to-peer file sharing application) and not the actual illegal activity.

"Port 80 scanning": April 15, 2003. A university's network administration logged their two PlanetLab nodes doing a scan of port 80 to many external sites. The two nodes were disconnected from the network. Reports of alarms from three different sites were received. The experiment was sending a low volume of TCP SYN packets to port 80 of a large number of non-existent systems in order to generate ICMP time exceeded responses which enhanced their network mapping data.

Lesson: A simple mapping experiment, generating a small amount of data and performing a straightforward measurement set off alarms at many locations. In this case, and in others, a measurement experiment has the same network traffic profile as a worm looking for hosts to infect (probing port 80 is a feature of CodeRed/NIMDA). It is against the PlanetLab Acceptable User Policy to generate "disruptive" network traffic but it's sometimes hard to know what type of traffic would be considered disruptive.

"spam relay" April 17, 2003. A university's network administration reported a large amount of spam traffic being passed through a PlanetLab node hosted at the university. They disconnected the system from the network. The traffic was traced to an "open proxy" service being run on PlanetLab. The service is not a completely open proxy and in this particular case, the proxy was accepting spam relay connections to create a "honey pot". Unfortunately, the spammers thought they had found a high bandwidth relay and the number of spam relay requests grew from 3000 connections to over 250,000 connections. The honey pot for port 25 was closed.

Lesson: It's a jungle out there. Simple services (like a proxy server) or simple experiments (like a honey pot) can have large, unintended outcomes. Institutions have network monitoring tools to detect "bad guys" and it's easy to get caught in the dragnet.

"120 probes": April 22, 2003. A non-PlanetLab site complained about a network monitoring alarm showing 120 TCP probes to their network from a PlanetLab node. The alarm happened on April 21st and the complaint was emailed to the second level network administrator (the university) who forwarded it to the local technical contact who forwarded it to the support mailing list. The traffic was traced to another mapping experiment.

Lesson: Only a handful of packets can set off alarms.

"Unauthorized use of account": April 23, 2003. A complaint email was received on the PlanetLab support list complaining that an unauthorized request to change an Ebay seller's email address was made from a PlanetLab node. The person had received from Ebay an authorization for a change of their email address and the email gave a PlanetLab node (by IP address) as the source of the change request.

Lesson: Again, it's a jungle out there. This was another incident around the semi-open proxy service CoDeeN [CODEEN]. Analysis of the logs showed that the original request came from a dialup line in Romania and the there was no searching around – there were few requests and they were very specifically targeted. The logs also revealed a complicated use of some HTTP-to-TCP gateways that had even crashed the proxy service a few times. Additional safeguards were added to the proxy service to resolve the problem [CODEEN-SEC]. But, services are hard to deploy on the wild, rough-and-tumble Internet because it seems that almost anything can be subverted and used for malicious/illegal purposes.

"spambots at Princeton": May 2, 2003. A university's network administration sent email to the local technical contact about an alarm for spam traffic being forwarded through a PlanetLab node that the university. Checking the logs of the semi-open proxy service, it seemed that the proxy was receiving the request that contained spam in its body, but it was not forwarding it.

Lesson: Any traffic in or out of any system is suspect. This is one of several incidents where PlanetLab nodes were fingered for generating questionable content when actually the nodes were the receivers of the content.

"Downey Savings": May 8, 2003. The security administrator for a bank emailed complaints to several universities complaining about multiple probes of their network from multiple universities. In total, three external sites reported the reported "UDP port scans" and they sent "abuse report" emails to 6 universities. Many PlanetLab nodes were disconnected from the network.

It took several days to track the experiment generating this traffic. At the time, the SILK module counted packets sent and received by each slice and the PlanetLab administrative slice collected this packet count information once every five minutes. This meant that high volumes of traffic could be easily traced to a slice but an experiment generating a very small volume would hide in the noise of the packet count numbers. When finally found and analyzed, each sliver of the experiment was sending approximately 10 packets every 20 minutes to random high port numbers on a subset of 2000 computers external to PlanetLab. The experimenter had purposely designed the program to be low volume and low frequency so as not to create any problem.

As a long-term effect of this incident, some nodes have not been returned to service for several months because of policy concerns at hosting institutions.

Lesson: This incident held several lessons:

1. ISPs take complaints very seriously and especially if the complaining entity is a business. Better tools for local administrators to control and identify PlanetLab node traffic would help handling the complaints;
2. Experiments that are designed to be "low profile" and well-behaved can set off alarms. This is magnified by the ability to run an experiment from many locations on the Internet and thus look like a DoS attack;
3. Keeping cumulative packet counts was not sufficient to find and track network traffic at PlanetLab nodes. There is a need for a better tool for mapping network accesses into and out of PlanetLab nodes to specific slice activity and thus to a specific experimenter.

Applications and Services

PlanetLab experimenters are writing distributed programs – distributed control and distributed execution. These are hard programs to write and get correct. Misbehaving applications can use up processor cycles, use up operating system resources, and send inappropriate network traffic.

"Nodes hanging": March 21, 2003. This incident is a collection of node "hangs". The nodes could not be logged into but they were still pingable. The problem was eventually traced to an application that was using up all of the system's file descriptors. This was a bug that the experimenter quickly fixed when the problem was pointed out to them.

Lesson: individual slices should have node resource limits. Additionally, some resources should always be available for administration. In this case, one slice used up all the file descriptors and thus made the system inaccessible for both users and administration.

"disk space": March 24, 2003. Node ran out of disk space. One application had a log file that grew to several gigabytes.

Lesson: The list of resources that must be controlled and allocated to slices keeps growing. Disk space is certainly added to the list that includes network bandwidth and file descriptors.

"cmu5 sockets": May 29, 2003. Some users complained about poor performance on some nodes. This was traced to an application that had thousands of open sockets – a bug that was easily fixed when it was pointed out to the experimenter.

Lesson: it's easy for a distributed application to seem to be working when it actually is making some mistake (overusing bandwidth, disk, sockets, processor). Users could use some feedback on the amount of resources they are using and how their application is impacting the whole of PlanetLab.

3 Tool Development

Figure 3 shows the development of PlanetLab management and monitoring tools over the period from November 2002 through June 2003. Based on the lessons learned from incidents, tools were developed to alleviate the work of dealing with each type of incident.

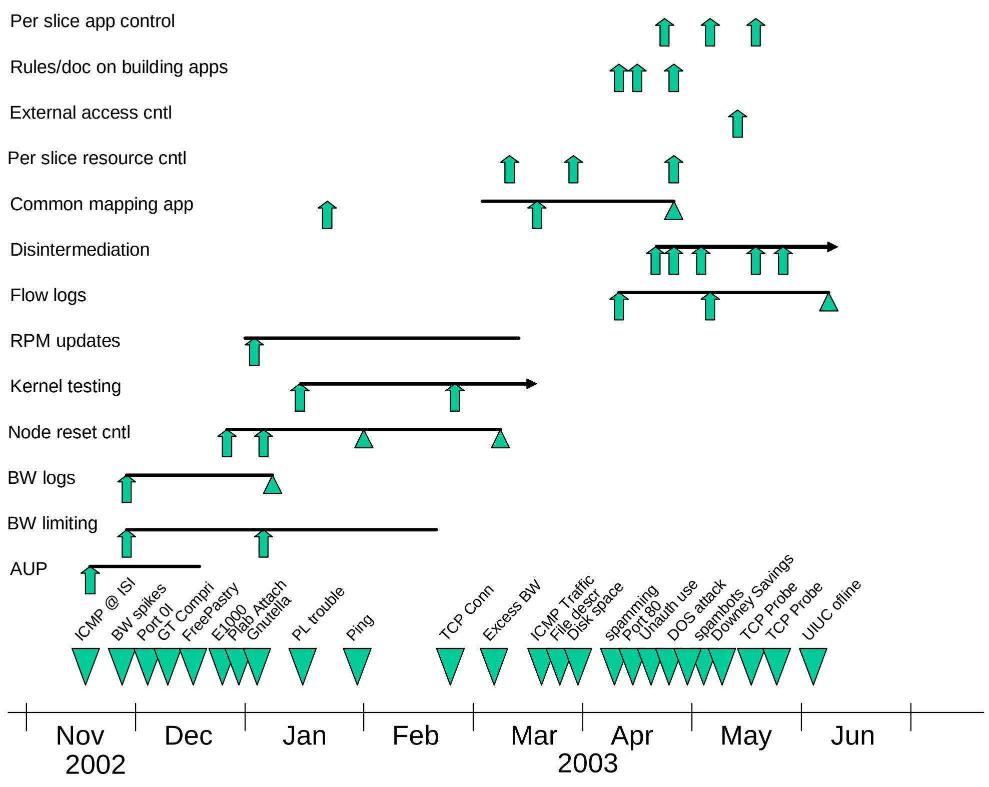

Figure 3: Development of PlanetLab management tools

Down Nodes Back In Service

PlanetLab has developed several methods to remotely restore nodes:

- Ping-of-Death
- Power Control Units
- Administrator email

"Ping-of-death" ("POD"), is a kernel patch that adds to the networking stack sensitivity to a particular type of ICMP packet. When a PlanetLab node receives a POD packet (and the packet is, of course, properly signed) the node instantly reboots. POD thus can reboot a node that is otherwise not accessible with ssh (the normal way of accessing a node).

POD doesn't work for systems with a kernel panic or are unavailable because of network connectivity problems. POD most often works on resource overuse problems where the computer is still operational; however, because of resource overuse, for instance, normal access through ssh is not possible.

About half of the PlanetLab nodes have power control units ("PCU") attached to them. These are devices accessible over the Internet that can power cycle the node. This was considered an ultimate method of restoring a node's operation. About 30% of the PlanetLab sites have PCUs installed.

The most common reason for PCU failure is mis-configuration. The local site administrators often had problems correctly installing and configuring the PCUs. The problems ranged from the wrong nodes being plugged into the wrong outlets to desktop systems remaining off when power was restored. The complete installation of the units was not tested (power cycled) which would have exposed many of these configuration problems. That step was eventually added to the installation process.

Additionally, power cycling a running Linux system often leaves the filesystem in a questionable state. If the filesystem is in a bad state *before* the power cycle (e.g., full), the power cycle won't clear the bad state. Also, if a power cycle caused the bad state in the filesystem, the rebooting process often would not be able to correct the problem.

If all else fails, email is sent to the local system administrators requesting the nodes be rebooted. This always successful but does sometimes take several days to get a node back online.

Tracing Traffic

PlanetLab's geographic distribution makes it ideal for mapping the Internet. It seems that most researchers first build a “Hello world” application that pings other PlanetLab and non-PlanetLab nodes to discover timing and connectivity information. Repeated pings, IP address space scanning and port scanning are just the activities that set off Snort, BigBrother and other network monitoring tool alarms. Even some “well designed” probing applications have set off alarms implying that some sites have very tight restrictions on probing and mapping activities.

To handle an inappropriate traffic incident, the problem is mapping the reported activity from the network traffic to the experimenter. A traffic report usually contains a time and a source and destinations IP address. Ideally, a tool that maps this information to a specific experimenter would be ideal.

The mapping is further complicated by the need to map network traffic that happened in the past – PlanetLab Support may not receive the report of inappropriate traffic report until several days after the actual event.

Most conventional traffic monitoring tools are for administration of data centers or ISP subnets. These often “snoop” on the network traffic on a subnet and report statistics and anomalies. Because PlanetLab nodes are distributed around the world on different subnets, tools for snooping on this distributed traffic are not readily available and would be difficult to deploy because of the different administrative domains

These problems (mapping, delay and distribution) motivated the development of tools that have each node collecting information on its own network traffic (in and out), saving that information and eventually reporting that information to a central repository.

At first, the SILK module counted network packets that were sent and received by individual slivers on a node and made these counts available in the `/proc` filesystem. A data collection application named “scout-monitor” ran in an administrative slice. Scout-monitor collected the packet counts into log files every 5 minutes. Every few hours, these data files are collected by PlanetLab Support, analyzed, and made available on the web. The PlanetLab web site contains daily updated tables showing the top 10 consumers of network bandwidth by slice and the top 10 consumers of network bandwidth nodes.

This packet count information was often sufficient to trace reports of traffic to slice usage but this was a very manual operation and tended to fail if the reported traffic was just a few packets.

To solve both the problem, the SILK module was enhanced to return information on which IP addresses the slivers were sending and receiving traffic from. An administrative application named “netflow” analyses this information every 5 minutes and calculates the “flows” – the connections from a source to a destination by some slice. This information is saved to a file. These files are kept on the node and are eventually copied to PlanetLab Support where they are available for analysis if problem reports arrive.

This flow information makes it very easy to trace from a network traffic report (IP address at some time) to the slice that created that traffic and thence to the experimenter responsible.

Disintermediation

As has been mentioned previously, a reported network problem follows a long path from the originally offended party to the experimenter who created the traffic. The steps are usually from the remote administrator to the institution network administrator (through ‘abuse’ mailing lists) to the local PlanetLab system administrator to PlanetLab Support. PlanetLab Support analyzes the report and identifies the slice generating the traffic. Email is then sent to the experimenter and the problem is resolved.

This process has many problems:

- It can take one or more days to resolve a problem;
- Several people are contacted who’s only job is passing the data through;
- The experimenters are buffered from the effects of their experiments;
- The job of analysis falls totally on PlanetLab Support

When network traffic sets off an alarm at some site, they have a source IP address that must be mapped back to the source’s administration. Most people do a reverse DNS lookup or *whois* to come up with a domain name that will accept an email. Currently, PlanetLab nodes get their DNS support from their hosting institution. This means that complaints about questionable network traffic go to the network administration of the hosting institution.

That is, the person receiving the email about the network traffic problem is not the person generating the traffic. This points out that there is no standard way of mapping a source IP/port address to an application.

PlanetLab has explored several ways to try and remove the people in the middle and allow the person generating the report to go directly to the experimenter.

Some of the PlanetLab nodes have had their reverse DNS lookups modified to point back to PlanetLab Support. This eliminates the communication through the hosting institution allowing the report to go directly to someone who can find the source application. This is not widely deployed because most sites have DNS controlled at the institution level and adding special lookups for individual systems is not easy.

The most successful tool has been making 'netflow' information available. Each PlanetLab node has a web server that displays source and destination IP address to slice mapping. Additionally, there is an email link on the page that will send email to the PlanetLab accounts assigned to that slice.

Thus, in the best case scenario, someone notices a network traffic problem, they browse the web page on the source PlanetLab node, identify who generated that traffic and sends email directly to that person. This eliminates all of the intermediate people and leads to a speedy resolution to the problem.

The people in the middle need to know if there are problems occurring in their networks, so some accommodation is made for this. For instance, the email address for sending email to the slice owner is actually an alias at PlanetLab Support that also sends email to the node's local technical contact and the PlanetLab support email lists.

This process of disintermediation must also include the education of users, site administrators and anyone who might notice network traffic problems – they must know to use this PlanetLab specific mapping mechanism and has lead to the formation of an "IT Advisory Board" made up of local site administrators to assess the best methods for resolving these problems.

Controlling Resources

PlanetLab experimenters are writing distributed programs – distributed control and distributed execution. These are hard programs to write and get correct. Misbehaving applications can use up processor cycles, use up operating system resources, and send inappropriate network traffic. When applications misbehave, they must be managed.

We took two approaches to this: proactive and reactive. Proactive means limiting the things an application can do wrong. Reactive means having the tools to stop or limit applications that go bad.

Proactive tools under development include per slice network bandwidth and system resource limits.

Node bandwidth limits are in place but these are not tunable for individual users. The default Linux system does not control system resources (like CPU time, file descriptors, memory) to the expanded virtual server system we are using.

Work is in progress to build a mechanism for low-level resource limiting and allocation on a sliver basis. The system being developed additionally enables in resource allocation algorithms (economic models, barter systems, etc) [SLICE].

The reactive tools allow a privileged administrator to shut off offending applications. There are at least four classes of administrators:

- The developers themselves
- Local site administrators
- Responsible principal investigators (PI's)
- PlanetLab Support administrators

Normally an application stays in control, but some times they "run away" and need to be stopped. The developers themselves need a tool to "pull the plug" on a program that has gotten away from them. This is difficult because some types of failures (extreme network traffic, for instance) make it nearly impossible to log into the node to stop the application.

The local site administrators have the ultimate ability to pull the plug on computers that have "run amok" but they also need some tools to discover the source of the problem, discover a contact to notify about the problem and, in extreme cases, to stop the offending program.

PI's also need some control over the activities that they are responsible for and should have to ability to shutdown the activity of a slice that is misbehaving.

PlanetLab Support, who has the responsibility to keep PlanetLab running, must be able to stop slices that are effecting the operation of PlanetLab and, for extreme problems, have a "big red button" that terminates the operation of offending slices, nodes and possibly the whole system.

These requirements have been approached with a several levels of mechanisms. The local site administrator has a PlanetLab account for a "management slice" on their PlanetLab nodes. The administrative slice lets them look at log files, running processes and stop and reboot the node. Their account also gives access to web pages at PlanetLab Support that use all of the remote node restart features (POD, PCU) to stop a node. This feature is also used by PlanetLab Support to stop a node.

Remotely logging into the nodes and killing their running processes stops individual slices. PlanetLab Support administration has tools that can thus stop a slice that is creating problems by killing the processes on all nodes. A subset of this functionality will eventually be available to PI's.

4 Conclusion

These "incident" reports show up many interesting dimensions of distributed system management:

- Maintenance of remote and distributed hardware – how best to diagnose and repair;
- Software system has bugs and failures – how best to proactively and reactively handle these;

- Network monitoring – how to monitor network traffic on a non-centralized system;
- Disintermediation – how to create a path from a network traffic report to the responsible person and application without involving a large set of intermediate people.

  They have also shown some observations about the Internet:

- Network monitoring is wide spread on the Internet and the pattern of acceptable traffic is fairly narrow;
- There are unknown implications to experiments. This ranges from setting off alarms to being a target for questionable activity;

PlanetLab has addressed these dimensions and observations by the development of tools for managing itself with the general goal of creating a testbed for further experimentation and refinement of distributed systems management.

5 References

[AUP] PlanetLab Acceptable Use Policy http://www.planet-lab.org/php/aup/ .

[CODEEN] Research project at Princeton University. http://codeen.cs.princeton.edu/ .

[CODEEN-SEC] Vivik S Pai, Limin Wang, KyoungSoo Park, Ruoming Pang, and Larry Peterson. The Dark Side of the Web: An Open Proxy's View. Submission to ADM HotNets-II, October 2003.

[PLANETLAB] Larry Peterson, Ton Anderson, David Culler and Timothy Roscoe. A Blueprint for Introducing Disruptive Technology into the Internet. In *Proceedings of ADM HotNets-I*, October, 2002. http://www.planet-lab.org/pdn/pdn-02-001.pdf .

[SILK] Andy Bavier, Plkmod: SILK on PlanetLab. http://www.cs.princeton.edu/~acb/plkmod/ .

[SLICE] Brent Chun, Tammo Spalink. Slice Creation and Management. PlanetLab PDN-03-013: http://www.planet-lab.org/pdn/pdn-03-013.pdf .

[VSERVER] Open source development project: http://www.linux-verver.org/ .

Appendix A: Major Incidents from November 2002 until June 2003

AL – network monitoring alarm that mis-identified an experiment at abuse
AUP – application went against acceptable use policy
BUG – application built wrong
INF – problem with infrastructure hardware/software
BW – bandwidth problems
HW – hardware
RES – resources

AL	14 Nov 2002	PL node sending ICMP packets to router every two seconds for two weeks CMU doing measurement. Network mapping One machine disconnected.	Talk of putting machines behind firewall. Realization of need for AUP. Added tiny web server and finger to each node.
BW	26 Nov 2002	"Bandwidth spikes". Video overlay network. Problems with coding and order effects. DMZ at Berkeley taken down.	Need for bandwidth limiting identified
AL	27 Nov 2002	"Port 0" Traffic to port 0 set off 5 university and 2 ISP snort, etc alarms. Nodes taken offline.	Feature requests for remote console, remote power control, failures database
AL	27 Nov 2002	"packet spikes from millennium" Spikes of traffic from PL nodes to external hosts and to addresses that don't exist	
AL	2 Dec 2002	"GT Compromised" Admins noticed smurf packets from PL nodes	Deduced that this was old reports on 27 Nov incident
BW	6 Dec 2002	"Seattle meltdown" PL nodes using all bandwidth. Disconnected Experimenter measuring max throughput	Need bandwidth limits
BW	11 Dec 2002	"FreePastry incident" 5 Pastry rings causing gobs of traffic during OSDI	
INF	19 Dec 2002	"One node from each site" E1000 driver Over the next few weeks, lost nearly 30% of PL nodes.	Thought it was related to Scout driver since didn't have console output. This happened over several weeks repeatedly loosing many nodes at one time. Updated e1000 driver at all nodes the end of Jan.
AL	20 Dec 2002	"Planetlab Attach" ScriptRoute mapping experiment set off ISPs alarms. Continuous probing of high port numbers. Several nodes taken offline	Had to explain to sysadmins that PL was experimentation and that actions were not DOS attacks.
AL	24 Dec 2002	"Attack major web sites" Time mapping experiment by Princeton on uky PL node. Pinging port 80. Experiment/attack ran for 5 minutes.	Detected high flows to multiple sites. Found: machine and when mapped to traffic pulses by slices. Scout-monitor.
INF	30 Dec 2002	"Cron hangage" All nodes hung in cron.daily.	Update was taking many hours since all nodes were being updated.

AL	31 Dec 2002	"Gnutella1" Net monitors noticed Gnutella traffic ("Limewire") which they do not allow	Researcher as running the index/search part of "Limewire" to watch Gnutella query traffic. Attempts to tell admins how to use sudo commands to check logs. Found: noting who was logged in at the time and searching filesystem for code being run.
BUG	6 Jan 2003	"eating cycles" Complaint that some experiment is using up lots of cycles on several nodes	Talked to researcher to mod application
BW	12 Jan 2003	"Canterbury bandwidth" 6GB of data in two weeks – expensive Nodes shut down	Experiment of distributed storage of large data files. LoKI storage system. Looking at bandwidth in general. Added compression to infrastructure (Ganglia).
AL	15 Jan 2003	"Planetlab trouble" Sustained bombardment of single machine	Also illuminated Cambridge bandwidth expense problems
HW	27 Jan 2003	"UCLA memory" Memory errors. Node replaced. Process took about a month.	
AL	28 Jan 2003	"Ping attack" Persistent, high-volume pings took out border routers at Berkeley site	Experiment failure – one sliver reporting to log node, log node fails, reporting sliver persistent causing ICMP 'port not found' messages.
INF	29 Jan 2003	"nodes rebooted" PL core user noted "many" node rebooted PL developer rebooted several machines while driver developing	
HW	10 Feb 2003	"interface up/down" Network interface on one node repeatedly going up and down. Resolved by moving node network connection	
INF	12 Feb 2003	"fragmented packets" Experimenter noted that one node drops all packets > 1500 bytes.	Starting to get questions about what's "around" PL nodes – throughput of routers, …
INF	27 Feb 2003	"Berkeley offline" Machines crashed with "shim_socket failure". Rebooted.	
INF	27 Feb 2003	"TCP connection problem" Experimenters started complaining about programmatic failure using sockets. 4 different researchers.	New version of scout module that wasn't totally complete. Started discussion regression testing.
INF	4 Mar 2003	"name problem" Experimenter noticed /etc/hosts on machine configured wrong	Hand corrected problem. Some node configurations very slightly due to manual fixes.
BW	4 Mar 2003	"utexas excessive bandwidth" Net admin noted node generating > 3.5Mbps continuously normal usage taken as abuse – reassured net admins	
INF	6 Mar	"weird socket interaction"	

	2003	raw socket implementation bugs	
HW	10 Mar 2003	"Stanford memory" Node filed repeatedly because of memory parity errors	
INF	13 Mar 2003	"Stanford memory" Machine crashed multiple times with memory errors. Service required	
INF	13 Mar 2003	"Basil node installation" Problems with installing nodes on new HW configuration. After 3 months, still not resolved	
INF	19 Mar 2003	"filtering ports" Experimenter noted that ports are filtered inconsistently for each of the PL nodes. Nothing done.	
AL	20 Mar 2003	"ICMP traffic" Net admin at two sites note large volume of ICMP traffic from/to PL nodes	
BUG	21 Mar 2003	"uiuc3" Hanging of many (10) PL nodes. Caused by coding bug that used up all file descriptors. Manual deletion of run-away processes and machine rebooting.	Need for per-slice resource allocation
BUG	24 Mar 2003	"disk space" One slice is using all available disk space on some nodes	
INF	24 Mar 2003	"NTPD incompatible with scout" NTPD's use of sockets doesn't work with latest scout module. Module debugged and updated	
BUG	7 Apr 2003	"sydney1" Experiment used all resources. Required manual resetting of 13 nodes. Program newly failing because of change in raw socket operation.	
INF	9 Apr 2003	"write on socket failure" Experimenter found problem with sockets. Bug in raw socket code.	
AL	15 Apr 2003	"port 80 scanning" Complaints from 3 net admins on scanning of nodes outside PL. Some nodes down for over a week.	Traffic hard to find in logged BW data (too small). Need flow information
AL	17 Apr 2003	"spam relay" Report that PL nodes being used as spam relays CoDeeN nodes were accepting CONNECT requests to create honey pot. Spammers thought they had an open relay so traffic increased fantastically.	
BW	17 Apr 2003	"rate limit" PL nodes at one site rate limited to smooth out site traffic. The PL nodes were using too much BW.	
AL	18 Apr 2003	"potentially infected" Monitoring sw detected possible Nimda attack (scanning IPs for port 80) from PL nodes. Mapping experiment that should have just used traceroute.	
AL	19 Apr	"possible Nimbda"	Because it can take day for a

	2003 21 Apr 2003	Monitoring sw detected possible Nimda from PL nodes More reports of port 80 scanning.	report to bubble through all the people, there are many waves of the same problem.
AL	22 Apr 2003	"port 80 outside PL" Network sw detecting PL nodes accessing many computers. More traffic through CoDeeN	
AL	22 Apr 2003	"120 probes" Person on dialup line complaining that PL nodes are probing his system	Hard to identify traffic because of low volume. Some admins upset because there was no "prior consent" from packet receiver. Talk of limiting outside PL access
AL	23 Apr 2003	"Unauthorized use of account" Use of CoDeeN proxy in scheme to steal accounts. PL node was identified because account's site report IP addr of computer request coming from.	CoDeeN
AL	24 Apr 2003	"DOS attack" Multiple PL nodes doing a GET on a node outside PL. "76 attacking machines making between 31 and 222 webpage requests" "Http_load" experiment.	
AL	25 Apr 2003	"Accessing porn" Network software noticed PL node accessing a questionable site. CoDeeN was acting as a proxy.	CoDeeN
AL	25 Apr 2003	"Open proxy" Report from "JSTOR" licensed journal archive that HTTP requests were coming from PL nodes. Hackers accessing JSTOR through CoDeeN proxy.	Questions about external access to CoDeeN
AL	29 Apr 2003	"high traffic" PL Support noticed large amount of UDP traffic and apps using hundreds of sockets. Experimenter found bugs in their application.	
AL	30 Apr 2003	"spam on IRC" Network sw noted PL node sending spam to IRC ports Close relationship with experimenter lead to quick resolution.	CoDeeN. Application gets security features added and is brought back online on 2003-05-21.
AL	1 May 2003	"2 DOS" Umich non-PL hosts hit with "DOS attacks" from multiple PL nodes. Umich unplugged.	Cmu5 experiment run amok
AL	2 May 2003	"spambots at Princeton" Network sw noted spam messages. Was spam sent to odd ICQ ports. Reconfigured CoDeeN to not allow tunneling	Discussions about how to get from attackee to experimenter.
AL	8 May 2003	"Downey Savings" Network sw detected "attack signatures" – port scans on nodes outside PL. Reported from several sites. Three sites taken offline. One site offline after one month.	Experimenter had thought he had made a well-designed and friendly measurement application. Any port scanning was done like traceroute. More than thousand nodes were measured, only three complained. Discussion about control of

			access outside PL.
AL	15 May 2003	“TCP Port Probe” Probing host outside PL.	Killed experiment to eliminate offending traffic.
AL	27 May 2003	“TCP Port Probe redux” Probing hosts outside PL.	First use of putting iptables filters on nodes – faster response to problem
BUG	29 May 2003	“cmu5 sockets” Complaints about performance on some nodes. Found and app using thousands of open sockets.	Bug in code
AL	2 Jun 2003	“UIUC offline” Campus exit complaining about too much ICMP traffic from PL node. Turned out to not be a PL node	
BW	2 Jun 2003	“Lots of traffic at MIT” Sysadmin noting 4000 packets/sec from PL nodes. Moved nodes to router with better reporting. Pointed to ‘netflow’ info.	
BW	2 Jun 2003	“ucb5 traffic” Report of large amounts of network traffic.	Experiment moved > 1TB of data
AL	3 Jun 2003	“irregular traffic” Responded with description of PL experiments	
HW	3 Jun 2003	“IT TRANSIT”. Routers at Ashburn stopped working	
NET	4 Jun 2003	“Routing to Sidney” Nodes seemed to be blocking ssh traffic.	
AL	6 Jun 2003	“Security concern”. One node doing scanning of high ports on many IP addresses	
ACCT	6 Jun 2003	“creating new account”. PI asking about creating accounts for students	
AL	6 Jun 2003	ICMP 'echo request' with data, non-standard TTL of 61, UDP with frag and don't frag bits set. Packet rate 5-10 pkts/sec.	
NET	6 Jun 2003	“HP and PlanetLab”. Configuring routers and systems at installation and for Internet2	
RES	8 Jun 2003	“millennium out of space”. Node ran out of disk space.	One slice had a 9.7BG log file
APP	1 Jun 2003	“problems with PlanetLab” – researcher noticed getting garbage information back when talking to many nodes	Application was not checking for fragmented packets.
HW	10 Jun 2003	“Princeton oops”: two nodes wouldn’t reboot because of errors from the RAID controller	
INF	22 May 2003	“Seattle bootCD”. Disk became corrupted, system needed reinstall. Tested installation with new bootCD.	Several weeks getting the bootCD working because of a SCSI disk controlled not supported by the bootCD
AUP	10 Jun 2003	“Berkeley portscan”. University security reported a portscan from a PL node	Slice sent 30 packets to reported IP address and other random IP addresses. Ruled against AUP
AL	10 Jun 2003	“Copyright Infringement”. Report that a PlanetLab node was running a cracked version of commercial software.	Researcher examined CoDeeN logs to discover true source of application usage.
AL	10 Jun 2003	“Jump in UDP traffic”	Pointed administrator at traffic info page on nodes

??	11 Jun 2003	"Rootkits" – request from site administrator to have special rootkit RPMs installed on the PL nodes	Special dispensation was given