PLANETLAB

# Using PlanetLab for Network Research: Myths, Realities, and Best Practices

Larry Peterson, Vivek Pai
Princeton University

Neil Spring
University of Maryland

Andy Bavier
Princeton University

PDN–05–028
June 2005

Status: Final Version.

# Using PlanetLab for Network Research: Myths, Realities, and Best Practices

Larry Peterson, Vivek Pai, Neil Spring, and Andy Bavier

June 29, 2005

PlanetLab is designed to be used by a wide range of network services and experiments. There are 428 slices running on PlanetLab (June 2005), and there are likely just as many different perceptions about what PlanetLab does and does not provide as a network platform. Some of these perceptions are true, some were true but are no longer valid, and some depend on researchers applying best practices. This report identifies some of these perceptions, discusses whether they are myth or reality, and describes best practices that can be applied to make PlanetLab as effective a research platform as possible.

## Perception: PlanetLab is not suitable for reproducible results.
## Verdict: Reality.

PlanetLab was never designed to be used for *controlled* experiments. It was designed to subject network services to real-world conditions. By running a service for months or years, researchers should be able to identify trends and understand the range of performance and reliability properties their service achieves. Any experiment that runs only for an hour will reflect only the conditions of the network (and PlanetLab) during that hour.

Various aspects of a service can be measured, of course, by avoiding heavily loaded times, using one of the available brokerage services (see below) to secure sufficient resources, and repeating each experiment often enough to generate statistically valid results.

**Perception: High load prevents accurate latency measurements.**
**Verdict: Myth, if best practices are used.**

Because PlanetLab machines are heavily loaded, no application can expect that a call to `gettimeofday()` right after `recv()` will return the time when the packet was received by the machine. Using in-kernel timestamping features of Linux, however, network delay can be isolated from (most) processing delay.

When a machine receives a packet, the network device sends an interrupt to the processor so that the kernel can pull the packet from the device's queue. At the point when Linux accepts the packet from the device driver, it annotates the buffer with the current time.[1] The kernel will return control to the current process for the remainder of its quantum, but this timestamp is kept in the kernel and made available in at least three ways:

1. The SIOCGSTAMP ioctl called after reading a packet gives the timestamp of the last packet. Code to use this ioctl is part of ping, but Linux kernel comments suggest the call is Linux-specific.
2. The SO_TIMESTAMP socket option combined with `recvmsg()`, in which the ancillary data includes a timestamp. The Spruce [5] receiver code uses this method. This approach may also be Linux-specific and is not widely documented, but can be run as a non-root user.
3. The library behind tcpdump, libpcap, gets these timestamps on the socket it uses to see all packets. This is the most portable option, but requires that you be root, which is easy on PlanetLab. The libpcap library has the added advantage that *sent* packets are also timestamped [6].

There is one reality, however: sending packets at precise times is more difficult, but still quite possible with some adjustments. Although PlanetLab nodes are often heavily loaded, processes that need to send packets at specific times can also measure when they have control of the processor. If the process is willing to discard measurements where the desired sending times were not achieved, then sending rate-paced data on PlanetLab simply requires more attempts than on unloaded systems.

To determine how load impairs the ability to send precisely, we measure how often we are able to send precisely-spaced packets in a train. Sent trains consist of eleven packets, spaced either by 1 ms, to test spin-waiting, or 11 ms, to test sleep-based waiting. We show how often the desired gaps were achieved for 1 ms gaps in

[1]See: linux/net/core/dev.c:netif_rx().

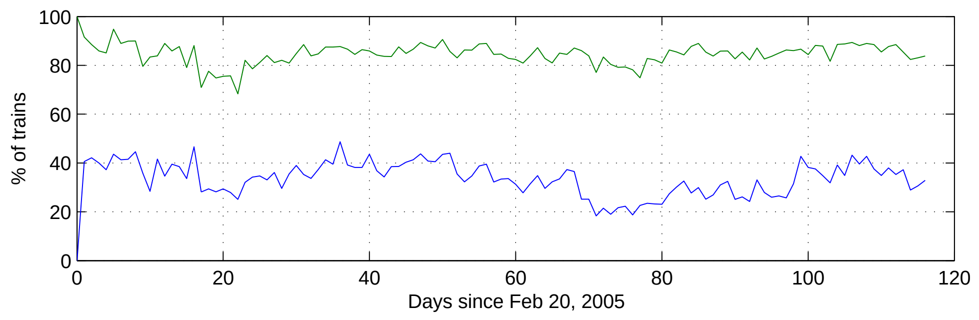

Figure 1: Timing statistics for 1 ms (spin-based) chirp trains. The green (upper) line indicates at least 5 consecutive gaps met the target timings, while the blue (lower) line indicates all gaps met the target.

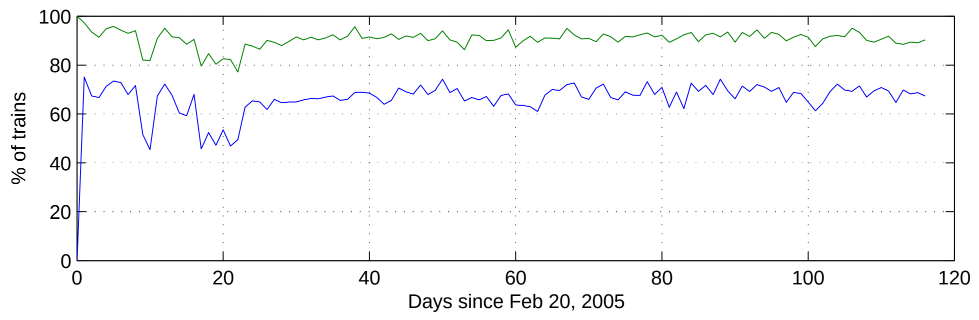

Figure 2: Timing statistics for 11 ms (sleep-based) chirp trains. The green (upper) line indicates at least 5 consecutive gaps met the target timings, while the blue (lower) line indicates all gaps met the target.

Figure 1, and show 11 ms gaps in Figure 2. These packet spacings were achieved using spin loops for the 1 ms gaps, and the `nanosleep()` system call (via the `usleep()` library call) for the 11 ms gaps. In all measurements, 10 gaps are used, and we measure how often the gaps are within 3% of the target either for all 10 gaps, or for any 5 consecutive gaps.

For both tests, at least five consecutive gaps have the desired intervals in 80–90% of the trains. For the 11 ms test, all 10 gaps had the correct timing 60–70% of the time. The 1 ms test did not fare as well: all 10 gaps met their target times in only 20–40% of the trains. For the shorter (5-gap) chirp trains, the results are quite good: sending 10 packets is sufficient to discard less than 20% of the measurements. For longer chirp trains, two to five times as many probes may have to be sent, but this overhead may be tolerable for many experiments.

Mechanisms for negotiating temporarily longer time slices or delegating packet transmission scheduling to the kernel are being discussed on PlanetLab's arch mailing list.

## Perception: The PlanetLab AUP makes it unsuitable for measurement. Verdict: Myth, if best practices are used.

The PlanetLab user Acceptable Use Policy [3] states:

> PlanetLab is designed to support network measurement experiments that purposely probe the Internet. However, we expect all users to adhere to widely-accepted standards of network etiquette in an effort to minimize complaints from network administrators. Activities that have been interpreted as worm and denial-of-service attacks in the past (and should be avoided) include sending SYN packets to port 80 on random machines, probing random IP addresses, repeatedly pinging routers, overloading bottleneck links with measurement traffic, and probing a single target machine from many PlanetLab nodes.

This policy is a result of experience with network measurements on PlanetLab, and is designed to prevent complaints of the form "PlanetLab is attacking my machine." Here we elaborate on steps to conduct responsible Internet measurement on PlanetLab. The goal of these practices is to make network measurements as easy to support as possible by building a list of hosts that "opt-out" of measurement without growing the list of PlanetLab sites that have requested to "opt-out" of hosting measurements.

**Have recognizable traffic.** Send packets with a consistent, uniquely-identifying source or destination port. This will make it easy to recognize traffic that is part of your experiment from the first email.

**Test locally.** Do not use PlanetLab to send traffic you wouldn't send from your home institution. Use a machine at your home institution first, so that if there are problems with your tool, you may discover them without causing network-wide disruption.

**Alert PlanetLab support.** Go beyond updating your slice description:[2] send a message to PlanetLab support detailing your intended measurement, how to identify its traffic, and what you've done to try to avoid problems. Occasionally the interface that traces packets back to their owners breaks; helping support avoid that saves them valuable time.

**Start slow.** Measurement run from PlanetLab can easily look like a distributed denial of service attack; starting with a few machines reduces the number of sites that receive complaints about denial of service attacks.

**Use Scriptroute.** Scriptroute exists as a layer for separating measurement logic from low-level details of measurement execution. It will prevent you from contacting hosts that have complained about traffic previously, can prevent inadvertently invalid packets that trigger intrusion detection systems, will limit the rate of measurement traffic sent, collects pcap-derived timestamps, and schedules probes using a hybrid between sleeping and busy-waiting. Scriptroute includes a tool for collecting the tree of paths from all PlanetLab nodes to a destination, using as few probes as possible; using this tool, you can traceroute from everywhere to a destination without looking like a distributed denial of service attack.

**Curtail ambition.** It is tempting to demonstrate implementation skill by running a measurement study from *everywhere* to *everywhere*, using many packets to get as accurate a result as possible, and using TCP SYN packets to increase the chance of discovering properties of networks behind firewalls. Resist! Overly-aggressive measurement increases the cost of the measurement and risk in reputation for only a marginal benefit to the authority of your result.

## Perception: PlanetLab is too heavily loaded to be usable.
## Verdict: Myth; once true, but not today.

While PlanetLab is likely to be chronically under-provisioned, and load can be especially high just before conference deadlines, this perception is misleading in two ways.

[2]To update your slice description, visit: https://www.planet-lab.org/db/slices/select_slice.php?dest=update_desc.php

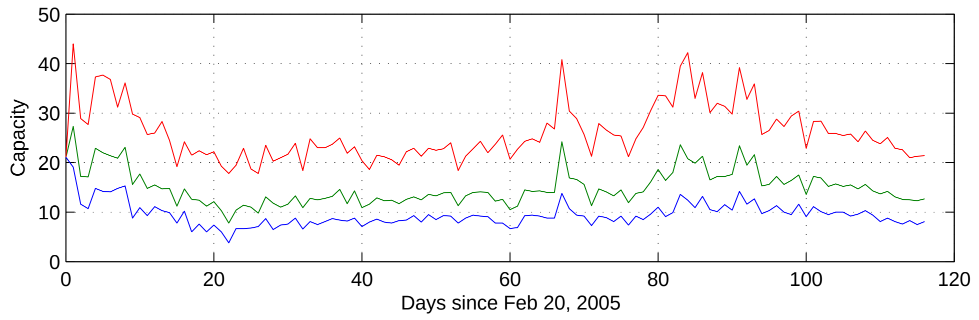

Figure 3: Available CPU across PlanetLab nodes. Median percentage available CPU is red (upper), $25^{th}$ percentile is green (middle), and $10^{th}$ is blue (lower).

First, PlanetLab now has two brokerage services (Sirius and Bellagio) that perform admission control to a pool of resources. Researchers can use these services to receive more than a "fair share" of the CPU—for fixed periods of time—during periods of heavy load.

Second, recent upgrades to the OS have made the system behave much better than when PlanetLab was first deployed. For example, CPU usage is charged to slices rather than threads, meaning that a slice with 100 threads no longer receives 100 times the CPU capacity as a slice with one thread; both slices now receive an equal share of the CPU. This often means the Unix-reported load average is misleading: a load of 100 means there are 100 runnable threads, but a slice with 99 runnable threads receives *the same* CPU allocation as a slice with only 1.

Also, PlanetLab's ability to police and kill slices that are using an excessive amount of memory has forced programmers to be careful about memory consumption, reducing memory pressure for everyone. Finally, an OS upgrade has allowed the system to use DMA for disk I/O rather than programmed I/O. This has been especially important when the node is swapping.

Experiments conducted during the run-up to the SIGCOMM deadline support the claim that PlanetLab has sufficient CPU capacity. Specifically, 360 of the 362 nodes (99%) running during the week of February 1–8, 2005 had an average of at least 10% of their processors available; 328 of the 360 nodes (91%) had at least 20% available. The processors certainly had a high load during this period (often exceeding 100), but a slice that wanted to consume its fair share of the CPU during this period would have received at least an average of 10% on virtually all of PlanetLab's nodes.

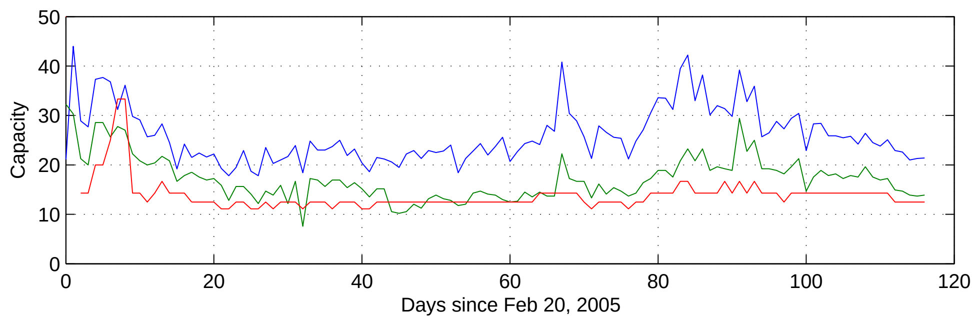

Figure 4: Median available CPU measurements using spin loops (blue, upper), load average (green, middle), and number of active slices (red, typically lowest).

This trend holds true across longer periods as well, as shown in Figure 3. The three lines are the median, 25th, and 10th percentiles of the available capacity across all nodes. Over this period, the majority of nodes showed at least 20% available capacity, while less than one-fourth had less than 10% free. Because of PlanetLab's fair scheduler, this approach presents a more realistic estimate of capacity than standard techniques, such as the load metric reported by top. For comparison, in Figure 4, we show the median capacities as measured directly using spin loops, via the inverse of the load average, and the inverse of the number of active slices. The top line, the true capacity, is significantly higher at almost all times. The reason for this difference is simple: not all active slices use their entire quanta, so the active slice count can give an inflated estimate. Likewise, slices that spawn many processes will increase the load average, but their processes only compete against each other for a fair share of the processor.

## Perception: PlanetLab cannot make resource guarantees to slices. Verdict: Myth; once true, but not today.

Resource guarantees could not be given before Version 3.0. Schedulers and other mechanisms are now available to make resource guarantees, but we do not yet have a policy about what slices should receive such guarantees. Typically, continuously running services on PlanetLab are robust to varying resource availability (and have not asked for guarantees), while short-term experiments have the option of using one of the admission control mechanisms (see previous item) to ensure they receive sufficient capacity for the duration of a run. Once we have enough experience to understand what guarantees make sense from a policy perspective, or someone develops a robust market in which users can acquire resource guarantees, resource guarantees are likely to become commonplace.

## Perception: PlanetLab is not representative of the Internet.
## Verdict: Reality; but what is representative of the Internet?

Exactly what it means to be representative of the Internet is an open and interesting research question. A good definition could help drive node placement in the future, and help researchers select which existing nodes they want their slice to run on. Until there is a widely-accepted definition, all results needs to interpreted according to the service's or experiment's sensitivity to various network properties.

Statements about PlanetLab not being representative of the Internet are usually taken to mean that too many of PlanetLab's nodes are connected to Internet2, or more generally, the global research and education network (GREN) [1]. While there have been modest improvements as commercial sites join PlanetLab and research sites connect machines to DSL and cable modem links (26 sites are purely on the commercial Internet), it remains the case that most PlanetLab sites are on the GREN. The question we need to address is how PlanetLab's network connectivity affects research. Consider the following examples.

Research that claims some new routing technique is able to find better routes than BGP are suspect if those better routes take advantage of well-provisioned research networks. On the other hand, claims that a service can find the best available route—as opposed to finding a better route than BGP—might be accurate even when running on the GREN.

Research that uses PlanetLab as a platform for observing the behavior of the rest of the Internet is not particularly handicapped by having to make those observations from the GREN. This is because the 60% of the autonomous systems that host PlanetLab nodes are either transit ASes (e.g., PlanetLab has multiple nodes on FASTNET) or multihomed ASes (e.g., they peer with both Internet2 and some commercial ISP). Of course, the other 40% are also able to reach the commercial Internet, but generally through a single intermediate AS. Packets destined to non-GREN sites are always routed over the commercial Internet, with PlanetLab nodes communicating with an average of 565,000 unique IP addresses a day. The bottom line is that measurement services like Scriptroute [4] have sufficient vantage points from which the full Internet can be probed.

Network services that attract real users—including users not at a current PlanetLab site—are also not unduly limited by PlanetLab's current topology. Moreover, this traffic can be the source of even greater information about network behavior. For example, by watching TCP connections between CoDeeN nodes at PlanetLab sites and Web clients/servers throughout the Internet, PlanetSeer was able to observe traffic traversing 10,090 ASes, including all tier-1 ISPs, 96% of the tier-2 ISPs, roughly 80% of the tier-3 and 4 ISPs, and even 43% of the tier-3 ISPs [7].

Finally, it is sometimes not the topology of the GREN that is most problematic, but it is the availability of unreasonably high bandwidths that calls results into question. We note, however, that slices are free to limit bandwidth they consume to emulate a lower bandwidth link.

## Perception: PlanetLab is not suitable for peer-to-peer networks. Verdict: Myth?

This is partly a different way of saying that PlanetLab is too I2-centric (e.g., it does not include enough DSL links), but it also highlights the point that PlanetLab is a managed infrastructure, and so not subject to the same churn as desktop systems (see next item).

One view is that while PlanetLab is not equivalent to a set of desktop machines—and it is not expected to scale to millions of machines—it is suitable as a “seed” deployment for a P2P service. This “seed deployment” allows researchers to show the value of their service and encourage end-users to load the service on their desktop machine. Similarly, PlanetLab nodes might be viewed as the “super nodes” of a P2P network. End System Multicast uses PlanetLab in this way [2].

## Perception: PlanetLab experiences excessive churn. Verdict: Myth; if best practices are used.

Major software upgrades aside, roughly 30% of PlanetLab’s nodes are down at any given time. Roughly one-third of these are down for several weeks, usually because a site is upgrading the hardware or blocking access due to an AUP or security issue. The remaining failed nodes are part of the daily churn that typically sees 15–20 nodes fail and an equal number of nodes recover on any given day.

There have been three times during the last two years when many PlanetLab nodes were down: (1) a security incident in December 2003 caused all the nodes to be taken off-line for a week, during which time we also upgraded the system from Version 1.0 to Version 2.0; (2) an upgrade from Version 2.0 to Version 3.0 during November 2004 caused more churn than usual for a two week period; and (3) a

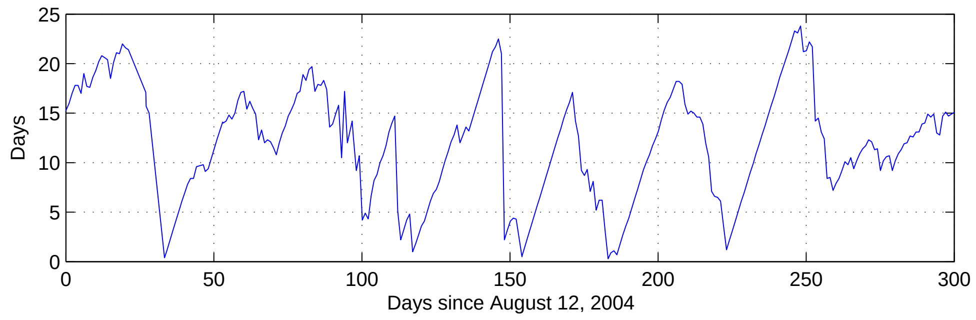

Figure 5: Median uptime in days across all PlanetLab nodes.

kernel bug in February 2005 took many nodes off-line for a weekend. Figure 5 shows that median uptimes are primarily affected by these testbed-wide events. Median uptimes are generally higher than 5 days, and often approach 15 to 20 days—much higher than what would be expected in typical home systems.

Given any churn, no users should expect that the storage offered by PlanetLab nodes is persistent, or that any set of machines, once chosen, will remain operational for the duration of a long-running experiment.

## References

[1] S. Banerjee, T. G. Griffin, and M. Pias. The interdomain connectivity of PlanetLab nodes. In *Proceedings of Passive & Active Measurement (PAM)*, pages 73–82, Antibes Juan-les-Pins, France, Apr. 2004.

[2] Y. Chu, S. G. Rao, and H. Zhang. A case for end system multicast. In *Proceedings of the ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, June 2000.

[3] PlanetLab Consortium. Planetlab acceptable use policy (AUP). https://www.planet-lab.org/php/aup/PlanetLab_AUP.pdf, Feb. 2004.

[4] N. Spring, D. Wetherall, and T. Anderson. Scriptroute: A public Internet measurement facility. In *Proceedings of the USENIX Symposium on Internet Technologies and Systems (USITS)*, pages 225–238, Seattle, WA, Mar. 2003.

[5] J. Strauss, D. Katabi, and F. Kaashoek. A measurement study of available bandwidth estimation tools. In *Proceedings of the ACM SIGCOMM Internet Measurement Conference (IMC)*, pages 39–44, Miami, FL, Oct. 2003.

[6] TCPDUMP.org Frequently Asked Questions. http://www.tcpdump.org/faq.html, July 2001.

[7] M. Zhang, C. Zhang, V. Pai, L. Peterson, and R. Wang. PlanetSeer: Internet path failure monitoring and characterization in wide-area services. In *Symposium on Operating Systems Design and Implementation (OSDI)*, San Francisco, CA, Dec. 2004.