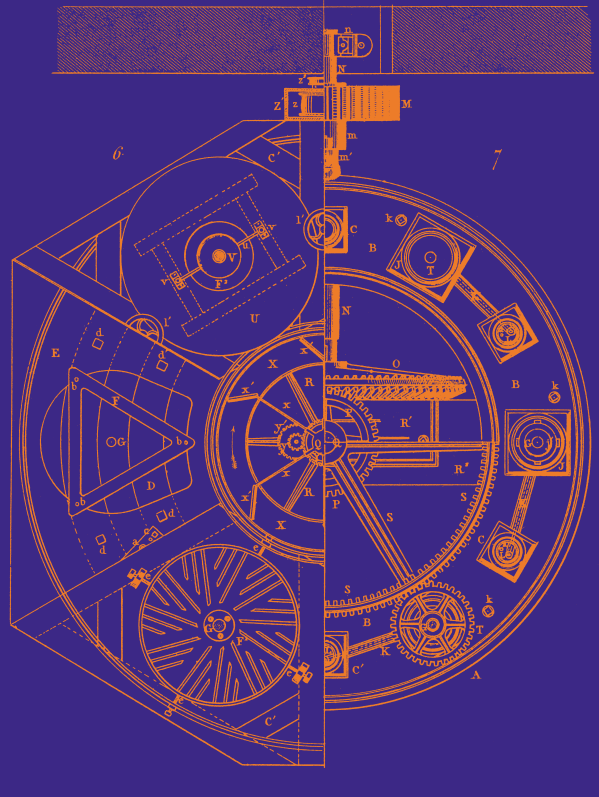
AUGUST 2007 VOLUME 32 NUMBER 4

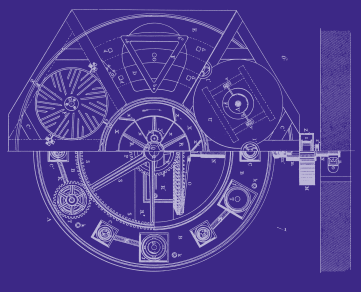
;login:

THE USENIX MAGAZINE



USENIX
The Advanced Computing Systems Association

USENIX Upcoming Events



2007 Linux Kernel Developers Summit

SEPTEMBER 4–6, 2007, Cambridge, U.K.
http://www.usenix.org/kernel07

Internet Measurement Conference 2007 (IMC 2007)

Sponsored by ACM SIGCOMM in cooperation with USENIX

OCTOBER 24–26, 2007, San Diego, CA, USA
http://www.imconf.net/imc-2007/

21st Large Installation System Administration Conference (LISA '07)

Sponsored by USENIX and SAGE

NOVEMBER 11–16, 2007, Dallas, TX, USA
http://www.usenix.org/lisa07

ACM/IFIP/USENIX 8th International Middleware Conference (Middleware 2007)

NOVEMBER 26–30, 2007, Newport Beach, CA, USA
http://middleware2007.ics.uci.edu/

6th USENIX Conference on File and Storage Technologies (FAST '08)

Sponsored by USENIX in cooperation with ACM SIGOPS, IEEE Mass Storage Systems Technical Committee (MSSTC), and IEEE TCOS

FEBRUARY 26–29, 2008, San Jose, CA, USA
http://www.usenix.org/fast08
Paper submissions due: September 12, 2007

2008 ACM International Conference on Virtual Execution Environments (VEE '08)

Sponsored by ACM SIGPLAN in cooperation with USENIX

MARCH 5–7, 2008, Seattle, WA, USA
http://vee08.cs.tcd.ie
Abstracts due: August 27, 2007

5th USENIX Symposium on Networked Systems Design and Implementation (NSDI '08)

Sponsored by USENIX in cooperation with ACM SIGCOMM and ACM SIGOPS

APRIL 16–18, 2008, San Francisco, CA, USA
http://www.usenix.org/nsdi08
Paper titles and abstracts due: October 2, 2007

2008 USENIX Annual Technical Conference (USENIX '08)

JUNE 22–27, 2008, BOSTON, MA, USA
http://www.usenix.org/usenix08
Paper submissions due: January 7, 2008

For a complete list of all USENIX & USENIX co-sponsored events, see http://www.usenix.org/events.

contents

;login:

VOL. 32, #4, AUGUST 2007

EDITOR
Rik Farrow
rik@usenix.org

MANAGING EDITOR
Jane-Ellen Long
jel@usenix.org

COPY EDITOR
David Couzens
proofshop@usenix.org

PRODUCTION
Lisa Camp de Avalos
Casey Henderson

TYPESETTER
Star Type
startype@comcast.net

USENIX ASSOCIATION
2560 Ninth Street,
Suite 215, Berkeley,
California 94710
Phone: (510) 528-8649
FAX: (510) 548-5738

http://www.usenix.org
http://www.sage.org

;login: is the official magazine of the USENIX Association.

;login: (ISSN 1044-6397) is published bi-monthly by the USENIX Association, 2560 Ninth Street, Suite 215, Berkeley, CA 94710.

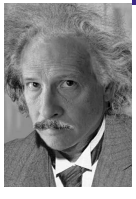
$85 of each member's annual dues is for an annual subscription to *;login:*. Subscriptions for nonmembers are $120 per year.

Periodicals postage paid at Berkeley, CA, and additional offices.

POSTMASTER: Send address changes to *;login:*, USENIX Association, 2560 Ninth Street, Suite 215, Berkeley, CA 94710.

RIK FARROW

musings



rik@usenix.org

I AM NO EINSTEIN, BUT I READ RE- cently that when researchers showed people pictures of smart-looking people, they did better on the multiple-choice test that followed. People shown pictures of bad role models had scores a full 33% lower. In that vein, and with a serious-looking guy in an old suit staring at me, I plan to carry out a thought experiment about the state of computer security.

Like Einstein himself, I will stand on the shoulders of those who have gone before me. Unlike Einstein, most computer security researchers are still alive today, because the field is still young, even if some of us no longer are ungrayed.

First off, I can say with great certainty that the state of computer security is poor. You can see for yourself that it has actually gotten better over time, but even so, you can't browse the Web in safety today. The Provos et al. paper, one of my favorite presentations during the HotBots workshop (see the HotBots summaries later in this issue), points out what Google has started doing as a service to its visitors. Instead of presenting links to Web sites that Google considers harmful, Google instead presents a "Malware Warning" page. If you still want to visit the site you have been warned about, you must cut and paste the link. Provos mentioned that 30–40% of people were clicking through the link on the warning page until Google engineers converted it into plain text. That points to another reason why the Internet will never be safe, but improving good sense in people is not something that can be done with software—at least, not yet.

In six months of searching Google's immense store of cached Web pages, Provos et al. found 450,000 examples of pages that resulted in the compromise of a Windows system when that page was visited using IE. The Google team runs IE and Windows within a VM environment, and it monitors that environment for changes to the registry, new files, and, best yet, new processes. The team also captures downloaded content and scans it for known examples of malware. Based on what its system discovers, a score gets assigned to the page in question, and high-scoring pages get labeled as harmful.

Google's approach appeared a conservative one to me, but when you consider the potential harm of labeling someone's site as harmful, that is a safe approach. I actually searched for examples of harmful

pages and discovered, on my very first hit, a nice example of a <script> tag that was being used to download and execute a 67-line Javascript program that checks for a cookie and creates it if it doesn't exist. The script finally creates an <iframe> tag that requests the next stage of drive-by download.

Niels has written a tool that you can use to scan your own Web sites: SpyBye [1]. I suggest that you at least monitor your own Web sites for surprise changes, and for fast-changing sites, SpyBye makes the task much easier to do.

Labeling URLs as harmful is a great symbol for the state of security today. The security industry makes billions of dollars selling tools that search for viruses and attacks, tools that will always be trailing behind the latest attacks. Attackers constantly morph their exploits and malware, making them just different enough to avoid detection. It is much simpler for attackers to generate multiple versions of malware (just change the encryption routine by a few instructions, use a different key, or change the code layout a tad) than it is for A-V or IPS vendors to keep up with this endless stream of new malware.

In my thought experiment, I have seen some nice advances. Techniques such as using nonexecutable stacks (what a great idea!) and randomizing code layout have made buffer-overflow attacks mostly a thing of the past, although not entirely, of course, because we still get patches for various buffer overflows. But these simple changes, even though one of them first required cooperation from some giant CPU vendors, have already make a world of difference. Still, our systems are not secure.

What about the humble Web browser? Our indispensable companion can hardly be considered secure, and there are good, well, bad, reasons for this. First, the Web browser represents a "flat space," as a Microsoft researcher put it during HotOS. What he meant should be obvious with a little thought. Your Web browser runs in a single protection domain, that is, one process. Any event that occurs within that protection domain affects the entire browser. And although browser vendors do attempt to limit the effects of scripts to the page downloaded by a particular site, there are ways around that (see the comments about iframe and script tags made earlier).

Dangerous at Any Speed

And browser technology lends itself to helping attackers. What other software do you use to fetch and execute code, sight unseen, in the context of your own user account? Add to the feature of executing remote code on others' behalf some more exciting features, such as the ability to install plug-ins that run as part of the browser. IE's Browser Helper Objects come to mind, as these are like DLLs and can hook into any event within the browser, for example, keystrokes. I can imagine software that waits until you visit any of a long list of financial institutions, then captures your keystrokes and packages them up neatly in a POST to some offshore hacker haven. But wait! That software already exists, and, according to the Symantex Internet threat summary [2], that offshore haven is the United States!

UK banks have attempted to get around the issue of the theft of authentication details by handing out two factor devices. Such devices are impossible to defeat, as they present pseudo-random passwords for each authentication event. Or are they? I can visualize some browser plug-in that waits until I visit a financial organization, then proxies the communication to an evil server as I authenticate, and finally displays an error to me once authentica-

tion is complete. The attacker now has an authenticated connection, via the proxy, to my account, in spite of the use of the authentication device. Bruce Schneier predicted this proxy attack during an RSA conference years ago [3], and malware that does this already exists. Once again, we are defeated by a flat browser space.

I found myself very intrigued by a presentation by a security researcher, Joanna Rutkowska [4], who announced the creation of the Blue Pill, a virtual machine shim that gets installed under Vista at boot time. Now everything above the VM can be controlled, as Vista is just a "guest" operating system! Other security researchers weren't so impressed, because there are too many easier ways to take control of systems. From the browser, down through many layers of GUI libraries, then C libraries, and finally into the kernel itself, the possibilities for exploitation are, well, almost endless. Using the Blue Pill, while interesting, is just overkill on today's systems.

Thought Experiment

Okay, now it's time to do the Einstein thing. Since all I have to do is think, not actually write code or implement hardware, I have total freedom. How can I imagine a way around the security issues we face today?

I think I shall start at the bottom-most level, with a secure boot process. Never mind that someone else has already invented this [5, 6]: my boot process will only load software that has been digitally signed and will use a secure hardware mechanism for verifying this signature. With a secure boot, I can ensure that I won't be swallowing a Blue Pill before I get started!

Next, I conjure up a very small kernel, a microkernel. Large applications are impossible to audit, much less prove that they are secure [7, 8]. By starting small, I will have at least a strong chance that my innermost ring of software can be trusted.

More hardware appears to be an excellent idea, so I will include the concept of hardware-based and -enforced privilege level and memory management. Ignore for the moment that this was invented in the 1960s, because it works poorly when the kernel and other trusted computing base has grown as bloated as it has. The hardware adds to my provably secure kernel because it prevents the kernel code from being modified. The kernel can still modify itself, which I foretell can be a danger, but by sticking to a small kernel I can reduce that threat to a minimum. No one has ever produced perfect software, not even Wietse Venema [9], so I want the added assurance that hardware protection features provide me with.

I seem to recall that early experiences with microkernels were disappointing because they performed poorly when compared to traditional, monolithic kernels. Laying out a traditional kernel design on my garage floor (since no other room is nearly big enough), I notice that, quite unsurprisingly, today's kernel neatly matches up to today's CPU design! Well, that's not much of a surprise, as OS programmers have had 45 years to work with a CPU design that involves using software interrupts for context changes. To say that microkernels perform poorly when compared to monolithic ones is like expecting a championship snowboarder to do well when using a tennis racket as a board. Of course microkernels perform poorly: They are running on hardware that was designed with monolithic kernels in mind!

Now look at that early hardware design. What were they thinking about back in 1960? Those pioneers were wondering how best to go about sharing mainframe computers, because they were extraordinarily expensive to build

and maintain. So the early OS designers built time-sharing systems so that many people could use a computer at the same time [10]. This showed real genius, but it makes a poor match for the computing situation we have today. After all, how many people do you share your desktop with when you are using it? There is yourself, along with the botherder who installed bot software and trojans on it, of course.

Today's computers, be they desktops or servers, are essentially single-user systems. There are multiple user accounts, and these are often used to good purpose on server systems. Even desktops have administrative accounts, but this separation of power is usually perverted by allowing the desktop's single regular user administrative privileges. Thus, any exploit against the user has the power of the administrator, proving to be no defense at all. And forget about Vista's new feature that requires entering your password to install any software. Once you have experienced this, well, from what the Microsoft Research guys tell me, you will disable User Account Control, as they told me they quickly did.

And servers? How many users does your Apache Web server run as? How about your Oracle or MySQL server? Just one? I thought so, so we have another case of a single-user system. Even though the server application acts on behalf of perhaps thousands of users, the server itself runs in the context of a single user: just one protection domain. Once again, we are dealing with the aftereffects of time-sharing systems, decades after we switched to new models of computing.

Multicore

You will soon own, if you don't already, multicore CPUs. In fact, I am guessing that if you don't already own these systems, you either manage or work with them routinely. Intel has already demonstrated an 80-core CPU [11], and real 160-core systems are not that far off. Sun has had an 8-core, 32-threaded chip, the T1, for over a year, and Intel and AMD are not far behind.

Multicore processors require new programming techniques to take full advantage of the multiple threads of processing available. We already know how hard it is for human beings to write, debug, and maintain concurrent code (witness the number of years it has taken to get rid of the giant locks in various UNIX-like kernels). And our server and desktop applications are nowhere near to being parallel in design. We will soon have multicore chips that can only be taken advantage of, in a limited way, by the supporting software, the operating system.

In my thought experiment, we have new tools for writing, debugging, and supporting software that deal naturally with parallelism, taking full advantage of having many cores and multiple threads. And I imagine that this may even require some hardware support—for example, for transactional memory (see the summary of David Wood's HotOS keynote on p. 90 of this issue). If parallel programming will require some hardware changes, perhaps it is time to revisit the design decisions that were made in the heat of the Cold War, back in the 1960s.

Sweeping clean my mental workbench, I begin to fashion processor support for fast IPC (Interprocessor Communications). The past CPU designs do this poorly, even though most large applications benefit from sharing memory. Sharing memory is very expensive because it requires that multiple levels of cache, from L1 right up to L3 if it's there, must be kept coherent and consistent. My mental design allows the mapping of small blocks of memory between threads, each running in its own protection domain, without having

to invalidate TLB (Translation Lookaside Buffers, used to speed up mapping of virtual memory into physical memory). The old models have got to go if we are ever to support secure microkernels and parallel programming.

I will also pin the microkernel to a single core, with multiple threads but also with L1 and L2 caches large enough that the microkernel runs at or near CPU speeds, not at the much slower speed of RAM.

Finally, I see on my mental workbench my new system: It is totally silent, of low power, and uses 128 threads to swiftly and securely serve my every need. I can browse the Web securely, with content from each site—even images— safely isolated by hardware, in their own protection domains. IPC passes rendered images from one thread to another thread that displays the bitmaps via its device driver, and scripts run only within the context of the site from which they were sourced. This total isolation is made possible through new designs in both processor hardware and software.

Einstein was working as a patent clerk when he wrote his Nobel Prize–winning papers. It was many years before anyone even began to recognize the advances mentioned in his papers written in 1905, and 16 years before he received the Nobel Prize [12].

I surely am not Einstein, even if I have (some) of his hair. But I just as certainly know that we cannot wait 15 years for a change in our computer architecture. We are already way overdue, having lived with time-sharing computer architecture, operating system, and programming paradigms way past their prime. The advent of multicore computing provides us with a rare, and much needed, opportunity to reinvent computing from the ground up.

Lineup

After that rant, you are probably expecting an issue heavy in security and OS design topics. Instead, I have provided some lighter reading for your summer vacation. (The heavier stuff will come later this year, I promise.)

We start off with an article by the authors of the best student paper at NSDI. BitTyrant works like any other BitTorrent client, with one important exception: It rewards reciprocity in a more intelligent manner. Read this article, and learn more about how BitTorrent works and how BitTyrant can improve your download experience.

Alva Couch expounds upon one of the most divisive issues of our time: certification. Rather than traveling down well-worn paths, Alva presents rational ideas, based on real life, as the basis for certifying sysadmins in a practical manner.

Octave Orgeron takes you for a quick tour of Solaris Logical Domains. Logical Domains are a new virtualization technology designed to improve upon existing VM schemes. If anything, Logical Domains appears to me to be a perfect extension to Zones and ZFS. Orgeron plans on writing other articles with tutorials about using it.

Emin Gün Sirer and I next discuss the use and abuse of power laws. Riding on the success of Gordan Moore, we take you on a trip where we extrapolate the future of computing.

Massimo Bernaschi and his co-authors explain their approach for migrating SSL/TLS sockets. Their approach will be appreciated by anyone who wants to improve the reliability of SSL connections with failover servers.

In the columns, David Blank-Edelman starts off with more on unusual but useful Perl modules, focusing this time on tricks with Perl and PDF. Dave

Josephsen joins *;login:* with his introductory column in this issue. Dave's focus is on monitoring and Nagios, but he intends to cover a much broader area than a single Open Source project. I've enjoyed reading Dave's writing in the past, and I think that you, too, will appreciate his insights.

Heison Chak tells us more about branches pulled from Asterisk. Some have not been satisfied with Asterisk and, not being able to pull the code base in their own directions, have branched off. Chak reveals all.

Robert Haskins has taken a sabbatical from "ISP Admin," after many years of writing this column. Many thanks are owed to Bob for writing for *;login:* and helping us see the world of networking from the ISP perspective.

We have, of course, a raft of book reviews, preceded by Robert Ferrell's amusing musings, "/dev/null." Robert bemoans the lack of hat-wearing Texans (oh no!), then lays into the failures of certification. To top off the list, Nick Stoughton tells us where the C standard has been and where the Committee plans to take it next. Nick tells us not to worry, that C will stay the same, yet get better. (Remember what I just wrote about multicore CPUs, and you should appreciate why C must change.)

I'd like to leave you with just two notions from my thought experiment, concepts that are as real as sand at the beach. First, we really stopped using time-sharing systems sometime in the 1980s, and it is time our programming and OS models caught up. Second, microkernels have gotten a bad rap because we ran them on the wrong hardware. Just as you can't jam a DVD into an 8-track player and expect good things to happen, the CPU designs we use must evolve too.

REFERENCES

[1] SpyBye: http://www.monkey.org/~provos/spybye/.

[2] Symantex Internet threat summary: http://www.symantec.com/enterprise/index.jsp.

[3] Bruce Schneier, http://www.schneier.com/blog/archives/2005/03/the_failure_of.html.

[4] Joanna Rutkowska, http://invisiblethings.org/.

[5] Trusted Computing Platform Alliance: http://domino.research.ibm.com/comm/research_projects.nsf/pages/gsal.TCG.html.

[6] Nexus: http://www.cs.cornell.edu/people/egs/nexus/.

[7] Kevin Elphinstone et al., "Towards a Practical, Verified Kernel," http://www.usenix.org/events/hotos07/tech/.

[8] Attacking the Core: Kernel Exploitation Notes: http://phrack.org/issues.html?issue=64&id=6#article.

[9] Wietse Venema, http://www.porcupine.org/wietse/.

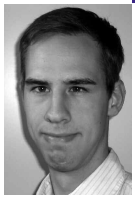
[10] MULTICS: http://www.multicians.org.

[11] 80-core demonstration: http://www.engadget.com/2007/02/11/intel-demonstrates-80-core-processor/.

[12] Albert Einstein's biography: http://nobelprize.org/nobel_prizes/physics/laureates/1921/einstein-bio.html.

MICHAEL PIATEK, TOMAS ISDAL, TOM ANDERSON, ARVIND KRISHNAMURTHY, AND ARUN VENKATARAMANI

building BitTyrant, a (more) strategic BitTorrent client



Michael Piatek is a graduate student at the University of Washington. After spending his undergraduate years working on differential geometry, his research interests now include incentive design in distributed systems, network measurements, and large-scale systems building.

piatek@cs.washington.edu

Tomas Isdal graduated with a MSc in Computer Science and Engineering from the Royal Institute of Technology, Stockholm, Sweden, and is currently a graduate student in the Department of Computer Science and Engineering at the University of Washington. His interests include peer-to-peer and distributed systems, Internet measurements, and network security.

isdal@cs.washington.edu

Tom Anderson is a Professor in the Department of Computer Science and Engineering at the University of Washington. He is an ACM Fellow and a winner of the ACM SIGOPS Mark Weiser Award, but he is perhaps best known as the author of the Nachos operating system.

tom@cs.washington.edu

Arvind Krishnamurthy is an Assistant Research Professor at the University of Washington. His research interests are primarily at the boundary between the theory and practice of distributed systems. He has worked on automated mechanisms for managing overlay networks and distributed hash tables, network measurements, parallel computing, techniques to make low-latency RAID devices, and distributed storage systems that integrate the numerous ad hoc devices around the home.

arvind@cs.washington.edu

Arun Venkataramani has been an Assistant Professor at the University of Massachusetts Amherst since 2005, after receiving his Ph.D. from the University of Texas at Austin by way of the University of Washington. His research interests are in the practice and theory of networking and distributed systems.

arun@cs.umass.edu

PEER-TO-PEER SYSTEMS OFTEN APPEAL to scalability as a motivating feature. As more users request data, more users contribute resources. Scaling a service by relying on user contributions—the P2P approach—depends on providing incentives for users to make those contributions. Recently, the popular BitTorrent file distribution tool has emerged as the canonical example of an incentive-aware P2P design. Although BitTorrent has been in widespread use for years and has been studied extensively, we find that its incentive strategy is not foolproof. This article describes BitTyrant, a new, strategic BitTorrent client. For users interested in faster downloads, BitTyrant provides a median 70% performance improvement on live Internet swarms. However, BitTyrant also demonstrates that selfish users can improve performance even while reducing upload contribution, circumventing intended incentives.

Bandwidth demands on Internet data providers are increasing. Google Video, Amazon's Unbox, and Apple's iTunes music store are just a few well-known examples of bandwidth-hungry services now delivering entertainment content. In addition, application vendors regularly distribute large patches to thousands of customers. As demand for these services and applications grows, bandwidth costs increase in turn.

Peer-to-peer (P2P) systems offer a promising approach to deferring these costs while increasing scalability. They avoid the bottlenecks associated with typical one-sided data distribution, where servers send data to clients. P2P designs exploit the fact that once a client begins receiving data, it can function as an additional server by redistributing that data, shifting load from the server to clients.

Shifting load from servers to clients means relying on clients to contribute capacity. Early P2P systems builders quickly realized that when given a choice, most users wouldn't contribute their resources. Instead, they would "free-ride," a modern-day tragedy of the commons where users consume system resources without providing any in return [1].

To combat the free-riding problem, subsequent P2P designs included explicit contribution *incen-*

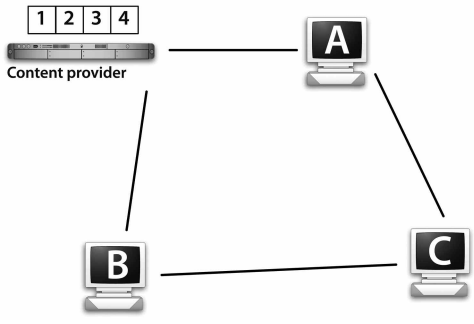


FIGURE 1A: A SAMPLE SWARM TOPOLOGY

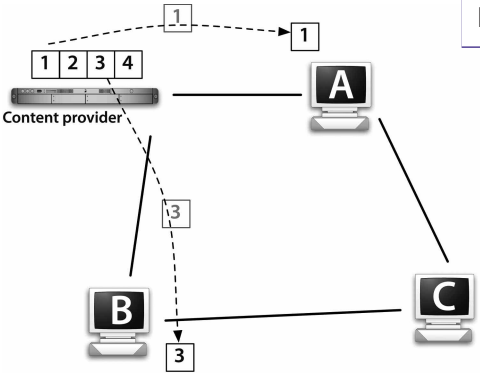


FIGURE 1B: THE PROVIDER SENDS RANDOM BLOCKS TO DIRECTLY CONNECTED PEERS A AND B.

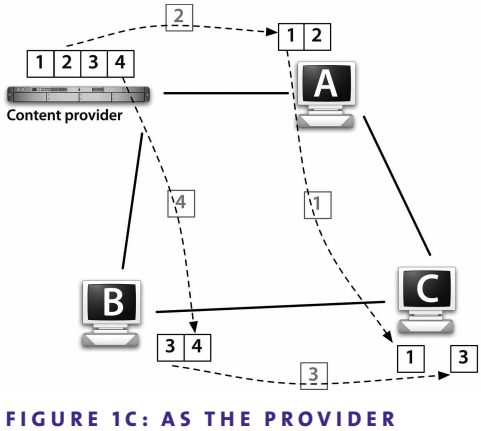


FIGURE 1C: AS THE PROVIDER SENDS MORE NEW BLOCKS, A AND B REDISTRIBUTE PREVIOUSLY RECEIVED DATA TO C, REDUCING LOAD ON THE PROVIDER.

tives. In these systems, increasing contribution improves performance and, as a result, free-riders receive poor service. Today, BitTorrent has become one of the most popular incentive-aware P2P systems and is used daily by millions of people worldwide. BitTorrent's policy is "tit-for-tat"; each individual user gives data to other peers that reciprocate, that is, send data in return.

Intuitively, BitTorrent's tit-for-tat policy makes sense. Each client acts in its local self-interest by rewarding peers that provide it with data. Further, this strategy can be carried out without the need for centralized enforcement, maintaining the decentralized nature of P2P networks. Our question is this: *Did BitTorrent get it right?*

In spite of its success, we find that BitTorrent's incentive strategy can be cheated. We've built a new BitTorrent client, *BitTyrant*, that circumvents intended incentives. Instead of improving performance by increasing contribution, BitTyrant improves performance by operating strategically, enabling its users to improve performance even while reducing their contribution.

How BitTorrent Works

Before describing BitTyrant, we'll first explore how BitTorrent works today. There are three pieces of relevant context: (1) how peers are organized, (2) how data is distributed, and (3) how peers prioritize requests. We examine these in the context of an example file distribution.

A BitTorrent user obtains a file by first joining a *swarm*, a set of peers already downloading the file. To join a swarm, clients contact a centralized coordinator, which returns a random subset of the existing peers to the new client. These peers form the new client's local neighborhood—the set of directly connected peers from which the client will send and receive file data. A sample swarm topology is shown in Figure 1a. In this example, three peers (A, B, and C) have just joined the swarm.

BitTorrent distributes a file by splitting it up into several fixed-size blocks. In Figure 1a, the content provider has a complete copy of the file, which it has split into four blocks. In practice, BitTorrent blocks are small, and a large file might be split into thousands of blocks, but we limit ourselves to four for simplicity. Content providers distribute data by sending randomly chosen blocks to directly connected peers. In Figure 1b, the content provider is directly connected to A and B, which receive blocks 1 and 3, respectively. After receiving these blocks, A and B can begin redistributing data to their directly connected peers—in this case, C. Figure 1c shows the next round in this process: the content provider continues to send new blocks to A and B while they concurrently redistribute previously received blocks 1 and 3 to C. This process continues until peers obtain all blocks and have a complete copy of the file.

In our example, each client receives only a few blocks from a few peers. In practice, clients are connected to dozens of peers that compete for scarce upload bandwidth. BitTorrent clients are faced with a decision: Given many competing requests and limited resources, which should be serviced? BitTorrent adopts a tit-for-tat strategy. First, a client ranks peers according to the rate at which they have been sending data in the recent past. Then the client provides the top k of these peers with an equal split of its upload capacity. The value of k is fixed and determined by a peer's upload capacity. In our example, suppose peer C has total upload capacity 20 with $k = 1$ and receives data from A and B at rates 15 and 10, respectively. In this case, C would reciprocate with A, providing it with data at rate 20.

Decisions about which peers receive data are reevaluated every 10 seconds, a tit-for-tat round. Each round, clients send data to a few random peers that have not "earned" it in a tit-for-tat sense, to explore their local neighbors for better pairings and to bootstrap new users into the tit-for-tat process. Once a new peer has received a few blocks, it can begin trading to induce reciprocation.

Building BitTyrant

Although BitTorrent's tit-for-tat strategy rewards contribution, the reward is not exact. A client that contributes quickly *tends* to receive quickly—with some variability. For instance, a DSL user might be directly connected to a peer behind a university's high-capacity link. Although the DSL user might send at rate 10, the university peer might reciprocate at rate 100. Mismatches like this arise because peers make decisions with limited information.

Ideally, tit-for-tat would match high-capacity peers with mostly high-capacity peers and low-capacity peers with mostly low-capacity peers, avoiding unfair mismatches. In practice, achieving this grouping is slow. Recall that BitTorrent clients search for better matches randomly. Because the bulk of BitTorrent users are low-capacity, high-capacity peers encounter one another comparatively infrequently, through random exploration. Furthermore, BitTorrent swarms are highly dynamic, with users arriving and departing rapidly. Even if a stable pairing arises, it may be short-lived.

Capacity mismatches among peers suggest a potential strategy for improving performance. If a client could quickly identify high-capacity peers, it might induce reciprocation even with small contributions, relying on the slow convergence of tit-for-tat to inhibit competition. Predicting the effectiveness of this strategy depends on how often mismatched pairings occur and on the extent of the imbalance in mismatches.

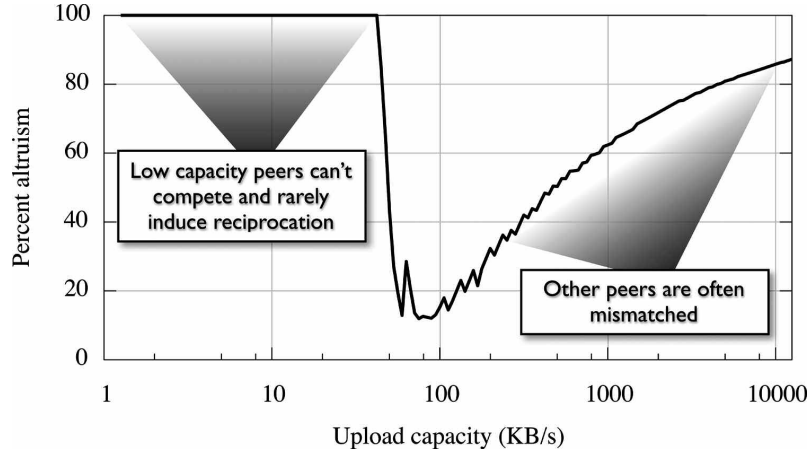


FIGURE 2: PERCENTAGE OF ALTRUISTIC UPLOAD CONTRIBUTION AS A FUNCTION OF CAPACITY

We examine the impact of mismatches through the lens of *altruism*, the contributions of a user that, if withdrawn, would not impact performance. For instance, if a high-capacity user sends data to a peer at rate 100 when rate 10 would suffice to induce reciprocation, we say that 90% of that contribution is altruistic. The altruistic proportion of a random BitTorrent connection can be computed statistically, and Figure 2 shows percent altruism across all connections as a function of upload capacity. Altruism is highest at the ends of the capacity spectrum. The lowest-capacity peers rarely induce reciproca-

tion, because their contribution rates are not competitive. These peers rely on the random exploration of others for all the data they receive. As a result, virtually all of their contributions could be withdrawn. At the high end, mismatches are frequent, with altruistic contributions increasing with capacity. For those peers in the middle, altruism varies, but it never reaches zero.

All peers make altruistic contributions, suggesting that a strategic client could improve performance by identifying those contributions and reallocating them intelligently. This is the approach taken by BitTyrant, a more strategic BitTorrent client that exploits altruism. BitTyrant deviates from the behavior of most existing BitTorrent clients in two ways. First, rather than exploring peer pairings randomly, BitTyrant infers which peers have high capacity and preferentially explores them. Second, BitTyrant does not split its upload capacity equally; instead, it dynamically varies the send rate to each peer to maximize return on investment.

To infer which peers have high upload capacity, BitTyrant relies on control traffic broadcast by all BitTorrent peers about which data blocks they have received so far. Measuring the rate of these block announcements provides an estimate of the download rate of a peer. Recall that tit-for-tat, although inexact, tends to reward higher contribution with a higher download rate, allowing a BitTyrant peer to infer that a peer downloading quickly may also upload quickly.

Although the download rate heuristic provides a good first approximation, it might not be accurate, and network conditions change over time. Further, simply picking *which* peers will receive data is only half the problem; BitTyrant also needs to choose *how quickly* to send to each of those peers. BitTyrant copes with both of these issues by dynamically selecting peers and rates at the same time. For each peer, BitTyrant maintains a benefit/cost ratio, where cost is the upload rate required to induce reciprocation, and benefit is the download rate resulting from that reciprocation.

BitTyrant sorts peers by their benefit/cost ratios, sending data to each in descending order until upload capacity is exhausted. After every tit-for-tat round, BitTyrant updates its estimates of cost and benefit according to peer behavior. A peer that reciprocates has its download rate (benefit) updated with the directly observed download rate. If after receiving data a peer does not reciprocate, BitTyrant increases the cost estimate (required upload rate). Finally, BitTyrant interprets continued reciprocation over many tit-for-tat rounds as a signal that its cost estimate is too generous and scales down the upload rate provided to the continually reciprocating peer.

These rules reflect the strategic nature of BitTyrant's approach. Motivated by the heavy skew of bandwidth capacity, BitTyrant actively seeks out the minority of high-capacity peers that provide the bulk of download throughput. To ensure continued reciprocation with these peers and maximize overall return on bandwidth investment, BitTyrant dynamically adjusts which peers receive data and sends rates to those peers.

Performance in the Wild

To evaluate BitTyrant, we measure its download performance in live Internet swarms. To provide an apples-to-apples comparison, we compare BitTyrant to Azureus, currently the most popular BitTorrent client implementation and the distribution on which BitTyrant is based. We crawled popular BitTorrent swarm aggregation Web sites, obtaining a set of 114 swarms, which we then downloaded concurrently with both BitTyrant and Azureus from two machines at the University of Washington. Both clients were given an

upload capacity limit of 128 kilobytes per second, to avoid interference from network cross-talk and to provide an evaluation of BitTyrant's effectiveness for modestly provisioned hosts.

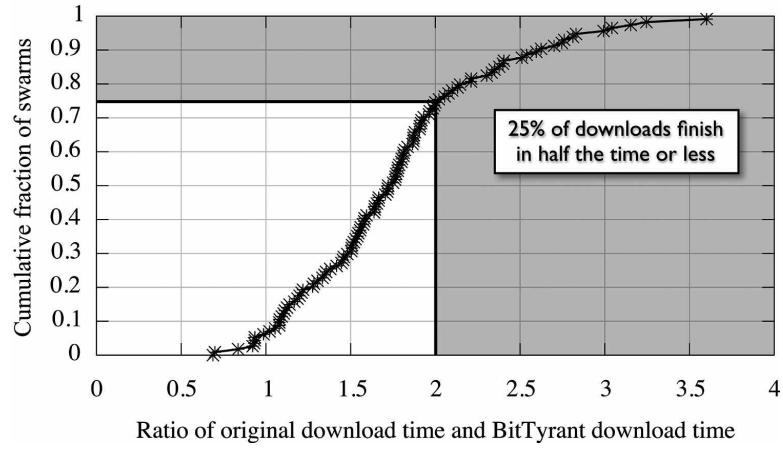


FIGURE 3: DOWNLOAD PERFORMANCE FOR 114 REAL-WORLD SWARMS, SHOWING THE RATIOS BETWEEN DOWNLOAD TIMES FOR AN EXISTING AZUREUS CLIENT AND BITTYRANT

For each swarm, we compute the ratio of Azureus's download time and BitTyrant's download time. For example, if Azureus downloads a file in 30 minutes and BitTyrant completes in 15, this ratio is 30/15 = 2. These completion time ratios are summarized in Figure 3, which gives the fraction of swarms (*y* axis) with a ratio of a particular value (*x* axis) or less. For example, at ratio 2, the function takes the value 0.75, meaning that 25% of BitTyrant downloads finish in half the time of Azureus or less. Depicted this way, every point to the right of ratio 1.0 represents a performance improvement. For the vast majority of live Internet swarms, BitTyrant's strategic behavior improves performance.

Although these results demonstrate the significant performance benefits BitTyrant can realize today, the long-term outcome of strategic behavior on aggregate BitTorrent performance is unclear. Our paper [2] provides further experiments comparing BitTyrant and BitTorrent behavior, in particular examining the performance outcome if all users adopt BitTyrant. The performance for all BitTorrent users today depends on the altruistic contributions that all peers make. BitTyrant improves performance by identifying these altruistic contributions and reallocating them if possible. Because the bandwidth capacity of Internet end hosts is so skewed, high-capacity and even moderate-capacity peers tend to have such a disproportionate share of total resources that even after BitTyrant allocates all available bandwidth strategically, excess capacity remains. BitTyrant presents these users with a choice. If they continue to contribute all available capacity even after identifying the altruistic portion, overall performance improves. Alternatively, overall performance degrades if altruistic contributions are withheld. Ultimately, whether or not incentives stronger than BitTorrent's tit-for-tat are needed in future P2P systems will be determined by user behavior.

The BitTyrant client implementation we developed during the course of our work is publicly available for Windows, Mac OS X, and Linux at http://BitTyrant.cs.washington.edu and has received hundreds of thousands of downloads to date. Full source code for all platforms is also available.

FUNDING ACKNOWLEDGMENT

This work was supported by NSF CNS-0519696 and the ARCS Foundation.

REFERENCES

[1] Eytan Adar and Bernardo A. Huberman, "Free Riding on Gnutella," *First Monday* (October 2000).

[2] Michael Piatek, Tomas Isdal, Thomas Anderson, Arvind Krishnamurthy, and Arun Venkataramani, "Do Incentives Build Robustness in BitTorrent?" *Proceedings of the 3rd USENIX Symposium on Networked Systems Design and Implementation (NSDI '07),* 2007.

ALVA COUCH

should the root prompt require a road test?

MUSINGS FROM A SYSTEM ADMINISTRATION TEACHER ON THE IMPORTANCE OF CAREFULLY OBSERVING BEHAVIOR, AND THE TRUE COST OF CERTIFYING A SYSTEM ADMINISTRATOR

Alva Couch is an Associate Professor of Computer Science at Tufts University, where he and his students study the theory and practice of network and system administration. He served as Program Chair of LISA '02 and was a recipient of the 2003 SAGE Professional Service Award for contributions to the theory of system administration. He currently serves as Secretary of the USENIX Board of Directors.

couch@cs.tufts.edu

HOW DOES ONE HIRE A COMPETENT system administrator? As one who has taught system administration at the college level a small number of times in the past few years, I cannot help but wonder what people are thinking when they hire a "certified" system administrator over an "uncertified" one. With a small number of exceptions, most certification programs require only passing a written test to become certified. In teaching system administration, I have given many written tests to my students, and I can testify from personal experience that it is possible to pass a written test on system administration and not have the slightest clue about how to function as a system administrator.

Because I possessed a "test form reader" that could read standard test forms, and because I possess in good measure Larry Wall's "attributes of a good programmer" including laziness, impatience, and hubris, I was strongly motivated to make multiple-choice questions "work" to evaluate students so that the card reader could do the work of grading for me. Thus I went to great lengths to make multiple-choice questions "difficult enough" to demonstrate system administration knowledge instead of luck. My early efforts included a test with 10 possible answers per question, and a test with 100 true/false questions, where with each incorrect answer a point gets subtracted from one's score, so that the "random chance" score is 0. I know of at least one student who passed each of these tests with flying colors whose incompetence as root became clear when one *observed the person's behavior*. So those tests were out of the running as an effective measure of system administration competence. The form reader went into the trash.

Currently, I use fill-in-the-blanks questions rather than multiple-choice questions, to give examinees much more creative ways to make mistakes. This takes a little more time to grade, but at least I hope that the student who passed the multiple-choice tests above without knowing anything would not have a chance of answering fill-in-the-blank questions correctly. Still, I would not give an examinee with a perfect score on these tests access to root on *my* systems.

Instead, I might watch a potential administrator perform some tasks and rate how well the person

navigated and managed the complexity of the tasks, how the person's own knowledge gaps were filled, and how many false starts and poor practices occurred between start and finish of each task. How can we put this "watching" into a reusable container that others can use as well? The answer may lie in studying how other fields certify their practitioners.

Licensing Drivers

My experiences as tester seem to indicate that no matter how carefully or well a written test is constructed, one cannot expect that a person who passes a written test on system administration can "take the driver's seat" without problems, any more than one can safely put a teenager in control of a car solely based upon his or her scores on the written exam. This leads me to consider parallels between driving tests and system administration certifications that might guide us to a better understanding of certification and what different kinds of certification might mean.

There are three components to obtaining a typical driver's license as a teenager:

- Experience: Has one read the rules and sat behind the wheel for a while?
- Knowledge: Is one aware of the laws and regulations?
- Function: Can one control a car in a realistic situation?

A typical teenager gains experience and knowledge from a driver training course and takes two tests: a written test of knowledge and a road test of driving under realistic conditions. A universal aspect of driver (and boat or pilot) licensing is that *passing the written test entitles one to take the road test*, and nothing more.

Following the parallels between driver licensing and system administration, *current written test certifications seem to entitle the certified person to a "road test,"* and nothing more. Employers must perform the road test themselves, the hard way, by putting "certified" individuals at the root prompt and observing in a live situation whether problems develop. This is like putting the teenager who has passed the written exam at the wheel, and then taking the license away only after the first pedestrian is run over.

What, then, comprises a good system administration "road test"? We can learn again from the components of typical driving road tests; one has to be able to function in a variety of common situations and make good decisions in those situations. But there is another component to the driving test that is often ignored: There is an *observer* who evaluates factors other than just being able to perform the tasks requested. Confidence, attitude, and situational awareness are being observed as well. The driver's test is not scored solely on whether one can navigate from point A to point B without accident, but rather on how one approaches the problem and handles unexpected events.

Situational Awareness

Here, human factors research can help us understand the problem of "driving" either automobiles or configuration changes. The "situational awareness" component of driving, flying, and even managing systems has been extensively studied, mostly to determine whether controls and cockpit layout aid or hinder the driver or pilot in performing a task. In human terms, the expert driver exhibits constant eye movement, checks constantly beside and behind the car when driving, and maintains a "situational awareness"

that helps the driver react quickly to contingencies. This awareness can be tested, but only through direct observation of someone driving, and not simply by observing successful navigation from point A to point B.

As an example, let us consider two system administrators engaged in a road test. Each system administrator is asked to configure security for an Internet service. The first system administrator (X) tries to edit an appropriate file, realizes that it is not writeable to his or her user, su's to root, makes the file world-writeable (chmod 777), edits the file, and then makes the file normally writeable (chmod 644), without checking the prior protections of the file. The second system administrator (Y) lists the file to check protections, looks up the appropriate format for the change in the documentation, su's to root, edits the file, and exits the root session. Which of these system administrators should pass the road test?

Passing the first system administrator (X) on this road test is like passing a new driver on a road test in which the driver smashes the bumper of a parked car when parking, but pays the owner in full for the damage and replaces the owner's bumper with one of his or her own choosing without referring to the model of car that was hit. Further, during the road test, the driver leaves the car on the highway with the motor running, steps out to shop (during which time anyone could have gotten behind the wheel and driven away), comes back as if nothing has happened, and considers this all to be good driving practice.

The main point of this simple example is that the road test should measure process rather than product. It is not sufficient that system administrator X successfully moved from point A to point B, given that the path taken from A to B was problematic and irresponsible. But, to determine this, the examiner must have a high level of situational awareness, so that it will be possible to determine when mistakes are being made. Finally, the examiner must be an expert and trained observer in order for the road test to be a reasonable measure of prowess. Speed is not as important as safety and wise choices.

There are a spectrum of tools available to certify system administrators, from purely written tests to "live tests" in which a problem must be diagnosed and repaired. But the preceding example shows a situation in which simply accomplishing an action is not enough; the administrator must also adhere to good practice throughout, in action, word, and deed. To the best of my knowledge, no certification program—including even the highly acclaimed RedHat program that does require a live skill test—requires a stringently observed road test of this kind, in which the person to be tested is watched carefully and evaluated on process by an expert human observer.

I hope that potential employers understand this and conduct road tests of their own before allowing a "certified" system administrator free rein over their systems.

The burning question that arises is whether we *can* or *should* provide some external neutral mechanism by which system administrator "road tests" can be performed, or whether such "road tests" are forever the responsibility of potential employers. In seeking a neutral, reusable "vehicle" for road tests, one can seek inspiration in the way that driver licensing is made reusable and widely applicable.

Measuring Trust

A second lesson we can learn from driver licensing is how one progresses from the basic certification to being certified at higher levels. The federal government in the United States mandates a "class system" in which there

are three main classes of commercial drivers, defined by the weight or passengers of the vehicle to be controlled:

- Class A: Any combination of vehicles with a gross vehicle weight rating (GVWR) of 26,001 or more pounds, provided the GVWR of the vehicle(s) being towed is in excess of 10,000 pounds
- Class B: Any single vehicle with a GVWR of 26,001 or more pounds, or any such vehicle towing a vehicle not in excess of 10,000 pounds GVWR
- Class C: Any single vehicle, or combination of vehicles, that does not meet the definition of Class A or Class B, but is either designed to transport 16 or more passengers, including the driver, or is placarded for hazardous materials

As well, there are two noncommercial classes of license:

- Class D: Regular noncommercial vehicles
- Class M: Motorcycles

Finally, there are "endorsements" that one must obtain in order to drive under special conditions:

- T—Double/Triple Trailers (knowledge test only)
- P—Passenger (knowledge and skills tests)
- N—Tank Vehicle (knowledge test only)
- H—Hazardous Materials (knowledge test only)
- S—School Buses (knowledge and skills tests) [1]

The license itself is only part of the picture: The general license class is a measure of knowledge, skill, and trust, whereas the endorsements are indications of knowledge and skill.

This system is not based upon measuring capabilities of drivers, but, rather, upon the kind of vehicle to be controlled and the kinds of risks that must be mitigated in controlling that kind of vehicle. This practice inspires me to rethink how we might categorize certifications for system administrators. Almost exclusively, we tend to think of certifications as proving proficiency for specific platforms or products or even specific courses of study. A "certification to configure Cisco switches" seems—in light of this discussion—to be similar in character to a "license to drive BMWs," whereas a "certification" based upon taking a specific course in networking seems similar to a "license to drive in Boston." Both of these kinds of "certifications" look more like "endorsements," that is, something to be attached to an overarching "license" to indicate specific capabilities.

Simplicity

Another thing one can learn from the driver's road test is simplicity: Several basic skills are tested and nothing more. The reason for the simplicity of road tests is straightforward: Road tests are expensive to administer and thus must be made generic to be cost-effective. Is system administration any different? If it is the same, then what categories can we suggest for licensing and testing, and what can we learn from the structure of driver testing that can help us? Let us try a simple thought experiment and consider how system administration certifications might be categorized by risk class and what endorsements might apply. This gives rise to a very different design for certification than what is available now.

Let us consider one model of coarse risk classes that might be the basis for a new model of licensing and certification:

- Class D: Individual workstation, including multiple workstations that are part of some LAN or enterprise
- Class C: Server, including the potential that multiple servers are administered
- Class B: LAN, including basic automation of workstation and server configuration
- Class A: Enterprise, including enterprise-wide design and implementation

I do not defend this as definitive; it is just a "straw dog" proposal for how a "license-based" scheme of certification might be structured. The key ingredient of this system is that the class of a certification is based upon the risk entailed in managing that kind of system or network, and not upon a capabilities model of the administrator. The question is not whether a driver is *capable* of driving a larger vehicle, but whether she or he can be trusted to do so both safely and responsibly.

In turn, each system administrator "road test" would measure whether a person can be "trusted" with the responsibilities of that level of management. This is not a platform-specific basis for trust, even though someone might well be tested in the context of a specific platform. What is being tested is not the knowledge of the platform, but, rather:

- How the candidate responds to contingencies
- Whether the candidate adopts responsible practices in making changes
- Whether the candidate is aware of the effects of his or her actions
- Whether the candidate has appropriate knowledge-acquisition skills and understands personal limits
- Whether the candidate has effective interpersonal skills
- Whether the candidate is aware of and acts in accordance with the legal, ethical, and professional obligations of the system administrator

In other words, an administrator who has complete mastery of Linux, but who obviously does not follow the code of ethics when it is more convenient to ignore it, should fail the test. This is analogous to the driver's test in which the candidate has exceptional control of a vehicle but drives on sidewalks where possible.

The second component of a license-based model is a set of "endorsements" that might be obtained to operate networks in subclasses of the classes just listed:

- Directory services
- Electronic mail and spam
- File service, SAN, NAS, etc.

Note that the currently available "certifications" have become "endorsements," and many of these do *not* require a live skill test.

The risk model derived from motor vehicle operator licensing seems to imply that we are missing an overarching level of certification—involving some form of road test—before and not as a substitute for the specialty certifications that system administrators currently pursue. This taxonomy suggests to me that current certifications are valuable only in certifying skills of those system administrators who "already know how to drive" and—in some sense—are already worthy of some form of "driver's license."

Driver Training

A world full of licensed drivers who learn how to drive by causing accidents is a frightening thought, but it seems that this is exactly what we have in

training system administrators. The lack of a road test that examines the most basic of skills, combined with certifications that measure only advanced knowledge, creates a knowledge gap that only experience fills. This hurts our image as professionals and makes the path to proficiency both haphazard and painful.

So what can be done to address this? I have no magic solutions, but I can suggest some strategies that might greatly improve training in the profession. Some of these are quite controversial, and I expect it would take years for us to agree on some of these points, but I will state them anyway:

- *Emphasize understanding of effect* of an action rather than just "what action to perform." Aim for situation awareness rather than mastery of recipes.
- *Replace "best practices" with "responsible practices"* as the most basic form of system administration education. In other words, emphasize responsibility rather than optimality in training beginners.
- *Emphasize professionalism and appropriate behavior* before skill and knowledge.
- *Emphasize experience as part of certification.* Just as truck drivers have to have a certain level of driving experience before they can achieve higher license classes, experience is underrated as a certification element.

To some extent, we are already doing this, but to some extent, we are also failing. Every employer has to devise a custom road test, or suffer the consequences. Every system administrator has to learn the hard way. "Road tests" (and the training that comes with them) are a distant pipe-dream that no one knows how to implement.

But there is one concrete and unavoidable conclusion from this essay that I do not believe is controversial: *Current certifications test knowledge and not professional practice*. If we develop a road test for system administration, it will not simply be about accomplishing tasks, but about behaving in a professional way in reacting to contingencies and requests. It will certify safety, responsible behavior, and situational awareness rather than demonstrating knowledge of how to configure Microsoft Windows. The latter is an endorsement that can best be earned when the candidate already has a firm foundation in professional practice.

This is not to say that current certifications do not have value. I am afraid, however, that the value that they represent is often misinterpreted or taken out of context. There is value in the knowledge of how to drive an 18-wheeler, but without a road test, there is no confidence that the person can be trusted to utilize that knowledge wisely. Change "18-wheeler" to "enterprise network" to obtain a clear picture of the crisis at hand.

Nothing springs into existence from nowhere without some form of evolution. We are "evolving" toward "road tests" at an alarming rate, in the same way that traffic accidents cause stoplights to spring up at busy intersections. It might be time for system administrators to rise to the occasion and fill the certification gap themselves, before someone else thinks of doing it for us.

REFERENCES

[1] Massachusetts Driver's License Web site.

OCTAVE ORGERON

an introduction to logical domains

PART 1

Octave Orgeron is a Solaris Systems Engineer and an OpenSolaris Community Leader. He currently offers consulting in the financial services industry but has experience in the e-commerce, Web hosting, marketing services, and IT technology markets. He specializes in virtualization, provisioning, grid computing, and high availability.

unixconsole@yahoo.com

IN RECENT TIMES, VIRTUALIZATION has become a requirement for many businesses looking to consolidate physical servers and increase utilization. This has led to many innovations at both the software and the hardware level to address the virtualization requirement. One such innovation is Sun Microsystems' new product called Logical Domains, or LDoms. This allows a single physical server to be virtualized into multiple discrete and independent operating system instances. LDoms present many opportunities for consolidation in modern data centers where physical space and power are at a premium.

This is the first of three articles that will introduce you to the Logical Domain technology. In this article, I will introduce the basic concepts and components of this new technology.

The Niagara Processor and the UltraSPARC Hypervisor

The Niagara processor is a chip multithreading design that leverages the power of multiple CPU cores running many hardware threads simultaneously. The first generation, known as the UltraSPARC-T1, was introduced on the Sun Fire T1000 and T2000 servers. The processor has up to eight CPU cores with four hardware threads each, for a total of 32 threads. The CMT design enables the processor to achieve significant increases in performance over the UltraSPARC IIIi processor for multithreaded applications. The processor has unique features, such as a cryptographic unit that traditionally would require an add-on accelerator card. In the future, the Niagara 2 platform will integrate more advanced features, such as 10-Gb Ethernet, enhanced cryptography, and enhanced floating-point performance. One of the more interesting features of the Niagara processor family is the support of a hypervisor for virtualization.

Hypervisors provide a virtualization platform for running multiple operating system instances. Hypervisors have been around since the 1960s, starting with IBM's CP/CMS, the ancestor of IBM's current z/VM solution. Until recently, such technology was only found on such proprietary platforms. However, with the advent of Xen and VMware ESX, hypervisors are becoming more common-

place. The hypervisor found in Sun's Niagara architecture, known as the UltraSPARC hypervisor, is a new addition to this growing virtualization methodology.

The UltraSPARC hypervisor is a thin layer of software stored within the ALOM CMT firmware. It creates a layer of abstraction between the operating system and the physical hardware. Traditionally, operating systems have the concept of nonprivileged and privileged access to the underlying hardware. The hypervisor introduces an additional layer of privileged access, known as hyperprivileged access. Hyperprivileged access enables the hypervisor to either expose or hide resources from an instance of an operating system. This allows resources to be grouped into logical partitions or domains. This is similar to Sun's Dynamic System Domains, with the main difference being that the resources are not electronically partitioned, but virtualized.

Resources such as CPU threads, cryptographic threads, and memory are partitioned into a logical domain. Other resources are virtualized and serviced through the use of Logical Domain Channels, or LDCs. LDCs provide secure communication and data pathways between LDoms and the hypervisor. This allows an operating system in one LDom to make an I/O request, which is serviced by another LDom that has privileged access to the underlying hardware. The abstraction reduces the I/O overhead in one LDom and passes it to another LDom that is capable of completing the request.

However, the hypervisor cannot accomplish this on its own. The processing of I/O requests requires CPU cycles, device drivers, etc. There is also the aspect of configuration and management of the platform as a whole. These different aspects of the platform lead to the division of responsibilities to unique logical domain types.

Logical Domain Types

There are several types of logical domains that can be configured. Each type plays a specific role in the logical domain architecture. Some of these roles overlap, but they can be separated for flexibility. The basic differences are shown in Table 1.

Logical Domain Type	Description
Guest	Domain that is a consumer of virtualized devices and services
I/O	Domain that has privileged access to a PCI-E controller but does not provide virtualized devices or services to guest domains
Service	I/O domain that has privileged access to one or more PCI-E controllers; provides virtualized devices and services to guest domains
Control	Service domain that runs management software to control the hypervisor configuration of the platform

TABLE 1: LOGICAL DOMAIN TYPES

GUEST DOMAINS

A guest domain is a virtualized environment that has no direct access to the underlying physical hardware beyond the CPU threads, cryptographic threads, and memory resources. It does not have direct ownership of any

hardware devices. A guest domain does not provide virtual services or devices to other LDoms. It is a consumer of the virtual services and devices provided to it by the control and service domains. Guest domains consist of the following components:

- CPU threads
- Cryptographic MAU threads
- Memory
- Virtual console
- Virtual OpenBoot PROM
- Solaris 10 Update 3 or above
- Virtual networking
- Virtual storage

The guest domain is the target virtual environment for deploying applications and services. It functions as a normal Solaris instance with the exception that its underlying networking and storage are completely virtualized. This means that normal Solaris operations such as Jumpstart, package and patch management, running network services, account management, etc., all function without any changes. Also, advanced features such as boot disk mirroring or network multipathing function transparently. It is even possible to run Solaris Containers within a guest domain, adding another layer of virtualization.

I/O DOMAINS

An I/O domain is a virtualized environment that has privileged access to a portion of the underlying hardware platform. Specifically, an I/O domain has privileged access to a PCI-E controller and the devices that are connected to its ports. This allows it to have direct control over network ports and storage that are connected to that PCI-E device tree. However, an I/O domain does not virtualize access to its hardware for guest domains. As such, I/O domains differ from guest domains by having:

- Privileged access to a PCI-E controller and its devices
- Physical access to networking
- Physical access to storage

I/O domains may be useful for applications such as databases that require direct or raw access to storage devices. However, they do consume an entire PCI-E controller and the devices connected to it. This can reduce the flexibility of the hardware platform, but it may be of some use for specific applications.

SERVICE DOMAINS

Service domains are virtual environments that provide virtual resources to guest domains. The service domain takes ownership of one or more PCI-E controllers, similarly to an I/O domain. However, it virtualizes the devices connected to those controllers as a service for guest domains. This is accomplished by having the kernel device drivers, within the service domain, front-ended by virtual device services. When a guest domain interfaces with a virtual device, the request is handled by the corresponding service domain through LDCs. This happens transparently to the operating system in the guest domain. Service domains differ from I/O domains by having:

- Privileged access to one or more PCI-E controllers and their devices
- Virtualized devices and services for guest domains

The control domain is a service domain with management software that is capable of configuring the platform. By default, the control domain is the first service domain for the platform and as such is referred to as the primary domain. This LDom can be accessed directly by the physical hardware console. This dual role allows the primary domain to configure, manage, and provide virtual services for the platform. The differences between the primary domain and a standalone service domain involve the former's physical hardware console, Logical Domain Manager software, and virtual console concentrator.

The Logical Domain Manager (LDM) software is the management layer that is aware of the mappings between the physical and virtual resources. The LDM software provides an easy command-line interface for the configuration and management of LDoms. Through the use of LDCs, the LDM software can control the hypervisor configuration. It also configures virtual services in service domains, controls dynamic reconfiguration, and provides virtual consoles for each LDom.

It is important to note that the control domain is the only domain that runs the LDM software and is responsible for configuring the server as a whole. For standalone service or I/O domains, no additional software is required beyond the standard Solaris installation.

Virtual Services and Devices

Logical domains are consumers in one way or another of virtualized services and devices. These virtualized services and devices form the building blocks for logical domains. They provide the processing, memory, and I/O components for logical domains. Tables 2 and 3 identify and describe virtual services and device types.

Virtual Services	Description
VLDC	Virtual Logical Domain Channels. These act as communication channels for logical domains and the hypervisor. Services such as dynamic reconfiguration, FMA events, Service Processor events, and communications between guest domains and services domains utilize VLDCs.
OBP	OpenBoot PROM. Each logical domain has its own OpenBoot PROM instance. The NVRAM variables are stored within the hypervisor.
VCC	Virtual Console Concentrator. The VCC provides a virtual console for each logical domain. This can only be provided by the control domain.
VSW	Virtual Switch Service. VSW provides virtual network access for guest domains to the physical network ports.
VDS	Virtual Disk Service. VDS provides virtual storage services for guest domains.

TABLE 2: VIRTUAL SERVICE TYPES

Virtual Devices	Description
VCPU	Virtual CPU. Each UltraSPARC-T1 CPU consists of 4, 6, or 8 cores with 4 threads. Each thread can be allocated as a virtual CPU.
MAU	Mathematical Arithmetic Unit. Each Niagara CPU core has a thread to a Cryptographic MAU, which provides accelerated RSA/DSA encryption.
Memory	Physical memory can be virtually mapped into a logical domain.
IO	PCI-E controller that is allocated to a service domain.
VCONS	Virtual Console. This port in a guest domain is connected to a VCC service in the control domain.
VNET	Virtual Network. This port in a guest domain is connected to a VSW service in a service domain.
VDSDEV	Virtual Disk Service Device. The VDSDEV is a physical storage medium that is virtualized by a VDS in a service domain.
VDISK	Virtual Disk. VDISK in a guest domain is connected to a VDS in a service domain.

TABLE 3: VIRTUAL DEVICE TYPES

In this article I have introduced the basic concepts and components of logical domains. By understanding the relationships among the different logical domain types and their virtual resources, it will be easier to explore this new technology. In my next article I will explain the installation and configuration of logical domains in detail.

RESOURCES

Home page for Logical Domains:
http://www.sun.com/servers/coolthreads/ldoms/index.xml.

Documentation for Logical Domains:
http://docs.sun.com/app/docs?q=ldoms.

Sun BluePrint document:
http://www.sun.com/blueprints/0207/820-0832.html.

Sun BigAdmin site for LDoms: http://www.sun.com/bigadmin/hubs/ldoms/.

Home page for OpenSPARC source code and specifications:
http://www.opensparc.net/.

EMIN GÜN SIRER AND RIK FARROW

some lesser-known laws of computer science

Emin Gün Sirer is an associate professor of computer science at Cornell University. His recent research centers on self-organizing peer-to-peer systems and a new operating system for trustworthy computing.

egs@systems.cs.cornell.edu

Rik Farrow is Editor of *;login:*.

rik@usenix.org

Note: This work is based on a Four Minute Madness talk created by Emin Gün Sirer and presented during HotOS XI .

GORDON MOORE'S FAMOUS LAW [1], which successfully predicted an exponential increase in the complexity of integrated circuits, has had wide impact. It has not only captured the reasons behind the meteoric rise of computer science as a discipline, but it also predicted and shaped expectations for new processors. It has withstood the test of time and has become a household concept across the globe. We look at some other trends in computer architecture that have so far been ignored and provide extrapolations of where other trends might lead us in the decades to come.

Moore's Law

Gordon Moore was working for Intel when he came up with his eponymous law. Moore's Law is not a real law, in the sense of a universal and invariable fact about the physical world (the sense in which real scientists use the term), but it comes close in describing a phenomenon that has shaped the latter half of the 20th century. At the time Gordon Moore came up with his law, he was attempting to extrapolate the growth of component density that could be successfully manufactured using integrated circuit technology. In 1965, Moore wrote, "The complexity for minimum component costs has increased at a rate of roughly a factor of two per year."

Five years later, the Caltech professor and VLSI pioneer Carver Mead actually started describing this brief statement as Moore's Law. Over time, Moore's Law evolved into the statement that the performance of computers doubles every 18 months. There are three remarkable things about this law. First, it was based on very scant data when it was first formulated. Gordon Moore was not afraid to make bold predictions based on facts as they were available to him. Second, the mere enunciation of the law affected the manufacturing standards to which new technologies were held, thus reinforcing the law itself. This suggests that the act of naming a law helps render it valid, or at least causes the world to bend slightly to accommodate the law, thus overlooking slight errors in extrapolation. Finally, any law that can last more than 40 years and carries its observer's name guarantees instant name recognition (not that Gordon Moore, a renowned computer architect and co-founder of Intel, would

have needed it), which creates incentives for other computer scientists to replicate the feat by observing other trends.

It is not surprising, then, that many other "laws," similar in spirit to Moore's, have been proposed. For instance, it has been well known for quite some time that disk drive capacities tend to double every 18 months. In an article published in *Scientific American* in 2005, this observation was dubbed "Kryder's Law," after Mark Kryder, senior vice president and CTO of Seagate. For this deed, Kryder may achieve the same level of fame as Moore.

Other people have attempted this dual feat of devising a law that successfully matches the growth of some computer-related feature and becoming famous. Barry Hendy of Kodak Australia coined Hendy's Law, "The number of pixels per dollar found in digital cameras will double every year." Although a user named Barry.hendy has edited Wikipedia to insert Hendy's Law under the discussion for Moore's Law, it has so far failed to achieve the level of notoriety that Moore's Law has received.

Power Laws

Our quick look around at existing laws suggests that the low-hanging fruit has already been picked. But that is not to say that nameable observations have been exhausted. Careful examination of other trends in computer system architecture may indeed allow us to find new laws of exponential growth, especially if we, in the footsteps of Gordon Moore, allow ourselves to work from sparse and noisy data sets. In the rest of this article, we examine various trends in the hope of predicting the future of computer systems.

The first and most obvious trend is a tactile one that is taking place on or under every desktop. From around 1950, up until about 1995, computers had a single power button. Old-timers will fondly remember that "big red button" that promised to cut power (in an emergency) to mainframe computers. The very allure of such buttons led to their being enshrined in plexiglass cases lest an enthusiastic visitor be tempted to see if the button actually did something [2]. Most other computers, from minicomputers to workstations to early PCs, also came with a single power button, often located in the back next to the power cord.

Fast-forward to 1995, and you can see that PCs now have two power buttons, one on the back and one on the front. The Power Button Law thus presents itself: The number of power buttons doubles every 50 years. Although our data set is small, we can already see evidence that this trend is continuing: If you look just beneath the power button on most PCs, you can see a budding "proto-button" that, depending on the operating system and perhaps the phase of the moon, restarts the machine or does nothing. We are confident that it will slowly develop into a full-fledged power button over time (Figure 1).

FIGURE 1: A 1995 PC CASE (TOWER FORMAT), WITH PROTO-BUTTON APPEARING JUST BELOW THE FRONT PANEL POWER BUTTON

The observant reader will already have detected a problem with our first candidate for the Power Button Law: We may have to wait until 2045 until we have enough data points (or power buttons). Perhaps other candidates for power laws will be easier to assess.

Until 1990, PC-class machines did not make noise. Well, there was the familiar keyboard beep, along with other less consequential sources of noise (fans and hard drives). What changed around that time is that PCs started to include CD drives capable of playing music CDs. These CD drives had a single volume control, typically a hardware knob.

Fast-forward several years, and PCs running Windows have four means of controlling volume: the hardware volume control knob, the audio driver settings, the system-wide mixer setting, and the application-level volume setting. Linux and Mac OS are not far behind, either! This leads us to the Volume Control Law: The number of volume controls on a computer doubles every five years. Note that this law has a shorter period, and thus it will be quicker to verify. Still, the number of data points acquired so far is vanishingly small, leading us to continue our search for power laws.

It is well accepted among researchers that sensors are tiny computing platforms fundamentally limited in resources. For academicians, this has led to a tremendous boon: There has been much research targeting sensor platforms. Yet, despite the seemingly universal belief that sensor platforms will forever resemble 8-bit computers from 1983, actual sensors deployed in the field have been evolving rapidly. In Table 1, we summarize the salient features of several sensor platforms (from a period covering 1998 to 2002 [3]) to see whether we can observe any architectural trends.

	WeC 1998	Dot 2000	Mica 2 2002
CPU speed	~4 Mhz	~8 Mhz	~16 Mhz
Program memory	8 Kbytes	16 Kbytes	128 Kbytes
RAM	0.5 Kbytes	1 Kbyte	4 Kbytes
Power	45 mWatts	45 mWatts	75 mWatts
LEDs	~0	~1	~2

TABLE 1: SENSOR MOTE EVOLUTION, SHOWING A SLOW BUT STEADY INCREASE IN RESOURCES

Several interesting trends present themselves. First, we can see that the CPU speed doubles every ~2 years, roughly in step with Moore's Law (so sensors are not so static after all, and Gordon Moore beat us to the observation, yet again). Similarly, program memory as well as RAM doubles every ~1.5 years (this is a conservative estimate that does not account for the type of breakthrough flash memory might bring). And although our data on the number of LEDs is noisy, it appears that the number of pixels available on a sensor is doubling every ~2 years.

Based on these laws, we can extrapolate what sensor motes will be like in 2020 (12 years from now, plus 1 to be conservative):

- Mote CPUs run at 1 GHz.
- Motes have 32 MB program memory and 1 MB RAM.
- Motes house somewhere between 64 and 192 pixels, which would resemble a Christmas tree if they were to consist of LEDs, so are more likely arranged as a scrolling dot-matrix LCD display.

Pushing our extrapolation out a bit farther to 2050, sensor motes will, by then, have:

- 65-GHz CPUs
- 2 TB of program memory and 67 GB of RAM
- 1024x768 pixel displays
- 4 power buttons
- 2048 volume controls

Summary

Following power laws to their illogical but consistent conclusions produces ridiculous results. Thus, we now have a candidate for a lasting law: The obsequious following of power laws inevitably leads to impossible predictions.

REFERENCES

[1] Moore's Law: http://en.wikipedia.org/wiki/Moore%27s_law.

[2] Alex Papadimoulis, "The Big Red Button": http://worsethanfailure.com/Articles/The_Big_Red_Button.aspx.

[3] The family of motes: webs.cs.berkeley.edu/papers/hotchips-2004-mote-table.pdf.

MASSIMO BERNASCHI, FRANCESCO CASADEI, AND SAMUELE RUCO

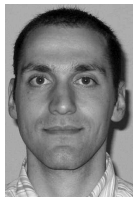
migration of secure connections using SockMi

Massimo Bernaschi joined the IBM European Center for Scientific and Engineering Computing (ECSEC) in 1987 in Rome, where he spent ten years working in the field of parallel and distributed computing. Currently he is Chief Technology Officer of the Institute for Computing Applications, which is part of the Italian National Research Council (CNR). He is also an adjunct professor of Computer Science at "La Sapienza" University in Rome.

massimo@iac.cnr.it

Francesco Casadei has a master's degree in Computer Science from Rome University "La Sapienza." Since October 2005 he has been with Quadrics Ltd as a software engineer. Before then, he spent six months at the IBM T.J. Watson Research Center (Yorktown Heights, NY) as a research scholar.

francesco.casadei@quadrics.it



Samuele Ruco graduated in Computer Science in 2006 from Rome University "La Sapienza." Since then he has been with Quadrics Ltd, a Finmeccanica Company, working on embedded systems and parallel programming.

samuele.ruco@quadrics.it

***SOCKMI* IS A SOLUTION FOR MIGRATING** SSL/TLS secure connections between Linux systems that extends recent work on TCP/IP migration [1]. Secure Socket Layer (SSL) and Transport Layer Security (TLS) [2] add security to any protocol that uses reliable connections, such as TCP, to establish a "virtual circuit" from a client to a server. We chose to provide support for SSL migration because it is one of the leading technologies used today to secure connections, especially those to applications hosted by Web servers. It can be used for encapsulation of various higher-level protocols such as http (to form https), ftp, smtp, and nntp. It can also be used to tunnel an entire network stack to create a Virtual Private Network (e.g., in OpenVPN [3]).

The migration mechanism involves the following levels:

- Network: IP packets must be redirected to the importing host (the target system).
- Transport: All TCP information must be transferred to the host that imports the connection.
- Application: Session keys and other sensitive data needed to ensure the integrity of the secure connection are likewise migrated to the target system.

There are a number of situations in which the migration of secure connections can be useful: for instance, when there are requirements of load balancing, quality of service, and fault tolerance and it is not possible (or is undesirable) to restart the connection. With respect to other solutions, (i) it is able to migrate both ends of a connection; (ii) it does not require cooperation on both ends; (iii) it can be activated in any phase of the connection; and (iv) it does not require changes to existing Linux kernel data structures and algorithms. Moreover, the mechanism used for sending specific application-level information can be used for any application protocol.

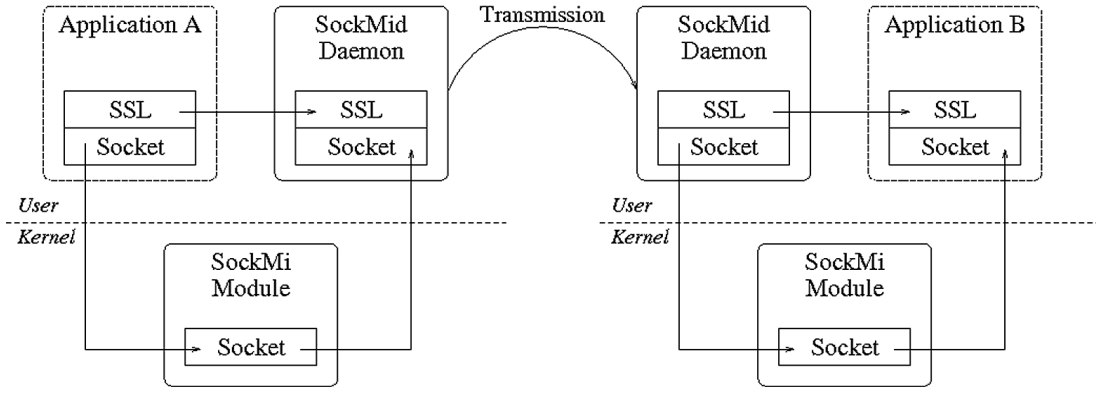


FIGURE 1: THE DESIGN OF THE MIGRATION MECHANISM IMPLEMENTED IN SOCKMI

The design of SockMi (see Figure 1) aims at achieving the following goals:

- Transparency: The connection endpoint that does not migrate should not be affected by the migration mechanism in any way with the exception of possible (but limited) delays owing to the "triangulation" mechanism described in Ref. 1. This implies that no information should be exchanged between the peers to accomplish the migration.
- Portability: The migration mechanism should have minimum impact on the underlying operating system, meaning that: (i) no patches to the kernel must be required; (ii) no new system calls must be introduced. To fulfill these requirements, we implemented the migration mechanism as a loadable kernel module (LKM) and defined an API (application programming interface) that hides all implementation details.
- Symmetry: It should be possible to migrate both connection endpoints.
- Versatility: It should provide a general mechanism that supports the migration of other application protocols.

The main components of SockMi are the module, the daemon, the API, and the IP redirection mechanism.

The module is the core of the TCP/IP socket migration mechanism. In particular, the module is responsible for: (i) saving (restoring) the state of migrating sockets during the export (import) phase; (ii) exchanging information about migrating sockets with the SockMi daemon; (iii) providing low-level primitives to activate and control the socket migration facility.

Besides the state of the socket, migrates also affect the corresponding "in-flight data." These data are found in the receive and transmit queues of the socket, which contain, respectively, packets received by the system but not read by the application and packets to be sent, or packets already sent but not yet ACKed.

The module holds sockets ready to be imported in three different import lists, corresponding to the TCP hash tables managed by the Linux kernel: (i) the bound socket list; (ii) the listening socket list; (iii) the connected socket list. These lists change their length dynamically when a new socket is received from the daemon or an application imports a socket. However, to avoid potential memory problems, we set a maximum length for each list. In case a list reaches its maximum length, no more sockets can be queued and the import fails with an error. The module is SMP-safe and supports the preemption mechanism available in the 2.6 kernel. Further information is reported in Ref. 1.

The daemon (SockMid) works in combination with the module to support the socket migration mechanism and in combination with the libappsockmi

library to support the application migration mechanism. The daemon carries out different tasks depending on the situation. During the *export phase*, it receives the application's data from the exporting process and the associated socket state from the module; during the *negotiation phase*, it communicates with other daemons running on other hosts in order to choose where to migrate the connection; finally, during the *import phase*, it writes the state of importing sockets to the module internal buffers and it handles the imported application protocols. Moreover, the daemon receives importing requests from local processes and checks whether a request matches an imported connection.

During the import and the export phases, the module and the daemon components need to exchange information about the state of migrating sockets. Since the module lives in the kernel address space whereas the daemon is a normal user process, it is not possible to resort to standard Inter Process Communication (IPC) primitives to exchange data between them. To overcome this difficulty we implemented a buffer sharing system via the mmap() primitive. The module is seen by the daemon as a character device that, through its mmap() file operation, makes its internal buffers available (i.e., it acts as a memory device). In this way, kernel buffers can be read and written by the daemon as if they were in user space. In addition, during these phases, the daemon and the libappsockmi library components need to exchange information about the application protocol data. To this end, the library sends http message requests to the daemon, which replies with a status code. When required, the reply message contains appropriate application data.

The socket migration entails the search for a host willing to "import" the connection. To be eligible to the import of a connection, a host must run an instance of SockMid, which defines and supports a communication protocol among daemons that run on different hosts. This negotiation protocol follows a plain request-response-confirm scheme that can be summarized as follows:

- When host A exports a connection, it sends a request in multicast to the daemons that run on other hosts.
- When a request arrives, a host—say, T—replies, provided that either the socket is explicitly exported to that host or no specific target host is defined in the request.
- If host A does not receive a valid response within a predefined timeout period, the migration fails.
- The first valid response triggers a confirmation mechanism by which host A notifies the daemon running on host T that it has been selected as the importing host.

Choosing the first valid response is a very natural yet simple policy. Other more sophisticated policies based on rules or heuristics could be used. For example, it could be useful to maintain statistics on previous migrations and select the target host in such a way as to achieve load balancing among importing hosts. Collective communication among the daemons relies on multicast. This means that all instances of SockMid have to join the same multicast group and bind() to the same UDP port.

SockMi provides a simple Application Programming Interface (API) to activate the connection migration mechanism. The API is implemented by two user space C libraries: libappsockmi and libsockmi, which are part of the distribution.

The libappsockmi library consists of two functions which provide applications with an easy-to-use method for importing and exporting secure connections: import_ssl() and export_ssl().

To import one or more secure connections, an application calls the library function import_ssl(). This function is designed to poll the availability of "exported" SSL sessions matching the import criteria specified by the application. If one or more matching sessions are available, then the function imports them immediately, by rebuilding the SSL session from the application data and by replacing the local socket associated with the connection. Otherwise, if no matching connection is available, the function waits until either a timeout occurs or one or more "exported" sessions becomes available for import. The prototype of the import_ssl() function is defined as follows:

```
int import_ssl(struct import_ssl_req *irqs, unsigned int nirqs, int timeout);
```

The arguments to import_ssl() are (in order) an array of import requests, the number of such requests, and the maximum waiting time until a successful import occurs (with a negative value blocking import_ssl indefinitely). The information required to formulate an import request are the following: (i) a pointer to the main OpenSSL structure to be replaced with the imported session; (ii) the preferred state the imported connection should have; (iii) the set of criteria a connection must match in order to be imported.

The import criteria let the application define the "properties" of the connection to be imported. Such criteria are the set of allowed secure connection states, the local and remote IP addresses, and the local and remote TCP ports. The import_ssl() function tries to fulfill all requests according to a best-effort policy. Upon completion import_ssl() returns one of the following values:

- 0, if the function timed out before any secure connection could be imported
- –1, if an error occurs
- the (positive) number of connections that have been successfully imported

Note that, even if successful, the function does not guarantee that *all* requests have been satisfied. Furthermore, even if the function returned an error, *some* secure connections may have been imported. Thus, upon return, the application should scan the array of requests and check the output field ssl_state, which either reports the state of the (possibly) imported session or is set equal to 0 to indicate that the request could not be satisfied.

Exporting SSL sessions is much simpler than importing, because there is neither need to specify criteria nor a wait time. To export secure connections an application calls the function:

```
int export_ssl(SSL *ssl, int state, int af, const void *to)
```

The first argument is a pointer to the main OpenSSL structure; it contains all the references to the information that needs to be transferred. The second argument is the state of the connection (e.g., connected client but SSL handshake not performed, connected server with SSL handshake performed, listening server). The last two arguments allow the network address of the importing host to be defined. The to argument may be a null pointer if there is no need to specify a target system. In this case the function export_ssl() lets the migration mechanism automatically select a target system, according to the internal policy of the SockMid daemon. This function returns 0 on success or –1 if an error occurs.

The libsockmi library consists of similar functions for sockets not associated to SSL sessions [1].

In the migration of secure connections, such as those provided by SSL, the difficulties arise primarily from the need of exporting and importing a num-

ber of keys and the information required to maintain consistency in the cipher subsystem. We tested our solution with the OpenSSL [4] implementation of the SSL and TLS protocols. OpenSSL is composed of two layers: (i) the SSL Record protocol, which is layered on top of TCP and allows the encapsulation of various higher-level protocols; (ii) the SSL Handshake protocol, which allows server and client to authenticate each other and to negotiate all security-related parameters (e.g., encryption algorithm, cryptographic keys) before the application protocol begins to transmit or receive data. The SSL Record protocol takes messages to be transmitted, fragments the data into manageable blocks, optionally compresses the data, applies a Message Authentication Code (MAC), and encrypts and transmits the result. Received data is decrypted, verified, decompressed, and reassembled, then delivered to the application. This protocol specifies four connection states: current read and write states and pending read and write states. Each state specifies a compression algorithm, an encryption algorithm, and a MAC algorithm. Thus the protocol must migrate all four connection states. In addition, it must migrate the parameters for the following algorithms: the MAC secret, the bulk encryption keys, the Initialization Vector (IV), and the sequence number for the connection in both read and write directions. The sequence number must be set equal to zero whenever a connection state becomes active, and it is incremented after each record. Moreover, it migrates other information, such as certificates and public and private keys. Currently, all these data are exchanged in the clear between the exporting daemon and the importing one. This assumes that the migration of OpenSSL connections happens in a controlled environment where there is no danger of key data being sniffed or malicious hosts offering to import connections just to grab connection-related data.

The target system uses this information to open new SSL sessions with the same SSL context. The data structures involved in defining the state of a secure connection can be determined by inspecting the SSL structure. These data structures have cross-references implemented as C pointers to memory locations. As a consequence, a simple approach based on data copy is not going to work, because pointers would make no sense in a different address space both for a migration to another host and for a migration to the same host. Thus a primary requirement of the migration mechanism is to preserve the referential integrity among the data structures that define the state of a connection.

To save the SSL information to a local memory buffer, an application calls the save_ssl() function defined as follows:

```
int save_ssl(void **pbuf, SSL *ssl, int ssl_state)
```

This function allocates a local buffer and saves to it the state of SSL connection. The arguments are (in order) a pointer to the output buffer, the pointer to the SSL structure that defines the connection, and the state of the connection. On success, this function returns the size (in bytes) of the allocated buffer. On error, –1 is returned.

To restore the SSL information from a local memory buffer, an application calls the load_ssl() function, defined as follows:

```
int load_ssl(void *buf, size_t buf_len, SSL *ssl, int ssl_state)
```

This function rebuilds session keys and other sensitive data needed to ensure the integrity of the secure connection. The arguments are (in order) the local memory buffer, the size (in bytes) of the buffer, the pointer to the SSL structure (whose contents are updated by the function), and the state of the secure connection. On success, the function returns 0; otherwise –1 is returned.

When a socket migrates to a different host it is necessary to redirect packets coming from the peer toward the host that imports the socket. To this purpose, we resort to a special combination of Network Address Translation (NAT) operations. In particular, we employ a Destination NAT (DNAT) such that packets received by the exporting host for the migrated socket are redirected to the importing host. For this redirection the standard NAT capabilities offered by the netfilter module of the Linux kernel are adequate [5]. The DNAT is triggered by the daemon running on the exporting host.

In addition, packets sent to the peer must have the same IP source address as the original host (for otherwise the peer would reply with an RST packet). In this case, we employ a Source NAT (SNAT) on the importing host such that the source address of packets sent by the imported socket is translated into the address of the exporting host. The SNAT required a modification to the standard NAT mechanism since the latter has a side-effect: The reply tuple is changed according to the applied address translation. The problem is that netfilter expects to receive packets having a destination address equal to the translated source address whereas, in our case, the DNAT sets the destination address equal to the real address of the importing host. To solve the problem, we resorted to the NAT helper mechanism available in the netfilter architecture. Basically, it allows us to invoke a custom procedure we wrote that performs the address translation but does not alter the reply tuple. Note that, in case the host that exports the socket can (or must) give up its IP address in favor of the host that imports the socket, it is much easier to add an "alias" IP address to the importing host. Further information about IP packet redirection is available in Ref. 1.

As shown in Figure 2, the migration consists of three phases: (i) export, (ii) negotiation, and (iii) import.

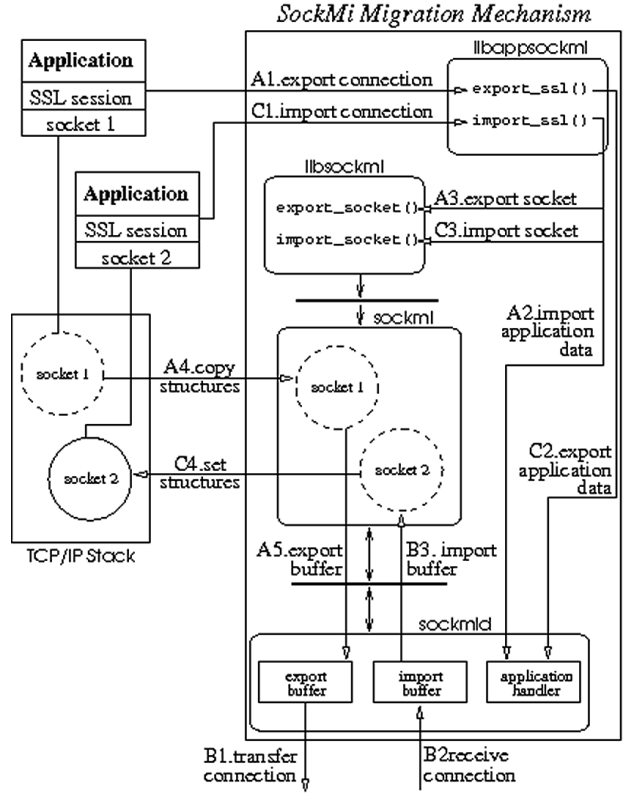


FIGURE 2: THE MECHANISM FOR THE MIGRATION OF AN SSL CONNECTION

The first phase is activated by the application that wants to export the secure connection. This phase can be summarized as follows:

A1. The exporting application calls the export_ssl() function.
A2. The libappsockmi library saves all required SSL information into a memory buffer. For this purpose, it uses the save_ssl() function. Then the library transfers the buffer to the daemon through an http message and the daemon saves the buffer in a local file that is transferred to the importing daemon during the negotiation phase (see B1 and B2).
A3. The library exports the associated socket using the export_socket() function.
A4. The module saves the state of the exporting socket into a memory buffer.
A5. Finally, the module exchanges information about the exporting socket with the daemon.

The negotiation phase can be summarized as follows:

B1/B2. The exporting daemon communicates with daemons running on other hosts in order to choose where to migrate the connection; the selected daemon receives all data about the exported secure connection.
B3. Then the importing daemon writes the state of the importing socket to the module internal buffers.

The last phase can be summarized as follows:

C1. The importing application calls the import_ssl() function specifying an array of requests.
C2. The libappsockmi library asks the daemon whether one matching connection is available. On success, the daemon's reply contains the data of the matching connection and the criteria used for importing the associated socket.
C3. The library imports the associated socket using the import_socket() function specifying the received criteria.
C4. Finally, the module restores the state of the importing socket into the kernel data structures.

SockMi is a mechanism, based on the cooperation of a kernel module and a daemon, that allows one to migrate an end of a secure connection to another Linux system running the same software. It is a complete solution for the migration of the network, transport, and application layers. It is compatible with OpenSSL version 0.9.7i and with Linux versions 2.4 and 2.6. The source code is available from http://sockmi.sourceforge.net.

Recently, we started to test the migration mechanism with real-world protocols and applications that use a secure connection (e.g., https, sftp, ssh). Another possible future activity is to add support for the authentication of the daemons that, in order to work in a potentially hostile environment, need to use secure channels for their communications. As to its porting to other UNIX-like operating systems, this depends mainly on the availability of the kernel source code. From this point of view, the porting looks possible, for instance, to systems belonging to the BSD family. OpenSSL is available on Windows systems, but at this time we have not yet analyzed if or how these systems can support the migration of TCP connections. This appears to be, in any case, a major effort since the implementation of sockets in Windows is significantly different with respect to UNIX-like operating systems.

REFERENCES

[1] M. Bernaschi, F. Casadei, and P. Tassotti, *SockMi: A solution for migrating TCP/IP connections*, 15th EUROMICRO International Conference on Parallel, Distributed and Network-Based Processing (PDP '07), pp. 221–228, 2007. Available from http://sockmi.sourceforge.net/public.html.

[2] T. Dierks and C. Allen, *The TLS Protocol*, RFC 2246, January 1999.

[3] http://openvpn.net.

[4] http://www.openssl.org.

[5] http://www.netfilter.org.

DAVID BLANK-EDELMAN

practical Perl tools: Peter Piper picked a peck of PDFs

David N. Blank-Edelman is the Director of Technology at the Northeastern University College of Computer and Information Science and the author of the O'Reilly book Perl for System Administration. He has spent the last 20+ years as a system/network administrator in large multi-platform environments, including Brandeis University, Cambridge Technology Group, and the MIT Media Laboratory. He was the program chair of the LISA 2005 conference and one of the LISA 2006 Invited Talks co-chairs.

dnb@ccs.neu.edu

THE ADOBE PORTABLE DOCUMENT Format (PDF) has become a lingua franca in the business and technology world. I'd hazard a guess that you probably read and perhaps generate one or several PDF documents a day as part of your daily routine. Documentation, invoices, electronic books, copies of presentations, and a whole bunch of other document types now commonly live in PDF format. Even though I write this column in plain text, when it gets typeset for this publication, a draft proof copy comes back to me in PDF format. I just ran a quick check, and I find that I have 2,678 PDF files on the laptop being used to write this column. That laptop (a Mac) treats PDF files as a "native" format and knows how to create and read them right out of the box.

I don't mind swimming in documents in this format because it is an "open standard." Adobe distributes the specifications for how PDF documents are constructed. Everyone is free to create programs that read and write this format, with Adobe's royalty-free blessing. Adobe announced in January of this year that they plan to submit the latest version of the PDF spec (1.7) to the International Organization for Standardization (ISO) so it can become a standard standard (vs. a non-standardized standard, I suppose. Paging Nick Stoughton . . .).

The PDF format may be open and ubiquitous, but I suspect I'm not alone in thinking about PDF files as a kind of black-box magic. My PDF-generating applications create PDF files, my PDF-reading applications display their contents, and that's close to the level of expertise I've desired on the subject. It is a little like the PostScript language. I've had to suss out enough PostScript to hand-edit recalcitrant PostScript files less than ten times in my life, and I didn't enjoy the process. (Remind me to tell you some day about the person I met who credibly claimed to have written an entire Web server in PostScript.)

With that level of technical apathy in mind, it will make sense that this is a column about creating and manipulating PDF files from Perl using higher-level interfaces. If you want to forge individual PDF XObjects yourself you'll need to use different Perl modules from those discussed here. If you want to know how to work with PDF files without

knowing too much about just how they represent their data, you've come to the right place.

Generating PDF Files from Perl

Let's start with nothing and see if we can wind up with something. There are a number of modules at the right level of abstraction for our purposes that can create new PDF files. Two of the popular packages are PDF::API2 and the Perl bindings to the commercial (with a more limited free version) PDFlib package. We'll take a really quick look at how to use both of them, starting with the free package.

PDF::API2 has an extensive list of PDF features, such as support for different font types and graphic formats. Unfortunately, this power comes with a little more pain than I'd prefer. The documentation assumes you already have some PDF experience and you are just searching for the module's methods to make use of your experience. A simple "Hello World!" looks like this (from the doc):

```
use PDF::API2;

$pdf = PDF::API2->new;

$fnt = $pdf->corefont('Helvetica-Bold');

$page = $pdf->page;
$page->mediabox('A4');

$gfx = $page->gfx;
$gfx->textlabel(200,700,$fnt,20,'Hello World !');

$pdf->saveas('/this/new/document.pdf');
$pdf->end;
```

Let's walk through this example line by line. After loading the module and creating a new PDF::API2 module, the first step is to request a font object. Think of it as a pointer to the font we will use later when we draw text. It is called a "core font" because the PDF standard blesses 14 fonts as "core fonts"; these are always available on any system. With this in place, we create a page object and set the MediaBox (i.e., the physical size) of that page. We then ask for a handle into the graphics content object of that page. Using this object, we can finally write some text onto the page. The text is placed at coordinates 200,700 (using the wacky PDF system of 0,0 being in the lower left of the page) and is rendered at a size of 20 points. The last two statements save the data out to the file and destroy the PDF object.

Whew . . . and that's a simple example. Just figuring out this little snippet of code can run you ragged if you are not familiar with the standard PDF nomenclature. For example, when I was first trying to understand what `$page->gfx` did I found I had to consult the source code in three separate submodules just to get the basic what, why, and wherefore for that line of code. The documentation is equally terse on other matters; for example, the `textlabel()` documentation lists its arguments, but it never says what the units for size should be (for that, I had to go track down the official PDF specification at http://www.adobe.com/devnet/pdf/pdf_reference.html). I'm not complaining as much as I'm warning you that you may be in for a bumpy ride with this module.

I'm not the only person who has noticed these shortcomings. There are several helper modules that provide a less daunting face for PDF::API2. For example, PDF::API2::Simple lets you write code like this:

```
use PDF::API2::Simple;
my $pdf = PDF::API2::Simple->new( file => 'output.pdf' );
$pdf->add_font('Helvetica-Bold');          # load the font
$pdf->add_page();                          # start a new page
$pdf->text('Hello World!',
           x => 200, y => 700,
           font => 'Helvetica-Bold', font_size => 20);
$pdf->save();
```

The other approach I'd recommend exploring when it comes to PDF creation is the use of the commercial package by the German company PDFlib GmbH (www.pdflib.com). PDFlib GmbH produces an exceptionally full-featured library for PDF creation and modification, with bindings for many different languages: Cobol, COM, C, C++, Java, .NET, Perl, PHP, Python, REALbasic, RPG, Ruby, and Tcl. Its library can basically handle anything you'd want to do relating to PDF files, including functions far beyond what PDF::API2 can handle. If you need to do heavy-duty PDF production programmatically, this is going to be a good bet.

PDFlib GmbH also provides a "Lite" version of their product that is free for noncommercial, personal, open source developer and research use. It is a considerably smaller subset of the commercial offerings, but it probably can do most of what the casual user needs. Let's take a quick peek at how to use PDFlib Lite from Perl.

There are two interfaces for this package we could consider using: the one that ships with PDFlib Lite and a wrapper module for it called PDFLib, which provides an object-oriented interface to it. The PDFLib wrapper module was last updated three years ago, so we're going to stick to the bundled version for this example. To help continue the comparison we've already started, let's look at a simple "Hello World!" example using this module as well (adapted from the example in the PDFlib Lite distribution):

```
use pdflib_pl;

my $pdf = PDF_new();

# ask each function to return -1 if there is an error
PDF_set_parameter( $pdf, "errorpolicy", "return" );

if ( PDF_begin_document( $pdf, 'hello.pdf', '' ) == -1 ) {
        die 'Unable to begin document: ' . PDF_get_errmsg($pdf) . "\n";
}

# 612 x 792 points is US letter-sized paper
PDF_begin_page_ext( $pdf, 612, 792, '' );

# load the font (in a particular encoding)
my $font = PDF_load_font( $pdf, "Helvetica-Bold", "winansi", "" );
if ( $font == -1 ) {
        die ' Unable to load font: ' . PDF_get_errmsg($pdf) . "\n";
}

# make it the current font
PDF_setfont( $pdf, $font, 20.0 );

# place and print the text on the page
PDF_set_text_pos( $pdf, 200, 700 );
PDF_show( $pdf, ' Hello World !' );

# finish the page
PDF_end_page_ext( $pdf, '' );

# finish the document
```

```
PDF_end_document( $pdf, '' );

# be nice and destroy the pdf object
PDF_delete($pdf);
```

I don't want to bore you with any more "Hello World" programs. We've seen the very basics of creating PDFs from scratch. We can get more complicated by importing images, drawing lines and shapes, and messing with text formatting and placement in a fairly straightforward way. Rather than going deeper into PDF creation, I want to switch topics now so we have enough space to cover the second activity people would like to use Perl for when dealing with PDFs.

Manipulating Existing PDF Files with Perl

Even if you don't need to create your own custom PDF files programmatically, you probably occasionally need to modify and manipulate existing files. For example, if you need to send someone the answer to a question found buried deep in the documentation, it may be better to send them just a few pages rather than the whole 800-page manual. Going in the opposite direction, you may want to concatenate several separate documents so you can send them as a single file to avoid confusion. It could be handy to extract all of the images or text from a PDF file to separate files. Perhaps you'd like to add a footer on all of the pages in an existing document with a message such as "Highly Confidential—Eat if Captured." All of these things and more are available to you courtesy of the right Perl modules.

Did I say "modules"? You could use separate modules (including one of the commercial PDFlib offerings) but there's actually an all-singing, all-dancing PDF manipulation module called CAM::PDF that can handle all of these tasks for you. Let's look at how to perform some of the tasks just mentioned using it. Before we go on, let me slake your curiosity by saying that the CAM:: in CAM::PDF comes from "Clotho Advanced Media," the company that originally developed the module.

Starting at the top of our wish list, to extract pages 1, 3, and 12 from a PDF file, we could use something like this:

```
use CAM::PDF;

my $pdf = CAM::PDF->new('pdf_reference.pdf');
$pdf->extractPages( 1, 3, 12 );
$pdf->cleanoutput('output.pdf');
```

Yes, it is that easy. We create an object that points to the input file, tell it to extract the right pages, and then write the document to a new file with cleanoutput() (as opposed to save(), which will append to the original file).

Appending two files is similarly easy:

```
use CAM::PDF;

my $pdf1 = CAM::PDF->new('pdf1.pdf');
my $pdf2 = CAM::PDF->new('pdf2.pdf');
$pdf1->appendPDF($pdf2);
$pdf1->cleanoutput('concat.pdf');
```

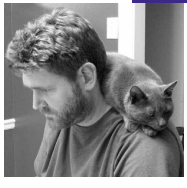
CAM::PDF comes with scripts to handle jpeg and text extraction and footer addition ("stamping"), so I won't include that code here. It comes with quite a few utility scripts like this, so it is worth your while to check out the package.

As a final postscript to this section, and as we fade into the sunset, I do want to mention that if CAM::PDF is not your cup of tea, the other module worth your consideration should be PDF::Reuse. PDF::Reuse's whole raison d'être was the desire to take an existing PDF file and use it as a template for the creation of other PDF files. For example, you could take a small PDF file with a picture of a business card and have it create a document with this card repeated in columns on the page for mass printing. Another possibility would be to send someone a customized PDF document with hyperlinks in the body that were personalized for the particular user. You probably can think of other ways this could come in handy. Both CAM::PDF and PDF::Reuse will serve you well in these cases.

I hope this column has demystified the process of PDF file creation and modification just enough so you can get what you want done without having to devote too much of your limited brain space to PDF minutiae. Take care, and I'll see you next time.

DAVID JOSEPHSEN

iVoyeur: a view from someplace nearby



David Josephsen is the author of *Building a Monitoring Infrastructure with Nagios* (Prentice Hall PTR, 2007) and is Senior Systems Engineer at DBG, Inc., where he maintains a gaggle of geographically dispersed server farms. He won LISA '04's Best Paper award for his co-authored work on spam mitigation, and he donates his spare time to the SourceMage GNU Linux Project.

dave-usenix@skeptech.org

GREETINGS. WELCOME TO *;LOGIN:'S* shiny new monitoring column. When Rik first approached me with the idea, I must admit my first thought was to wonder if there was enough subject matter to fill a semimonthly column for a reasonable length of time. Is systems monitoring really that deep? If you have any experience with the large enterprise-strength monitoring apps, then you know that vendors don't seem to think so; they view systems monitoring as a largely turnkey affair: Purchase license, install agent, reboot server, repeat.

Even the corporate-backed open source upstarts seem to share this opinion to a certain degree [1]. While the Patrols and OpenViews of the world clamor to support the largest number of gadgets, the Hyperics and Zenosses appear to be differentiating themselves based on their auto-discovery tools and ease of configuration. If the vendor claims of "zero to monitoring solution in 30 minutes" are to be believed, then a monitoring column might not be a particularly entertaining prospect for you.

But as a good friend of mine once (quite rightly) said, "Knowing that there is a Web server on port 8080 is about 2% of the problem." Systems monitoring, it turns out, is anything but a turnkey affair. Just behind the shiny facade of port scanners and SNMP traps lies a stunningly complex problem, involving a question, the answer to which is unique for each person who asks it. It is a problem in fact that I think we have yet as a community to fully understand, much less actually solve.

Consider for a moment what happens when you type a URL in your browser and get back an error page. At that moment, the actual status of the Web "service" in question is a quantum superposition; it is, to you, in a Schroedinger's state. You have observed an error page, but that isn't necessarily indicative of a problem with the Web site itself. There are a great many things that could be wrong that have nothing whatsoever to do with the Web server. The blame might rest with your system's network connection, DNS, an unfriendly filter, or a mistyped "ip route" command by some sleepy admin somewhere in the worldwide mass of interconnected routers between you and the Web page you seek. Some of these you can test for, and some are more difficult to detect. The Web site is up *and* it is

down. There is an objective reality—a singular state—but for the moment it eludes you. You'll have to tease it out.

Teasing things out, however, is a talent your monitoring system doesn't possess. It checks exactly the parameters you tell it to check, and it returns the result. If you called the parameter "Web service," then that's what the monitoring system will tell you is down, and if you aren't careful about choosing the parameters, it might even tell you everything is fine in the presence of a problem—an even more distressing proposition. If only knowing the state of the cat were as simple as opening the box. Arguably, the pinnacle of our error detection capability at this point is end-to-end monitoring, involving scripts that mimic user behavior, thereby encountering the same problems a user would. But end-to-end monitoring programs are somewhat of a copout, because they don't actually give you the state of the cat either. They tell you that there is a problem (from the perspective of the monitoring system), but not where the problem might actually reside. Their real intent is to catch errors that more specific checks such as port scanners might not. Monitoring systems, it seems, are not (yet) capable of making the observations necessary to solve our quantum conundrum.

So you can call this notification from your monitoring system a "Web outage" on the reporting interface if you like, but that doesn't make it true. Like the demanding helplessness of the user crying, "The interwebs are broken," there's information there, but not very much, and it's of questionable accuracy. Perhaps knowing where the problem lay is not critical to you; it's enough to know that there is a problem, and you'll take it from there. But perhaps automated site-to-site failover depends on bulletproof detection of a specific error or set of errors, or maybe the problem is chronic and requires a human to detect patterns in the service availability over time (false alarms make pattern hunting a bit more difficult). Either way, the monitoring system probably hasn't actually answered the question it was intended to answer, and many of the humans using the system won't be aware of the distinction. In systems monitoring, the area where the humans and system meet is especially problematic.

Really the core of the monitoring problem is that we've created ourselves some rather untrustworthy machines. There's just an awful lot of places where things can go bad, and for all of our fancy packet pitching, today's PCs are very much islands unto themselves, barely aware of their own state, much less that of the network around them. We, like an unfortunate mix between detective and geologist, rely mostly on forensics to gain what insights we can, using netflow, syslog, utilization graphs, and monitoring tools. And being every bit as untrustworthy as the systems they are trying to monitor, the monitoring box itself can have all of the same problems. In the end all it can give you is its own crudely gleaned opinion of the current state of a set of services from a single static point in the network, which is often a poor substitute for knowing the service state firsthand.

So asking one fallible machine in a fallible network its opinion about the fallible machines surrounding it might not be so great an idea. Doing so is not unlike paying a guy $5 to watch your car at 2:30 a.m. in Tijuana. (Usually it works out fine, but that doesn't make it a good idea.) And speaking of misplaced faith in humanity, the humans in this equation are equally as fallible as the machines (if not more so). For one thing, in classic, failure-to-quantify-the-risk fashion, we sysadmins and our managers seem to place an unfounded amount of trust in our monitoring systems. As if calling them "monitoring systems" somehow imbues them with a magical immunity from mistaking a DNS failure for a Web site outage, or even just crashing outright.

But alas, our monitoring tools betray us. They crash like normal systems and are largely dependent upon the same network infrastructure as the other systems. And yet for some reason the false positives surprise us as much as false negatives; it "feels" like this sort of thing shouldn't happen to the monitoring system. It seems ironic, when there's no real reason it should. It's telling even that I used the word "betray." So there is an emotional component here, and its most common effect is to cause us to ignore a monitoring system that has proven itself to be unduly chatty, or sometimes incorrect. We don't "lose faith" in Tomcat when it runs out of threads and starts handing out 500s, but for some reason we are quick to anthropomorphize and discredit a monitoring server for its digressions, even though it may be the worst possible server to take with a grain of salt.

With "normal" systems—the ones without the magical "monitoring system" moniker—we mitigate the risk of failure with redundancy, incorporating load balancers, VRRP, and BGP multihoming; redundancy is an industry unto itself, and it could certainly help out in a monitoring context. It's not uncommon for a large organization to have a failover monitoring box, and large installs sometimes require numerous monitoring systems to aggregate alerts to a master in order to scale, but these setups don't improve the resolution of our failure detection ability.

Parallel systems have potential in this regard; two opinions are better than one. Yet curiously, monitoring systems are seldom deployed this way. (If you have one, I'd like to hear about it.) This might be because having parallel monitoring systems agree on a given service state is a difficult problem to solve, which in itself is a decent proof of the fallibility I just alluded to. What do you do when two systems disagree? Further, avoiding things such as redundant notifications requires that the monitoring systems be somewhat aware of each other's opinion of the current state of things, making prospects even hairier.

Leslie Lamport is intimately familiar with getting parallel computational entities to agree on states. His work on the Byzantine Generals [2] problem is used widely at NASA and the aerospace industry to design fault-tolerant flight-control systems. His work, and the work of those at SRI, showed that $3n + 1$ processors are generally required to tolerate n faults. In layman's terms and warped to suit our needs, the opinions of 4 monitoring systems would be needed to reach a trustworthy agreement on a given service if one of them were malfunctioning. I don't have any fancy math to back this up, but it "feels" like the odds of 1 out of 4 PC-based monitoring systems misbehaving at any given moment are good. So monitoring systems, it seems, may need to be a bit more redundant than we're used to before they can begin to give really meaningful opinions. We can't simply toss another box in without making things a lot more complicated, and it turns out we'd need to throw in quite a few before seeing a real return on the investment.

None of this is to say that systems monitoring is impossible or hopelessly broken. In practice monitoring tools usually work pretty darn well, and they are certainly better than nothing at all. My monitoring systems have saved my gravy more times than I can remember. But it's useful, I think, to imagine a reference system, some inexpensive, Byzantine failure–proof, massively parallel monitoring system communicating securely via out-of-band channels and telling us with flawless accuracy and resolution about specific problems and their causes without burdening the network with traffic or the systems with bulky agents. It introduces no security flaws, has an infinite amount of trending and utilization data on every metric we can imagine on every server, has a network device in the environment, and can do complex event correlation and aberrant behavior detection in real time. Maybe it has

some of those heuristics and biological diversity I'm always reading about, and what the heck, it runs Plan9, and doubles as a margarita machine. This makes it easier to imagine the huge space of gray between the reference system and the system you probably have in your shop today. That enormous gray space is what the vendors are ignoring when they say, "0 to monitoring solution in 30 minutes."

So, needless to say, I happily took Rik up on his offer. In this column, I want to explore the gray space, providing practical solutions, advice, code, and general food for thought. My sincere hope is that perhaps somewhere along the way we'll both gain a better understanding of the problem, and maybe move a few gradients closer to the monitoring system of our dreams. Expect topics to range from network architecture to SNMP to security to data visualization to temperature sensors to dealing with humans and back again, running the gamut of what you as a sysadmin might run into in the course of implementing and maintaining a monitoring system.

To a large extent the information I provide will be specific to Nagios [3], which is probably the most nearly ubiquitous open source monitoring program today. This is not, however, a column about Nagios. I would prefer that you think of Nagios as a reference implementation language rather than as a design requirement. If systems monitoring has an XML-like means of specifying solutions, a prototyping language that is relatively easily translated between disparate systems, then Nagios, with its (almost painfully) open architecture and liberal lack of design assumptions, is probably the closest thing I've seen to it. So my use of Nagios in this column is only to ensure that the solutions discussed herein have a good chance of being translated to whatever you happen to use (and if that's Nagios, then all the better).

Feel free to shoot me email [4] or comment on my blog [5] if you would like to talk about something specific or just want to say hi. And finally, believe it or not, I honestly plan to maintain a better signal-to-noise ratio in my future articles, so sorry for the theoretical ramble. I promise to have some nitty-gritty for you in the next issue. Take it easy.

REFERENCES

[1] http://books.slashdot.org/comments.pl?sid=230333&cid=18695063.

[2] http://research.microsoft.com/users/lamport/pubs/pubs.html#byz.

[3] http://www.nagios.org.

[4] dave-usenix@skeptech.org.

[5] http://www.skeptech.org.

HEISON CHAK

deviating alternatives to Asterisk

Heison Chak is a system and network administrator at SOMA Networks. He focuses on network management and performance analysis of data and voice networks. Heison has been an active member of the Asterisk community since 2003.

heison@chak.ca

SINCE THE RELEASE OF 1.2 IN 2005 and 1.4 to follow in 2006, the Asterisk open source project has gained momentum and critical mass. With the increasing number of deployments and user base, stability and scalability are becoming critical concerns. Although Digium is standing behind Asterisk, maintaining stability in such a large-scale project is easier said than done.

Major cleanup and code audit procedures have been put in place. As a result, more efficient codes and channel drivers enhance system compatibility and stability. All latest Digium (e.g., PRI and analog FXO) interfaces also feature hardware echo cancellers to help reduce host CPU utilization. These are examples of commitments from Digium to ensure stability of the Asterisk PBX software.

Despite community effort and dedication from developers to make Asterisk a better system, many are still looking for controversial changes. Although they do not provide the same functionality and some are not even designed to be a PBX, here are a few deviating alternatives to Asterisk:

- SIP Express Router (SER)
- Broadsoft
- FreeSWITCH
- sipX
- OpenPBX

By and large, these alternatives were developed by those who started doing research on Asterisk and quickly realized that it wasn't what they were looking for.

SIP Express Router

Asterisk can be configured as a SIP registrar, acting as an endpoint user agent to the originating call leg and then creating a new call to the receiving phone—thus staying in the middle of the call. A real proxy, such as SER, is never the endpoint of a call, handles call control on behalf of user agents, and does not maintain state during a call. SER supports SIP connections with more features (it can be a registrar, proxy, or redirect server) and has better scalability.

It is quite common to use SER in conjunction with Asterisk, especially when someone is looking for PBX functionality and has a multitude of networks to traverse. Remember, SIP doesn't play nicely with NAT. Using a proxy can lift some of the burden.

Broadsoft

With Asterisk Realtime, some of the configuration management can be offloaded to the backend database, and alteration to config files will not require a reload of Asterisk. This may be a step ahead, but it is nowhere near the big leap Broadsoft has put together: Its carrier-grade soft-switch platform provides a homogeneous view of configuration steps. From provisioning SIP accounts to assigning numbers to call features (e.g., call waiting, voicemail) on accounts, whether they are administrative tasks or user-driven options, the configuration interface is quite similar. This is a big benefit to carriers, as they can use the same interface day in and day out.

The soft-switch runs under Solaris and is usually installed and supported by the vendor. The software can be provisioned to support administrative view (for provisioning SIP accounts or assigning numbers) and customer view (for end users to manage their subscribed features).

FreeSWITCH

FreeSWITCH is an open-source soft-switch started by a bunch of Asterisk developers who respect many Asterisk architecture decisions but disagreed with the following:

- Monolithic architecture—processing, user interface, and data all residing within the same entity
- Limited support of UNIX flavors and GCC
- Limited support of 8-kHz audio sampling

FreeSWITCH has been available since January 2006 and supports a wide range of protocols, including SIP, IAX, H.323, and Jingle (GoogleTalk). Although it has IVR capability, it is most beneficial to carrier-grade applications and is not really a PBX system.

Asterisk IP PBX

Despite the richness of features in Asterisk, many applications have undergone redesign to cope with user demands and community feedback. Let us find out where some of these deficiencies may be.

MeetMe is well known for its ability to take over traditional conference servers based on TDM (time-division multiplexing). Because of the nature of VoIP, it can easily support a larger number of participants; the number of legs on a teleconference is limited by CPU, memory, and bandwidth rather than the number of licensed ports, as on legacy systems.

However, MeetMe relies on Zaptel devices as a timing source. Upon a closer look at this dependency, one sees that it is actually the Zaptel driver that is responsible for mixing audio in its buffers, and it does so for PRI or analog legs. When an IP-based audio stream (e.g., RTP or IAX) is added to the mix, MeetMe creates a proxy pseudo device and handles the conversion of reading RTP or IAX frames and writing out to the pseudo device as well as reading from the pseudo device followed by writing out RTP or IAX frames. In the absence of a Zaptel device, a software-based dummy driver (ztdummy) generates timing from the usb-uhci kernel module or uses the internal high-resolution kernel timer in Linux 2.4 and 2.6, respectively. Such dependency adds complexity to normal operations and an additional burden to upgrades.

Voicemail

Users generally praise Asterisk's ability to deliver voicemail as email attachments. However, many may soon realize that deleted voicemail attachments reappear in their voicemail box when they dial into the system—there was no easy way to synchronize email deletion with the actual voicemail storage. Some have implemented periodic removal of voicemail messages: After an email attachment has been sent, voicemail storage is swept for messages older than a predefined age. Others have implemented hooks in their IMAP storage to allow voicemail messages to be sent and to allow IMAP clients to delete messages as well as for Asterisk to remove voicemail if so requested by users calling from their phones.

The latest development provides support for IMAP voicemail storage in Asterisk, so that users can manage voicemail messages in a synchronized fashion with either their phone or their favorite mail user agent.

Fax and Asterisk

We love HylaFAX for what it does, and even if there are less complicated ways to manipulate faxes in Asterisk, many are still sticking with HylaFAX. HylaFAX is a software fax machine that communicates with fax modems and has been around for quite some time. SpanDSP is a library for digital signal processing. Two Asterisk applications, app_rxfax and app_txfax, use SpanDSP to send and receive faxes. When it works, it is very efficient: It receives the fax as a tiff image, which can be converted into PDF and emailed to a user with just a few lines of code in the dialplan. However, because it is not distributed with Asterisk, it requires some patching, and whenever there is a major upgrade (Asterisk or Zaptel), the application is prone to failure.

Alternatives

SIPfoundry's sipX is a SIP-based IP PBX. Because of its SIP proxy capability, it is believed that sipX can scale better. As long as the PBX system only has SIP user agents and all of the IP voice trunks are delivered via SIP, sipX may be a viable option. Also note that sipX does not support fax communication.

Another alternative is OpenPBX, which is a fork of Asterisk 1.2. It maintained some of the features and applications that were deprecated in Asterisk 1.4 and has significantly improved the build process and environment as compared to 1.2. However, community support of OpenPBX is nowhere near that of Asterisk. On a recent SIP exploit that alerted Asterisk and other SIP vendors to patch their systems, OpenPBX revealed its weakness in responding to the incident. Only time will tell whether this fork will survive.

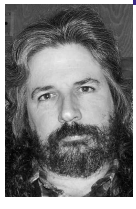
A number of service providers running Asterisk 1.2 are holding off on upgrading to 1.4 and awaiting Asterisk 1.6's arrival, due by the end of this year. Other have moved away from Asterisk totally and onto FreeSWITCH or Broadsoft.

REFERENCE

http://www.asteriskguru.com/downloads/asterisk_stability_and_security.ppt.

ROBERT G. FERRELL

/dev/random



Robert G. Ferrell is a chronically underemployed information security geek who enjoys surfing (the Internet), sashimi (it makes great bait), and long walks (to the coffee machine and back).

rgferrell@gmail.com

AS I TYPE THIS, I'M SITTING IN AN information security conference in Dallas. Of the two hundred or so in the auditorium where the opening remarks are being delivered, I am the only person wearing a hat. I find this odd. This is, after all, Texas, and according to the attendee list most of these folks are from some corner of the Lone Star state as well. I happen to be sporting a tobacco-brown leather "outlaw" cowboy hat with brass studs on it, which, like myself, is admittedly a bit over the top, but I'm talking about *no* other headgear in evidence at all. A sea of thinning hair, highly polished skin, and hair-simulating appliances stretches from wall to wall, a stark chiaroscuro from some febrile milliner's nightmare.

I've lived here in cowpoke country all my life, with only a few brief sojourns into the uncivilized wilds of not Texas, and never have I witnessed such a paucity of head coverings as in the past few years. Perhaps it's the influx of immigrants from the less hat-aware regions, or mayhap the threat of an explosive device nestled covertly on that sweaty bald spot is simply too great a risk in a country that spends the better part of each day jumping hysterically at the approach of imagined boogeymen. It has become abundantly clear that our national paranoia has reached pathological levels when the blinking lights on children's toys trigger EOC activations. Even sacred hat-wearing traditions are not safe when collective common sense is on protracted holiday.

I'm going to swing this literary oil tanker around now and talk about certifications, the alphabet soup for the career, as it were. There have been way too many words written on the subject of professional certs, and as I am loathe to miss my chance to chunk some verbiage on this steaming pile, I bloody well shan't.

Certifications are perfectly appropriate, in my opinion (and I'm the one with a column here, so that's a surprisingly salient point), for selected critical professions. Doctors and airline pilots immediately spring to mind. I doubt that anyone really wants drug prescriptions from, body parts modified/removed by, or themselves at 35,000 feet in the hands of, a person who may or may not have any idea what she/he is doing. Medical and air trans-

port licensure certifications take a lot of study, skill, and time to achieve, and generally they denote that the person in question is at least nominally competent to perform the job. That's what certifications were originally intended to convey: evidence that a body of knowledge had been mastered, that a degree of manual and/or intellectual skill had been demonstrated, and that the certifying body was reasonably confident this person wouldn't kill anyone unintentionally.

Then, gradually, insidiously, *certmania* crept in. I suppose at first there was a general, if unrealistic, faith that the certification process would be made sufficiently rigorous to warrant some degree of confidence being accorded to graduates thereof, but it didn't take too long to prove that trust unfounded. Certification, like virtually everything else in modern society, is subject to market forces and therefore to a chimeric form of circular argument I like to call "suborbital logic" because it never really makes it all the way around. Allow me to illustrate.

Let's say a bunch of industry folks complain that the lack of widespread enforced standards applicable to people in their profession generates bad public relations for the whole group, despite that fact that until this bellyaching hits the mainstream media the public is barely even aware of said profession. These concerned professionals decide to form a working group/intellectual black hole to come up with educational and functional guidelines for ensuring minimal competency among their current and future cohorts. A niche is born, and a flock of circling corporate predators descends to feast upon the neonate. They've evolved, after all, to smell this sort of nascent opportunity from miles away.

Quicker than you can say, "Michael DeBakey," vendors with visions of conspicuous consumption dancing in their heads bypass the ponderous industry working group and come up with their own certifications, designed more to pad the coffers of said corporations than, say, provide any meaningful measure of professional competency. Immediately thereafter, hideously overpriced and underinformative training classes spring up like weeds in a fallow pasture, dotting the landscape with false promises and shattered expectations. The offerings expand exponentially until the carrying capacity of the econosystem is exceeded and the pack must feed on itself to bring things to equilibrium.

After the blood tide recedes, the surviving certs take on an air of seeming legitimacy that grows with each hapless recruit who swells their turgid ranks. Corporate HR managers start to take note, cautiously listing the certs in the "desired skills" area of job announcements. Eventually, if enough people get certified and enough articles are written extolling the virtues of a given cert, it creeps into the "mandatory skills" list and at that point occupies a permanent seat on the resumé lingo security council.

Once the disease reaches the tertiary phase, the mere presence of the appropriate acronym on a job application ensures that it will at minimum leapfrog the initial screening stage. Conversely, the absence of said alphabetic sputum will virtually guarantee a precipitous plunge into the round file. Many of these certs will now begin, Ouroboros-like, to require experience that companies are no longer willing to provide to applicants who don't already possess the cert. Thus is the cycle complete.

So what, you may well ask, if these certs are provided on a for-obscene-profit basis? What does that matter if the end result is a better-qualified work force? Not much, if that were true. The fact of the matter is that many of the more widely accepted certs require little more than passing a multiple-guess

test and agreeing to a code of ethics, the enforcement of which is nebulous at best. They're basically the more expensive equivalent of the cool diplomas you could order from the backs of comic books when I was a kid.

I'll leave you with this simple question: Would you trust your gall bladder surgery to a guy who took a five-day training class and passed a multiple-choice exam on the third try? Ima Quaque, CNORP (Certified Nonessential Organ Removal Professional).

Man, I hope the bar is open back at my hotel.

NICK STOUGHTON

an update on standards

REVISION FEVER

USENIX Standards Liaison

nick@usenix.org

IT SEEMS AS IF, SUDDENLY, SEVERAL of the big IT standards we care about are simultaneously being revised. POSIX started its revision two years ago and is nearly complete. C++ started (officially) this year. And now C is also talking about a revision.

All ISO standards go through a periodic maintenance requirement, in which, every five years, a standard must be re-affirmed, revised, or withdrawn. Of the three standards I mentioned, all are close to such a decision point.

I've written about the POSIX revision recently, so I'll simply mention that the Austin Group, the working group that maintains POSIX, is on track to complete its revision early in 2008. The third draft of the revised standard has just been published and is currently in ballot.

I'll devote a separate article to the C++ revision later in the year. The newest project is the revision of the C language, expected to be complete sometime in 2010 or later, and hence has been dubbed "C1x."

The C language hasn't changed all that much since its inception. Most programs written in the 1970s to Dennis Ritchie's original specification will still compile and run under a modern C compiler conforming to the ISO-C 1999 standard. True, there have been a few new keywords and concepts added to the language, but by and large it still holds true to its original intent. The most significant overhaul of the language came with that 1999 revision.

At the time, the committee put together a charter for the work they were about to undertake. This same charter is currently being reexamined to see whether it needs any changes in guiding us through the "C1x" revision. Since I wasn't on the committee for the 1999 revision, I found this document very insightful, and I believe the core principles are worthy of repetition here.

C's Principles (from the C9X Charter [1])

Before embarking on a revision of the C Standard, it is useful to reflect on the charter of the original drafting committee. According to the original Rationale Document in the section entitled "Purpose":

> The work of the Committee was in large part a balancing act. The Committee has tried to improve portability while retaining the definition of certain features of C as machine-dependent. It attempted to incorporate valuable new ideas without disrupting the basic struc-

ture and fabric of the language. It tried to develop a clear and consistent language without invalidating existing programs. All of the goals were important and each decision was weighed in the light of sometimes contradictory requirements in an attempt to reach a workable compromise.

In specifying a standard language, the Committee used several guiding principles, the most important of which are:

1. Existing code is important; existing implementations are not. A large body of C code exists of considerable commercial value. Every attempt has been made to ensure that the bulk of this code will be acceptable to any implementation conforming to the Standard. The Committee did not want to force most programmers to modify their C programs just to have them accepted by a conforming translator.

On the other hand, no one implementation was held up as the exemplar by which to define C: It is assumed that all existing implementations must change somewhat to conform to the Standard.

2. C code can be portable. Although the C language was originally born with the UNIX operating system on the DEC PDP-11, it has since been implemented on a wide variety of computers and operating systems. It has also seen considerable use in cross-compilation of code for embedded systems to be executed in a free-standing environment. The Committee has attempted to specify the language and the library to be as widely implementable as possible, while recognizing that a system must meet certain minimum criteria to be considered a viable host or target for the language.
3. C code can be nonportable. Although the Committee strove to give programmers the opportunity to write truly portable programs, it did not want to force programmers into writing portably, to preclude the use of C as a "high-level assembler"; the ability to write machine-specific code is one of the strengths of C. It is this principle that largely motivates drawing the distinction between a strictly conforming program and a conforming program.
4. Avoid "quiet changes." Any change to widespread practice altering the meaning of existing code causes problems. Changes that cause code to be so ill-formed as to require diagnostic messages are at least easy to detect. As much as seemed possible, consistent with its other goals, the Committee has avoided changes that quietly alter one valid program to another with different semantics that cause a working program to work differently without notice. In important places where this principle is violated, the Rationale points out a quiet change.
5. A standard is a treaty between implementer and programmer. Some numerical limits have been added to the Standard to give both implementers and programmers a better understanding of what must be provided by an implementation and of what can be expected and depended upon to exist. These limits are presented as minimum maxima (i.e., lower limits placed on the values of upper limits specified by an implementation) with the understanding that any implementer is at liberty to provide higher limits than the Standard mandates. Any program that takes advantage of these more tolerant limits is not strictly conforming, however, since other implementations are at liberty to enforce the mandated limits.
6. Keep the spirit of C. The Committee kept as a major goal to preserve the traditional spirit of C. There are many facets of the spirit of C, but

the essence is a community sentiment of the underlying principles upon which the C language is based. Some of the facets of the spirit of C can be summarized in phrases like

(a) Trust the programmer.
(b) Don't prevent the programmer from doing what needs to be done.
(c) Keep the language small and simple.
(d) Provide only one way to do an operation.
(e) Make it fast, even if it is not guaranteed to be portable.

The last proverb needs a little explanation. The potential for efficient code generation is one of the most important strengths of C. To help ensure that no code explosion occurs for what appears to be a very simple operation, many operations are defined by how the target machine's hardware does it rather than by a general abstract rule. An example of this willingness to live with what the machine does can be seen in the rules that govern the widening of char objects for use in expressions: Whether the values of char objects widen to signed or unsigned quantities typically depends on which byte operation is more efficient on the target machine.

One of the goals of the Committee was to avoid interfering with the ability of translators to generate compact, efficient code. In several cases the Committee has introduced features to improve the possible efficiency of the generated code; for instance, floating point operations may be performed in single-precision if both operands are float rather than double.

Goals for C1x

But why change C at all? Isn't it good enough as it stands? And wasn't the last revision somewhat of a failure? Most compilers still aren't fully conforming.

The short answer is that the current standard is not quite good enough for today's hardware and is just bad enough to need tinkering with. As hardware gets more and more complex and as multicore processors become the normal minimum, applications need additional promises from the language as to what is happening at the hardware level to be sure that they run correctly. Most modern compilers have added their own extensions to the C language to give programmers control over things such as alignment, atomic memory access, and the like. One of the new principles will be based on existing practice: There will be a high bar to get a new feature into the language if something like it is not already in a commercially shipping implementation (not some experimental system released yesterday to some of my close friends).

The Committee also agreed that mistakes were made in the 1999 revision, and we should learn from them. One of the bigger mistakes was thinking that Fortran was the competition, and trying to add the kitchen sink with respect to things such as complex arithmetic. The Committee needs to learn from the mistakes of the past. The major goals for this revision are in the areas of security, parallelism, dynamic libraries, vendor-specific additions, extended character sets, and embedded systems.

Concurrency or parallelism is one of the main motivators. With multicore processors and widespread use of the POSIX pthread model in general acceptance (not to mention some other threading models that are also popular), the language needs to describe a more comprehensive memory model. There was some discussion of this in the C++ revision context in the February 2007 edition of *;login:*.

At the April 2007 meeting, the ISO-C committee looked at a number of the extensions provided by gcc and has agreed so far that papers seeking to standardize any of the following will be looked on favorably:

- Statements and declarations in expressions
- Locally declared labels
- Referring to a type with typeof
- Inquiring on alignment of types or variables
- Thread local storage
- Specifying attributes of functions
- Specifying attributes of variables
- Specifying attributes of types

This list does not preclude other items from being considered; it simply describes the "we really want these features" list so far.

Security was generally regarded as another important aspect. Bad programmers can easily write applications that contain vulnerabilities, although good programmers can write very secure code in C. This isn't a problem with the language as such, but the Committee feels that there should be an increased focus on the security aspects of the language. Among other things, this will mean that the infamous gets() function will finally be removed from the standard!

When considering security, there is an interesting distinction between, as one Committee member described it, "Murphy and Machiavelli." There are at least two distinct classes of C user: those who are writing low-level code, such as an operating system kernel or system library, and those who are writing higher-level applications. The low-level programmers want to get every ounce of performance out of the system; they need to write defensive code but have little need for new "security" features that may result in slower code. They may need Machiavellian techniques to achieve their goals. In contrast, the second class of developer wants every bit of help the system can give! There are concerns about both Murphy and Machiavelli for these people, and although the principle of "trust the programmer" still holds, there is also the possibility of the programmer saying, "I don't trust myself; please check me!"

It's still early days in this process, and no specific proposals for security enhancements have surfaced. If you think you may have something to contribute to this revision, please feel free to contact me to discuss how to make a proposal.

REFERENCE

[1] http://www.open-std.org/jtc1/sc22/wg14/www/charter.

book reviews

ELIZABETH ZWICKY WITH SAM STOVER, AND RIK FARROW

THE COMPLETE APRIL FOOL'S RFCS

Thomas A. Limoncelli and Peter H. Salus, compilers

Peer-to-Peer Communications, LLC, 2007. 390 pages.

ISBN 978-1-57398-042-5

This is one of those books that don't need a review so much as a description. For instance, at home I have a Richard Scarry alphabet book with flaps. It doesn't need reviewing. You either are a toddler, and understand that this is unbelievably perfect, or you aren't, and you don't. Similarly, what we have here is all the RFCs released for April Fools' Day, up through now. Some of you are giggling pleasantly at the thought, and you will enjoy the book. Some of you are intrigued at the idea, and you will enjoy the book, too. Some of you have no idea what an RFC is, and the odds are pretty good you won't find them funny. (If you have no idea what April Fools' day is, it's a day when people make jokes, traditionally in the form of fake news items. For instance, some day you should check out how Google Maps recommends you drive from, say, Albuquerque, New Mexico, to Toulouse, France. Some maps might recommend just not doing this, because there is an ocean in the way. But last April, Google quietly developed some new suggestions.)

On April 1, the otherwise generally sober IETF issues an RFC that is not completely serious. Some of these are very, very famous—the Avian Transport Protocol, for instance: Tired of good old UTP? ATP will take you further, but there are significant latency and packet loss issues because it uses pigeons. Some of them are less well known, although I have actually cited HTCPCP (the HyperText Coffee Pot Control Protocol) before. And one of them, actually one of the funniest ones, is not a joke at all (and was released in November). However, the need to document the NULL encryption protocol left people with a certain space for hijinks.

Frankly, you know you're a geek when you find the NULL encryption protocol documentation funny. I laughed out loud, and immediately felt a bit guilty. However, I also know that many of the other people who visit my house also find this sort of thing funny (since I have been known to read particularly bad bits of review copies of books at dinner parties), which makes this a perfect coffee table book for my sort of household. Your less technical visitors are going to find it utterly baffling. Whether this is a good thing or a bad thing is really your call.

A+, NETWORK+, SECURITY+ EXAMS IN A NUTSHELL: A DESKTOP QUICK REFERENCE

Pawan K. Bhardwaj

O'Reilly and Associates, 2007. 744 pages.

ISBN 978-0-596-52824-9

HEAD FIRST PMP: A LEARNER'S COMPANION TO PASSING THE PROJECT MANAGEMENT PROFESSIONAL EXAM

Jennifer Greene and Andrew Stellman

O'Reilly and Associates, 2007. 644 pages.

ISBN 978-0-596-10234-0

This is not a full review of the *A+, Network+, Security+ Exams in a Nutshell* book, which I did not read in its entirety. It is included here because it makes an interesting contrast to *Head First PMP*.

These two books are both specifically about exams, but they take extremely different approaches. It starts with the subtitles. *A+, Network+, Security+ Exams in a Nutshell* is in fact just what its subtitle says: a desktop reference, something you pick up to look up a specific fact, and then put down again. *Head First PMP*, by contrast, is companionable. It is meant to be read in its entirety, and to actively assist you in passing the exam.

Both books have sample questions, but *A+* prints the question immediately followed by the answer. It is effectively impossible to use these as review questions, because most readers will have noticed the answer by the time they've finished reading the question. *Head First PMP* prints all the questions relating to a chapter, has a page break, and then shows all the answers, so that it's easy to actually try to answer the questions without interference. In addition, *A+* gives the correct answer and then a brief restatement of why that answer is correct. *Head First PMP* gives the correct answer and then an explanation of how you should have gotten to that answer given the question and the other answers. (It will point out distractor information in the question, show you how you could eliminate other answers, and so on.)

Both books have exercises you can do to prepare for the test. *Head First PMP* includes these exercis-

es—things such as writing new exam questions, doing crossword puzzles, and various sorts of other questions. *A+* suggests that you go out and get test computers and then try things, except when that's not plausible, in which case the author suggests asking your local administrator. Some of the exercises strike me as merely unhelpful with a multiple-choice test, some of them are pointless (you're supposed to pick up a laser-printed page and note that it's warm, for instance), and some of them are likely to try your administrator's patience well beyond breaking point—"Contact the system or network administrator. Determine which networking protocol is used in the network and why."

Both books talk explicitly about the exams. *A+* provides tips direct from the testing organization ("Read the questions slowly and carefully"), whereas *Head First PMP* talks about question design, suggests a good place in the testing procedure to write down the formulas you'll need, and often delves into peculiarities of the test.

Both books have mistakes, too. In *Head First PMP*, I caught several bloopers in test questions—probably two or three places where it would say something like "The right answer is B, because . . ." and give an explanation that matched answer D, instead. This sort of error is fairly benign. You get confused for a moment, but it sorts itself out if the text is good. I'd prefer perfection, but this is well within my expectations for a first printing. I didn't catch any of these errors in *A+*, probably because I didn't read all the questions, but I did catch a couple of content errors. Take the description of Mac OS X:

"The MAC OS X is used primarily on Apple Macintosh computers. Apple has recently released the Intel version of MAC OS X that can be installed in place of Windows on Intel-based personal computers. . . . The applications that run on Apple computers running the MAC OS need to be written specifically for the MAC OS platform. This is due to the fact that the technology behind microprocessors used in Apple computers is entirely different from the technology used in Intel microprocessors."

Confused? Well, even if you know nothing about OS X, you should be, since the passage contradicts itself. It is also factually wrong and conceptually wrong. No, OS X can't be installed in place of Windows; yeah, all right, you can do some fiddling and install an operating system with some code overlap, but don't try to run the OS X installer. And conceptually, it would be mighty handy to actually understand executable compatibility, otherwise known as why you can install Windows and your favorite UNIX-derived operating system on the very same hardware and still not be able to run the same programs on both.

Both books are faced with the peculiarities of the exams they are about. Exams don't match to the real world perfectly, and these exams have some pretty spectacular divergences from day-to-day life. For instance, I work in a small startup. Let's just say that our project lifecycle does not involve 44 well-defined processes. In fact, I have worked as a consultant for a large corporation with PMPs as project managers, and we still didn't use the full process that the PMP exam talks about. The PMP exam also uses some specialized terminology, sometimes using a term more broadly or more narrowly than people in the industry normally do. *Head First PMP* is up front about this. It says straight out that not all organizations use all parts of the process and that you need to learn them all just in case, and it notes when terms are used unexpectedly. The *A+* book takes the exam's point of view and behaves as if the exam maps well to normal usage, which leads to statements such as "A [peer-to-peer] network is also known as a workgroup. . . . These networks are suitable for only about 10 computers. . . . A network operating system (NOS) does not need to be installed on any computer." This is very specialized usage of the terms "peer-to-peer" and "network operating system." Outside the *A+* context, these words refer to very different things, and these sentences are delusional, the kind of thing that make you think not "What a skilled network technician!" but "I wonder what color the sun is on that planet!" It is unkind to learners to let this sort of thing go by without mentioning it, just as it would be unkind to lead them to expect that every organization has a project management plan and a well-defined change management process for every part of it.

Head First PMP will make a picture out of anything. If it's well suited to a picture, there will definitely be one. Ifit can be depicted graphically (it's a process with inputs and outputs, for instance), it will be. If there have been two pages in a row with no pictures, they'll make up an example and illustrate that. *A+* limits itself mostly to the standard icons the Nutshell Guide series uses and screen shots, although it contains pictures when there's really no other way to show something. But it doesn't go at all out of its way, so that topics such as relative sizes of PC motherboards don't get pictures.

I didn't read *A+ etc. Exams* in its entirety because I don't think I could have. I'm not sure anybody

could, except in very small doses. Admittedly it is labeled "A desktop quick reference" and you don't expect reference books to be gripping end-to-end reads, but there aren't a lot of situations where you need a desktop reference guide to an exam. In fact, the only one I can think of is the situation I find myself in, where I sometimes teach course material that isn't aimed at these exams but where it is a selling point to make it useful for these exams. Being able to look up what the exam covers is useful information for me, and the book is perfectly adequate for that.

I read *Head First PMP* in its entirety, and I did the majority of the exercises. I'm pretty certain that if I read it again and did all the exercises, I'd pass the exam. I also learned some useful stuff about project management, as well as about the PMP exam.

I'd recommend *Head First PMP* to people who're interested in the PMP exam. I'd really only recommend the *A+* book to people who really need a desktop reference to the relevant exams.

INSIDE NETWORK PERIMETER SECURITY

Stephen Northcutt, Lenny Zeltser, Scott Winters, Karen Kent, and Ronald W. Ritchey

Sams Publishing, 2005. 660 pages.

ISBN 0-672-32737-6

I was predisposed not to like this book. It has too many authors, to start with, and then I glanced at the title and thought, "That's interesting, a book on security inside the network perimeter," and was disappointed to discover that it was meant to be an inside look at network perimeter security.

But actually, it's a good look at network security (with a concentration on perimeter security). The writing is a little bit uneven, but not terribly, and there are only a few places where the seams between chapters show as information gets repeated with an inconsistent slant. It covers the important stuff, including some often-neglected topics such as logging, troubleshooting, and VPN integration. And it treats current hype with the restraint it deserves, acknowledging that "intrusion prevention" is a good thing when used judiciously but not a staggering work of genius that will save the world.

Its chapter on auditing underestimates both the naiveté of novice testers and the fragility of average networks—yes, it does include "Get consent, because people have been arrested and convicted for testing security without permission," but it fails to say things such as "Most networks include networked devices that are not multi-purpose computers and those devices tend to have extremely fragile IP implementations." Pretty much every network scan I've ever seen has brought something crashing down, with varying degrees of havoc. The document scanner that disabled its Ethernet interface whenever you pinged it so that the most gentle and considerate of network scans caused it to disappear was merely comic; the friend who brought an entire manufacturing line down was considerably more scarred by the process. Oh, and by the way, if you war-dial a large organization, there will be somebody at work and they will notice. The results of this can also range from comic to deeply disturbing, but deeply disturbing is pretty well guaranteed if you're running during the day and lots of people notice. They develop conspiracy theories, and complain a lot, because frankly, it's very annoying to have the phones all ring one after another.

The final chapters are the weakest, ending with an extended riff on the castle metaphor, which is strained to start with and is entirely abandoned at the end.

Overall, this book provides a good overview of network perimeter security. I'd recommend it to a reasonably experienced network administrator who wanted to understand the whole picture of network security.

AMPLIFYING YOUR EFFECTIVENESS: COLLECTED ESSAYS

Gerald M. Weinberg, James Bach, and Naomi Karten, editors

Dorset House, 2000. 134 pages.

ISBN 0-932633-47-1

As you can see from the summary information, this is a small book, and not a recent one. It was also an accident (I don't know whether I slipped or the publisher did, but it's certainly not the book I intended to request). But it was there, and I read things, so I read it. And I had a really good time reading it, with some moments of enlightenment.

For sheer fun, I recommend the project management haiku, some of which manages to be not only funny but actually insightful: "If a project fails/but we keep working on it/has it really failed?"

My favorite for insight is "Good Practice Hunting," which is about why "best practice" is going to depend on what and where you're practicing. I also love the explanation of the Satir change model, showing why it is that group productivity goes down and people get all weird when you replace the awful broken system they're used to with a nonawful system. These are both essays that I can see myself shoving on people who need to understand their concepts.

This books deals mostly with highly accessible thinking about technical management issues; it would be good for technical managers and the people who have to manage them (from above or below).

ENDPOINT SECURITY

Mark S. Kadrich

Addison-Wesley Professional, 2007. 383 pages.

ISBN-10: 0-321-43695-4; ISBN-13: 978-0-321-43695-5

REVIEWED BY SAM STOVER (SAM.STOVER@GMAIL.COM)

I'm going to start this review by being as honest as I possibly can. When I got this book, I was convinced that it was going to be an utter waste of time. Boy, was I wrong. I found this work to be engaging, interesting, informative, provocative, and most of all fun. I don't necessarily agree with everything the author says, but I do think he's asking the right questions. For example, in Chapter 2, page 25, in a section titled "We're Not Asking Vendors Hard Questions," he lists two questions that he always asks when a vendor is peddling a product to him: (1) What type of systems development life cycle (SDLC) do you use? and (2) What software analysis tools do you use to discover coding flaws in your software?

I don't know about you, but I think those are two fine questions, and I've added them to my repertoire. This book hones in on the problem of security in a way I find very interesting. We all know we have a problem, but the people who are selling us the fixes are dependent upon the problem to stay in business. After exploring that topic in the first three chapters, the author moves into a discussion of where the problems really should be solved—at the endpoint (hence the name of the book). Chapters 4–6 deal with accepting that endpoints are the "real" targets, and how to start building an environment where endpoint security is the goal. To be sure, it doesn't make much sense to spend millions of dollars on the latest and greatest network monitoring devices if your endpoints are ignored. I don't think the author is saying that the monitoring devices are unnecessary, just that it is too easy to forget about the endpoint—and that's where the action is.

Chapter 7, "Threat Vectors," briefly discusses applications and operating systems as means to compromise. Security geeks will find this chapter fairly commonplace, but I think the author does a good job of getting the point across that users, and their desire for more/faster/better applications, ultimately drive the insecurity market.

Chapters 8 through 12 address the different kinds of endpoints and provide a laundry list of things to examine for each type. Each chapter is built uniformly, with emphasis on system checks, hardening measures, application gotchas, and more. This is really the meat of the book—the chapters leading up to this really serve to justify the attention that we have to give to our endpoints, and these chapters provide a step in the right direction. The author addresses Windows, OS X, Linux, PDA/SmartPhones, and embedded devices. I found the methods and suggestions to be informative and beneficial, especially for new users or those unfamiliar with a particular OS.

The paradox, however, is that this book starts something that cannot be finished. Operating systems and applications are always going to evolve and change, and we'll constantly be playing catch-up in order to keep our endpoints secure. I'm not disagreeing with the author—I think endpoint security is extremely critical. But user education, which this book also advocates, is another important part of the equation. You can have the best protected system around, but if the user willingly installs unknown software, the gig is up.

In all, this is a well-written book that's relatively short and an easy read. I found the author's sense of humor to be dry and witty. The constant comparisons between security devices and home heating and/or plumbing devices are fitting and humorous. Although I'm not sure that securing the endpoint will end all woes, it is definitely a process that is too easily overlooked in the search for the Ultimate Security Solution and needs more attention than it is getting. This book should definitely serve as a primer for anyone looking to move in that direction.

REVERSING: SECRETS OF REVERSE ENGINEERING

Eldad Eilam

Wiley, 2005. 624 pages.

ISBN 0-7645-7481-7

REVIEWED BY SAM STOVER (SAM.STOVER@GMAIL.COM)

It has been a long time since I have read a book that I like as much as *Reversing*. Too long. But the wait was worth it: This book is amazing. If you are planning on doing any kind of reverse engineering, this is the first book you should buy. It probably won't be the only book, but it should definitely be the first.

The book is divided into four parts: Reversing 101, Applied Reversing, Cracking, and Beyond Disassembly. Each part comprises 2–4 chapters, with 13 chapters in all. Experienced reversers probably

won't have to read the book from start to finish, and honestly, I don't think anyone else will either. It's my experience that people get into reversing because they have a goal in mind, and the sections discussed in this book are diverse enough that you'll probably want to skip right to the part that interests you. Go right ahead—that's exactly what I did. I went right to Chapter 8 (Reversing Malware), read enough to realize that I didn't know enough, and went back to the Reversing 101 chapters and built a foundation.

There are four chapters in the Reversing 101 section. The first provides a short intro that gives an explanation of reversing techniques, as well as a discussion on justification. Reversing has a bad connotation in some circles, so it's a good idea to familiarize yourself with those circles before you start reversing everything and anything you can get your hands on. This section rounds out with an overview of low-level software and Windows OS fundamentals, then a solid chapter on the tools that you'll be using.

For me, the second part was where this book really came into its own. The Applied Reversing chapters are unique in that they each addresses reversing from a different angle. This provides insight into the various reasons *why* people reverse. Chapter 5 emphasizes reversing for product compatibility because it is sometimes impossible to obtain support or documentation for an API with which you need to interface. Reversing that API could be the answer you need to quickly obtain enough information to ensure that your program uses it effectively. The whole chapter is dedicated to reversing a set of undocumented Windows APIs and learning how they work. Chapter 8 approaches reversing from the standpoint of understanding how malware works—and we all know that malware authors have no intention of making our jobs any easier when it comes to figuring out what they are trying to accomplish. Again, the author makes skillful use of an existing binary (Hacarmy.D backdoor) and steps through the reversing techniques. You can download the backdoor file from the book Web site.

The three Cracking chapters reverse (pun intended) the focus of the book. Now you are the author of the code, and you have to stop other people from reversing your work. The author details how (and why) people crack copy protection, then spends an entire chapter on anti-reversing techniques that you can use to make cracking that much harder. The final section was, quite honestly, a bit out of my reach. Chapter 12 deals specifically with reversing .NET and Chapter 13 addresses decompilation. Not being a .NET or a C programmer by trade, I didn't spend much time with these chapters, but if you need something in these areas there is a ton of info there.

Bear in mind several caveats before grabbing this book. First, in the opening section of Chapter 4, the author outlines the two different reversing methodologies: Offline Code Analysis and Live Code Analysis. In Chapter 5, the author openly acknowledges that this book focuses almost exclusively on Offline Code Analysis instead of "running programs in the debugger and stepping through them." This focus largely results from the suitability of the printed media, and although I understand completely, it was a bit of a letdown. Live Code Analysis, coupled with System Monitoring (discussed in Chapter 4), is much easier for the neophyte reverser, but much harder for the author to write about. Second, I'm sure that the people writing the code that you will be reversing will eventually find out about this book as well, so you can expect that anti-reversing techniques will become more abundant and effective. As with any security technology, the leap-frog of offense and defense will always continue. I just hope it results in another book as good as this one.

Overall, the book was extremely well written, with an extraordinary emphasis on walking through examples of each technique. A lot of the tools and procedures used in each chapter were very similar, but with different goals in mind. This is an exemplary book, and I commend the author on making a voodoo subject approachable to anyone, even me (no small feat).

MYTHS OF INNOVATION

Scott Berkun

O'Reilly Media, 2007. 176 pages.

ISBN-10: 0596527055; ISBN-13: 978-0596527051

REVIEWED BY RIK FARROW

This slim volume shatters many myths I had about invention and innovation. Like many people, I harbored the notion of the "Eureka" moment, a high-intensity experience that will lead to a world-changing invention and that will make me wealthy as a by-product. Berkun dispels this notion quickly, in the very first chapter, as he points to innovator after innovator who spent many years working to come up with the great idea, and many more years developing that idea into a successful product. Euripedes, the Greek who gives us the saying, had worked long and hard to solve the problem of de-

termining whether a gift to a king was pure gold or not. The displacement of the water in his washtub gave him the insight he needed, and the rest is, uh, history.

Berkun's intent is not simply to dispel myths, although he does that with clear and insightful writing. He is also out to teach us about how innovation really works. Berkun also explains how to quickly stamp out innovation via criticism and typical management techniques. He points to environments where innovation thrives, as well as explains why great ideas often languish. I'll give you one hint: Just having an excellent notion is not enough. The history of innovation rarely records failures, but it does tell us that those who succeeded overcame hurdles not only of implementation but also of cultural resistance to change.

If you ever had what you thought was a great idea, or work at a company that considers itself to be in the business of innovation, you owe it to yourself to read this book. It is an easy and pleasant read, and it might be just the thing you need to innovate and follow through on your inspiration—or perhaps just decide that you need to work someplace else.

LIVE LINUX CDS

Christopher Negus

Prentice Hall, 2007. 430 pages.

ISBN 0-13-243274-9

REVIEWED BY RIK FARROW

I will start off by confessing that I didn't read this entire book. I have been using Knoppix, a live CD, for many years now as part of my Linux Hands-On security class, and wondered if this book would make the process of remastering Knoppix CDs any easier. I can say that the chapter on remastering Knoppix did indeed make my life simpler. I found the directions clear and accurate. At one point, I thought I had found a mistake in the instructions, but I had merely misread the book. I wanted a feature to be there, the copying of all files and directories in /etc/skel, but the book says that Knoppix only copies the Desktop and .kde directories, and the book was right.

This book comes with directions for copying and remastering many Live Linux CDs. It also comes with a DVD that contains multiple Live Linux images, so you can boot different versions of Live CDs and see how they work pretty much effortlessly. The early chapters (which I did read) about creating CDs, the Linux boot process, and the many types of Linux Live CDs available were clear and easy reading. I can recommend this book to anyone who prefers a set of up-to-date and proofread instructions over what you can find on the Web.

USENIX notes

USENIX MEMBER BENEFITS

Members of the USENIX Association receive the following benefits:

FREE SUBSCRIPTION to *;login:*, the Association's magazine, published six times a year, featuring technical articles, system administration articles, tips and techniques, practical columns on such topics as security, Perl, VoIP, and operating systems, book reviews, and summaries of sessions at USENIX conferences.

ACCESS TO ;LOGIN: online from October 1997 to this month: www.usenix.org/publications/login/.

ACCESS TO PAPERS from USENIX conferences online: www.usenix.org/publications/library/proceedings/

THE RIGHT TO VOTE on matters affecting the Association, its bylaws, and election of its directors and officers.

DISCOUNTS on registration fees for all USENIX conferences.

DISCOUNTS on the purchase of proceedings and CD-ROMs from USENIX conferences.

SPECIAL DISCOUNTS on a variety of products, books, software, and periodicals. For details, see www.usenix.org/membership/specialdisc.html.

TO JOIN SAGE, see www.usenix.org/membership/classes.html#sage.

FOR MORE INFORMATION regarding membership or benefits, please see www.usenix.org/membership/ or contact office@usenix.org. Phone: 510-528-8649

USENIX BOARD OF DIRECTORS

Communicate directly with the USENIX Board of Directors by writing to *board@usenix.org*.

PRESIDENT

Michael B. Jones, *mike@usenix.org*

VICE PRESIDENT

Clem Cole, *clem@usenix.org*

SECRETARY

Alva Couch, *alva@usenix.org*

TREASURER

Theodore Ts'o, *ted@usenix.org*

DIRECTORS

Matt Blaze, *matt@usenix.org*

Rémy Evard, *remy@usenix.org*

Niels Provos, *niels@usenix.org*

Margo Seltzer, *margo@usenix.org*

EXECUTIVE DIRECTOR

Ellie Young, *ellie@usenix.org*

SUMMARY OF USENIX BOARD OF DIRECTORS MEETINGS AND ACTIONS

Toni Veglia

The following are the actions taken by the USENIX Board of Directors since December 2006.

CONFERENCES

A proposal from Tal Garfinkel to hold the First USENIX Workshop on Offensive Technologies (WOOT '07) was approved. The Board authorized the formation of a steering committee to look into the possibility of organizing a workshop on multicore technologies. It was agreed to form a steering committee for any future Workshops on Real, Large Distributed Systems (WORLDS). USENIX will also encourage the community at large to submit proposals for new workshops.

OUTREACH/GOODWORKS

An additional $17,610 was added to the 2007 Standards Budget to support participation in the C++ Standard revision. USENIX will support EuroBSDCon with a $5,000 grant. USENIX will sponsor the Richard Tapia Celebration of Diversity in Computing Conference at the $10,000 level.

FINANCES

The Board approved the USENIX Association Non-Qualified Deferred Compensation Plan. The Board agreed to instruct the USENIX Portfolio Manager to diversify USENIX investments away from companies deemed "socially irresponsible." LISA tutorial registration fees were increased by $10 per day.

POLICY

The Board agreed that USENIX's official policy in cases of plagiarism would be, at a minimum, to inform the next higher level at the person's workplace.

FLAME AND STUG AWARDS

2007 FLAME AWARD WINNER: PETER HONEYMAN

The USENIX Lifetime Achievement Award ("The Flame") recognizes and celebrates singular contributions to the UNIX community of both intellectual achievement and service that are not recognized in any other forum.

Peter Honeyman receiving the Flame Award from Matt Blaze at the 2007 USENIX Annual Technical Conference

In the words of the presentation:

Dr. Peter Honeyman has had a profound and lasting impact on the field of computer science. While many know Peter for his seminal contributions to computing systems, such as Honey DanBer UUCP and Disconnected AFS, it is his efforts as a mentor that we wish to honor with the USENIX Lifetime Achievement Award. Peter's often highly unconventional stewardship of the countless students, researchers, and advisees he has touched is the stuff of graduate student legend. His penetratingly insightful (and potentially hazardous) questions and comments, combined with a paradoxically unflinching loyalty, consistently have led those under his tutelage to the pinnacle of achievement in security, systems, and networking. Peter's questioning during conferences and doctoral defenses, although sometimes frightening, always demanded better from those of us who attempt to advance science.

We also wish to honor Peter's mentorship of the technology community: few people have so selflessly shared their time and counsel to ensure a lasting venue for high-quality discourse. In particular, his efforts as supporter and Board member have been instrumental in the birth, growth, and continuing success of USENIX.

2007 STUG AWARD WINNER: GUIDO VAN ROSSUM

The STUG award recognizes significant contributions to the community that reflect the spirit and character demonstrated by those who came together in the Software Tools User Group (STUG). Recipients of the annual STUG award conspicuously exhibit a contribution to the reusable code-base available to all and/or the provision of a significant enabling technology to users in a widely available form.

Guido van Rossum accepting the STUG Award at the 2007 USENIX Annual Technical Conference

In the words of the presentation:

The Python programming language is known for many things. Most important, it pays homage to *Monty Python's Flying Circus*. It is a dynamic, object-oriented language with simple, yet efficient, high-level data structures. Guido van Rossum, the originator of Python, emphasized readability and ease of use and reuse. Python's elegance has made it an increasingly attractive tool for scripting, rapid application development, and general programming. We believe that developers are attracted to Python because such thought was put into making the syntax obvious and simple; for instance, Python, unlike most other dynamic languages, uses indentation to group statements.

In an article describing his first experiences with Python, Eric S. Raymond wrote, "The long-term usefulness of a language comes not in its ability to support clever hacks, but from how well and how unobtrusively it supports the day-to-day work of programming" (quote from www.python.org/about/success/esr/). Python is open source, free software. In fulfillment of van Rossum's original goals, the community of Python programmers has spread across multiple operating systems and hardware platforms.

In light of his contributions in the STUG spirit and to the realization of a major enabling technology, USENIX recognizes Guido van Rossum with the 2007 STUG Award.

USACO UPDATE

Rob Kolstad,
USACO Head Coach

USENIX is the Platinum sponsor of the USA Computing Olympiad, the premier pre-college computer programming competition. The USACO conducts half a dozen Internet-based programming contests through the year, hosting about 1,000 students in each one. The USACO contests are "open" in the sense that they encourage participation from students around the world.

Each contest is typically offered not only in English but also in another half-dozen languages.

Surprising to some Americans, a huge percentage of non-USA students speaks English with enough skill to solve computer contest problems. Contests commonly garner students from 66 different countries.

Contests are complemented with 200 hours of online training. This training program offers instruction in the relatively esoteric area of algorithmic programming and requires students to solve sets of contest-like programming tasks before moving on to subsequent sections of training.

For USA students, the goal of the contests is to earn one of 16 berths at the USA Invitational Computing Olympiad (USAICO), a nine-day competition, conducted this year at Colorado College in Colorado Springs, Colorado. While the arrival and departure days are relatively short (arrival mid-afternoon at the Denver airport, departure at 0730 to return to the airport), the intervening week is a busy one. Four three-hour programming contests and two five-hour contests keep the students on their toes as they vie for four positions on the USACO traveling team, which will represent the USA at the International Olympiad on Informatics. This year's IOI will be held August 15–22 in Zagreb, Croatia.

The sixteen participants and six coaches hailed from around the country, with no school truly dominating the list:

Seniors: Zarathustra Brady, Magnolia Science Academy, Van Nuys, CA; Richard McCutchen, Montgomery Blair High School, Rockville, MD; John Pardon, Durham Academy Upper School, Chapel Hill, NC; Bohua Zhan, West Windsor-Plainsboro HS South, Plainsboro, NJ

Juniors: Artur Dmowski, Stuyvesant HS, Corona, NY; Boping Lai, The Roxbury Latin School, Lexington, MA; Kevin Lee, Bergen County Academies, Closter, NJ; Yongqian Li, Niskayuna High School, Niskayuna, NY; Spencer Liang, The Harker School, Cupertino, CA; Haitao Mao, Thomas Jefferson High School for Science and Technology, Vienna, VA; Jacob Steinhardt, Thomas Jefferson High School of Science & Technology, Vienna, VA; Ye Wang, Nicolet High School, River Hills, WI; Louis Wasserman, Montgomery Blair HS, Derwood, MD

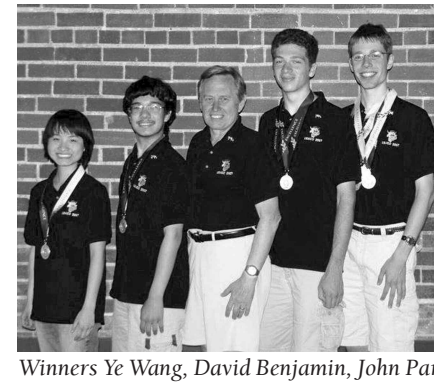
Sophomore: David Benjamin, Harrison High School, West Lafayette, Indiana

Freshmen: Neal Wu, Baton Rouge Magnet High School, Baton Rouge, LA; Scott Zimmermann, Montgomery Blair High School, Gaithersburg, MD

Ye Wang is the first exchange student to qualify for the USAICO competition. Although she attends high school in Wuhu, China, she spent her junior year in River Hills, Wisconsin, and thus was eligible to represent the USA in the national and international competitions.

Half a dozen coaches set problems and shepherded the students through the week. They included director Don Piele, recently retired from the University of Wisconsin; Rob Kolstad, aspiring consultant; Brian Dean, up-and-coming faculty member at Clemson University; Berkeley grad student Percy Liang; MIT student Alex Schwendner; and problem-setter extraordinaire Richard Peng, who has just completed his first year at Canada's prestigious University of Waterloo in Toronto, Ontario.

The four contestants chosen to represent the USA were seniors Richard (Matt) McCutchen and John Pardon, junior Ye Wang, and sophomore David Benjamin (see photograph). This is the



Winners Ye Wang, David Benjamin, John Pardon, and Richard (Matt) McCutchen, with director Don Piele (center)

first time a woman has earned a berth on the extraordinarily competitive traveling team. These students solved extremely challenging problems like "Moving the Hay," by Richard Peng:

"After he partitioned his farm into R (1 <= R <= 200) rows and C (1 <= C <= 200) squares conveniently labeled 1,1 through R,C, Farmer John spent days cutting the hay and stacking a huge amount of it in square 1,1. He then undertook the task of mapping out the N (1 <= N <= 80,000) haypaths through the farm so that he could deduce the maximum rate he could move hay from square 1,1 to square R,C.

"Each haypath uniquely connects the middle of two rectilinearly adjacent partitioned squares and has some capacity limit L_i (1 <= L_i <= 20,000,000) that is the maximum amount of hay that can be transported in either direction across the haypath. He's just positive that he can move hay at a reasonable rate to the other side of the farm but he doesn't know what the fastest rate is. Help him learn it."

As a heritage from the days of holding our training camp in Wisconsin, USACO problems tend to concern cows, farms, and Farmer John's dilemmas. This

problem is challenging not only for its algorithmic difficulty but also because the time limit was 1.0 CPU seconds on a 700MHz processor used for automatic grading. Doesn't look that hard, does it?

USENIX ASSOCIATION FINANCIAL REPORT FOR 2006

Ellie Young

The following information is provided as the annual report of the USENIX Association's finances. The accompanying statements have been reviewed by Michelle Suski, CPA, in accordance with Statements on Standards for Accounting and Review Services issued by the American Institute of Certified Public Accountants. The 2006 financial statements were also audited by McSweeney & Associates, CPA's, whose unqualified opinion accompanies the complete financial statements. Accompanying the statements are several charts that illustrate where your USENIX and SAGE membership dues go. The Association's complete financial statements for the fiscal year ended December 31, 2006, are available on request.

USENIX continues to be a healthy organization. For fiscal year 2006, USENIX broke even in operations. The additional $736K in gains in investment income, interest, and dividend income from the Reserve Fund meant that we ended the fiscal year with a $742K increase in net assets.

USENIX MEMBERSHIP DUES AND EXPENSES

USENIX averaged 5,400 members in 2006, slightly down from 2005. Of these, 2,500 opted for SAGE membership as well, and 500 people are SAGE-only members. Chart 1 shows the total USENIX membership dues revenue ($553K) for 2006, divided by membership type. Chart 2 presents how those dues were spent. Note that all costs for producing conferences, including staff, marketing, and sales and exhibits, are covered by revenue generated by the conferences. Chart 3 demonstrates how the money allocated to student programs, sponsorship of other conferences, and standards activities ($284K) was spent in 2006. Chart 4 shows how the USENIX administrative expenses were allocated ("Misc." covers such items as renewals, taxes, licenses, consultants, temp help, and training). Chart 5 shows where the $224K to provide SAGE benefits and services was spent (note: SAGE member dues revenue was $131K).

USENIX ASSOCIATION
STATEMENTS OF FINANCIAL POSITION
As of December 31, 2006 and 2005

ASSETS	2006	2005
Current Assets		
Cash & cash equivalents	$ 1,245,162	$ 1,138,826
Receivables	63,087	47,081
Prepaid expenses	47,203	39,003
Inventory	4,388	5,913
Total current assets	1,359,840	1,230,823
Investments at fair market value	6,047,657	5,343,665
Property and Equipment		
Office furniture and equipment	503,596	477,724
Less: accumulated depreciation	(452,814)	(421,264)
Net property and equipment	50,782	56,460
Other assets	248,521	206,515
	$ 7,706,800	$ 6,837,463
LIABILITIES AND NET ASSETS		
Current Liabilities		
Accrued expenses	$ 671,921	$ 589,575
Accrued income taxes payable	-	11,200
Contributions held for OpenAFS	176	1,200
Deferred Revenue	55,290	40,000
Total current liabilities	727,387	641,975
Long-term Liabilities	248,521	206,515
Total liabilities	975,908	848,490
Net Assets		
Unrestricted Net Assets	6,730,892	5,988,973
Net Assets	6,730,892	5,988,973
	$ 7,706,800	$ 6,837,463

USENIX ASSOCIATION
STATEMENTS OF ACTIVITIES
For the Years Ended December 31, 2006 and 2005

	2006	2005
REVENUES		
Conference & workshop revenue	$ 3,407,994	$ 3,281,826
Membership dues	552,978	572,560
Product sales	8,017	12,858
SAGE dues & other revenue	131,290	128,039
General sponsorship	0	2,925
Total revenues	4,100,279	3,998,208
OPERATING EXPENSES		
Conferences & workshops	2,645,671	2,709,818
Membership; login:	394,439	387,520
Projects & GoodWorks	284,472	291,935
SAGE	224,313	164,111
Management and general	433,327	457,473
Fund Raising	112,143	97,782
Total operating expenses	4,094,365	4,108,639
Net operating surplus/(deficit)	5,914	(110,431)
NON-OPERATING ACTIVITY		
Donations	0	0
Interest & dividend income	245,024	171,766
Gains & losses on marketable securities	556,471	32,907
Investment fees	(62,185)	(59,267)
Other non-operating	(3,305)	(20,197)
Net investment income & non-operating expense	736,005	125,209
Increase/(decrease) in net assets	741,919	14,778
Net assets, beginning of year	5,988,973	5,974,195
Net assets, end of year	$ 6,730,892	$ 5,988,973

USENIX ASSOCIATION
STATEMENTS OF FUNCTIONAL EXPENSES
For the Years Ended December 31, 2006 and 2005

	Conferences and Workshops	Programs and Membership	Student Programs, Good Works and Projects	SAGE	Total Program	Management and general	Fund Raising	Total Support		2006 Total		2005 Total
Operating Expenses												
Conference & workshop-direct	$ 1,747,016	$	$	$	**1,747,016**	$	$	**0**	$	**1,747,016**	$	**1,878,613**
Personnel and related benefits:												
Salaries	557,566	112,292	12,825	62,924	**745,607**	176,676	56,728	**233,404**	0	**979,011**		**1,002,359**
Payroll taxes	42,624	8,585	980	4,810	**56,999**	13,506	4,337	**17,843**	0	**74,842**		**74,591**
Employee benefits	121,927	24,557	2,805	13,760	**163,049**	38,634	12,405	**51,039**	0.0	**214,088**		**209,267**
Membership/products		6,241			**6,241**			**0**		**6,241**		**8,068**
Membership/login:		164,301			**164,301**			**0**		**164,301**		**137,433**
SAGE expenses				117,813	**117,813**			**0**		**117,813**		**52,061**
Student programs, Good . Works, and projects			262,250		**262,250**			**0**		**262,250**		**263,473**
General and administrative	176,538	78,463	5,612	25,006	**285,619**	204,511	38,673	**243,184**		**528,803**		**482,775**
	$ 2,645,671	$ 394,439	$ 284,472	$ 224,313	**$ 3,548,895**	$ 433,327	$ 112,143	**$ 545,470**	$	**4,094,366**		**4,108,639**

USENIX ASSOCIATION
STATEMENTS OF CASH FLOWS
For the Years Ended December 31, 2006 and 2005

	2006	2005
CASH FLOWS FROM OPERATING ACTIVITIES		
Change in net assets	$ 741,919	$ 14,778
Adjustments to reconcile increase in net assets to net cash provided by/(used for) operating activities:		
Depreciation	31,550	41,524
Decrease/(Increase) in receivables	(16,006)	39,336
Decrease in inventory	1,525	4,916
Decrease/(Increase) in prepaid expense	(8,200)	14,313
Increase in accrued expenses	81,322	277,019
Increase in accrued income taxes	(11,200)	11,200
Increase in deferred revenue	15,290	38,140
Total adjustments	94,281	426,448
Net cash provided by operating activities	836,200	441,226
CASH FLOWS PROVIDED BY/(USED FOR) INVESTING ACTIVITIES:		
Purchase of investments	(3,743,603)	(2,423,109)
Sale of investments	3,743,603	2,423,109
Net investment income designated for long-term purposes	(142,510)	(105,057)
Realized & unrealized gains on investments	(561,482)	(32,907)
Disposition of equipment	0	1,643
Purchase of property & equipment	(25,872)	(15,210)
Net cash used for investing activities	(729,864)	(151,531)
Net change in cash & equivalents	106,336	289,695
Cash & equivalents, beginning of year	1,138,826	849,131
Cash & equivalents, end of year	$ 1,245,162	$ 1,138,826
Cash payments for:		
Interest	-0-	-0-
Taxes	-0-	$3,532

Chart 1: USENIX Member Revenue Sources 2006

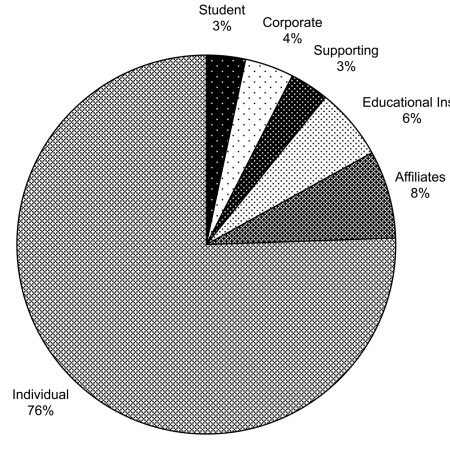


Chart 2: Where Your 2006 Membership Dues Went

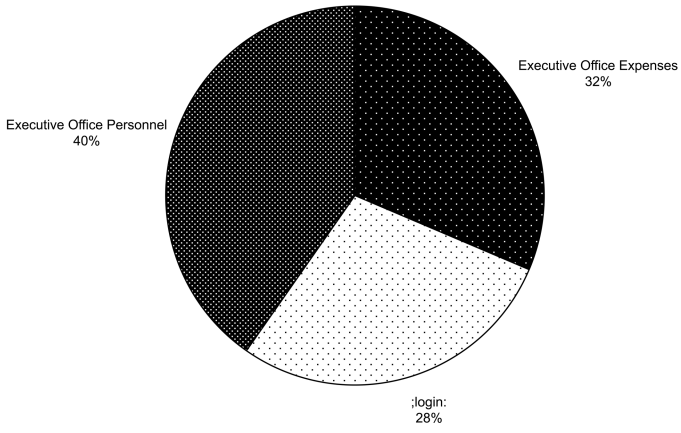


Chart 3: Student & Support for Other Programs Expenses 2006

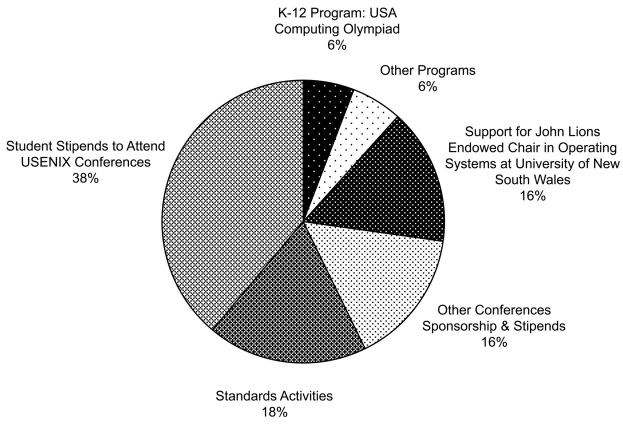


Chart 4: USENIX Administrative Expenses 2006

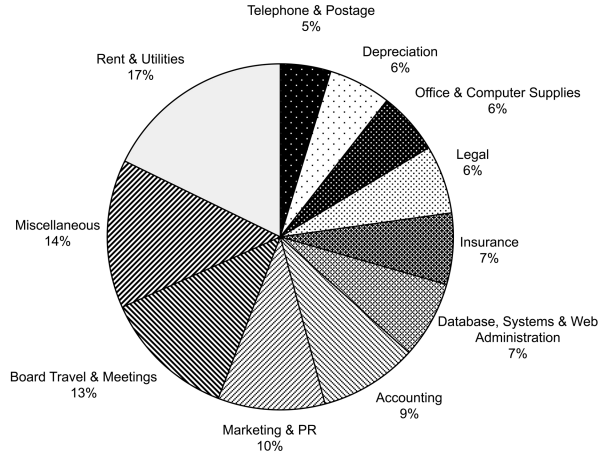
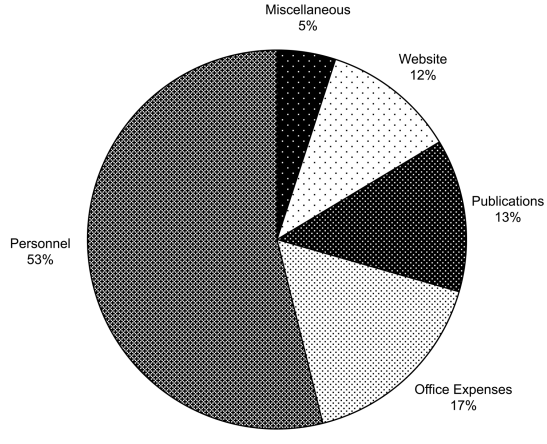


Chart 5: SAGE Expenses 2006



conference reports

THANKS TO OUR SUMMARIZERS

NSDI '07: 4th USENIX Symposium on Networked Systems Design & Implementation

Cambridge, MA
April 11–13, 2007

KEYNOTE ADDRESS

Security of Voting Systems

Ronald L. Rivest, Viterbi Professor of Computer Science, Massachusetts Institute of Technology

Summarized by Soila Pertet (spertet@ece.cmu.edu)

Voting systems should provide end-to-end security where voters can verify that their vote was cast as intended and counted as cast. However, end-to-end security is complicated by the need to maintain voter privacy while ensuring verifiability.

Ronald Rivest presented several approaches for providing end-to-end security, even in the absence of trusted computing platforms. Each approach relied on voter receipts and public bulletin boards. Voters receive a receipt when they cast their vote; this receipt does not reveal how they voted. All ballots get posted to a public bulletin board, and voters can use their receipt to protest if their vote was not cast as intended.

Rivest described two cryptographic approaches for designing secure voting systems that preserved voter privacy: mixnets (Chaum) and public mixing (Adida). These approaches randomly permute encrypted ballots so that the ballots cannot be correlated back to the voters. Rivest also presented a three-ballot scheme that did not rely on cryptography. In this scheme, each voter casts three ballots with at least one but no more than two votes for each candidate. The voter keeps an arbitrary copy of one of the ballots as a receipt, and all three ballots are posted to the public bulletin board.

Rivest briefly discussed Internet voting. In his opinion, Internet voting is vote-by-mail made worse, because voters are denied an enforced moment of privacy to cast their vote free of coercion or bribery. This problem is compounded by the fact that about one in every four PCs belongs to a botnet.

Rivest also mentioned that security is not all about technology; election officials and poll workers are an important part of the process. He recommended that the audience members get more involved in the election process, for instance by volunteering to be poll workers.

Many attendees asked questions. How can ordinary users (who might not understand cryptography) gain confidence in voting systems? Rivest answered that they can gain confidence in the voting system through indirect verification, where an expert of their choice examines the publicly available source code and cryptographic algorithms for the voting system. Can you cast your vote using virtual technology, for example, have your vote initially cast on a USB key at home? To do this you would need a trusted electronic agent that can represent you (e.g., a cell phone). There is still a lot of research that needs to be done before we get to this point: Building trusted computing platforms is a hard problem. Another attendee pointed out that there is a big gap between the ideas proposed in academia and what is being done in practice, so how can we bridge this gap? Rivest answered that we can bridge this gap by getting involved (e.g., volunteering to be a poll worker) or working with the state representatives to raise the standard. What level of tamper protection do the cryptographic schemes provide? Do they simply detect security violations or can they recover from the violations once detected? It is possible to recover from security violations; for example, you can replace a cheating mix server if the proofs do not match, and then redo the computation.

CONTENT DELIVERY

Summarized by Anupama Biswas (anupamabiswas@gmail.com)

- ***Do Incentives Build Robustness in BitTorrent?***

  Michael Piatek, Tomas Isdal, Thomas Anderson, and Arvind Krishnamurthy, University of Washington; Arun Venkataramani, University of Massachusetts

  Awarded Best Student Paper!

Michael explained that the mechanism used by BitTorrent functions tit-for-tat for reciprocating with clients. He proved this statement by using a client BitTyrant that replaces the strategy used by BitTorrent. The BitTorrent strategy does not eventually lead to improvement in performance. BitTyrant responds to those clients that do not make altruistic contributions by degrading performance. Also, robustness in BitTorrent is not due to incentives. BitTorrent does not address the problem of performance degradation if the peers strategically manipulate the system.

The key idea for BitTyrant is to carefully select peers and contribution rates so as to maximize download rates per unit of upload bandwidth. The strategic behavior of BitTyrant is executed simply through policy modifications to existing clients without any change to the BitTorrent protocol. Michael showed the performance of BitTyrant, evaluated on real swarms, establishing that all peers, regardless of upload capacity, can significantly improve download performance while reducing upload contributions. Also, the performance is affected as peers individually benefit from BitTyrant's strategic behavior, irrespective of whether or not other peers are using BitTyrant. Peers not using BitTyrant can experience degraded performance owing to the absence of altruistic contributions. Taken together, these results suggest that incentives do not build robustness in BitTorrent. In addition to the primary contribution, BitTyrant, the efforts to measure and model altruism in BitTorrent are independently noteworthy. First, the model used is simpler and is still sufficient to capture the correlation between upload and download rates for real swarms. Second, existing studies recognizing altruism in BitTorrent consider small simulated settings or few swarms that poorly capture the diversity of deployed BitTorrent clients, peer capacities, churn, and network conditions. One of the questions asked concerned the possibility that a hacker might just give the impression of having a higher upload time. This might lead to it being assigned more bandwidth than the other peers. Michael said that this situation has been handled successfully by BitTyrant.

- ***Exploiting Similarity for Multi-Source Downloads Using File Handprints***

  Himabindu Pucha, Purdue University; David G. Andersen, Carnegie Mellon University; Michael Kaminsky, Intel Research Pittsburgh

Bindu Pucha presented a new approach for downloading similar data from multiple sources using a technique called File Handprints. The approach presented for downloading a specific file is a mix of the existing approaches. Currently approaches such as BitTorrent try to locate an exact copy of the object the user is looking for but do not look for the desired object in similar sources. Another approach is looking for chunks of the data object in multiple sources, which involves performing a number of lookups. This leads to limiting the scalability of the system. The approach presented locates similar objects using a constant number of lookups and inserting a constant number of mappings per object. The handprinting is similar to shingling, fingerprinting, and deterministic sampling. It uses the technique of exploitable similarity and not document resemblance. When performance was checked against the performance with BitTorrent, it was found to exceed the performance of BitTorrent in locating a file from multiple resources. The download time was faster for a given P2P connection.

- ***Cobra: Content-based Filtering and Aggregation of Blogs and RSS Feeds***

  Ian Rose, Rohan Murty, Peter Pietzuch, Jonathan Ledlie, Mema Roussopoulos, and Matt Welsh, Harvard University

Blogs and RSS feeds are becoming increasingly popular. However, the problem lies in finding and tracking interesting content in blogs, which currently is a cumbersome process. A solution such as providing the users with the

ability to perform content-based filtering and aggregation across the millions of available Web feeds, obtaining a personalized feed containing the articles of a user's interest, will be of real value. Also providing real-time updates of articles of interest helps the user to avoid having to keep tabs on a multitude of interesting sites. Ian provided such a solution in the paper. The blog search sites available do not present a clear idea as to how well these sites scale to handle large number of feeds and users and also provide low time delay for the searches. The contents of the various blogs keeps on changing, and it is not known how fast the current Web searching techniques can search such blogs with minimum time delay. Similar content posted on various blogs, which requires the user to search through multiple blogs, can be aggregated using SharpReader and FeedDemon, both of which collect stories from multiple sites along thematic lines (e.g., news or sports). A single RSS feed looks into a individual blog. It does not aggregate the data with similar content.

He presented Cobra (Content-Based RSS Aggregator), a distributed scalable system that provides personalized views of articles to users taken from potentially millions of RSS feeds. The information is collected by the system that crawls, filters, and aggregates vast numbers of RSS feeds. It delivers to each user a personalized feed based on the user's interests. Cobra consists of a three-tiered network of crawlers, filters, and reflectors. Crawlers scan the Web feeds. Filters match the crawled Web feeds to user subscriptions, and reflectors provide recently matching articles on each subscription as an RSS feed, which can be browsed using a standard RSS reader. This system is capable of handling a large number of source feeds and users, keeping the latency time low.

OVERLAYS AND MULTICAST

Summarized by Murtaza Motiwala (murtaza@cc.gatech.edu)

■ *Information Slicing: Anonymity Using Unreliable Overlays*

Sachin Katti, Jeff Cohen, and Dina Katabi, Massachusetts Institute of Technology

Sachin Katti presented a new technique for distributing content anonymously in overlays without keys. Although Freenet allows anonymous distribution of content, it has very few users, since it relies on exchange of keys. Overlays are ideal for anonymous content distribution; however, they cannot be used as is since they don't involve any public keys and are not reliable owing to the large degree of node churn (nodes joining and leaving the overlay).

Information slicing achieves each of these objectives by splitting the original message and sending the slices on disjoint paths between the source and the destination. The technique presented has the key feature that only the destination gets all the pieces and is able to decode the original message, while none of the intermediate nodes gets the complete message. The anonymity of the sender and receiver is achieved by letting each node know only its next hop. Also, information slicing is able to deal with the node churn in overlays by adding redundancy using network coding to deal with loss of data resulting from the loss of a node on the path.

The authors evaluated their technique using simulation as well as on PlanetLab. Sachin presented the evaluation of anonymity, churn resiliency, and throughput performance in the talk.

Anonymity was evaluated by using an overlay of 10,000 nodes in which attackers can control nodes, snoop traffic, and collude. Entropy was used as the metric to determine the amount of information leaked to the attackers. The anonymity of the scheme was found to be comparable to Chaum mix, despite its having no keys.

In order to measure the resiliency of information slicing to node churn in the overlays, the authors compared it to onion routing with source coding. The results showed that information slicing had a much better resiliency to churn compared to onion routing. The throughput achieved by information slicing was also found to be much better than onion routing, because information slicing tries to use parallel paths to the destination. The results from the evaluation on PlanetLab were found to match the results from simulation.

There was a question on how link disjointedness was achieved, to which Sachin replied that the scheme required the source to be smart about picking paths in the overlay to get disjoint paths, for example by choosing nodes in different ASes. To a comment about anonymity decreasing with churn, Sachin noted that there was a tradeoff between increasing redundancy to protect against higher churn and having lesser anonymity. There was also a question on what the authors considered to be the throughput to which the response was that the throughput was the one observed at the destination. Also, on the subject of how information slicing can be incorporated in current P2P applications, the answer was to create a list of users who are interested in using information slicing and then choose paths using those nodes.

■ *SAAR: A Shared Control Plane for Overlay Multicast*

Animesh Nandi, Rice University and Max Planck Institute for Software Systems; Aditya Ganjam, Carnegie Mellon University; Peter Druschel, Max Planck Institute for Software Systems; T.S. Eugene Ng, Rice University; Ion Stoica, University of California, Berkeley; Hui Zhang, Carnegie Mellon University; Bobby Bhattacharjee, University of Maryland

Animesh used the example of an overlay that is optimized for data dissemination and in which a control overlay is used to build and repair the overlay and a gossip protocol is used to disseminate membership information and nodes

use probes to select a parent based on the metrics delay and available bandwidth. He quickly noted that such an approach will not scale in the case of large groups and high membership churn. He then proposed the idea of using separate control and data overlays, where the control overlay can be shared among different data overlays.

The control overlay in their architecture uses the anycast primitive for selecting overlays, which Animesh noted is a key factor in building efficient overlays. The nodes also keep aggregated information of metrics such as spare bandwidth capacity and depth. This helps in quickly pruning the trees without doing an in-depth first search when an anycast request comes in.

SAAR was evaluated on Modelnet using about 350 nodes with different available bandwidths. The evaluation showed that SAAR enables low join delays. The experiments showed that the initial SAAR delay is high; also, in the case of a single tree 99% of the delay was less than 2 seconds. The evaluation also showed that SAAR provided good streaming quality as compared to ESM, whose quality was poor. SAAR performed much better for the single- and multiple-tree cases.

To the question of whether SAAR introduced a single point of failure and how secure it was, the authors responded that they had not addressed freeloading and malicious behavior in the control overlay; however, mechanisms in structured overlays may be applied here to counter those.

■ ***Ricochet: Lateral Error Correction for Time-Critical Multicast***

Mahesh Balakrishnan and Ken Birman, Cornell University; Amar Phanishayee, Carnegie Mellon University; Stefan Pleisch, Cornell University

Mahesh Balakrishnan presented Ricochet, which is used to provide a time-critical reliable multicast in data centers. In a data center, it is common for a node to subscribe to multiple multicast groups. This could lead to situations in which there is a high data rate at some nodes, leading to overload and eventual dropping of packets. The challenge in such systems is to recover packets in real time and the solution must scale in the number of receivers, senders, and multicast groups. After observing several existing multicast solutions for data centers the authors observed that the latency (i.e., the time taken to recover from lost packets) is inversely proportional to the data rate.

Although Forward Error Correction (FEC) can be used to provide reliability guarantees with no retransmissions, the node has to wait for "r" data packets before generating the FEC and hence again the latency is inversely proportional to the data rate. The authors propose Lateral Error Correction (LEC), a new reliability mechanism to allow packet recovery latency to be independent of per-group data rate. In LEC, a node exchanges XORs of incoming data packets with c randomly chosen receivers. The protocol scales, as it is gossip style and has tunable per-group overhead.

Mahesh noted that the bandwidth overhead of Ricochet is proportional to the additional number of packets used for error correction. Also, the computation overhead is small, since XORs are very fast to compute (150 to 300 microseconds/packet). Also, the number of intersections between the various multicast groups is not exponential, as it is limited by the actual number of nodes in the system.

The authors evaluated Ricochet on a 64-node cluster using three packet-loss models (uniform, burst, and Markov). The evaluation showed that most lost packets were recovered in 50 milliseconds. Also, Ricochet was found to scale to hundreds of multicast groups and was about 400 times faster than SRM. The evaluation showed that Ricochet was resilient to bursty losses as well and could handle short bursts of 5 to 10 packets well. To handle even higher bursts, the authors used staggering in Ricochet. Amazingly, with a stagger of 6, Ricochet can recover 90% of packets from a burst loss of 100 packets.

There was a question on how Ricochet might work in wide area networks and not clusters. Mahesh replied that for a wide area network the losses might not be independent and the nodes might have to use an intelligent way to pick the people to talk to. Ricochet is available for download from http://www.cs.cornell.edu/projects/quicksilver/Ricochet.html.

WIRELESS

Summarized by Murtaza Motiwala (murtaza@cc.gatech.edu)

■ ***WiLDNet: Design and Implementation of High Performance WiFi Based Long Distance Networks***

Rabin Patra and Sergiu Nedevschi, University of California, Berkeley, and Intel Research, Berkeley; Sonesh Surana, University of California, Berkeley; Anmol Sheth, University of Colorado, Boulder; Lakshminarayanan Subramanian, New York University; Eric Brewer, University of California, Berkeley, and Intel Research, Berkeley

Rabin Patra presented the motivation, design, and implementation of WiFi-based Long Distance (WiLD) networks in developing regions. WiLD uses 802.11 radios because they are low-cost, have no spectrum costs, and have good data rates. At present, WiLD has been deployed in a number of places in the developing regions, including in India at Arvind Hospital (12 clinics approximately 15 km apart) and in Ghana. From their experience, they found that the point-to-point performance of WiFi over long distances is poor; for example, on a 60-km link the performance of TCP is 0.6 Mbps vs. 6 Mbps for UDP.

The design of WiLD is focused on fixing 802.11 by replacing CSMA with TDMA and enforcing synchronization on

multiple links to avoid collision losses. The design has the constraints of not involving any hardware changes and not permitting any modification of end hosts through modification of WiLD routers. Also, the routers are inexpensive and thus have low processing power. Rabin explained that the problems with using 802.11 over long distances are with ACKs, as they are inefficient over long links and also the ACK timeouts are very short compared to the delay over a long link (~110 km). Also, higher propagation delay increases the likelihood of collisions at the receiver end. The authors thus made the choice of using sliding window flow control at the MAC layer and disabled 802.11 MAC ACKs completely. They also enabled simultaneous sends and simultaneous receives by providing a 12-dB isolation. For recovering from losses, WiLD uses bulk ACKs or adaptive FEC. For using either of the techniques there is a tradeoff between bandwidth and delay; thus for bandwidth-sensitive protocols they used bulk ACKs, whereas for delay-sensitive ones they used FEC.

The authors implemented WiLD by modifying the Atheros madwifi driver, and they used the Click router to perform FEC encoding and decoding. Their evaluation showed that for a single-hop case, WiLDNet's performance increases with increase in distance. Also, in the multihop case, WiLDNet is more spectrum-efficient than traditional 802.11. Rabin noted that their future work was to look into remote network management and planning for WiLDNet.

There was a question regarding what kind of traffic WiLDNet was intending to optimize. Rabin responded that they were looking at a mix of applications including images, video conferencing, and Web browsing. As to whether the authors had evaluated WiLDNet against available wireless solutions such as WiMAX, Rabin replied that they had not done so, since they had a strict cost factor in mind.

■ ***S4: Small State and Small Stretch Routing Protocol for Large Wireless Sensor Networks***

Yun Mao, University of Pennsylvania; Feng Wang, Lili Qiu, and Simon S. Lam, The University of Texas at Austin; Jonathan M. Smith, University of Pennsylvania

Yun Mao presented S4, a routing protocol for large wireless sensor networks that achieves small state and small stretch. Yun noted that there are numerous challenges in coming up with a point-to-point routing protocol for wireless sensor networks, owing to limited resources and to the RF phenomenon. Also, there is a huge debate going in the community on whether the routing protocol should have a small state or small stretch (path length compared to the optimal path length). Yun noted that for wireless sensor networks, state and stretch were related, as routing protocols, which aim at providing low stretch, invariably require more state. Yun noted that shortest path routing gives optimal stretch but requires $O(n)$ state whereas hierarchical routing, which requires only O[square_root(n)] state, gives paths with large stretch. Other proposals, such as geographical routing and virtual coordinate routing, are also unable to avoid the state vs. stretch tradeoff.

S4 uses the theoretical ideas from the compact routing algorithm to achieve a small state (i.e., of O[square_root(n)]) with a constant bound on the worst-case stretch of 3. S4 uses two types of nodes: beacon nodes and regular nodes. The beacon nodes are O[square_root(n)] in number and know how to route to the regular nodes close to it (cluster), whereas each of the regular nodes knows how to reach the beacons. S4 uses two rules to route in the network: Inside a cluster, route using the shortest path; outside a cluster, route to the beacon closest to the destination node. The challenges facing S4 are to ensure that no flooding takes place and that each node maintains its routing state for reaching the beacons and to provide resiliency to link and node failures.

Yun presented the evaluation of S4 using high-level simulations with no loss and reliable nodes, TOSSIM packet-level simulations using lossy links, and the mica2 testbed of 42 nodes. The authors used the Beacon Vector Routing (BVR) protocol from NSDI '05 as the benchmark for comparison with S4. The results of the simulation and evaluation showed that S4 had smaller average stretch and variation compared to BVR. S4 also achieves smaller state and is unaffected by obstacles in the paths.

There was a question regarding the complexity of the code and the amount of memory required by the code itself. Yun replied that the complexity of the code for S4 was similar to that for BVR; however, they didn't have any actual numbers. There was also an inquiry on how the beacons were placed and if there was an intelligent method for placing them. As to whether there were any real applications that used 4,000 or more sensor nodes, Yun replied that there were no such applications at present, but he said they might be seen in the near future.

The software can be found at http://www.cs.utexas.edu/~lili/projects/s4.htm.

■ ***A Location-Based Management System for Enterprise Wireless LANs***

Ranveer Chandra, Jitendra Padhye, Alec Wolman, and Brian Zill, Microsoft Research

Jitendra Padhye presented the design and evaluation of a management system for wireless LANs. The system has been deployed and is in use on one of the floors in the Microsoft building. Although there has been lot of work in the area of managing wireless LANs, they each have their shortcomings: Some cannot cover the area properly, whereas others are too expensive to deploy or are not scalable.

The wireless management system presented was deployed on the DAIR platform, which consists of attaching air monitors to ordinary desktops in offices. These air moni-

tors are used to collect wireless data and send it to a central database. DAIR has several advantages: Since desktops are ubiquitous in practically every office, this leads to a dense deployment with no extra cost. It is also robust, as desktops are always up. Also, such a scheme can use very simple algorithms for determining the location of the wireless nodes, measuring the quality of the wireless signal, etc. Furthermore, storing the data collected from the sensors in a central place allows historical analysis of data. In the talk, Jitendra presented the use of DAIR in estimating the transmission rate obtained by wireless clients at various locations on the office floor.

The aim of DAIR is to locate the clients at the granularity of an office on the floor and, since the air monitors were deployed on the desktops in the offices, it was a simple task to determine to which office the air monitor belonged. Also, to pinpoint the location of the client at the granularity of an office only required the use of simple algorithms, such as choosing the air monitor receiving the strongest signal or using the spring-and-ball method or the strongest AM method. The air monitors also intercept the packets from the clients to the nearest AP. For each conversation between the client and the AP, the system simply chooses for analysis the data from the air monitor that received the highest number of packets for that conversation. The authors were able to detect an AP that was not functioning properly using this system and were able to alert the network operators and get the AP fixed. The system also detected an area of poor coverage in the office and reported it to the network operators.

Finally, Jitendra noted that the attachment of air monitors on the desktops causes an additional load of about 2% to 3% on the desktop and contributes additional traffic of less than 10 kbps. To the question of whether the authors evaluated any other technique besides choosing the data from the air monitor that received the highest number of data packets, Jitendra replied that they did not, since they didn't see much packet loss with their technique of using the one that saw the highest number of packets. There was also a question regarding interaction between floors, to which Jitendra replied that they did investigate that possibility and that it was easy to detect with their scheme if the client was connecting to an AP on another floor, although their evaluation was concentrated on a single floor. Hari Balakrishnan asked how their techniques compared to the ones used by cellular companies. Jitendra explained that in a cellular network, if the cell phones themselves are used as the monitors, there is no way of telling where the cell phone (mobile client) is located.

POSTER SESSION

Summarized by Eric Eide eeide@cs.utah.edu

- ***Distributed Data Management for Storage-centric Sensor Networks***

  Devesh Agrawal, Gal Niv, Gaurav Mathur, Tingxin Yan, Deepak Ganesan, Prashant Shenoy, and Yanlei Diao, University of Massachusetts, Amherst

Devesh summarized StonesDB, a database system for wireless sensor networks. Because emerging sensor network devices have increasingly high-capacity and energy-efficient flash memories, StonesDB stores collected data on the nodes of a sensor network. The complete StonesDB system has two levels. The lower layer consists of the sensor network nodes, each of which runs a local database. The upper layer, which receives and routes user queries to appropriate sensor nodes, consists of resource-rich proxies that implement distributed data management services atop the sensor network. The StonesDB architecture is intended to optimize energy efficiency at its sensor nodes, and this requires careful and novel implementations of many database components. A second challenge lies in designing StonesDB to support a variety of sensor network platforms with varying resource constraints, which requires a corresponding variety of design tradeoffs and optimization points.

- ***Lightweight OS Support for a Scalable and Robust Virtual Network Infrastructure***

  Sapan Bhatia, Marc Fiuczynski, Andy Bavier, and Larry Peterson, Princeton University

Sapan presented recent work on VINI, a virtual network infrastructure designed for evaluating new protocols and services. VINI seeks to offer a high-performance network testbed, one that supports many concurrent experiments and that also provides each experiment with fine control over issues such as routing and traffic shaping. Such control requires OS support that is not found in testbeds such as PlanetLab, which isolates experiments from each other via Linux VServers. Sapan's poster focused on two areas of work that allow VINI to overcome the VServer "virtualization barrier." The first was more complete virtualization of the Linux network stack and the initial deployment of this OS work onto VINI. The second was a secure bootstrap system for the VINI control plane, which helps to address security concerns that arise from VINI's virtualized network stacks.

- ***Real Time-Sharing in Emulab through Preemption and Stateful Swapout***

  Anton Burtsev, Prashanth Radhakrishnan, Mike Hibler, and Jay Lepreau, University of Utah

Anton and Prashanth presented work that allows experiments within the Emulab network testbed to be "swapped out" without losing state. An Emulab experiment is analo-

gous to a UNIX process: Its resources include a set of testbed-managed hosts, and its state includes the contents of the memory and disks on those hosts. Currently, when the resources for an experiment are released, the contents of node-local memories and disks are lost. Thus, experiments are normally swapped out only when they are complete. Anton and Prashanth are working to preserve node-local state, however, which will allow Emulab experiments to be swapped out (and back in) more freely. Their goal is for such transitions to be transparent to the software that executes within the experiment and, except for scheduling delays, transparent to users of the testbed as well. The poster outlined the many challenges of stateful swapout as well as their current solutions.

■ ***The Case for Conditional Link Metrics and Routing***

Saumitra M. Das, Purdue University; Yunnan Wu and Ranveer Chandra, Microsoft Research, Redmond; Y. Charlie Hu, Purdue University

Saumitra presented this poster, which described the need for and potential benefits of "conditional link metrics." A conditional metric is one that varies according to context: For example, the cost of sending a packet over a given link may depend on the set of links that the packet has already traversed. Such metrics provide ways to express interdependencies within a network. Among other examples, Saumitra proposed using a conditional metric to capture the dependency between network coding and routing. Network coding is a link-layer technique that can reduce wireless transmissions, but the opportunities for network coding depend on traffic patterns and hence depend on routing decisions. Saumitra and his colleagues have implemented two systems that utilize conditional metrics to help practical network coding and multiradio networks, with promising results.

■ ***Residential Broadband Networks: Characteristics and Implications***

Marcel Dischinger, Andreas Haeberlen, and Krishna P. Gummadi, Max Planck Institute for Software Systems; Stefan Saroiu, University of Toronto

Marcel presented current work in the measurement of residential broadband networks. Such networks are increasingly important for emerging Internet applications, such as VoIP and P2P systems, but there is little data that characterizes these networks at scale. Marcel and his colleagues obtained measurements from 1,500 residential broadband hosts spread over 11 major cable and DSL ISPs. These measurements were acquired through probe trains and required no cooperation from the measured hosts. The poster summarized the collected data, which shows that residential networks are often quite different from academic networks. For instance, the data shows that some ISPs allow short bursts of traffic—perhaps to speed up Web page downloads—but significantly rate-limits large flows.

■ ***Characterizing and Replaying Proprietary Workloads***

Archana Ganapathi, Armando Fox, and David Patterson, University of California, Berkeley

Archana described the problems faced by developers who must predict or test the behavior of a networked system. For example, the maintainers of a commercial Web service may need accurate models of their system to estimate future resource demands, but generally they have insufficient tools and test resources. Academic researchers, in contrast, are often interested in analyzing production systems but are unable to obtain actual application traces from companies. Archana described an approach for solving both problems: Use machine-learning techniques to generate artificial but realistic workloads that drive systems into desired behaviors. The poster summarized two tools in development: AWE-Gen, which generates synthetic workloads from actual trace data, and AWE-Sim, which replays the workload against the target system, thereby creating the desired system conditions.

■ ***A Deductive Framework for Programming Sensor Networks***

Himanshu Gupta and Xianjin Zhu, Stony Brook University

Xianjin's poster described a novel programming methodology for wireless sensor networks, one based on logic programming. Many sensor network applications are designed to collect facts about the world and process queries against those facts. This design closely matches the main structuring principles of logic programming languages, suggesting that logic programming can be a good fit for sensor network computing. A user-written logic program specifies behaviors in a declarative and high-level fashion. Ideally, these programs can be automatically compiled into distributed and resource-efficient code for the nodes within a sensor network. The goal of Xianjin's research is to design and implement the framework that makes this vision a reality. In particular, this work requires new and energy-efficient techniques for evaluating logical "joins": the process of searching for data that is needed in order to satisfy one or more queries.

■ ***Network Troubleshooting: An In-band Approach***

Murtaza Motiwala, Georgia Institute of Technology; Andy Bavier, Princeton University; Nick Feamster, Georgia Institute of Technology

Murtaza presented Orchid, an in-band network path diagnosis system for locating faults in a packet-based network. Most network diagnosis tools produce their own network traffic to locate faults. This traffic is out-of-band with respect to normal application traffic, and, as a result, it can fail to detect a variety of faults that affect application traffic. Orchid, however, inserts a diagnostic header into the packets that carry application data. When a flow begins, Orchid transmits a probe packet to record the addresses of routers along the flow's network path. After this, routers along the path use the Orchid header within data packets

to record faults. Orchid-enabled routers need only a small amount of state (a single counter) per active flow. Experiments show that the current Orchid prototype, implemented with Click and deployed on PL-VINI, can accurately diagnose many faults with only small network overhead.

■ ***Amazon S3 for Science Grids: A Viable Solution?***

Mayur Palankar, Ayodele Onibokun, and Adriana Iamnitchi, University of South Florida; Matei Ripeanu, University of British Columbia

Amazon's Simple Storage Service (S3) offers pay-as-you-go online storage, and as such, it provides an alternative to in-house mass storage. In this poster, Mayur and his colleagues evaluated S3 as a storage facility for the DZero Experiment, an international high-energy physics collaboration. Traces from the DZero community over 27 months show that 560 users worldwide transferred 5.2 PB through DZero. Mayur and his colleagues characterized S3: They observed availability and data access performance, and they evaluated the feasibility, performance, and costs of a hypothetical S3-supported DZero collaboration. They concluded that S3 could be a viable storage system for DZero in terms of availability and performance, but that it would be expensive—in excess of $1.1 million per year. Costs could be reduced by using BitTorrent along with S3, exploiting data usage and application characteristics to improve performance. Finally, Mayur noted that S3's current security architecture is inadequate for science collaborations such as DZero in terms of access control, support for delegation and auditing, and built-in assumptions of trust.

■ ***XMon-BGP: Securing BGP Using External Security Monitors***

Patrick Reynolds, Oliver Kenney, Emin Gün Sirer, and Fred B. Schneider, Cornell University

Patrick presented a low-cost and incrementally deployable way of securing the Border Gateway Protocol (BGP). BGP connects autonomous systems within the Internet: It constitutes critical infrastructure but has well-known security problems. Previous attempts to secure BGP, entailing new routers or extensive modifications to router operation, have not been widely deployed. XMon-BGP proposes to secure legacy routers that employ (unsecured) BGP by deploying external security monitors (XMons). An XMon examines traffic to and from a legacy device and checks it for conformance against a security specification. It can thus protect against compromised routers, misconfigurations, and even insider attacks. Multiple XMons can communicate via an overlay to compensate for autonomous systems that have not deployed an XMon. The XMon software runs on a trustworthy computing platform (Nexus, the subject of a companion poster) that can vouch for the correctness of the XMon outputs. Experiments showed that XMon-BGP works well, that it has no trouble keeping up with BGP traffic, and that a relatively small deployment of XMon-BGP could secure a majority of all Internet routes.

■ ***Nexus: A New Operating System for Building Trustworthy Applications***

Alan Shieh, Dan Williams, Kevin Walsh, Oliver Kennedy, Patrick Reynolds, Emin Gün Sirer, and Fred B. Schneider, Cornell University

Dan described the design and implementation of Nexus, a new operating system for trustworthy computing. Traditional operating systems lack abstractions and mechanisms for using increasingly available secure coprocessor hardware. In contrast, Nexus leverages such hardware to support trustworthy applications that have strong behavioral guarantees, without restricting users to particular software applications. One of the new mechanisms in Nexus is "active attestation," which securely captures properties of software components. This can be used for both local and remote access control. Dan complemented his poster with a live demonstration of applications on Nexus. A media server checked active attestation labels on requests to ensure that the media players (i.e., clients) would not leak the content to disk. Active attestation labels also distinguished user-keyboarded messages from script-generated spam. Finally, Dan demonstrated Nexus's support for legacy applications by running Linux, X Windows, and various applications such as Firefox and Thunderbird on Linux atop Nexus.

■ ***The SPINDLE Disruption Tolerant Networking Project***

Christopher Small, Rajesh Krishnan, and the members of the SPINDLE team, BBN Technologies

Christopher presented the SPINDLE project, which is developing new technologies for disruption-tolerant networks (DTNs). Commonplace networks, such as those based on TCP/IP, require stable end-to-end paths in order to operate. To relay messages, there must be a complete path from source to destination (and back). In contrast, a disruption-tolerant network supports reliable communication in more hostile and poorly connected environments. The SPINDLE project is developing new routing algorithms for DTNs that take disconnection and link discovery into account. In addition, they are developing techniques that use caching, distributed indexing, and content-based data retrieval to improve access to data in the face of network disconnections. An application can use a declarative specification to describe the routing, resource management, and other policies that a DTN should use in handling its data.

■ ***Scaling Full-Mesh Overlay Routing***

David Sontag, Massachusetts Institute of Technology; Amar Phanishayee and David Andersen, Carnegie Mellon University; David Karger, Massachusetts Institute of Technology

David Sontag described recent work that improves the scalability of routing in one-hop overlay networks. Routing in existing overlay networks, such as RON, scales poorly because every node communicates with every other node.

David presented a new algorithm that scales much better while still supporting best one-hop routing over the complete network. In the new algorithm, every node measures the paths to all of its neighbors, but it sends that information only to a subset of the other nodes in the network. The trick is that these subsets are chosen so that, for any two nodes A and B in the network, there is at least one node C that receives the neighbor data for both A and B. Thus, the common node C can tell A and B about the best path between A and B. David and his colleagues are now evaluating the effectiveness of their new algorithm in the RON testbed. Among other tasks, they are investigating the resilience of the new algorithm to node and link failures.

■ ***Efficient Cooperative Backup on Social Networks***

Dinh Nguyen Tran and Jinyang Li, New York University

Dinh presented BlockParty, an online backup system for cooperating groups of friends. The benefits of online backups are well known, and P2P networks provide convenient, online, and physically distributed storage. However, implementing a backup system within a traditional P2P network is difficult because of node churn, misaligned incentives, and ill-suited models of trust. BlockParty therefore allows each user to specify the other users with which he or she will cooperate. In practice, users choose to cooperate if they are also real-world friends. In comparison to other P2P systems, BlockParty has limited choices for storing data. Therefore, a primary concern of BlockParty is efficient utilization of disk space. Dinh described their scheme for coding data blocks, which saves spaces on BlockParty hosts. Finally, although BlockParty users trust each other not to deny service, each node periodically (and efficiently) verifies that its neighbors hold the expected backup data. If loss is detected, BlockParty undertakes repairs. The project software is available at http://www.news.cs.nyu.edu/friendstore/.

TOLERATING FAULTS AND MISBEHAVIOR

Summarized by Yun Mao (maoy@cis.upenn.edu)

■ ***Beyond One-Third Faulty Replicas in Byzantine Fault Tolerant Systems***

Jinyuan Li, VMware, Inc.; David Mazières, Stanford University

Jinyuan Li stated that a Byzantine fault tolerant (BFT) system will behave correctly when no more than f out of $3f + 1$ replicas fail. In particular, BFT aspires to two properties: consistency (or safety), meaning all operations execute as if they are sequentially conducted on replicated state machines, and liveness, meaning protocols can make progress even with malicious replicas. When an attacker controls more than f failures, in a traditional BFT system, the system behavior is totally unexpected. Jinyuan argued that there is a large space between complete correctness and arbitrary failures. He first introduced the fork consistency, which is a relaxation of the linear consistency. He used a card-swipe access control service as the application to demonstrate that fork consistency is useful because it leaves replicas or clients with "forked views," and the misbehavior will eventually be revealed via out-of-band communication. The misbehavior that is nonerasable can be used as proof of attack. Jinyuan then showed that it is possible to still achieve fork consistency when no more than $2f$ failures happen.

The BFT2F protocol is based on the PBFT protocol. The intuition is that each replica keeps its execution history and the client waits for $2f + 1$ matching replies instead of $f + 1$ in PBFT. The problem of achieving fork consistency is that the protocol has to be a two-round protocol, and the liveness property is sacrificed if clients crash between rounds. The communication overhead is also not negligible. To avoid using a two-round protocol, Jinyuan said it is necessary to further relax the consistency guarantee to a fork * consistency. In a fork * consistency, it is possible for an honest replica to execute an operation out of order, but at least any future request from the same client will make the attack evident. Later, the optimization of the BFT2F protocol to achieve fork * consistency was discussed, and the performance penalty was studied. Finally, he showed that it is possible to generalize BFT2F to BFTx. This makes it possible for the system designer to tune the tradeoffs among consistency, liveness, and failure handling.

Someone from MIT asked how much correctness is sacrificed in the deployment. The answer is that the correctness is exactly guaranteed as either the fork or fork * consistency model specifies. Petros Maniatis from Intel Research at Berkeley asked about whether the two different weaker consistency models make a difference in the application. Jinyuan answered that the main differences are in performance and liveness. In an application like the card-swiping example, or other typical access control applications, these properties might be desirable.

■ ***Ensuring Content Integrity for Untrusted Peer-to-Peer Content Distribution Networks***

Nikolaos Michalakis, Robert Soulé, and Robert Grimm, New York University

Nikolaos described a security problem from the successful P2P CDN systems: What if a malicious peer changes the content in an arbitrary way and sends it to the client? What if you want to see a sweet, good-looking Britney Spears but the peer gives you a bald, crazy one? The goal of this paper is to detect (but not prevent) bad replicas for both static and dynamic contents. However, this goal is not as easy as it appears initially. Nikolaos tried to go through the entire design space that includes the existing solutions and found some problems: Client verification could be quite expensive for small devices; if the client downloads multiple copies and accept the majority, the load of CDN

is at least tripled and can only tolerate less than 50% misbehaving replicas; using other peers as spies could put them into a spy list by attackers easily and diminish the effectiveness; if volunteers forward bad content to a verifier, it is hard to draw a conclusion as to who corrupted it. All these lessons suggested that attestation is necessary. However, if only a few trusted verifiers are selected, the system doesn't scale well because the load on the verifiers is as much as the load of the entire CDN. In sum, the lessons learned are that the solutions that preserve existing servers and clients are not sufficient. Servers must sign their responses and clients must verify them. Replicas must produce attestation records for accountability. Sampled forwarding from clients is desirable to reduce network traffic.

Then Nikolaos presented the idea of the paper, "Repeat and Compare." Repeat essentially simulates the response-generation process. The requirement is to get rid of all nondeterminism so that, with the same external inputs and parameters, the identical results can be repeated. He argued that nondeterminism is possible to achieve in Web-related applications. In fact, in many cases, given the random seed, the pseudo random generator works just fine. The Compare part consists of two stages: forwarding attestation records to verifiers and then detecting misbehaving replicas. The attestation records are forwarded to verifiers via clients with probability p to a randomly selected verifier. Checking the freshness of an object is also a little tricky, requiring a trusted global synchronized clock to make timestamps to detect misbehavior. Eventually, the bad replicas are published based on a punishment policy to reduce the incentive to cheat.

Ryan Peterson from Cornell University asked about how many parties in the CDN system need to be changed in order to support Repeat and Compare. Nikolaos said that changes made at the client, verifier, and server are needed. The hard part is at the client side. However, it is inevitable that the client must be changed, because it has to be able to reject the misbehaving replica.

■ ***TightLip: Keeping Applications from Spilling the Beans***

Aydan R. Yumerefendi, Benjamin Mickle, and Landon P. Cox, Duke University

Confidentiality is harder to achieve than you might think! Aydan made two points at the outset of his talk. First, access control misconfigurations are widespread. A Kazaa usability study found that many users share their entire hard drive with the rest of the Internet. Second, even if you have perfect security and configuration, your privacy will only be as secure as your least-competent confidant who shares the information with you. Unfortunately, cryptography, secure communication channels, and intrusion detection systems do not prevent these problems.

Aydan proposed a new approach to prevent information leaks: a privacy management system called TightLip. TightLip takes a different path from conventional security software, in that it allows users to define what data is important and who is trusted regardless of the software that accesses them. There are three key challenges to TightLip: first, how to identify sensitive files and trusted hosts and protect that metadata; second, how to track the flow of the sensitive data through an OS and identify potential leaks; third, how to develop policies for dealing with the leaks. TightLip differs from most of the related work because it requires no change to the applications and hardware, and only minor modifications to the operating system.

The key concept in TightLip is a new OS object: doppelgänger processes. Doppelgängers are copy processes that inherit most of the state of an original process. They are spawned when a process tries to read sensitive data. The kernel returns sensitive data to the original process and scrubbed data to the doppelgänger. These two processes run in parallel. As long as the outputs for the two processes are the same, the original's output does not depend on the sensitive input with very high probability. The input/output are monitored by the system call arguments and result. When a difference is found, TightLip invokes a policy module, which can direct the OS to fail the output, ignore the alert, or even swap in the doppelgänger process. The scrubbing process depends on the data format. By default, it replaces each character from the sensitive data source with "x." Finally, by running several conventional benchmarks, Aydan showed that TightLip prototype overhead is quite modest.

Someone from Georgia asked about whether you can still forward emails when the email is marked sensitive. Aydan responded that it depends on the policy module. Then the concern from the questioner was that the configuration of the policy module could be as complicated as the applications.

Amin Vahdat from UCSD expressed some concern that as the data flows inside the application, more and more output might depend on the sensitive data so the false-positive rate could be high. Aydan said it's a common problem for all information flow analysis. However, for a subset of applications such as Web and P2P clients, the flows are quite simple and the false-positive rate is low.

Emin Gün Sirer from Cornell University asked what factors make it harder or easier for TightLip to scrub data. Aydan said it totally depends on the application. For example, scrubbing an email text is fairly easy, but to scrub a binary matrix might be very hard.

NETWORK MEASUREMENT

Summarized by Prashanth Radhakrishnan (shanth@cs.utah.edu)

■ ***Peering Through the Shroud: The Effect of Edge Opacity on IP-Based Client Identification***

Martin Casado and Michael J. Freedman, Stanford University

Martin Casado, who presented this talk, started by speaking about the dependence of IP addresses on client identification in today's Internet. He noted that edge technologies such as NATs, proxies, and DHCP obscure the client's identity and thus motivated eliminating the effects of these on server identification of clients.

Their approach was to use the Web as a measuring platform to measure the effect of edge opacity, perform analysis on the results, and develop methods for servers to eliminate these effects.

To measure the Internet edge, they use active content execution at the clients. Clients are made to execute the active content in two ways: by "bugging" existing Web pages and by redirecting a percentage of CoralCDN's requests through measurement servers. From their measurements, about 60% of the clients were behind NATs and most NAT sizes were quite small. Also, IP deallocation resulting from DHCP was slow. Moreover, 15% of the clients were behind proxies, which were generally larger than NATs. Martin concluded that proxies pose a major problem for IP-based client identification and then discussed techniques for real-time proxy detection for servers.

During the Q&A session, Amin Vahdat (UCSD) asked if they had made a comparison of network characteristics (RTT, bandwidth, loss-rate, etc.) between clients behind proxies and NATs versus those directly connected. Martin said that the data was generally a bit messy for such analysis, but he noted that RTTs through proxies were longer and that NATs didn't seem to affect the RTT much.

Justin Cappos (University of Arizona) pointed out that Martin had mentioned DNS blacklisting as one of the motivations for this work, so he asked if Martin had any idea on the number of SMTP mail servers that use proxies. Martin replied that their measurements were just for the Web and do not include mail servers. John Agosta (Intel) observed that an adversary could potentially use this system to discover IP addresses behind a proxy to create a hitlist. Martin acknowledged that possibility but noted that in their system clients had to explicitly talk to the servers (thus reducing the probability). Tom Anderson (University of Washington) asked for Martin's comment on the ethics of gathering information by surreptitiously running "spyware" on clients. Martin answered that they have stayed well within the security model. He added that there was an implicit contract that when you go to a Web site you could execute things on the Web site and he further cited the example of Google Analytics, which gathers statistics by similar means.

■ ***A Systematic Framework for Unearthing the Missing Links: Measurements and Impact***

Yihua He, Georgos Siganos, Michalis Faloutsos, and Srikanth Krishnamurthy, University of California, Riverside

Yihua He gave this talk on finding the missing links in current Internet topology at the AS level. He explained the need for an Internet topology and noted that the topologies derived using the current state-of-the-art techniques are incomplete because they underestimate the peer-to-peer links between ASes.

In this work, they collect data about AS edges using multiple methods such as existing BGP routing table dumps, exploring Internet routing registry, and inferring Internet Exchange Point (IXP) participants. All the links are validated by reverse traceroute. As a result, they found 40% more AS links and 300% more peer-to-peer AS links, most of which are at IXPs. Yihua noted that as a result of these "new" peer-to-peer links, the ASes could avoid using their providers to reach many destinations, lowering ISPs' costs and increasing revenue.

During the Q&A session, Vytautas Valancius (UIUC) asked about the total number of distinct edges collected. Yihua said that it was roughly 50,000.

Nick Feamster (Georgia Tech) noted that their conclusion about most of the peer-to-peer links being at exchange points was interesting. He then asked whether Yihua had an idea of how close the inter-AS connections are at specific exchanges. Yihua said that the measurements for this are inaccurate because the traceroutes are done only from selected points. Nick also asked why they did not consider routes collected at exchange points, especially since the missing edges are at the exchange points. Yihua acknowledged that it was a good idea to try out.

EMULATION AND VIRTUALIZATION

Summarized by Prashanth Radhakrishnan (shanth@cs.utah.edu)

■ ***The Flexlab Approach to Realistic Evaluation of Networked Systems***

Robert Ricci, Jonathon Duerig, Pramod Sanaga, Daniel Gebhardt, Mike Hibler, Kevin Atkinson, Junxing Zhang, Sneha Kasera, and Jay Lepreau, University of Utah

Robert Ricci presented the talk on Flexlab. He contrasted two popular methods for evaluating networked systems, namely, "emulators" that provide control and reproducibility but lack realism and "overlay testbeds" that provide realism but lack control and reproducibility. Rob introduced Flexlab as a hybrid method that combined the merits of emulators and overlay testbeds, while eliminating the demerits.

Rob then described the Flexlab architecture. An application runs inside the emulator hosts along with a monitor that reports the application's network operations. The application's traffic passes through the path emulator, controlled by a pluggable network model. The network model may take its input from a measurement repository or may be driven in real time by the application's behavior reported by the monitor.

Given that accurate modeling of the Internet is still an open problem, they explore the approach of modeling the Internet, in real time, from the application's perspective. The application's behavior running inside the emulator (Emulab) is used to generate traffic in the overlay testbed (PlanetLab) and collect Internet measurements. The network conditions experienced by the PlanetLab traffic is applied to the application's path emulator, giving the impression that the Emulab hosts communicate across the Internet.

The evaluation results, from running microbenchmark iPerf, indicated that FlexLab could accurately emulate Internet traffic conditions. Through a case study with BitTorrent, they showed that FlexLab was able to remove some of the artifacts of PlanetLab host conditions (namely, CPU availability) that hurt BitTorrent throughput.

During the Q&A session, Dave Marwood (Google) asked whether users of the application-centric Internet modeling platform were limited to the network effects considered in FlexLab. Rob answered that users were limited to those, but he noted that the other network effects will be seen on latency, bandwidth, and packet-loss measurements. He said that as future work they plan on adding run-time validations to the system. Mark Chiarini (Tufts University) had a question on the throughput spikes of BitTorrent seen only in FlexLab, but not on PlanetLab, early on in the experiment. Rob said that it may be due to BitTorrent trying to ramp up its download rates. He noted that those spikes depended on the amount of CPU available and postulated that such spikes don't happen in PlanetLab because CPU is scarce.

Peter Druschel (MPI-SWS) asked whether, to use FlexLab, PlanetLab is always required to be online. Rob answered that their paper talks about two offline models that do not require PlanetLab. He further added that in future work they plan to record the application's Internet measurements from a run and replay it during its future runs.

■ ***An Experimentation Workbench for Replayable Networking Research***

Eric Eide, Leigh Stoller, and Jay Lepreau, University of Utah

Eric Eide began by saying that experiment testbeds such as Emulab provide resource management, but there still is a need for managing the experiment workflow. Workbench helps researchers manage their activities, software artifacts, data, and analyses and enables them to navigate through experiment history and replay or branch from any point. Effectively, Workbench is fundamental for repeatable research. They have evolved the Emulab testbed management software to be the basis for experimentation with Workbench.

Eric gave an overview of the current experiment lifecycle of Emulab: Users create persistent experiment definitions containing logical resources, "swap-in" experiments to allocate physical resources, and "swap-out" experiments to deallocate physical resources. He noted that in its current state Emulab confuses experiment definition with instance.

Workbench breaks experiments into multiple abstractions. A "template" is now the experiment definition. It is a versioned repository that stores the topology, parameters, and software. Creating a template "instance" involves assigning values to the parameters and allocating physical resources. A "run" is a context for doing a unit of work. An "activity" is a collection of processes, workflows, etc., that execute within a run. Finally, a "record" is a flight recorder of a run that saves all the things produced within a run in the database. "Record" is used for replaying experiments.

Given that this is a user tool, Eric discussed a couple of user case studies. He mentioned that the experimenters who used Workbench found the transition to Workbench relatively easy and the abstractions intuitive. He also noted that Workbench served as a useful platform for communication of results. Finally, Eric discussed a couple of problems they were facing with storage and node failures in PlanetLab that they plan to address in future work.

During the Q&A session, Pankaj Thakar (VMware) asked whether they had considered use of virtual machines for the Workbench activities. Eric agreed that it would be a good idea to consider. Mark Chiarini (Tufts) commented that Workbench is an excellent way of capturing scientific process and making research reproducible. Peter Druschel (MPI-SWS) observed that similar to Workbench's event capturing mechanism or experiment replay, low-level events at finer granularity need to be captured for debugging purposes. He asked if Eric viewed these as related or as separate concerns. Eric replied that they represent a continuum and that the latter is something that would best be implemented as part of the testbed infrastructure that Workbench could leverage. He also mentioned that this is currently a work in progress.

■ ***Black-box and Gray-box Strategies for Virtual Machine Migration***

Timothy Wood, Prashant Shenoy, and Arun Venkataramani, University of Massachusetts Amherst; Mazin Yousif, Intel, Portland

Timothy Wood presented this talk on Sandpiper, a system targeted at virtualized data centers. Sandpiper monitors resource usage and automatically detects and removes hotspots by leveraging virtual machine (VM) migration and dynamic resource allocation.

Timothy gave an overview of Sandpiper's architecture. A per-physical-machine agent monitors VM (CPU, memory, and network) resource usage and reports to the central control plane. They explore two approaches to gather VM resource usage information, namely, the (application- and OS-independent) black-box techniques and the (application- or OS-specific) gray-box techniques. The central server has a hotspot detector to decide when to migrate VMs using a combination of resource thresholds and historical data trends. It also includes a profiling engine to decide on the amount of resources to allocate to VMs, again based on historical data. Finally, the migration manager decides where to migrate VMs to mitigate hotspots. This decision is based on heuristics that take into account the physical machines' percentage resource utilization and the cost of migration measured in terms of the VM memory size.

Timothy then spoke about their evaluation results. The results demonstrated the effectiveness of migration and also showed a scenario (detection of memory hotspots) where black-box techniques were insufficient and gray-box techniques had to be used. Timothy concluded with brief descriptions of related and future work.

During the Q&A session, Diwaker Gupta (UCSD) noted that they seem to treat all resources as equal, but in reality applications may be more dependent on some specific resource. Timothy agreed that this was true, because they do not include application-specific knowledge in the decision. Diwaker also pointed out that gray-box techniques could be employed from outside VMs. Timothy said that they were planning on doing that in the future.

Pankaj Thakkar from VMware observed that in Sandpiper a VM could potentially keep gaining memory. Timothy acknowledged that in a real deployment they would need some cap on the allocations. Ryan Peterson (Cornell) asked whether they take into account factors such as frequency of flash crowds. Timothy answered that flash crowds were assumed to be rare and that their profiling takes recent flash crowds into account.

DEBUGGING AND DIAGNOSIS

Summarized by Prashanth Radhakrishnan (shanth@cs.utah.edu)

- ***Life, Death, and the Critical Transition: Finding Liveness Bugs in Systems Code***

  Charles Killian, James W. Anderson, Ranjit Jhala, and Amin Vahdat, University of California, San Diego

  Awarded Best Paper!

Charles Killian gave this talk on finding bugs in distributed systems by using model checking techniques. He started with a brief background on model checking, up to the state of the art where model checking is used on unmodified systems code to detect bugs that violate safety properties. With an illustration of the Pastry system, he argued that liveness properties (i.e., conditions that should always "eventually" be true) are richer and more natural for expressing errors in distributed systems.

Since a distributed system's state space explodes exponentially, exhaustive search techniques are insufficient to find violations of liveness, which are not expressed with short or even bounded executions. Charles introduced the notion of dead state space from which liveness can never be achieved, and this state space corresponds to errors. To find the dead state space, they combine the exploration of existing model checkers with random executions from every state encountered. An execution that ended in a dead state may have early transient states, which can lead to live states. To help identify the liveness error, they automatically find the critical transition that pushed the system into the dead state using a binary search between an identified transient and a dead state. Charles then detailed how they employed this technique to find a bug in the Pastry system.

Charles briefly spoke about the implementation of their software model checker, MaceMC, which is built over MACE (a language for defining event state systems). The user needs to set up the system in MACE by defining the system states, events, and transitions. Charles concluded with the lessons they learned, including the insight that it is possible to learn new safety properties from violations of liveness properties and the kinds of bugs distributed systems were prone to.

During the Q&A session, Ken Birman (Cornell) pointed out that it may be difficult to find liveness properties in large distributed systems. For instance, large DHT-based systems may undergo continuous churn and may "eventually" never satisfy the liveness property. Charles said that there could be other liveness properties for which this technique could be effective, but they have not studied such large systems yet. Mark Chiarini (Tufts University) observed that dead state identification is a reformulation of the halting problem and confirmed that it was determined experimentally.

- ***WiDS Checker: Combating Bugs in Distributed Systems***

  Xuezheng Liu, Wei Lin, Aimin Pan, and Zheng Zhang, Microsoft Research Asia

Xuezheng Liu presented this talk on WiDS checker, a unified framework to check distributed systems through simulation and reproduced runs from real deployment. Their approach was to use library-based deterministic replay coupled with predicate checking. Xuezheng started with an example illustrating the complexity of bugs in distributed systems.

Their system is implemented in the middle layer between the distributed application and the OS, which provides the

ability to intercept OS API, inject failures, simulation, and replay capabilities. All nondeterminism is logged and Lamport clocks are used for consistent group replay. During replay, WiDS interprets the entire distributed system as a sequence of events ordered by the "happens-before" relation. The entire system is replayed in a simulated process. Predicate checking happens at event (message receive, timer expiration) boundaries, includes liveness properties, and is decoupled from the replay (i.e., predicate checking happens on separate state copies). They found 12 bugs in four well-studied distributed systems (Paxos protocol, Boxwood, BitVault, and Chord), including a specification bug in Paxos.

Xuezheng noted that the downside of their work is that applications need to be ported to the WiDS middle layer. He concluded with a comparison of WiDS to other related systems. During the Q&A session, Mike Dahlin (University of Texas, Austin) asked about the runtime overheads involved. Xuezheng explained that their API interception layer is lightweight and thus their overheads were minimal.

■ ***X-Trace: A Pervasive Network Tracing Framework***

Rodrigo Fonseca, George Porter, Randy H. Katz, Scott Shenker, and Ion Stoica, University of California at Berkeley

Rodrigo Fonseca presented the talk on X-Trace, a pervasive network tracing framework. X-Trace gathers end-to-end execution traces, including various applications and the network stack, across administrative domains in the wide area. The goal is to capture the causal structure of a task, which is a specific activity that includes many operations at different abstraction levels, components, and administrative domains. Task ID and operation ID are propagated along the edges and nodes report the operations to a reporting infrastructure. Rodrigo noted that there are no layering violations in X-Trace.

X-Trace requires network support for opaque extension headers and device support for metadata propagation and reporting. The cost of X-Trace is minimal given the limited tracing metadata and asynchronous reporting. Also, nodes inside an administrative domain could send the reports to a domain-local repository for desensitizing information. Since the tracing functionality is independent across multiple layers, X-Trace supports partial and incremental deployment.

Rodrigo illustrated the working of X-Trace in a multilayered system with multiple node failures. The X-Trace software, APIs, and a public reporting service are available for general use.

During the Q&A session, there was a question on the overheads involved for normal operation. Rodrigo answered that tracing could be selectively enabled at different layers. Mark Chiarini (Tufts University) asked about sending the metadata in-band, in the case of UDP, for example. Rodrigo replied that they may not want to do it for fear of introducing extra bytes in the application stream if care is not taken when stripping the metadata at the other end. Mike Dahlin (University of Texas, Austin) asked about the information reported by the nodes. Rodrigo said that recording the edges is fundamental to X-Trace and other information would be application specific. Pankaj Thakkar (VMware) asked about the benefits of capturing traces externally in a nonintrusive manner. Rodrigo pointed out that there is a continuum on the amount of intrusiveness with varying tradeoffs and that they chose to explore this design point.

■ ***Friday: Global Comprehension for Distributed Replay***

Dennis Geels, Google, Inc.; Gautam Altekar, University of California at Berkeley; Petros Maniatis, Intel Research Berkeley; Timothy Roscoe, ETH Zürich; Ion Stoica, University of California at Berkeley

Gautam Altekar presented the talk on Friday, a system for debugging distributed applications through a combination of deterministic replay, symbolic debugging, and a language for expressing distributed conditions and actions. Gautam laid out the important features of Friday, namely, global comprehension, cyclic debugging, a familiar programming and debugging environment, usability in PlanetLab, and its support for legacy C/C++ applications. He compared Friday to other related tools (including WiDS, presented earlier in the same session) and noted that only Friday supported all these features.

Their general approach is similar to that of WiDS: continuous logging, consistent distributed snapshots, deterministic replay, and cyclic debugging. Unlike WiDS, logging is done in an application-transparent manner through libcall interpositioning. Friday provides users with the ability to specify global predicates in extended Python. The global predicates use the distributed breakpoint and watchpoint primitives that are implemented locally by leveraging GDB. Gautam also presented the predicate checking code for a couple of case studies, including a bug in a secure routing protocol.

Gautam noted that their approach is limited by the ability to specify relevant predicates and is susceptible to the quirks of the systems they leverage, namely, GDB and Python. As future work, they planned to improve the language support for debugging and also to explore ways to make root-cause isolation easier.

During the Q&A session, Mike Dahlin (University of Texas, Austin) enquired about the learning curve involved in using Friday's predicates. Gautam said that they have found specifying C predicates in Python to be complicated and that they were thinking about using domain-specific languages for improving the ease of predicate specification. James Anderson (UCSD) asked how they log discrete

events; the answer was that each system call event is annotated with a Lamport clock. Then James brought up the case where multiple writes are merged into a single read with TCP. Gautam acknowledged that the scenario is not currently handled.

NETWORK LOCALIZATION

Summarized by Yun Mao (maoy@cis.upenn.edu)

■ ***Network Coordinates in the Wild***

Jonathan Ledlie, Harvard University; Paul Gardner, Aelitis; Margo Seltzer, Harvard University

Locality is important in P2P file-sharing systems such as Azureus, or maybe almost all distributed systems. A typical way of exploiting locality is to use network coordinates (NCs) to predict the network distances and to choose peers based on the prediction. However, in the "wild world," developers from Azureus found that the coordinates are inaccurate and unstable to use, which motivates this research work. Jonathan gave a little background on the BitTorrent design, in particular, the peer discovery process after a new node joins the network. He argued that locality-biased swarms have improved bandwidth among peers and reduced inter-ISP bandwidth.

Jonathan next gave a short tutorial on how Vivaldi, the state-of-the-art NC system, works and their methodology to study and refine such a system. Their goal was to observe and understand the causes of inaccurate prediction of Vivaldi, test new techniques in simulation and real environments, and release their results to the latest Azureus version. To collect data, they had instrumented Azureus clients run on PlanetLab, logging every update to collect a detailed picture with a PL-to-non-PL latency matrix. They also summarize the coordinates' behavior in the software, such as instant errors and stability. One caveat of the measurement is that it is impossible to force all clients to run the latest version of Azureus, so different versions of the algorithms or parameters might be mixed. Although the NC maintenance messages are all piggybacked on existing control packets so that no additional messages are required, the authors discovered that the view of the network to the routing tables is limited. That is, there is a local bias in the communication pattern that leads to damage of the NC accuracy. Moreover, when a node receives an update from a node far away, the coordinates tend to be very unstable. Their insight is that the NC optimization should be against the whole network, with recent updates being more decisive. They proposed the idea called neighbor decay by maintaining a recent neighbor set, and they scale the force of each neighbor by its age, which limits the impact of high-frequency (near) neighbors and extends that of low-frequency (far) ones.

In the evaluation, Jonathan demonstrated that the coordinates are more stable than Vivaldi, and they improved the performance of the actual application to speed up DHT lookups. Detailed information is available at http://pyxida.sourceforge.net/.

Eugene Ng from Rice University was wondering how much benefit the underlying system can get from the gain of the new NC system on top of Vivaldi. Jonathan gave a positive answer, especially when the destination is only one hop away.

Petros Maniatis from Intel Research at Berkeley asked whether the actual Azureus users were happier after the coordinates system was deployed. Jonathan said yes, based on the feedback that he got indirectly from Paul Gardner. They were also building tracker optimization to see whether the ISPs are happier to see more local traffic. It is a gradual process because not all users will update to the latest version at once.

Eugene Ng said that based on the figures, there was still some degree of coordinates drifting. He was concerned about how much staleness could affect the performance. The answer was that all stale information was cut off after 30 minutes. It was true that if some node A was experiencing huge network latency change, and another node B only talked to it briefly, then B might have a problem. But this would be a rare event.

■ ***Octant: A Comprehensive Framework for the Geolocalization of Internet Hosts***

Bernard Wong, Ivan Stoyanov, and Emin Gün Sirer, Cornell University

Bernard discussed why geographic information about Internet hosts is useful. Some typical applications include location-aware content distribution, network monitoring and attack localization, and geography-based service discovery. However, commercial IP to ZIP code databases provide quite rough granularity and are prone to provide stale data. He presented the Octant system, which copes with the problem from a perspective of system constraints: The constraints are set from network latency measurement and other sources. He argued that Octant differs from other systems in three ways: first, it uses both positive and negative constraints; second, the system can give the users a confidence-factor-like number to reason about uncertainty; finally, Octant only needs the aid of a small number of landmarks to solve the system of constraints geometrically.

The simplest format of a constraint is described by a circle. A node is known to be inside the circle if the constraint is a positive constraint, and it is outside if the constraint is a negative constraint. Typically, a positive constraint is derived from the union of all positive circles, and a negative constraint is derived from the intersection of the negative circles. How does Octant derive constraints? By measuring the latency from the node to a landmark, we can derive both a positive and a negative constraint. To represent the

complex, irregular constraint regions, Bezier curves are used because of their preciseness and conciseness.

The hard part of the work is in deciding the radius of the circles based on the latency measurement. Bernard said that as one can use the speed of light as a conservative bound, measurement results show a strong correlation between latency and distance. Furthermore, a more aggressive bound can be used when additional geographic information is used. However, sometimes those geographically nearby nodes are separated by large latencies as a result of either long, indirect routes or high inelastic delays. Octant iteratively localizes intermediate nodes as additional landmarks to deal with indirect routes, and it models the high inelastic delays as height to cope with the latter problem. In the evaluation section, Octant was compared against GeoLim, GeoPing, and GeoTrack in terms of accuracy when using PlanetLab nodes as landmarks. The median error was 22 miles, and Octant outperformed other systems. He encouraged the audience to give a try at http://www.cs.cornell.edu/~bwong/octant/query.html

Someone from Harvard University asked whether there are some commonalities in the cases where errors of Octant are high, and how the system would perform outside the United States. The answer was that usually in those cases the hosts have very high latencies to all landmarks. Since PlanetLab doesn't have many nodes outside the United States, Octant doesn't work very well outside the United States. But as the node number increases, Octant is expected to do well too.

Tom Anderson from the University of Washington asked why in the evaluation Octant is not compared to the latest system, Topology-based Geolocation (TBG), published in IMC '06. Bernard responded that at the time of paper submission TBG's code was not available.

INTERNET INFRASTRUCTURE

Summarized by Anupama Biswas (anupamabiswas@gmail.com)

- ***dFence: Transparent Network-based Denial of Service Mitigation***

  Ajay Mahimkar, Jasraj Dange, Vitaly Shmatikov, Harrick Vin, and Yin Zhang, The University of Texas at Austin

Denial of Service (DoS) attacks are a common problem affecting the availability of Internet services. Ajay presented a novel approach in mitigating DoS attacks. It is a network-based defense system called dFence. DoS has received a lot of attention but no apt solution has been provided to date. One of the reasons is that most of the defense mechanisms require software modification on either the routers or the customer ends or both, which reduces the transparency of the networks to the ISPs and customers. Another reason is that the solutions are not compatible with the existing TCP/IP implementations. Hence there might be deployment issues. Yet another reason relates to SYN cookies, which are backward-compatible but are not used by many users, because they are set off by default. They are provided with standard Linux and FreeBSD distributions. Finally, users cannot wait until the Internet is reengineered so that DoS attacks can be prevented in a better way. An immediate, easily deployable solution is required, and if there are no DoS attacks the solution should not affect network performance.

Ajay explained that dFence in a way provides a transparent solution to the existing Internet infrastructure as it does not require any software modifications at either routers or the end hosts. It dynamically introduces special-purpose middleboxes on the path of the hosts under attack. It intercepts the IP traffic in both the forward and backward directions and applies stateful defense policies that mitigate a broad range of spoofed or unspoofed attacks.

One of the questions asked was how this system differs from the Cisco CAR. Ajay answered that Cisco CAR only does inbound traffic interception and stateful mitigation whereas their approach does outbound traffic interception, which helps to enable more policies.

- ***RBGP: Staying Connected in a Connected World***

  Nate Kushman, Srikanth Kandula, and Dina Katabi, Massachusetts Institute of Technology; Bruce M. Maggs, Carnegie Mellon University

It has been observed that BGP dynamics does not take care of packet loss when network links go down. This problem mainly occurs if there are multiple paths from the source domain to the destination domain. Through the paper Nate presented the idea that if the underlying network is still connected then the Internet domain remains connected. R-BGP is the solution that guarantees that a domain will remain connected to a destination as long as it has a policy-compliant path to that destination after convergence. The solutions provided to reduce data loss involve the complexities of the Internet and the BGP protocol. The solution provided through Resilient BGP (R-BGP) is much simpler. The data plane is isolated from any harmful effects that might occur while waiting for BGP to converge to the preferred route: The data plane is set to forward packets on precomputed failover paths. Hence packet forwarding continues unaffected throughout convergence and the routing table is not flooded with entries of all possible failover paths. This solution allows two challenges to be met: low overhead and continuous connectivity. To ensure low overhead, there is a single entry of the failover path for each neighbor. Continuous connectivity is guaranteed by providing a small amount of update information with each BGP update. Thus R-BGP works similarly to BGP except that it ensures connectivity in Internet domains as long as the underlying network is connected.

One question involved what happens if the link comes back. How does the router behave? Nate answered that in such a situation a nonfailover path will be chosen over the failover path.

■ ***Mutually Controlled Routing with Independent ISPs***

Ratul Mahajan, Microsoft Research; David Wetherall, University of Washington and Intel Research; Thomas Anderson, University of Washington

The Internet is made of ISPs that cooperate among themselves to carry traffic as well as compete with each other as business entities. No individual ISP can tune the traffic to flow through a particular route based on its self-interest; all must agree while each keeps its self-interests in mind. BGP, the most used routing protocol, provides some control, allowing ISPs to configure their outgoing traffic but giving no control over incoming traffic. This presents another problem. Suppose the incoming route fails; then the ISP cannot shift the traffic to another route as it is beyond its control. This problem is not new but can be mitigated through network engineering or by using newer routing protocols such as RCP.

The solution suggested by Ratul is the development of an interdomain routing protocol called Wiser. Wiser has the same overhead as BGP and it is complete and practical in all senses to run across multiple ISPs. There is no need for the ISPs to disclose any kind of sensitive information. Also, it allows ISPs to exert full control over the paths and make decisions based on their own interests and optimization criteria. He explained how Wiser builds the coordination mechanism on existent bilateral contracts that are already in place and is incrementally deployable across pairs of ISPs. Each of the downstream tags advertises routing with costs that are similar to BGP Multi-exit Discriminators (MEDs). Each of the upstream ISPs then selects the path with an amended process. This process considers the sum of its own costs and those reported by the downstream ISPs. Hence both the upstream and the downstream ISPs exert control on their route choices. This protocol has in-built mechanisms to discourage potential abuse.

■ ***Tesseract: A 4D Network Control Plane***

Hong Yan, Carnegie Mellon University; David A. Maltz, Microsoft Research; T.S. Eugene Ng, Rice University; Hemant Gogineni and Hui Zhang, Carnegie Mellon University; Zheng Cai, Rice University

Hong presented an experimental network, Tesseract, that provides direct control of a computer network. This computer network can be under a single administrative domain. In a typical IP network today, the desired control policy of an administrative domain is implemented via the synthesis of several indirect control mechanisms. The design that evolved from 4D architecture tries to overcome the problem. It promotes the idea of decomposing the network control plane in four different planes: decision, decomposition, discovery, and data. The network consists of something known as network decision elements. There are two abstract services to enable direct control: the dissemination service, which carries opaque control information from the network decision elements to the other nodes in the network, and the node configuration service, which provides an interface through which the decision elements command the nodes to carry out the desired control policies. The dissemination service enables plug-and-play bootstrapping in this network. The various distributed functions implemented on the switch nodes are neighbor discovery, dissemination, and node configuration services.

Tesseract reduces the need for manual code and enables a variety of different network policies to be implemented without making changes to the actual network. Hong's paper demonstrates the successful working of Tesseract with normal IP forwarding in an Ethernet network. Also, the paper evaluates its responsiveness and robustness when applied to different backbone and network technologies. It is seen that Tesseract is resilient to failures. Some questions about the scalability of the 4D network and the aftermath of a network failure were left unanswered.

HotBots '07: First Workshop on Hot Topics in Understanding Botnets

Cambridge, MA
April 10, 2007
Summarized by Rik Farrow, with help from Dan Geer

HotBots 2007 focused on understanding the current state of Botnets, and this workshop delivered what it promised. The workshop PC read 32 papers and accepted 11 for the workshop. The papers varied from detailed analysis of dissected bots to theoretical notions of potential bots. I particularly liked the talks that focused on reality, for example, on the trend toward peer-to-peer (P2P) botnets and on the research into the sizes and frequency of botnets in the wild.

You can read the papers included in this workshop at http://www.usenix.org/events/hotbots07/.

PEER-TO-PEER

■ ***Peer-to-Peer Botnets: Overview and Case Study***

Julian B. Grizzard, The Johns Hopkins University; Vikram Sharma, Chris Nunnery, and Brent ByungHoon Kang, University of North Carolina at Charlotte; David Dagon, Georgia Institute of Technology

Julian Grizzard began the day with this paper about P2P botnets. Earlier botnets relied on IRC for Command and Control (C&C), a feature that made bots easier to detect. An ngrep of network traffic can turn up IRC commands,

and as IRC has declined in popularity relative to IM, this can be a dead giveaway. Moving to P2P does make bots harder to detect, but it also means that the bot-herder loses fine-grained control over his or her botnet. With IRC C&C, the bot-herder can issue commands to all the bots currently connected to an IRC channel (or some subset of those connected) and have them carried out immediately. With P2P used as C&C, issuing commands becomes a process of sharing files, which may contain commands or an entirely new executable, a process that takes time. The bot-herder no longer can launch a devastating DDoS on a moment's notice, or fire orchestrated salvos of packets from different botnets to make detection of the DDoS sources more difficult. As it turns out, these uses of botnets have declined in importance anyway, a point made several times throughout the workshop's presentations.

P2P botnets do gain something else by giving up IRC for C&C, and that is robustness. Losing control of the C&C IRC server once meant losing potentially thousands of bots. With P2P botnets, there is no centralized point of control. Not only does this protect the botnet owner's investment but it makes bots more difficult to detect and the entire botnet more resilient to countermeasures.

Grizzard and his fellow researchers analyzed captured software, Trojan.Peacomm, a P2P bot. Peacomm uses the Overnet protocol (once used for P2P filesharing), a distributed hash table based on the Kademlia algorithm. The initial trojan infection, via spam, uses a promise of displaying a movie as an incentive and results in the bot joining a P2P network and downloading subsequent versions (secondary downloads) from other members of the botnet. Encrypted URLs are distributed using Overnet, decrypted using a static key, and new executables are downloaded using HTTP. This general technique, multiple staged infection with downloads over HTTP, seems common, based on later presentations. Network traces of the captured bot revealed a list of 10,105 unique IP addresses in packets received from Overnet hash table lookups, but the bot only contacted 4,200 during the time monitored.

During the Q&A session, Dan Geer asked, "Why not poison the distribution network if the encryption key is known?" Grizzard answered that the legal panel would have an answer to that in the afternoon. Nicholas Ianelli, a member of the CERT technical staff, mentioned that the key is known, and the Overnet packet used includes a unique meta-ID field that makes these packets easier to recognize. Someone else mentioned that more sophisticated packers are being used to make reverse engineering of malware more difficult.

Grizzard ended by pointing out that the secondary injection downloaded other specific malware, including rootkits, spam relays, email address harvesters, the trojan propagator, and a DDoS agent.

An Advanced Hybrid Peer-to-Peer Botnet

Ping Wang, Sherri Sparks, and Cliff C. Zou, University of Central Florida

Cliff Zou described hypothetical hybrid botnets. In this research, his group designed an advanced P2P botnet that would be harder to shut down, monitor, or hijack. Their design has two classes of bots, servants and clients. Servants have fixed IP addresses and act as the C&C network for the botnet. Clients get their commands and new versions of malware from the servants. Any bot can act as a sensor host and have performance information sent to the sensor host from clients. The sensor host then reports to a servant. Clients contain a fixed list of servant addresses, but no client has all servant addresses.

In analysis, removing 80% of servants bots leaves 95% of client bots connected, so this type of botnet would be very resilient. Possible defenses include poisoning servants using honeypots or capturing servants early in the infection process. Someone asked whether it was wise to publish research like this, and Zou responded that they felt it was appropriate to be forward-looking about new styles of attacks.

■ ***A Distributed Content Independent Method for Spam Detection***

Alex Brodsky, University of Winnipeg; Dmitry Brodsky, Microsoft Corporation

Alex Brodsky presented Trinity, a distributed database designed to collect source addresses of spam relays while remaining resilient to attacks. Brodsky pointed out that using blacklists has become less effective over time, as bots provide spam relays that often route spam via an ISP's mail gateway, as well as sending a relatively small number of emails. Trinity uses a SpamAssassin plug-in to capture a spam sender's email address by parsing Received headers email envelopes that have been classified as spam and sent to Trinity servers. The servers are selected to store spam relay IP addresses using the DHT hash, so no single server holds all the addresses. Servers also replicate the addresses received, so one or more servers can be lost without the service being disabled. Trinity also uses reputation scoring by collecting answers from different peers, to guard against spammers running a server and poisoning the database.

Trinity is under development.

MEASUREMENT

■ ***The Ghost in the Browser: Analysis of Web-based Malware***

Niels Provos, Dean McNamee, Panayiotis Mavrommatis, Ke Wang, and Nagendra Modadugu, Google, Inc.

Niels Provos described the results of a project he started at Google in 2006. Niels had become increasingly aware that browser vulnerabilities provide fertile ground for the in-

stallation of malware. In what he terms "drive-by downloads," Niels tells how Web sites with imperfect security wind up with modified pages. The modifications include IFRAMES or JavaScript that causes the browser to download a first-stage exploit. The first stage then downloads the real malware, which will be spyware for financial information, spam relays, bots, and tools for actively attacking systems.

Google already trawls the Web and caches pages. What the Google team has done is build a system that uses heuristics to decide whether a page may be malicious. The suspect pages get loaded into an instrumented version of Internet Explorer (IE) running within a VM-encapsulated copy of Windows. Network activity (downloading malware), changes to the browser state or registry keys, and software installs that occur after loading the suspected page in IE determine whether the page really does result in a drive-by download. Google then marks pages determined to result in the downloading of malware by returning a warning page as a search result. The current setup scans up to 500,000 pages a day, and under 0.5% of all pages scanned have been infected.

Besides compromising Web servers to install the download code, Niels said that other methods of serving up code included delegated (subsyndicated) sponsored ads and third-party widgets, such as the Preying Mantis counter that started delivering malware to visitors instead of being "just a counter." Popular sites were less likely to be vandalized as these sites are better maintained, and large sites were much faster at responding to reports of infected Web pages. Niels also mentioned that sometimes a server hosting many Web sites will be compromised, and pages for all the hosted sites will then be infected.

Fabian Monrose (Johns Hopkins) asked how Google handles infected pages. Niels explained that Google serves up a warning page. Earlier versions of this page allowed users to click through to the infected site, and 30–40% of users did so. Someone else asked whether exploiters are using Google Analytics, and Niels said he wasn't sure, but he expected that some exploiters were using it to keep track of the number of sites they had exploited. In response to Dan Geer's question whether the robots.txt file is honored during this process, Niels answered that Google follows the standard for crawling Web pages and ignores those. Fabian asked if Google is censoring the Web, and Niels replied that they are observing that some Web pages exploit your browser. You can still cut-and-paste your way past the warning.

■ ***My Botnet Is Bigger Than Yours (Maybe, Better Than Yours): Why Size Estimates Remain Challenging***

Moheeb Abu Rajab, Jay Zarfoss, Fabian Monrose, and Andreas Terzis, Johns Hopkins University

Andreas Terzis started by saying that botnet sizes vary according to how and when they are measured. Some researchers have found botnets with 350,000 members, whereas others report that botnets rarely exceed a few thousand bots. Andreas suggested several metrics for measuring, including descriptions of how the botnets were measured. For example, the infection footprint will be the largest size, as it represents all systems detected as infected, but effective size represents the number of bots available at one time. The difference between these two measurements can easily be one or more orders of magnitude, for example, 45k infected but only 3k active.

There are certainly difficulties in measuring botnets. A total of 48% of IRC servers used for bot-herding block join messages. Another counting technique involves querying DNS servers for cached copies of bot server addresses, which provides a list of domains that have at least one infection. Another issue has to do with overlap between botnets and their owners. The authors determined that 25% of 472 botnets that they tracked were associated into only 90 groups. Some of the overlap occurs because bots can be commanded to clone themselves, instantly creating a related botnet.

■ ***Toward Botnet Mesocosms***

Paul Barford and Mike Blodgett, University of Wisconsin—Madison

Mike Blodget described the Botnet Evaluation Environment (BEE), a testbed for experimenting in a secure and flexible fashion with botnets that is Emulab-enabled. Mike explained that the attraction of botnets to organized crime makes study of this phenomena an important area of research. Bots and botnets are also growing in complexity as well as in their resistance to dissection. It is common to find malware that is packed (obfuscated), can detect whether it is being run in debugger mode, or is run within a VM. Techniques used for determining whether a VM is present include checking the interrupt vector and looking for VMWare tools.

BEE provides support for bots and the services they require, such as DHCP, DynDNS, and IRC. The authors are building a library of OS/bot images using both bots built from source and bot binaries. For security, they block all UDP traffic from BEE, use an unroutable ten-net within BEE, and firewall all traffic between the test network and the experimenters' network.

DETECTION, RESPONSE, AND ANALYSIS

■ ***Wide-Scale Botnet Detection and Characterization***

Anestis Karasaridis, Brian Rexroad, and David Hoeflin, AT&T Labs

Anestis Karasandridis described how his group detects bots by analyzing flow logs. As part of AT&T security efforts, they have been analyzing flow logs for evidence of bots and C&C servers. AT&T runs a Tier 1 network, what

I once would have called an Internet backbone, and collects *8 to 10 billion flow logs per hour*. They use triggers, such as hosts that scan, relay spam, or attack other systems, then apply heuristics, described in their paper, to whittle down the number of flows. Some heuristics are obvious, such as selecting flows to the common IRC ports. Others are much more sophisticated, for example, looking for a pattern of flows that would match the PONG response from bots to the C&C server within a particular time window.

Using this and other techniques, the group detected 376 unique C&C server IP addresses between August 2006 and February 2007. During the same period, they discovered 6 million bots, and they continue to find about 1 million bots per month. They tested the accuracy of their data by contacting other ISPs or looking within the data, measuring a false positive level of less than 2%.

Nicholas Ianelli (CERT) asked about the percentage of commands that were encrypted. Brian Rexroad answered that perhaps 5% use some channel encryption or obfuscation. Someone else asked what other means were used to verify correctness, and Brian answered that they do perform some packet capture of suspected bots. Another question concerned the high number of bots seen, and the answer was that because unique IP addresses are counted, dynamic addresses can affect this. In the paper, they mentioned that there is considerable churn, with bots changing channels or servers every 3-4 days. Another person asked about using their algorithms on Arbor Network boxes, but the authors didn't know whether this would work. Brian did make a comment that games can attempt many connections to servers and fail, making them appear like scanners, and thus bot clients. Finally, Niels Provos asked if they had looked at anyone else's Netflow data. Andreas said they are using such data to protect their customers (which are other ISPs). Niels then inquired, "If we asked politely, might we exchange info?" Brian answered that it might be possible.

■ ***Rishi: Identify Bot Contaminated Hosts by IRC Nickname Evaluation***

Jan Goebel, RWTH Aachen University, Germany; Thorsten Holz, University of Mannheim, Germany

Thorsten explained how his group had created a simple script that looks at communication channels for common IRC commands and watches for patterns in bot nicks. Nicks are used in the IRC protocol to identify a client connection to an IRC channel, and each must be unique per channel. Rishi uses ngrep to collect lines and a 1700-line Python script for analysis. Vern Paxton asked whether you could use Bro, and Thorsten answered that this is a prototype, but perhaps.

Liss asked whether botnet owners actually use nicks to partition their networks. Thorsten did notice checks for .edu hosts and for country domain, so they can group commands (.edu hosts will have more bandwidth). Nicholas Ianelli says he has seen partitioning in nicks, such as *p* for private network. Someone asked about using machine learning techniques, which had been shown in one of the slides, and Thorsten confirmed that they would work. Another person asked how the method compared to other methods. The answer was that it did better than Nepenthes (see "Advanced Honeypot-Based Intrusion Detection" in the December 2006 *;login:*). Niels Provos asked whether this assumes that IRC is still being used. Thorsten said that gamers use IRC as well as members of the whole underground economy. In response to Niels's question "If people start using dictionaries, can you use other events?" the answer was that indeed heuristics could help here.

■ ***AS-Based Accountability as a Cost-Effective DDoS Defense***

Daniel R. Simon, Sharad Agarwal, and David A. Maltz, Microsoft Research

Fabian Monrose, the session chair, introduced this as "the evil bit talk" and we soon discovered why. Dave Maltz provided motivation for a DDoS defense by pointing out that 4000 bots can overload a 4-Gbps link ($25,000 to $50,000 per month) for about $1,600 a month cost for the attacker. To protect against a 50,000-bot network costs $10 million over three years, to pay for 3 OC48s from a provider for $210,000/month. Other solutions include Content Distribution Networks (CDNs) such as Akamai and are very expensive.

The proposed solution is an architectural change, involving just software and not hardware, to identify trusted sources with a persistent attribute, and then receivers can blackhole sources using whatever method they want. The solution assumes pairwise trust among Autonomous Systems (AS). Each ISP modifies customer relationship software (used for billing customers), keeps track of their IP addresses, and must be willing to install filters on particular source-address pairs (you can only filter sources that target your destination). The Filter Request Server, FRS, forwards data to your router and is used to install this filter.

Border routers at the edges of accountable networks add the "evil bit" to packets (Bellovin's evil bit, or Crocker's well-maintained bit, a.k.a. the anti-evil bit). This bit indicates that marked packets came from a trusted source.

Rik Farrow leaped to his feet and provided several comments: You can set the evil bit on packets coming from any AS you don't like; you can clear all evil bits because you are opposed to this solution; trust between ASes may not (probably does not) exist (since these are often competitors); and ISPs can slap the evil bit on repeat offenders instead of installing filters on their own routers. Someone else mentioned that FRS could be used to overwhelm the ability of routers (a DoS by overusing the filtering function of routers). Dave still maintained that this is a cheap, on-the-fly reputation system. Someone else suggested that it

would be easy to frame opponents, while a final suggestion was to check Netflow logs to determine guilt at the user's ingress point.

CASE STUDIES

■ ***Panel: Legal Issues About Botnet Tracking and Response***

Panelists: Jon L. Praed, Internet Law Group; Jody R. Westby, Global Cyber Risk, Adjunct Distinguished Fellow to Carnegie Mellon CyLab; David Dagon, Georgia Institute of Technology; Alexander Muentz, OnSite E-Discovery

Dagon started with a scenario where a student accidentally starts to proxy all his IP traffic from his dorm room via your honeynet, and you capture data such as his social security number and email to his doctor, and to his lawyer, who happens to be in Europe where privacy laws are stronger. Dagon pointed out that there are many laws that could be applied here that offer protection of the student's privacy: FERPA, HIPPA, and GLBA. Also, nearly every university has a policy pertaining to Human Subjects that can also affect privacy. The best approach is to have a clear policy that addresses privacy issues.

Dagon also suggested that you operationalize your research, that is, make it useful to the organization, to gain allies in operations. You should also sandbox your investigations, using a Truman network.

Alex Muentz described the applicable U.S. federal laws. The Computer Fraud and Abuse Act, 18 USC 1030, provides you with a great deal of protection when you act as a provider. Working within a sandbox does not give you that protection. The Stored Communications Act, 18 USC 2701, and Electronic Communications Privacy Act/Wiretap Act, 18 USC 2511, are similar to 18 USC 1030, so don't sandbox for maximum protection, but act as provider working to maintain the network or system.

John Praed's group actually works as private investigators, trying to track down wrong-doers using log and other data to do so. He suggested that even working as a "white hat" can get you in trouble via overly aggressive conduct. He mentioned winding up in trouble because of an ambitious local DA who wants to be governor. He also pointed out that Putin in Russia could well act aggressively to protect "businessmen" in Russia who look like "black hats" to us. Praed also suggested moving cautiously in government investigations, as illegal search can cause evidence to be tossed out. There are also cases of "black hats" suing the "good guys," as in spammers suing SpamHaus to be removed from the blackhole list.

Praed said that in the future, the primary economic driver for criminals will be cyberextortion over hard drive contents and captured keystrokes. Already, he has seen examples of blackmail backed up with physical threats.

Jody Westby spoke quickly as time was running out for this session. She discussed procedural and practical methods for combating bots. The problem exists in 243 countries, with 1.2 billion users, and lots of juridicial issues. A recent Council of Europe Cybercrime Convention agreed to by the United States will be changing the U.S. legal scene soon.

Someone asked about attacking evil servers, and the unanimous response was "Don't do it!" Don't even log in unless you have authorization.

■ ***A Case Study of the Rustock Rootkit and Spam Bot***

Ken Chiang and Levi Lloyd, Sandia National Laboratories

Ken Chiang told us about reverse engineering Backdoor .Rustock.B or Spam-Mailbot.c. Chiang and Levi Lloyd started by packet capture and could see that the bot uses HTTP POSTs, with encrypted payloads, for C&C. That made them interested in recovering the key used in order to learn more about the C&C channel.

Chiang told us that Rustock has three different phases of deobfuscation. The first phase deobfuscates the rootkit loader, which contains the second deobfuscation routine. The rootkit includes the third deobfuscation routine and unpacks the spam module. The rootkit then destroys the magic numbers in the PE and MZ headers to help defeat RAM forensics. The rootkit adds a value to the services registry key to restart itself upon boot, and it hides this key once the rootkit is running. The rootkit hooks several system calls to hide itself and injects the spam module threads into services.exe.

Levi took over and talked about the spam module. The C&C channel uses HTTP POST and performs a key exchange with a login.php script. The RC4 key is stored in a global sruct in memory, created by the client, encrypted using the server's public key and sent to the server. The server can decrypt the key with its private key, and then the server responds with a check. Once the client responds, the client is ready to accept commands. Rekeying and a new login occur every few minutes. The real reason for the exchange is to collect a list of mail servers, some spam content, and a list of targets to which to send the spam.

Dan Geer asked, "Is a good botnet better run than the average home system?" Levi answered that he could agree with that, as the client gets a list of processes to kill, which includes other bot software.

■ ***The Anatomy of Clickbot.A***

Neil Daswani, Michael Stoppelman, and the Google Click Quality and Security Teams, Google, Inc.

Neil Daswani began by explaining the business model exploited by Clickbot.A. What most of us know is that advertisers pay for click-throughs, and what has occurred to many of us is that this mechanism appears to be ripe for fraud. It turns out that this is indeed an issue for Google,

and Clickbot.A provides a stunning example of just how complex exploits can be.

To begin with, the attacker set up Web servers acting as doorway sites, using doorway.php, signed up to be subsyndicators, then signed up referral accounts to get paid for page views. Then Clickbot.A went after syndicated search engines, worked to be under the radar, sending very few clicks from each bot. Google did notice a pattern in the click-through traffic, and it marked Clickbot.A clicks invalid based on a recognizable pattern. This scheme also used redirectors, a form of proxy that strips off identifying information, such as the Referer: header line in the client's request.

Clickbot.A is an IE Browser Helper Object (BHO), likely because BHOs run within the process space of the browser and have access to the entire DOM library. In what was described as a "slow infection," desktops infected rose from 100 in May 2006 to 100,000 in mid-June 2006. Clickbot.A was distributed as a trojanned game download. The botnet server was written using PHP, and it did not initially have any form of access control, making it impossible to know how many desktops were infected. You can see screen output from the script in the paper. Neil also mentioned that only 7 out of 24 AV packages even recognized Clickbot.A as malware in June 2006.

Clickbot.A generates traffic by contacting the botmaster site, requesting a keyword and doorway site URL, then using this information to choose an ad to click on, but it does this only once every 15 minutes. Neil presented some back-of-the-envelope calculations about the amount of money a successful click-fraud campaign like this might make, obtaining a value of about $50,000. Google claims that there is less than a 10% rate of click fraud.

WORK-IN-PROGRESS REPORTS

Steve Santorellis, Microsoft, announced conferences coming up in Sydney in July and in France in November for Law Enforcement (LE) and training of LE; some academia will be coming to these events. Contact steves@microsoft.com.

Michael Collins (CERT/NetSA), SEI, described measurements of machines that have likely been compromised (unclean machines). He expected that IP addresses of compromised systems would be randomly distributed, but he instead found that IP addresses cluster on certain netblocks rather than in any random block on the network; CERT has been monitoring a large /6 or /8 network since 2002. Looking for tightly packed addresses, Collins compared 600,000 bot addresses and found that unclean addresses tend to cluster. His data shows that uncleanliness is persistent over six months and that spamming but not phishing is closely related to unclean machines.

Dan Geer spoke about security metrics. He reminded attendees that the Metricon workshop will be at USENIX Security in Boston this summer and that CMU has an economics workshop this summer. Dan said that little data sharing is going on in financial and energy sectors about Internet-related fraud, but that insurance companies are really interested in this. In terms of presenting statistics, the best you can do now is trend analysis, and used the CSI/FBI surveys as an example. He advised that if you present statistics, be consistent about how you collect your data and the terms used to present them. Dan did point out that eBay takes down 1000 fraudulent sites per day, and eTrade reported a loss of $0.12 per share on $18 million in profits—losses based on stolen identity via keystroke logging.

Thorstein Holz of the University of Mannheim, Germany (and a frequent *;login:* contributor), described examples of HTTP-based bots. He did malware analysis using Nepenthes to collect examples. His group found that many new bots use HTTP in their analysis, with many having second- or even third-stage downloads. Recently he has seen bots using HTTP control channels and encrypted commands; he showed examples of doing ping, creating UID, getting second- or third-stage code, getting new code, and periodically querying the same HTTP server. Also, he has seen bots actually sending email for communicating, like slow-motion IRC.

D. Dagon of Georgia Tech pointed out that there are tens of thousands of new versions of malware appearing and that these are not being created by industrious individuals. Instead, queen software and code generators morph existing malware into new versions that won't be identified by AV. He suspects that there is just a small group of people doing this. His agenda is to find the people doing this. He also wants to collect more examples, so we can do analysis and learn more about obfuscation techniques.

William Zalewski of AOL claimed that bots providing SOCKS proxies are a silent but growing threat to the Internet. Based on his own analysis of traffic, he believes that there are many more relays than are active and that some provide reverse-connect proxies (where the bot goes out through a firewall or NAT, then offers to relay external connections internally). He has seen reverse-connect proxies that are totally nonrandom in behavior, connecting every 20 minutes. He has seen both v4 (a simple nine-byte setup) and v5 SOCKS proxies (a more complex four-packet exchange setup) being used.

M. AbuRajab of Johns Hopkins suggests clustering malware by activity, not by the features of the binary. Basically, he wants a feature vector for bots. Botnets apparently are purpose-built, not off-the-rack, a reflection of a clustering in the underground economy.

Masashi Eto, NICT, presented an integrated analysis of threats in large networks. His group, NICTER, monitors a

darknet of 100,000 IP addresses for real-time detection of incident candidates. They use automatic capture of malware, Nepenthes, and code analysis. Eto demonstrated some very cool 3D animated visualizations, at the very end of the WiPs and the workshop.

HotOS XI: 11th Workshop on Hot Topics in Operating Systems

San Diego, CA
May 7–9, 2007

KEYNOTE ADDRESS

■ ***Transactional Memory: What's the OS Got to Do with It?***

David A. Wood, University of Wisconsin—Madison

Summarized by Ian Sin Kwok Wong (iansin@eecg.toronto.edu)

Multicore processors are here but we do not have the parallelism we need in applications to take advantage of this new architecture. Like Dave, most of us agree that parallel programming is hard because people think sequentially and as a community we have been using the same programming models for the past 30 years and thus have not acquired enough experience with parallel programs. An application can be parallelized through the use of threads. However, accesses to shared data must be carefully synchronized to maintain application correctness. Otherwise, deadlocks, live locks, or data races might result.

Transactional memory (TM) is one way to ease this burden while allowing concurrent execution of a program. TM originates from database systems and its declarative model makes it an attractive proposition. The programmer says what he or she wants and the system deals with the "how," while maintaining ACID properties. Software TM is slow and Dave believes that the next logical step is a hybrid implementation, which is basically a best-effort software TM with hardware acceleration for the common case. However, besides performance, the goals that TM systems are trying to achieve include unlimited transactions, long-running transactions, and unlimited closed nesting. In order to be successful, these facilities should be provided with modest hardware support.

Dave then gave the audience an overview of TM terminology and introduced LogTM-SE, which can be dubbed an "almost" virtualizable TM system. LogTM-SE is a hybrid TM that explores eager version management and eager conflict detection in the design space. LogTM-SE uses simple hardware support and exposes the interface to the software, which in turn implements the required policies.

In the Q&A period, Emin Gün Sirer (Cornell) argued that he did not believe TM to be the solution. In his opinion, locking instructions are a simple sequence of 12 instructions but the main problems are what students are being taught and what and how systems are being built (in reference to Linux). His question was, "Now that we have TM, what does this do for the average programmer?" Dave's response was that programming language experts would be required and that this subject was not his domain of expertise. Another interesting question from Kai Shen (Rochester) dealt with the state of the art in "transactionalizing" large systems. Dave replied that the main problem behind such an effort is that the simple close-nested abstraction was not powerful enough for highly complex systems and dirty tricks were required because of the lack of open-nested transaction support. He continued by arguing that the open challenge in transactionalizing large complex applications is to learn when simple abstractions are not sufficient and come up with extensions that will be usable by the average programmer.

COPING WITH CONCURRENCY

Session Chair: Armando Fox, University of California, Berkeley

Summarized by Ian Sin Kwok Wong (iansin@eecg.toronto.edu)

■ ***Is the Optimism in Optimistic Concurrency Warranted?***

Donald E. Porter, Owen S. Hofmann, and Emmett Witchel, The University of Texas at Austin

Donald Porter argued that the conservative mutual exclusion provided by locks, especially when locking is coarse-grained, is detrimental to performance. However, using a fine-grained locking scheme for better concurrency is very complex. Optimistic concurrency, achievable through transactional memory, removes the serialization points that locks suffer from. Donald argued that porting systems to leverage optimistic concurrency is a lot of work for potentially marginal benefits and his talk focused on quantifying the benefits of optimistic concurrency on multicore platforms.

He explained how his tool, called Syncchar, measures data independence in applications by analyzing the conflicts in the address sets that are accessed within critical sections. The tool was then used in a case study to measure data independence in a standard Linux 2.6 kernel. Although they found that most locks used in Linux were fine-grained, they also found 95% data independence on the dcache lock—which indicates a good opportunity for improvement. The study also compared Linux against TxLinux, an implementation that converts 32% of spinlocks to use transactions. Donald outlined some limitations, such as the pathological behavior of linked lists, which caused many conflicts and was due to the way the data structure was organized. The talk concluded with the question of whether optimistic concurrency will help your average system and the answer was, "It depends."

In the Q&A session, Dave Wood (Wisconsin) asked whether the linked-list pathological case was similar to open-nested transactions and whether it would be beneficial to raise the level of abstraction throughout Linux. Donald indicated that he looked into this and found the best way was to get rid of the data structure altogether to avoid the conflict. Although this might not be the best solution, it would be a first step before building nesting and correctness conditions. Ding Yuan (Illinois) followed with a question regarding how Syncchar dealt with multiple locks being held for a critical section. The answer was that each lock is treated individually for now, but ideally Syncchar should coalesce related locks. Sandhya Dwarkadas (Rochester) then asked whether Linux was the best place to use transactional memory and whether this would make the system simpler. Donald argued that transactional memory gives more options for tuning the system for performance and that is part of the reason for using transactional memory on the Linux kernel.

■ ***Thread Scheduling for Multi-Core Platforms***

Mohan Rajagopalan, Brian T. Lewis, and Todd A. Anderson, Programming Systems Lab, Intel

With significant architectural differences in many-core processors compared to traditional SMT architectures, Mohan argued that we need to rethink the way threads are scheduled for better performance. Most important, we need to know what threads to run, and on which cores to run them such that they benefit from cache locality. The goal of this work is to develop an automatic solution that is both portable and easily programmable, although it might not perform as well as a hand-tuned application.

The contribution of this work is an automatic scheduling framework that makes it relatively easy for the average programmer to achieve good performance without an in-depth knowledge of the underlying multiprocessor architecture. This is achieved through minimal programmer annotations, whereby related threads are tagged with a Related Thread ID (RTID). The system uses the RTIDs and their attributes for a best-effort placement of threads and updates them at runtime. Thread placement can also be explicitly guided by the expert programmer.

Timothy Roscoe (ETH Zürich) asked Mohan where the operating system (OS) was in this work. To Timothy, this is a runtime system. Mohan responded by saying the design was a runtime one but the scheduling problem is the same as traditional OS scheduling. A hanging question from Michael Isard (Microsoft Research) was whether threads were the right level of abstraction.

■ ***Automatic Mutual Exclusion***

Michael Isard and Andrew Birrell, Microsoft Research, Silicon Valley

Andrew Birrell argued that in software development we want correctness, efficiency, and maintainability. Increasingly, people want to program for parallel architectures and threads, together with locks as synchronizing primitives, as a popular way to implement parallel programs. He argued that the use of locks gets harder as a project grows in size and relies on the programmer to make decisions to maintain application correctness (e.g., to preserve the acyclic locking order). Failure to program locks correctly results in deadlocks, data races, and oversynchronization, among other problems. An alternative is the transactional memory paradigm. The complexity is moved from the programmer to the experts, but the programmer still needs to reason about the concurrent parts of a program.

They propose to solve these problems by using automatic mutual exclusion (AME). AME is similar to an event-based system augmented with transactions that can run concurrently. Transactions are presented differently to the programmer. Essentially the whole program is run as a transaction and the programmer must explicitly indicate when a code fragment is expected to run outside of a transaction. AME is composed of a thread pool and asynchronous methods. Each async method runs as a transaction and transactions execute concurrently with the help of threads. What is produced is a correctly synchronized concurrent program that the runtime system needs to execute efficiently. The responsibility of making intelligent decisions is thus moved from the programmer to the intelligent runtime system. To deal with blocking IO calls, AME implements a yield system method that breaks a method into atomic fragments. Any method may call one or more yields but the caller must be aware of this, since their state becomes visible.

After the talk, Jon Howell (Microsoft Research) agreed that labeling things to be outside transactions makes sense but wondered about yield propagation. From previous experience, he argued that most functions end up having the yield annotation, making them practically useless. Andrew replied by saying that they don't know yet and are in the process of implementing the system.

MODERN ABSTRACTIONS

Session Chair: Landon Cox, Duke University

Summarized by Ramakrishna Gummadi (ramki@cata-rina.usc.edu)

■ ***Hype and Virtue***

Timothy Roscoe, ETH Zürich; Kevin Elphinstone and Gernot Heiser, National ICT Australia

Timothy Roscoe presented a call to think about virtual machines differently for research purposes than what is done today. He argued that research on virtual machines has not provided new insights into operating system abstractions or structures; instead, it has focused more on building better hypervisors or on developing new uses or applications around them. However, the work of building hypervisors

involves reimplementing many of the abstractions of traditional OSes, such as protection and sharing of hardware resources, and efficient communication. So, the VMM research in this space has not been productive from a research perspective. Moreover, VMMs suffer from problems such as implementation complexity, poor performance, and large TCBs (Trusted Computing Blocks).

However, Roscoe noted that a key area in which VMMs represent a clear advance is in providing an application-level abstraction, since one can bundle entire applications as ready-to-run packages. So, OS designers can leverage this benefit of VMMs to save themselves the burden of porting applications while building new OS abstractions and facilities. The speaker outlined some really new research directions for disruptive virtualization, such as new kernel and OS API designs based on transactional memory, concurrent hardware, and high-level languages, as well as new kernel implementation techniques based on verifiable languages and machine-checkable formal specifications. He also called for OS and VMM designers to carefully examine the performance of resulting systems using improved and more meaningful metrics for measuring VMM isolation and scalability. The ultimate outcome of such efforts would thus be new and disruptive OS research that actually has the chance to succeed in the real world because of the availability of crucial application support provided by VMMs.

Gün Sirer from Cornell asked what to do with applications where one wants to keep some of the old POSIX interface yet include a new API. Roscoe answered that we might need to decide between having a legacy or a new interface. Margo Seltzer from Harvard remarked that the Program Committees at conferences should become more open-minded about accepting OSes that didn't necessarily support the usual suite of compatible applications and/or present performance numbers for such applications. Jeff Mogul from HP asked whether there is an analogy between the narrow waist occupied by IP in the Internet stack and VMMs in OSes. Roscoe replied that we shouldn't standardize a thin waist and that not having a thin waist in OSes is not a problem.

■ ***Relaxed Determinism: Making Redundant Execution on Multiprocessors Practical***

Jesse Pool, Ian Sin Kwok Wong, and David Lie, University of Toronto

Jesse Pool suggested a system that provides relaxed determinism guarantees in order to practically allow redundant execution of threaded processes on multiprocessors. The motivation is that future multiprocessors will have enough resources to allow processes to be executed redundantly in order to provide guarantees such as reliability, as well as security through diversity. Unfortunately, today's systems don't allow practical multithreaded applications to run redundantly because of various nondeterministic execution scenarios encountered under real-world settings.

Jesse described their system called Replicant, which provides reasonable performance while tolerating nondeterminism. Their key insight is that, in many cases, the event ordering seen by applications will not result in significant differences in application behavior, so some event orderings can be profitably relaxed. This approach is similar to the relaxed memory consistency models used in modern processors to achieve respectable performance. Thus, Replicant loosely replicates the order of events among the executing replicas, and it relies on determinism hints inserted by the developer in order to enforce a precise ordering of event delivery across replicas.

Replicant incorporates several system facilities to achieve such a nondeterministic but correct execution. First, each replica is executed in an OS sandbox called a harness, which captures the process-specific OS state. Second, a matcher component in the kernel is used to fetch and replicate inputs from the external world and deliver them to the harness. It is also responsible for determining when outputs from the harness should be made externally visible. The Replicant system has been implemented for Linux, and it has successfully managed to run several applications in the SPLASH-2 benchmark suite.

Jason Flinn from Michigan asked whether one can also use annotations as performance hints. For example, the replicas could all use shared memory, and the locks in the code can be thought of as annotations that indicate this possibility. Jesse responded that this was likely to slow down performance. Diwaker Gupta from UCSD asked whether the replicas can be thought of as state machines, and the speaker said that the replicas have only to provide identical outputs, so their implementation need not conform to the state-machine execution model. Diwaker also wondered whether Replicant would scale to more than two replicas and Jesse responded that they had tried more than two replicants. Emmett Witchel from Texas asked how the matcher could accurately deliver answers to inherently nondeterministic system calls such as gettimeofday. Jesse replied that the same answers to all replicas have to be returned for such system calls only when such calls were used in sequential regions of threaded code.

■ ***Compatibility Is Not Transparency: VMM Detection Myths and Realities***

Tal Garfinkel, Stanford University; Keith Adams, VMware; Andrew Warfield, University of British Columbia/XenSource; Jason Franklin, Carnegie Mellon University

Tal Garfinkel pointed out that recent worries about being vulnerable to stealthier rootkits are unfounded, because building transparent VMMs is effectively impractical. This is because there are numerous easily detectable anomalies between real and virtual hardware, as described in the

paper, and possible countermeasures to such anomalies are demonstrated to be infeasible. The take-away conclusion of the talk and the paper was therefore that transparent VMMs are unrealizable from both a performance and an engineering standpoint.

Tal talked about various discrepancies that an application running inside a VM can detect. For example, some CPU instructions are nonvirtualizable, so applications can execute them and observe their side effects in order to detect whether they are running in a virtualized state. Second, the emulated hardware is out of date with the CPU capabilities, because a VM typically emulates old but well-understood hardware resources such as chipsets and disks from the 1990s. Also, applications can easily detect features of emulated hardware resources such as TLB sizes that behave differently on virtualized hardware than on native hardware. Finally, there are a lot of timing discrepancies exposed through both local and remote time sources that allow an application to construct covert channels for detecting that it is running on a VMM. Although it is theoretically possible for a VMM to provide perfect transparency to applications through techniques such as time dilation, the resulting emulation overhead and the overall performance impact would make implementations impractical, while potentially opening up further, more subtle sources of vulnerabilities. Tal concluded that virtual machines can never truly approximate real hardware and that there are both good and bad outcomes as a result.

Timothy Roscoe from ETH Zürich pondered whether another form of nontransparency in VMs is when nonvirtualized programs running on virtualized hardware end up automatically exploiting the nontransparent behavior of VMs to do bad things, such as polymorphic viruses using nondeterministic cycle counts. Tal said that was an interesting idea, pretending to be a VM to fool malware into believing it is being run in a sandbox, at which point it exits. Tal mentioned some work at Symantec along these lines. John Wilkes from HP wondered whether there exists hardware to detect rootkits, and whether such hardware would be practical. Tal suggested that you write something that permits only authorized VMs to run. Emin Gün Sirer declared that he didn't believe in digital signature schemes, referring to authorizing VMs.

ALGORITHMS FOR PROFIT

Session Chair: Emmett Witchel, University of Texas at Austin

Summarized by Vinod Ganapathy (vg@cs.wisc.edu)

■ *Don't Settle for Less Than the Best: Use Optimization to Make Decisions*

Kimberly Keeton, Terence Kelly, Arif Merchant, Cipriano Santos, Janet Wiener, and Xiaoyun Zhu, Hewlett-Packard Laboratories; Dirk Beyer, M-Factor

Kim Keeton said that complex systems problems often present a large search space, with complex tradeoffs, with the best and worst solutions differing by as much as an order of magnitude. Currently, ad hoc domain-specific solutions are used to arrive at solutions to these optimization problems.

Keeton then went on to argue that the approach to this problem should be to use mathematical programming to solve these optimization problems. A math program has input parameters, objective functions, and constraints. The first step should be to formally describe the problem, following which commercial solvers can be employed to arrive at a solution. An alternative is to use meta heuristics, such as genetic algorithms, to arrive at solutions to optimization problems. Keeton described the generic structure of a genetic algorithm and showed how it can be used to, for example, minimize total penalty in an optimization problem.

Keeton also exploded several myths regarding optimization. (1) Myth: "Simple heuristics are good enough." Reality: Simple heuristics may be good enough, but what does "good enough" mean? (2) Myth: "One may have to oversimplify the problem to make it amenable to a math program solver." Reality: One can use alternative optimization techniques; one does not have to shoehorn the problem into the solver available. (3) Myth: "Optimization is too slow." Reality: This claim cannot be made in general; the cost of optimization depends on the specific problem to be solved. (4) Myth: "Inaccurate data can lead to bad decisions." Reality: This cannot be helped, and so a sensitivity study is necessary.

Someone asked whether there are any guidelines on when math programming is useful, or when machine learning is useful. Kim replied that you should use statistical machine learning to uncover what variables are important. Brian Knoll of Michigan wondered whether the main challenge was in trying to express all costs in the same currency. Will this not result in bad data, which results in inaccurate results? Kim answered that we must express everything in the same currency. In their paper they did so using dollars (real currency) to quantify the costs of various options.

■ ***Hyperspaces for Object Clustering and Approximate Matching in Peer-to-Peer Overlays***

Bernard Wong, Ymir Vigfússon, and Emin Gün Sirer, Cornell University

Bernard Wong explained that the motivation for this work is that services such as Gnutella provide a search primitive that can conduct approximate search (so a search for "Bratney Spears" will still yield files related to the real goal of the search "Britney Spears"). However, Gnutella is slow. One solution is provided by systems such as Chord, Pastry, etc., all of which use distributed hash tables (DHTs). However, DHTs do not support approximate search, but need the exact key.

Wong then went on to present the hyperspace model, which achieves the best of both worlds by supporting approximate search on P2P overlays. A hyperspace is a high-dimensional space in which objects that are "close by" in the sense of having small edit distance are located close together in the hyperspace (i.e., the Euclidean distance between these objects is small). The main challenge in a hyperspace is to correctly choose the basis for the high dimensional space, and the cost of a poor selection of labels is that it can lead to poor clustering, and thus poor search results.

The questions mainly concerned how difficult it was to choose an appropriate basis. Wong mentioned that basis selection must be repeated over time, depending on the current search queries that were popular.

■ ***Optimizing Power Consumption in Large Scale Storage Systems***

Lakshmi Ganesh, Hakim Weatherspoon, Mahesh Balakrishnan, and Ken Birman, Cornell University

Lakshmi Ganesh began by saying that much money and energy get wasted at data centers (e.g., $7.2 billion in a recent year, of which $2.4 billion was spent to cool disks). Thus, a technique is needed to reduce this amount (e.g., by spinning down disks from which data is not immediately needed).

Ganesh then went on to present a file system–level solution to the problem by using a log structured file system for this purpose. Using a log structured file system ensures that new data is appended only to the end of the log. Thus, all the disks that are not currently being written to can be spun down provided that the data that is accessed most often from these disks get cached. In addition, log cleaning can be used to concentrate popular data on the same disk.

Q&A: In terms of how many reads hit the cache, someone mentioned that server disks are more sensitive to power than laptop disks and wanted to know if using laptop disks would mitigate this problem. Another audience member suggested that this problem might be mitigated by buffering before actually performing writes to the disk.

One questioner asked whether the power reduction numbers presented in the talk and in the paper take into account the full machine or just the disk subsystem. The answer was that they only take into account the disk subsystem.

GUARANTEES FOR THE FUTURE

Session Chair: Amin Vahdat, University of California, San Diego

Summarized by Ramakrishna Gummadi (ramki@catarina.usc.edu)

■ ***Can Ferris Bueller Still Have His Day Off? Protecting Privacy in the Wireless Era***

Ben Greenstein, Intel Research Seattle; Ramakrishna Gummadi, University of Southern California; Jeffrey Pang, Carnegie Mellon University; Mike Y. Chen, Intel Research Seattle; Tadayoshi Kohno, University of Washington; Srinivasan Seshan, Carnegie Mellon University; David Wetherall, University of Washington and Intel Research Seattle

Ben Greenstein argued that today's wireless devices severely compromise a user's privacy because of various limitations in the design and implementation of wireless protocols. Such privacy threats should be seen as imposing an economic cost on the threatened users, because the exposed information includes data about browsing history, location history of previously visited access points, information about applications running on a user's computer, and details about capabilities and features of the hardware on the user's computer.

Using measurements from a publicly available SIGCOMM '04 trace of 802.11 network usage, Ben pointed out that exploiting these vulnerabilities allows an adversary to identify and track the locations of more than 25% of the user population with an accuracy of 99% or better. He then outlined the systems challenges involved in building privacy-preserving wireless protocols, such as a privacy-enhanced 802.11 MAC. Such research challenges fall into three main categories: building a naming architecture with anonymity properties better than those afforded by pseudonyms; building resource discovery and binding protocols that let a user search for, select, bind, and then migrate to resources such as access points; and limiting information leakage by preventing the inadvertent exposure of implicit identifiers that allow an attacker to identify and classify the hardware and software being used by a user with high accuracy.

Jason Flinn from Michigan asked whether the privacy and performance metrics in the paper represent commonly accepted metrics, and the speaker replied that he hoped so. Steve Hand from Cambridge asked what the most identifying feature in 802.11 was, and Ben replied that it was the set of IP destinations accessed by a user. Margo Seltzer from Harvard asked whether there is a difference between

her generation and the current one in terms of privacy expectations: Younger people today seem to be more willing to openly reveal their locations and activities on sites such as MySpace. The audience thought that adverse impact in terms of future study and employment opportunities owing to lax attention to privacy could soon cause people to take their privacy more seriously. The final question dealt with the implications of changing the MAC address frequently to provide anonymity, because many protocols use MAC addresses for various functions such as authentication. Ben replied that one could change the MAC address slowly, such as each time you associate to an AP, so that the changed address could be used for authentication, in conjunction with a more permanent address.

■ ***Auditing to Keep Online Storage Services Honest***

Mehul A. Shah, Mary Baker, Jeffrey C. Mogul, and Ram Swaminathan, HP Labs

Mehul Shah talked about the importance of providing a third-party auditing facility for online services such as storage and the challenges involved in provisioning reliable internal and external audit facilities for such services.

Mehul described a catastrophic data-loss scenario for a user using an online storage service provider. The user could not make a well-guided selection of the storage service she used because of lack of reliable information about the relative service qualities offered by competing storage providers. The speaker pointed out that, in the real world, there is already considerable appreciation of the functionality provided by auditors and insurance agents who contract them. He said that there are two main approaches to auditing: external and internal. In external auditing, externally visible interfaces are used to measure and predict the properties of a service. In internal auditing, information about the extent to which a service follows best practices and processes internally to ultimately meet its service objectives is assessed. Both types of audits are needed, and they complement each other. Mehul pointed out several goals and properties of audits and said that auditing was motivated by demands placed by insurance providers or by government regulations.

Mehul then explained the interfaces and hooks that are necessary for storage services to maintain the service's SLAs. A main challenge is to preserve privacy while ensuring data longevity and integrity. The auditing process must also be bandwidth-efficient. Mehul then proposed a privacy-preserving approach for auditors to verify that the data stored by the service providers is correct. It uses hashes on encrypted data and ensures that the key used to encrypt the data need not leave the service provider, while simultaneously guaranteeing that the service provider has not lost any of the user's data. He concluded by saying that there is therefore both a need and a mechanism for efficiently and privately verifying the performance of online storage providers.

David Lie from Toronto asked how one can audit rapidly changing data. Mehul proposed the use of batching. Landon Cox from Duke asked whether an alternative approach would be to keep a separate copy of the data with the customers themselves, or, alternatively, to keep a copy with the auditors. Mehul replied that one of the assumptions of the work is that the customers are not expected to keep any copy of the data they originated with themselves, and that auditors should not necessarily be trusted with the original data. Hakim Weatherspoon from Cornell asked whether it is possible for the auditor to be anonymous. Mehul replied that the solution is then to use data sampling. Gün Sirer from Cornell asked whether the reputation provided by auditing is really meaningful, and Mehul replied that auditing indeed helps develop reputation. Finally, the audience wondered whether moving toward data assessment through auditing or more simply toward better data backup is ultimately the right thing to do.

■ ***A Web Based Covert File System***

Arati Baliga, Joe Kilian, and Liviu Iftode, Rutgers University

Liviu Iftode presented a Web-based covert file system called CovertFS. Liviu first outlined the requirements for a steganographic file system centered on Web services for media sharing and storage: providing plausible deniability, allowing online access and sharing, and providing information hiding for confidential documents and information. He then presented the main design concepts of CovertFS and the challenges involved in providing a file system abstraction build on top of a Web-based system for sharing photos.

In CovertFS, both data and metadata such as inodes are stored in photos. The root of the file system is then accessed through a hash of the encryption passphrase entered by the user. CovertFS includes techniques to hide access patterns that may reveal its hidden purpose behind photo accesses. They include ways to manage frequently changing image data owing to file system writes by using immutable allocation maps and avoiding photo access hotspots resulting from metadata accesses, which could tip off an external party, by using image chains. CovertFS also includes facilities for access control, such as allowing read-only access on publicly shared photos, and replication to manage Web site unavailability. Finally, plausible deniability can be provided by having multiple levels of CovertFS, the top few of which are potentially less incriminating. Finally, Liviu talked about how to manage both active adversaries who can perform steganalysis and passive adversaries who can mount traffic analysis. He concluded by saying that they are currently building a working prototype that they hope to evaluate in terms of latency, scalability, security, and privacy.

Mary Baker from HP wondered whether write accesses can be more covertly managed by pretending that images are being manipulated for common operations such as red-eye

removal. The speaker agreed. Margo Seltzer from Harvard asked whether one can build a cooperative steganographic service and whether the threat model would be different. Gün Sirer from Cornell wondered whether it is possible to use file systems optimized for write-once, read-many workload, such as an ISOFS. Finally, Jason Flinn from Michigan wondered why a file system should be used for covert sharing in the first place. Liviu answered by saying that a file system is an abstraction familiar to users.

PANEL: PUTTING THE SCIENCE IN COMPUTER SCIENCE

Session Chair: Margo Seltzer, Harvard University

Summarized by Vinod Ganapathy (vg@cs.wisc.edu)

Panel Members: Dawson Engler, Stanford University; Butler Lampson, Microsoft; Jay Lepreau, University of Utah, virtualized by Jeff Mogul, HP Labs; Brian Noble, University of Michigan, virtualized by Yuanyuan Zhou, UIUC

Brian Noble started by saying that the good news was that the OS community had begun to think about usability and not just performance. However, the bad news is that we are really poor at evaluating our work. Most work proceeds by conducting a toy user study or presenting anecdotal evidence with excuses such as "We don't need a user study," "Our colleagues thought that our system was neat," and "User studies are too hard!" Noble went on to conclude that the main reason we don't conduct user studies was because of fear of the unknown.

Noble mentioned the need to collaborate and work with experts in the HCI area to conduct meaningful user studies. He encouraged the community to learn the rules, as IRBs are typically not the enemy. He did, however, mention the long cycle times needed to conduct a user study and also the need to overprovision the resources needed [because a subject, once used in a study, cannot be used in that same (redesigned) study once again]. Therefore there is a need to conduct several little pilot studies before the final study.

The second speaker, YY Zhou, spoke about the need to put "Nerdiness" into "Hackers." She had began in Princeton as a theory student, and therefore as what she believed was a "nerd." Her advisor then convinced her to become a "Hacker," and so she is now what she calls a "Neker."

Zhou then described a typical research cycle, in which we select a problem, either based on demands, trends, and challenges in the real world or as a response to other papers from the community, then abstract it, solve it in the abstract, and develop a proof-of-concept solution. The next step is to conduct a user study to evaluate these proofs of concept. She then mentioned that it is time for hackers to admit that nerds can be cool too. For example, in her own work she uses machine learning and data-mining techniques, and she said that the community must consider publishing in venues such as SysML and control theory conferences.

The third speaker, Jeff Mogul, presenting Lapreau's slides, mentioned the need to not just have "reproducible research" but "replayable results." The goal is to have the ability to replay the entire system, i.e., software plus hardware, so that we can fiddle around with parameters and see how the system responds. This is possible using a virtual machine infrastructure, data repositories, experimental management systems, and grids.

The fourth speaker, Butler Lampson, proposed that to bring science to computer systems research, it was very important to write precise specifications for the systems that we build. The techniques and tools needed to write specs have all been developed in research over 10–15 years ago. The basic idea is to build a system and write a simulation proof that the system built indeed conforms to the spec.

Lampson said that by writing a spec, it is often possible to learn things about the system that the designers couldn't have learned otherwise. He mentioned the example of a student whose thesis committee he was on. This student was designing a CVS-like system on P2P over a DHT but had a flaw in his design that became obvious when he encouraged the student to write a spec of his system. Lampson then switched gears to say that scientists, contrary to popular belief, do *not* replicate experiments.

The audience then joined the discussion. One member asked whether a bad user study is better than no user study at all, to which Noble replied that what we currently have are only bad user studies, which is why we need to collaborate with HCI folks. Another audience member mentioned that the time and effort needed to conduct a detailed study such as the one that the panelists were referring to takes a lot of time, which means that the students will graduate without enough papers—students nowadays need at least three or four decent papers in top venues to get good jobs, so isn't there a conflict here? One panelist strengthened this view by saying that even universities nowadays don't hire such people, so we don't breed that kind of culture in our universities now. Another audience member questioned the need for writing specs by asking why specs are important. Isn't impact on the real world more important?

NEW SOLUTIONS, OLD PROBLEMS

Session Chair: Yuanyuan Zhou, University of Illinois at Urbana-Champaign

Summarized by Diwaker Gupta (dgupta@cs.ucsd.edu)

■ ***Purely Functional System Configuration Management***

Eelco Dolstra, Utrecht University; Armijn Hemel, Loohuis Consulting

Keeping software up to date poses a huge system administration challenge, in large part owing to problems with existing configuration management tools: Dependency handling is not perfect, having multiple versions of software on a system is often problematic, configuration files often undergo destructive modifications on updates, etc. In this talk Eelco Dolstra presented a new Linux distribution—NixOS (http://nix.cs.uu.nl/nixos)—built around a purely functional system configuration manager called Nix.

The key insight is that most of these problems arise because of the imperative model used by most configuration tools—users need to describe *how* to get things done as opposed to *what* needs to be done. Drawing motivation from the programming language world, NixOS uses functional programming paradigms such as *referential transparency* to manage system configuration. In particular, Nix is completely stateless, and the entire system configuration is rebuildable from a single, declarative specification. Once built, packages are immutable. Multiple versions of packages are supported and dependencies are propagated in the build system.

A consequence of this design is that if one of the core libraries (say, libc6) undergoes modification, pretty much the entire system has to be rebuilt, resulting in significant storage overhead. Another practical problem arises because NixOS generates configuration files in a declarative manner as well, so there is no easy way to manually modify a configuration file, primarily because tracing the place in the build system where it was generated is nontrivial. Nix does provide repositories with prebuilt packages to save build time for end users.

■ ***Processor Hardware Counter Statistics as a First-Class System Resource***

Xiao Zhang, Sandhya Dwarkadas, Girts Folkmanis, and Kai Shen, University of Rochester

Processor hardware performance counters started out as verification and debugging aids, but they have evolved into a rich source of statistical information invaluable for several applications such as CPU scheduling, self-managing applications, and benchmarking tools. These performance counters are usually managed in hardware and read using some low-level interface by the operating system. In this talk, Sandhya Dwarkadas made a case for hardware counters to be managed as first-class entities by the OS.

Management means providing a high-level API for applications to use, and also virtualizing the counters on a per-process basis. This would allow things such as measuring the number of cache misses for a particular application. There are several applications that would benefit from such facilities: Resource-aware OS schedulers could use hardware counters as input to a counter-based resource model; hardware counters could also be used to distinguish between CPU-intensive and memory-intensive requests to do online workload modeling.

However, there are several issues to be hashed out. Security is a concern: Can counters act as covert channels to leak information? Another concern is performance: Currently, counters are attractive because they are extremely efficient. However, OS management might make counters significantly slower, nullifying the benefits of management. The main push of the paper was to encourage a dialog between hardware vendors and OS developers.

■ ***Microdrivers: A New Architecture for Device Drivers***

Vinod Ganapathy, Arini Balakrishnan, Michael M. Swift, and Somesh Jha, University of Wisconsin—Madison

It is well known that the bulk of the bugs in OS code come from device drivers, primarily because device drivers are hard to get right, extremely hard to debug, and often written by those who are not kernel experts. Vinod pointed out, however, that the fundamental problem was the architecture of monolithic kernels (e.g., Linux). Since the device driver runs in the kernel's address space, a faulty driver can easily take the whole system down. Earlier projects have tried to address this issue by moving device drivers to user space, but these approaches suffer from either poor performance or incompatibility with commodity OSes.

Microdrivers is a new approach to building device drivers that is both efficient and backwards-compatible. The key idea is to split each driver into a kernel driver and a user-space driver interacting over a driver runtime. Performance-critical functionality sits in the kernel driver; the rest is delegated to user space. However, the real clincher for this approach is that existing device drivers can be semi-automatically converted to microdrivers.

The code generation takes place in two stages. First, a "splitter" detects function-level split in driver code. Second, the "code generator" takes as input marshaling annotations required for serializing complex data structures and outputs code for the different components.

This talk sparked a lot of discussion and questions. One of the biggest complaints was that the evaluation in this paper doesn't actually implement the microdriver approach (since everything executes in the kernel). Another observation was that many bugs usually occur in corner cases and a split based on functionality might not be able to capture it. Clean maintenance of split drivers as the upstream driver code evolves also poses a challenge.

4-MINUTE MADNESS

Session Chair: Rebecca Isaacs, Microsoft Research Cambridge

No summary available, but see the Sirer-Farrow article in this issue, based on a talk from this session.

WEB 2.0

Session Chair: David Wetherall, Intel Research and University of Washington

Summarized by Vinod Ganapathy (vg@cs.wisc.edu)

■ ***MashupOS: Operating System Abstractions for Client Mashups***

Jon Howell, Microsoft Research; Collin Jackson, Stanford University; Helen J. Wang and Xiaofeng Fan, Microsoft Research

Howell presented background on Web 2.0 technologies such as AJAX and also browser policies, such as the Same Origin Policy. He then presented Mashups, for example, where housingmaps.com uses information from both Google Maps and Craigslist to present a listing of available houses.

Howell mentioned that the problem was the binary security model of the Web, where content from domains can be isolated perfectly using IFRAMES, or has no security at all (e.g., a SCRIPT executes in the context of the page that includes it). Howell thus argued for the need for sophisticated interaction among various components of a page. The solution is a new abstraction called a ServiceInstance, which is akin to a process on an OS. Each service instance is associated with a single domain, and each resource has its own service instance. Service instances are created by using the FRIV tag (a new construct introduced in HTML). Service instances allow for limited communication, and they are thus a hybrid of FRAMES and DIVS. Howell also went on to mention that MashUpOS can be implemented with script rewriting (e.g., using the Browsershield framework).

The questioners asked whether service instances were akin to adding a new element to the process hierarchy, and if so, why this was indeed a right abstraction. Another questioner also asked how to label pages from different domains: Should the browser do it, or should applications do so cryptographically? Howell said this was an issue they were currently examining.

■ ***Live Monitoring: Using Adaptive Instrumentation and Analysis to Debug and Maintain Web Applications***

Emre Kıcıman and Helen J. Wang, Microsoft Research

Emre Kıcıman said the motivation for this work was that huge amounts of code were downloaded on the client side in today's Web 2.0 sites, with sites such as Google Maps downloading up to 50,000 lines of code to the client side to improve user experience. Thus lots of code executes on the browser, and there are third-party dependencies, too (e.g., using Mashups).

Kiciman explained the need for end-to-end visibility, which would help Web application developers better understand and tune their applications. He argued for an on-the-fly rewriting technique that would be deployed with the code that is downloaded on clients, which would help the application developer with issues such as performance and correctness debugging of these applications. The key was that this offers a different deployability model, where deployment is immediate, with very fine-grained control over who's using what instrumentation. Also, all this is possible in their system without any changes to the server or the client (and is done via Javascript rewrites).

In the Q&A session, one questioner expressed concern about the "willy-nilly" rewriting being performed by their system, and about all this code executing in the browser, which he mentioned did not make him feel very comfortable from a security standpoint. A second questioner asked whether different instrumentation at different sites would cause problems in understanding bugs.

■ ***End-to-End Web Application Security***

Úlfar Erlingsson, Benjamin Livshits, and Yinglian Xie, Microsoft Research

Úlfar Erlingsson said that today's data transmitted over the Web is rich data; that is, it can contain embedded scripts, which can be used to launch cross-site scripting attacks. The standard solution adopted nowadays is serverside sanitization (e.g, by disallowing scripts). However, this is a hard problem to solve, because the server must now parse the scripts in exactly the same way that browsers do. Bugs in doing so can result in security holes and worms (e.g., the Yamanner Yahoo! mail worm, the Samy Myspace worm).

Erlingsson went on to argue that security applies to both servers and clients, and he described a mechanism called METS that achieves this. METS allows expression of security policies. The idea is that these will be specified by the server but will be enforced at the client (within the browser). Thus, METS ensures high-fidelity enforcement of security policies. The basic idea behind METS is not new—they are much like inline reference monitors.

One questioner mentioned that currently the onus of enforcement was on the browser, so METS relies heavily on browser manufacturers to ensure security. Is this practical? A second questioner said that the problem was that today's Web languages such as HTML lack a real specification and that several current problems could go away if we have a precise specification for HTML. A third person questioned the practicality of having Web server designers writing METS policies. Web server designers in today's environ-

ment can be as clueless as an end user. How practical is it to assume that they will be able to write meaningful policies?

FINDING A BETTER WAY

Session Chair: George Candea, EPFL

Summarized by Diwaker Gupta (dgupta@cs.ucsd.edu)

- ***HotComments: How to Make Program Comments More Useful?***

  Lin Tan, Ding Yuan, and Yuanyuan Zhou, University of Illinois at Urbana-Champaign

All of us who have programmed understand the value and pitfalls of source code comments. In this talk, Lin Tan discussed the feasibility of analyzing comments and detecting inconsistencies between code and comments. This is a fairly tall order: Bear in mind that comments are imprecise, unstructured, and often written in prose and cannot be tested or verified. However, empirical evidence suggests that often programmer assumptions and expected usage (for instance, code that requires a lock to be held) are most succinctly captured in source code comments. Failure to convey these assumptions to users and other developers often leads to bugs.

The goal of this work is to leverage these assumptions and example usages to detect inconsistencies between code and comments (and thereby detect potential bugs). A code-comment mismatch indicates either a bug or a wrong comment, both of which can lead to bugs, so detecting mismatches is certainly useful. But how feasible is it to extract structure from comments? This work uses Natural Language Processing (NLP) along with clustering on topics such as "locks" to analyze comments: Comments are mapped to predefined templates.

The authors extracted 530 rules from 5 different Linux subsystems and detected 12 new bugs (2 of which have been confirmed by Linux developers). For lock-related and call-related topics, the system "just works" and almost no user intervention is required. For other kinds of comments, more templates need to be defined.

- ***Towards a Practical, Verified Kernel***

  Kevin Elphinstone and Gerwin Klein, National ICT Australia and the University of New South Wales; Philip Derrin, National ICT Australia; Timothy Roscoe, ETH Zürich; Gernot Heiser, National ICT Australia, the University of New South Wales, and Open Kernel Labs

As a foundation for building secure systems, researchers have advocated a small Trusted Computing Base (TCB) that can be manually audited. This paper takes a much stronger stand: Kevin presented their efforts at building a formally verified kernel about which properties can be proved and a guaranteed correct implementation can be provided. The idea is to start with an abstract model and transform it all the way down to a high-performance implementation in C and then assembly code.

One obvious implication of this approach is that any code changes will invalidate proofs. To avoid this, the kernel model is made as detailed as possible—this is done in Literate Haskell. The documentation is embedded in the code of this abstract model, so a single spec can generate the reference manual, as well as a kernel prototype in Haskell that can be run through a user-level simulator. The kernel is modeled as a big state machine with events as inputs—things such as manipulation of low-level state (page tables) as well as preemption have also been modeled.

The kernel prototype in Haskell is about 3,000 lines of code with an accompanying 53,000 lines of code of proof! There exist a proof of termination and a proof of correctness for all but one system call. The authors stated that the transformation from Haskell to an actual C version would be manual, since automatic code transformation would miss significant opportunities for optimization.

- ***Beyond Bug-Finding: Sound Program Analysis for Linux***

  Zachary Anderson, Eric Brewer, and Jeremy Condit, University of California, Berkeley; Robert Ennals and David Gay, Intel Research Berkeley; Matthew Harren, George C. Necula, and Feng Zhou, University of California, Berkeley

Jeremy began by noting the difference between bug finding and soundness analysis: Bug finding involves heuristics and approximations; soundness, in contrast, ensures the *complete absence* of a particular class of bugs. Soundness has been perceived to be extremely hard to attain in practice, and the goal of this work is to make soundness analysis practical for a large system such as Linux. There is some confusion in terminology, since soundness and completeness have slightly different meanings in the literature.

The authors use a combination of lightweight annotations and hybrid checking to make incremental progress toward reliable software. The new goal for system evaluation is to "minimize the amount of untrusted code" as opposed to "maximizing the number of bugs found." They began with a limited subset of the Linux kernel (around 400,000 lines of code) and subjected it to three different analyses:

- Deputy: memory and type safety checks
- Count: deallocation safety
- BlockStop: call graph analysis to identify interrupt handlers that may block

Overall the authors conclude that these analyses can be done efficiently and new tests can be added to make the tests more conclusive. All the code and findings are available at http://ivy.cs.berkeley.edu. Surprisingly, this approach is not effective at *finding* bugs.

CHIMIT '07: 2007 Conference on Human Interfaces to the Management of Information Technology

Cambridge, MA
March 30–31, 2007

Summarized by Alva Couch (couch@cs.tufts.edu) and Marc Chiarini (marc.chiarini@tufts.edu)

Just what do system administration and ethnography have in common? One might think that ethnography is about studying cultures, but in a broader sense, ethnography means "writing about and describing humans and human behavior." For better or worse, system administration involves many human-to-human and human-to-machine interactions. Ethnographers have been studying the behavior of system administrators and other technical professionals, and they have come to some surprising conclusions about the nature of our profession, and how and why we interact socially with humans and technically with machines and networks.

CHIMIT '07 brought together human factors experts, system administrators, and researchers to study how the practice of system administration is affected by human factors such as software interfaces and social context. Papers analyzed how system administrators work, analyzed failures of human process, proposed novel interfaces to administrative data, and suggested improvements in process, software, and structure of support organizations. Unlike the typical LISA paper, which describes "how" best to perform a task or perhaps the options available for addressing a problem, papers at CHIMIT concentrated upon "what system administrators really do" and "why things are the way they are" from a human perspective. The conference was sponsored by ACM, in cooperation with USENIX.

OPENING PLENARY

■ *Information Technology in the Wild*

Stephen Barley, Stanford University

Stephen Barley has been engaged in more than 20 years of ethnographic studies of technical professionals and how they fit into organizations. "Barley's law" is that "what you get is never quite what you plan." First-order effects of employing a technology are often followed by second-order effects that are primarily sociological in nature. Technologies alter work practices, which change or create social divisions, which become demographic divisions (adopted as part of personal identity) over time.

Barley spun an elaborate tale illustrating this principle, starting with the nineteenth-century industrial revolution. The divide between professionals and technical support staff is—in many contexts—a sociological second-order effect of the introduction of technology. Barley's definition of a technician is someone who "creates a bridge between the real and the representation." The accepted myth is that professionals (such as those responsible for interpreting X-rays and business data) can do "anything that technicians can do." Barley's ethnographic studies show that there is little overlap between "professional" and "technical" staff expertise sets and neither is well-equipped in the skills and knowledge necessary to do the job of the other.

Stories of technological change teem with examples of unintended sociological consequences. Telecommuting was first promoted as a way to reduce pollution. Companies could "comply" with the Clean Air Act by setting up telecommuting programs, even though these programs statistically account for the activities of less than 1% of the workforce. This led to a large number of telecommuting programs that few employees used, along with marketing ploys to attract business from the almost insignificant population of telecommuters. Barley and others collected over 3000 articles on telecommuting over many years and demonstrated—through ethnographic analysis—how the story evolved over time.

Another social factor is the rise of the independent consultant. The consultant is perceived as a free agent who takes long vacations and pursues a leisurely existence. Ethnographic studies show that consultants work harder, and for longer hours, than their full-time counterparts, and they often ignore long-term financial planning, which is built into most employee compensation plans. Roughly one-half of all consultants surveyed have no retirement savings plan at all. Conversations after the talk revealed an innovative strategy: There are consultant organizations that function like "employers of record," allowing consultants to "appear" to be fully employed and accrue benefits through the organization without answering to the organization.

Barley asserts that positive change in organizational structure and sociology seldom comes from inside an organization; there must be a "contravening force" to accomplish change. The nineteenth-century evolution of the manager created a situation analogous to that of user versus system administrator and was only countered by the evolution of contravening labor unions and organizations. The information revolution has created a divide that separates "professional" and "technical" staff in much the way that the industrial revolution separated "management" from "labor," and Barley's assertion is that only contravening forces similar to guilds or labor unions will create a balance between "professional" and "technical" needs.

PAPER SESSION I: STUDIES OF SYSTEMS MANAGEMENT

- ***Design Guidelines for System Administration Tools Developed Through Ethnographic Field Studies***

  Eben M. Haber and John Bailey, IBM Almaden Research Center

System administration tools have several weaknesses that can be discovered by ethnographic methods and that have not been suggested by system administrators themselves. Using direct observations of practice as a guide, the authors identify several improvements that could be made in system administration and configuration management tools, including support for planning and rehearsal, support for long-running change operations with limited change windows, and non-blocking user interfaces that do not force the system administrator to wait at the terminal for long operations to complete. Other suggested enhancements include progress indicators for long operations as well as execution time prediction.

Configuration management is an especially hard problem which deserves special attention and tool capabilities. Observation of system administrators indicated that enhanced collaboration tools for quick exchange and comparison of information are needed and that communication and trust between administrators is often a major bottleneck. To illustrate this, a video shows a chat and phone session between two administrators in which trust and disbelief play a major part in impeding troubleshooting. One administrator cannot believe that another is correct in asserting that the base configuration of a device has changed. It is the communication between the two administrators that is the bottleneck, rather than the technology.

- ***Deciding When to Trust Automation in a Policy-Based City Management Game: Policity***

  Kenya Freeman Oduor, IBM Software Group; Christopher S. Campbell, IBM Almaden Research Center

Trust is a major factor in accepting "self-managing" systems as part of IT infrastructure. It is difficult to observe the social bases of system administrator trust directly, but one can reason from analogy to other kinds of expert systems. In this paper, the authors utilized relationships between players and expert "assistants" in a computer simulation game as analogous to relationships between system administrators and autonomic control systems. The chosen game was Policity, a computer simulation in which players try to run a large city by making infrastructure decisions.

Researchers embedded reliable and unreliable "assistants" in the game, asked students to play the game, and observed their reactions and evolving trust relationships. Some students were given access to high-reliability assistants, whereas others' assistants functioned at a lower level of reliability and sometimes gave poor advice. Observation of actions showed that unreliable assistants were eventually ignored by the players, and reliable assistants were also ignored, but to a lesser degree. One conclusion was that trust is not a simple matter of reliability; it also requires some exposure of the underlying decision process. This study suggests that, to be trusted, an autonomic management solution should reveal some of the logic behind its decisions and/or statistics on the accuracy of previous decisions.

- ***Towards an Understanding of Decision Complexity in IT Configuration***

  Bin Lin, Northwestern University; Aaron B. Brown and Joseph L. Hellerstein, IBM T.J. Watson Research Center

Low-level configuration procedures for machines are error-prone, are complex, and lead to inconsistencies in service, but many administrators still perform much—if not all—configuration by hand. To evaluate whether automation is an appropriate substitute for manual changes, it is appropriate to seek metrics for the complexity of the human part of the process. The authors identify three kinds of complexity:

- Execution complexity: How complex and error-prone is the execution of commands?
- Parameter complexity: How difficult is it to give commands appropriate parameters?
- Memory complexity: How much must one remember in order to configure systems effectively?

To understand how these kinds of complexity could affect nonexpert system administration, the authors appeal to an analogous system: route planning on a map.

Human subjects were asked to solve a variety of route-planning problems and their attitudes and actions were observed. Initial results showed wide variations in human performance. A factor analysis showed that task completion time varied most as a result of the amount of guidance given and constraints imposed. An analysis based upon only these factors yielded some surprising results.

It is a popular belief that showing guidance will shorten the time taken to complete a task. In this study, showing guidance information lowered the user-perceived difficulty of route planning but did *not* improve their performance in using the tool. In fact, task times were similar in the two cases, whereas error rates were somewhat lower when guidance information was *not* shown. This very counterintuitive result suggests that guidance information may be an impediment rather than an aid when attempting to assist nonexpert system administrators in completing complex tasks.

PAPER SESSION II: DESIGNING FOR IT MANAGEMENT

- ***Cube Management System: A Tangible Interface for Monitoring Large Scale Systems***

  Elliot Jaffe, Aviva Dayan, and Amnon Dekel, Hebrew University of Jerusalem

In large installations, it is especially difficult to analyze and exchange logging information for technical problems in a way that is easily exchanged with others. The Cube Management System (CMS) is a log analysis system based upon an innovative tangible interface. Physical "cubes" are arranged in a grid in an interface in which cubes can be physically picked up and moved around. Each cube is an active computing device whose location in the grid corresponds to the identity of a processing element such as a rack, blade, or disk. The status of the device to which a cube corresponds is indicated by three LEDs: red for trouble, yellow for marginal operation, and green for proper function. Each cube is labeled (via an alphanumeric display) with the component to which it corresponds, and it also contains current log information and problem descriptions. Cubes can be moved to any physically accessible location in the IT infrastructure and also passed among system administrators in order to share information. Prototypes have been implemented both with physical cubes containing standalone computers and with virtual cubes in a 3D graphical interface.

Trouble isolation and delegation of responsibility in CMS are accomplished via a "zooming" mechanism that changes cube identities in the grid to represent different kinds of components at different functional levels. A system administrator places a cube of interest into a special tray on a grid, which has the effect of changing the grid so that it represents components at another level. Thus one can very quickly zoom down from, for example, a rack with a problem, to the blades in the rack with problems, to the disks of each blade that have problems. The mechanism preserves context by allowing one to set aside a cube at one level and return to it after having drilled down to a problem spot. By shuffling cubes on the work surface and handing them to technicians responsible for problem resolution, one can quickly locate, identify, and resolve problems at varying scales. An additional advantage is "ambient information transfer": A colored light allows multiple technicians to be subtly aware of the health of monitored systems without taxing other important cognitive functions.

- ***Activity-based Management of IT Service Delivery***

  John Bailey, Eser Kandogan, Eben Haber, and Paul P. Maglio, IBM Almaden Research Center

Outsourcing has been thought to be a panacea for a variety of information technology problems, but it is difficult to transition from insourced to outsourced IT solutions. The ethnographic study of the outsourcing process in a particular organization suggests that the key to a smooth transition is a "transition manager" who coordinates design and deployment staff. Current transition managers oversee the transition process via a spreadsheet of task states and/or existing project management or ticketing software.

The transition manager actually requires five kinds of information:

- People: Staffing needs for each task
- Resources: Requirements for hardware, connectivity, etc.
- Calendar: Schedule and expectations for task completion
- Tools: Integrated email mechanisms for "context passing" from manager to staff and vice versa
- Process: An ITIL diagram of the transition process and its state

The paper proposes that a "big board/shared workspace" for the transition manager would greatly improve the efficiency of the transition process, integrate data sources, and make outsourcing practical in more cases. The workspace would need to extend traditional tasks with ad hoc steps, support team collaboration, and support reuse of ad hoc activity knowledge (including mining and analysis of such). An added benefit would be improved cost estimates of service delivery activities. The big board is not without obstacles, of course: How does one assure the efficient and accurate mapping of activity data? What are the requirements of the data mining and analysis components? How can the "Rockwell Effect"—in which staff feel their every move is being recorded and analyzed—be mitigated? Will filters, "perspectives," and alerts be sufficient to help the transition manager deal with information overload?

- ***IT Ecosystems: Evolved Complexity and Unintelligent Design***

  Jim L. Lentz and Terry M. Bleizeffer, IBM Software Group

Most IT infrastructures are not designed from scratch; rather, they evolve over time as a result of external and unforeseen forces, leading to final designs with significant flaws. The results of this evolution are often inferior to designing a new solution from the top down. Evolving via small changes can lead to ad hoc employee organization, poorly interlocked and defined responsibilities, overly diverse IT environments, and overly "personalized" IT practices.

A different set of design issues is encountered when the problem is approached from the top down. "Unintelligent design" practices include:

- Concentrating upon a small subset of the overall solution
- Making simplifications that make the product less useful
- Relying upon GUI design alone without considering process
- Blaming architects for complexity of the ecosystem
- Believing that all customer requirements must be addressed exactly as described

Getting control of complexity in an evolved ecosystem is difficult. The authors suggest strategic encapsulation of complexity. Complex parts of process are limited to those that are mission-critical, and domain experts are trained to manage those processes via specialization, much as the general practitioner often refers a patient to a specialist. In this way, the final design preserves the business process and places expertise where it matters, in the most complex parts of the process.

PANEL

- ***Is Automation a Panacea for Management of Information Technology?***

  Michael Beck, Emerging Technologies Group; David N. Blank-Edelman, Northeastern University; Tom Limoncelli, Google; Paul P. Maglio, IBM Research

  Moderator: Alva Couch, Tufts University

Is automation a panacea for management of information technology? Yes and no. The panel was in some sense in agreement on this answer: Yes, for the easily automated processes; no, for the inherently human ones. Human processes include dealing with policies and workflow, including account management. The practical approach of capturing scripts and replaying them works in many cases, and autonomics promise to automate all but troubleshooting in the long run.

DINNER INVITED TALK

- ***Interfaces for Everyone***

  Rob Kolstad, Ph.D.

Interfaces—including complex ones—are everywhere. The talk began by a look at the number of buttons and dials and indicators that the average person has to manipulate in order to prepare for work on a given day, including the alarm clock, shower, toaster (with labels in German), shoelaces (including hundreds of ways to tie laces), and automobiles. This was followed by a detailed tour of computer interfaces through history, starting with the abacus and ending with the Wii. Throughout history and a regular day, we deal with fairly complex interfaces that require training and context. History shows, as well, that interface developments occur in spurts with long intervening pauses. The last interface development occurred over 15 years ago.

A common way of designing interfaces is to refer to "design patterns," but Rob believes that many of these have become "design ruts." These include the incessant insistence upon use of a mouse when a keyboard will do; the lack of respect for users' prior training in typing; and the idea that command-line interfaces are "too dangerous" or "provably less effective than CLIs." The subsequent discussion included the question, "Should people have a driver's license in order to utilize a CLI?" There was no clear answer.

PAPER SESSION III: TECHNOLOGY AND USE

- ***Network-Centricity: Hindered by Hierarchical Anchors***

  Steve Abrams and Gloria Mark, University of California, Irvine

Ethnographic study of the interactions of a failed design team exposes several social "anchors" that impede progress when moving from hierarchical ("stovepipe") to network-centric business infrastructure. A division of a large company wished to institute a division-wide calendaring system and combine data from several existing site-specific calendaring systems into one coherent calendar. A design team composed of members from three sites was given six weeks to design a new calendaring solution to be adopted by all sites. Ethnographic researchers observed all communications and meetings and analyzed the results of the interactions.

In the first meeting, two design philosophies emerged that kept the sites from communicating and compromising. The "bottom-up" camp was concerned with pleasing the boss and integrating existing technologies, while the "top-down" camp was concerned with surveying user needs and selecting a new and distinct division-wide solution. Four kinds of unresolvable differences and inflexibility arose:

- External authority: An authority figure requires one kind of solution.
- Technology: Our current design only supports one kind of solution.
- Asymmetric communications: Leaders are unresponsive to communications from other site representatives.
- Work commitments: Other job requirements interfere with deliverables.

These "anchors" were repeated throughout three weeks of meetings, in different guises, and finally led to team disbandment. The key lesson is that any successful transition process must necessarily avoid these patterns of interaction.

- ***Managing Technology Use and Learning in Nonprofit Community Organizations: Methodological Challenges and Opportunities***

  Cecelia Merkel, Umer Farooq, Lu Xiao, Craig Ganoe, Mary Beth Rosson, and John M. Carroll, Pennsylvania State University

This study concerns how to develop sustainable information technology in small-scale nonprofit organizations. Small nonprofit organizations often suffer from ad hoc planning processes, a lack of risk analyses, and migratory volunteer workforces, leading to impromptu and "opportunistic" IT development strategies: Make changes while

someone is available to make them. This often leads to unreliable or even unusable IT services.

Researchers utilized the "participatory action research" model in which they are both partners and observers of technology development practices. In several case studies, they attempted to partner, suggest improvements, and then "fade out" of the picture to leave a more sustainable infrastructure.

The case studies illustrate the usefulness and limitations of technology partnering. A food bank providing 12,000 bags of groceries to 800 households was hampered by the lack of a technology plan and needed help in transitioning from ad hoc to structured IT planning. A watershed council was stalled on a Web site design issue; partners helped them define audiences of the Web site and tune it to the organizational mission. A historical society suffered from an ad hoc approach to technology in which long periods of stasis were interrupted by occasional grants; partnering failed to remedy lack of interest in moving toward a more stable solution. Lessons learned include that partnering can help nonprofits engage in planning processes, leverage domain expertise in designing or improving IT infrastructure, and work toward acceptance of the need for change.

■ ***Supporting Expertise Awareness: Finding Out What Others Know***

Christian Dörner and Volkmar Pipek, University of Siegen; Markus Won, University of Bonn

This paper describes an attempt to utilize activity log information to characterize expertise profiles of members of a "freelance network" of more than 200 technical professionals. Rather than using a "yellow-pages" approach in which each member lists his or her domains of expertise, the authors engaged in a three-year study in which they attempted to infer members' domains of expertise from logs of member activities such as browsing Web sites and interacting with newsgroups. Members who posted to particular newsgroups and/or read expertise-centric news sites or documents were deemed to have more expertise, based upon the time intervals in which they sustained such activity.

The expertise filtering is accomplished by an environment called eXact, which takes as input a graphical representation of the filtering to be done. The filtering is done in several stages, including collection of the statistics, filtering for privacy, and generation of hypotheses that match gathered data. These hypotheses are collected in a "knowledge garden" and used as additional information in building teams and training new members. The result is a set of hypotheses that—in practice—represent "extra information" but are not considered definitive as an expertise measure.

Two major obstacles were encountered during this study. First, there was no resolution to the "eternal struggle" between privacy and the need to characterize individuals. Second, it would be possible to "fake" expertise data by generating meaningless log entries, though the authors assert that there would be strong negative social ramifications of such behavior if discovered, including ostracism.

PAPER SESSION IV: USABILITY AND SECURITY

■ ***Looking for Trouble: Understanding End-User Security Management***

Joshua B. Gross and Mary Beth Rosson, Pennsylvania State University

In this ethnographic study, twelve users handling sensitive data were surveyed about their attitudes toward security. What do users know, how do they manage it, and whom do they perceive as responsible? The end result of this work is to define "personas" that can be utilized in designing better security policies. A "persona" is a personality type, defining likelihood of specific behaviors, unlike a "role," which instead describes responsibilities. Informally, three kinds of personas arose in the study: "vigilant," "reliant," and "careful." A vigilant person takes personal responsibility for security, a reliant person places the responsibility for security upon others, and a careful person relies upon incomplete knowledge and is "careful" within that context.

Eleven of the twelve users exhibited significant lack of knowledge about security issues. One user reported "I feel lost." There is serious concern over security; one user reported "I would never work in this field again if I were involved in a major incident." But there is a disconnect between "concern" and "responsibility." Only one of twelve users took personal responsibility for security; the others placed responsibility on technical solutions or staff. This lack of knowledge extends to physical security; several participants had confidential files open on their desks during the interview.

The authors conclude by listing several methods whereby the knowledge of users about security might be improved. These include activities that build trust in institutional policy, use of information escrow (in which a trusted human agent relays all sensitive information), and adding visual and audio cues in software to increase trust and social presence. Knowledge of the personas of users can be utilized in designing better risk-reduction measures.

■ ***User Help Techniques for Usable Security***

Almut Herzog and Nahid Shahmehri, Linköping University

Security is not a user's primary task; understanding current security mechanisms involves specialized knowledge that almost no users possess. A typical security pop-up, such as "Do you wish to trust this IP address?" usually translates in the mind of a user into "Do you want to get your work done or not?." Thus these pop-ups, in the context of an average user, provide no protection and are merely an an-

noyance. To address this, one must express to the user—clearly and concisely—the nature of the decision to be made and its impact. The authors consider the role of online help in aiding the user to make intelligent security decisions.

There are several forms of online help, including documentation, context-sensitive help, assistants, wizards, staging, and social navigation. Staging refers to the process of training the user in steps toward more advanced security aims (e.g., training the user on one new security idea per day). Social navigation refers to the practice of showing the statistics for each decision for other users on the network; this is controversial because the majority may in fact choose incorrectly. Several capability matrices show the strengths and weaknesses of each approach but in the end all are less than reliable and offer no substitute for built-in security mechanisms that cannot be overridden. The authors indicate a "slight preference" for wizard-based mechanisms in cases where changes are made infrequently.

BSDCan—The Technical BSD Conference

Ottawa, Canada
May 18–19, 2007

USENIX is a Gold Level Sponsor of this event.

■ *Open Source Security Lessons*

Wietse Venema

Summarized by Julian C. Dunn (jdunn@aquezada.com)

Wietse Venema is perhaps most well-known as the author of the Postfix mail transport agent, which is arguably just as popular as Sendmail, the granddaddy of all mail transport agents. He began his talk with an amusing interlude showing how difficult it is to quantify the popularity of Postfix using Google Trends, given that the term "Postfix" has multiple definitions.

Before Postfix, Venema was instrumental in creating SATAN (Security Administrator's Tool for Analyzing Networks) and, prior to that, the widely deployed TCP Wrappers, which continues to be shipped with many open-source operating systems. He shared some of the lessons learned from his time spent in the open-source security space, not only in terms of the technology but in terms of publicity as well. For example, Venema showed several media quotations prior to the release of SATAN that claimed the tool would cause widespread destruction on the Internet. Of course, when the tool was released, no appreciable increase or decrease in system compromises was recorded, according to the various CERT organizations Venema surveyed.

Venema has had much better media relations around the release of Postfix, claiming that a single *New York Times* article heralding the release of the "IBM Secure Mailer" project (as Postfix was then known) single-handedly changed IBM's attitude toward open source, eventually convincing the company to become involved in many open-source communities such as the Apache Foundation and Linux kernel development. Venema concluded his talk by humorously alluding to the fact that even though his original motivation in writing Postfix was to provide an alternative to the complexity of Sendmail, the number of lines of code in Postfix has now exceeded that in Sendmail. However, he now considers the feature set of Postfix to be more or less complete.

■ *Recent Improvements to the FreeBSD Ports Monitoring System*

Mark Linimon

Summarized by Bill Moran (wmoran@potentialtech.com)

Mark Linimon provided some demonstrations of his ongoing work to tame the FreeBSD ports system. The FreeBSD ports system provides a convenient method for building and installing third-party software on FreeBSD, and it currently includes over 17,000 applications. Mark has done a great deal of work creating a reporting mechanism that summarizes much of the development of the ports system so that problems can be more easily discovered and addressed. The results of his efforts can be seen at http://portsmon.freebsd.org/.

■ *Network Stack Virtualization for FreeBSD 7.0*

Marko Zec

Summarized by Bill Moran (wmoran@potentialtech.com)

Marko Zec (http://www.tel.fer.hr/zec/) demonstrated his work virtualizing the FreeBSD network stack. By abstracting the vnet structure an additional layer, Marko was able to create completely independent networking environments within a single FreeBSD instance, each with its own IP information and routing table, thereby providing an excellent opportunity to use FreeBSD as a network research platform or to improve FreeBSD's existing jail system. A live CD is available for download from http://www.tel.fer.hr/imunes/, and Marko is working to get his improvements merged into the mainline FreeBSD source tree.

■ *Varnish HTTP Accelerator*

Poul-Henning Kamp

Summarized by Julian C. Dunn (jdunn@aquezada.com)

Poul-Henning Kamp is a FreeBSD kernel developer who has worked on a multitude of both kernel-space and "userland" applications ranging from disk encryption to embedded systems. Lately, he has been working on the Varnish HTTP Accelerator project (http://varnish.projects.linpro.no/), which aims to provide inbound HTTP acceleration for busy Web sites such as VG (http://vg.no), the Web site for a popular Norwegian newspaper.

Kamp began by explaining why Squid, the classic HTTP caching solution, is programmed poorly. He outlined the methods in which it "fights the kernel" by trying to explic-

itly separate memory and disk storage. He denounced this methodology, saying that because the kernel provides virtual memory services there is no need for user applications to do this work. Doing so results in excessive system calls, which lowers performance. In contrast, Varnish simply allocates large objects in virtual memory and lets the kernel manage memory in the optimal way.

Varnish also maximizes performance in many other ways by using careful programming tactics, for example, by avoiding expensive text-processing operations if they can be avoided. In addition, Varnish's configuration language (VCL) is preprocessed and compiled into binary, then dynamically loaded for speed. Multiple configurations can be loaded concurrently and an interactive command-line interface (CLI) manager can switch configurations, in addition to doing other cache operations such as purging objects, retrieving cache statistics, and so on.

Future work on Varnish will see features such as edge side includes and URL rewriting added. Kamp hopes to eventually see the project moved under the auspices of the Apache Software Foundation, as there would be a natural synergy with the Apache HTTP Daemon.

■ ***FreeBSD Security Features***

Robert Watson

Summarized by Bill Moran (wmoran@potentialtech.com)

Robert Watson (http://www.watson.org/~robert/) gave an excellent overview of his ongoing work extending the FreeBSD security model, first providing an overview of ACLs (access control lists). ACLs offer an extremely flexible method of describing permissions on filesystem objects. Unfortunately, the two leading systems of ACLs, POSIX and NT, are not compatible. FreeBSD supports POSIX ACLs, but there is interest in supporting NT ACLs, since NFSv4 uses them. Robert also described the powerful security auditing tools introduced in FreeBSD 6.2. These tools are required by Orange Book and other evaluations and provide a method for fine-grained monitoring of systems. FreeBSD's audit tools are based on Solaris and Mac OS, but these tools can be extended with the concept of audit pipes, which allow the administrator to create multiple filters of audit events. Finally, Robert covered mandatory access controls (MACs), which supplement discretionary access controls (such as filesystem permissions). If you've worked with SE Linux, the MAC framework will feel familiar.

■ ***Portsnap***

Colin Percival

Summarized by Bill Moran (wmoran@potentialtech.com)

Colin Percival described the road he took to developing portsnap, an updating tool specifically designed for the FreeBSD ports tree. Because of his role as FreeBSD security officer, Colin started writing portsnap to make the distribution of port updates more secure, but he also managed to significantly improve the speed and bandwidth usage by writing a customized compression program called bsdiff that is aware of byte substitutions. I was also interested to hear Colin describe existing technologies, such as HTTP pipelining and DNS SRV records, that are largely unused but could solve many problems plaguing the Internet.

■ ***How Open Source Projects Survive Poisonous People***

Brian Fitzpatrick and Ben Collins-Sussman

Summarized by Bill Moran (wmoran@potentialtech.com)

Being a member of several groups (not all of them open source software groups), I decided to attend the lecture by Ben Collins-Sussman and Brian Fitzpatrick on how groups can survive poisonous people. Ben and Brian took turns covering various aspects of four tenets: comprehension, fortification, identification, and disinfection. I found their insights enlightening, but the highlight was when they asked the room if anyone knew what "bikeshed" referred to, only to find that not only did everyone in the room know, but the man who popularized the phrase, Poul-Henning Kamp, was sitting in the back of the room.

■ ***Failover and Load Balancing with pfSense***

Scott Ullrich and Chris Buechler

Summarized by Chris Buechler (cbuechler@gmail.com)

I was one of the presenters for this session and a co-founder of this project. pfSense is a FreeBSD-based firewall distribution using OpenBSD's pf packet filter, with a Web interface for configuring all aspects of the system. This presentation focused on the failover and load balancing functionality available in the system.

Five main topics were covered: CARP, multi-WAN failover, policy-based routing and failover, DNS failover, and incoming and outgoing load balancing.

CARP allows for seamless hardware failover, to accommodate hardware failure, or firewall maintenance and upgrades without loss of connectivity. Typical CARP configurations and deployments were discussed.

Multi-WAN failover allows the use of multiple Internet connections, and upon failure of a connection, the remaining WAN connection(s) can be automatically used to maintain connectivity. Common deployment scenarios were illustrated and discussed.

Policy-based routing and failover enables routing of traffic based on IP protocol, TCP or UDP port, and source or destination IP, among other possibilities. Upon failure of the preferred routing destination, backup destinations can be utilized. Generally this is configured in combination with multi-WAN. Common configurations were given.

DNS failover combines with the previously mentioned functionality to update your public DNS records upon failure of a WAN connection. This enables the multi-WAN functionality to be used for inbound access from the Internet, with automated failover.

Incoming and outgoing load balancing combines with multi-WAN and policy-based routing to allow multiple Internet connections to be load-balanced for outgoing traffic, or for inbound traffic from the Internet, it allows for load balancing between multiple servers (for example, a Web server farm). Some of the deployments in production today were illustrated and discussed.

At the end, we logged into a production firewall cluster and showed how this functionality is configured and works in a real-world installation.

Overall, the feedback we received was mostly positive, but in hindsight we tried to pack entirely too much into the allotted time frame. We also assumed that those attending a presentation on some of the advanced pfSense functionality would know about the basic functionality, which was mostly correct, but there were a decent number of people who weren't familiar with the project at all. Because of time constraints, we could only give a high-level overview of the previously mentioned functionality, and we couldn't leave users with specific information on how to configure the various deployments discussed. In hindsight this presentation would have been much more useful to our attendees if it were one of the longer tutorials rather than a one-hour presentation. In the future, we'll need to watch our scope or go for a longer session.

■ ***UTORvpn: A Cross-Platform OpenSource SSL VPN Implementation***

Russell Sutherland

Summarized by Chris Buechler (cbuechler@gmail.com)

This session was presented by Russell Sutherland, a network engineer at the University of Toronto. He began by going over the various common types of VPN implementations in production environments today. The problem with most VPN solutions in wide deployment today is cost or lack of cross-platform support. The University of Toronto needed a VPN solution that worked with numerous platforms and would scale to thousands of users, but didn't cost a fortune. Enter OpenVPN.

OpenVPN is an open source SSL VPN solution that has an open source client available on numerous operating systems, including Windows 2000/XP and newer, FreeBSD, NetBSD, OpenBSD, Mac OS X, Linux, and Solaris. As such, it met their requirements for cross-platform compatibility. Its authentication and authorization capabilities also were able to tie into the university's existing Kerberos authentication and LDAP authorization systems.

Russell logged into the university's Web interface where users can sign up for OpenVPN access, to show how they have automated the build of the Windows installer on the FreeBSD server using NSIS, so each user has a customized Windows installer available with the appropriate certificates and configuration built in. He also showed the management interface and the type of reporting and statistics they gather from the log files, and how they manage the certificates, all via a custom-written Web interface.

I'm already a happy OpenVPN user, but this gave me some ideas on how to get more out of it. Russell did an excellent job of introducing people to OpenVPN and showing how it can be used in large deployments.

Announcement and Call for Papers **USENIX**

6th USENIX Conference on File and Storage Technologies (FAST '08)

USENIX, the Advanced Computing Systems Association, in cooperation with ACM SIGOPS, IEEE Mass Storage Systems Technical Committee (MSSTC), and IEEE TCOS

http://www.usenix.org/fast08

February 26–29, 2008 **San Jose, CA, USA**

Important Dates

Paper submissions due: *September 12, 2007, 9:00 p.m. EDT (firm deadline)*
Notification of acceptance: *November 1, 2007*
Final papers due: *January 8, 2008*
Work-in-Progress Reports/Poster Session proposals due: *January 16, 2008*

Conference Organizers

Program Chairs
Mary Baker, *Hewlett-Packard Labs*
Erik Riedel, *Seagate Research*

Program Committee
Andrea C. Arpaci-Dusseau, *University of Wisconsin, Madison*
Randal Burns, *Johns Hopkins University*
Howard Gobioff, *Google*
Christos Karamanolis, *VMware*
Kim Keeton, *Hewlett-Packard Labs*
Geoff Kuenning, *Harvey Mudd College*
Darrell Long, *University of California, Santa Cruz*
Petros Maniatis, *Intel Research Berkeley*
Robert Morris, *Massachusetts Institute of Technology*
Brian Noble, *University of Michigan*
Alma Riska, *Seagate Research*
Antony Rowstron, *Microsoft Research, UK*
Jiri Schindler, *Network Appliance*
Margo Seltzer, *Harvard University*
Doug Terry, *Microsoft*
Theodore Ts'o, *IBM*
Andrew Warfield, *University of British Columbia*
Ric Wheeler, *EMC*
Theodore Wong, *IBM Research*
Erez Zadok, *Stony Brook University*

Steering Committee
Andrea C. Arpaci-Dusseau, *University of Wisconsin, Madison*
Remzi H. Arpaci-Dusseau, *University of Wisconsin, Madison*
Jeff Chase, *Duke University*
Greg Ganger, *Carnegie Mellon University*
Garth Gibson, *Carnegie Mellon University and Panasas*
Peter Honeyman, *CITI, University of Michigan, Ann Arbor*
Merritt Jones, *MITRE Corporation*
Darrell Long, *University of California, Santa Cruz*
Jai Menon, *IBM Research*
Margo Seltzer, *Harvard University*
Chandu Thekkah, *Microsoft Research*
John Wilkes, *Hewlett-Packard Labs*
Ellie Young, *USENIX Association*

Overview

The 6th USENIX Conference on File and Storage Technologies (FAST '08) brings together storage system researchers and practitioners to explore new directions in the design, implementation, evaluation, and deployment of storage systems. The conference will consist of two and a half days of technical presentations, including refereed papers, Work-in-Progress reports, and a poster session.

Topics

Topics of interest include but are not limited to:

- Archival storage systems
- Caching, replication, and consistency
- Database storage issues
- Distributed I/O (wide-area, grid, peer-to-peer)
- Empirical evaluation of storage systems
- Experience with deployed systems
- Manageability
- Mobile and personal storage

- Parallel I/O
- Performance
- Reliability, availability, disaster tolerance
- Scalability
- Security
- Storage networking
- Virtualization

Deadline and Submission Instructions

Submissions will be made electronically via a Web form, which will be available on the FAST '08 Call for Papers Web site, http://www.usenix.org/fast08/cfp. The Web form asks for contact information for the paper and allows for the submission of your full paper file in PDF format.

Submissions must be full papers (no extended abstracts) and must be no longer than thirteen (13) pages plus as many additional pages as are needed for references (e.g., your paper can be 16 total pages, as long as the last three or more are the bibliography). Your paper should be typeset in two-column format in 10 point type on 12 point (single-spaced) leading, with the text block being no more than 6.5" wide by 9" deep. Submissions longer than this will not be reviewed.

Authors must not be identified in the submissions, either explicitly or by implication (e.g., through the references or acknowledgments). Blind reviewing of full papers will be done by the program committee, assisted by outside referees. Accepted papers will be shepherded through an editorial review process by a member of the program committee.

Simultaneous submission of the same work to multiple venues, submission of previously published work, and plagiarism constitute dishonesty or fraud. USENIX, like other scientific and technical conferences and journals, prohibits these practices and may, on the recommendation of a program chair, take action against authors who have committed them. In some cases, program committees may share information about submitted papers with other conference chairs and journal editors to ensure the integrity of papers under consideration. If a violation of these principles is found, sanctions may include, but are not limited to, barring the authors from submitting to or participating in USENIX conferences for a set period, contacting the authors' institutions, and publicizing the details of the case.

Authors uncertain whether their submission meets USENIX's guidelines should contact the program chairs, fast08chairs@usenix.org, or the USENIX office, submissionspolicy@usenix.org.

Accepted material may not be subsequently published in other conferences or journals for one year from the date of acceptance by USENIX. Papers accompanied by nondisclosure agreement forms will not be read or reviewed. All submissions will be held in confidence prior to publication of the technical program, both as a matter of policy and in accordance with the U.S. Copyright Act of 1976. Submissions violating these rules or the formatting guidelines will not be considered for publication.

One author per paper will receive a registration discount of $200. USENIX will offer a complimentary registration upon request.

Best Paper Awards

Awards will be given for the best paper(s) at the conference.

Work-in-Progress Reports and Poster Session

The FAST technical sessions will include slots for Work-in-Progress reports, preliminary results, "outrageous" opinion statements, and a poster session. We are particularly interested in presentations of student work. Please send WiP submissions to fast08wips@usenix.org.

Birds-of-a-Feather Sessions

Birds-of-a-Feather sessions (BoFs) are informal gatherings organized by attendees interested in a particular topic. BoFs will be held in the evening. BoFs may be scheduled in advance by emailing the Conference Department at bofs@usenix.org. BoFs may also be scheduled at the conference.

Registration Materials

Complete program and registration information will be available in November 2007 on the conference Web site. The information will be in both HTML and a printable PDF file. If you would like to receive the latest USENIX conference information, please join our mailing list: http://www.usenix.org/about/mailing.html.

Announcement and Call for Papers **USENIX**

5th USENIX Symposium on Networked Systems Design & Implementation (NSDI '08)

Sponsored by USENIX in cooperation with ACM SIGCOMM and ACM SIGOPS

http://www.usenix.org/nsdi08

April 16–18, 2008 **San Francisco, CA, USA**

Important Dates

Paper titles and abstracts due: *October 2, 2007*
Complete paper submissions due: *October 9, 2007*
Notification of acceptance: *December 21, 2007*
Papers due for shepherding: *January 25, 2008*
Final papers due: *February 19, 2008*

Conference Organizers

Program Chairs

Jon Crowcroft, *University of Cambridge*
Mike Dahlin, *University of Texas at Austin*

Program Committee

Paul Barham, *Microsoft Research*
Ken Birman, *Cornell University*
Miguel Castro, *Microsoft Research*
Jeff Chase, *Duke University*
Steve Gribble, *University of Washington*
Matthias Grossglauser, *EPFL*
Krishna Gummadi, *Max Planck Institute for Software Systems*
Steven Hand, *University of Cambridge*
Brad Karp, *University College, London*
Dina Katabi, *Massachusetts Institute of Technology*
Eddie Kohler, *University of California, Los Angeles*
Sue Moon, *KAIST*
Robert Morris, *Massachusetts Institute of Technology*
Sylvia Ratnasamy, *Intel Research*
Luigi Rizzo, *ICIR*
Timothy Roscoe, *ETH Zürich*
Srinivasan Seshan, *Carnegie Mellon University*
Emin Gün Sirer, *Cornell University*
Amin Vahdat, *University of California, San Diego*
Arun Venkataramani, *University of Massachusetts Amherst*

Steering Committee

Thomas Anderson, *University of Washington*
Mike Jones, *Microsoft Research*
Greg Minshall
Robert Morris, *Massachusetts Institute of Technology*
Mike Schroeder, *Microsoft Research*
Amin Vahdat, *University of California, San Diego*
Ellie Young, *USENIX*

Overview

NSDI focuses on the design principles and practical evaluation of large-scale networked and distributed systems. Systems as diverse as Internet routing, peer-to-peer and overlay networks, sensor networks, Web-based systems, and measurement infrastructures share a set of common challenges. Progress in any of these areas requires a deep understanding of how researchers are addressing the challenges of large-scale systems in other contexts. Our goal is to bring together researchers from across the networking and systems community—including communication, distributed systems, and operating systems—to foster a broad approach to addressing our common research challenges.

Topics

NSDI will provide a high-quality, single-track forum for presenting new results and discussing ideas that overlap these disciplines. We seek a broad variety of work that furthers the knowledge and understanding of the networked systems community as a whole, continues a significant research dialog, or pushes the architectural boundaries of large-scale network services. We solicit papers describing original and previously unpublished research. Specific topics of interest include but are not limited to:

- Novel architectures for communications systems
- Mobility and wireless system architecture challenges
- Sensor networking and other energy-constrained systems
- Novel operating system support for networked systems

- Virtualization and resource management for networked systems
- Highly available and reliable networked systems
- Security and resilience of networked systems
- Overlays and peer-to-peer systems
- Distributed storage, caching, and query processing
- Federated, autonomous, and self-configuring networked systems
- Large-scale networked systems testbeds, design, and evaluation
- Network measurements, workload, and topology characterization
- Managing, debugging, and diagnosing problems in networked systems
- Practical protocols and algorithms for networked systems
- Application experiences based on networked systems

What to Submit

Submissions must be full papers, at most 14 single-spaced 8.5" x 11" pages, including figures, tables, and references, two-column format, using 10-point type on 12-point (single-spaced) leading, with a maximum text-block of 6.5" wide x 9" deep. Papers that do not meet the requirements on size and format will not be reviewed. Submissions will be judged on originality, significance, interest, clarity, relevance, and correctness.

NSDI is single-blind, meaning that authors should include their names on their paper submissions and do not need to obscure references to their existing work.

Authors must submit their paper's title and abstract by October 2, 2007, and the corresponding full paper is due by October 9, 2007. All papers must be submitted via the Web form, which will be available on the Call for Papers Web site, http://www.usenix.org/nsdi08/cfp. Accepted papers may be shepherded through an editorial review process by a member of the Program Committee. Based on initial feedback from the Program Committee, authors of shepherded papers will submit an editorial revision of their paper to their Program Committee shepherd by January 25, 2008. The shepherd will review the paper and give the author additional comments. All authors (shepherded or not) will produce a final, printable PDF and the equivalent HTML by February 19, 2008, for the conference Proceedings.

Simultaneous submission of the same work to multiple venues, submission of previously published work, and plagiarism constitute dishonesty or fraud. USENIX, like other scientific and technical conferences and journals, prohibits these practices and may, on the recommendation of a program chair, take action against authors who have committed them. In some cases, program committees may share information about submitted papers with other conference chairs and journal editors to ensure the integrity of papers under consideration.

Previous publication at a workshop is acceptable as long as the NSDI submission includes substantial new material. For instance, submitting a paper that provides a full evaluation of an idea that was previously sketched in a 5-page position paper is acceptable. Authors of such papers should cite the prior workshop paper and clearly state the submission's contribution relative to the prior workshop publication.

Authors uncertain whether their submission meets USENIX's guidelines should contact the Program Chairs, nsdi08chairs@usenix.org, or the USENIX office, submissionspolicy@usenix.org.

Best Paper Awards

Awards will be given for the best paper and the best paper for which a student is the lead author.

Birds-of-a-Feather Sessions

Birds-of-a-Feather sessions (BoFs) are informal gatherings organized by attendees interested in a particular topic. BoFs will be held in the evening. BoFs may be scheduled in advance by emailing the USENIX Conference Department at bofs@usenix.org. BoFs may also be scheduled at the conference.

Registration Materials

Complete program and registration information will be available in January 2008 on the conference Web site. The information will be in both HTML and a printable PDF file. If you would like to receive the latest USENIX conference information, please join our mailing list at http://www.usenix.org/about/mailing.html.

;login:

USENIX Association
2560 Ninth Street, Suite 215
Berkeley, CA 94710

POSTMASTER
Send Address Changes to *;login:*
2560 Ninth Street, Suite 215
Berkeley, CA 94710

PERIODICALS POSTAGE
PAID
AT BERKELEY, CALIFORNIA
AND ADDITIONAL OFFICES