

# PlanetFlow: Maintaining Accountability for Network Services

Mark Huang Princeton University 35 Olden St Princeton, NJ 08544 mlhuang@cs.princeton.edu	Andy Bavier Princeton University 35 Olden St Princeton, NJ 08544 acb@cs.princeton.edu	Larry Peterson Princeton University 35 Olden St Princeton, NJ 08544 llp@cs.princeton.edu
--	---	--

## ABSTRACT

PlanetFlow is a network auditing service that maintains comprehensive, permanent accountability for all traffic generated by PlanetLab services, in accordance with common Internet practice and the terms of the PlanetLab Acceptable Use Policy. PlanetFlow audits the usage of PlanetLab network resources in order to facilitate the resolution of complaints, limit liability, and minimize problematic behavior.

The current implementation of PlanetFlow consists of a low overhead flow classifier, an autonomously managed distributed database, and a publicly accessible Web interface. PlanetFlow currently processes up to 4 TB of generated traffic per day, and incurs negligible CPU and storage overhead.

## 1. INTRODUCTION

PlanetLab [2, 12] is a geographically distributed overlay platform designed to support the deployment and evaluation of planetary-scale network services. As of August 2005, it consists of over 580 machines at 275 sites in 30 countries, and supports over 450 research projects. PlanetLab continues to grow, adding approximately 1 site and 2 nodes per week.

PlanetLab is unique among research platforms in that all of its hosting institutions trust an external organization, PlanetLab Operations, to secure and maintain their nodes. In return, researchers at each institution are allowed to run their code on nodes hosted at other sites. As a result, accountability is always a primary issue when a problem arises. When the administrator of an outside network notices unwanted or unrequested traffic from a PlanetLab node and wants to file a complaint, he or she usually consults the WHOIS database to determine who is responsible for the node. Because each PlanetLab node is the property of the hosting site, most complaints are directed to a standard abuse alias at each site rather than to the support staff at PlanetLab Central.

It is often unclear to PlanetLab site administrators who to turn to when such situations arise. On one hand, the traffic was generated on their network by machines owned by researchers at their institution. On the other hand, the traffic could have been generated by any researcher affiliated

with PlanetLab. Without a system of accountability in place to determine who actually generated the offending traffic, PlanetLab would either have to bear legal responsibility for all traffic, or severely restrict the types of traffic that could be generated by its researchers.

We have developed a network auditing service called PlanetFlow that provides this system of accountability. The service is unique in the sheer scale of its responsibilities and requirements. PlanetLab is large: the platform supports more than 1,000 different users on 580 nodes. Any distributed auditing system of this scale must run autonomously. For a system of accountability to be effective, it must also be comprehensive. PlanetLab services generate a large amount of traffic, sometimes exceeding 4 TB per day. Simply managing this amount of information is a systems challenge in itself; running the service on PlanetLab requires that it do so efficiently using limited resources and privileges. Finally, for the system to help in resolving complaints, it must be fast as well as usable by the general public.

PlanetFlow provides comprehensive accountability for all traffic that leaves the PlanetLab network. It provides a public interface as easy to use as the WHOIS database for quickly determining accountability and enforces the “chain of responsibility” that exists between PlanetLab, its researchers, and its hosting sites. We believe that any wide-area network of shared resources that interacts with the public Internet—including other research networks, commercial ISPs, and virtual private server (VPS) networks—must maintain such a chain of responsibility to survive in the increasingly vigilant Internet. Because of its demonstrated usefulness in resolving complaints, PlanetFlow has become an integral part of PlanetLab’s architecture. Regardless of the underlying implementation of future versions of the platform, as long as PlanetLab provides its researchers with access to the Internet, it will need to hold them ultimately accountable for their actions.

## 2. ACCOUNTABILITY

Based on our experience of responding to hundreds of incidents since the inception of PlanetLab, we maintain that

a comprehensive system of accountability is both necessary and sufficient for resolving complaints, as well as for keeping PlanetLab secure. The Internet currently depends on the WHOIS database to provide accountability. We believe that any networked platform with a public presence must provide similar accountability, or assume liability for its users' behavior. Even when the users of a platform are entirely trusted, the possibility of compromise and abuse always exists. An effective auditing system isolates the liability for a particular incident to a single user, rather than the entire platform.

The incident response policy of PlanetLab is, thus, to put people who complain about PlanetLab traffic in direct contact with the researchers who generated it. PlanetLab researchers are responsible for resolving any complaints that arise from the operation or use of their services. This policy not only ensures that incidents are resolved as quickly and thoroughly as possible, but also shifts a significant portion of the support burden away from site administrators and PlanetLab Central. In the worst case, if the distributed virtual machine (VM) in which every PlanetLab service runs is compromised, the service can be temporarily or permanently disabled by PlanetLab Central without destroying its audit trail.

Of course, maintaining accountability is not the only solution to the problem of disruptive traffic. The most direct solution would be to obviate the need for maintaining an audit trail at all by actively preventing disruptive or abusive traffic from leaving PlanetLab. For example, PlanetLab could maintain a "reverse IDS" on every node that filters all outgoing traffic. However, the high false positive rate of IDSs—itsself the root cause of most traffic complaints—would also plague any attempt to implement a reverse IDS. Most existing IDSs expect standard Internet traffic and are easily triggered by research traffic, or any traffic outside the norm of the commercial Internet.

Another solution would be to close PlanetLab to the public Internet, and to maintain a whitelist of sites that have opted to receive traffic from PlanetLab. However, in addition to requiring continual maintenance, a whitelist would severely restrict the scope of any Internet mapping or measurement experiment. Content distribution networks (CDNs) such as CoDeeN [15] or CoralCDN [4] could not be supported on PlanetLab, since it is unlikely that a majority of content providers would "opt in".

As a research platform, PlanetLab encourages the development of services that push the envelope of current technology. Stifling research traffic with filters would only hinder its primary mission. Holding researchers accountable for their own traffic, wherever it is generated, is not just the only practical solution, it may be the only legal one as well. It is our belief, supported by the implications of the PlanetLab Acceptable Use Policy (AUP) and Consortium Agreement, that PlanetLab is essentially a "common carrier" such as an ISP. In other words, providing accountability is the extent

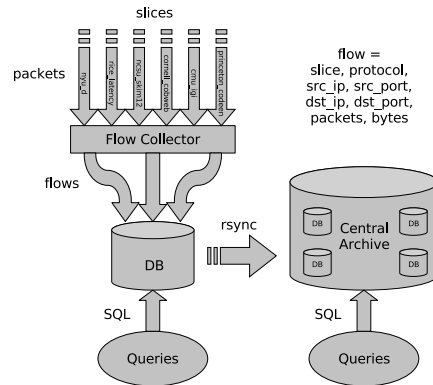
of PlanetLab's legal responsibility, which is fortunate given that it also happens to be the most practical method of minimizing problematic behavior.

### 3. PLANETFLOW

PlanetFlow is the name of the network auditing service maintained by PlanetLab Central. Its purpose is to maintain comprehensive, permanent accountability of PlanetLab network traffic as transparently and efficiently as possible, primarily to assist in resolving complaints. Because the data that PlanetFlow collects is also useful for research, PlanetFlow stores accounting data in a format amenable to mining, aggregation, and other queries more advanced than simply determining which VM (or slice) sent a particular packet.

The PlanetFlow service runs in a regular, unprivileged slice on every PlanetLab node and performs all privileged operations (such as reading packet headers generated by other slices, an operation not normally allowed) through the Proper service [9]. PlanetFlow consists of four primary components:

- a flow collector that classifies outgoing packets into IP flows;
- a database that stores the flows;
- web and administrative interfaces for querying the database; and
- a central server for storing, querying, and archiving flow data from all nodes.



**Figure 1: PlanetFlow components.**

The current implementation of PlanetFlow is optimized for low overhead, fast insertion time, reasonable query time, and archivability. Each component of the implementation was chosen carefully to meet these specific goals. Ease of implementation and flexibility in both maintenance and usage were also considered. From statistics gathered by CoMon [11], PlanetFlow typically adds less than 1% CPU overhead to a system; in the worst case, the overhead reaches 3%.

### 3.1 Flow Collector

PlanetFlow captures the IP headers of all outgoing packets, as well as their associated meta-data (e.g., originating slice and timestamp), with the `u_logd` program. To conserve space, packet headers are not recorded individually, but are instead classified into *flows* for which daily packet and byte counts are kept. The advantage of using `u_logd` instead of a program like `tcpdump` is that `u_logd` reads packets from the kernel in batches through a Netlink socket [8] in order to reduce context switching. A single `recvmsg()` call on a Netlink socket can transfer a batch of 64 or more packet headers. To transfer the same number of packet headers through a raw packet socket, such as one that `tcpdump` would open, `recvmsg()` would have to be called 64 times. Additionally, because Netlink data can be multicast, it is possible for multiple flow collectors to run simultaneously in different slices, without incurring significant additional overhead.

We extended `u_logd` to perform flow accounting for PlanetFlow. `u_logd` now classifies every outgoing IP packet into a unique flow keyed on the following attributes:

- Slice ID
- IP protocol number
- IP source address
- TCP or UDP source port, ICMP Echo ID, GRE key, or PPTP Call ID
- IP destination address
- TCP or UDP destination port

`u_logd` maintains a bounded cache of active flows, and stores a cumulative packet and byte count for each flow. Every five minutes, `u_logd` empties its cache into a database. Flows spanning multiple five minute time intervals are aggregated daily. Previous versions of PlanetFlow did not aggregate flows, which caused busy nodes to generate excessive amounts of usually redundant data. Periodic database updates and daily aggregation render PlanetFlow data adequate for auditing purposes, but unsuitable for certain types of real-time decision-making.

It is likely that the CPU overhead of PlanetFlow would increase with higher throughput rates, but because of flow aggregation, storage overhead would not. The storage requirement is directly related to the number of individual flows generated by a node. Mapping experiments that contact a large number of peers require the greatest amount of storage to audit, but higher link speeds would not necessarily increase the number of peers contacted in such experiments.

### 3.2 Flow Database

Previous versions of PlanetFlow used a flat text file to store flow information, which made it difficult to perform any but the most simple queries on the auditing data. Even these

queries could take several minutes to complete. To reduce query times and improve the usability of the interface, the current implementation now uses the open-source database MySQL to store flow information. MySQL is a fast, general-purpose database that, like all SQL servers, supports a standard and flexible query language. MySQL data files are particularly easy to archive and manage.

Two indices into the database optimize insertion operations and common-case queries. The first index, by flow, ensures that each flow in the database is unique and reduces insertion time. The second index, by IP destination address, speeds the most common queries when the destination is known. Using indices in a relational database to access only a few fields of data, does introduce significant storage overhead. Since the indices duplicate nearly all of the information in the database, storage requirements are increased by 150% or more. A busy node can generate 50 MB of compressed MySQL data and index files per day; an average node generates about 15 MB per day. Previous versions of PlanetFlow that stored flow data in a flat text file generated only 5-10 MB of compressed data per day, but could occasionally end up generating more because they did not aggregate flows.

Ease of management was another key reason that MySQL was chosen to implement the flow database. Because MySQL tables are just files and MySQL databases are just directories, flow data can be made immediately available on the archive server by simply copying the raw table files to it with a tool such as `rsync` [14]. Other database systems cannot merge tables or databases together without some amount of overhead. Using MySQL enables PlanetFlow data to be centrally aggregated and archived with no import overhead, improving the scalability of the system.

### 3.3 Query Interface

A Web interface to PlanetFlow runs on the default HTTP port of every PlanetLab node. The first instinct of many complainants is to type the IP address of an unknown peer into their Web browsers. The Web interface asks complainants to search the database for the offending traffic that they received, and to report their concerns directly to the maintainers of the service that generated it. In most cases, complaints can be quickly resolved through a simple explanation of the service, without the intervention of PlanetLab Support.

Most PlanetFlow queries now take seconds rather than minutes to return, and the Web interface has become highly usable as a result. If the number of complaints filed and ultimately resolved is any indication of success of the new implementation, then more than half of the approximately 100 traffic complaints received through the Web interface from May 2004 to July 2005 were filed since April 2005, when the current version of PlanetFlow was deployed. Complaints received via e-mail rather than through the Web interface are almost always resolved by PlanetLab Support by accessing the Web interface and putting complainants in direct contact

with the responsible researchers. Much of the burden of resolving traffic complaints has thus been shifted to the actual users of PlanetLab who generate traffic.

The Web interface also offers an aggregate summary of all flows sent by each active slice. This information may be used to estimate the bandwidth requirements of each slice, as well as to identify the most active users of a node.

A stand-alone administrative interface to the database called `pfgrep` is available to PlanetLab administrators to perform text-based searches similar to those that can be performed through the Web interface. In many cases, `pfgrep` is faster to use than the Web interface. A streaming sensor interface [13] is available for other slices to subscribe to at:

```
http://localhost/flows
```

The interface provides flow updates every five minutes in an easily parseable format.

### 3.4 Archive Server

All PlanetFlow data is permanently archived on a central server so that complaints about traffic can be resolved even if the node that generated it is unreachable. The archive server runs the same Web interface that each node does, and simply extends each query over all of its archived databases, another feature of MySQL that makes it highly suitable for PlanetFlow. The archive server contacts every node periodically to download new data.

The archive server currently stores five weeks' worth of data from about 500 active nodes. To keep the consolidated database size manageable, the archive server periodically compresses and offloads old databases onto DVD. Archive DVDs, and thus all PlanetFlow records, are kept forever and may be queried using `pfgrep` or other applications that can parse MySQL records. Currently, the archive server is outfitted with 500 GB of storage and keeps about 300 GB of data available for querying at any one time.

## 4. CHALLENGES

Choosing the necessary components of PlanetFlow was relatively straightforward. However, piecing them together into a scalable system was a careful, laborious process. Ensuring that the system ran stably and reliably despite uncertain load and inevitable corruption was an additional challenge. Finally, working PlanetFlow into the architecture of PlanetLab was a major goal from the outset, yet brought with it a host of additional requirements.

Seemingly minor implementation choices in PlanetFlow reflect the difficulty of the process. PlanetFlow's particular database requirements only revealed themselves after multiple failed attempts to implement PlanetFlow's database with PostgreSQL. Most notably, the overhead of exporting PostgreSQL tables to a central archive (almost an afterthought in the design of the system) became too great beyond a deployment of a few nodes. Capturing packets with a raw packet socket, the kind that `tcpdump` uses, proved too slow on

a real, fully loaded PlanetLab system even though it performed admirably in the lab. Aggregating flows at an hourly, rather than a daily, interval generated vast amounts of data that, while possibly useful for research, turned out to be redundant for resolving most security complaints. Slight changes in SQL syntax had enormous impact on the usability of the web interface; reducing the latency of the most common query was the primary focus of several schema redesigns.

As soon as the system was deployed to more than a few nodes for more than a few days, problems arose. When disparate components are pieced together, reliability always surfaces as the primary concern. Any complex system in wide deployment on PlanetLab will eventually experience problems such as filesystem corruption, database corruption, process termination, bugs, and race conditions caused by system load. PlanetFlow deals with these problems on several fronts. The flow collector attempts to limit the data loss caused by temporary database failures by re-opening its connection to the database every 5 minutes. A maintenance script runs every 10 minutes and restarts the flow collector, database, and web interface if they are stuck or dead. The script also securely polls PlanetLab Central daily for software updates at a randomized time to avoid congestion, attempts to repair corrupt tables, and compresses finished tables for later pickup by the archive server.

The final challenge to implementing PlanetLab was moving it into a slice in order to adhere to the PlanetLab philosophy of running management services in slices whenever possible. Because it runs in a slice, PlanetFlow is subject to certain runtime restrictions. PlanetFlow must use less than 5 GB of disk space per node, must not consume more than its fair share of memory<sup>1</sup>, and must be efficient in its use of CPU. In its current deployment, PlanetFlow does not approach these limits because of design considerations and because the rate and amount of traffic generated by PlanetLab services is currently manageable. If resource limits become a problem as PlanetLab continues to grow, guaranteed resource specifications may be requested for the PlanetFlow slice. The facilities through which guaranteed CPU and bandwidth may be reserved on nodes are currently under active development.

## 5. LIMITATIONS

PlanetFlow has proven very useful for resolving complaints caused by certain kinds of traffic, but it of course has its limitations. For instance, PlanetFlow cannot be used to completely address problems that arise from objectionable content. If someone determines that a PlanetLab node served copyrighted, illegal, or otherwise objectionable content to a specific set of hosts, PlanetFlow can be used to identify the

---

<sup>1</sup>Memory limits on PlanetLab nodes are lightly enforced with an out-of-memory killer called `p1_mom`. When memory is about to be exhausted, `p1_mom` kills the slice that is currently consuming the most amount of memory.

service which did so. However, it cannot be used to inspect or recover the contents of the traffic, since it logs only flow data. As previously mentioned, PlanetLab considers itself a common carrier and so does not inspect or censor content. When asked to, the responsible researchers must satisfy site administrators or other authorities that their service is not being used for illegal purposes.

Complaints about misuse of a service—for example, using a CDN to propagate spam or commit fraud—present a similar problem. PlanetFlow can be used to track such complaints back to the service, but additional information is usually required to identify the party that abused it. Many services keep their own application-specific logs for this purpose. We could make it easier for services to correlate their logs with PlanetFlow by enabling them to annotate their flows. For instance, a CDN could tag each connection with the IP address of the requester by making a special `setsockopt()` call. PlanetFlow could retrieve the tag from the meta-data of each packet and store it in the flow logs as additional, application-specific information. If privacy of this information is of importance, it could be made accessible only to PlanetLab administrators.

The primary goal of PlanetFlow’s web interface was to minimize the responsibility and involvement of site administrators in complaint resolution. The web interface has met with mixed success. On one hand, given the rise in the number of complaints filed through it, it is evident that people do use it. Furthermore, the interface is so easy to use, we have begun to receive almost frivolous complaints. On the other hand, many complaints regarding PlanetLab are still sent to abuse aliases at hosting sites, which consumes both their support resources and, often, patience. Given enough complaints, site administrators may consider hosting PlanetLab nodes to be too much trouble, unless their researchers continuously emphasize the value of PlanetLab to their institutions.

## 6. RELATED WORK

We believe that other distributed computing platforms, as they become more open and widely deployed, will eventually have to deal with complaints from the general public as well. PlanetLab is unique in both its deployment model and network access—its nodes are at autonomous hosting sites not under the direct control of the global platform management authority and all slices share a single IP address and port space on each node. Many of the difficulties encountered in implementing PlanetFlow on PlanetLab would not be relevant on platforms that are more tightly controlled or that can provide IP addresses for each of its projects’ points of presence. However, our experience building a large, distributed database system that efficiently deals with massive quantities of information and provides a usable interface to its data, should be relevant to the developers of any distributed logging service.

The designers of the WASCo distributed computing plat-

form [6] envision an infrastructure on which untrusted, semi-trusted, and trusted clients run their services on “surrogates”, much like researchers on PlanetLab run their services in slices. However, the legal relationship between clients and servers in the WASCo model is not as clear as the one between PlanetLab sites and the Consortium. Network security is handled on a case-by-case basis through a combination of techniques: tunneling, attack prevention, and logging. Tunneling is an interesting idea for limiting the possibility of launching anonymous attacks, but would not be appropriate on PlanetLab. The primary goal of PlanetLab is to provide planetary network access to services, some of which are *intended* to serve as public proxies. Attack prevention through detection and its disadvantages for PlanetLab were discussed in Section 2. Logging is only briefly mentioned, but we believe that its importance to any distributed platform cannot be overstated. A distributed logging infrastructure must scale with the size of the platform while remaining both comprehensive and usable.

The XenoServer [7] project is also building a public infrastructure for wide-area distributed computing. Like a PlanetLab node does, a XenoServer runs multiple virtual machines called Xen [1] VMs. All VMs connect to a Virtual Firewall-Router (VFR) which enables them to share a single physical network interface, similarly to how a PlanetLab node implements virtual networking. However, the VFR implements IP sharing using Network Address Translation (NAT) or proxy ARP, which makes the job of flow accounting somewhat easier since each VM is assigned its own IP address. If XenoServers are widely deployed and it becomes desirable to monitor all of them in aggregate, a port of PlanetFlow to XenoServer would be straightforward.

Grid architectures [3] limit network access to data exchange between participating Grid nodes. As such, the Grid currently cannot support network services that require open access to the Internet. The Grid dedicates physical machines to a single experiment for specific times, which would ease the task of accounting for network traffic: packet timestamps could be referenced against the schedule of experiments on the node. However, if the Grid leveraged virtual machines to multiplex resources among experiments and allowed them open access to the Internet, it would have to maintain a system of accountability similar to PlanetFlow.

Other large networks that peer with the Internet rely on the data generated by their private routers to perform flow analysis. A variety of packages exist to process and aggregate NetFlow data exported by Cisco routers. Like PlanetFlow, the OSU Flow-tools package [5] was designed to assist in resolving security incidents and complaints. Flows are stored in binary files sorted chronologically, and require the use of central “crunching servers” to generate reports suitable for analysis. MySQL has been used in the past to store and analyze NetFlow data [10], but PlanetFlow uses MySQL in a more distributed and efficient manner for the more specific purpose of mapping traffic to services, rather than for gen-

eral analysis.

## 7. CONCLUSIONS

PlanetLab's growth depends on building management tools for a large, distributed, multi-use infrastructure. PlanetFlow is a prime example. It has helped PlanetLab Support successfully handle hundreds of complaints about traffic originating from PlanetLab. We believe that other distributed service deployment platforms will generate traffic similar to PlanetLab's, and therefore can benefit from our experiences with PlanetFlow.

## 8. ACKNOWLEDGMENTS

We would like to acknowledge Paul Brett and Mic Bowman of Intel Corporation, who realized the need for network auditing on PlanetLab and developed the initial version of PlanetFlow.

## 9. REFERENCES

- [1] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield. Xen and the Art of Virtualization. In *Proc. 19th SOSP*, Lake George, NY, Oct. 2003.
- [2] A. Bavier, M. Bowman, D. Culler, B. Chun, S. Karlin, S. Muir, L. L. Peterson, T. Roscoe, T. Spalink, and M. H. Wawrzoniak. Operating System Support for Planetary-Scale Network Services. In *Proc. 1st NSDI*, San Francisco, CA, Mar. 2004.
- [3] I. Foster, C. Kesselman, and S. Tuecke. The Anatomy of the Grid. *Int'l. Journal Supercomputer Applications*, 15(3), 2001.
- [4] M. J. Freedman and D. Mazières. Sloppy Hashing and Self-Organizing Clusters. In *Proc. 2nd Int. Workshop on Peer-to-Peer Systems (IPTPS'03)*, Berkeley, CA, Feb. 2003.
- [5] M. Fullmer and S. Romig. The OSU Flow-tools Package and Cisco NetFlow Logs. In *Proc. 14th Systems Administration Conference (LISA '00)*, New Orleans, LA, Dec. 2000.
- [6] S. Goyal and J. Carter. Safely Harnessing Wide Area Surrogate Computing or How to Avoid Building the Perfect Platform for Network Attacks. In *Proc. 4th WORLDS*, San Francisco, CA, Dec. 2004.
- [7] S. Hand, T. Harris, E. Kotsovinos, and I. Pratt. Controlling the Xenoserver Open Platform. In *Proc. 6th International Conference on Open Architectures and Network Programming (OPENARCH)*, San Francisco, CA, Apr. 2003.
- [8] N. Horman. Understanding and programming with netlink sockets. <http://people.redhat.com/nhorman/papers/netlink.pdf>.
- [9] S. Muir, M. Fiuczynski, L. Peterson, J. Cappos, and J. Hartman. Proper: Privileged Operations in a Virtualised System Environment. In *Proc. USENIX '05*, Anaheim, CA, Apr. 2005.
- [10] J.-P. Navarro, B. Nickless, and L. Winkler. Combining Cisco NetFlow Exports with Relational Database Technology for Usage Statistics, Intrusion Detection, and Network Forensics. In *Proc. 14th Systems Administration Conference (LISA '00)*, New Orleans, LA, Dec. 2000.
- [11] V. Pai and K. Park. CoMon: A Monitoring Infrastructure for PlanetLab. <http://comon.cs.princeton.edu>.
- [12] L. Peterson, T. Anderson, D. Culler, and T. Roscoe. A Blueprint for Introducing Disruptive Technology into the Internet. In *Proc. HotNets-I*, Princeton, NJ, Oct 2002.
- [13] T. Roscoe, L. Peterson, S. Karlin, and M. Wawrzoniak. A Simple Common Sensor Interface for PlanetLab. *PlanetLab Design Note PDN 03-010*, May 2003.
- [14] A. Tridgell and P. Mackerras. The rsync algorithm. *Technical Report TR-CS-96-05*, 1997.
- [15] L. Wang, V. Pai, and L. Peterson. The Effectiveness of Request Redirection on CDN Robustness. In *Proceedings of the Fifth Symposium on Operating Systems Design and Implementation*, Boston, MA USA, December 2002.