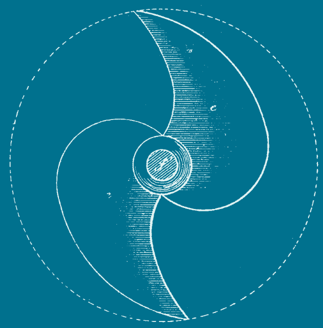
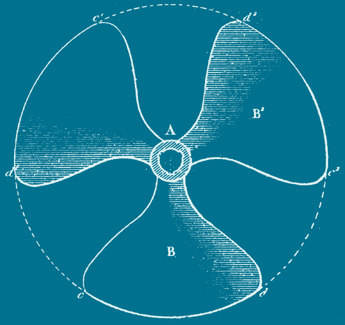
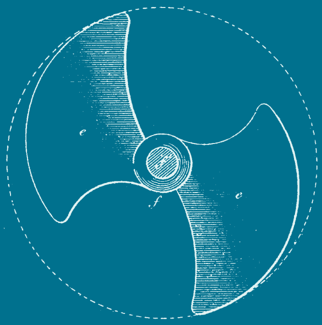
AUGUST 2006 VOLUME 31 NUMBER 4

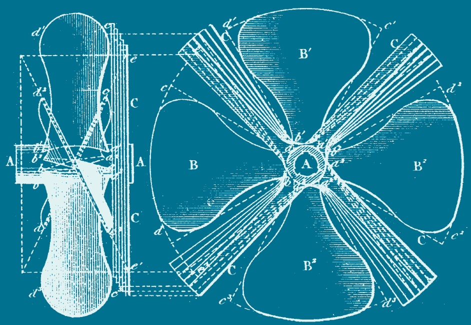
;login:

THE USENIX MAGAZINE



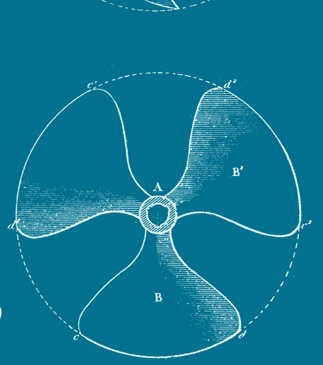






USENIX
The Advanced Computing Systems Association

USENIX Upcoming Events



3rd Workshop on Real, Large Distributed Systems (WORLDS '06)

NOVEMBER 5, 2006, SEATTLE, WA, USA
http://www.usenix.org/worlds06

7th USENIX Symposium on Operating Systems Design and Implementation (OSDI '06)

Sponsored by USENIX, in cooperation with ACM SIGOPS

NOVEMBER 6–8, 2006, SEATTLE, WA, USA
http://www.usenix.org/osdi06

Second Workshop on Hot Topics in System Dependability (HotDep '06)

NOVEMBER 8, 2006, SEATTLE, WA, USA
http://www.usenix.org/hotdep06

ACM/IFIP/USENIX 7th International Middleware Conference

NOV. 27–DEC. 1, 2006, MELBOURNE, AUSTRALIA
http://2006.middleware-conference.org

20th Large Installation System Administration Conference (LISA '06)

DECEMBER 3–8, 2006, WASHINGTON, D.C., USA
http://www.usenix.org/lisa06

5th USENIX Conference on File and Storage Technologies (FAST '07)

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

FEBRUARY 13–16, 2007, SAN JOSE, CA, USA
http://www.usenix.org/fast07
Paper submissions due: September 6, 2006

4th Symposium on Networked Systems Design and Implementation (NSDI '07)

Sponsored by USENIX, in cooperation with ACM SIGCOMM and ACM SIGOPS

APRIL 11–13, 2007, CAMBRIDGE, MA, USA
http://www.usenix.org/nsdi07

11th Workshop on Hot Topics in Operating Systems (HotOS XI)

Sponsored by USENIX in cooperation with the IEEE Technical Committee on Operating Systems (TCOS)

MAY 7–9, 2007, SAN DIEGO, CA, USA
http://www.usenix.org/hotos07
Paper submissions due: January 4, 2007

2007 USENIX Annual Technical Conference

JUNE 17–22, 2007, SANTA CLARA, CA, USA
http://www.usenix.org/usenix07

16th USENIX Security Symposium

AUGUST 6–10, 2007, BOSTON, MA, USA

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

contents

;login:

VOL. 31, #4, AUGUST 2006

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.

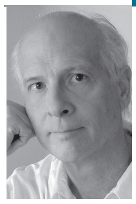
$85 of each member's annual dues is for an annual subscription to *;login:*. Subscriptions for nonmembers are $115 per year.

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

ONE OF THE ADVANTAGES OF BEING a "greybeard" is that I get to go on about just how hard things used to be. You know, like how I had to walk to school barefoot, in the snow, and uphill in both directions. That is a bit of an exaggeration, of course. I did have shoes. But I did also build my own first computer, and I modified the motherboard in my second as well.

I worked for a while at NorthStar Computers, a Berkeley startup whose first product was a disk controller for floppy drives. By the time I arrived in 1979, the company had been selling Horizon-2s both in kit and assembled forms. I got one of the last four kits and went to work.

The Horizon-2 [1] used a 6-MHz Z80, eight-bit processor on the motherboard, along with two serial ports, one parallel port, and twelve connectors on an S-100 bus. That "100" in the bus name means 100 pins per slot, so just soldering in the connectors meant 1200 careful solder joints. Disk storage was in the form of proprietary 5.25-inch floppy disks that held 90 kilobytes each, and memory maxed out at 64 kilobytes—although you couldn't use the last 8 kilobytes, as the ROM, containing the boot code and floppy disk drivers, overlapped 1 kilobyte of this memory space.

Later advances included double-sided drives with a whopping 360 kilobytes per disk, then a five-megabyte hard drive (which all by itself cost more than the entire system).

Instead of using NorthStar DOS, I would use CP/M, created by Digital Research, a company whose name might have become as famous as Microsoft except for some executive decisions made while dealing with IBM. I eventually added a surplus 20-MB hard drive, patched a driver into CP/M, and sold the souped-up Horizon-2 to a collective [2] which ran a grocery delivery business on this system for several years.

Fast Forward

Besides the observation that this primitive system, and ones like it, were used to run businesses, some other principles found in the Horizon-2 survive to this day. For example, every memory cell in DRAM [3] must be refreshed every 64 ms or less by reading in a row of memory, then rewriting it. Memory refresh occurs in parallel with the real work done by memory, supplying data to the processor. In the Horizon and other S-100 bus designs, CPUs accessed memory synchronously, controlling all bus signals at the awesome speeds of up to 10 MHz.

I continued to use systems with CPUs running at less than 10 MHz through most of the 1980s. Compiling was slow but not impossible, because hard disks sped up disk-bound operations tremendously over floppy disks. But even at these slow bus speeds, access to memory was not instantaneous.

Before a row of memory can be read, the address that represents that row must be decoded. The decoding takes time, generally several bus-clock cycles (a delay called RAS; see [4] and [5]). After address decoding, the contents of memory are copied from sense registers to where they can be copied to the bus. The rate at which the actual copying of memory takes place depends on the type of memory, the width of the bus, and the frequency of the bus.

Over the years, CPU and DRAM chip developers have come up with schemes for improving the transfer rates between the CPU and DRAM. Bus frequencies have gotten faster, buses have gotten wider, and DRAMs have also gotten faster—but nowhere near as fast as CPUs have become. For example, a 3.2-GHz Xeon processor is running four times faster than its memory bus when various tricks, such as multiple reads per bus cycle, are taken into account.

Caches were designed to hide this discrepancy as much as possible. Level 1 cache resides on the processor die itself and runs at the same clock rate as the processor. You might find yourself wondering why CPU designers don't use more Level 1 cache than we typically see, but the problem is that chip designers must make several tradeoffs. Cache uses Static RAM (SRAM), which is much faster than DRAM, does not require refreshing, but also requires many more transistors to implement a single bit. More transistors means more die real estate, as well as more energy to support. Also, the process of address decoding is an issue for cache just as it is for DRAM, and, to speed up address decoding, various schemes that can quickly determine if a cache line holds a particular address are used.

If there is a Level 1 cache miss, perhaps the appropriate line of memory will be found in Level 2 cache, which exists off the CPU die but takes longer to access. Since this is also SRAM, it too is faster than DRAM, and Level 2 cache generally occupies a separate chip (or chips), so there can be more of it. But these memory accesses are slower than those for Level 1, forcing the CPU to wait for memory to become available. If the desired addresses in memory are not found in any cache, then DRAM must be accessed. And DRAM introduces both the RAS setup time and data transfer at the fastest rate that the memory bus supports. During this time, the processor may become idle as pipelines empty, and will continue idle until the wait state induced by slower memory ends.

Even 6MHz Z80 processors suffered wait states, and this problem has only gotten worse over time. Chip designers have worked around this problem as best they can, and we can see this in system benchmarks. That is, faster processors generally mean better benchmarks, but these benchmarks do not scale linearly. In other words, increasing the processor speeds by 20% do not result in real-world benchmarks increasing by 20%. In fact, the relation between processor speeds and benchmark performance varies all over the place, something you will notice if you read motherboard reviews. And memory is a major source of this variation. Other sources include the other buses and I/O devices.

Steve Johnson, during his Guru session at Annual Tech '06 in Boston, demonstrated that how you store and access elements in an array can make enormous differences in CPU-bound performance. Processing data elements stored sequentially in two different arrays was many times faster than accessing data elements that appeared separated by just a small number of bytes. Johnson's talk stirred up a lot of discussion, because it makes the points about caches, bus speeds, and memory concrete, as well as uncovering some interesting performance surprises.

The Fix

Fixing these performance bottlenecks poses a big problem for the leading CPU vendors, Intel and AMD. AMD has taken the lead currently through the use of faster bus designs. But is this the only way to go? Sun Microsystems engineers have bet that there is another way that will work well for some, although not all, applications.

I had heard about a new line of Sparc CPUs that include multiple cores, each able to quickly (one cycle) switch among four threads. The multithreading capability of Sparc chips is old news, but having up to eight cores per chip was something new. I had learned about the new Sun designs at BSDCan (see summary in this issue) and read what I could find about the design before I came to Boston. Then, the first night there, Richard McDougall of Sun walked up with a T2000, a 2U rackmount system containing the new CPU. McDougall and Jim Maury used the T2000 during their Solaris performance and debugging class and set it up for use during Peter Galvin's Solaris classes, so people had plenty of time to "experience" the new Sun design.

What the Sun engineers said was that when running a highly threaded application such as Apache or Oracle, the eight cores, each running at 1.2 GHz, can operate without wait states as much as 80% of the time. This figure really had me wondering about the percentage of wait states for dual Xeon processors running at three times that clock rate. Whereas the rackmounted T2000 had the usual noisy-as-a-vacuum-cleaner fans, the processor itself has a short heat sink, with no fan on it, and it remained cool to the touch—a feature that Sun hopes will make this new twist on old Sparc technology popular where heat is an issue.

Of course, if your application is single-threaded and includes lots of floating-point calculations, then this design will not work well for you. (There is a single floating-point processor for all cores in the current multi-core Sparc, something that will change in the future.) However, since I have been writing and dreaming about how multi-core systems could be a much better design for future operating systems and for security, I was pretty excited about my face-to-face encounter with the T2000.

I have often written in my column that games, not business software, have been the strongest driving force behind the adoption of Windows and the design of faster PC hardware. If you're not playing first-person shooters, you don't need multi-gigahertz processors and fan-cooled video cards. I used word-processing software (WordStar) that worked quite well on a 10 Mhz processor for many years, writing code, documentation, and even books without using a system that could double as a space heater or high-end graphics workstation. There certainly are applications that do make excellent use of fast hardware today, such as Web browsers, Google Earth, and the many applications of cluster computing we hear about.

The point I'd like to leave with you is that CPU speed is not the current barrier to performance that it might seem to be. System performance relies on the entire system and will be limited by the slowest component or the slowest pathway to that component.

The Lineup

In the June issue, Kurt Chan wrote about disk types and architectures from the perspective of a NAS vendor. In this issue, Mark Uris writes about his experiences building a high-performance, multi-terabyte storage system at NCAR. The design of DataMonster uses nearline enterprise SATA drives, a design decision that at first glance appears to conflict with what Chan wrote. But I believe that the applications found at NCAR mainly result in large file transfers, not OLTP, which Chan cited as the main reason for building with enterprise drives.

Mark Burgess has written the first in what I hope will be a series of articles about configuration management. Darrell Fuhriman has written an article explaining identity management from the perspective of a sysadmin, and Andy Seely writes about administering systems used to support DoD operations and how that differs from normal sysadmin.

In the Security section, Eric Sorenson discusses creating your own network black hole as a technique for monitoring your network. Pete Herzog has written about Dru Lavigne's research into TCP/IP services, uncovering the history behind many of the obscure services you may have noticed.

Several authors of papers that appeared in the NSDI symposium (see summaries in this issue) have written articles about their projects. KyoungSoo Park and Vivek Pai write about CoBlitz, a Content Distribution Network (CDN) that works with unmodified Web browsers and servers; it performed faster than BitTorrent in real-world tests. Better yet, you can start using CoBlitz today. Ryan Peterson, Venugopalan Ramasubramanian, and Emin Gün Sirer write about Corona, a practical publish-subscribe system for the Web that solves the problems posed by RSS and current publish-subscribe systems.

In the Columns section, David Blank-Edelman begins a two-part series about the use of `tie()` to associate variables with databases, a practice some people (but not David) consider controversial. Robert Haskins discusses traffic-shaping tools that can be used both by ISPs and by any organization with a lot of Internet-bound traffic. Heison Chak explains the differences between soft and hard phones that are using VoIP. Finally, there is an excellent assortment of book reviews.

REFERENCES

[1] The NorthStar Horizon: http://www.old-computers.com/museum/computer.asp?c=50.

[2] The Purple Rose Collective: http://fic.ic.org/video/purpleroseinfo.php.

[3] Explanation of DRAM: http://en.wikipedia.org/wiki/DRAM.

[4] A clear explanation of different frontside bus types and what overclocking means: http://www.directron.com/fsbguide.html.

[5] Guide to understanding DRAM terms: http://www.dewassoc.com/performance/memory/memory_speeds.htm.

writing for *;login:*

Writing is not easy for most of us. Having your writing rejected, for any reason, is no fun at all. The way to get your articles published in *;login:*, with the least effort on your part and on the part of the staff of *;login:*, is to submit a proposal first.

PROPOSALS

In the world of publishing, writing a proposal is nothing new. If you plan on writing a book, you need to write one chapter, a proposed table of contents, and the proposal itself and send the package to a book publisher. Writing the entire book first is asking for rejection, unless you are a well-known, popular writer.

;login: proposals are not like paper submission abstracts. We are not asking you to write a draft of the article as the proposal, but instead to describe the article you wish to write. There are some elements that you will want to include in any proposal:

- What's the topic of the article?
- What type of article is it (case study, tutorial, editorial, mini-paper, etc.)?
- Who is the intended audience (syadmins, programmers, security wonks, network admins, etc.)?
- Why does this article need to be read?
- What, if any, non-text elements (illustrations, code, diagrams, etc.) will be included?
- What is the approximate length of the article?

Start out by answering each of those six questions. In answering the question about length, bear in mind that a page in *;login:* is about 600 words. It is unusual for us to publish a one-page article or one over eight pages in length, but it can happen, and it will, if your article deserves it. We suggest, however, that you try to keep your article between two and five pages, as this matches the attention span of many people.

The answer to the question about why the article needs to be read is the place to wax enthusiastic. We do not want marketing, but your most eloquent explanation of why this article is important to the readership of *;login:*, which is also the membership of USENIX.

UNACCEPTABLE ARTICLES

;login: will not publish certain articles. These include but are not limited to:

- Previously published articles. A piece that has appeared on your own Web server but not been posted to USENET or slashdot is not considered to have been published.
- Marketing pieces of any type. We don't accept articles about products. "Marketing" does not include being enthusiastic about a new tool or software that you can download for free, and you are encouraged to write case studies of hardware or software that you helped install and configure, as long as you are not affiliated with or paid by the company you are writing about.
- Personal attacks

FORMAT

The initial reading of your article will be done by people using UNIX systems. Later phases involve Macs, but please send us text/plain formatted documents for the proposal. Send proposals to login@usenix.org.

DEADLINES

For our publishing deadlines, including the time you can expect to be asked to read proofs of your article, see the online schedule at http://www.usenix.org/publications/login/sched.html.

COPYRIGHT

You own the copyright to your work and grant USENIX permission to publish it in *;login:* and on the Web. USENIX owns the copyright on the collection that is each issue of *;login:*. You have control over who may reprint your text; financial negotiations are a private matter between you and any reprinter.

FOCUS ISSUES

In the past, there has been only one focus issue per year, the December Security edition. In the future, each issue may have one or more suggested focuses, tied either to events that will happen soon after *;login:* has been delivered or events that are summarized in that edition.

MARK URIS

DataMonster: building a cost-effective, high-speed, heterogeneous shared file system

Mark Uris has been doing system administration within the Scientific Computing Division of the National Center for Atmospheric Research for more than eighteen years. He is involved in the design and administration of infrastructure services for the Web, email, DNS, and security for NCAR.

uris@ucar.edu

IN THIS ARTICLE I DESCRIBE HOW WE built DataMonster, a high-speed heterogeneous shared file system. The project was built for the Scientific Computing Division (SCD) located at the National Center for Atmospheric Research (NCAR), where we support both supercomputers and very large datasets. The datasets are located on a multi-petabyte archival system made up of StorageTek's silos housed within SCD. The most popular datasets needed to be more readily accessible to NCAR's sixty-six-member university users conducting atmospheric and earth science research. DataMonster would provide the means to accomplish this. Over the course of this project, DataMonster's nature and composition changed, from a high-priced racing car to a cost-effective commuter car with race-car performance, by juggling choices for hardware and software.

The project named DataMonster called for a vast array of storage devices and heterogeneous servers interconnected by Fibre Channel (FC) switches and cables. It was needed to:

- Allow SGIs, Suns, IBMs, and Linux servers to share large datasets
- Eliminate supercomputers from countless accesses to the mass storage archives by having datasets online
- Supply the Visualization Lab with finer-grained datasets for visualization simulations
- Give users the ability to download datasets throughout the world
- Eliminate long wait times and high-priced processors for receiving data by using a hybrid of SATA- and FC-based storage units tailored for mid- to high-speed data accesses

DataMonster would outperform technologies such as NFS by replacing 25–30 megabytes/s maximum transfer rates with speeds in excess of 60–70 megabytes/s for writes and over 100 megabytes/s for reads. The initial amount of data targeted called for 16 terabytes, and estimates up to 100 terabytes were projected within a few years.

Experience with Storage Devices

We never lacked for benchmarks on storage device performance. A number of storage systems were herded into the NCAR data center for direct I/O testing against an SGI Origin 2000 including an Apple Xserve RAID, an Nexsan ATABoy, and a StorageTek ATA/SATA-based storage unit plus Sun and SGI offerings. SATA currently ran at a clock speed of 150 megabytes/s and ATA, also known as IDE, ran at 133 megabytes/s as outlined by their respective standards committees. Although SATA had a higher speed rating, it actually performed at comparable speeds to ATA. There was no rush among vendors to replace low-cost ATA disks with SATA ones. SATA-II was to be available in the immediate future and would represent a major speed breakthrough at 300 megabytes/s. The SATA standard committee had developed a ten-year road map where SATA would reach 600 megabytes/s and replace ATA technology. For more information on SATA see, in the June 2006 issue of *;login:*, Kurt Chan's article on comparing disk performance.

Although we hoped to discover a diamond in the rough, we found little variation from one system's performance to another's. For an ATA/SATA-based storage unit with FC, connection speeds of 50–70 megabytes/s were observed for writes and over 100 megabytes/s for reads. Units housing FC disk drives would double these speeds. The mainline vendors offered larger chassis capable of holding terabytes of disk, whereas vendors such as Apple had only a 3U chassis. Satisfied that we had a foothold on the storage subsystem of DataMonster, we turned our attention to finding a heterogeneous shared file system. This is where the journey really began.

Finding a Heterogeneous Shared File System

We had previous experience using SGI's CXFS file system in our Visualization Lab, but this was only on SGI servers. We know that CXFS ran on a number of different platforms, but we didn't know how it actually performed in a heterogeneous server environment.

We asked SGI to provide us with a list of reference sites running a mix of Sun, SGI, AIX, and Linux systems. Although the references checked out, using CXFS carried a hefty price tag. SGI controlled the entire hardware and software pipeline, from storage devices to switches. There is nothing like sticker shock to bring even the most lofty plans back down to earth. Estimates based on different storage configurations ranged from $18 to $20 per gigabyte for 20 terabytes of FC-based storage. This rate would decrease when SGI introduced SATA-based storage devices.

We started fishing around for alternative solutions. Sun had been doing a lot of work on their QFS shared file system. We studied it to see how it compared to SGI's CXFS. Sun didn't provide shared file service to all the diverse platforms in our environment. This was a show-stopper. Sun, like SGI, required use of their hardware (e.g., storage, switches) to use the product. The Sun quote for FC-based storage ran around $15 per gigabyte for 20 terabytes. Other solutions on the market were directed at Linux server configurations only, such as Cluster File System (CFS) Lustre, RedHat GPS, Sistana GFS, and the IBM GPFS just released for Linux servers. We appeared to be headed for a high-priced solution or no solution at all.

A New Solution Emerges

It was by sheer luck that I discovered in a trade journal Apple's plan to introduce their Xsan system in the immediate future. Apple touted their shared file system as one-fifth the cost of the traditional offerings. Apple's plan involved using their Apple Xserve RAID storage devices to drive the storage cost down. Based on just storage, the average cost would be around $3.25 per gigabyte for a ATA-based system with an FC interface. This didn't factor in the cost of server software, client software licensing, HBA cards for clients, cables, and a switch. But it was well below the cost of other solutions on the market.

The Xsan would be managed on a Apple system running Mac OS X. As I dug into the details of Apple's shared file system I found that it was a rebranded ADIC StorNext file system (SNFS). Unlike other vendors, ADIC offered only the server and client software needed to create a shared file system, leaving selection of storage devices, switches, metadata server hardware, and operating systems up to the implementer. The metadata management software had been ported to a number of different operating systems. Twenty terabytes of storage suddenly seemed financially obtainable, but who was ADIC?

The ADIC SNFS

I hadn't dealt with ADIC before, so I made contact with the local sales rep. ADIC had traditionally done large-scale data management systems and had acquired Mountain Gate and their CentraVision file system (CVFS). They had set up a team of fifteen engineers in the Denver area, only an hour's drive from our site, to work on CentraVision, renamed StorNext File System (SNFS). We decided to set up a testbed to evaluate ADIC's SNFS performance in a simulated work environment where different vendor operating systems were interacting. This should reveal any problems before installing it on production servers. ADIC agreed to provide on-site installation and back-end support for setting up this trial evaluation of their product. The ADIC engineering staff would work through any problems we encountered.

Assembly of Testbed Components

To set up the ADIC SNFS, a Storage Area Network (SAN) had to be built. The SAN is the basic building block on which any high-speed file system resides. In a SAN, connections between systems and storage are made through one or more FC switches. The systems, storage, adapters, cables, and switches make up a SAN; the cables and switches are referred to as FC fabric. The SAN component was built with the following components:

- Apple Xserve RAID, with 1 terabyte of data space
- QLogic FC switch, with eight ports on the switch supporting 1GB FC
- Netgear four-port 100BaseT Ethernet switch
- Sun E450 server, ADIC SNFS client
- Sun 280 server, serving as ADIC SNFS metadata server and an SNFS client

The initial layout and testing dealt with making sure there was connectivity among the different system components (see Figure 1). What appeared to be a straightforward configuration turned out to require a lot of tweaking with cables and switch settings before the servers were able to communicate with the storage device. The Apple Xserve RAID came with an Apple FC HBA card that we used in one of the Suns. This caused a number of problems. The device driver for the card was set up for direct communication between the Apple storage unit and the server in which the card resided. When the FC line was run into the switch, it would dominate the

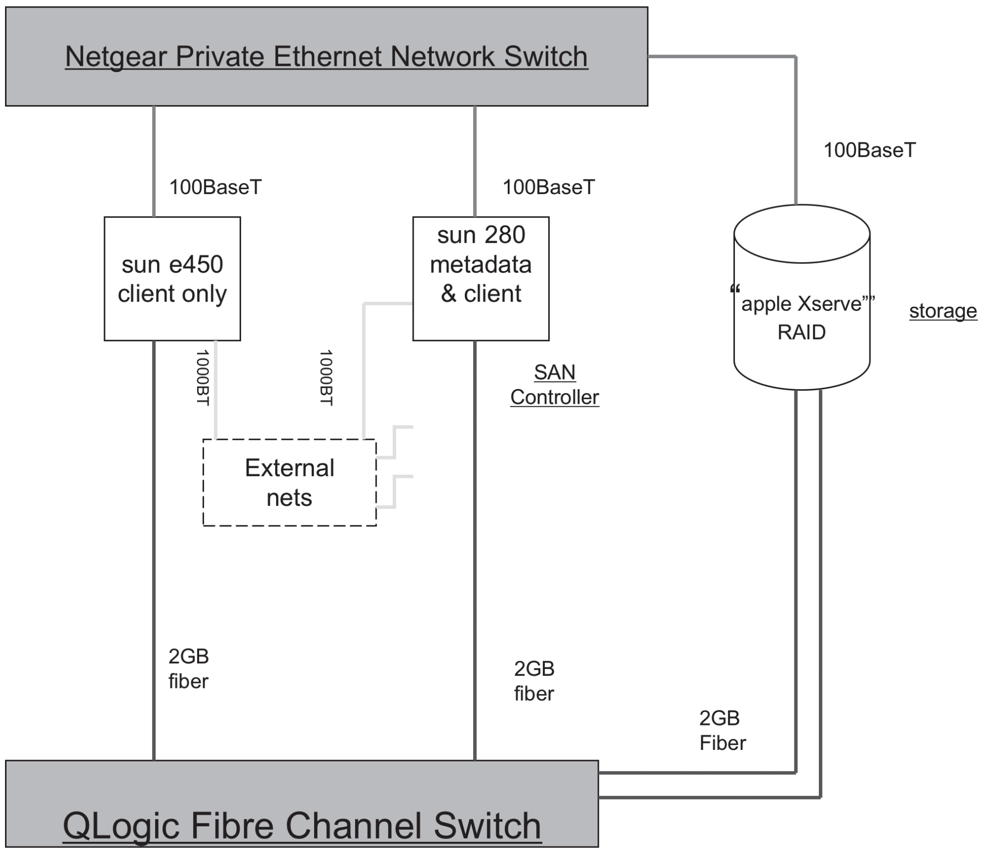


FIGURE 1: TESTBED.

communication channel and the other Sun server became unable to communicate with the storage unit. The problem was eventually overcome by buying another PCI FC HBA card from Sun. Once this was installed and run into the switch, both servers were able to communicate with the Xserve storage unit.

To secure the network between the servers and the storage device, a private Ethernet network was set up. This network was reachable only by logging in, using a security token, to the Sun 280, which housed the ADIC metadata server software. A Netgear switch on this private network allows the testbed hosts and storage device to communicate over 100BaseT Ethernet connections. It is isolated from the public Internet. The ADIC metaserver receives file requests on this private Ethernet network from the SNFS clients, but the actual file transfers are carried out on the SAN. SNFS requires dedicated space for metadata and journaling of the shared file systems that are created. This space should theoretically reside on a separate storage device in a production environment. But to expedite the installation of the testbed, one of the logical unit numbers (LUNs) on the Apple storage unit was used for this purpose.

In addition to carrying out metadata service, the Sun 280 server would be an SNFS client. The ADIC Web interface was brought up to do the actual file system creation and build. The SNFS configurations and builds were completed without any problems. We then installed the client software on each Sun server. It looked like we were headed into the home stretch. The only thing left was to mount the file systems on the clients, similar to what is done for NFS mounts. But the clients wouldn't mount the SNFS file system. A call was placed to ADIC to assist us in troubleshooting the problem. They were able to replicate the configuration we were running in their lab, but there everything worked without any problems. An ADIC site engineer spent the entire day at our site inspecting the configuration. He couldn't find the problem either. Finally, one of the ADIC engineers asked what version of firmware we were running on the Apple storage unit. We found that we were two revisions behind the system they had in their lab. After we upgraded the firmware, we were up and running.

Performance Results from the Testbed

From our experience with previous tests, we were not really concerned about file transfer speeds since the metadata server is more of a traffic cop, and once the transfer takes place it has speeds almost identical to the storage devices directly connected to a server. We had heard horror stories of metadata servers becoming overloaded and becoming a bottleneck for file system activity. We needed the file system to be able to handle at least ten simultaneous transactions at any given time without impacting performance. We also tested files to determine whether any corruption occurred during system activity. The main standard benchmark suite used was iozone, which can be downloaded from the site www.iozone.org. Other available tools include xdd from the University of Minnesota and bonnie (www.coker.com.au/bonnie++). The initial results of testing were as follows:

- After more than 10 terabytes of data written to and from the file system, no file corruption or system crashes occurred.
- Aggregate sustained write rates were 50 megabytes/s, lower than we expected. We didn't know whether this was limited by hardware or the LUN sizes we created.
- Aggregate sustained read rates were 100 megabytes/s, higher than we expected.
- Over 100 files could be opened per second without degradation of metaserver performance.

Without attempting to optimize data stripe size or further tweaking to increase performance, we found the overall performance and stability of the system to be close to what we needed.

On to a Production Environment

The initial testbed results were reported to the different organizational units within SCD. As soon as possible, the dataset developers wanted SNFS to be set up between the Data Support section's Sun V880 server and the UCAR Community Data Portal's Sun V880 for a number of dataset projects that they were initiating. The amount of shared storage was set at 16 terabytes and was to be increased to approximately 80 terabytes within a few years. The testbed would have to wait.

We had tested and placed a Nexsan ATABoy system into production and focused on its big brother, the ATABeast storage system, which holds 16.8 terabytes. Two factors were involved in our decision. First, we didn't want to stack up a number of smaller-capacity (3.5-terabyte) units, which would mean running two FC connections for each unit into the switch and a more complicated file system layout. ATABeast would require only two connections, as opposed to ten connections for Xerve RAIDs. Second, Apple was staying with ATA technology and would not switch to SATA-II disks in their newer units such as Nexsan. Nexsan ATABeast had a price point of $2.90 per gigabyte. So the first storage unit in DataMonster would be named after a beast. Somehow everything started fitting together. We decided to add it to the testbed, where we could test it and tweak its performance before installing it on two production servers. An evaluation unit was brought in for testing. If it performed well, we had the option of buying it. These are the results of benchmarking the Nexsan ATABeast with ADIC's SNFS:

- Approximately 7 terabytes of data were written and read back and no data corruption was observed.
- A single running process sustained I/O rates for large files at 100 megabytes/s for writes and 180 megabytes/s for reads without tuning stripe breadths or adjusting buffer cache sizes.
- The aggregate write rate for large files with multiple concurrent processes on two hosts was a little over 120 megabytes/s, and the maximum sustained read rate for large files with multiple concurrent processes was over 220 megabytes/s.
- The rate of metadata operations (e.g., file opens, closes, etc.) was a little over 250 per second and occurred with a single process making the system calls. This rate did not scale when an additional process on the host made system calls at the same time.

ATABeast performance had blown away any fears and doubts we had about a larger storage unit being slow. We had never seen numbers in this range for storage testing of ATA/SATA devices, let alone for a storage unit with a large storage capacity. This made the decision to go with a Nexsan ATABeast a foregone conclusion. Forget additional testing: We were going directly to production. All that was left was installing ADIC's SNFS on the Sun V880s. We used the Sun 280 from the testbed as the metadata server. Since it was already configured, no additional installation would be needed. We also needed a storage unit for housing the journaling and metadata for each file system we created. For this we used the Apple Xserve RAID from the testbed. The testbed was being devoured by DataMonster.

We installed a private network among the metadata server, storage devices, and Sun servers using a more robust switch than what was used in the testbed. Our data center had a McData Switch with 128 ports, and we used this as the main FC hub. A group of ports on the switch were zoned off for shared file servers and storage units. By this point, we had the production environment up and running without encountering any major problems. Another ATABeast was ordered and put into production within a few weeks after the first was up and running. DataMonster was starting to come to life.

Current ADIC SNFS Environment

The current production ADIC SNFS environment has been modified over the past year (see Figure 2). We replaced the original metadata server with a high-availability configuration of two Sun 210 servers with 8GB of memory on each server. The file system journaling was spread out over the Apple Xserve RAID, but closer investigation of the unit revealed that each of their two controllers only supports half the disk drives. We had thought that each controller communicated with all the drives. Losing a controller on the unit would cause half the file systems to be lost. Additionally, replacing the controller on the Apple unit wasn't a simple swap, so downtime was required to reconfigure it back into production. The Apple Xserve RAID was replaced by a Cipricio storage unit that resolved both of these problems. The Netgear switch on the private Ethernet network between the servers and storage unit was replaced by a Cisco network switch.

Another Sun V880 was added to the configuration, but this was to be used only for large dataset computations. The earlier ATA disk drives we used weren't truly serial but had been modified to imitate serial access. SATA ca-

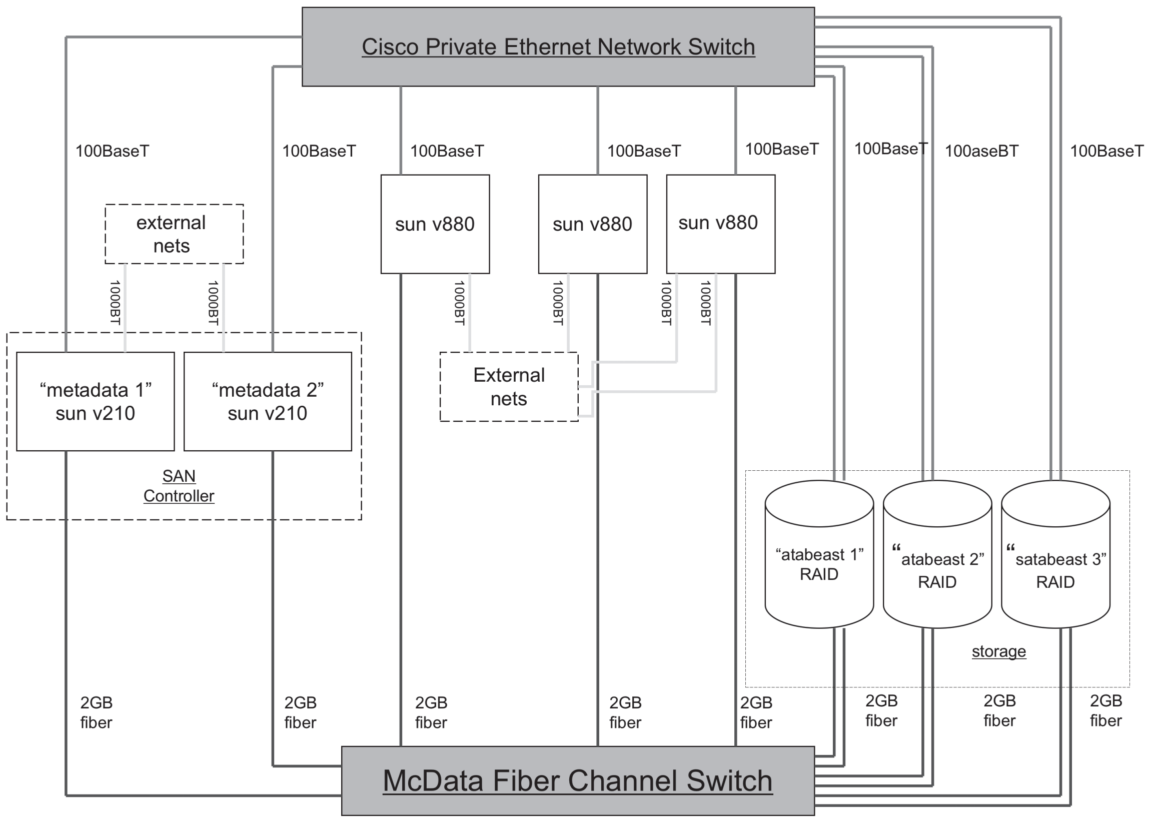


FIGURE 2: DATAMONSTER CURRENT CONFIGURATION.

bling works with ATA storage and peripheral devices. A new generation of SATA-II disk drives have emerged that are truly serial and run at double the access speeds of the first-generation ones in the earlier storage units. They run at about the same speeds as the FC disk drives. We then added a Nexsan SATABeast with these drives to the SAN; it was dedicated to shared datasets requiring heavy computational processing. The speed of the new drives ensured that I/O waiting would not cause a bottleneck for the processors.

Future Evolution

This is not the first nor will it be the last time we have had to transform a testbed into a production environment. After the initial pain of working through the testbed configuration (and I have heard similar war stories from other CXFS administrators), the production system is extremely stable and reliable. The next step will be to add SGI servers—used predominantly for visualization—to the ADIC SNFS system.

Unfortunately, ADIC has a restriction on the number of hosts it can support at 128. This is not a hard limit, but anything above this level would require major involvement by ADIC since it would saturate the metadata servers. This is a serious issue for us, because we run a large number of Linux servers. We are investigating the use of IBM's GPFS or CFS's Lustre for the large production Linux clusters. In the future we will probably run a hybrid of shared file systems, one for the heterogeneous servers such as Suns and SGIs and one for the large Linux clusters. User home directories and smaller static data would continue to reside on NAS systems. Still, we would prefer to run everything under one shared file system.

The overall cost of the system is a major consideration for us. The costs associated with using SNFS software are minimal compared to the hardware and software costs from a single vendor. The ability to select optimal storage devices and switches gives us a tremendous amount of flexibility in buying open-market cost-effective hardware. Based on previous experience and testing of shared file systems, we believe that ADIC's SNFS performed extremely well. The cost of ADIC's SNFS run around $5000/client, with no restrictions on the number of processors per client. Yearly maintenance runs around $5000 for 24/7 support. The amount to be spent on switches and servers is up to the implementer. The main benefit derived, of spending approximately $3 per gigabyte for storage, cannot be overemphasized, since there are no limits or restrictions on the amount of storage we can add to a system. In the future the cost of storage will be measured in terabytes, not gigabytes, owing to the boost in disk capacity per disk and the storage industry's tremendous growth.

MARK BURGESS

configuration management: models and myths

PART 1: CABBAGE PATCH KISS

Mark Burgess is professor of network and system administration at Oslo University College, Norway. He is the author of cfengine and many books and research papers on system administration.

Mark.Burgess@iu.hio.no

WHEN I WAS EIGHTEEN AND FRESH out of school, I worked for a couple of summers as a gardener in the well-to-do houses and estates of the English countryside, together with a small team of other gardening hands. Each sunrise, we were transported at breakneck speeds around the veins of the Midlands in a rusting station wagon (appropriately known as "estate cars" in the U.K.) by my boss, Ken, a soon-to-retire adventurer with a British Air Force moustache and a love of music and the outdoors. We would discuss the great Russian composers as we clipped away at hedges and tilled over flowerbeds.

As workers, we were somewhat human in our modus; it was occasionally precarious work, and we were not altogether skilled or efficient, especially after the lunch breaks during which we would ritually consume as many rounds of beer at the local pub as there were workers on that particular day (a matter of honor, and not something that I could conceivably survive at present). These were "Bruckner episodes," as Ken noted, ever trying to get me to listen to that rather mediocre composer, whose work he referred to as "traffic-light music." I later learned this meant that just when you thought it was about to finally go somewhere, it would stop, dither, and then attempt to start again. Quite.

I believe I learned several good lessons about maintenance from my stint in the English Country Gardens. The first was "Always let your tool do the work," as Ken used to point out, with a nudge and a wink and then a guffaw. In other words, know how to use the tools of the job rather than breaking your back with manual labor (aside from various carnal interpretations).

The second was about careful execution. Gardening is nothing if not a strenuous job. It seems straightforward, with everything under control, until you mistakenly cut a prize flower in two or fall foul of a mix-up—"Oh, I thought she said *do* destroy the garden gnome"—enshrined among the taskforce as "Always read the destructions." On one occasion, enthusiastic overzealousness was cut short when a friend of mine stepped backward onto a rake (in classic Tom-and-Jerry style) and emptied a wheelbarrow full of earth into the client's newly filled swimming pool. This was not policy, but we liked to think of it as one of those inevitable once-in-a-lifetime (or -workday) freak accidents. Yes, we would have been naive to believe otherwise.

Gardening was, I think, my first exposure to the idea of *configuration management*, that is, the routine maintenance of everything from the most sublime stately garden to the most ridiculous cabbage patch. Size is not important; the same errors are made in spite of scale. It is very much about seeing how the result of extensively planned and orchestrated order generally falls foul of both growing weeds and the misguided hands of the reputedly infallible maintainer. (In spite of the best-made plans of pubs and men, anything that can go wrong is not necessarily to be blamed on a pint of Murphy's.)

In this series I want to talk about configuration management in system administration from the perspective of a researcher in the field. This is my cabbage patch, a topic that I have been interested in for some fifteen years now; it is also a fascinating theoretical and practical problem, which has been overshadowed in recent years by tool talk and XML incantations. Yet there is much to be said about this topic from a research point of view, and all those eager to rush out and build their own tool should pause to take heed of what has been learned. It's not about tools; it's about principles and assumptions, just as it's about knowing the limitations and learning to deal with them. So I want to dangle a carrot of rationality in front of you, to turn the discussion away from the cosmetics of tools back to the science of the problem.

What Is a Configuration?

As with any Geek Tragedy we start by introducing the *dramatis personae* and speaking the names of our daemons to show that we do not fear them. We begin with the notion of the *configuration* itself. This seems perhaps to be a rather obvious concept, but, as we know from bitter experience, assuming the obvious to be uninteresting is the best way to avoid seeing the wood for the trees. So what is configuration about? Here are some definitions for configuration, which I found rustling through the weeds:

> The appearance formed by the arrangement and relation of parts.
>
> An arrangement or layout.
>
> Functional or physical characteristics of hardware/software as set forth in technical documentation (from a software document).
>
> The specific assemblage of components and devices that makes up the components of a complete system.

Of these, I like the first and the last the best. It is clear that configuration has to do with identifiable patterns and how the pieces fit together to make a whole. In the last definition we also mix in the idea of a system, that is, that a pattern might actually perform a role within some functional mechanism.

What Is Configuration Management?

The IEEE Glossary of Software Engineering Terminology (Standard 729-1983) defines configuration management as:

> the process of identifying and defining the items in the system, controlling the change of these items throughout their life-cycle, record-

ing and reporting the status of items and change requests, and verifying the completeness and correctness of items.

This definition is a software engineering definition, but it captures the main features of what we understand by host or network configuration management. I like especially that the definition talks about the process surrounding the configuration (i.e., management) and not merely the definitions or the tools used. In particular, the idea of change management is included as part of the process, as is verification, which implies some kind of monitoring.

Often system monitoring is separated from the idea of configuration implementation. Possibly this is because it was originally assumed that implementation would always be executed by humans. Elaborate monitoring systems have been devised, but these are often read-only. As we shall see later in the series, this is a key challenge to be met.

State and Configuration

In computer science we talk a lot about states. The idea of a state is part of the fundamental theory of computation, and most of us have an intuitive idea of what is meant by the concept, so can we relate configurations to states? What these ideas have in common is that they are all alternative values of a variable quantity, such as a register. A configuration is somehow a pattern formed by a number of state variables. Let's recap the different ideas of state to give the devils their names.

The simplest idea of state is that of a *scalar* value, or a single item. For example, file attributes in UNIX are simple register-like values. When we write `chmod 664 file-object` we are defining the state of the mode register for the file object. The state of the register belongs to the set {000,001,002, . . . ,776,777}, which is called the *alphabet* of the state. We think of each value as a separate symbol of an alphabet.

When the possible values the variable can take are limited, we speak of a finite number of states. An algorithm that uses and manipulates the values is then known as a Finite-State Machine.

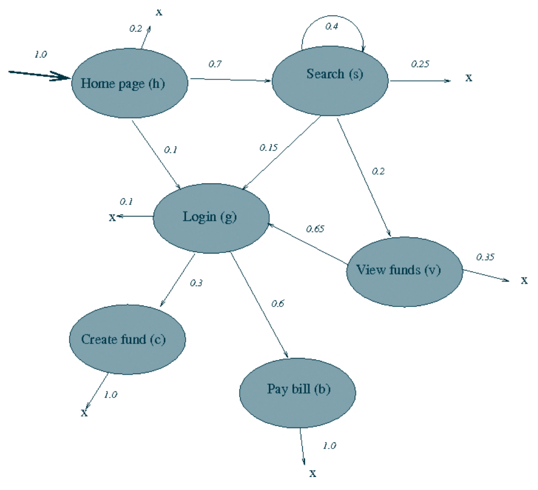


FIGURE 1: A FINITE-STATE SYSTEM FOR AN INTERNET BANK WEB SITE, WITH TRANSITION PROBABILITIES MARKED IN. ARROWS REPRESENT POSSIBLE TRANSITIONS.

In a dynamical system, change is represented by changes of state. In other words, certain variables or registers change their values from one symbol in the alphabet to another. The different state values can be shown as locations (see Figure 1), and transitions can be drawn as labeled arrows, where the labels remind us of the cause of the transition (i.e., the operation that was executed that resulted in the change). Most of us remember the state diagrams for process execution in the kernel, labeled with names such as "ready," "waiting," "dispatched," "idle," and "zombie."

Transitions between the states occur as a result of different actions. Sometimes we control those changes, and sometimes we merely observe the changes. This is an important thing to remember, as it is often assumed that once a value has been set, its state remains fixed until we alter it. But because of the external environment of a system, that is not true. We might plant the seeds, but they grow all by themselves, thanks to rain and sun.

What about "containers" more complicated than scalar registers? Well, permissions of more contemporary file systems have more complicated attributes than the UNIX file system. Access Control Lists are *lists* of scalar values, whose state can include lists of user identities to be granted access. The system process table is a list of state information, split into different columns, that is, formed from different categories or *types* of information. These are embellishments on the idea of a list. Okay, so we need lists, but these are not much more complicated than scalars.

Text files and *databases*, however, reach a new level of complexity. Text files are ubiquitous for holding configuration information on UNIX-like operating systems. Databases are the other universal approach for storing values.

The contents of files and databases are not of a fixed length, so we cannot think of them as mere registers. If they are ASCII-encoded files, then we can say that they are composed from a finite alphabet but that is not the same as having a finite number of states. File size is only limited by memory.

The most pressing feature of files is that they have *structure*. Many UNIX files have a line-based structure, for instance. XML files, which are rampantly popular, have a structure based not on lines but on a parenthetical grammar. Most programming languages are not reliant on starting a new line for each new statement either: Their formatting is also based on a scheme of statements organized in nested parentheses.

What is important about the structure of information is not what it looks like, but how easy or difficult it is to understand the information in these structures—not just for humans, but for computer algorithms too. Any tool or scheme for determining the content of a file object or database must deal with the real complexity of the structure used by that file object to store and represent information.

Let's abstract this idea to get a better idea of what we mean. With structured state information, we are not just deciding the color of a single rose for our garden; rather, we are planning the landscaping of the entire estate. We cannot expect a single spade to deal with the complexity of this garden. The tools that create and maintain such a structure need considerably more intelligence. This idea has been studied at length in computer science, because it is a problem of far-reaching and general importance.

Patterns, or Variations of State

As a state changes, either in time or across different locations or objects, it forms a pattern. It paints a picture or sows a garden. The pattern might be so complex that it seems inconceivably random, or it might form some kind of simple and recognizable structure. The structure might be best comprehended as one-dimensional, two-dimensional, etc. As you see, there are any number of characterizations we might use to accurately describe the pattern.

Arguably the most basic question we can ask is the following: Is the pattern formed from *discrete symbols*, such as individual flowers or entries in a database, as in the specification in Figure 2?

```
xxxxxxx----------------
xxxxxxx----------------
xxxxxxx----------------
xxxxxxx----------------
-----------------------
-----------------------
-----------------------
```

FIGURE 2: DISCRETE SYMBOLS USED TO BUILD DISCRETE SYMBOLS. HERE THE ALPHABET IS {X,-, } AND WE PLANT OUR FLOWER BEDS TO REPRESENT A FLAG.

Or is it a *continuous variation*, such as the curve of the path through the rolling mounds or the changing level of the system load average (see Figure 3)?

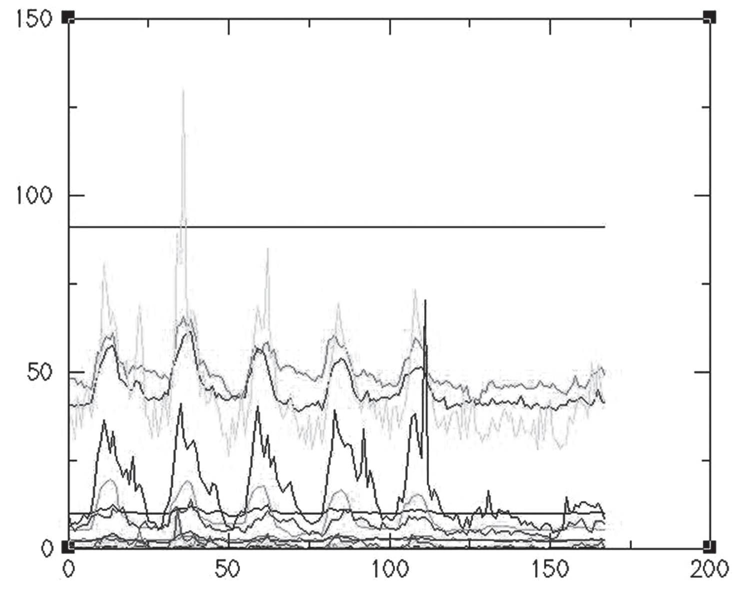


FIGURE 3: A CONTINUOUS CHANGE OF STATE, SHOWING THE CHANGE IN AVERAGE LOAD LEVEL OF CONNECTIONS TO VARIOUS SERVICES OVER THE COURSE OF A WEEK.

This choice between discrete and continuous points exemplifies a fundamental dichotomy in science: the competition between *reductionism* and *holism*. That is to say, discrete values generally emerge from trying to break things down into atomic mechanisms, whereas continuous values come about by stepping back for a broader, approximate view. Variables that try to capture average behavior, for instance, vary approximately continuously (as in Figure 3). The theory for describing patterns in these categories is rather different in each case, which explains why intrusion detection and anomaly detection are so difficult.

Patterns are everything to us, not just in configuration management, but in the management of anything. We organize things according to patterns. Patterns are how we understand when things are the same or different. There are patterns in time (repetition, scheduling, etc.) and there are patterns in space (e.g., disk management, directory management, and cluster management). When we manage things poorly it is often because we have no good way of describing and therefore measuring the patterns of the resources that we need to deal with. Indeed, it is a paradox that beings that are so good at recognizing patterns are often quite inept when it comes to utilizing them. This is odd indeed, because humans are excellent language processors, and the principal way of describing discrete patterns is with language. The study of computer languages is the study of patterns. Continuous patterns are an essential part of our visual and auditory recognition. The language of these is the calculus of differential mathematics.

What discrete and continuous patterns have in common is that patterns of either kind are difficult to identify. Our brains are incomprehensibly successful at identifying patterns (so much so that we see patterns even when they are not there), but finding algorithms for recognizing and even classifying patterns and for associating meanings with them (semantics) can be one of the most difficult problems to solve.

Does that mean we should not try? Is management a waste of time? Clearly it is not. Much has been accomplished throughout history by our willingness to forego complexity and employ simple patterns that we can comprehend more easily. Of course, tackling this issue involves some sacrifice, but identifying the patterns offers greater predictability and hence reliability. It is an important lesson that the real intellectual achievement is to simplify a problem to its core essence—anyone can make something more complicated. Indeed, science or natural philosophy is about looking for *suitably idealized approximations* to complex patterns, not about wallowing in detail. Also in configuration management, we must forego complexity to achieve reliable management. This is a theme I'll be discussing in future issues.

COM:POSTscript

At risk of turning this into a Bruckner episode, we must leave it there for this time, before the fruits of this batch get us embroiled in a jam. In the next part of the series I want to talk about the ways in which patterns can be classified by algorithms. This is an important step toward the automation of configuration management. We'll see why the only real tool we have for pattern matching symbolically is the regular expression and why intrusion-detection systems are trying to solve a hopeless task—identifying who is a pod among the gardeners!

Gardens can be impressive tiered sculptures full of botanical variety, jungles rich in diversity, or scrapyards full of junk. It's the same with our computer configurations. Which is easier to understand? Which is better to live with? These are not easy questions to answer, but they are among the questions we shall try to address in upcoming issues.

DARRELL FUHRIMAN

identity management identified

Based in Portland, Oregon, Darrell has been managing UNIX systems for nearly thirteen years in a wide variety of environments. This fall he will be changing gears and pursuing a master's degree in geography at Penn State.

darrell@garnix.org

AS ANY IT WORKER WHO HAS SPENT time in the enterprise knows, the question of creating and managing accounts across a variety of systems is a thorny one.

Windows doesn't easily talk to UNIX, which doesn't easily talk to your ticketing system. None of them talks to your HR system, which has its own set of logins, plus everything wants to manage its own password and of course nothing stores passwords in the same format. And just how do we assign someone a phone extension?

When an employee gets hired, tickets have to be opened with each of the groups managing each system until finally, possibly days later, the new employee has all the access he or she needs. But then over the years, the employee changes roles and needs a different set of systems until finally, several years later, the employee leaves. Then someone has to figure out what systems the employee has access to (since the employee may not actually know anymore) and set about removing access, deallocating resources, forwarding mail to the manager, etc.

These are the kinds of problems that an Identity Management System is meant to solve.

What Is Identity Management?

In an enterprise IT infrastructure, we have a wide variety of systems, many of which need to know something about who's using it. What each application needs to know varies dramatically: An HR system may need detailed information on every employee; a Web server may simply need to know only basic authentication and authorization information; and the PBX may need to know who a person's assistant is, so the phone can be automatically forwarded if it's not answered after a certain number of rings.

Identity Management (IdM, if you're hip) in its simplest form is nothing more than the tracking of all attributes about an individual and the synchronization of those attributes among different data sources. Known as a *metadirectory*, this is the initial concept behind what we now call Identity Management.

This differs from the concept of account management, which is the process for allocating resources to a person (a login account, a home directory, etc.) and removing those resources when told to. Because there are typically numerous resources that need to be allocated to a person, it has traditionally been important to automate these processes as much as possible. That's the traditional view of account management, and one that has been much discussed at past system administration conferences.

In practice, account management often fails to take into account the question of when and which resources should be provisioned as a user's role changes and, more importantly, when they should be deprovisioned. In other words, we need to regularly update the resources allocated to a person through his or her entire tenure within the organization.

As system administrators, we have often automated the process of creation and deletion. I don't think I know a system administrator of significant experience who hasn't (re)implemented this multiple times; the process that triggers the creation and deletion is usually some action that is external to the network such as email, a phone call from HR, or the arrival of someone in their cube.

Rather than having a manual initiation, if we take our metadirectory, where we share data among different systems, and we add actions triggered by changes in these data, we would have a system that managed a user's resources and permissions with no manual intervention. We would have an Identity Management System.

You might say, "I can do that," or you might say, "That sounds hard." Either way, this is much harder than you think it is. At one regional conference I attended most of the stories were about how poor planning had led to repeated attempts to deploy the solution—one large company was about to attempt their fourth roll-out because of unrealistic timelines and a lack of understanding of the complexity of the problem.

So, How Do I Do It?

The modern enterprise typically has dozens of connected data sources (and accounts on each one), so there isn't going to be a single group that knows everything about all the systems. That means you need to get to know your other IT groups—you'll need them.

I worked on an IdM project at a university where our initial step was to do a survey to discover what accounts existed on which systems. We discovered seventeen different account types, including:

- Calendaring
- Ticketing system
- Student information database
- MySQL databases
- Temporary wireless access
- Online course management

Even after the discovery phase was officially over, we continued to find new account types up to a year afterward.

The next thing was to identify roles (Student, Staff, Student Employee, etc.) and associate them with the types of accounts they were allowed. This was especially daunting in a university, where roles were quite fluid. It was common for Students to become Faculty, for Faculty to become Staff, and Staff to become Students, often maintaining more than one role at the same time.

After identifying roles, you should build a matrix identifying which roles have what attributes. This will be invaluable as you progress in the construction of your IdM system.

The lesson here is that when implementing an IdM solution, the single most important thing you can do is to plan well. That doesn't necessarily mean you need to go crazy with the GANTT charts right away—what it does mean is that you have to really understand your data.

Planning a Deployment

We know we have a large number of account types and resources we want to manage, but the single biggest mistake you can make at this point is to bite off more than you can chew.

A successful implementation is a carefully *staged* implementation. By implementing your IdM in stages, you learn something new in each stage, and you can move from relatively simple implementations and workflows to more complicated ones.

For instance, as your first step you could implement a metadirectory that simply moves attributes from one data source to another, possibly modifying the data along the way.

This requires that you understand first the data in each source, then where that data needs to go, and finally who is allowed to change it. Reread that sentence—it's a key one.

Let's look at a fictitious corporate phonebook—a relatively simple application with two pieces of data, namely, a name and a phone number. Unfortunately, our phonebook has three methods of access—people can look it up inside Outlook, which looks in the Active Directory, or it can be accessed via a Web page, which pulls that information from a UNIX-based LDAP server, and of course people can use the directory functionality built into the PBX and phone.

So we have three data sources that ideally should have the same data. Obviously, the PBX knows best what someone's phone number is, but it doesn't really have any way of knowing the name assigned to it, unless you tell it. Similarly, we need to get the phone number back into the two phonebooks—probably through a manual process.

So, let's look at a table that reflects this:

Attribute	*Contributing Data Source*	*Export Mapping*
First Name	AD givenName, LDAP givenName	AD givenName, LDAP givenName, PBX FIRST_NAME
Last Name	AD sn, LDAP sn	AD sn, LDAP sn, PBX LAST_NAME
Full Name	AD givenName + sn, LDAP givenName + sn	AD cn, LDAP cn, LDAP gecos
Phone Number	PBX EXTENSION	AD telephoneNumber, LDAP telephoneNumber

In this table we have all[1] of the information we need to start implementing a metadirectory. We know that the PBX controls our extension and it needs to go onto both AD and LDAP. More importantly, we know that changes to that information in sources outside of the PBX are incorrect—and a good IdM system will correct those mistakes automatically.

1. This is a bit of a lie. For example, you still have to figure out how to deal with multi-valued attributes, and this simple table makes no mention of data transformation requirements—for instance, an LDAP date string probably can't be directly fed into an RDBMS DATE type.

Of course you need to do this for each and every attribute—and even a simple system is likely to have dozens of attributes that need to migrate.

So all of this is nice, but wouldn't it be better if rather than simply *moving* the data—which still requires a manual intervention at each data source to create—we *create* the data. That's where our metadirectory becomes an IdM system.

Because people usually have phone extensions assigned to them when they are hired, we can define a workflow around the creation of a user. Now, when we add an employee to the system our IdM system will execute a series of actions to ensure the user is provisioned appropriately.

In our example, assume our PBX provides an interface for programmatically adding extensions, configuring voicemail, etc. We can take advantage of this so that when we create a user in the Active Directory, the IdM system executes a workflow that creates an extension in the PBX and in return gets the phone number, which it then distributes to any export-mapped source.

By doing that, we've taken one manual step out of the process of adding a user. More importantly, we've gained insight into the quirks of our IdM system and we've programmed an external action to be triggered based on an action. But why stop there?

If we move the entire account creation system inside the IdM system, then not only can we add phone extensions but we can create a mail login and set its quota, create a home directory, add them to NIS, etc.

At this point, why should we stop with this interface? If you connect it to your HR system, you can automatically trigger the creation of accounts when a person is hired—and in a move that will make your security officer happy, you can disable or delete them when a person leaves the company.

Let's say we want to design a workflow to allow your helpdesk to change someone's quota. But what happens if you have charge-backs for your disk allocations, which require approval for the charge? Rather than update the account immediately, before the change occurs you must send it up the chain to the person's manager who can approve the charge. And then if we discover that there's not enough space for that allocation, we must have the system open a ticket with your system administrators to migrate the mailbox to another server—and automatically update the quota when the ticket is closed.

As you might guess, these workflows can grow very complicated. This is why it is absolutely critical that you understand your data before you start an implementation of an IdM system. This is doubly true if you have a large number of systems or types of users.

Because IdM is so daunting, it's highly unlikely you'll want to grow your own solution. Most of the software currently available comes from the big names: Oracle, Sun, Novell, and Computer Associates. These products have a variety of tools for defining what each data source has versus what it should have and, most importantly, for executing actions based upon that.

Most of these tools have native support for many directory sources, such as LDAP, NIS, and Active Directory. The best also include the ability to write custom code to connect with other sources and the ability to insert custom code to do data transformation.

A Plea

Unfortunately, not all software vendors have realized the importance or magnitude of the problems of account and identity management. Many prefer to work in their own self-contained world, each housing their own user database, passwords, and often even access controls.

Much of what is managed by an IdM system is necessary only because we are trying to connect these worlds together. A better long-term solution is to integrate applications into single identity stores and, where that is not possible, to provide external interfaces for easy integration into the enterprise IdM system.

There's no reason Web-based course management software should have its own database of users, nor your trouble-ticketing system, nor your enterprise calendaring system. This is a house that cannot stand for long, and it is critical that we, the in-the-trenches practitioners, demand more of our vendors.

Conclusion

In the end, the idea of an Identity Management System is a simple one—monitor attributes, and when those attributes change, perform actions based on those attributes' values.

However, this simple concept has very complex implications. To manage this complexity, it's important that your implementation be very well planned and begun only after obtaining a thorough understanding of the data you are working with.

If you're careful, your IdM system will improve data integrity, flexibility, and security and can automate numerous mundane tasks—and that thorny problem of systems management will be greatly reduced.

RESOURCES

Mark Dixon and Pat Patterson of Sun both write interesting blogs on Identity Management: http://blogs.sun.com/roller/page/identity?catname=%2FIdentity; http://blogs.sun.com/roller/page/superpat?catname=%2FIdentity.

Dave Kearns of Network World writes on IdM and security topics: http://www.networkworld.com/topics/identity-management.html.

Kim Cameron writes a more technical blog on IdM, metadirectories, and security: http://www.identityblog.com/.

ANDY SEELY

system administration of command and control systems

Andy works for Northrop Grumman as the technical lead for the Global Command and Control System at the Headquarters, U.S. Central Command (J6), in Tampa, Florida. He taught Linux and UNIX courses at the University of Maryland in Europe for five years, has been a command and control system administrator for over a decade, and was the promoter for the 2006 GCCS System Administration and Engineering Conference. His wife Heather is his init process.

seelya@centcom.mil

IN THE MILITARY COMMAND AND CONTROL (C2) environment there are many missions that come together to allow a commander to fight, but it is the system administration mission that is the least understood. My day-to-day system administration tasks are typical of those in any systems support shop, but they have a unique flavor that forces me to think creatively about sysadmin problems. The C2 sysadmin focuses on maintaining totally reliable and robust systems; a system crash on one side of the world can mean death on the battlefield on the other side of the world. Yet a C2 system is highly conservative, and the C2 syadmin is not allowed access to many traditional tools such as compilers, debuggers, or network sniffers. In this article I explain how the limited environment and mission focus changes the way I and my colleagues approach the job of system administration.

Mission Orientation

My systems are common: a Sunfire v240 with 2 GB of RAM and two 800MHz processors, Dell 2650 rack-mounted servers, Gateway desktop workstations, etc. But put these systems in a C2 context and the term *mission oriented* forms the basis of all support efforts. It is from mission orientation that all other critical issues are derived. A C2 system is ultimately a system for which failure may mean the death of American soldiers far from the point of that system's support effort. The soldiers who depend on the products of a C2 system execute missions where life and death lie in the balance; a C2 system administrator has to keep this in mind every time a server is rebooted.

Mission orientation is relevant to industry in that it can bring a dedication and drive to systems support. Other systems support shops may exhibit these traits: Civilian law enforcement and emergency response, finance, and legal systems, for example, all may have the same life-or-death importance found in a C2 system. With the stakes this high, a mission-oriented C2 system presents an interesting, unique, and above all challenging system administration opportunity.

Critical Support Issues

Maintenance of C2 systems is focused on getting the data or functionality to the end user. All other considerations fall to a distant second place. Downtime for backup and recovery drills is not typically an option. Key goals in C2 support include uptime, alternate data paths, failover, and hardware backups. The bottom line to any C2 support is a simple, repeatable solution that may be implemented in a minimum amount of time by a less-than-seasoned system administrator.

Like any other systems shop, my uptime metrics are important and scrutinized closely by leadership. More importantly, C2 systems may be at the core of sustained military operations in such a way that uptime is the only thing that matters. Uptime is maximized through a close understanding of the relationship between the kernel and the application and the network interface. Under normal situations a server may appear to need a reboot. In critical situations a careful analysis may show that bouncing a network interface using ifconfig may be all that's needed. Manual restart of system services via /etc/init.d to allow system resources proper allocation may also serve to "reboot the process" instead of rebooting the computer. Unless the problem resides in the core of the system, memory thrashing, process table management problems, or I/O subsystem problems, it's likely the system itself does not need a reboot. The problem with this type of repair is that it requires a senior sysadmin to do the analysis and come to a conclusion quickly. If the problem causes degradation to the point of service interruption for longer than it would take for a normal reboot, then the window of opportunity has been missed.

Failover is a goal for any important server system, but it is not always as viable in a Government Off The Shelf (GOTS) software world as it is in a commercial environment. Commercial software may be explicitly designed to perform a failover function; DNS, SMTP, and HTTP services, for example, are easily set up in a pooled environment that will allow any given server to fail without the entire service suffering a failure. Unless it's specifically contracted to be so written, GOTS software does not typically allow this type of pooling. In most cases failover of a C2 system is a manual process. Certain failover points may be set up in hardware, such as disk mirroring and firmware instructions for boot order to recover automatically from a failed boot disk, multiple network interfaces on different switches to allow for failure of a network leg, or multiple power sources from different power legs to allow for inconsistencies in the power grid.

Systems may die for a variety of reasons, from software that degrades its own configuration, to hardware that stops passing electrons, to a hurricane that flattens the server room. When a system dies, there must be an alternative solution to the data-processing mission. In military terms this alternative routing is known as an ALTROUTE. When the software repairs, system reboots, and hardware swaps can't fix the problem or can't fix it fast enough, there must be an alternative. Every critical C2 system must have a viable and tested ALTROUTE plan in place. Regular exercising of that ALTROUTE is essential to ensure that procedures are viable, personnel knowledgeable, and the end user of the data can actually successfully accomplish the mission in an ALTROUTE configuration.

When C2 hardware fails it must be repaired or replaced. Hardware warranty contracts can be expensive and, in some locations, impossible, and maintaining on-site hardware technicians can be even more expensive. There is a cost-benefit relationship where the variables are cost and mission restoration time. In most situations the greatest speed for return to service may be had at the lowest cost by simply having one or more complete spares on the shelf for every critical bit of hardware in service. This allows me to make rapid change-out of a failed server while buying a cheaper warranty plan for getting the failed system repaired.

My ultimate solution to satisfy service delivery and critical goals is one that may be accomplished by the most junior member of my team. Can my most junior sysadmin restart a service instead of a server? Can he or she implement the ALTROUTE procedure or pull the dead server out of the rack and replace it with a cold spare? Can this lowest-paid member of the team identify the critical points for C2 systems support? After years of C2 system administration work, I've come to truly appreciate that critical and simple are values that cannot be separated.

Dealing with GOTS

GOTS software development is not subject to the same pressures as open source software. Transparency is considered a bad thing, and I have almost never been granted access to source code. Because government contracts for development define clear boundaries for product deliverables, interoperability with other software and open APIs for system administration analysis just aren't there. I cannot assume that GOTS software will behave in a fashion consistent with commercial software or with related GOTS software, or even with different versions of itself. This leads to significant problems with integration and performance tuning. How do I right-size a server's physical RAM if there's no documentation on how much RAM a product requires to run correctly? How do I correctly configure the site border protection when the behavior of a product's network interface is so poorly understood that there are frequent and heated debates on such fundamental issues as whether the product uses TCP or UDP? How do I explain to the user community why their applications don't run correctly when they were developed for a workstation security model greatly different from their own?

Successful administration of GOTS requires a savvy, creative system administrator with a handy box of stock operating system tools, including truss, strings, files, snoop, ptree, and od. If the tool didn't ship with the OS, then I can't use it. C2 systems are subject to configuration management rules that sometimes make good sysadmin practices impossible. If I were to install gdb and Ethereal, for instance, I would be committing a security violation that could potentially shut my operation down. The solution is to be creative with the tools you have. The following are examples of some problem-solving techniques I've had to use in a Solaris and Bourne shell environment.

Problem: GOTS software listens on the wrong port. Suspect this is a configurable option but don't know where the configuration file is.

Solution: Find all ASCII files in the software's installation directory and grep for the bad port number:

```
find /path -print | xargs file | egrep -i "ascii|command|text|script" | awk -F:
'{print $1}' | while read fn; do; grep PORTNUM $fn > /dev/null && echo $fn
>> results; done
```

Problem: The configuration file is known and in a standard location, but the software does not change behaviors when configuration values are changed. Suspect more than one configuration file is on the system.

Solution: Either do a system-intensive global find for all files of the same name or truss the process on startup to find the specific file in use:

```
find / -name "config.file" -print &
truss -a -e -f /path/to/start 2>&1 | grep config.file
```

Problem: GOTS software turns out to be a GOTS wrapper around commercial software. Network behavior is on the right ports and protocols, but it is not behaving correctly. For example, my GOTS DNS turns out to be an older version of BIND and it doesn't resolve TLDs on my closed network segment.

Solution: Test your assumptions. Is BIND using the correct root name servers for the network? Snoop the network interface for the correct root server while doing a name resolution. Or just validate the hints file; commercial software rolled into a closed government network may not have been set up correctly in the first place.

```
snoop -v root.server.IP.address
```

Problem: GOTS software is delivered as an ELF binary with no documentation. Command line switches are suspected but not known. How do I get a help listing?

Solution: Most GOTS doesn't conform to the --help convention, but sometimes a product will try to correct you when you make a request it can't handle. Feeding the program gibberish will sometimes produce a usage report. If not, the strings command and some educated guesswork can help. Try to get a usage message with the following:

```
./program -asdf
./program /asdf
./program asdf
```

You can use strings to find common arguments:

```
strings program | egrep -i "debug|help|-v|-i"
```

Problem: GOTS software output is in a binary format that resists analysis. While troubleshooting, it's suspected that a data stream issuing from the software is corrupt.

Solution: Capture a known-good data stream and compare using od and diff. Use the -c option to od to display the human-readable elements of the data stream:

```
od -c stream.one > od.one
od -c stream.two > od.two
diff od.one od.two
```

Problem: GOTS software is doing significant IPC. There is a failure in the main process, but it's suspected that the problem is with a subsystem. As usual, there's no documentation.

Solution: Use ptree to analyze the way the processes fork() and isolate situations where a subprocess is hanging awaiting IPC input; zombie processes are a good indicator that IPC is being handled poorly in the software. Use truss -f to follow system calls in each fork and identify problems with resource management; for example, a parent calls may wait() on a child or the child may be blocked indefinitely owing to a dead IPC pipe:

```
truss -f -o /tmp/trussfile ./start.program &
[PID]
/usr/proc/bin/ptree PID
```

GOTS software is most commonly developed under government contract. Occasionally the government will use a commercial or open-source software solution but wrap the software in a GOTS package for consistency. I find this to be a special challenge; to fix the GOTS package, I have to be extra-intimate with the standard software when I walk in the door. Full understanding of the way a software package works "in the real world," as we say in the C2 shop, is critical to debugging why it doesn't work for us. I try to isolate problems with the core software configuration, the GOTS software delivery, or system integration in a GOTS environment, in that order.

Dealing with Legacy Systems in a Conservative Environment

The C2 environment is highly conservative. Change happens slowly and for good reason. The latest and greatest system introduces more problems than it solves:

1. Bringing a new system online requires significant planning to ensure minimum downtime, and it never goes the way you expect it to. Why introduce risk into a stable system?
2. Current systems are better understood by the operator and administrator than any new system.
3. Legacy software may be less of a hot target for hackers, crackers, and script kiddies, whereas the newest thing is the one everyone wants to break.
4. Newer systems tend to have more and newer features, which can lead to information and option overload. Missions may be enhanced by a greater range of options, but mission operators may spend so much time figuring out and playing with the options that the mission falls to the side. Administrators end up responding to "problems" about new features that have nothing to do with the mission.

The key to system administrator survival in this environment is to focus on the service delivery mission and not get distracted by new features and new software. The latest, from my C2 perspective, is not always the greatest.

Keeping Personal Skills Up with the Industry

It's a real challenge to be a serious system administrator in a fast-moving industry. Being a C2 system administrator makes keeping up feel to be impossible. How do I maintain status as a professional system administrator while balancing the requirement to maintain systems that seem like they never change? I've found the keys to be an awareness of the movement of the greater industry and a focus on generalized, creative skills. I believe that limitations breed creativity, and this can be a bonus for the C2 sysadmin. Being a pathological optimist doesn't hurt either.

My key points for keeping up in industry include the following:

1. Professional association memberships. Although I certainly don't have time to read every magazine that shows up in the mailbox, I do have time for abstracts. Professional organizations are the heartbeat of the industry, and even a casual exposure can help.

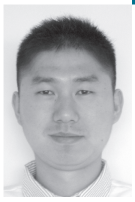
2. Certifications. Some folks think a certification is not worth the effort, but keeping an OS certification current forces a C2 sysadmin to follow changes in the OS and demonstrates commitment to the profession.
3. Training, seminars, and conferences. Subject to funding, of course, technical interchange with broader industry is the single most beneficial thing a C2 sysadmin can do to prevent professional atrophy.
4. Home hobby systems. I may not be supporting the cutting edge on the job, but you can bet I do at home.
5. Involvement with the open source community, IRC channels, local and user groups; these are the grassroots of the whole industry.

Serving a Higher Mission

At the military commands where I've worked, I've become dubiously famous for my verbal stream editing. I never speak a sentence that doesn't come through the command `sed -e 's/problem/challenge/g'`. But I truly believe that the difference between a problem and a challenge is whether you control the situation or it controls you. The issues my team and I face every day with GOTS software, limited tools, critical missions, and atrophy of skills are all challenges when we approach them with creativity and personal responsibility. There are few sysadmin jobs with such an impact on the lives of others. A good sysadmin in the industry is not necessarily a good sysadmin in the C2 world, but a good C2 system administrator is likely to have great potential in industry. The creative thinking required by closed systems more than makes up for the lack of cutting-edge systems, while the mission orientation encourages drive and focus uncommon in the industry.

KYOUNGSOO PARK AND VIVEK S. PAI

CoBlitz: a scalable large-file transfer service



KyoungSoo Park is a fourth year Ph.D. studentin computer science at Princeton University. His research focuses on the performance and security issues in large-scale distributed systems such as content distribution networks (CDNs) and the domain name system (DNS).

kyoungso@cs.princeton.edu

Vivek Pai is an Assistant Professor at Princeton University, where his research spans operating systems, networking, and high-performance applications. His research group is responsible for the CoDeeN system, described at http://codeen.cs.princeton.edu.

vivek@cs.princeton.edu

WITH THE RECENT EXPLOSIVE GROWTH in the scale and the size of Internet file downloads, we need techniques that provide high performance without burying servers under heavy loads. CoBlitz provides a timely solution to this, using unmodified Web browsers and servers as its clients and origin servers, a locality-aware front-end network, and a bunch of caching reverse proxies that have shown performance that meets or exceeds BitTorrent in most cases. And, best yet, CoBlitz is available for use today.

As bandwidths to the home increase, large file downloads are becoming increasingly popular on the Internet, with movies and software distributions commonly ranging from the hundreds of megabytes to several gigabytes. In its first five months of providing videos, Apple's iTune store provided over 15 million copies of TV shows and movie trailers, and Google Video now provides free downloading of thousands of video clips, from humorous home videos to professional music videos. By using small display sizes and high compression ratios, these files tend to be relatively small compared to HD-quality broadcast. However, as end-user bandwidths increase, we can imagine services providing much higher-quality downloads, with a corresponding increase in file sizes. New versions of Linux have long been distributed through the Internet, and their mirror sites get very busy shortly after every new release.

For sites that have a burst of traffic after every new release and whose users are sufficiently sophisticated, peer-to-peer protocols such as BitTorrent can be useful to offload traffic from the origin server and to leverage bandwidth available from the clients. For other scenarios, however, this option may not be as attractive, particularly if the content is not bursty or predictable, if the user population does not have browser plug-ins, or if the content consumers are other programs that only understand HTTP transfers. In these cases, it may be desirable to have a managed service that can offload the large file traffic from the origin, while still providing transfer via standard HTTP.

Although Content Distribution Networks (CDNs) have successfully provided this service commercially for standard Web content, very large files can pose some challenges for them. In particular, CDNs typically exploit main-memory caching of

Web objects, since they have a whole-file access model and a mean transfer size of around 10 kilobytes. Main-memory caching allows CDNs to reduce latency (since main-memory transfers can be hundreds of times faster than disk) and improve throughput, by avoiding disk seeks. If CDN providers start serving multi-gigabyte transfers from these same boxes, the competition for the machine's main memory can result in thousands of small files being evicted when a large file becomes popular. If several large files become popular simultaneously, then the main memory may not have enough room to cache them all and will thrash on disk accesses.

This is the scenario that we examine: how to provide a service that can *transparently* support the efficient transfer of very large files, using standard HTTP infrastructure (clients and servers). It should be capable of handling flash crowds as well as files that have a longer-lived demand, all without pre-positioning content or rehosting or reformatting the files.

Our Solution: CoBlitz

Our solution to this problem is a system called CoBlitz, which operates without any modification of the HTTP protocol, the servers, or the clients. CoBlitz evenly distributes the load of handling large-file requests to many participating nodes, to maximize resource utilization and reduce the origin server load dramatically.

The main idea of CoBlitz is to transform the large-file distribution problem into a regular small-file CDN scenario. CoBlitz transparently splits large-file requests into many requests for pieces of the file, called chunks, and has the chunks cached at multiple proxies in the content distribution network. Each chunk represents a certain range of a file, but with a little tweaking, the proxy handles it like a regular small file, thus benefiting from whatever caching strategy the proxy uses. To reduce the memory consumption on each node, CoBlitz arranges downloads so that each node is only responsible for caching specific ranges of the file. The server sees a reduction in traffic once CoBlitz caches the file to be served. Clients also see downloading speed improvement, because CoBlitz takes advantage of parallel chunk downloads while delivering to its client.

Another benefit lies in the easy deployment: CoBlitz looks like any other Web CDN, and using it is as simple as rewriting the links to be served to point to CoBlitz instead of directly referring to the origin server. The CoBlitz service just looks like a regular Web server to the client, so existing browsers, download tools such as wget and curl, or even Web services agents will all operate normally.

The Gory Details

How is this possible? Figure 1 shows the basic operation of CoBlitz. To incrementally build on an existing CDN, all of the "smarts" of CoBlitz are contained in a daemon that runs as a separate process on each CDN node. This agent looks like a standard Web server from the outside, but internally it splits the large-file request into many chunk requests, sends them to the CDN, merges the responses on the fly, and delivers them to the client in order. To improve the end-user throughput by multi-path parallel chunk downloads, it keeps a TCP-like sliding window of "chunks" and dynamically adjusts the window size to prevent the origin server or the infrastructure itself from getting overloaded.

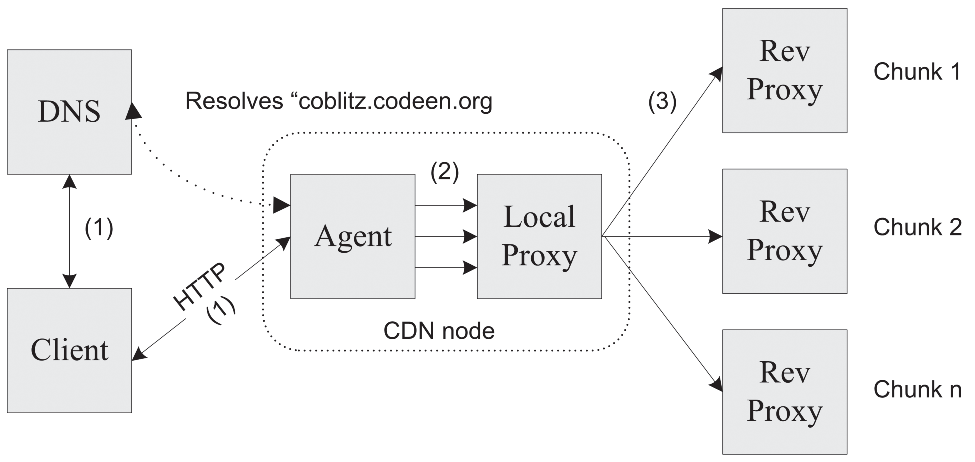


FIGURE 1: STEP-BY-STEP COBLITZ OPERATION.

Here is how CoBlitz works step by step:

1. A browser (or any HTTP-aware client) asks for a CoBlitzed URL. The format of a CoBlitzed URL is http://coblitz.codeen.org:3125/URL, where the URL is the original link before CoBlitz. Then the browser resolves the name "coblitz.codeen.org," which finds the closest CDN node to the client, and sends the request to it. The agent running on the CDN node is listening on port 3125 and accepts the request from the client.
2. Upon receiving the large-file request, the agent splits it into chunk requests and hands them to its local proxy. The local proxy runs a deterministic hashing algorithm (called Highest Random Weight [5]) to map each chunk request to a reverse proxy in its peer set. Selecting peers and maintaining the peer set are being done at the CDN level; this topic will be discussed later in the article.
3. The selected reverse proxy receives and serves the chunk request. If the chunk request is a cache hit, it is served from its cache right away, but in case of a cache miss, the reverse proxy fetches it from the origin server. The reverse proxy uses an HTTP/1.1 byte-range query to fetch the chunk rather than the whole file from the origin server. After fetching the chunk, the reverse proxy caches it as a small file rather than a range of a file.

As mentioned, the agent keeps a sliding window of chunks, and it retries any slow chunk via a different replicated reverse proxy. The retry timeout is calculated by a combination of the exponentially weighted moving average and standard deviation for recent chunks. For each retry, the timeout exponentially backs off to avoid getting overly aggressive. We allow up to two parallel downloads per chunk and let them compete with each other in case of retry. In practice, we see that about 10–20% of the chunks are retried.

The size of the chunk window gets adjusted as the transfer progresses. Whenever a retry kills off the head chunk in the window, we decrease the window size by one chunk. So when there is a problem, we can shut down the whole window within one maximum round-trip time (RTT). We increase the window size by 1/log(x) chunks, where x is the current window size (i.e., we use 1 when $x = 1$), when a chunk finishes in less than the average chunk downloading time. We increase a bit aggressively when the window size is small as with the "slow start" phase in TCP, but when the window size converges, the window growth slows down.

Challenges

Although the basic algorithm is relatively simple, the real challenge is to run it in a real distributed environment. With CoDeeN as its base delivery CDN [6], CoBlitz has been operational on PlanetLab [4] (on 600+ nodes, at 300+ sites, and in 30+ countries) for over two years. With the feedback of its real users, CoBlitz has fixed a number of problems in operation and evolved its peering algorithm.

CoBlitz adopts *unilaternal, asynchronous peering* as its peer-selection policy. Each node is independent in choosing its own peers (reverse proxies) and does not depend on any synchronous group membership maintenance algorithm, which can incur prohibitive delays in practice. The rationale behind this decision is to favor simplicity and robustness and to survive partial network connectivity [2] problems with minimal effort. As a result, scalability is easily achieved, because one can simply add more nodes as they are available, without changing or reconfiguring the peering structure.

However, unilateral peering does not guarantee perfect clustering (a clique in which each member knows which nodes other members are peered with) by design and can produce many different target reverse proxies for the same chunk, owing to differences in the peering sets. This behavior can be undesirable, because it can overload the origin server in case of cache misses, reducing resource utilization. To address the problem, we introduce *proximity-based multihop routing*, which routes the request to the best peer in the local neighborhood. Instead of going to the origin server from the first hop, each hop reruns the HRW algorithm with its own peer set to see whether any better node exists, and it reforwards the request to the better node if it exists. Each hop repeats this process until there is no better node and only the last node, a local optimum node, sends the request to the origin server. This algorithm creates an implicit overlay tree for each chunk, and nodes in the path to the origin cache the chunk while delivering it to their descendants. In this way, if other nodes near the intermediate nodes in the tree look for the chunk, the request is served without getting to the best node, distributing the load. In practice, 3–15% of chunks require an extra hop, and less than 1% of chunks are forwarded more than once.

Performance

To get some sense of the relative performance of this system, in Figure 2 we compare CoBlitz with BitTorrent and direct downloading. We use 400 PlanetLab nodes around the world as clients to simultaneously fetch a 50MB file from a single Web server at Princeton. Although CoBlitz is not intended to replace BitTorrent, this test gives some sense of CoBlitz as a reasonable choice for similar scenarios. This particular test is designed to resemble a flash crowd, and more tests can be found in our paper [3]. We tune BitTorrent for performance, allowing the origin and all peers to act as seeds for the duration of the test. BitTorrent clients take a variable amount of time to find their peers, and for the sake of a fair comparison, we have the CoBlitz clients delay by the same amounts. We show four measurements: direct downloading from the origin, BitTorrent, CoBlitz (in order), and out-of-order CoBlitz. Whereas CoBlitz normally operates by delivering all data in order to the client, we can configure it to deliver chunks as they are ready, and let the client assemble them. (Although this breaks our goal of using unmodified clients, it also allows us to see what price we pay for HTTP compatibility.) Other tests are shown in our paper [3].

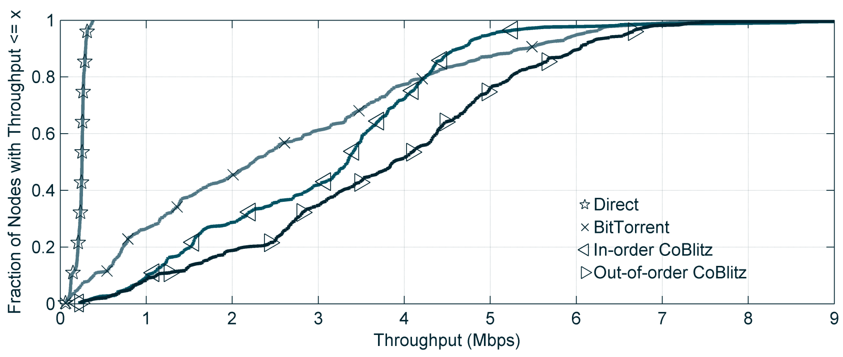


FIGURE 2: THROUGHPUT DISTRIBUTION FOR ALL LIVE PLANET-LAB NODES.

The most obvious lesson from this test is that both CoBlitz and BitTorrent are greatly preferable to having a large number of clients download directly from a single server. Even a well-connected campus such as Princeton is able to achieve only 250 Kbps on average to our worldwide clients. BitTorrent does much better, achieving 2.52 Mbps. CoBlitz outperforms BitTorrent at 79% of the clients, and it achieves an average download rate of 2.99 Mbps. The out-of-order CoBlitz shows the absolutely best performance at 3.68 Mbps, beating BitTorrent across the spectrum by 55–86%. The higher performance comes at the cost of incompatibility and would require our own browser plug-in, so we do not deploy this option.

In addition to better performance, CoBlitz better utilizes the contents fetched from the origin server as well. By fetching one copy from the origin server, CoBlitz serves 43–55 other nodes, whereas BitTorrent serves about 35 nodes. CoBlitz achieves about a 98% cache hit rate in this test, even when the document has never been seen before, dramatically reducing traffic to the origin.

Can I Use It Now?

CoBlitz was designed and developed to easily provide public access and is being used as Fedora Core mirror in six different locations worldwide, as well as a document-caching server for the CiteSeer Digital Library [1], providing over 50,000 papers through CoBlitz. Figure 3 shows the aggregate throughput of the CoBlitz Fedora mirror when Fedora Core 5 was released, on March 20, 2006. We have only a single origin server to serve Fedora Core, but with the help of CoBlitz, it achieved a peak delivery throughput of 700 Mbps and sustained over 400 Mbps for several days. This traffic was client-limited; even at these rates, we had extra capacity in our system, since our other tests have shown aggregate throughputs as high as 3 Gbps.

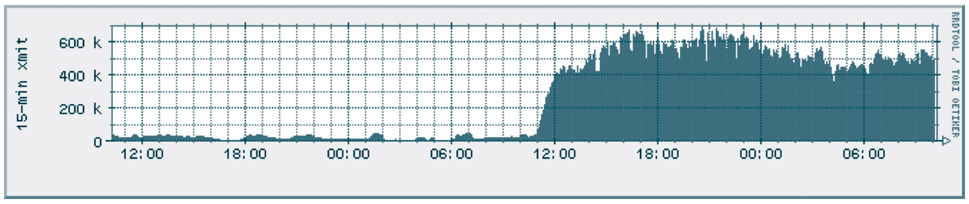


FIGURE 3: COBLITZ TRAFFIC IN MBPS (LABELED AS "K") ON RELEASE OF FEDORA CORE 5, AVERAGED OVER 15-MINUTE INTERVALS.

In running CoBlitz as a public service, we have to balance simplicity with operational overhead. As mentioned earlier, using CoBlitz simply involves adding the prefix “coblitz.codeen.org:3125” to any URL. For example, if you want to serve http://www.example.com/big-file.zip through CoBlitz, you simply need to convert it to http://coblitz.codeen.org:3125/www.example.com/big-file.zip and make it a link on any page you want. However, allowing this to happen to any URL would open us to unlimited bandwidth-shifting and possibly other forms of abuse, such as transferring copyrighted material without permission. To keep this service running for the technical community, we have restrictions in place that disallow public use of CoBlitz for entertainment media formats (e.g., images, audio, or video) but allow downloads of software, PDF documents, etc. Full details are at our Web site, http://codeen.cs.princeton.edu/coblitz/. Universities can use it virtually without restriction, as can other sites we whitelist. If you have a technical or nonprofit site and would like to try CoBlitz for your media transfers, please send email to KyoungSoo (kyoungso@cs.princeton.edu) to inquire about getting added to the whitelist.

REFERENCES

[1] CiteSeer Scientific Literature Digital Library: http://citeseer.ist.psu.edu/.

[2] M. J. Freedman, K. Lakshminarayanan, S. Rhea, and I. Stoica, “Non-transitive Connectivity and DHTs,” in *Proceedings of the 2nd Workshop on Real, Large Distributed Systems (WORLDS ’05)* (Berkeley, CA: USENIX, 2005).

[3] K. Park and V. S. Pai, “Scale and Performance in the CoBlitz Large-file Distribution Service,” in *Proceedings of the 3rd Symposium on Networked Systems Design and Implementation (NSDI ’06)* (Berkeley, CA: USENIX, 2006).

[4] L. Peterson, T. Anderson, D. Culler, and T. Roscoe, “A Blueprint for Introducing Disruptive Technology into the Internet,” in *Proceedings of the First ACM Workshop on Hot Topics in Networks (HotNets-I)* (Princeton, NJ, 2002).

[5] D. G. Thaler and C. V. Ravishankar, “Using Name-based Mappings to Increase Hit Rates,” *IEEE/ACM Transactions on Networking*, 6(1) (Feb. 1998): 1–14.

[6] L. Wang, K. Park, R. Pang, V. S. Pai, and L. Peterson, “Reliability and Security in the CoDeeN Content Distribution Network,” in *Proceedings of the 2004 USENIX Annual Technical Conference* (Berkeley, CA: USENIX, 2004).

RYAN PETERSON,
VENUGOPALAN RAMASUBRAMANIAN,
AND EMIN GÜN SIRER

a practical approach to peer-to-peer publish-subscribe

Ryan Peterson is a graduate student in the Department of Computer Science at Cornell University. He is interested in distributed systems and networking.

ryanp@cs.cornell.edu

Venugopalan Ramasubramanian is a graduate student in the Department of Computer Science at Cornell University and is currently affiliated with Microsoft Research, Silicon Valley Lab. His current research interests lie in applying principled approaches to build sound yet practical distributed systems.

rama@microsoft.com

Emin Gün Sirer is an assistant professor of computer science at Cornell University. He received his Ph.D. in computer science from the University of Washington. His recent research includes self-organizing peer-to-peer systems, reputation systems, and a new operating system for trusted computing.

egs@cs.cornell.edu

THE WEB HAS FAILED TO FULFILL ITS promise of delivering relevant news and information in a timely fashion. In fact, it doesn't deliver anything on its own at all; instead, it requires its users to explicitly poll information sources. Checking for updates by pointing, clicking, and reloading Web sites, whether the sites are Slashdot, news, or online classifieds, is not only slow, inefficient, and cumbersome for users, but it places an unnecessary bandwidth burden on content providers. Recent attempts to automate this process, with the aid of feed readers, have created more problems than they have solved. A system that detects updates to content anywhere on the Web and delivers it to users via an asynchronous channel, such as an instant message, would do much to relieve the burden on users and content providers alike.

Recent years have seen an explosion of online services, including countless blogs and frequently updated news sites. Consequently, sifting through newly published data for useful and interesting information has become a daunting task. The Web is based on a pull-based architecture that forces users to receive new content by explicit polling, which adds to the difficulty of discovering new information. Past efforts to replace the pull-based paradigm with a push-based publish-subscribe paradigm have not seen wide-scale adoption in practice. This is primarily because publish-subscribe systems have required content providers to make significant changes to their infrastructure and workflow for publishing information.

Such inefficiencies motivate a notification system for the Web that alerts users when sites are updated without demanding any support from content providers. The increasingly popular Really Simple Syndication (RSS) standard provides one alternative. RSS is both a format for publishing content and a system for detecting published updates. Content providers publish updates to special XML-formatted documents called feeds or channels. Clients subscribe to their favorite RSS feeds using special feed-reader software, which checks for updates by periodically polling the feeds and comparing their contents with the results of the previous poll. Unfortunately, RSS places a large burden on content servers: Every client subscribed to the same site must poll that site independently and repeatedly. This has forced content providers to limit client polling rates based on IP address, to save bandwidth. For instance, Slashdot allows a maximum of two polls per client each hour. Such polling-rate limitations restrict update detection time to an average of 15 minutes at best. Also, because content providers limit clients by IP address, RSS is especially impractical for clients behind NATs, since all clients behind a NAT share an IP address. For content providers, RSS traffic incurs ongoing costs as it tends to be "sticky": Once users subscribe to a site, they are unlikely to unsubscribe even as their interest in the site diminishes, resulting in wasted bandwidth. Overall, the automation that RSS provides compounds the problems inherent in a pull-based architecture.

Luckily, the recent emergence of self-organizing overlays, where nodes arrange themselves to provide a service without a centralized authority or administrator, provides a starting point for building practical systems that address the flaws of previous notification systems.

We have built and deployed a peer-to-peer publish-subscribe system for the Web called Corona [2] that uses cooperative polling and intelligent allocation of available resources to efficiently discover new information on the Web and push it to users. Corona enables clients to subscribe to Web sites called *channels*, monitors these information sources, and quickly disseminates any detected updates to clients interested in those channels. The central tradeoff in any such system revolves around the most limited resource: bandwidth limits imposed by the physical link capacities on the polling client side, the physical bandwidth limits on the server side, and the bandwidth limits stemming from the polling limits set forth by the content providers. Corona uses mathematical optimization to achieve the best possible update performance given the bandwidth available to the system. It does this by setting up a constrained optimization problem of maximizing a performance function while a cost function does not exceed a given limit. In one incarnation, used in our current deployment, it can minimize average update detection time for all subscriptions while guaranteeing that content providers never see any more requests than they would allow if the clients polled the servers directly. Thus, Corona does not require content providers to make any changes to their systems.

Corona can be accessed conveniently through instant messengers. Clients subscribe to channels by simply registering a screen name with one of several popular instant messaging systems we support [1]. When Corona discovers new updates for a subscribed channel, it pushes the updates to users as concise instant messages showing the updated portions of the Web site.

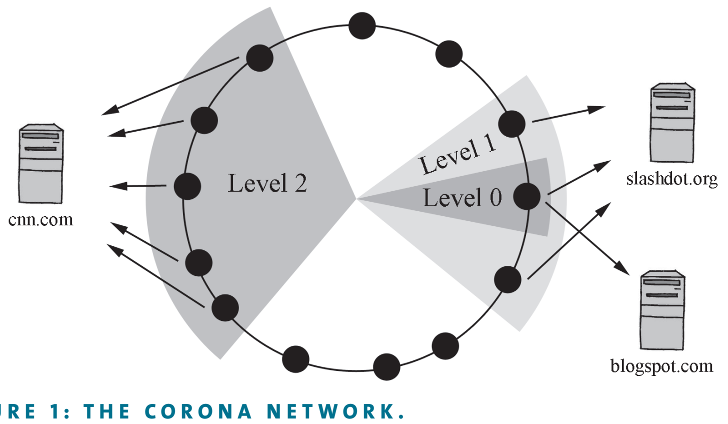


FIGURE 1: THE CORONA NETWORK. ARROWS DENOTE PERIODIC POLLING.

Corona is structured as a logical ring of nodes in an overlay network. We built Corona on top of Pastry [3], but any structured overlay that has a uniform distribution of nodes is sufficient. Corona sits between users, who submit subscriptions into the system and await updates, and content providers, which post the updates that Corona detects. Figure 1 depicts Corona's internal structure and how it polls content servers. The solid black dots represent the system's nodes, each of which is mapped to a unique location on the ring. Each channel has a designated home node, derived from a hash of the channel's URL. The home node for a channel is responsible for periodically polling the channel for updates and delegating neighbors to poll the channel as well if doing so would improve performance according to the system's performance goals. The notion of a group of adjacent nodes polling the same channel is captured formally by using polling levels, shown as shaded regions in the figure. We say that a channel has a polling level of 0 if only its home node polls the channel. A polling level of 1 means that all nodes within a one-hop neighborhood of the channel's home node poll the channel. Therefore, the number of nodes polling a channel increases exponentially as the channel's level increases. Nodes polling the same channel share updates with each other and instant-message new updates to subscribers.

The challenge here is to determine the best polling levels for the channels, each of which has a different number of subscribers, feed size, and update rate. Corona accomplishes this using a novel optimization framework, which assigns polling levels based on informed tradeoffs between network bandwidth and update detection time. Corona uses this to achieve specific performance goals, subject to given resource constraints. If network bandwidth and server load were not issues at all, the optimal solution would simply be to assign each channel the maximum polling level, so that every node would poll for every subscription. However, network bandwidth is a concern in the Internet, and, as we have discussed, content providers limit polling to prevent clients from inundating them with requests. When assigning polling levels to channels, intuitively it seems that one should give more popular channels higher polling levels. That is, if there are many subscriptions for a particular channel, there should be more nodes polling it, so that updates are detected more quickly. A channel with only one or two subscribers, in contrast, should only be polled by a couple of nodes, since there are fewer clients that benefit from updates to that channel. Similar tradeoffs hold for the channel size and update rates, which complicate the process of determining the optimal solution and render it NP-hard. Corona sets up this tradeoff as an optimization problem that describes the performance goals and constraints, then solves for the polling level of each channel using an approximate search algorithm. The algorithm runs in a matter of seconds to find a solution that is within one channel per node of the optimal solution [2].

Since there is no central authority in Corona, the system uses a completely distributed approach to assigning polling levels to channels. Each node solves the optimization problem locally and uses the results to decide the polling level of each channel it is currently polling. If the polling level of a channel changes, it notifies the nodes at the next higher polling level. Since each node uses only local information in its optimizing computation, Corona employs a mechanism that occasionally aggregates information from multiple nodes so that each node has an approximate system-wide perspective.

By assigning different polling levels to each channel, Corona can achieve a variety of different performance goals. We have already described one performance goal for Corona, that is, to minimize average update detection time across all subscriptions while ensuring that content servers do not see an increase in bandwidth consumption as clients opt to use Corona instead of legacy RSS. We call this performance goal Corona Lite. Our experiments show that this strategy improves detection time by a factor of 20, finding updates in an average of only 45 seconds after they have been published. Suppose, however, that it is important to receive prompt updates (say, 30 seconds on average) when a blogger posts a new message. Another performance goal, called Corona Fast, is well suited for this scenario: It guarantees that, on average, updates are detected within a specified amount of time, while minimizing bandwidth consumption.

The traditional approach for allocating resources in a publish-subscribe notification system is to use heuristics tailored to specific performance metrics. One such system is FeedTree [4]. Like Corona, FeedTree is a completely decentralized system for detecting Web updates, taking advantage of cooperative polling among nodes to reduce update detection time. However, FeedTree uses heuristics instead of analytical models to assign each channel to the optimal number of polling nodes, so it cannot guarantee that it gets updates to subscribers as quickly as possible.

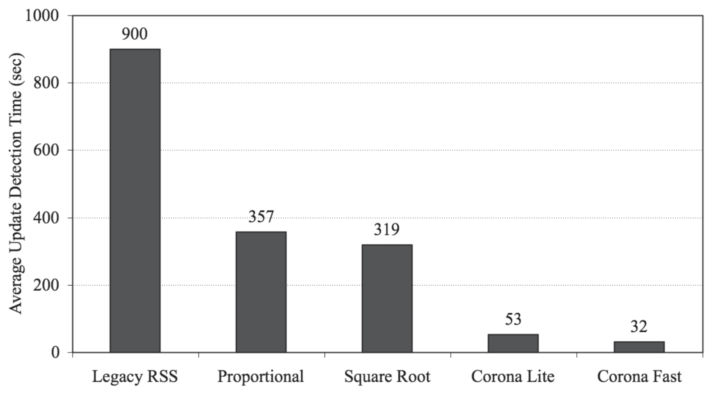


FIGURE 2: CORONA VERSUS HEURISTICS.

An intuitive heuristic-based strategy for polling nodes for updates is to poll each channel with a number of nodes proportional to the popularity of the channel, that is, to the number of subscriptions. This strategy represents the scenario where all the clients interested in a channel cooperate and share updates. An alternative heuristic, which gives more weight to the more unpopular channels, is to set the number of nodes per channel proportional to the square root of channel popularity. We compared these heuristics to legacy RSS systems and Corona to determine their effectiveness in detecting updates. The results are summarized in Figure 2. Using a typical polling period of 30 minutes, legacy RSS discovers new updates an average of 15 minutes after they have been published. We find that both heuristics improve detection time almost threefold compared to legacy RSS. In contrast, treating the problem formally as a mathematical optimization problem enables Corona Lite to provide more than an order of magnitude increase in performance, using the same amount of bandwidth as naive RSS and heuristic methods. (Corona Fast uses slightly more bandwidth for even better update detection latency, as its optimization goal is to guarantee a targeted update latency without limiting its bandwidth consumption.) Overall, heuristic approaches, although easy to devise, tend to be tailored to specific scenarios or benchmarks, do not perform well for nontrivial problems, and ultimately do not provide any guarantees.

Corona is a practical publish-subscribe system for the Web that solves the problems posed by RSS and current publish-subscribe systems. It polls on the user's behalf, intelligently allocating resources to achieve performance goals, instead of requiring clients to repeatedly poll content servers directly. Corona provides strong performance guarantees, ensuring that subscribers receive up-to-the-minute information from their favorite Web sites without introducing yet more unnecessary traffic onto the Internet.

More importantly, Corona exemplifies a new method for building decentralized Internet services. We have shown that a rigorous approach to distributed system design where performance goals and cost metrics are expressed formally can yield practical high-performance systems that vastly outperform heuristics commonly encountered in system design today. We believe that as systems become more complex and difficult to reason about, heuristics are unlikely to be successful in substantially improving system performance, and using mathematical optimization can yield more efficient systems with better resource utilization.

REFERENCES

[1] Corona: http://www.cs.cornell.edu/people/egs/beehive/corona/, May 2006.

[2] V. Ramasubramanian, R. Peterson, and E. G. Sirer, "Corona: A High Performance Publish-Subscribe System for the World Wide Web," in *Proceedings of the 3rd Symposium on Networked Systems Design and Implementation (NSDI '06)* (Berkeley, CA: USENIX, 2006).

[3] A. Rowstron and P. Druschel, "Pastry: Scalable, Decentralized Object Location and Routing for Large-Scale Peer-to-Peer Systems," in *Proceedings of IFIP/ACM International Conference on Distributed Systems Platforms* (Heidelberg, Germany, 2001).

[4] D. Sandler, A. Mislove, A. Post, and P. Druschel, "FeedTree: Sharing Web Micronews with Peer-to-Peer Event Notification," in *Proceedings of the 4th International Workshop on Peer-to-Peer Systems* (Ithaca, NY, 2005).

USENIX Membership Updates

Membership renewal information, notices, and receipts are now being sent to you electronically.
Remember to print your electronic receipt, if you need one, when you receive the confirmation email.

You can update your record and change your mailing preferences online at any time.

See **http://www.usenix.org/membership**.

You are welcome to print your membership card online as well.
The online cards have a new design with updated logos—all you have to do is print!

ERIC SORENSON

enterprise routing sinkholes using Linux and open source tools

Eric Sorenson does network administration at Transmeta in Santa Clara, California. He likes commuting by bicycle and dislikes Sarbanes-Oxley compliance audits.

eric@transmeta.com

OCCASIONALLY YOU'LL SEE THEM ON the evening news or a photoblog—a previously forgotten septic tank or abandoned sewer line has opened up and swallowed somebody's Volkswagen, and the puzzled owner is standing above his submerged car scratching his head. They're called sinkholes, and the photos are amusing precisely because it's happening to somebody else. So why would you, a sophisticated system and network administrator, want to inflict one on your enterprise?

The routing sinkhole I'll be discussing in this article has a few characteristics in common with those staples of the eleven o'clock news: They are both uncommon and unexpected, and their purpose in life is to swallow up anything that comes their way. Unlike a disused septic tank, however, a routing sinkhole has a specific useful purpose: to provide administrators with detailed statistical data about traffic that ends up falling into it. More specifically, a sinkhole is a trap for traffic that we know to be illegitimate because it's addressed to parts of RFC1918 private address space not in use by our enterprise or any other privately connected network we know about. The very existence of this traffic is evidence that *something* untoward is happening on the network. The sinkhole's job is to help us find out what that something is.

Before going much further, I should mention that this is not an invention of my own devising. I first read about routing sinkholes in Richard Bejtlich's excellent book *Extrusion Detection: Security Monitoring for Internal Intrusions* [1], and this implementation is based on his description in Chapter 5 of that book, "Layer 3 Network Access Control." However, it differs in two significant ways, which I believe make it worth describing separately. First, I use Linux on commodity hardware for the sinkhole platform, rather than the Cisco Bejtlich uses; second, I detail the setup of a NetFlow-based collection point for historical session and statistical data, whereas the book stops at enabling NetFlow on the interface and just uses IOS commands to show real-time NetFlow summaries. Sinkholes are also conceptually similar to the work CAIDA has done with Network Telescopes [2], which watch unused routable addresses for DoS traffic and backscatter.

Architectural Overview

So what, exactly, makes a sinkhole? There are really two parts to it, which I'll walk through setting up in the rest of the article. First, specialized routing rules need to be set up and propagated throughout your enterprise network to direct the known-bad traffic toward the sinkhole. Second, the sinkhole needs to generate statistics so that administrators can draw conclusions about the nature of the traffic the sinkhole sank.

The routing component has a couple of presuppositions about the network environment that ought to be explicitly stated. I'm going to assume your network:

- Uses some chunks of private address space, either in 10.0.0.0/8, 172.16.0.0/12, or 192.168.0.0/16.
- Uses some type of routing internally to direct traffic to various subnets or (minimally) toward a default gateway that has an Internet connection.
- Does not rely on your default gateway to get to private networks that are in use.

The neat hack that makes the sinkhole work in an environment that meets these criteria is this: For an IP router, the "best" path to a particular destination is the one with the longest prefix. In other words, routers prefer the most specific route. The sinkhole takes advantage of this predilection by suggesting itself as a path to various destinations but with extremely unspecific prefixes, so the route to any legitimate internal network will be preferred by all the routers. Only fall-through traffic, whose destinations are not overridden by longer prefixes, hits the sinkhole. (Though I should note that all the sinkhole destinations are more specific than your default gateway, which suggests itself as the route to 0.0.0.0/0.0.0.0.) The sinkhole then accepts the traffic that uses it, records its existence, and promptly throws it away.

As for the statistics, there are several components that need to work together. The goal is to use the large body of open source tools that use Cisco's NetFlow packet format, which is widely used to summarize conversations of traffic that pass through a router. We'll put together a flow probe, which makes your Linux box pretend to be an expensive Cisco router by recording Layer 4 session data (source and destination IP, source and destination port, protocol). The probe then forwards that data in the form of NetFlow packets to the flow collector, which receives the traffic and records it for posterity. The collector also answers queries from a flow frontend, which administrators use to check up on sinkhole activity from the safety of their browsers or terminals.

Routing Implementation

Regardless of which internal routing protocol you use, the first step in asserting your sinkhole's presence on the network is to start advertising some routes to it. There are a couple of different ways to go about this: Either set up static routes on the sinkhole's upstream router and redistribute them into your dynamic routing protocol, or have the sinkhole participate directly in dynamic routing with something like Quagga [3]. I've just redistributed the static routes from the core router immediately upstream from the sinkhole, which is easier to set up but requires that the sinkhole also have static routes back to your enterprise's subnets, or it'll sink legitimate traffic too!

In this example, the sinkhole uses the two-IP subnet 10.10.15.16/30 for its link to the upstream router, the routers use OSPF among themselves, and there are two internal subnets we care about reaching from the sinkhole. The servers reside on 10.0.10.0/24 and the desktops live on 10.0.8.0/23, as shown:

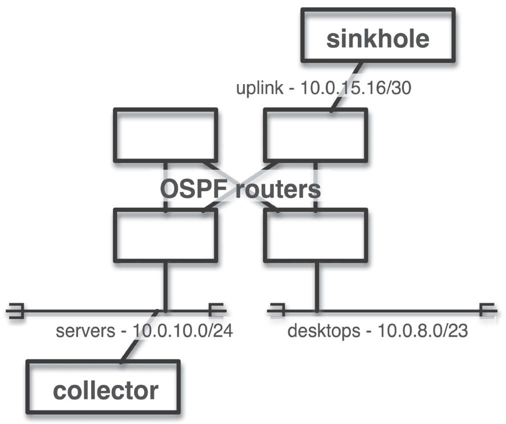


FIGURE 1: RELATIONSHIP OF THE INTERIOR ROUTERS AND DESKTOP NETWORK TO THE COLLECTOR AND SINKHOLE.

Assuming the router's end of the uplink is 10.0.15.18/30, we'd set up the RFC1918 routes to point to the sinkhole and then enable redistribution from statically configured routes into OSPF so that the rest of the OSPF-speaking routers will learn them:

```
ip route 10.0.0.0 255.0.0.0 10.0.15.17 10
ip route 172.16.0.0 255.240.0.0 10.0.15.17 10
ip route 192.168.0.0 255.255.0.0 10.0.15.17 10
router ospf
redistribution static
```

Here's a shell script snippet to configure the routes on the sinkhole box itself:

```
# known-good nets that should remain reachable
ip ro add 10.0.8.0/23 via 10.0.15.18
ip ro add 10.0.10.0/24 via 10.0.15.18
# repeat for any other networks you don't want to toss
# ...
# then add the 'blackhole' routes
ip route add blackhole 192.168.0.0/16
ip route add blackhole 172.16.0.0/12
ip route add blackhole 10.0.0.0/8
```

At this point you should be able to traceroute to nonexistent private addresses and see your packets disappear into the sinkhole. Running tcpdump on your sinkhole's Ethernet interface should provide some interesting—but probably somewhat frightening!—output. Make sure you save your changes, with `write mem` on the router and `ip route > /etc/sysconfig/network-scripts/route-eth0` (RHEL) or their equivalents. Now it's time to set up statistics collection.

Collection, Summarization, and Reporting

There are myriad different NetFlow-related resources on the Net, including several different collector implementations, plug-ins for popular packages such as NTop, and lots of Web and CLI tools for extracting the data. I'm going to focus on just a couple of packages: fprobe [4] and nfdump [5].

Setting up fprobe should be a straightforward configure; make; make install; the only difference between our sinkhole setup and a normal fprobe-running Linux router is that we want to avoid reporting on traffic that's directly to or from the sinkhole's IP. Fortunately, fprobe uses Berkeley Packet Filter (BPF) syntax such as tcpdump, so this is easy to accomplish. I've picked an unused high port on the NetFlow collection host (named "collector") and direct fprobe's output there with the last argument:

```
/usr/sbin/fprobe eth0 -a 10.0.15.17 -l 1 -f "not host 10.0.15.17"
collector:23001
```

The recipient of the NetFlow packets sent out by fprobe is nfcapd, part of the nfdump distribution. Again, this is well-behaved open-source software, so a quick download from SourceForge and configure; make; make install cycle later and we are ready to log some flow statistics. The most important parts of starting up a NetFlow collector are, first, to make sure you have plenty of free space on the disk partition you're writing to, and, second, to match the port you're listening on with the one to which your probe will be sending. In this case, we end up with a command line like the following:

```
/usr/bin/nfcapd -w -D -l /nsm/netflow/sinkhole -p 23001
```

For the impatient or untrusting, it's easy to verify that things are working as expected. Fire up tcpdump and watch for incoming traffic to port 23001. You should see inbound traffic from your probe to the collector, and after a few minutes the byte count of the log files in the directory specified in the -l option to nfcapd should increase.

Reporting, the final piece of our sinkhole implementation, comes courtesy of another tool from the nfdump distribution. This time it's nfdump itself, which is to NetFlow capture files what tcpdump is to pcap files. nfdump was installed along with nfcapd, so it should be ready to run. After a few hours of capture, the sinkhole has probably picked up some interesting traffic. Nfdump's flow aggregation abilities let us get a quick summary of the flows we've captured; the foremost useful aggregation for our sinkhole will be on source IP and destination port. This will help find hosts that are scanning for the same service on different hosts. A side effect of the aggregation is that the Dst IP Addr in the following output is represented as zeros—we'll drill down without aggregation in the second example to see the details of each individual flow matching the pcap-style expression host 10.0.10.15. The output can be further customized to show just the relevant fields using the -o fmt: . . . option string.

```
# nfdump -o "fmt:%pr %sap - %dap %pkt %byt %fl" -a -A srcip,dstport -R .
Proto    Src IP Addr:Port    Dst IP Addr:Port     Packets     Bytes      Flows
UDP      10.0.8.198:0  -     0.0.0.0:161          164         17384      42
UDP      10.0.10.15:0  -     0.0.0.0:137          520         46800      104
TCP      10.0.10.12:0  -     0.0.0.0:22           32572       1.9 M      15930
# nfdump -o "fmt:%pr %sap - %dap %pkt %byt"  -R . "host 10.0.10.15"
Proto    Src IP Addr:Port    Dst IP Addr:Port     Packets     Bytes
UDP      10.0.10.15:137  -   172.16.130.1:137     5           450
UDP      10.0.10.15:137  -   172.16.41.1:137      5           450
```

The capture, collection, and reporting infrastructure is up and running now, but an additional element is necessary to make the whole thing go: the application of human intelligence to analyze the data and investigate the results.

Analysis of the Results

In his book *The Tao of Network Security Monitoring* [6], Richard Bejtlich describes three categories of network traffic events:

> *Normal* traffic is anything that is expected to belong on an organization's network . . . *suspicious* traffic appears odd at first glance but causes no damage to corporate assets. . . . *Malicious* traffic is anything that could negative[ly] impact an organization's security posture. *(p. 361)*

By definition, none of the traffic that ends up at the sinkhole is normal, so our analysis will be aimed at separating the malicious traffic from the merely suspicious.

Remember from the network diagram that 10.0.10.0/24 is the server subnet and 10.0.8.0/23 are desktop subnets. In the nfdump output snippet from the previous section, there's a desktop that's doing repeated SNMP requests (UDP to port 161), a server with frequent NetBIOS name requests (UDP to port 137), and another server that is doing a huge amount of ssh scanning (TCP to port 22). The general procedure for investigation is to find out as much as possible about each of these flows from the nfdump data, and if there are still unresolved questions about the nature of the traffic, try to capture full packets from the sinkhole box itself. It's here where the advantage of a general-purpose Linux box as a sinkhole comes into play. Assuming we're proactive enough to notice suspicious traffic flows while they are still ongoing, it's easy to set up tcpdump captures on the sinkhole itself to get whole packets. This can be invaluable in determining what variant of a worm is sending out a particular scan, for instance.

In our example matrix, it turned out the ssh scans were due to an administrator's host-key gathering script that used erroneous DNS data to determine which hosts to scan. Fixing the script to limit its search by subnet rather than hostnames sped up its execution by a factor of 5. The NetBIOS requests were due to bad name registrations on the master browser, from hosts that were connected with a VPN client but registered the address on their remote LAN. Full-packet capture showed that the SNMP requests were to a branch of the host MIB used by HP printers to report status. The requests were generated automatically by a printer driver to which a user had pointed the user's home printer.

Conclusions and Future Work

Analyzing traffic that ends up in a routing sinkhole can be an enlightening experience. It turns out that there is a fair amount of background noise to various nonroutable address blocks: In addition to the three analyses shown here, I found that there was substantial bleed-through from Windows laptops that moved between our wired and wireless networks, which are on different RFC1918 blocks that are not reachable from each other. Not every packet that ends up at the sinkhole is evidence of malicious activity (unless you count NetBIOS Name Service as malicious!).

I regret that I did not have a sinkhole set up for the outbreaks of the Sasser and Welchia worms. It would have greatly aided in combating those worms, for two reasons. First, having historical data about which IP addresses were infected first and how the worm spread would have helped the labor-intensive mop-up efforts. Second, without a sinkhole to attract the bogus addresses generated by the worm scanners, hundreds of bogus packets per second were routed toward our firewall, which collapsed under the DoS load. With the sinkhole in place I'm almost looking forward to the next outbreak. Err . . . scratch that, actually.

Further refinements include setting up the very impressive NfSen Web-based frontend [7] to the nfdump collector; fine-tuning the list of networks that are blackholed, perhaps to include all IANA-reserved blocks from RFC3330; and setting up NetFlow exports on all the router interfaces that support it. Alerting and automated response are also on the horizon, but obviously a good amount of data ought to be collected to set appropriate thresholds before we start paging people out of bed or automatically shutting down switch ports.

It may not be able to swallow a Volkswagen, but the routing sinkhole is a very useful weapon in the fight against entropy and chaos on the enterprise network.

REFERENCES

[1] Richard Bejtlich, *Extrusion Detection: Security Monitoring for Internal Intrusions* (Addison Wesley Professional, Reading, MA, 2006).

[2] CAIDA Network Telescopes: http://www.caida.org/analysis/security/telescope/.

[3] Quagga: http://www.quagga.net/.

[4] Fprobe: http://fprobe.sf.net/.

[5] Nfdump: http://nfdump.sf.net/.

[6] Richard Bejtlich, *The Tao of Network Security Monitoring: Beyond Intrusion Detection* (Addison Wesley Professional, Reading, MA, 2005).

[7] NfSen: http://nfsen.sf.net/.

PETE HERZOG

the protocol historian

Pete is the Managing Director and co-founder of ISECOM, the Institute for Security and Open Methodologies, and is directly involved in all ISECOM projects. His main objective is to make security make sense.

pete@isecom.org

THE MANY PROTOCOLS CREATED, thriving, dying, and dead are quietly being documented in detail beyond that of the RFCs that introduced them. Accidental protocol curator and historian Dru Lavigne has been going beyond the technical details of Internet protocols since early 2001 to get the human side of invention.

Dru found that what started as a project to list common port numbers mapped to their associated applications for the appendix of the OSSTMM (www.osstmm.org), the standard methodology for security testing, would quickly evolve into more. Now as the OPRP (Open Protocol Resource Project; www.isecom.org/oprp/) it is one of the main projects for the open, nonprofit, security research community, ISECOM (www.isecom.org). When she volunteered in 2001 to assist in reviewing the OSSTMM, she couldn't help but notice that the mappings were woefully incomplete and, in her opinion, "not much of a help to anyone who would be interested in knowing which application was most likely associated with a port." Keep in mind, this was years before Fyodor introduced nmap -sV. She had already experienced her own frustrations in scouring the Internet looking for information on various ports. This seemed like the perfect opportunity to organize her previous research forays and make them publicly available so others could benefit as well. And since no one had previously shown any interest in this section of the OSSTMM, it became hers to do with as she could.

The actual goal of the OPRP is to provide a quick reference for those who are wondering what application may be running on a particular port. This has actually become easier since LAS (www.localareasecurity.com) created a Firefox plug-in to allow quick searches of the OPRP. The OPRP is meant to augment, not supplant, the official repository of registered port numbers (www.iana.org/assignments/port-numbers). This is the reason why Dru tries to contact the original protocol registrants for a description to include within the OPRP. The IANA has been registering port numbers for over two decades, and much has changed during that period: Products have come and gone and been EOL'd, and companies have been merged and purchased and perhaps swallowed by the dot-com bubble. The OPRP tries to determine whether each particular port is still in use today, and if so, in what products one can expect to find its usage.

The OPRP isn't meant to be a definitive source or a guarantee of what is running on a particular port. That is impossible, seeing that it is trivial to change the default port for almost any TCP/IP application. However, for the security tester or sysadmin reviewing firewall logs, it gives a starting point to see what is supposed to be there and if that is a likely application for the given environment.

The number of her protocol descriptions has now surpassed 1500. Currently there are approximately 5000 IANA registered protocols. That means she has curated descriptions for roughly one-third of the registered protocols. To put this in perspective, approximately 165 protocols have been registered thus far in 2006. Dru waits six months after those protocols are registered before contacting the registrants to give them time to get their protocols in use. "I quickly learned that it wasn't productive to contact registrants immediately, as protocols are often registered in the early stages of product development," Dru states. "I've found that a window of six to eight months after registration is most effective; by that time, the protocol is often actively in use and 'out in the wild.'"

Dru started with a simple guiding principle: who better to know whether a protocol is still in use and who better suited to provide a useful description than the person who registered the protocol? In her first round of contacts, she simply emailed all of the email addresses found in the IANA official list of registered ports. Since many of those addresses were long extinct, she saved all of the nondelivery messages for the next stage. It also did not help that only recently has IANA began dating registrations. However, a surprising number of email addresses did still work, and several hundred descriptions were received and input into the OPRP.

In the next stage she used her Google skills to see if she could find the remaining registrants. That garnered another 500 or so descriptions. Now she has two folders she works with: the nondelivery messages for newer but extinct email addresses and a folder for email that was successfully delivered but to which she didn't receive a response.

Stage three involved finding contact information for the companies that had registered protocols but for which the original registrant was unresponsive or could not be found. Although some may be unresponsive for trade-secret or corporate confidentiality reasons, another problem is sometimes that the protocol seems to have disappeared completely. "At this stage I'm still working out a plan for how to get descriptions for the protocols which perhaps didn't survive company mergers," she says. "For example, how many DEC and Compaq protocols are still being used in HP products? Or what of the protocols that were registered by companies since swallowed by IBM, Nortel, or Cisco?"

While some might think that a hobby or job as protocol historian may be dull, Dru finds it fascinating. She says she just naturally likes to organize information. She has a particular fondness for protocols, which is a natural extension of her need to know how things work. She is also fascinated by history, including the history of the Internet and TCP/IP. This is apparent in how the OPRP has started to become a repository of descriptions of historical protocols. IANA simply puts a "de-registered on date" note on file. "I would hate to see the name and history behind a de-registered protocol lost forever," Dru says. "I currently have 1185 delivered emails which I haven't received a response to and 118 nondeliverable emails." Her goal is to catalog them all.

When asked about herself, Dru says, "For those that are curious about my age: Neil Armstrong said, 'That's one small step for man but one giant leap for mankind' on my fourth birthday. At the time this was memorable simply because I was irritated that a bunch of boring grownups had preempted my favorite TV shows in order to talk endlessly about the same news clip. Since then, I've come to appreciate that seemingly small actions have ripple effects. This is part of what attracts me to open source. It is also a prime motivator for the many projects I am involved with, including the OPRP."

When she returned to school to study networking, she was bemused that most classmates found protocols to be so much boring theory. "I'm fascinated by anything that gives insight into how things work. I'm also fascinated by the stories behind how things came to be, so I was naturally drawn to RFCs and Internet history," she says.

She's just like any other busy person in IT who somewhere along the way became "known" within various open source communities. As to where she works (being a protocol historian doesn't pay a salary), Dru says it's not a short answer. She says every day is a bit different, with the threads of several ongoing works intertwining. She's been teaching IT certifications, most recently in Ottawa, since 1998 and is the acting chair of the BSD Certification Group (www.bsdcertification.org), a registered nonprofit with a goal of providing an IT certification for assessing the skills of BSD system administrators. She's also been a system administrator since 1996, starting with Novell and Microsoft systems and later integrating these with Linux and BSD systems. Since 2000, she has been writing technical documentation for various products, courseware and labs for various curricula, a column for O'Reilly (www.onlamp.com/pub/ct/15), and, most recently, another for IT Toolbox (blogs.ittoolbox.com/unix/bsd). She also attends and/or speaks at various technical conferences as well as meeting regularly with my local BUG (BSD User Group) and GOSLING (Get Open Source Logic INto Government). However, the OPRP project is something that she cares about deeply, and it puts her in touch with the movers and shakers of the information age.

"I've received everything from very terse replies indicating that the protocol is still in use but covered under an NDA to long essays on the details of the protocol," she explains. "Some responses could be considered a marketing slick, but that's fine as it still answers the fundamental questions, 'Is this protocol still in use, and what company/application(s) are using it?'"

Dru says she's been pleasantly surprised at the overwhelming positive response by registrants to the OPRP and has had only two belligerent responses since 2001. "I think this speaks to the professionalism shown by the registrants and the respect in the IT community for the ISECOM organization," she says. "I've also been humbled by receiving responses from very big names in the IT industry, the type of names that networking geeks such as myself considered to be demigods when it comes to the Internet and TCP/IP."

Some of these legendary responders include Bob Braden, whose research interests include end-to-end network protocols, especially in the transport and Internetwork layers (en.wikipedia.org/wiki/Bob_Braden); Joe Touch, whose interests include Internet protocols, network architecture, high-speed and low-latency nets, network device design, and experimental network analysis (www.isi.edu/touch/bio.html); Joe Pato, whose current research focus is on the security needs of collaborative communities, addressing both large-scale inter-enterprise models and the challenges of ubiquitous devices (www.hpl.hp.com/personal/Joe_Pato); and Linus Torvalds, who, she says, responded within fifteen minutes on an Easter Sunday.

For Dru to say which protocol has the best story is like asking a kid in a candy store what her favorite candy is. It really is hard for her to say. Since every protocol has a story, oftentimes a story of genius and hopes and dreams of real people trying to push forward the information age to an even greater age of ubiquity and enlightenment. For instance, there is the port number that represents the birthdate of a developer's daughter. Another port number represents the date of a wedding anniversary. Then there are protocols that were as ubiquitous as HTTP is today but that have since become extinct, including protocols that represent the excitement of the dot-com era but that never saw a single shipped product. TCP 1456 was registered for use by OpenMind, a groupware application published by DCA and then Attachmate. Even though it won Product of the Year in 1995, it is no longer in production or commercially available. TCP/UDP 1305 was originally registered as pe-mike, but the company was bought out and no products were ever released to a customer that used this protocol.

This is exactly what Dru likes best about being a protocol curator: the human drama behind the invention. Those she finds most notable are as follows:

"MilliCent used to use ports 1180, 2180 and 3180," says one email response Dru received. "When it existed, [it used] TCP. Now it doesn't exist and it uses neither. This MilliCent protocol was originally created for DEC, then Compaq and now HP. The project is now defunct, but it was great."

The response regarding port 1989 says, "Originally developed by the University of Sydney and Message Handling Systems Py Ltd, Australia, and first sold in 1989. MHSnet has been used to build message networks where the links range from poor quality up to Internet quality. It was used by the Australian Govt Department of Foreign Affairs and Trade to build a message network between embassies and posts. It has been used to build many private networks but was also the backbone of an academic network (ACSnet) in the early days of networking in Australia."

Another responder wrote, "I think 585 was an administrative error. I believe it had been originally ear-marked for IMAP-SSL, but that turned out to be 993. I either never knew or have long forgotten what caused that situation, and alas, we can no longer ask Jon (Postel). In any case, if 585 is alive, I don't know anything about it."

The response regarding ports 309, 709, and 710 says, "When I was employed at Entrust, Inc. (1994–2001), I registered those ports (which are all based on my daughter's birth statistics, time, date, and weight, respectively). . . . Note that 709 is deprecated in favour of 829 (PKIX CA/RA; 829 is the wedding anniversary of the fellow who registered that one, Carlisle Adams, the CA in CAST)."

The protocol Gopher, which was the precursor to the Web on port 70, had been hugely popular until it got eclipsed by the Web. Dru received the following response in regards to this behemoth that has nearly shrunk to nothing: "Internet Gopher popularized the notion of distributed information systems before the World Wide Web. Client and server software is available for most popular platforms. Although the original Gopher developers at the University of Minnesota are no longer actively working on this project, other groups are. For instance see http://gofish.sourceforge.net/

and http://gopher.quux.org:70/devel/gopher/pygopherd and the usenet newsgroup comp.infosystems.gopher."

Going back to the backbone of ARPANET, the precursor to the Internet, Dru received: "51 was implemented on the BBN IMPs, which formed the backbone of the original ARPANET and later MILNET (a.k.a. Defense Data Network, DDN). It was used to add a layer of indirection to ARPANET addresses, which were originally tied to the physical ports on each particular IMP (like IMP 18, port 4). Logical addresses made it possible to keep the same ARPANET address without being tied to one particular physical port. However, the use of IP made this moot, since once IP was used, packets were sent to a particular IP address, rather than a particular ARPANET address, and an IP address resolution protocol was used to do the mapping. And of course, the shutdown of the ARPANET made it REALLY moot. However, that's not to say that there's not an old military network somewhere still running IMPs (although I REALLY doubt it)."

Some protocols are more specific and more rare. One such protocol is the one registered for port 91. The responder writes, "The port assignment was for a protocol peculiar to equipment and arrangements of equipment use in the MIT Lab for Computer Science over 20 years ago. All the equipment is now long gone. As far as I know, the protocol has not been reassigned, but I have not tracked such things. As I recall, we used it for TCP, not UDP, but it was a long time ago. The tool in question also handled the Chaos net protocol, a completely different network that I think never propagated elsewhere."

The final response Dru provides is of a dot-com invention that, although perhaps superior to what is now commonly implemented, never got released publicly. The registrant of port 1228 writes, "Florence was a proof-of-concept remote method invocation facility with application-hinted client-side caching designed to improve latency in hierarchical arrangement of nodes. It was hastily rushed into production by a dot-com whose time was running out. I was one of two engineers in charge of its design and implementation. There was one application, a business-to-business exchange running atop Florence, that made it to the demo stage, but as iventurelab.com is now thoroughly and completely out of business and—to my knowledge—nobody purchased the IP, I doubt if any of the source code implementing it actually still exists. The idea was sound, and I've been meaning to re-implement the concept atop a more portable software infrastructure and release it as open source software, using this assigned port, but frankly, this is #3 on my list even of open source priorities, and implementing yet another remote method invocation facility gratuitously incompatible with SOAP, while worth it for the performance gain of protocol-supported response caching, will probably not get anyone too excited about using it."

Other protocols of note from the OPRP are shown in the table on the next page.

Number	*Transport*	*Application*	*RFC/Vendor's URL/MS KB Article*	*Description*
47	TCP	deprecated		Originally registered as NI FTP, the Network Independent File Transfer Protocol, known as "Blue Book." It operated over many years in the UK academic community, primarily over x.25.
51	NPC	deprecated	www.ietf.org/rfc/rfc851.txt	Was used by IMP Logical Address Maintenance on the original BBN ARPANET. It was used to map ARPANET addresses to physical ports on an IMP. This functionality was superseded by TCP/IP.
61	TCP	deprecated		Originally registered as NI Mail and also known as "Grey Book." It was a mail protocol based on RFC 822, operating over NIFTP (see port 47).
81	TCP UDP	deprecated		Originally registered as HOSTS2 Name Server; its registered use seems to have been long deprecated.
96	TCP UDP	DIXIE	http://www.ietf.org/rfc/rfc1249 .txt	Was used by DIXIE, which has since been replaced by LDAP on port 389.
105	TCP	deprecated	www.ietf.org/rfc/ rfc2378.txt	Was the CCSO Name Server, the backend of the Ph function of Eudora. It has since been replaced by LDAP.
402	TCP UDP	deprecated	http://web.archive.org/web/ 19991009142042/www-genie .mrrl.lut.ac.uk/interfaces.html	Registered for the genie protocol, but unused since 1998.
692	TCP	deprecated	http://www.hyperwave.com	Was used for the Distributed Interactive Services (DIS) protocol for core-level access to Hyperwave's backend server architecture.
1228	TCP	deprecated		Originally registered for Florence, a proof-of-concept remote method invocation facility with application-hinted client-side caching designed to improve latency in hierarchical arrangement of nodes. It never shipped, as a result of a business failure.
1427	UDP	deprecated		Was used by a private, experimental protocol developed as part of DARPA-funded research.

Like any other open project, though, the OPRP does have its detractors. Some registrants disagree that the OPRP should include nonregistered usage. Dru's philosophy is that there should be an entry for what is likely to run on a port: for example, well-known worms or Trojans as well as usage by common, though unregistered, applications. She feels this is what will be most useful to an administrator.

However, since worms and Trojans are discovered more quickly than she has time to research, anyone is welcome to add an entry to the OPRP. It is an open database, after all. She reviews these entries before inclusion and her deciding factor for permanent entry is based on the reliability of the information. Something as simple as an URL pointing to supporting documentation, however, can be considered reliable information.

However, Dru does keep tight control over what is entered and she does review it all personally. For future would-be researchers and curators, she advises, "Some registrants have changed the name of the protocol or the company since the original IANA registration and have not updated their info with IANA. If you see a description for a registered protocol in the OPRP, which isn't a Trojan or marked as for unregistered use, that description and name change is from the registrant. Please don't try to add an entry with the outdated IANA information, as it won't be included in the OPRP."

It's an ambitious project. When asked when she thinks it will be finished, she says, "Probably never." As long as IANA continues to register protocols, entries will need to be updated. The OPRP needs to have at least a description for every registered protocol. With that, she comments, "I think any article on protocols should make a reference to Postel (http://www.livinginternet.com/i/iw_mgmt_iana.htm). Postel's contributions to the IANA and RFCs are deeply appreciated by the networking community and a conversation on ports isn't complete without paying respect to him."

As far as anyone can tell, Dru's ambition and busy schedule have already tagged her as a remarkable person, especially within the open source and Internet communities. Her dedication and contribution as a protocol historian are nothing short of amazing.

Dru's final comment to those out there is this: "If you have registered a protocol but haven't received the OPRP questionnaire, email me to request a copy. If your registered protocol needs an updated description, email me the details. If you have contacts for a large corporation's intellectual property department and want to sort out what registered protocols are or aren't still in use, just email me."

You can contact Dru easily at dru@isecom.org. The OPRP is available at www.isecom.org/oprp/.

DAVID BLANK-EDELMAN

practical Perl tools: tie me up, tie me down (part 1)

David N. Blank-Edelman is the Director of Technology at the Northeastern University College of Computer and Information Science and the author of the book *Perl for System Administration* (O'Reilly, 2000). He has spent the past 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 '05 conference and is one of the LISA '06 Invited Talks co-chairs.

dnb@ccs.neu.edu

WITH APOLOGIES TO PEDRO ALMODÓVAR and to the kink community readers who will be disappointed when they realize this isn't the column they were hoping for, I'd like to dive deeply into one of my favorite Perl features: the tie(). After reading a bit on this topic, I suspect you'll either have your jaw on the floor or will have slapped your forehead in disgust at those wacky Perl people (what will they think of next?).

Let's start with some basic background. Once upon a time, relatively early in Perl's history, there existed a function called dbmopen(). dbmopen() made a hash variable special by tying it to an on-disk database backend. Usually a hash variable keeps its keys and values in memory, but when dbmopen'd the data would actually live in an {n,s,g}dbm or Berkeley DB database on disk. Accessing the data would cause a transparent fetch or store of the data from or to the database without any effort on the programmer's part.

This meant three great things:

- Much larger data sets could be used because all of the data didn't all have to live in memory at once.
- It was trivial to have the data persist after the program had quit.
- All you needed to know was the usual hash semantics; no advanced fiddling or faddling was necessary.

Zoom forward in time to Perl 5. Perl 5 added a tie() operator to the language which allowed for the same sort of magic to be applied to other kinds of variables. It also abstracted the mechanism even further, such that you could tie more than just a database to a variable. What sorts of things could be used? That's where the jaw dropping starts. This two-part column will give you a taste of the amazing things that have been done with this simple concept and how to use it for your own ideas.

Before we get to the fireworks, I feel compelled to mention that not everyone is enamored with tie(). For instance, in *Perl Best Practices*, the Damian Conway book mentioned in this column before, he says:

> Don't tie variables or filehandles. . . . Tied variables make any code that uses them less maintainable, because they make normal variable operations behave in unexpected, non-standard ways.

Using my most cogent debate skills honed in elementary school, my rebuttal is, “Yes, I definitely agree with this sentiment. But I don’t care.”

Just to be clear, I’m all for maintainable Perl code (and have actually written some in my time), but I think trade-off can be worth it. This particular abstraction is too powerful to give up just because it has the potential to impact maintainability. An even more compelling argument for me is the amount of creativity this particular feature of the language has unleashed in the Perl community.

In the interests of self-disclosure, I should mention that these may be the words of someone addicted to the power of tie(), so you should make up your own mind about what is important to you before you go down this path. In next issue’s column on this topic we’ll talk about how to implement our own tie()-based code. At that time we’ll also discuss the recommended alternatives to tie(), just so you have all of the tools available when making that decision.

So let’s jump on the bus and take the first part of a whirlwind tour of some of the more interesting things I’ve seen done with tie().

More Complex Backends

Early on we mentioned that tie() had its origins in a mechanism for storing and retrieving values from a simple database. An easy evolutionary step from that notion is the ability to retrieve information from a more complex backend. For example, let’s say we had a table in a relational database of network hosts that looked like this:

ether (primary key)	Name	ipaddr
00:16:cb:b7:c8:81	Omphaloskepsis	192.168.0.1
00:04:E2:07:AC:17	Dave	192.168.0.4
00:0C:F2:24:9A:45	Otherdave	192.168.0.7

With Tie::DBI, we could write the following:

```
use Tie::DBI;

tie my %hosts, ‘Tie::DBI’, {

        table   => ‘hosts’,
        key     => ‘ether’,
        CLOBBER => 1   # allows read-write access to database
};
```

Now we have a hash called %hosts whose keys correspond to the primary key column (ether) of the database. This means that:

```
print join (“\n”,keys %hosts);
```

will print a list of the Ethernet addresses stored in the database. If we use any of those keys to retrieve a value from %hosts, we get back a reference to an anonymous hash containing that record’s information. To see an example, we could add these lines:

```
use Data::Dumper;
print Dumper($hosts{‘00:04:E2:07:AC:17’});
```

and this would yield:

```
$VAR1 = {
        'ether' => '00:04:E2:07:AC:17',
        'name' => 'dave',
        'ipaddr' => '192.168.0.4'
};
```

To access an individual field, the syntax is your standard hash of a hash syntax:

```
print $hosts{'00:04:E2:07:AC:17'}->{'name'},"\n";
                    # arrow not strictly necessary
```

We could change the data in the database (providing the CLOBBER flag is set at tie() time) with a plain ol' assignment operation:

```
$hosts{'00:16:cb:b7:c8:81'}->{'name'} = "shouldhavebeendave";
```

If we put standard databases aside for a moment, it is worth noting that people have applied the same principle to other less obvious backends, for instance:

```
use Tie::DNS;

tie my %resolve,'Tie::DNS';
print $dns{'example.com'},"\n"; # prints "192.0.34.166"
```

In the second part of this series we'll go over the steps necessary to use another backend of your choosing.

Transformers

No, I'm not talking about the robots from Hasbro that turn into trucks. This is a class of modules where the data stored in a tie()'d variable is transformed during the retrieval of that value. Here's a simple example:

```
use Tie::Comma;                  # loads a magic tied hash called %comma

my $a = "12345678";
print "$comma{$a}\n";            # prints 12,345,678
print "$comma{$a,2}\n";          # prints 12,345,678.00
```

Here we've created a magical hash that transforms the format of a value simply by looking that value up in the hash. Yes, this looks a bit like a fancy printf() statement, but I'm just trying to limber you up. Shortly we'll get into some very strange territory around variable retrieval, so I want you prepared for when things start to deviate from the usual understanding of reality.

Fancy Lookups

Normally we don't think very hard about the actual retrieval process with hashes. We've always been taught that hashes store a set of key/value pairs. To retrieve a certain value, you need to present the hash with the unique key associated with that value (hence the term *associative arrays*). But what if we could play around a bit with this assumption?

What if, for instance, we could make those lookups be case-insensitive? Imagine you had a hash with the following in it:

```
my %banks = ('sasquatch trust' => 3000);
```

To get the value from this hash for that bank, you have to say $banks{'sasquatch trust'}; $banks{'Sasquatch Trust'} doesn't work. However, if you use the Tie::CPHash module, it will:

```
use Tie::CPHash;

tie my %banks, 'Tie::CPHash';
%banks = ('sasquatch trust' => 3000);

print $banks{'sasquatch trust'},"\n";         #prints 3000
print $banks{'Sasquatch Trust'},"\n";         #prints 3000
```

You could even use a key based on a trendy, creative capitalization scheme:

```
print $banks{'saSqUatCh Trust'},"\n"; prints 3000
```

Tie::CPHash retains the original capitalization of the key when first stored in the hash and makes that information available if you need it.

Case-insensitive lookup is peanuts compared to our next example. What if you could store a date range for a key? Tie::RangeHash lets you do that (and more):

```
use Tie::RangeHash;

tie my %semester,'Tie::RangeHash';
$semester{'2006-09-06,2006-12-15'} = 'Fall semester';
$semester{'2007-01-08,2007-04-27'} = 'Spring semester';
$semester{'2007-05-08,2007-08-21'} = 'Summer semester';

print $semester{'2007-04-16'},"\n";  # prints 'Spring semester'
print $semester{'2007-06-01'},"\n";  # prints 'Summer semester'
```

If date and other ranges aren't powerful enough for you, how about regular-expression lookups? Tie::RegexpHash stores regular expressions as hash keys:

```
use Tie::RegexpHash;

tie my %rhash, 'Tie::RegexpHash';
$rhash{qr/[sS]asquatch/}   = "Sasquatch Trust and Savings";
$rhash{qr/\d{5}(-\d{4})?/} = 'US zip code';
$rhash{qr/^bucky/}         = 'invented by Buckminster Fuller';

print $rhash{'Sasquatch Bank'},"\n";
                    # prints "Sasquatch Trust and Savings"
print $rhash{'02114'},"\n";        # prints "US zip code"
print $rhash{'02114-2132'},"\n";  # prints "US zip code"
print $rhash{'buckyball'},"\n";    # prints "invented by Buckminster Fuller"
```

In this code we've defined keys based on some simple regular expressions (which could have been arbitrarily complex) instead of using your standard scalar keys. When presented with a key to look up in the hash, if a regular expression matched, the corresponding value is returned. Looking up a key that didn't match against any regular expression previously stored in the hash will return undef, just as expected.

By now I'm hoping that your creative juices are flowing. By perverting the usual lookup conventions of a hash we can unlock some pretty interesting programming possibilities. Let me show you one more example of this and then we'll get polymorphously perverse with our Perl variables. One of my favorite examples for fancy lookup modules based on tie() is Tie::NetAddr::IP. In this case, instead of providing regular expressions as keys, you instead provide IP range definitions (in CIDR notation):

```
use Tie::NetAddr::IP;

tie my %network, 'Tie::NetAddr::IP';
# load a list of our IP networks
$network{'192.168.0.0/24'} = 'server net';
```

```
$network{"192.168.1.0/24"} = "UNIX net";
$network{"192.168.2.0/24"} = "PC network";
```

With those definitions in place, we can now look up addresses. For example, if you found a machine had the address 192.168.0.24 and wanted to know what network it was on, it would be as simple as this:

```
print $network{'192.168.0.24'},"\n";  # prints "server net"
```

Magic Return Values

Your grip on (Perl) reality should be a little looser by now, so I trust you won't be too put out if I show you a couple of examples where you get more out of a scalar than you'd ordinarily expect:

```
use Tie::Scalar::Timestamp;

tie my $timestamp, 'Tie::Scalar::Timestamp';
print $timestamp,"\n";  # prints the current timestamp in
                        # (by default) ISO8601 format
```

With Tie::Scalar::Timestamp, you are creating a magic timestamp scalar that returns the current timestamp (in a format of your choosing) each time you access this variable. Sure, you could write a subroutine to do this, but that subroutine won't interpolate into strings as nicely as a Tie::Scalar::Timestamp tied variable.

A little more interesting magic can be found in the various modules that create scalars that can return a value from a predefined set of values each time you retrieve the contents. Tie::Cycle and Tie::Scalar::RingBuffer work this way. There are also bivalue modules, such as Tie::FlipFlop and Tie::Toggle, that switch between two possible outputs each time the variable is accessed. Here's one of these modules in action:

```
use Tie::Cycle;

tie my $round, 'Tie::Cycle', [qw( row row your boat )];

# each time we access $round, it returns the _next_ value in the list
print $round,"\n"; # prints "row"
print $round,"\n"; # prints "row"
print $round,"\n"; # prints "your"
print $round,"\n"; # prints "boat"
print $round,"\n"; # prints "row"
print $round,"\n"; # prints "row"
```

This is obviously a contrived example (unless you do a lot of campfire computing), but you can see how this might be useful in those situations where you need to repeatedly cycle over a set of values.

Wish Lists (warning: cliff hanger!)

In the final section of this part of the series we are going to further blur your notion of how hashes and other variables ""should"" work. To do that, let's go wild and assemble a wish list of things we wish a hash could do:

- Have elements that would automatically expire after a certain amount of time had elapsed.
- Keep a history of all changes made to it over time.
- Restrict the updates that are possible.
- Always keep track of the top N values or the rank of the values stored in it.

- Always return the keys in a sorted order based on the values in that hash.
- Transparently encrypt and decrypt itself.
- Easily store and retrieve multiple values per key.

As you have probably guessed, all of these things and more are possible thanks to tie()-based modules, and that's just using hashes. Rather than rush through the steps necessary to make this magic happen, we're going to hold off until next time to learn how to fulfill all of these wishes. Also, in the next part of this series, we'll look into how to actually write our own tie()-based module [and its tie()-less equivalent for those of you disenchanted with tie()]. If you get desperate before the next column to learn how this is done, please see the modules on CPAN in the Tie:: namespace plus the perltie man page that ships with Perl. Until then, take care, and I'll see you next time.

ROBERT HASKINS

ISPadmin: traffic shaping

Robert Haskins has been a UNIX system administrator since graduating from the University of Maine with a B.A. in computer science. Robert is employed by Shentel, a fast-growing network services provider based in Edinburg, Virginia. He is lead author of *Slamming Spam: A Guide for System Administrators* (Addison-Wesley, 2005).

rhaskins@usenix.org

IN THIS EDITION OF ISPADMIN, I LOOK at the area commonly referred to as "traffic shaping." Traffic shaping is the process by which network operators manage the somewhat random flow of packets to and from their networks to achieve the desired flow characteristics. Synonyms for "traffic shaping" include "packet shaping," "bandwidth limiting," "rate limiting," and "bandwidth management." These terms (among others) are commonly used in this article and elsewhere.

Some of the traffic flow characteristics a network operator might like to achieve would include the following:

- Adhering to customer service level agreements (SLAs)
- Ensuring fair use of egress bandwidth (commonly referred to as Internet traffic)
- Managing egress bandwidth links so as to not exceed committed and/or purchased data rates and (potentially) associated monetary charges
- Guaranteeing per application level minimum or maximum rates of use

In the enterprise space, traffic shaping is also used, though for different reasons. For example, bandwidth limiting (traffic shaping) can be used in conjunction with a corporate firewall to control access to time-sensitive applications such as Citrix and Remote Desktop. However, an enterprise is unlikely to be using traffic shaping to control peer-to-peer traffic such as BitTorrent, as those applications will normally be banned altogether.

Background

Bandwidth limiting is normally deployed on networks that don't have some other means OF controlLING them. For example, traffic shaping is not normally required on egress networks that are servicing dial-up networks, owing to the slow nature of analog modems. DSL, cable modem customer premises equipment, and/or provider-side equipment normally have a simpler form of traffic shaping built in. As a result, bandwidth limiting is normally only necessary on Ethernet and similar access technologies such as wireless.

Although some service-provider-class wireless equipment does have built-in policy control, most if not all consumer-grade wireless access devices (which might be in use on university networks) do not. If consumer-grade wireless access devices are in use on the network, packet shaping can be very helpful in limiting abuse.

Wired Ethernet access is often utilized in what are often called MDUs, short for "Multiple Dwelling Units." MDU is telco-speak for apartment and condominium complexes, dormitories, and similar types of building structures.

The boundary between policy enforcement (see my June 2006 ISPadmin column) and bandwidth shaping is a little blurry. The biggest difference would be the level of flexibility bandwidth shaping allows when compared to policy enforcement mechanisms. Typical policy enforcement engines are used for provisioning user connections one by one, but bandwidth-shaping policies can be defined at an aggregate level for all connections. Bandwidth-shaping systems can be controlled by a global policy enforcement engine such as Broadhop. This integration gives the network operator the most flexibility, as IP and bandwidth policy can be set at multiple points on the network.

Although bandwidth shaping cannot help directly with denial-of-service attacks and malware activity, IT can be useful in helping to determine the perpetrator(s), either internal or external. Many solutions allow the network operator to sort connections via bandwidth, flows, and failed flows. In the case of a commercial appliance solution, a GUI facilitates quick access to this information.

What Can Be Shaped, Exactly?

First, we need a word or two about bandwidth limits. It's important to note that bandwidth limits can be "hard" or "soft." In the case of hard limits, the user is strictly limited when the hard limit is reached. With soft limits, so long as another higher priority request isn't outstanding, the user can exceed the preset limit. The types of bandwidth controls can be broken down into three categories:

- Per-user
- Per-application
- Priority-based

Per-application can also be thought of as per-port. For example, email (SMTP) is synonymous with TCP port 25.

PER-USER

It is often desirable to limit each user's bandwidth usage, by itself as well as in conjunction with other applications and/or ports. Also, it would be very useful to have a default per-user bandwidth profile that could be customized if and when necessary.

PER-APPLICATION

One of the biggest bandwidth hogs is peer-to-peer traffic such as BitTorrent or Kazaa. Many commercial bandwidth shapers can identify and limit such traffic based upon the protocol, port, and connection characteristics of the traffic in question. For example, Voice over Internet Protocol (VoIP) traffic can have dedicated bandwidth, so users don't experience voice dropouts and other annoying behavior because of lack of bandwidth.

PRIORITY-BASED

A very strict configuration employed by a university might be to allow peer-to-peer traffic *only* when there is idle bandwidth. In this way, the network operator can allow the "most important" traffic to pass, and the "less important" connections must fight for a smaller share of the connection.

Implementations

There are several ways bandwidth shaping can be implemented, including:

- Routers
- Open source solutions
- Commercial "appliance" devices

Each of these will be examined in the following sections.

ROUTERS

The simplest and potentially cheapest way to implement bandwidth shaping is to activate policy routing on existing routers. Policy routing enables changes to routing tables via general terms. For example, all SMTP traffic could be policy routed to a spam-washing device in your network via policy routing. Both Juniper [1] and Cisco [2] have this functionality built into the core software routing engines. However, the downsides to implementing bandwidth shaping via policy routing are rather significant:

- Routers are not designed or optimized to be packet shapers.
- Implementing packet shaping in routers will likely cause performance degradation.
- Routers will have less packet-shaping functionality than a dedicated bandwidth-limiting device.

One simple way routers could be used to shape traffic via policy routing would be to route "bandwidth hogs" to a slower egress connection. For example, assume a college has two separate egress links to the Internet, one being an DS3 (45 Mbps) and one a T1 (1.5 Mbps). All users and IPs would be initially routed out the DS3 connection and their usage tracked. If a particular user (or IP) exceeded a preset threshold, then that user or IP would be routed out the slower connection until such time as its usage dropped. At that point (or when the user called into the support center and was informed of the reason for the slower connection), the user could be routed back out the faster connection.

The bottom line is that routers are rarely used as bandwidth-management devices except in simple network designs and the most lightly loaded networks. However, they can be used to augment other bandwidth-control mechanisms.

OPEN SOURCE SOLUTIONS

Bandwidth management can be implemented by using one or more open source components. Among the approaches that can be used to achieve this are:

- Squid, iptables, and CBQ (class-based queuing) [3]
- Iproute2 plus iptables [4]
- Snort, iptables/ipchains, and CBQ

One benefit of any open source solution is that you have the ability to tailor it to exacting requirements. A build-it-yourself solution is not for the faint of heart, as it requires deep knowledge of firewalls and routing, as well as the interactions between the two. Also, home-grown solutions require time and testing to be successfully utilized. However, a do-it-yourself deployment may be preferable in certain cases.

COMMERCIAL APPLIANCES

Arguably the most well known commercial bandwidth-controlling device is the Packeteer PacketShaper [5]. According to the company's Web site, the device can control over 500 application types. Packeteer manufactures a number of models, ranging from a low end of 6,000 flows and 2 Mbps of traffic to a high end of 1,260,000 flows and 1 Gbps of traffic. Other manufacturers of commercial bandwidth shapers include XRoads Networks [6] and Cymphonics [7].

These devices are usually deployed at the egress points of MDU (and other Ethernet-based customer) networks, so that all customer traffic goes through the device. This enables the shaper device to control all traffic to and from the network in question.

Issues

One big issue with bandwidth-management devices is that an appliance device failure could cause the attached MDU network to fail completely. Packeteer has engineered their copper-based devices to automatically pass all traffic if the device should fail. With fiber-optic-media network connections, a fiber bypass device is required. This device would route the photons around the failed appliance automatically.

Another issue with bandwidth-shaping devices surrounds virusES and worms. It is difficult for the bandwidth-shaping device to discern between legitimate user traffic and malware traffic.

Summary

Bandwidth-limiting devices are common features of networks where shared access causes contention for limited Internet egress. IP routing policy systems can be used in conjunction with packet-shaping devices, though they aren't usually used in place of such systems. Bandwidth can be limited on a per-user, per-application, and priority basis and/or a combination of these methods. Some approaches used to shape bandwidth include routers via routing policy, collections of open-source components, and commercial appliances. Problems with bandwidth shaping include planning for device failure and the inability of the device to discern "real" traffic from virus and worm traffic.

I wish to thank Pete Carey and Rik Farrow for their help with this article.

REFERENCES AND FURTHER READING

[1] Juniper QoS chapter in JUNOSe 7.1 documentation:

http://www.juniper.net/techpubs/software/erx/junose71/swconfig-qos /download/parameters-config.pdf.

[2] Cisco Bandwidth Management and Queuing paper: http://www.cisco.com/en/US/tech/tk331/tk336/technologies_design _guide09186a0080237a48.shtml.

[3] Bandwidth Limiting HOWTO: http://www.tldp.org/HOWTO /Bandwidth-Limiting-HOWTO/index.html.

[4] Linux Advanced Routing & Traffic Control HOWTO: http://lartc.org/howto/.

[5] Packeteer Packetshaper: http://www.packeteer.com/products /packetshaper/.

[6] XRoads Networks traffic shaping products: http://www.xroadsnetworks .com/products/EdgeXL.xos.

[7] Cymphonix: http://www.cymphonix.com/.

Basics of traffic shaping: http://cc.uoregon.edu/cnews/winter2002/traffic.html.

SecurityFocus article on traffic shaping: http://www.securityfocus.com /infocus/1285.

Squid proxy home: http://www.squid-cache.org/.

Iptables home: http://www.netfilter.org/.

CBQ.init traffic-based queuing script implementation: https://sourceforge.net/projects/cbqinit.

Iproute2: http://www.policyrouting.org/iproute2.doc.html.

Ipchains: http://people.netfilter.org/~rusty/ipchains/.

HEISON CHAK

choosing IP phones

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

AS VOIP GAINS POPULARITY, MORE businesses and home users are looking for innovative ways to incorporate this technology into their everyday lives to enhance communications with colleagueS, business associates, friends, and families. In most cases, the piece of equipment and technology they choose for VoIP enablement will dictate the prospect they have with VoIP.

What is a good IP phone? Is there a single VoIP client that runs on Windows, Linux, and Mac? What features should I look for when sourcing IP phones? These are common questions asked by IT managers and technology experts who are involved in the implementation of a VoIP platform. The quick answer is, "It all depends . . ."

Soft Phone

Whether you are looking for a soft phone or a hard phone, gathering requirements from different user groups is essential to successful deployment of chosen technology. Here are examples of what a soft phone should do:

- Run on multiple platforms, including Windows, Mac, and Linux
- Have the same interface for configuration across all supported platforms
- Work well behind NAT gateways
- Support noise and echo cancellation
- Utilize minimum bandwidth and offer various CODECs
- Support out-of-band Dual Tone MultiFrequency (DTMF; better known as Touch Tone)
- Display caller ID

X-Lite (by CounterPath, formerly Xten) is one of those software-based SIP phones with multi-platform support, running on Windows, Mac, and Linux. The next versions up, X-Pro and eyeBeam, support even more CODECs and have video conferencing capability. Most will find X-Lite or eyeBeam sufficient to connect local and remote users to the company's IP PBX for accessing voicemail and for normal voice calls.

The protocol-savvy may prefer an implementation based on Inter-Asterisk eXchange (IAX); not only can it guarantee more effective use of bandwidth, but it can eliminate confusion regarding out-of-band DTMF, a problem to which some SIP setups are prone. Some IAX soft phones offer more choices of CODEC over their SIP counterparts. An example is IAXComm, an open-source application that runs on Windows, Mac, and Linux. Of course, this will require an IP PBX that supports the IAX protocol, Asterisk.

A soft phone setup isn't complete without a good headset, especially on laptops. Acoustic echo, whereby voice received leaks into the stream of audio sent, is a common problem found with laptop users. As a result, the distant party can, irritatingly, hear his or her own voice. The effect often worsens with higher amplitude of the echo or with longer delay. By physically separating the microphone and speaker, such problems can be prevented. Since most laptops today are shipped with Bluetooth, using a soft phone with such technology may not be a bad choice, if your budget allows.

Hard Phone

With traditional PBXs, it is common for vendors to employ proprietary signaling on handsets and phone switches. Investment in handset equipment from any particular vendor will often lead to costly upgrades and maintenance of the system. With IP PBX, you now have the option of managing your own telephony equipment and perhaps have significant influence on what features such equipment should provide.

Basic IP phones residing on employee desktops, in cafeterias, in reception areas, or at loading docks should have the following essential features:

- A multi-line LCD display
- Caller ID information and call status
- The ability to switch multiple lines
- Call hold, retrieve, and transfer features
- Speed dialing and redialing from call history
- Volume adjustment

The Grandstream GXP2000, Snom 190, and Siemen OptiPoint 410 are examples of basic IP phones; they range from $100 to $400.

Power users, managers, and executives, with more demanding telecommunication needs, often opt for sophisticated models, which include features such as these:

- Full graphic display rather than multiline LCD
- Color displays and backlit buttons
- Built-in full-duplex speaker
- Auxiliary port for headset support
- Message waiting indicator (MWI) (some basic phones also have this feature)
- 10/100 Mbps switched PC port
- Compliance with power over Ethernet (802.3af)
- Web browser
- Voice encryption
- Support for various CODECs
- Conference capability

Cisco's 7960 and 7970 are phones in this class. High-end phones typically cost $400 and up. The Cisco 7914 module expands conferencing and transfer capability of the Cisco 7900 series phone, making it ideal for console operators who need to set up bridges, route incoming calls, and forward calls. The alternative is to use the vendor's softphone app running as a PC interface.

To ensure that your investment in hardware is protected, firmware images are generally made available by vendors to implement new features and to provide bug fixes (alas, they sometimes introduce new bugs). These firmware images are usually loaded at boot time from a TFTP or HTTP server, along with provisioning parameters. IP phones are usually provisioned in two steps, first with some generic parameters, such as server or proxy addresses, and second with phone- or account-specific parameters. After all, each employee's desk phone should have a unique extension number.

Although PoE (power over Ethernet)–capable IP phones alleviate the need to run 110/220 VAC power and have desks cluttered with power bricks and cable, misuse can often lead to expensive repair or a total writeoff of equipment. It is important to understand that not all PoE equipment currently in use is compliant with the standard IEEE 802.3af. According to the standard, the power-sourcing equipment supplies power either on spare pins (10/100 Base-T) or over data pins (1000 Base-T), whereas the powered device must be able to accept power from both options. It is obvious that power supplied to a powered device on the wrong pins will cause damage. Long before the PoE standard was approved in June 2003, vendors had been shipping switches and devices with proprietary PoE capability. This problem is expected to diminish; however, it is always best to check with the vendor for proper compliance, especially in mixed-vendor situations (e.g., using Cisco 7960 IP phones with non-Cisco PoE switches).

Conclusion

SIP has gained a significant foothold in the past months, becoming the protocol many companies are betting their businesses on. On the one hand, there are enterprise telecommunication manufacturers turning away from their well-respected proprietary protocols toward SIP. On the other hand, businesses are searching for SIP peers to allow them to bypass expensive toll charges. It remains too early to tell whether or not SIP will become the predominant protocol for VoIP, but the protocol itself will sure be around for quite some time. Nonetheless, it is advisable to invest in VoIP infrastructures capable of supporting different protocols. In terms of soft IP phones, it may be as easy as running separate applications for different protocols. For hard IP phones, vendors will need to provide different firmware images for each of the protocols they support.

book reviews

ELIZABETH ZWICKY

zwicky@greatcircle.com

with Sam Stover and Rik Farrow

STATISTICS HACKS: TIPS & TOOLS FOR MEASURING THE WORLD AND BEATING THE ODDS

Bruce Frey

O'Reilly, 2006. 336 pages.
ISBN 0-596-10164-3.

I have to admit, I am a pro-statistics person. Yes, I know, that kind of redefines "geeky." I'm still whining about a bad experience years ago where somebody decided to move all the "average-sized" mailboxes first. She meant the mean. The mean mailbox size was something like 34 kilobytes. The standard deviation was something like 145 kilobytes. Not surprisingly, there were no "average-sized" mailboxes. And you know what? When I tell this story most people smile and nod politely and edge away. (The rest of them mutter supportive indignant things about non-normal distributions and appropriate uses of averages.)

Anyway, I happen to think that a basic understanding of statistics and probability is *really important*. It will keep you from making all sorts of stupid mistakes, ranging from the above mistake (she wanted the mode, which would have led her to the 100,000 empty mailboxes), to submitting conference papers where you draw sweeping conclusions based on 7 data points per category, to saying confidently, "Oh, that's a one-in-a-million chance" without noting that you're talking about one in a million file writes on a system that does 20 million a day.

Some day, somebody is going to write the perfect math book for system administrators. This is not that book, but it does have what you really need: the basics of statistics and probability required to do something intelligent with most of the problems you encounter. The information on testing is particularly hard to find elsewhere in any useful and palatable form. It will also teach you how to win games of chance, in case you wish to spend your spare time on bar bets or, more likely these days, poker.

To use this book, you're going to need to be reasonably comfortable with math. You don't need to actually be good at it or anything; it doesn't ask you to do anything more complicated than addition and multiplication. But it doesn't have the space to do a lot of hand-holding. On the flip side, if you are seriously interested in statistics, you're going to want more than this. But for an average system administrator, this book provides just the right amount of detail, enough for a clever person to do a good-enough approximation.

GOOGLE, THE MISSING MANUAL, 2ND EDITION

Sara Milstein, J. D. Biersdorfer, and Matthew MacDonald

O'Reilly, 2006. 446 pages.
ISBN 0-596-10019-1.

More true confessions: I spend an absurd amount of time at dinner tables where I'm the only person employed in the computer industry who does not work for Google. You might be surprised how unenlightening this is if you actually want to use Google. Still, I expected that I knew most of what there was to know. I didn't.

I mean, I knew how to use Google as a calculator (try searching for "6 tsp to sticks of butter") and what Adsense and Google Answers are, and I have a Google home page and a Gmail account and all that fun stuff. I can Froogle and search for images. (Anybody with a toddler and a computer must learn to use image search!) But I didn't know about using Google via SMS, or Google Analytics, or some of the tricks for getting phrase searches to work right.

So on the whole I found this book educational—more educational than I had expected. My Google-employee husband found out some things, too. I'm willing to bet pretty much anybody will get something useful about Google out of it.

It has some flaws; first, Google changes too fast for a mere book to keep up. Second, there are way too many platforms and interests out there, so any given user is going to be skipping lots of stuff. I use a Treo and a Macintosh. I'm sure the coverage on how to make your cell phone use Google well is really handy if you don't have a keyboard. And if you use a PC, it's nice to have all the PC-specific goodies covered too (although if you use something that's neither a PC or a Mac, those sections are going to be a big yawn, as there's little to no mention of UNIX platforms).

Even so, I liked it. It's hard to see how one technically minded person would get enough out of it to justify the purchase cost, but it would probably be worth it for a computer-literate family member with an interest who wasn't a serious geek, or as a shared resource for a group.

LINUX TROUBLESHOOTING FOR SYSTEM ADMINISTRATORS AND POWER USERS

James Kirkland, David Carmichael, Christopher L. Tinker, and Gregory L. Tinker

Prentice-Hall, 2006. 571 pages.
ISBN 0-13-185515-8.

If you are an experienced system administrator who wants Linux information, you will find useful troubleshooting information in this book. Unfortunately, it tries to cover all of system administration as well. Sometimes it's right (if you are going to cover all of system security in 60 pages, it is in fact important to tell people not to try to fix a compromised machine but instead to reinstall it), sometimes it's misguided (if you are going to cover all of backups in 20 pages, the towers of hanoi schedule is not one of the things you ought to be including), and sometimes it's just too compressed to make sense.

Mostly, it simply avoids providing explicit instruction about troubleshooting techniques. There are lists of useful tools and samaple problems with solutions, but these are more hints than procedures you could apply to your own problems. Unfortunately, when there are instructions about troubleshooting, they're not very good ones. For instance, they advise troubleshooting network problems from the lowest stack layer up. This is very logical, but it isn't what anybody ever does, for a variety of good and not-so-good reasons. (For instance, the hardware is rarely broken, and it's way more trouble to get out of your chair and look at it than it is to type commands without moving.) Recommending it suggests that the authors felt the need to provide a system, but they don't have experience actually teaching people to troubleshoot.

I love the idea of this book, and I'm pretty fond of some of the information. But for system administration advice, you'd be better served by any current system administration text, for troubleshooting I still don't know of a good reference, and all that leaves is Linux basics, which are nicely covered, but don't take up that much of the space, and are widely available elsewhere.

COMPUTER PRIVACY ANNOYANCES: HOW TO AVOID THE MOST ANNOYING INVASIONS OF YOUR PERSONAL AND ONLINE PRIVACY

Dan Tynan

O'Reilly, 2005. 177 pages.
ISBN 0-596007752.

On the good side, this is a level-headed discussion of the various ways of protecting your privacy on-line. It's written for a not-extremely-technical-but-not-yet-extremely-paranoid audience and should help those people become appropriately nervous. It walks a fine line between sounding alarms about everything and ignoring genuine risks, and it seems to me to hit about the right balance. That is, sometimes I think it's too cavalier and sometimes I think it's paranoid enough to turn off a reader who believes in the fundamental trustworthiness of business and government, and yes, such people do exist in this day and age, and they need to read this kind of book, too.

So mostly I liked it. Once again, however, it's mostly oriented toward Windows machines. Actually, the Macintosh gets mentioned a couple of times but UNIX (in any form or flavor) is never even whispered, as far as I can tell. This is not such a big deal, because most of the book deals with platform-independent issues such as workplace privacy, public information, and government issues. (I think my father the Windows-hater would find it plenty useful.)

I'd also like to see some more mention of encryption. It comes up occasionally, but not with an explanation of what terms such as "weak" and "strong" might mean to an average user, or big warning boxes saying "HEY! Don't lose your password! That would be bad!" And that whole public-key private-key thing? It's neither explained nor mentioned.

This is a good book for handing out to your PC-using friends and relatives who're somewhat worried and pretty technically savvy. Because it spends a considerable time on issues that aren't related to computers you personally own, it will have information of interest to serious technical people who're not already privacy activists, but you may have to skip largish parts if you run your own UNIX boxes at home—you probably understand the issues and can't apply the suggested solutions in a couple of sections.

WRITING SECURITY TOOLS AND EXPLOITS

James C. Foster

Syngress, 2005. 664 pp.
ISBN 1-59749-997-8.

Reviewed by Sam Stover

I'd like to start out by saying that this book is not designed for people new to security, nor to programming, for that matter. While the first chapter definitely has that "read this if you just want to talk the talk," after that it goes uphill fast.

The moment you turn the page from Chapter 1 to Chapter 2, it's go time. The tutorial on assembly (with the goal of making sense of shellcode) is not for the weak of heart. I'll admit I had to reread several of the sections in this chapter, taking me back to my college days. In some ways this book really does read like an academic text, but the goal is to teach, and it certainly does that. When a certain point would just "click," it made it all worthwhile.

Once you get through the shell-code chapter, you'll jump into a chapter for each of the three main types of exploits: stack overflows, heap overflows, and format string vulnerabilities. All three chapters follow the same basic format, with just the right number of examples, along with discussion on hurdles to overcome when trying to find and/or prevent these types of vulnerabilities.

Now that you've seen the three main classes of exploits, we move into the second part of the book, which shows you how to find vulnerabilities and code exploits for them. Chapters 6 and 7 focus on local and remote exploits, race conditions, and socket coding. Both chapters contain a fair number of case studies where actual exploits are used to apply the concepts you learned in the first five chapters.

The remaining five chapters focus on application-specific coding for Ethereal, Nessus/NASL, and Metasploit, with Metasploit getting a total of three chapters. I found these chapters to be useful, with the caveat that the information is extremely redundant if you have certain other Syngress books. For example, I recently reviewed *Penetration Tester's Open Source Toolkit*, which contains the first two "Extending Metasploit" chapters as well as the "Coding for Nessus" chapter. The names of the chapters were the same, as were all of the figures, tables, etc. I found the cut-and-paste mentality a bit disappointing in spite of the technical value of the chapters. Fortunately, at least for Metasploit, there is a third chapter that addresses topics such as Inline Egg and Meterpreter, so all is not lost. Although I haven't read it yet, there is another book, *Buffer Overflow Attacks*, by the same author and publisher, that appears to have a lot of overlap as well. Just glancing over the table of contents shows that the stack, heap, and format string chapters look very similar, as do the shell-code and assembly sections. They are not exact duplicates, but too close for my comfort.

All in all, I think this book is a valuable reference, and I would recommend it to anyone interested in learning about exploit development from top to bottom. The chapter recycling from other books is a big disappointment, but this is only relevant if you have the other books. And maybe now that you've read this review, your expectations will be managed, and you won't be as annoyed as I was.

I don't want to let the duplicate chapters overshadow the value of this book as a whole though. It's a decent exploit book with plenty of examples. If you don't have the other books, this is probably just as good a place to start as any.

PGP & GPG: EMAIL FOR THE PRACTICAL PARANOID

Michael Lucas

No Starch Press, 2006, 216 pages.
ISBN 1-59327-071-2.

Reviewed by Rik Farrow

Lucas's book is aimed squarely at the people who know they should be using PGP or GnuPG but just haven't gotten there yet. His easy-going writing style gets across key ideas—for example, the differences between private-key and public-key cryptography. But the real strength of the book lies in the chapters devoted to using either the command line GnuGP tools or the GUI-based PGP. Lucas takes you step by step through creating your own key pair, sharing your public key, and maintaining your own keychain. You should buy this book for your boss or less technical buddies, or for the people who should have started using GPG by now.

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, Java, 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.

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*.

TO THE EDITOR

In the article about virtual firewalls that was published in the December 2005 edition of *;login:*, I mentioned that a full "shrink-wrapped" version of the virtual firewall would be made available on my Web site.

The latest version is now available at http://www.cs.drexel .edu/~vp/VirtualFirewall/ as a VMware virtual machine configuration. The distribution is a compressed tar file of all the files required to configure and boot the VM for the virtual firewall.

The release creates a 1Gb virtual disk running OpenBSD 3.7.

Instructions on installing the booting the VM are provided on the site.

—*Vassilis Prevelakis*

THE USENIX ASSOCIATION FINANCIAL REPORT FOR 2005

ELLIE YOUNG
ellie@usenix.org

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 2005 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, 2005, are available on request.

FINANCIAL STATEMENT SUMMARY

USENIX continues to be a healthy organization. At the beginning of 2005, USENIX had budgeted for our having a modest surplus at year end in operations. However, we incurred a deficit of $110K in operations, which was mainly due to a drop in membership revenue, and lower than projected attendance at the USENIX Annual Technical Conference. This deficit was covered by the interest and dividend income in the Reserve Fund. We ended the fiscal year with $15K increase in net assets.

USENIX MEMBERSHIP DUES AND EXPENSES

USENIX averaged 5,400 members in 2005, a 7% drop from the previous year. Of these, half opted for SAGE membership as well.

Chart 1 on the facing page shows the total USENIX membership dues revenue ($573K) for 2005, divided into membership types.

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 describes how the money allocated to student programs, sponsorship of other conferences, and standards activities ($263K) was spent in 2005.

Chart 4 shows how the USENIX administrative expenses were allocated. (The category "Misc." covers such items as renewals, taxes, licenses, consultants, temp help, training, etc.) Chart 5 shows where the $164K in expenses to provide SAGE benefits and services was spent. (Note: SAGE member dues revenue was $128K.)

USENIX ASSOCIATION
STATEMENTS OF FINANCIAL POSITION
As of December 31, 2005 and 2004

ASSETS	2005	2004 (Unaudited)
Current Assets		
Cash & cash equivalents	$ 1,138,826	$ 849,131
Receivables	47,081	86,417
Prepaid expenses	39,003	53,316
Inventory	5,913	10,829
Total current assets	1,230,823	999,693
Investments at fair market value (board designated reserve)	5,343,665	5,205,701
Property and Equipment		
Office furniture and equipment	477,724	464,569
Less: accumulated depreciation	(421,264)	(380,152)
Net property and equipment	56,460	84,417
Other assets	206,515	328,014
	$ 6,837,463	$ 6,617,825
LIABILITIES AND NET ASSETS		
Current Liabilities		
Accounts payable and accrued expenses	$ 589,575	$ 312,556
Accrued taxes payable	11,200	
Contributions held for OpenAFS	1,200	
Deferred Revenue	40,000	3,060
Total current liabilities	641,975	315,616
Long-term Liabilities	206,515	328,014
Net Assets		
Unrestricted Net Assets	5,988,973	5,974,195
Net Assets	5,988,973	5,974,195
	$ 6,837,463	$ 6,617,825

USENIX ASSOCIATION
STATEMENTS OF ACTIVITIES
For the Years Ended December 31, 2005 and 2004

	2005	2004 (Unaudited)
REVENUES		
Conference & workshop revenue	$ 3,281,826	$ 3,264,893
Membership dues	572,560	632,252
Product sales	12,858	8,825
SAGE dues & other revenue	128,039	145,673
General sponsorship	2,925	
Total revenues	3,998,208	4,051,643
OPERATING EXPENSES		
Conferences & Workshops	2,709,818	2,704,759
Membership, login:	387,520	428,755
Projects & GoodWorks	291,935	229,239
SAGE	164,111	208,757
Management and General	457,473	441,554
Fund Raising	97,782	112,885
Total operating expenses	4,108,639	4,125,949
Net operating surplus/(deficit)	(110,431)	(74,306)
NON-OPERATING ACTIVITY		
Donations	0	50
Interest & dividend income	171,766	161,436
Gains & losses on marketable securities	32,907	199,495
Investment fees	(59,267)	(66,348)
Other non-operating	(20,197)	(2,313)
Net investment income & non-operating expense	125,209	292,320
Increase/(decrease) in net assets	14,778	218,014
Net assets, beginning of year	5,974,195	5,756,181
Net assets, end of year	$ 5,988,973	$ 5,974,195

USENIX ASSOCIATION
STATEMENTS OF FUNCTIONAL EXPENSES
For the Years Ended December 31, 2005 and 2004

	Conferences and Workshops	Programs and Membership	Student Programs, Good Works and Projects	SAGE	Total Program	Management and general	Fund Raising	Total Support	2005 Total	2004 (Unaudited) Total
Operating Expenses										
Conference & workshop-direct *	$ 1,878,613	$	$	$	$ **1,878,613**	$	$	$ **0**	$ **1,878,613**	$ **1,855,355**
Personnel and related benefits:										
Salaries	510,820	124,085	17,541	73,607	**726,054**	209,682	66,623	**276,305** o	**1,002,359**	**1,048,123**
Payroll taxes	38,013	9,234	1,305	5,478	**54,030**	15,604	4,958	**20,561** o	**74,591**	**101,586**
Employee benefits	106,647	25,906	3,662	15,367	**151,582**	43,776	13,909	**57,686** o	**209,267**	**208,898**
Membership/products		8,068			**8,068**			**0**	**8,068**	**8,254**
Membership/login:		137,433			**137,433**			**0**	**137,433**	**164,866**
SAGE expenses *				52,061	**52,061**			**0**	**52,061**	**39,630**
Student programs, Good Works, and projects			263,473		**263,473**			**0**	**263,473**	**200,083**
General and administrative *	175,725	82,794	5,954	17,598	**282,071**	188,411	12,292	**200,703**	**482,775**	**499,154**
	$ 2,709,818	$ 387,520	$ 291,935	$ 164,111	$ **3,553,384**	$ 457,473	$ 97,782	$ **555,255**	$ **4,108,639**	**4,125,949**

USENIX ASSOCIATION
STATEMENTS OF CASH FLOWS
For the Years Ended December 31, 2005 and 2004

	2005	2004 (Unaudited)
CASH FLOWS FROM OPERATING ACTIVITIES		
Change in net assets	$ 14,778	$ 218,014
Adjustments to reconcile increase in net assets to net cash provided by/(used for) operating activities:		
Depreciation	41,524	58,485
Net investment income designated for long-term purposes	(105,057)	(89,553)
Realized & unrealized gains on investments	(32,907)	(199,495)
Decrease in receivables	39,336	4,848
Decrease in inventory	4,916	5,690
Decrease/(Increase) in prepaid expense	14,313	(28,907)
Increase in accrued expenses	277,019	204,520
Increase in accrued taxes	11,200	
Increase/(Decrease) in deferred revenue	38,140	(9,940)
Total adjustments	288,484	(54,352)
Net cash provided by operating activities	303,262	163,662
CASH FLOWS PROVIDED BY/(USED FOR) INVESTING ACTIVITIES:		
Purchase of investments	(2,423,109)	(2,832,308)
Sale of investments	2,423,109	2,832,308
Disposition of equipment	1,643	
Purchase of property & equipment	(15,210)	(20,029)
Net cash used for investing activities	(13,567)	(20,029)
Net change in cash & equivalents	289,695	143,633
Cash & equivalents, beginning of year	849,131	705,498
Cash & equivalents, end of year	$ 1,138,826	$ 849,131

Chart 3: Student & Support for Other Programs Expenses 2005

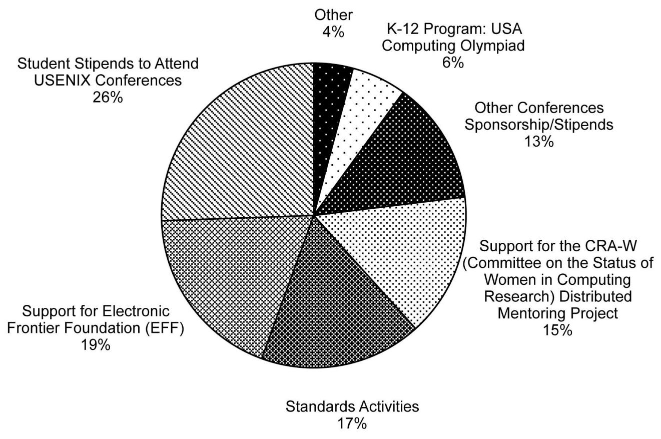


Chart 4: USENIX Administrative Expenses 2005

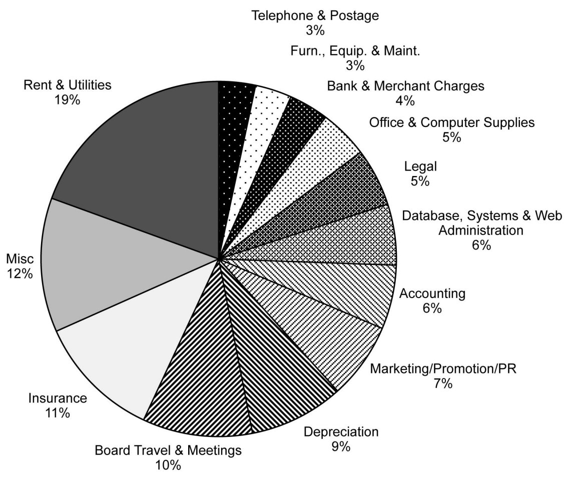


Chart 1: USENIX Member Revenue Sources 2005

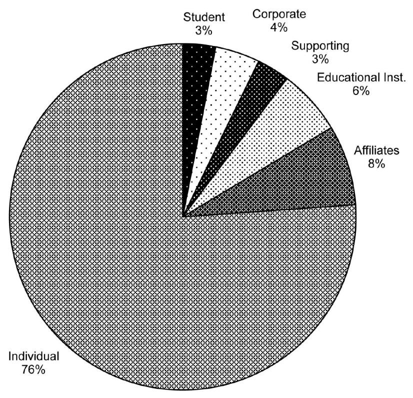


Chart 2: Where Your 2005 Membership Dues Went

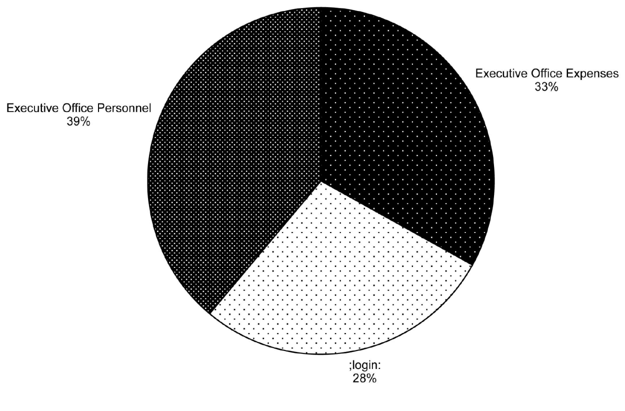
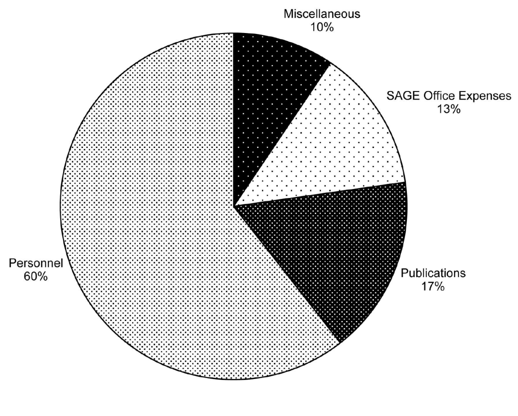


Chart 5: SAGE Expenses 2005



SUMMARY OF USENIX BOARD OF DIRECTORS MEETINGS AND ACTIONS

ELLIE YOUNG,
Executive Director

The following is a summary of the actions taken by the USENIX Board of Directors from March through June 30, 2006.

CONFERENCES

A proposal from Dan Geer to co-locate and sponsor a first-time securitymetrics.org workshop at USENIX Security '06 was approved.

CRA IMAGE OF COMPUTING TASK FORCE

USENIX will contribute $10K in 2006 toward funding a full-time professional to develop messages and deliver strategies to create a more positive image of computing as a career. Other sponsors, such as Microsoft, HP, ACM, and CRA, will provide funds as well.

SPONSORSHIP

It was agreed that USENIX would sponsor the 2006 BSDCan conference at a $5,000 level. It also was agreed to support the 2006 Grace Hopper Women in Computing Conference at a $10,000 level.

NEW SUPPORTING/SPONSORSHIP CLASS

It was agreed to create new levels of sponsorship: Supporter, $5,000; Partner, $6,000–$49,000; Patron, $50,000 and up. For information on the benefits offered at each level, please contact cat@usenix.org or ellie@usenix.org.

FINANCES

Registration fees for the 2006 Security Symposium were raised by $20 to cover an anticipated shortfall in revenue for the event.

To keep up with rising costs, the registration fees for the LISA '06 technical sessions were raised by $20/day; workshop fees will be $150.

Booth fees for vendors will be raised for the first time in four years, by $195, effective Sept. 1.

A new remuneration package for tutorial instructors was approved.

It was moved to thank everyone involved in the successful audit of the 2005 finances.

A revised budget was approved. It has a projected deficit and includes spending up to $50,000 for a feasibility study and subsequent migration to another database system for the Association.

AWARDS

It was agreed to work with the membership in instituting some new awards. Evard and Cole will put together a proposal.

SPECIAL INTEREST GROUPS

A proposal to change the USENIX policy document to reflect the current situation regarding Special Interest Groups and the status of SAGE and to remove the old language related to Special Technical Groups was adopted, as follows:

10. Special Interest Group: SAGE

SAGE is a Special Interest Group of the USENIX Association. Its goal is to serve the system administration community by:

- Establishing standards of professional excellence and recognizing those who attain them
- Promoting activities that advance the state of the art or the community
- Providing tools, information, and services to assist system administrators and their organizations
- Offering conferences and training to enhance the technical and managerial capabilities of members of the profession

SAGE is a class of membership within USENIX. Members can join SAGE in addition to their USENIX membership or join as a SAGE-only member.

SAGE SIG governance: SAGE is overseen by a subcommittee of the USENIX Board of Directors. The USENIX staff and SAGE members develop and deliver benefits and programs.

NEXT MEETINGS

The next regular meetings of the USENIX Board of Directors will be held on August 3–4 in Vancouver, B.C., Canada, and on December 5 in Washington, D.C.

2006 USENIX AND STUG AWARDS

USENIX LIFETIME ACHIEVEMENT AWARD AWARD

The 2006 "Flame" Award was awarded to Radia Perlman for her many contributions to networking in routing (link state distribution), as well as in security (sabotage-proof networks). In particular, USENIX acknowledges Dr. Perlman's work in developing the spanning tree algorithm that is de rigueur in all LAN products in use today. She is currently a Distinguished Engineer at Sun Microsystems.

SOFTWARE TOOLS USER GROUP AWARD

The 2006 STUG Award was given to Bram Cohen for his work on BitTorrent, one of the most popular file distribution methods on the Internet. USENIX feels that BitTorrent is a software tool that, like the namesake of this award, provides "significant enablement" to users of the Internet and makes files, regardless of content, much more widely available to all.

For more information on these awards, see www.usenix.org/about/flame.html and www.usenix.org/about/stug.html.

conference reports

THANKS TO THE SUMMARIZERS

Asad Awan
Maria R. Ebling
Rik Farrow
Andreas Haeberlen
Guy Lichtman
Nikitas Liogkas
Nikolaos Michalakis
Alan Mislove
Tapan Parikh
Swapnil Patil
Ansley Post
Niraj Tolia
Alex Varshavsky

SUMMARIES

NSDI '06: 3rd Symposium on Networked Systems Design & Implementation

San Jose, CA
May 8–10, 2006

KEYNOTE ADDRESS

PARALLEL COMPUTING AND SEARCH: A PERSONAL PERSPECTIVE

Udi Manber, VP of Engineering, Google Inc.

Summarized by Nikitas Liogkas

Udi Manber started this keynote address by describing his past efforts in computer science. When he started, thirty years ago, people thought that the speed of computation would stop being an issue soon. Then, twenty years ago, he was involved in writing parallel programs that should be robust against unexpected failures. Recently, he worked for the Yahoo! Web portal. He is now the VP of Engineering for Google, a company working on search, a new research area, based on the premise that computing power, main memory, and disk space are very cheap. Traditionally, the "search" feature was considered a commodity, not interesting from a research point of view. This is now changing. Google has made it its mission to "organize the world's information and make it universally accessible and useful." Within Google, the corporate culture is optimized for innovation: Employees can devote 20% of their time to work on their own projects. In fact, many successful products have come out as a result of these efforts through Google Labs. Google has an enormous user base; the traffic on Google servers never stops: They were receiving 1,000 queries/s at 2 a.m. PST on December 25!

The speaker went on to note that humans have not become smarter in the past ten years, but computers have: There are now considerable infrastructure, tools, and algorithms to help us with searching. He described the lifetime of a Google query and outlined potential failures, both in hardware and in software, and how to mitigate their impact by utilizing replication and redundancy. At Google, servers experience an average of 1.9 machine failures per job, so failure is a fact of life. Fortunately, fault-tolerant software makes cheap hardware practical. He then described some of the tools currently used to achieve the desired effects at Google. The key insight behind many of these tools is that many large data sets can have a simple structure; these include the Web page repository, and query logs, as well as the health records of machines. He also briefly mentioned MapReduce, a tool that extracts relevant information for each record of input and then aggregates the results.

He said that a personal goal of his is to make parallel programming easy to use, put search on top of it, and take it to the next level. Nevertheless, search is very hard, for numerous reasons. First, people do not really know how to search efficiently. In addition, content authors do not know how to make their content findable, while schools do not teach proper searching techniques, and universities hardly do research in this area. To make search easier and optimize the search results for users, Google utilizes certain tricks, such as providing the best answer at the top, and using wildcards. There is also a spelling-correction feature, which uses the Web as a contextual lexicon to guess misspelled search queries.

The speaker also shared with us a story about a restaurant search result, where the entire Web page consists of a single image, and how that makes it difficult to provide good results to users. He then went on to describe how Google tries to address all these shortcomings by running usability tests with real users and also by analyzing the logs. He gave an example of such a study by showing anonymous traces of search actions and attempted to analyze user behavior by exploring a specific "full moon" query. Lastly, he prompted all the researchers in the room to build tools for searching and analysis of queries, noting that having fast CPUs and ample storage has the potential for changing the field in a deep way.

During the Q&A period, Brian Noble from the University of Michigan inquired about a potential curriculum of a class on searching, but he did not get a concrete answer. A researcher from the University of Utah noted that Google could even change the culture of a country and warned about the related responsibilities. Michael Freedman from NYU brought up the possibility of adversarial search; another researcher asked about Google's cached pages and whether password-protected pages are cached. The speaker replied that adversarial search is indeed a concern and that cached pages are submitted directly to Google on a voluntary basis; there is no automated tool that Google uses to guess passwords for pages. Emin Gün Sirer from Cornell brought up the idea of searching in a personal context, based on what you have searched for in the past, and the speaker mentioned that this is already offered as a service by Google. Emil Sit from MIT inquired about potential research directions for search, and classes of problems, but the speaker replied that he cannot disclose what Google is currently looking into. A researcher from HP Labs asked whether the problems to solve are different for different search engines, in particular for Yahoo! versus Google. According to the speaker, Yahoo!, being a portal, may be looking at the same problems from a different perspective, but there are definitely many commonalities. Petros Maniatis from Intel Research inquired whether employees have the freedom to define the direction of the projects they work on, or whether questionable or semi-illegal projects are filtered out early. Interestingly, the speaker noted that Google's mantra ("do no evil") is actually followed by management and technical employees alike and that Google already filters out credit card number results, thereby protecting user privacy. Lastly, Peter Druschel from the Max Planck Institute for Software Systems wondered about the role of Google as a media organization providing a variety of opinions and the great responsibility this entails. The speaker replied that the results for a query are not produced manually, but rather through an algorithm, and thus manual intervention is not performed on the results. As such, whatever Google provides is presumably free of bias.

WIDE-AREA NETWORK SERVICES I

Summarized by Niraj Tolia

■ ***Experience with an Object Reputation System for Peer-to-Peer Filesharing***

Awarded Best Paper

Kevin Walsh and Emin Gün Sirer, Cornell University

This talk, given by Kevin Walsh, described the design of Credence, a decentralized peer-to-peer reputation system and the evaluation results from a long-term deployment of the system. The system allows peers to make statements regarding the authenticity of files. While deciding on whether to fetch an object, a client would make a network query to gather previous votes on the object. The votes would be weighted, with a higher weight being given to peers that have voted similarly to the client in the past and a negative weight being given to votes from peers that have had contrary voting patterns in the past. Like-minded peers can be found either through direct correlation or via transitive trust relationships. An implementation of Credence within the Limewire client for Gnutella networks validated its goals as it identified all known large decoys and a number of smaller decoy attacks on the system. More information can be found at http://www.cs.cornell.edu/people/egs/credence/.

During the Q&A session, Phil Gibbons from Intel Research asked about Sybil attacks and whether a single client could cheat the system by providing almost entirely good answers but a few negative ones for the files the client was interested in. Kevin replied that this was hard to do as malicious votes would need to generate a large number of positive votes and therefore would have to do honest work. Further, as an effective attack would require multiple clients, attackers interested in different files would cancel each other out. Jonathan Duerig from the University of Utah asked a similar question about honest users turning into malicious ones and voting down authentic content. However, in this case, even though users might not initially download the content, they will ultimately move down the list, download it, and positively vote on it. This would further eliminate the now-malicious peers from the web of trust. Robert Ricci from the University of Utah asked how many people were required to vote on decoys before Credence became effective.

- *Corona: A High Performance Publish-Subscribe System for the World Wide Web*

Venugopalan Ramasubramanian, Ryan Peterson, and Emin Gün Sirer, Cornell University

This talk, given by Venugopalan Ramasubramanian (Rama), described Corona, a framework for detecting updates to online data while delivering high performance, availability, and scalability. This work is motivated by a large number of online sources that are dynamic and frequently updated. However, polling these sources by a large number of clients is not scalable, delivers poor update performance, and can be responsible for high server load. This work expressed the problem as a mathematical optimization problem that allowed for fast update detection and informed resolution of tradeoffs between bandwidth and latency. The system is layered on top of a structured overlay to decide on node allocation problems for given workloads. For more details, see http://www.cs.cornell.edu/people/egs/beehive/corona/.

During the Q&A session, Hari Balakrishnan from MIT asked whether the techniques used in Corona (a pull system) would also be applicable to push systems. The answer was that some of it would be, since resource allocation with a large number of nodes can also be a crucial problem for push systems. Steve Hand from the University of Cambridge asked about the feasibility of a structured overlay with Corona's optimization being built in from the start. Rama replied that they haven't looked into this but it would be an interesting case to consider.

- *Scale and Performance in the CoBlitz Large-File Distribution Service*

KyoungSoo Park and Vivek S. Pai, Princeton University

This talk, given by KyoungSoo Park, was motivated by the fact that most Web content distribution networks (CDNs) are optimized for small files whereas larger file transfer (100 MB to >2GB) is becoming more prevalent today. Using current CDNs for large files can be wasteful, because each large file evicts a much larger number of smaller files from the CDN's cache, increases memory pressure, and might waste resources as a result of overprovisioning. This work presented the CoBlitz system, which allows CDNs to handle large files by splitting larger objects into chunks and distributing these chunks over different proxies. CoBlitz has been shown to be 55–80% faster than BitTorrent. The talk also discussed how the system addressed the challenges of scalability, robustness, peering set differences, and origin-server load. The evaluation of the system contained controlled experiments as well as results from a real-world deployment over the past two years. More information can be found at http://codeen.cs.princeton.edu/coblitz/, and any public URL can be accessed using the system by surfing over to http://coblitz.codeen.org:3125/.

During the Q&A session, Robert Ricci from the University of Utah asked whether the performance comparison to Shark and Bullet were based on new experiments. However, it turned out that the results were actually directly taken from previously published results, because source or binaries for the systems were unavailable. Jawwad Shamsi from Wayne State University wondered if the experimental setup used different network links for BitTorrent and CoBlitz systems. However, it turns out that the same machines and links were used, to provide a fair comparison. Nick Feamster from Georgia Tech asked whether CoBlitz was faster than BitTorrent because it did not have to deal with free-riding (such as tit-for-tat) or policies such as local-rarest-first. It was argued that this was not the case and that performance was similar because BitTorrent also has seeds to help in such situations, whereas CoBlitz does not. Michale Sirivianos from the University of California, Irvine, mentioned that BitTorrent seemed to be much more scalable for an extremely large number of nodes than CoBlitz. However, KyoungSoo mentioned that CoBlitz too can deliver this scalability and that real usage has shown that CoBlitz can deliver higher mean download bandwidth than BitTorrent in a number of cases. Further, the goal of the CoBlitz project is not to replace BitTorrent but instead to improve Web CDNs.

REPLICATION AND AVAILABILITY

Summarized by Alan Mislove

- *Efficient Replica Maintenance for Distributed Storage Systems*

Byung-Gon Chun, University of California, Berkeley; Frank Dabek, MIT Computer Science and Artificial Intelligence Laboratory; Andreas Haeberlen, Rice University/MPI-SWS; Emil Sit, MIT Computer Science and Artificial Intelligence Laboratory; Hakim Weatherspoon, University of California, Berkeley; M. Frans Kaashoek, MIT Computer Science and Artificial Intelligence Laboratory; John Kubiatowicz, University of California, Berkeley; Robert Morris, MIT Computer Science and Artificial Intelligence Laboratory

Emil Sit started by presenting the motivation for this work, which is to study how to provide efficient replica maintenance in a distributed storage system. Most systems in this area work by creating an initial number of replicas, then monitoring the system for replica failures and responding by creating new replicas. Thus, the high-level problems are determining how many initial replicas to create and deciding which replica failures must be responded to.

To demonstrate why such an analysis is necessary, Sit presented a straw-man naive replication algorithm, showing that the bandwidth cost of replica maintenance is very high (over 250% of optimal). This is largely because most solutions cap the number of existing replicas and focus only on availability, rather than durability. To fix these flaws, the authors present the Carbonite maintenance algorithm, which attempts to provide data durability as a practical goal. From Carbonite, Sit distilled three lessons for replica maintenance:

First, Sit noted that extra replicas (above an artificial cap) avoid the need for future replicas. Thus, the authors recommended removing the artificial limit on the number of replicas alive at any one time. They showed that removing this limit results in an acceptable number of replicas and does not cause the number of replicas to grow without bound.

Second, Sit noted that to provide high availability and durability, the system must be engineered to provide fast repair times. However, since the repair time is often limited by the uplink bandwidth from existing replicas, increasing the number of replicas (in the previous point) increases the rate at which repair can be performed. Additionally, Sit recommended placing replicas on random hosts, to ensure that a variety of access links can be used for repair.

Third, the authors also attempted to handle "bursts" of permanent failures. Using a Markov-chain-based model, they derived the number of replicas required for systems with certain failure properties (and showed that three replicas are sufficient for PlanetLab).

In the Q&A session, James Bacons from the University of Michigan asked how the analysis holds up under more dynamic churn. Sit responded by describing how the system will adapt over time and explaining that the availability fraction does not have to be estimated. Ryan Peterson from Cornell asked about increasing availability by picking a node to create a new replica rather than by choosing one randomly.

■ *PRACTI Replication*

Nalini Belaramani, Mike Dahlin, Lei Gao, Amol Nayate, Arun Venkataramani, Praveen Yalagandula, and Jiandan Zheng, University of Texas at Austin

Jiandan Zheng began this talk by noting that a number of replication systems exist, with each one focusing on one or two of the properties of (i) practicality through partial replication, (ii) arbitrary consistency semantics for applications, and (iii) topology independence by allowing data exchange between any two nodes. Since no replication system currently provides all three of these properties, the authors present the PRACTI architecture for data replication, which allows greater flexibility in choosing tradeoffs and potentially opens up new areas of the design space.

Zheng demonstrated why providing all three PRACTI properties is hard by showing a set of files, including A and B, which are replicated on two desktop computers and one handheld one. Since the handheld computer is space-limited, it can only replicate B and not A. Now, if writes come in on the first desktop for A (resulting in A') and then B (resulting in B'), and the handheld machine is updated, it will only receive the update B' for B, since it is not replicating A. However, if the handheld is used to update the second desktop machine, that machine will have A and B', which does not reflect causal consistency.

To solve these problems and present a unified architecture, the authors presented the following ideas. First, PRACTI uses peer-to-peer log exchanges to synchronize metadata between nodes, similar to Bayou, with the metadata logs always sent first and the data logs fetched on demand. Second, for efficiency, PRACTI allows imprecise invalidations via object group summaries. This provides topology independence (since it is peer-to-peer), arbitrary consistency (since entries are ordered), and practicality (since only metadata is sent, and it can be aggregated).

For their evaluation, the authors compareD the PRACTI system to other replication systems such as Coda and showed that the requirement of a server connection makes other systems much slower.

In the Q&A session, Emin Gün Sirer from Cornell asked whether causal message logging with carefully tracked updates would result in a system that is close to PRACTI.

■ *Exploiting Availability Prediction in Distributed Systems*

James W. Mickens and Brian D. Noble, University of Michigan

James Mickens began the talk by pointing out that the traditional paradigm for handling churn in distributed systems is to use overly pessimistic assumptions. This work, in contrast, wants to understand, predict, and exploit churn. As a preview, by exploiting availability, the authors can reduce data copying in DHTs and are able to avoid unreliable hosts in delay-tolerant networks.

To try and predict when a machine will become unavailable, the authors examineD multiple classes of availability predictors, including predictors based on Markov chains and linear predictors. To evaluate and compare these predictors, the authors tested the predictors on two data sets, one from Microsoft Research and another from the PlanetLab testbed.

They found that PlanetLab is highly predictable in the short term but unpredictable in the long term, since failure events in PlanetLab are unpredictable and infrequent. However, they found the Microsoft Research trace to be much more predictable, largely because many of the machines were almost always on and others exhibited weekly and daily availability patterns.

To examine whether application-level availability differs from machine availability, the authors used the predictors on a trace from the OverNet DHT. They found OverNet availability to be significantly less predictable than the machine availability from PlanetLab and MSR. The authors showed that this is due to an increased level of entropy in the OverNet trace, which fundamentally limits the predictability.

Lastly, to show how their analysis can be used in real-world scenarios, they pointed out that the Chord DHT places replicas on the first *k* of *N* successors to increase availability. They showed that by focusing data on highly available nodes within the next *N* successors, the amount of extra storage required goes down, and the availability goes up. Second, they showed how machine predictability can be used to examine how viruses propagate.

In the Q&A session, topics of discussion included the accuracy of the availability data with respect to ICMP pings and firewalls, and how to use correlated history between machines. Additionally, Mickens was asked whether the predictors could be inverted if the accuracy dropped below 50%.

TOOLS

Summarized by Nikolaos Michalakis

■ *To Infinity and Beyond: Time-Warped Network Emulation*

Diwaker Gupta, Kenneth Yocum, Marvin McNett, Alex C. Snoeren, Amin Vahdat, and Geoffrey M. Voelker, University of California, San Diego

Diwaker Gupta explained the idea of using time dilation to perform tests beyond the limits of the hardware. The motivation for this work was to predict, using currently available hardware, the performance characteristics of hardware that does not yet exist. The key to being able to do this is to recognize time as yet another resource of the system that can be manipulated. Time dilation means that the operating system perceives time as only a small factor of the real time, so with a dilation factor of 10, 10 seconds of real time is perceived as 1 second of system time. Time dilation scales all system components uniformly, but the speaker focused on network dilation for this talk.

To implement time dilation the authors used Xen, and they modified the Xen hypervisor to scale time with a few modifications to the guest OS. To validate that time dilation does not change system behavior, the authors ensured that the same bandwidth and latency characteristics were seen in both the real and the perceived configuration. More specifically, their results showed that time dilation does preserve packet behavior of a single TCP flow, and it is accurate for multiple flows under varying bandwidth. Their experiments tested BitTorrent on multiple nodes. The speaker mentioned that one fundamental limitation to using time dilation in experiments is that they would take longer by a factor of the dilation ratio to finish. Part of the authors' future work would be to implement time dilation with an unmodified OS.

The first question after the talk was focused on what hardware characteristics are not captured by time dilation. The answer to that was that time dilation will work well if hardware improves linearly. Jonathon Duerig from the University of Utah asked how time dilation performs with large systems. Diwaker replied that they haven't experimented with more than 50 machines. Kevin Walsh from Cornell asked whether they scaled all the resources uniformly. The answer was that only temporal resources were scaled. On the question of how the software of 2006 would affect measurements of the hardware of 2026, the answer was that not all changes can be predicted with time dilation.

■ *The Dark Oracle: Perspective-Aware Unused and Unreachable Address Discovery*

Evan Cooke, Michael Bailey, and Farnam Jahanian, University of Michigan; Richard Mortier, Microsoft Research Cambridge, UK

Evan Cooke started the talk by explaining how serious the problem of unused network address spaces is for security and that it is challenging to identify such address spaces, because they are highly distributed over the prefix space. The goal of the work on Dark Oracle was to automate the process of discovering dark addresses by participating directly with allocation, routing, and policy systems. The architecture is composed of two major components. The first is the address allocation data sources. There are three main sources of allocation data: external routing data, internal routing data, and host configuration data (e.g., dhcp). The second major architectural component is the address manager, which utilizes the address allocation data to provide a map of dark addresses.

To reduce the number of misclassified visible addresses as dark addresses, the Dark Oracle samples addresses more often, and to smooth fluctuations it adds a delay from the transition of a visible to dark address. Using this technique the authors found many more dark addresses. More specifically, the benefits of using a perspective-aware approach to dark address discovery are that one can construct both incoming and outgoing honeynets, and automating the process results in an order-of-magnitude more unused addresses being discovered. Finally, by using this approach pervasive honeynets can be built that are better at detecting without being detected.

■ ***Pip: Detecting the Unexpected in Distributed Systems***

Patrick Reynolds, Duke University; Charles Killian, University of California, San Diego; Janet L. Wiener, Jeffrey C. Mogul, and Mehul A. Shah, Hewlett-Packard Laboratories, Palo Alto; Amin Vahdat, University of California, San Diego

Patrick Reynolds talked about Pip, an infrastructure for comparing actual behavior and expected behavior to expose structural errors and performance problems in distributed systems. The idea is that the programmer defines the expected behavior of the system and Pip compares actual system runs with the expected model. Pip can find both structural and performance bugs. It works by feeding it application traces, which it uses to reconstruct a behavior model. The expected behavior is input to the Pip checker and the checker returns unexpected events.

Expectations are specified by recognizers and aggregates. Recognizers validate or invalidate individual execution path instances by essentially mapping paths to sets. Recognizers are written in a domain-specific language. Aggregates assert properties of sets, essentially modeling aggregated expectations on sets of paths. Since manual work for writing expectations can be too much, Pip can generate expectations automatically from the application's behavior model. Also, Pip offers a tool for visualizing paths, execution timelines, and performance that helps discover graphs of unexpected behavior. Using Pip, the authors managed to find several bugs in FAB and SplitStream, and by reusing Pip they checked that they did not reappear after the fix.

Timothy Roscoe from Intel Research asked whether the authors could get some theoretical guarantees from using other parsers or model checkers. The speaker replied that using other parsers would not be good to express behavior. Emil Sit from MIT asked where could he run Pip feasibly. The answer was that in practice Pip can reasonably handle about 5 million events. Eric Eide from the University of Utah asked whether Pip's domain-specific language was suitable for expressing the details of the application's programming model. Patrick replied that their language is not bound to any specific programming model. They use messages and event handlers that span different models.

WIDE-AREA NETWORK SERVICES II

Summarized by Ansley Post

■ ***OASIS: Anycast for Any Service***

Michael J. Freedman, New York University and Stanford University; Karthik Lakshminarayanan, University of California, Berkeley; David Mazières, Stanford University

Micheal Freedman presented an infrastructure for anycast that can support a wide variety of systems. In general, OASIS tries to find the best replica among many to provide a particular service. Currently, most users must manually select a replica of a service to use, based on their current geographic location. OASIS seeks to improve upon this by automatically finding the best replica for a user. Once an infrastructure for finding the best replica is in place, it can be used by many services and will not have to be reimplemented for each service.

To implement the OASIS service there were several possible solutions that trade off efficiency versus accuracy. One possible solution that is accurate but very costly is pinging the host making the request from all points where the service is provided and then choosing the one with the lowest latency. The authors instead chose to probe each IP prefix, using the insight that 99% of /24 addresses are in the same location. This is a good solution, which does not require probing each requesting host.

The OASIS system consists of two types of nodes: a large set for measurement and a small reliable core to provide the anycast service. A client contacts a resolver, which uses the measurements gathered to direct the request to the best replica given a certain anycast policy specified by the service. The reliable OASIS core uses gossip to disseminate information about failures and the services being provided.

OASIS has been deployed on PlanetLab for six months and is currently used by numerous services. OASIS can potentially provide a large performance improvement and can be easily integrated into an existing service. Other metrics for the best replica can be used, such as load. To configure a service for use with OASIS, the developer must provide a service description, a proxy (in this case PlanetLab), and a set of statistics, such as load on each replica.

■ *OverCite: A Distributed, Cooperative CiteSeer*

Jeremy Stribling, MIT Computer Science and Artificial Intelligence Laboratory; Jinyang Li, New York University and MIT Computer Science and Artificial Intelligence Laboratory via University of California, Berkeley; Isaac G. Councill, Pennsylvania State University; M. Frans Kaashoek and Robert Morris, MIT Computer Science and Artificial Intelligence Laboratory

This talk was given by Jeremy Stribling, who showed a system, OverCite, that is a distributed and cooperative version of the popular CiteSeer service. Citeseer is a repository of computer science research papers and metadata. The service is currently run on two servers by the University of Pennsylvania. These two servers are often overloaded by the requests that it has to service, often resulting in the system being unavailable to people trying to use it. In the current architecture it is hard to add new resources.

The solution presented is to build a distributed version of CiteSeer, OverCite, which is cooperative and to which new resources can easily be contributed for running the system. The goal of this system is to leverage the parallelism of CiteSeer and for each site that is running the service to have a low resource and administrative burden. It works by distributing the responsibilities of the CiteSeer system to a large number of nodes. These responsibilities include storing research documents and the metadata about the documents, responding to search queries, and crawling the Web for new documents.

To accomplish this, a DHT is used to store all documents and metadata. Documents are partitioned into groups, and hosts in the system are assigned to these groups. Search queries are directed to the right group for service. Some time was spent selecting the right grouping factor. This factor is a tradeoff between latency and the level of parallelism possible. Crawling activity is coordinated using the DHT. OverCite is built upon DHash, Searchy, OKWS, and the OASIS system discussed in the previous talk. It is currently running on the MIT Ron testbed plus some private nodes. The authors hope to make the deployment publicly usable in the near future. Additionally they are planning an open API to the data to allow the creation of new applications on top of OverCite.

■ *Colyseus: A Distributed Architecture for Online Multiplayer Games*

Ashwin Bharambe, Jeffrey Pang, and Srinivasan Seshan, Carnegie Mellon University

Colyseus is a system for building massively multiplayer online games (MMOGs). MMOGs have grown exponentially in the past decade; they have scaled because the games often employ certain “tricks” such as partitioning the world onto different servers, keeping many replicas of the gaming universe, or maing slow-paced games. First-person shooters (FPSs) are fast-paced games and as such no MMOG with a large number of players currently exists. The goal of this work is to build a cooperative architecture for FPSs that is very scalable.

A game in Colyseus is broken into components that are immutable, such as the map of the game world, and pieces that are mutable, such as the players and weapons. For each frame of action the game state is updated by a function. If the objects are partitioned across servers this function can be run in parallel. The problems that must be solved to do this are object discovery and replica synchronization. To do this in Colyseus each object has primary and secondary replicas. These replicas are kept weakly consistent; inconsistency is tolerable as this is often present for a short time in decentralized games. Object discovery is accomplished by every object publishing its position. Queries are then made for all objects in a particular area. Several optimizations allow this to be done efficiently, and these are presented in the paper. An example of such an optimization is the leveraging of the physics of the game world to predict which objects can be prefetched.

The experimental evaluation of Colyseus shows both the bandwidth required for running an FPS game and the consistency that is achieved. The bandwidth overhead is 4–5 times higher than a server-based game but still feasible and scales well with number of nodes. The view is consistent to within 0–0.8 objects. These results indicate that building a game on top of Colyseus is indeed feasible.

END-SYSTEM DESIGN

Summarized by Guy Lichtman

■ *Na Kika: Secure Service Execution and Composition in an Open Edge-Side Computing Network*

Robert Grimm, Guy Lichtman, Nikolaos Michalakis, Amos Elliston, Adam Kravetz, Jonathan Miller, and Sajid Raza, New York University

Nikolaos Michalakis presented Na Kika, an architecture for scaling dynamic content with a focus on collaborative efforts. Dynamic content is popular and easy to build but hard to scale. Examples include popular mashups such as zillow.com. Current platforms do not address small independent content producers and present a clear tension between extensibility and securit, which Na Kika tries to reconcile.

The architecture uses DNS to direct clients to nearby proxies that are organized in a structured overlay. Scripts are published like static content and executed on the proxies. The programming model is based on scripted event handlers. OnRequest and onResponse handlers are used for producing requests and responses, respectively. Service modularity is achieved through the descriptive nature of HTTP messages. Handlers are selected based on HTTP message properties and execute a most specific match. Composition is achieved through putting event handler pairs, one after the other through a nextStage predicate. The same mechanisms used for event handler selection and composition are used to enforce security admission and emission control through two additional stages, separating the client and the server from the Na Kika pipeline. Hosted code is contained through script sandboxes, but this doesn't prevent scripts from overusing memory and cpu. Na Kika uses an approach of controlling consumption only under congestion. No hard quotas are used. Throttling is used to reject requests and, as a last resort, terminate active requests.

Evaluation included using the Wise-MD application, a Web-based education tool developed at the NYU medical school. It is spread across multiple universities across the world and is multimedia intensive, with both static and dynamic content. One developer ported Wise-MD to Na Kika in 2 days. Simulated traces with Na Kika cold cache and Na Kika warm cache were executed and the results were compared to those of a single server: Na Kika both cold and warm cache clearly outperform a single server in terms of failure rate and bandwidth seen by clients. For dynamic content as well, the response time is significantly lower. Additionally, Nikolaos provided four examples demonstrating Na Kika's extensibility and how building extensions is easy and scalable. Each extension took less than 100 lines of code to implement using Na Kika.

■ *Connection Conditioning: Architecture-Independent Support for Simple, Robust Servers*

KyoungSoo Park and Vivek S. Pai, Princeton University

Vivek Pai started by stating that, overall, server performance is great, owing to many OS contributions, Moore's law, and load balancers, but that poor server performance is often seen in CGI, db, and network. There has been a lot of research in building server software in the past 10 years, including async IO, events, and so on. But this hasn't mattered too much. Apache is still the most popular server (with a 75% market share). Users love multi-process servers, as they make it easier to add features and modules. Pai makes a case to go with the flow and bring research benefits to Apache. He points out some economics: a Walmart Linux machine costs $400, an HP DL320 is under $3000, but a 100 Mbps WAN runs $30,000/month. The main conclusion here is that hardware costs are dwarfed by network costs.

Connection Conditioning's new approach is to make servers simpler, more robust, and easier to program, defend, and share. They might be a little slower, but that's OK, since hardware is cheap. Pai suggests doing this by using an old idea, UNIX pipes, which are good for text processing but aren't used for servers. In Connection Conditioning (CC), instead of having clients talking to the big server, the server instantiates filters. The clients talk to the filters, which bundle the requests and pass them on between the filters to the servers. Filters are separate processes; thus you can write them however you like, using threads, processes, or events. Communication is done via UNIX domain sockets, thus allowing passing of the socket/request bundle. The server sees TCP sockets, and responses are sent directly via the client socket. The main idea behind the filter design is that the inbound path is lightweight and the outbound path is heavyweight; thus, filtering the requests and passing the response socket works well.

Experience shows that modifying Apache takes less than 50 lines of change to add support for CC filters, and flash takes around 30 lines. Additionally, writing a new server that is designed to take advantage of this library is easy. A sample server, CC-Server, took about 80 lines of code. CC-Server handles non-parallel processing and is a proof-of-concept server which isn't meant to replace Apache. Performance tests using a single file test and a file mix similar to the SpecWeb99 static content benchmark show that CC-Server performance is at least comparable to Apache's. Robustness tests applying incomplete connections show that Apache dies at around 150 connections per second, whereas CC-Apache stays steady around 2000 requests per second.

■ ***PCP: Efficient Endpoint Congestion Control***

Thomas Anderson, Andrew Collins, Arvind Krishnamurthy, and John Zahorjan, University of Washington

Anderson began by saying that TCP is known to be suboptimal. The basic thrust of PCP is to take a systems approach to the problem and optimize it for the common case. TCP starts slowly and increases bandwidth until loss is detected. But most transfers are small to moderate and thus stay at the wasted capacity stage. From a systems point of view, if we could control the network we could easily build functionality into the network to solve the problem. As an example, consider ATM rate control, where the network indicates the safe rate to transmit. The real problem is that it is not trivial to change the network. Anderson raises the following questions: Can we accomplish something that is as good without changing the network? What is the best that we can do?

PCP uses the assumption that endpoints can cooperate. PCP does well both with and without fair queuing in the middle of the network as compared to TCP, which performs worse with fair queuing systems. PCP's approach is to directly emulate the behavior the router would use if we had control of the underlying network. The basic idea is to query the network for what rate it can support. Its main goals include minimal transfer time, negligible packet loss, work conservation, stability under extreme load, and fairness. TCP achieves only the first three goals.

Performance evaluation measured PCP versus TCP on US RON nodes. PCP outperforms TCP, since if there is a loss TCP goes through a painful recovery process that takes time to recover from. Even for 4 PCPs it outperforms TCP. The tests show that PCP isn't causing TCP back-off. Simulation shows that PCP actually does better at fairness than TCP, as packet losses are not a good way to achieve fairness. Even in a fair-queue router PCP does very well compared to TCP.

MEASUREMENT AND ANALYSIS

Summarized by Guy Lichtman

■ ***Availability of Multi-Object Operations***

Awarded Best Paper

Haifeng Yu and Phillip B. Gibbons, Intel Research Pittsburgh; Suman Nath, Microsoft Research

Haifeng Yu started by saying that placing objects for multi-object operations is hard. There are various ways to make assignments. The paper's main contributions include:

1. Identity assignments as a critical design choice.

2. Operations classified as either strict or loose, using two strategies, RAND (random) and PTN (partition).

3. Proposed design to approximate PTN under dynamic settings.

4. MOAT implementation and evaluation.

Some existing systems use assignments that are close to PTN, such as Glacier. Glacier becomes PTN only if nodes are evenly distributed on the ring. Existing systems similar to CAN (GFS, FARSITE) use RAND assignment.

Comparing RAND and PTN shows that they give dramatically different behaviors. There is a crossing point very close to the 1.0 fraction of objects needed for the operation to succeed. All other assignments fall between RAND and PTN. At crossing point RAND is better than PTN. This allows classifying to types of operations: loose and strict. Theoretical results show that PTN works best among all assignments for strict, and likewise RAND for loose. Haifeng proposed Group DHT to solve the limitation of Glacier. He further presented MOAT, a wide-area object storage system. It supports a range of assignments.

PlanetLab and simulation were used for evaluation with a TPC-H benchmark workload. Glacier behaves similarly to Chord, but Group DHT outperforms Glacier. The difference between assignments can be three 9s. Object assignments have critical impact on the availability of multi-object operations. It is not about "concentration" and "spread." The results show that it is impossible to combine best assignments for both strict and loose.

■ ***Subtleties in Tolerating Correlated Failures in Wide-area Storage Systems***

Suman Nath, Microsoft Research; Haifeng Yu and Phillip B. Gibbons, Intel Research Pittsburgh; Srinivasan Seshan, Carnegie Mellon University

Suman Nath stated that traditional replication techniques assume failure independence. In practice, failures are not independent. Correlated failures include DoS attacks, viruses, etc., and hurt availability dramatically. Techniques to fight against correlated failures include predication, modeling, and overprovisioning.

Suman presented how they found that existing failure pattern prediction techniques give negligible benefits. Additionally, simple failure models have dramatic inaccuracies, and overprovisioning doesn't help much, either. The study is based on three failure traces: Planetlab, Web server, and RON trace. The traces are long and thus capture rare correlated failures. The study looked at several ways of mitigating correlated failures, including failure pattern prediction using introspection (OceanStore), modeling correlated failures (used in Glacier), and using smaller fragments. Suman designed and implemented IrisStore, a distributed storage system similar to OceanStore. The system is based on the design principles learned in this study and show that during the SOSP 2005 deadline PlanetLab experienced numerous failures but still IrisStore maintained over 90% availability.

■ ***Open Versus Closed: A Cautionary Tale***

Bianca Schroeder, Adam Wierman, and Mor Harchol-Balter, Carnegie Mellon University

Schroeder started out with a motivation example on scheduling static Web requests and a question: Is it possible to improve response time by better scheduling of requests to Web servers? To try to answer this question she proposed an implementation of SRPT (shortest remaining time) scheduling and used workload generators to test the idea. Two generators were used. One had a tuning knob for request rate; the second had a tuning knob for number of users and think time. These generators brought different results for plain Apache. Using the TCP-W benchmark with one generator, SRPT gave a huge improvement; with the other generator, a small improvement.

These differences arise because there is a fundamental difference between open and closed systems used for simulation and generation. Closed systems can be viewed as user driven. Each user resides in an endless loop of sending a request, then waiting during a think time period, then repeating the cycle. This differs from the open system model, where the user sets the request rate. In an open model there is no maximum number of simultaneous users. There are popular generators that use both models, but very little documentation and literature about these models exist. An analysis of both systems shows an order of magnitude difference in the response times. Open systems have a higher response time. Variability affects open systems much more than closed ones. This is because of the inherent dependence between arrivals and completions in a closed system. In some sense a closed system reduces burstiness.

A hybrid option, which is equivalent to open arrivals of session and then a closed stream of requests per session, would probably be more suitable. To analyze what real Web workloads model, we can look at the number of requests per visit. The more requests per visit, the closer it is to a close system. There is of course an area in between, for which the hybrid option is more suited.

ARCHITECTURES AND ABSTRACTIONS (AND EMAIL)

Summarized by Swapnil Patil

■ ***An Architecture for Internet Data Transfer***

Niraj Tolia, Carnegie Mellon University; Michael Kaminsky, Intel Research Pittsburgh; David G. Andersen and Swapnil Patil, Carnegie Mellon University

In the first talk of the session, Tolia presented a new architecture, DOT, for Internet data transfers. Current applications have tightly coupled content negotiation and data transfer; DOT separates the two and introduces a service to handle all data transfers. DOT provides a modular, plug-inbased architecture that helps applications easily use new innovations in data transfer. DOT currently supports three plug-ins: for storage, multi-path transfers, and portable storage devices. DOT uses content-based techniques for chunking-based data transfers and application-independent naming semantics. As a part of the evaluation, DOT has been used for file transfer application, production mail server (Postfix), and multi-path transfers. DOT performs with a negligible overhead, and integration of DOT in the Postfix mail server required less than 200 lines of code. Additional information about the project is available at http://www.cs.cmu.edu/~dga/dot.

There were several questions. What kind of security does DOT support? DOT will use convergent encryption for individual chunks and their hashes. This is a work in progress. Does the sender have the flexibility to decide what chunking policy to use? Yes, as long as the sender and the receiver agree on the chunking policy. How does this compare with LBFS chunking? Like LBFS, DOT uses Rabin fingerprinting for chunking.

■ ***OCALA: An Architecture for Supporting Legacy Applications over Overlays***

Dilip Joseph and Jayanth Kannan, University of California, Berkeley; Ayumu Kubota, KDDI Labs; Karthik Lakshminarayanan and Ion Stoica, University of California, Berkeley; Klaus Wehrle, RWTH Aachen University

This talk, by Dilip Joseph, was motivated by a need for legacy applications to support the new network architectures, services, and overlays. OCALA (Overlay Convergence Architecture for Legacy Applications) provides an overlay-agnostic approach for legacy applications and as a proxy for incremental deployment. OCALA adds a new Overlay Convergence (OC) layer below the transport layer in the network stack. This OC layer replaces the network layer and has two components, overlay-dependent and overlay-independent sublayers. This helps users on a regular IP network communicate through the overlay. In addition, OCALA can enable applications running on the same machine to use different overlays and stitch different overlays to help users of these overlays to communicate with each other. OCALA has a very extensible architecture, making it easy to implement a new overlay network. At the end, Dilip gave a quick demo of their techniques by accessing his home machine, behind a NAT, through the OCALA proxy. OCALA is available for download at http://ocala.cs.berkeley.edu/.

■ ***Distributed Quota Enforcement for Spam Control***

Michael Walfish, J. D. Zamfirescu, Hari Balakrishnan, and David Karger, MIT CSAIL; Scott Shenker, University of California, Berkeley, and ICSI

In this talk, Michael Walfish presented the design and implementation of a quota-based spam control system. The basic goal is to control sent messages by making it more expensive to send bulk mail. An allocator gives each sender a certain quota of mail messages that it can send. Every message the sender sends has a stamp, which is verified by the receiver. On getting email, the receiver checks whether the stamp is valid. The receiver does this by asking the quota enforcer if this stamp has been seen before. If it is a new stamp, then the email is valid and the enforcer stores the record for this stamp. If this stamp was seen before, then the mail is discarded as spam. The main challenge is the design of the enforcer to store billions of stamps, tolerate Byzantine and crash faults, and provide a fast stamp lookup. The evaluation showed that the system can easily handle over 80 billion messages per day.

People asked a couple of questions at the end of the presentation. How do the micropayments work in distributed quota enforcement? The details about micropayments are in the paper. Can probabilistic marking be used, instead of providing such a large infrastructure? If you want to avoid infrastructure costs, then probabilistic marking (or any other technique) can be used.

■ ***RE: Reliable Email***

Scott Garriss, Carnegie Mellon University; Michael Kaminsky, Intel Research Pittsburgh; Michael J. Freedman, New York University and Stanford University; Brad Karp, University College London; David Mazières, Stanford University; Haifeng Yu, Intel Research Pittsburgh and Carnegie Mellon University

The final talk of the session, by Michael Kaminsky, described another proposal to control spam. This work proposes a new whitelisting system that works on social networking by exploiting "friend-of-friend" relations among email correspondents. This technique automatically populates the whitelists by using attestations among different users. For example, suppose A and B know each other, B and C are friends, but A and C don't know each other. Using the fact that A knows B is not a spammer and B knows that C is not a spammer, then A may conclude that C is unlikely to be a spammer. These attestations are kept at the server and are checked when any mail is received from a sender. In addition, RE uses cryptographic private matching techniques to preserve the contacts and whitelists. Using email traces, the evaluation showed that RE can accept about 75% of received email and can prevent up to 88% of the false positives incurred by existing spam filtering.

In the questions that followed, someone asked why the results show that you lose 13% of email. This happens because the one-hop social network does not have enough whitelists. How many hops of social network can work? You can have any number of hops. This increases the whitelist size, which will in turn take more time for verification. What happens if a spammer gets access to the address-book whitelist? Currently, RE does not handle this threat model.

WIRELESS AND SENSOR NETWORKS

Summarized by Asad Awan

■ *PRESTO: Feedback-driven Data Management in Sensor Networks*

Ming Li, Deepak Ganesan, and Prashant Shenoy, University of Massachusetts, Amherst

Ming Li presented PRESTO (PREdictive STOrage), an energy-efficient data management architecture for query processing and data acquisition in sensor networks. PRESTO architecture capitalizes on cooperation between resource-rich proxies and resource-constrained sensor nodes by splitting system intelligence between these two network tiers. Li demonstrated this key design principle of PRESTO by discussing the pros and cons of the sensor-centric and proxy-centric architectures. Sensor-centric architectures achieve greater energy efficiency (through in-network processing) and accuracy while sacrificing query-response latencies and introducing complexity at sensor nodes. The proxy-centric approach pushes intelligence to the sensor network edge, thus reducing query latencies, while trading off energy efficiency or data accuracy.

PRESTO achieves low query latencies by replying to queries using a data cache of proxies on the network edge. Energy efficiency and data accuracy result from exploiting the predictable structure of sensor data. Finally, proxy-to-sensor feedback allows the system to adapt to data and query dynamics. The model-driven push component of PRESTO captures deviations in sensed data relative to the prediction model. Sensor nodes only transmit anomalies, thus reducing communication costs and providing an accuracy bound on data cached at proxies. However, if the cached data is not sufficient to respond to a query, PRESTO acquires data using a model-driven pull of sensor data. Data from push/ pull operations is refined using interpolation and reprediction to maintain high cache accuracy. The proxy observes acquired data for trend changes and provides feedback to adapt the data model used at sensors. Similarly, the system adapts to query dynamics using feedback in the form of tuning parameters to sensors.

PRESTO uses an Autoregressive Integrated Moving Average (ARIMA) to model the time series of observations, allowing extraction of both long- and short-term data trends. Further, the asymmetric processing requirements of ARIMA (with computationally intensive model training and low-processing cost data prediction) fit well with the computation split between resource-rich proxies and resource-constrained sensor nodes. Li concluded the talk with performance evaluation results, which showed the efficacy of PRESTO in terms of communication costs, accuracy, and query latency.

■ *Practical Data-Centric Storage*

Cheng Tien Ee, University of California, Berkeley; Sylvia Ratnasamy, Intel Research, Berkeley; Scott Shenker, University of California, Berkeley, and ICSI

Cheng Tien Ee presented PathDCS, an approach that enables data-centric storage (DCS) in sensor networks without requiring point-to-point routing. Ee explained that queries over sensor networks require locating data pertaining to observed events. In the DCS approach, data identified by its content is stored at a specific location. Any query for this content is routed to this location, reducing messaging overhead. DCS is an optimal solution (under certain conditions; cf. Ratnasamy's GHT paper). DCS requires a location reference system and a data-to-location mapping. Ee explained that the current DCS approaches rely on point-to-point routing, which has significant performance overheads. Existing approaches (e.g., GHT) elegantly use a geographical reference system. However, the need to account for (geographical) network boundary and coverage holes in such approaches has yielded complex solutions. PathDCS provides a simple and scalable solution. It builds on a widely used routing primitive, namely, tree routing.

Using an animation, Ee explained the fundamentals of the PathDCS algorithm. PathDCS defines storage location using beacon nodes acting as reference locations. Trees rooted at each beacon are built. The PathDCS algorithm selects a data storage node based only on the identifier of the data and the beacon node ids and does not depend on the location of the source of the data. Data routing simply uses the existing tree paths. Ee also described how beacons to be used in PathDCS can be elected. Ee explained that to handle failures and load balancing, one-hop neighbors of beacons take over. The key intuition here is to minimize the changes in path to the beacon and hence to the storage locations. Further, PathDCS relies on refreshing data to account for path (and storage location) changes. Simulation and system-based evaluation results were presented. The results showed that PathDCS achieves a good load balance, has high route completion and data lookup success, and is robust in failures.

Walsh asked how nonuniformity of data events would affect the load balance of the system. Ee replied that for such a scenario, balancing keys and local replication can be used. Michalakis commented that one-hop replication might not be a good idea, because failures may be geographically correlated. Ee replied that the paper focuses primarily on routing, not storage. The use of one-hop replication is thus to increase lookup success by considering the effects of path fluctuation. Other mechanisms to improve replication can be used and are complementary. Ganesan inquired how the system design would change if the network were heterogeneous. Ee replied that since beacons require more resources, powerful nodes can be used as beacons. They can also be used to cache data to reduce load for the rest of the network.

■ *Geographic Routing Without Planarization*

Ben Leong, Barbara Liskov, and Robert Morris, MIT CSAIL

Ben Leong presented the Greedy Distributed Spanning Tree Routing (GDSTR) algorithm. In geographic routing, packet destinations are specified with x-y coordinates. Packets are first forwarded greedily, that is, to the neighbor closest to the destination. However, greedy forwarding causes a packet to end up at a dead end, where all the neighbors of a node are further from the destination than it. Leong explained that the existing approach is to forward the packets around the voids in the network on the faces of a planarized version of the connectivity graph. It turns out that planarization is difficult to achieve in a distributed environment and was only recently solved by Kim et al. using CLDP. Leong commented that the complexity and overheads of this scheme motivated GDSTR, which does not require planarization.

GDSTR routes packets around voids by routing on an annotated spanning tree, called a hull tree. Convex hulls are used to aggregate information about the points that are reachable along different branches of the tree. By forwarding packets up the tree (i.e., decreasing levels of the tree) the packet can be forwarded to an appropriate subtree that contains the destination. A key decision in routing a packet around a void is to choose a forwarding direction (clockwise or anticlockwise). A good forwarding direction choice yields fewer hops to the destination. Leong explained that GDSTR uses two extermal-rooted trees to “approximate” a void, and the simple heuristic of choosing the tree with a root that is closer to the destination is often equivalent to choosing the optimal forwarding direction. GDSTR guarantees packet delivery in a connected graph. Simulation-based comparisons among GPSR, GOAFR, GPVFR, and CLDP showed that GDSTR outperforms the other approaches, in terms of hop stretch, by a large margin when the average node degree of the network topologies is low (i.e., when voids are large). Results for GDSTR with different numbers of hull trees around voids showed that using two trees achieves significant improvements in hop stretch performance over using only one tree. Having more than two trees yields marginal benefits. Leong commented that the computation and memory overhead of GDSTR are also low. Similarly, the bandwidth overhead is an order of magnitude lower for GDSTR than for CLDP, while routing performance of GDSTR is up to 20% better.

In response to a question from the audience about the effect of overlapping convex hulls on routing performance, Leong responded that routing performance will be worse if there are intersecting hulls, since the size of the search tree will be increased. However, this is only if the destination falls within the intersection of the hulls.

FILE AND STORAGE SYSTEMS

Summarized by Andreas Haeberlen

■ *Virtualization Aware File Systems: Getting Beyond the Limitations of Virtual Disks*

Ben Pfaff, Tal Garfinkel, and Mendel Rosenblum, Stanford University

Ben Pfaff proposed a new storage solution for virtual machines (VMs) that combines the benefits of virtual disks and distributed file systems. Virtual disks provide useful features such as rollback and versioning, which allow users to specialize VMs for different tasks. However, these features are only available at disk granularity, which makes it difficult to roll back individual files, and virtual disks cannot be shared among multiple VMs. Distributed file systems offer both fine-grain access and sharing, but they lack other features such as versioning and specialization. Ben argued that we can get a better storage solution by combining the two; the result is a virtualization-aware file system.

In the second part of his talk, Pfaff presented a prototype system called Ventana. In Ventana, virtual disks can be recursively specialized, which results in a tree of versions. Multiple VMs can share a branch of this tree, so that changes by one VM become visible by all others; also, there is a mechanism that can be used to restrict access to certain branches. To interact with virtual machines Ventana uses the NFS protocol, which is understood by most operating systems. Ben concluded his talk by discussing several use cases, for example, how to use the branch hierarchy to apply a security patch efficiently in multiple virtual machines.

■ ***Olive: Distributed Point-in-Time Branching Storage for Real Systems***

Marcos K. Aguilera, Susan Spence, and Alistair Veitch, HP Labs

The talk was given by Marcos Aguilera, who argued that a widening gap between storage capacity and transfer rates makes it increasingly difficult to handle large volumes of data. For example, an administrator may want to archive a snapshot of a volume for further reference, or run a "what if" installation of a new software package without affecting the main copy. Aguilera presented Olive, a distributed and replicated storage system that addresses these problems by providing an efficient branching operation. By creating a new branch, the user obtains a second copy of a volume which can evolve independently from the first.

Aguilera pointed out that the main technical challenge in Olive was to provide strong consistency, and he described the mechanisms Olive uses to achieve this. Specifically, Olive provides linearizability, which implies that the state captured by a branch is one that could also have resulted from a crash. Aguilera also presented evaluation results from an implementation of Olive in the federated array of bricks; he showed that a new branch can be created in tens of milliseconds and that the per-branch metadata is small enough to allow dozens of branches.

■ ***Pastwatch: A Distributed Version Control System***

Alexander Yip, Benjie Chen, and Robert Morris, MIT CSAIL

In the last talk of the conference, Alexander Yip presented Pastwatch, a cvs-style version control system that supports disconnected operation. In Pastwatch, users do not have to be connected to a central server to commit changes; for example, a small group of developers can use it to collaborate while on an airplane and later merge their changes into the main repository when they are connected to the network again. Of course, this can result in write conflicts if multiple disconnected users modify the same file. Pastwatch handles this by lazily creating branches, which are visible to the users and can be merged later.

Each Pastwatch user maintains his or her own local copy of the repository, which is organized using a special data structure called a revtree. The revtree structure is such that two repositories can be synchronized simply by forming the set-union of the revtree nodes. This allows updates to spread in an ad-hoc manner and yet ensures eventual consistency. Pastwatch has been in production use for over a year, and an implementation is available for several major operating systems. More information is available at http://pdos.csail.mit.edu/pastwatch/.

In the Q&A session, Yip fielded several questions regarding how Pastwatch handles write conflicts. Brad Karp asked how users could find out about new branches; the answer was that Pastwatch displays an explicit warning during synchronization. Eric Eide asked what would happen if two users reconciled the same branches; the answer was that Pastwatch would create another branch, but that this had rarely been observed in practice.

FreeBSD Developer Summit and BSDCan

Summarized by Rik Farrow

On May 10, I headed off to Ottawa, Canada, for a several-day adventure with the three BSD communities. BSD, which started off as the Berkeley Software Distribution when Bill Joy arranged to ship out nine-track tapes containing assorted software (such as vi and csh, which he wrote, and sendmail), has forked twice into three groups. FreeBSD, the largest community, focuses on building a mainstream server/network operating system, with multiprocessor support. NetBSD, the next largest community, specializes in porting the BSD operating system to as many target CPUs as possible. Currently, 59 CPU architectures are supported. OpenBSD, a fork from NetBSD, is best known for its focus on improving security.

I caught the second day of the FreeBSD Developer Summit, an invitation-only meeting of about 50 developers. Eight long talks were packed into a long day, with a pub trip for lunch. Having a pub break somewhat disturbed my note-taking ability, but I will provide you with an overview of the talks, as well as some links if you want to search deeper.

The Developer Summit is a chance for FreeBSD developers to meet in person to catch up on the status of projects and plan for future work. Another key aspect of the summit is the chance for developers to meet each other in person—something that's especially important given the limitations of electronic communication.

The morning began with Dario Freni and Scott Ullrich discussing a LiveCD version of FreeBSD, called FreeSBIE (www.freesbie.org). You can use FreeSBIE like Knoppix, a popular Debian Linux–based LiveCD; that is, you can boot from the CD and use FreeBSD without installing anything on your hard drive. The developers described how the image can be made small enough to fit on a business-card-sized CD (8 MB), using a new toolkit called sysutils/freesbie, and to create other purpose-built CDs using FreeSbie.

Next off, Colin Percival described his Update 2.0 project. Percival recently became the FreeBSD security officer. Updating a FreeBSD system currently involves either collecting new sources and performing a make world in /usr/src or installing from scratch (which does ensure a clean upgrade). The Update 2.0 system supports installing binary security patches, making installed FreeBSD systems easier to maintain. For now, the system only works with security patches. Apple and the Mozilla Foundation use a version of Update v1. Part of making the binary update system officially supported involves moving it to a formal project infrastructure rather than using ad-hoc systems he's assembled previously, and part is making it a tool that can be reused by administrators to deploy their own updates, not just the security updates.

The KAME project involved the creation of a reference IPv6 and IPSec implementation for BSD operating systems in general. KAME began with Japanese researchers (http://www.kame.net/project-overview.html), and the IPv6/IPSec implementation has been merged into FreeBSD since 4.0. The KAME project has not supported the FreeBSD SMP architecture introduced in FreeBSD 5.0, so the FreeBSD stack has become the authoritative source of general IPv6 code.

Robert Watson spoke for the first of many times. Watson (http://www.watson.org/~robert/) has added auditing capabilities to TrustedBSD, a version of FreeBSD. The auditing support is based on Apple's audit implementation as found in Darwin, and it uses the same format as Sun's BSM, as there are already tools available for perusing those audit records. Audit records of this type refer to secure operating systems in the tradition of the Orange Book, and now the Common Criteria.

He has also decided to add NFSv4-style ACLs to the existing POSIX.1.e-style in TrustedBSD, and he hopes that Kirk McKusick will implement backup and restore support for ACLs. The decision to retrofit/update the MAC framework is based on four years of deployment experience, which ends up being mostly cleanup, since there are a number of companies shipping with the framework and they are, in fact, generally happy with it.

Watson also talked about needing to reduce the number of firewalls supported in FreeBSD from four to three (really!). Ip6fw will be eliminated, as ipfw now has full IPv6 support. The other firewalls supported in FreeBSD are pf and ipfilter.

Sam Leffler picked up after lunch. Leffler writes code for 802.11 infrastructure support for various wireless devices. Although many of the devices are Linux-based, Leffler prefers to begin working in the FreeBSD programming environment. Some of the work Leffler has done has not been folded into the system, because programmers need to modify existing drivers so that they will work with the extensions he has written for wireless roaming, repeaters, virtual APs, and WDS (Wireless Distribution System).

Requests for someone to take responsibility for some part of the kernel code were not uncommon during this conference. Anyone can become a part of the FreeBSD community by contributing patches, even for documentation. The more time and useful patches or code you contribute, the more important you become to the community. From an outsider's perspective, this concept looks very appealing and straightforward.

Randall Stewart of Cisco spoke next about SCTP. SCTP appeared over five years ago as an alternative to TCP, and Stewart wrote both a reference implementation and a book about SCTP. SCTP sets out to solve many of the weaknesses of TCP and includes the ability to multiplex streams within a single connection. Although there are five new system calls involved with SCTP (and kernel support also in Linux 2.6 and Solaris 10), there are currently no FreeBSD man pages. But SCTP has been used in telephone applications in China, Cisco BGP implementations (because it has protection against RST attacks), SIP proxies, and satellite communications.

Robert Watson took over at this point, covering a myriad of topics very quickly. Besides his work in secure systems, Watson has been at the forefront in removing the Giant lock from the networking stack (see Michael Lucas's article in the October 2005 *;login:*). Watson explained where Giant had been removed, then mentioned that there were device drivers where Giant is still used (which hurts SMP performance). Watson called for people willing to rewrite some critical device drivers (which is not an easy task), and also for people willing to measure the performance of applications such as Apache and MySQL (LAMP) on multiprocessor servers.

Marco Zek spoke briefly about IP stack virtualization. Stack virtualization provides multiple network stacks, for class exercises, honeypots, and to use with jails. The combination of jail technology with stack virtualization provides a significantly lighter-weight implementation than other OS virtualization approaches such as Xen. At the end of his talk, Zek set up a demonstration where people in the audience could hook up to a virtual network he had created using a GUI-based, drag-and-drop tool via an access point attached to his laptop. Network traffic in the virtual network could be seen within the GUI.

The final session of the day covered the use of FreeBSD in embedded systems. Poul-Henning Kamp mentioned that there is a need for a flash file system, one that levels writes to flash to extend its lifetime, as well as improvements in gdb for debugging over a serial port. There is an embedded system mailing list, small@freebsd.org, for people interested in FreeBSD on embedded systems. Warner Losh spoke about work he has done porting FreeBSD to embedded systems, and he showed examples of hardware he had worked with.

The general observation was made that FreeBSD is being widely used in the embedded and high-end embedded spaces, but that FreeBSD developers need to do a better job of supporting that community. Particular targets are providing a better Web site and online community, providing reference hardware information for platforms such as ARM and MIPS, and working on improving bundling and targeting tools for embedded environments.

The formal sessions had ended at this point, but after dinner many FreeBSD developers gathered in the eighth-floor lounge in the residence hall (where many people were staying) for hacking into the wee hours. This went on at least until Saturday night.

BSDCAN

Dan Langille began organizing the BSDCan conference several years ago. The USENIX Association was one of two large donors in 2006 and helped to pay the expenses of the speaker travel at this three-tracked conference (http://www.bsdcan.org/2006/). The conference is held at the University of Ottawa in classrooms in the SITE (computer science) building. The low cost of the conference, preconference tutorials, and on-campus accommodations help make this conference popular with those with small budgets. There were 193 attendees this year.

Langille began by explaining where to eat (pubs) and that there would be wireless access as soon as the University of Ottawa provided a route and a hole in their firewall, and then he announced other BSD gatherings. There will be a EuroBSDCon in Milan, Italy, sometime in the first half of November 2006, an AsiaBSDCon in Japan in March 2007, then another EuroBSDCon in Copenhagen in December 2007.

As there were three tracks and only one summarizer (me), all I can do is share some of the notes I took from the sessions that I attended. I first listened to Russell Sutherland of the University of Toronto talk about using FreeBSD in edge routers. He had tried using Linux because it includes policy-based routing, but he prefers FreeBSD's ipwf over iptables. Although the default route is to Internet2, he forwards traffic from dorms as well as commercial Internet traffic away from Internet2 (see Robert Haskins' article about packet shaping in this issue).

Poul-Henning Kamp (http://phk.freebsd.dk/) spoke next. Kamp is well known for his work with embedded systems and FreeBSD appliances. He showed pictures of multiple Soekris (soekris.com) boxes nailed up on the wall of his workshop, serving as firewalls, routers, and servers. Kamp explained that most disk drives were limited to room-temperature environments, making them unsuitable for use in very cold (or hot) environments. He discussed the use of flash memory instead of disks and explained that most flash memory expects to be used with FAT file systems. Kamp provided an interesting tip: that just tying a knot at the end of a cable run will act as a coil, similar to the ferrite coils you often see attached to device cables. Both the knot and the ferrite coils are supposed to damp down voltage surges by using inductance. Kamp also mentioned the use of nanoBSD, a stripped-down FreeBSD, for use in firewall appliances.

I next listened to Warner Losh talk about FreeBSD ARM running an ATMEL System on Chip (SoC). In the embedded-systems market, vendors will take a processor design like the ARM and pack as many devices onto the chip as possible. Losh mentioned that the ATMEL SoC he was working with reused the pins on the chips for multiple purposes: A set of pins could be used for serial, USB, or Ethernet, depending on what you wanted to do. His particular application was to build a small system that provided accurate timing signals (something Kamp is also interested in), and you can find his slides at http://people.freebsd.org/~imp/bsdcan2006/text0.html.

I've done some work with other embedded-systems programmers, and I asked Losh about using the JTAG interface, a serial interface used for debugging integrated circuits. Losh said he did not use it.

FREEBSD

In the next talk I attended, Robert Watson explained how FreeBSD works. I had often wondered about the various versions (currently 4, 5, and 6, with 7 to come sometime in the foreseeable future). The three active versions are all maintained, but new developments will only appear in the newest track, 7. The *stable* version means just that (it is considered stable and safe to use), whereas *current* means the latest build, ready for testing (terminology that I think takes some getting used to). In practice, you want a stable version, unless you want to find bugs and interesting corner cases. As a new release gets close to release, it goes through *code slush* (no new features), *code freeze*, *beta*, *release candidate,* and then finally the release itself.

The BSD License is one thing that distinguishes BSD from Linux, as it encourages commercial use. Another is how FreeBSD advances. There is a core group of nine developers, elected by committers, who can commit code into the CVS trees. The core group manages but does not control direction. What gets developed leads the way, although the core group does make the final decision about what gets committed. The core group also handles disputes and lends authority when things need to get done.

There are 346 committers, 185 of whom are ports committers (people who commit changes to the applications supported on FreeBSD). Currently, there are over 13,500 ports supported, an average of 73 ports per committer. There are also over 1500 ports maintainers, people who help with ports but cannot commit changes. Throughout the conference, there were mentions of parts of the kernel, drivers, and ports in need of a volunteer to maintain them. Consistent work on a project leads to becoming a committer, and this also includes work on documentation.

FreeBSD is also backed by the FreeBSD Foundation, which provides real support, including help with legal and licensing issues, hardware donations, and funds for the FreeBSD developers conference and some travel fees.

WIPS

Robert Watson introduced the work-in-progress talks, starting with Poul-Henning Kamp's Varnish project. Varnish is a Web-caching server designed for the needs of newspapers and other sources that change Web content quickly. Varnish can be loaded at any time, can have multiple configs (VCL language) loaded at once, can include conditionals and forwarding, can do this with clusters, has a command-line interface, and can pull up statistics on objects, logging to shared memory segment, and a logdaemon processes this shared memory in the format of your choice (Apache, custom, and real-time views). Varnish appears under a BSD license and is sponsored by Norway's biggest paper, Verdens Gang (www.vg.no).

Murray Stokely (murray@freebsd.org) spoke about the Summer of Code (SoC), which provides $4500 grants to students to work on coding projects. Google spent $2 million last summer (2005). Of the 400 applications for BSD, 20 were funded, representing half of last summer's SoC grants. At the time of the conference it was already too late to apply for 2006 SoC, but there is always a need for mentors. Stokely mentioned several related URLs: code.google.com, netbsd.org/contrib/projects.html, and freebsd.org/projects/summerofcode.html.

Jan Schaumann (jschauma@netbsd.org) discussed a medley of SoC projects, including bpg (licensed PGP for BSD), Apple's HFS+ support, NDIS driver support ported from FreeBSD, memory-based file systems (being ported *to* FreeBSD), userspace filesystem support, journaling for FFS, automated regression testing framework, zeroconf, and improving mbufs. Google will select the top 20 of 80 BSD-related SoC applications this year.

Christian S.J. Peron discussed his work with TrustedBSD, including auditing work targeted at the Common Criteria CC/CAPP. Kernel and OS parts are relatively mature, but lots of key userspace programs are not there yet. Lots of programs do not understand audit, so you can insert NO_AUDIT into make.conf to prevent it from being included in packages you build in the ports system. Login, ssh, logout, and other things were changed to support audit context. Peron worked from OpenBSM library to get the bits into place. He created a general-purpose audit submission mechanism to avoid code replication and made use of tokens for event type, subject, optional text token, and return value. Su needs to submit an audit trail; he wants to do this with sudo too (this might appear in the Apple branch). CAPP also requires user/group manipulation recording, audit records for system halts or shutdowns, and audit records of daemons that execute code in the context of other users, such as cron, at, and sendmail.

Csaba Henk (csaba.henk@creo.hu) talked about a userspace file system interface named FUSE. The current FUSE patches into the VFS, then requires a context switch to go to the userspace file system and another context switch to return, which is expensive. He used FUSE Linux (by Miklos Szeredi: fuse.sf.net), an easy-to-use API that helped in the port to FreeBSD (as part of SoC). The kernel component is being written from scratch to keep it in the BSD license, and because the Linux VFS structure is too different for a straight port. Henk mentioned that DragonflyBSD has a message-passing design and needs to use a userspace file system.

Colin Percival wrote FreeBSD Update, which led to bsdiff, which is used by Apple and Firefox for distributing updates. He is now rewriting Update, because the build code is very complicated currently. In the new version you select which version of code is to be updated, and Update will use source code and a set of patches rather than a CVS code tree. FreeBSD Update will become officially supported. People who build special versions on other platforms will be able to use either the older or the newer (2.0) version.

Warner Losh (imp@freebsd.org) talked about the state of processor ports for embedded-system support. With FreeBSD/ARM, the Core functionality is good, and work-in-progress includes the p4 tree, Cirrus Logic EP-9302, XScale improvements, AT91RM9200 Boot loader, and IIC (I2C) and MMC infrastructures. The Intel world is difficult to support because of nondisclosure issues. Losh said that we need to improve cross-building support and GDB protocol support.

Kip Macy also talked about work on supporting the new T1 Ultrasparc architecture, where there can be up to eight cores on a single chip, each supporting four threads. An advantage of the Sparc design is that it provides about the same performance as dual-Xeon processor systems at less than half the TDP (Thermal Design Power). But there is still work needed in getting the port stable, as the current sun4v version will panic with pmap races under 90% load. Volunteers are welcome, said Macy.

BOFS

After the last WiP, Langille announced that there would be a Postgres conference in Toronto in July (conference.postgressql.org). Then he listed six BoFs for the evening, including a Google BoF, an open cryptographic BoF, and a BSD certification BoF. I attended the latter and learned that the FreeBSD organization has spent a lot of time on, and gone a long way toward, producing a certification test for BSD sysadmins.

SATURDAY MORNING

As befits a conference where many attendees stay up late hackin or drinking, the first sessions began at 10 a.m. Saturday morning. I listened to Reyk Floeter (reyk@vantronix.net) discuss some features of OpenBSD support for wireless. He first explained that any connection could be deauthenticated, even when WEP was in use, as that portion of a wireless frame is not encrypted or authenticated. He went on to explain that OpenBSD hostapd had been used at What the Hack in Europe last summer, and he showed maps of the coverage. The OpenBSD hostapd apparently supports wireless roaming, which is interesting.

Floeter also talked about using the trunk interface in OpenBSD in failover mode. The trunk interface allows several network interfaces to be combined so that they perform as one, but with both higher throughput and failover. He then discussed improvements in OpenBSD IPSec implementation and support for the IEEE WLAN access protocol, 80211i/WPA2. He ended the talk with a plea to "Stop the Blob," a reference to using binary code blocks provided by vendors unwilling to provide documentation or support for open source projects. Stop-the-Blob T-shirts were on sale in the lobby of the conference building.

FREE BEER

I stayed in the same classroom so that I could hear Greg Lehey (grog@freebsd.org) talk about free beer. I thought this was a reference to BSD licensing (as in Poul-Henning Kamp's "free as in free beer" license) but this talk was really about brewing beer, using FreeBSD running on an old 386DX box, a set of relays, two temperature sensors, a light bulb, and a refrigerator to control the temperature of the future beer's wort while it is fermenting. So I not only learned a lot about beer history and brewing, I also learned about using FreeBSD as a control system. I could have also listened to Poul-Henning Kamp talk about his own FreeBSD control projects, which include many remote applications in environments with extreme weather, a talk that occurred in another track.

After lunch (at a pub), John-Mark Garner (jmg@freebsd.org) gave a presentation about writing device drivers in FreeBSD. Of course, you can't learn how to write device drivers in an hour, but Garner did a good job of providing an overview of the framework available. I finally learned what has happened to minor devices (made unnecessary because of devfs). Garner also talked about softc, a newer, more efficient framework for writing device drivers, about how resources (memory, IRQs, and ports) should be handled, and about bus probing and DMA.

Chris Buescher and Scott Ullrich discussed the various firewalls available in the BSD environment. BSD suffers from an embarrassment of riches here, and the presenters created a large chart, which you can find in their slides at pfsense.org/bsdcan/, to compare the features of the three firewall families, ipfw, ipfilter, and pf. They went on to explain the m0n0wall project, a version of FreeBSD stripped down for use in firewall appliances and controlled completely through the use of PHP over a Web interface (m0n0wall.ch). They then described their own project, pfSense (pfsense.org), where they forked their own version from m0n0wall because they wanted to create a firewall install that was much more featureful. Whereas m0n0wall is based on FreeBSD 4.1 for its faster network performance, pfSense uses FreeBSD 6.1, which has wireless networking support that FreeBSD 4.1 lacks. PfSense includes networking tools, such as tcpdump and HSFC traffic-shaping, borrowed from OpenBSD, and uses pf for firewall support, giving it the ability to do OS fingerprint–based blocking.

Dan Langille ended the conference by giving away books and T-shirts. Some books were given to people chosen randomly [by using `random()` to assign numbers to all attendees, then sorting] and for various feats. Someone won a book by spending over six hours trying to get through Canadian customs. (There was actually someone who had spent longer, but he had already won a book.)

HotMobile 2006: 7th IEEE Workshop on Mobile Computing Systems and Applications

Summarized by Maria R. Ebling, Program Chair

Like the first WMCSA, the goal of this workshop was to foster interaction among practitioners of mobile computing. In keeping with this goal, we decided to return with a small, informal workshop, one with few papers but significant discussions. We accepted just nine papers, but we had two significant group discussions, two exciting panels, and an insightful keynote address. Approximately 40 people attended the two-day event on April 6–7, 2006, at the Semiahmoo Resort, Washington, USA.

To reflect these changes, during the opening remarks the organizers announced a name change for this workshop. They reported that the workshop will now be known as HotMobile 2006: The 7th IEEE Workshop on Mobile Computing Systems and Applications. USENIX is an in-cooperation sponsor of this workshop.

What follows is an overview of the workshop's proceedings summarizing the formal presentations, but omitting the discussions that followed. The vast majority of this overview focuses on the presentations that are not represented by papers. You will find that the paper summaries contained in this overview are extremely brief and are intended only to help you identify those papers you would like to read in full. Those readers interested in a longer summary should refer to the Digest of Proceedings that appears at the end of the workshop proceedings. This digest includes a summary of the discussions that followed each of the presentations.

This overview is based on the written notes taken by two student volunteers, Tapan Parikh and Alex Varshavsky. They took excellent notes, although they did not always know who was speaking and my notes were not always complete. If anything has been reported in error or omitted, the responsibility lies squarely on my shoulders and not theirs.

OPENING DISCUSSION

The workshop's initial discussion revolved around the following statement: "Resolved: The mobile phone is the only device people will carry in the future." We started by taking a quick straw poll in which only six attendees voted in favor of the resolution. After the straw poll, attendees began discussing the resolution. Each attendee had been randomly assigned to argue the Pro position or the Con position. The discussion period started with small groups of people from each position. After about 20 minutes, we then switched to having all the Pros gather their arguments and all the Cons gather their arguments. Again, after about 20 minutes, we opened the floor to debate. Each side had about 5 minutes to present its case and then open discussion ensued. It should be noted that, at times, certain members of the groups argued in favor of the opposing side.

■ *Pro Position*

Cell phones are already ubiquitously deployed. Gartner believes that in 2005 the number of cell phones sold will have reached 780 million units and that the number will hit 2.6 billion by 2009. Also, in India and China, cell phones are believed to be the primary computing device. Given such a high penetration of mobile phones, application developers will concentrate on developing applications for the phones, especially since computing power and storage are not an issue.

■ *Con Position*

Today, people use a variety of different devices, including cell phones, watches, PDAs, MP3 players, and laptops. Combining the functionality of all these devices into a single cell phone device, resembling a Swiss army knife approach, may result in a device that may do many things, but none of them well. For example, it is unclear what a user interface of such a device would look like. Because the price of single-use devices will go down significantly, it may be more appropriate for users to carry specialized devices that have the right form factor and the right user interface for the task at hand (e.g., an iPod). Also, fashion has a say in what devices people carry with them. For instances, some people wear watches for reasons that have nothing to do with time (e.g., esthetics).

After a lively and interactive discussion, with various attendees taking up their assigned position as well as occasionally arguing for the other side, we took another vote. This time the result included nine votes for the Pro position. Although one attendee jokingly noted that this was not a scientifically valid approach, the discussion was interesting and set the proper tone for the workshop: one of interaction and discussion.

MOBILE PHONES AS APPLIANCES

The theme of the first paper session, chaired by Gaetano Borriello, was considering mobile phones as appliances. John Barton presented the first paper, entitled "Mobile Phones Will Become the Primary Personal Computing Devices." He argued that because of increasing storage and computing power, mobile phones will eventually replace PCs. Users will utilize large displays and input devices available at public places for easier interaction with their mobile phone. After the talk, John took questions from the audience.

John Davis then presented the second talk, on "Supporting Mobile Business Workflow with Commune." The paper proposes a workflow management system for a mobile workforce that utilizes "mini-workflows," network-isolated components that can be offloaded onto mobile clients by leveraging Web services.

LOCALIZATION

Natalia Marmasse chaired our second paper session, on localization. Nishkam Ravi presented the first paper, entitled "Indoor Localization Using Camera Phones." He proposed an indoor localization scheme based on camera phones worn as a pendant by the user. The camera phone automatically takes pictures and transmits them over GPRS to the centralized server, which localizes the user by matching the current picture to the database of preloaded pictures. The discussion following this paper focused on a few issues: training costs, accuracy, and whether the entire system can run on the phone.

Alex Varshavsky then presented a paper entitled "Are GSM Phones *the* Solution for Localization?" He argued that localization using GSM-based mobile phones may be adequate and sufficient for many interesting location-aware applications. The authors show that, with GSM-based fingerprinting, it is possible to achieve 2–5m median error indoors, perform room-level localization indoors and achieve 70–200m median error outdoors. Moreover, by tracking signal stability, it is possible to detect places people visit with very high accuracy.

IS LOCALIZATION A *SOLVED* PROBLEM?

Following our paper session on localization, Gaetano Borriello (University of Washington) moderated a panel session exploring the question of whether localization is a solved problem. Three people sat on the panel: Dieter Fox (University of Washington), Mike Hazas (Lancaster University), and Jeff Hightower (Intel Research Seattle). Gaetano opened the panel by presenting four questions to each of the panelists and giving them each a chance to respond.

Prefacing his first question with "Cell phones are the location-aware platform of choice. We should focus all our attention on improving location systems on phones (accuracy, privacy, performance, etc.). There are no other viable platforms." Gaetano asked, "If it doesn't work on a cell phone, why bother?" Dieter responded that it does not matter because everything can be integrated into the cell phone, if not now, then in the not-too-distant future. Most of what can be done on a laptop will be appropriate for a cell phone. Techniques for providing context awareness should be independent of this sort of detail. Mike responded that people may need to interact with other devices besides phones. Other devices may want to know where they are (e.g., a car or bus). Jeff agreed with the statement and added that everything is converging into phones. He felt that phones are the way to go, except for some enterprise IT and asset-management usages.

"Indoor location systems will piggy-back whatever outdoor system becomes dominant," continued Gaetano, so "can special-purpose indoor infrastructure even be practical for deployment or will location systems have coverage?" Mike responded that indoor location systems are too expensive, especially the ones that require special hardware. So indoor localization systems may need to be different. Jeff stated that specialized infrastructure is reasonable in certain environments, but support for indoor and outdoor localization should be implemented on the same device. We don't want to carry one more device. If additional hardware is required, it should be integrated into mobile phones. Dieter argued that most applications will not require special infrastructure. WiFi signal-strength information will be available in virtually any building. We just need to be smart about how to use it. If necessary, binary sensors or RFID can give additional location context.

For the third question, Gaetano began, "Getting a coordinate is a solved problem. No more papers need to be published on the issue," asking "Shouldn't research now only focus on what to do with the coordinates to solve real problems?" Jeff agreed, stating that coordinate research is basically done. Research should focus on place detection, learning, and labeling; combining activity inference with location; and designing applications with location awareness. Mike argued that we are not yet done, because we still do not know how to deploy the coordinate-based systems, and today's solutions are often expensive or otherwise impractical. Dieter thought that we are mostly done with coordinate-based localization, but the devil is in the details. It is not clear how to combine location with activity recognition. It is also not clear how to learn and maintain personalized maps (predict the location to which the person is going). In addition, combining information from multiple people is interesting.

And now on to Gaetano's final question: "The only people who really care about location privacy are researchers, lawyers, and bloggers. When you get right down to it, regular people just don't care that much, so let's stop worrying about it, OK?" Dieter responded that it all depends on the context. He argued that most people have a problem with being tracked, though he noted that elderly people might accept it. He also felt that there will be continuous erosion of privacy. Gaetano interjected that the indirect use of information is possible and then reported that a professor had complained about his students being tracked because he can be indirectly tracked that way as well. Mike added that although regular people think that they care about privacy, they really do not. Paranoid users have few applications to choose from, because applications are often provided by third parties. The remaining users can be bought out (e.g., by customer "loyalty" cards). He added that Scott McNealy may have had it right when he said, "You have zero privacy anyway. Get over it." Jeff argued that privacy will always be an important design goal. Regular people are pragmatic, privacy is not all or nothing, and it is not solely a technology issue. The goal is to help people avoid socially awkward situations, to support clarity in interpersonal interactions, and to provide transparency and reciprocity. He noted that there were several findings in the Ubicomp Reno paper:

- Use binary choice—disclose what is most useful or don't disclose.
- Levels of denial are needed (say that the system is "busy" as opposed to "deny").
- Blurring is used for clarity, not privacy ("I am in Seattle" may be more meaningful to outsiders than "I am on 45th and 30th").
- Actions convey complex meaning and/or intention.

At this point Gaetano opened the panel to questions and comments from the audience. An active discussion ensued.

KEYNOTE: MOBILE APPLICATIONS SUPPORT FOR ENTERTAINMENT

Frederick Kitson, Vice President and Director of the Applications Research Center at Motorola Labs, gave the keynote address on Friday morning. He focused on the future of mobile applications and showed us a wide variety of the kinds of research his team is working on at Motorola. This overview presents a sample of the visions he shared with us.

More than 70% of i-mode revenue comes from entertainment applications, such as music, sports, personalized TV, imaging, and games. One of the goals of Motorola's research is to drive seamless mobility: to simplify effort, satisfy full mobility, and amplify the user experience. In fact, Motorola requires seamless mobility within its own product lines.

He described a vision of "cache and carry" that transparently "mobilizes" dynamic content. Users consume only a fraction of the content they pay for, in part because the content they capture is not located where they want to consume it. This research focuses on what might happen if they could provide a mobile content experience that moves the content to the user transparently, economically, and just before it is needed. Such a vision requires that the system behave intelligently with respect to battery, storage, and bandwidth consumption as well as with respect to the user's interests and consumption history.

He also described iRadio, in which users have six channels of dynamic content. Each channel has 90 minutes of cached content that can be streamed from the collection device (a PDA) to the user's car radio.

He described the "Push to X" technology, which was originally called iDEN. With this technology, you might have an existing audio connection and, while you are talking, you add visual content to it. He pointed out that standards are changing to increase bandwidth to support these types of services.

Fred then defined a vision of "Ambient Communication." Today, communication is intentional and conscious. It requires that one person call, text, or chat with another person. It requires that other person to interrupt his or her day to receive the communication. He argued that tomorrow communication might be unintentional or subconscious. In this world, one person might send a message to another person without knowing it and the other person might receive that message peripherally, or "ambiently." People will feel more connected in a less obtrusive manner and will have social awareness through context disclosure.

He then presented some interesting and daunting statistics: In the United States, fewer than 10% of WAP phone users actually use the browser; furthermore, among those that do, 50% are lost with each additional click. To address these challenges, Motorola has been working on a system called SCREEN3, which transmits data to idle cell phones in the background, with no noticeable effect on the handset's performance. The data is personalized and scrolls by as the user looks at the screen. If the user pushes any button, the scrolling stops. If the user clicks on something, more content is displayed. The analogy Fred used to represent the amount of data displayed with each additional click is "bite, snack, meal, feast." The "bite" and "snack" are cached on the phone. As the user requests the "snack," the "meal" is prefetched, and so on.

Motorola has considered combining this model for content delivery with location-based services. Services in the user's vicinity could scroll across the phone. The interested user can then easily obtain additional information with a single click. Another important application domain is advertising. Approximately $400 billion each year is spent on advertising. Mobile advertising is a big market and has the potential to be far more effective than billboards, magazine ads, and the like.

He also described integrated content consumption, which would allow users to capture more content "like this" across all of their devices, including both mobile and home devices. The content could be previously stored content as well as upcoming broadcasts that could be recorded. It would aggregate that content with Web images, news articles, songs, and the like.

BREAKOUT DISCUSSIONS

During the last afternoon session on Thursday, we introduced the breakout discussion topics:

- Impact of various networking technologies (gold)
- Application issues (green)
- Device symbiosis (silver)
- Cross-disciplinary research (blue)
- Privacy (red)

Each attendee had been assigned to a team prior to arrival, indicated by a colored star on the name badge. After the discussion questions and team assignments were introduced, each team of approximately eight members broke off to begin discussion. With the weather so pleasant, many teams chose to sit outside on the benches and rocking chairs. On Friday, each team was allotted 10–15 minutes to present a summary of their discussion and allow other attendees a chance to ask questions and to voice their own opinions.

Impact of Various Networking Technologies

The gold team was asked to consider the impact of various networking technologies. The initial questions they were asked to consider included the following:

- What is the impact of community-based networks?

- What might be the impact of having wireless connectivity at highway speeds? Does such functionality create new application scenarios?
- Will we, as a research community, need to support disconnected operation in ten years?

They addressed each of these questions.

With respect to the impact of community-based networks, Nina Bhatti reported that the team had two schools of thought on why these networks are important. The first is that such networks provide special content or special values for local communities. The second had to do with improving the cost structure so that more people have access to the network, thus reducing the digital divide.

Regarding having wireless connectivity at highway speeds, the gold team predicted that it would increase traffic accidents but thought it could provide value in the form of additional information from signs as the car drives past, or by enabling distributed content to be shared among vehicles. They thought it might be more useful to think about traffic-routing scenarios using vehicle-to-vehicle communications (e.g., sharing information on traffic speeds or avoiding congestion on freeways). They also wondered whether the question was addressing the car networking or the people networking across this wireless connectivity. [Editor's note: It should be noted that the gold team's discussion took place before the paper on this topic was presented.]

The gold team also discussed the need for disconnection research. They pointed out that there are two types of disconnected operation: visible and invisible. Invisible disconnection attempts to hide the discontinuous operation from the user (e.g., Outlook uses this approach). Alternatively, visible disconnection makes users aware of discontinuous operation and allows users to act in a way that respects the paradigm (e.g., users do not expect immediate receipt or response of SMS messages). The gold team felt that we still need caching research as we design for disconnection and for vehicular computing. They also identified store and forward as a very powerful idea in this work. Finally, they pointed out that users want to be disconnected at times.

Application Issues

The green team was asked to consider application issues. The initial questions they were asked to consider included the following:

- Why don't we see more applications research?
- What are future directions for mobile applications?
- What is needed from the research community for mobile applications to succeed?
- What characterizes a good application paper?
- What makes good application research?

James Scott reported on the discussion of "Team Green." In response to why we do not see more application research, the team felt that users are tricky. Evaluation of application research suffers from measurability and repeatability challenges. They noted that you need long periods of time to interact with users and that it often goes wrong the first few times. They also observed that good application ideas typically lead to products and patents, but not to open research. Further, applications are generally regarded as engineering rather than research, so you will see "Usage patterns of XX" or "Privacy issues of XX" but not simply "XX" itself.

For future directions of mobile applications, they see health and fitness as well as elder care as important areas. They also see interest in social mobile applications, although they noted that the value proposition of these applications is weak. Finally, they see numerous research issues around thin clients as replacements for PCs.

Next they discussed what is needed from the research community for mobile applications to succeed. The first need was for people, as employees of startup companies. They also noted a need for a shift in expectations and rewards toward more rigorous, deep research rather than least-publishable units. Such a shift should benefit applications research, which already has significant overhead.

They identified the characteristics of good applications research. The biggest one was iterating on the application. They also noted that it was important that researchers resist the temptation to stop after a single iteration.

They made some suggestions of how things could change to better support application research. One suggestion was for review forms to ask for ratings regarding the extent to which a paper describes a piece of work that contributes a building block or builds on top of an existing body of work. Review forms could also assess the extent to which the research provides suitable levels of comparison against other work, using common quantitative measures whenever possible. The second suggested change is to create a journal of impactful research, which contains papers describing only work that was created by one institution or group and also used by *another* institution as an enabler for their work.

- *Device Symbiosis*

The silver team was asked to discuss device symbiosis. The initial questions they were asked to consider included the following:

- What role will mobile phones play in the future?
- Will we ever attain the vision of exploiting devices in the user's environment? Why or why not?

The silver team answered a completely different, though related, set of questions.

John Barton reported that the team began by defining device symbiosis as two or more devices being combined, as peers offering independent value, for a task. Device symbiosis happens opportunistically and spontaneously; it is not configured. It happens wirelessly because mobility causes dynamics.

They continued by describing possible applications of device symbiosis. The applications ranged from home and consumer applications (e.g., phone headsets, games, music, and entertainment), to mobile-travel applications (e.g., mobile radio, electronic wallets, and location-based advertising), to business applications (e.g., face-to-face groupware, HotMobile projectors, and opportunistic augmentation).

They continued with a discussion of the importance of standards to the success of device symbiosis and the need to achieve critical mass. They then made an analogy to the Web. Prior to DNS, networking was for geeks who could make sense of things like 196.192.13.10, and they argued that this is the state of device symbiosis today. DNS gave us human-understandable names for devices. Further, we have search engines that allow people to search all sources known to the search engine. Device symbiosis will require similar functions to support spontaneous connections forced by mobility. Similarly, when they discussed the role of location technology, they noted that searches for symbiotic devices must be constrained by the user's location and that with device symbiosis, users will be able to physically identify spammers.

Finally, the team identified challenges facing device-symbiosis researchers. The first challenge was how to create critical mass. The second concerned standards. The third challenge focused on user experience.

- *Privacy Issues*

The red team was asked to consider privacy issues. The initial questions they were asked to consider included these:

- What mechanisms do we need to support privacy?
- How should we evaluate the privacy of mobile systems/applications? What is the value/price for privacy?
- Have we solved the privacy problem with location-based (and other context-aware) services?

Like the silver team, the red team devised their own approach to the breakout discussion. Mark Corner reported that they first asked themselves what makes this environment different. The answer they came up with is that, although there is some overlap with traditional privacy concerns, mobile computing presents much greater integration with daily activities. In addition, although many attacks are not new, the barriers are lower. Furthermore, they found that the risks are much more subtle. The risks include behavioral information and not just bank records and the like. As in traditional privacy concerns, users often do not understand the risks involved (especially the new risks related to mobile privacy), their exposure often goes unnoticed, they do not understand how to protect themselves, and they cannot make informed decisions.

The team then advanced three proposals.

Symmetric Privacy: In this scheme, there would be full disclosure of all disclosures. In other words, all requests for information are disclosed to the user. This scheme brings to mind the "watch the watcher" model. There would be a mandatory audit trail that records all disclosures of personal information and activity scans that look for exposures the user may have missed.

Aggressive User Interfaces: In this scheme, the system would inform the user about leaked information. It would create an N map for people and/or mobile devices and produce embarrassing reports about their lives. It would rely on social networking. It could even create phishing attacks against a user's own phone.

Help the User: This scheme uses the information collected from the Aggressive User Interface scheme to show how the information was leaked and demonstrate better behavior. It would actively obfuscate to spread disinformation and provide digital anarchy for mobile devices. In fact, mobile devices should impersonate others (e.g., by swapping grocery store loyalty cards).

- *Cross-Disciplinary Research*

The blue team was asked to consider issues concerning cross-disciplinary research. The initial set of questions they were given included the following:

- How do techniques from other fields (e.g., machine learning) apply to mobile computing research? Which ones are most important?

- If you could make one change in your previous mobile computing research projects, what would that change be and why?
- In what field do you see mobile computing making the most inroads?

In response to how techniques from other fields might apply to mobile computing and which ones are most important, the blue team thought control theory provided the basis for adaptive mobile applications (e.g., as bandwidth changes, so does behavior) and that statistical inference techniques provided the basis for fusing location sensor data. They also identified security, networking and operating systems, machine learning, human-computer interaction, industrial engineering, sensor systems, robotics, game theory, and social psychology as providing fertile grounds for cross-disciplinary research.

Regarding the ability to change history, the blue team offered a number of thoughts. The first would be to have anticipated the Web mindshare by engaging earlier and more deeply with the early Web developers and embracing Web practitioners. The second was to pay more attention to issues of data revocation and caching (e.g., erasing an address on Google). The remaining ones included performing more user studies, focusing on existing hardware, contributing more to open source, focusing more on applications, and providing controlled exposure rather than complete transparency.

As an example of how we could have done a better job as a community, Satya asked us to consider the dawn of mobile computing. He pointed out that people were addressing interesting questions but that the Web was ignored by most of us. He thinks that if we had engaged earlier, many deep aspects of this model would have been done in a better way.

The third question the blue team addressed was in what fields they saw mobile computing making the most inroads. The team identified medicine and health (e.g., personal sensors for the elderly, personal fitness, and chronic illness), transportation (e.g., a traffic signal adjusting to the passage of a bus), business processes and workflows (e.g., mobile Web services), gaming and entertainment, logistics and distribution, and privacy models.

FINDING THE RIGHT BALANCE FOR USERS

Our next paper session, chaired by Eyal de Lara, examined how to find the right balance for users. The first paper of this session, presented by Varun Marupadi, was entitled "Presence-Exchanges: Toward Sustainable Presence-Sharing." Varun and his colleagues suggesed introducing a trusted broker into presence-sharing applications, so that misbehaving users could not learn about the presence of others without sharing their own identity.

For our second paper in this session, Anthony Nicholson presented "Exploiting Mobility for Key Establishment." He observed that most Internet traffic today is unencrypted, and he blamed the lack of easy-to-use tools available to users. He and his colleagues propose a model in which keys are established insecurely and are then automatically confirmed by exchanging cryptographic hashes of the keys over many different paths, utilizing inherent user mobility and overlay networks.

SECURE MOBILE COMPUTING

Ramón Cáceres (IBM Research) moderated a panel session exploring the question of whether we might attain secure mobile computing anytime soon. Four people sat on the panel: Carl Ellison (Microsoft), Steve Gribble (University of Washington), Helen Wang (Microsoft Research), and Jason Hong (Carnegie Mellon University).

Ramón opened the panel with a brief discussion of why we should talk about mobile security. He noted that the following articles appeared in the popular press:

- The New York Times had recently reported on a study that found that RFID tags are vulnerable to viruses (15 March 2006).
- PC World found that a virus can pass from PCs to mobile devices (28 February 2006).
- Yahoo! News reported on a virus that can jump from mobile devices to PCs (23 September 2005).
- BBC News reported that the first mobile phone virus had been created (16 June 2004).

Ramón also presented the panelists with a list of questions:

- Can we achieve secure mobile computing anytime soon?
- Is security in mobile computing different from security in general computing?
- Can we build usable security and privacy functions into mobile environments?
- Will trusted computing hardware and virtual machines play a big role in security mobile systems?

He then invited each panelist to make a short presentation.

Carl Ellison compared mobile computing platforms to 1980s PCs. They both support single users; they have a handful of software providers; they have low CPU power as compared to "real" computers; they have a small amount of memory; they hunger for features; and they use tricks to achieve features in spite of their limitations. The result of these limitations is that they have significant security vulnerabilities. These platforms also differ in that mobile computing platforms have been networked from day one. Consequently, they are not physically protected via isolation and face even worse potential security problems. He argued that our industry needs discipline. We need to assume hostile users from day one; we need to partition the platform; we need TPM-style measurement of partitions; we need to ensure that all channels are access-controlled using strong authentication, strong authorization, and thorough ceremony analysis.

Steve Gribble opened his remarks by saying, "Hold on a minute . . . we still haven't figured out secure *nonmobile* computing!" He named spyware, phishing, worms, denial-of-service attacks, and flawed software as examples that support his statement. He identified three wide-open issues that have nothing to do with mobility. These include giving users a conceptual model of security, building attributable networks, and enabling safe sharing in a hostile environment. He argued that mobile devices exacerbate security issues. They tend to be much more promiscuous. They are generally built on weaker, closed systems. They face greater physical threats, such as theft. But he also pointed out some opportunities. For example, mobile devices may allow us to use the physical context of the device together with digital security by requiring the user to touch the device to authorize a communication. He also observed that cell networks at least feel better guarded than the Internet, though he admitted that may simply be an illusion. Finally, he felt that there is still time to design before we get into the same mess we have with the Internet. He ended his presentation with three open questions that he would like to answer. First, why did his IMAP client complain about a bogus certificate from Romania? Second, if he leaves his mobile device in another room for 15 minutes, what's the worst that might have happened? Third, if he receives an SMS message with the subject "Remember to upgrade your Treo OS software," how does he know who sent it, whether he should read it, and whether he should believe it?

Helen Wang continued with a presentation about the threats of smart phones. She showed that smart phones are gaining ground fast: 30 million were shipped in 2004, and it is estimated that 100 million will ship in 2007. They combine the portability of cellular phones with the computational and networking power of PCs. They offer rich functionality and features. She then pointed out some of the many threats that can compromise smart phones: attacks from the Internet (e.g., worms, viruses, and Trojan horses), infections from the desktop via sync (e.g., compromise one and you can compromise both), and peer attacks or infections. She then showed a substantial list of mobile malware from 2004–2005. She then talked about smart-phone zombies and the problems they can cause, which range from SMS spamming, to identity theft, to denial-of-service attacks (e.g., to base stations) and distributed denial-of-service attacks (e.g., to call centers), to remote wiretapping.

Jason Hong opened by summarizing his opinion: "Outlook not so good." He argued that secure mobile computing faces significant challenges that range from mobile devices containing important information to having significant usability, cultural, and economic issues. He showed three news articles showing loss of important data, all from March 2006, and talked about the strong incentives for theft because of it. He shared statistics that approximately 20% of WiFi access points are returned because people couldn't figure out how to make them work and he guesstimated that about 80% of WiFi access points are not secured. He continued with the observation that phishing attacks are stunningly effective. He argued that we need security models that are invisible and *extremely* easy to use. He also discussed some of the cultural issues around cookies, which were originally meant for maintaining state and have become a pervasive means for tracking people online. He also pointed out that the algorithm that the United States seems to use with respect to handling important society issues is to wait for a disaster and then legislate, which is both slow and suboptimal. He then discussed the economic issues. One of the problems we face is that although the estimated cost of phishing in the United States is about $5 billion and although solutions for this problem already exist, the estimated cost of implementing those solutions is greater than $5 billion.

MAKING THE CONNECTION

Our final paper session was chaired by Carla Ellis. This session focused on making connections. The first paper, "Measurements of In-Motion 802.11 Networking," was presented by Richard Gass. This paper studies the ability of a commodity laptop to communicate with 802.11 APs while being driven in a car traveling at speeds between 5 and 75 mph. The findings reveal that a significant amount of data can be pushed through the wireless link, but the performance suffers owing to application-related problems, such as protocols with hand-shaking and long round-trip times.

The second paper was presented by John Barton and examined "Connection Time for Strange Devices." John presented experiences connecting small mobile computers to other computers. The results show that the benefit of connecting phones to larger displays and keyboards may outweigh the burden of making the connection.

CLOSING THOUGHTS

The formal and informal feedback I received after the workshop indicates that people enjoyed the return to the informal, highly interactive workshop format. That success came because of the hard work of numerous individuals. This includes the members of the program committee: Michael Beigl, Nina Bhatti, Gaetano Borriello, Yatin Chawathe, Mark Corner, Carla Ellis, Adrian Friday, Hiromichi Hashizume, Jason Hong, Yih-Chun Hu, Natalia Marmasse, Bhaskar Raman, M. Satyanarayanan, and Doug Terry. They had a very difficult task and did a great job of choosing papers consistent with the new vision for the workshop. Panels are tricky to organize and are generally either really good or, well, not so good. Gaetano Borriello and Ramón Cáceres both did an outstanding job in putting together two very successful panel discussions. Special thanks go to Fred Kitson for sharing his visions of the direction in which mobile applications are heading and to Nina Bhatti for recommending such an outstanding speaker.

Anthony Joseph, Kay Beck-Benton, Eyal de Lara, and Paul Castro, all members of the organizing committee, deserve thanks for their efforts in organizing the workshop and ensuring that the event ran smoothly. A special thanks goes to Kay Beck-Benton, our Local Arrangements Chair, who helped with tasks too numerous to mention and went above and beyond the call of duty. Her help was invaluable and much appreciated.

Finally, I am pleased to report that Nina Bhatti and Eyal de Lara have agreed to be the General Chair and Program Chair, respectively, for HotMobile 2007. They plan to keep the same informal, highly interactive format. I hope to see you there next year!

Announcement and Call for Papers **USENIX**

5th USENIX Conference on File and Storage Technologies (FAST '07)

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/fast07

February 13–16, 2007 **San Jose, CA, USA**

Important Dates

Paper submissions due: *September 6, 2006, 9:00 p.m. EST (this is a firm deadline; sorry, no extensions)*
Notification of acceptance: *November 7, 2006*
Final papers due: *December 19, 2006*
Work-in-Progress Reports/Poster Session proposals due: *January 12, 2007*

Conference Organizers

Program Chairs

Andrea C. Arpaci-Dusseau, *University of Wisconsin, Madison*
Remzi H. Arpaci-Dusseau, *University of Wisconsin, Madison*

Program Committee

Ashraf Aboulnaga, *University of Waterloo*
Mary Baker, *Hewlett-Packard Labs*
Bill Bolosky, *Microsoft*
Scott Brandt, *University of California, Santa Cruz*
Randal Burns, *Johns Hopkins University*
Peter Corbett, *Network Appliance*
Mike Dahlin, *University of Texas, Austin*
Jason Flinn, *University of Michigan, Ann Arbor*
Dharmendra Modha, *IBM Almaden*
Erik Riedel, *Seagate*
M. Satyanarayanan, *Carnegie Mellon University*
Jiri Schindler, *EMC*
Margo Seltzer, *Harvard University*
Kai Shen, *University of Rochester*
Anand Sivasubramaniam, *Pennsylvania State University*
Muthian Sivathanu, *Google*
Mike Swift, *University of Wisconsin, Madison*
Amin Vahdat, *University of California, San Diego*
Carl Waldspurger, *VMWare*
Erez Zadok, *Stony Brook University*

Overview

The 5th USENIX Conference on File and Storage Technologies (FAST '07) 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
- Mobile storage technology
- Parallel I/O
- Performance
- Manageability
- Reliability, availability, disaster tolerance
- Security
- Scalability
- Storage networking
- Virtualization

Deadline and Submission Instructions

Submissions will be done electronically via a Web form, which will be available on the FAST '07 Call for Papers Web site, http://www.usenix.org/fast07/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.

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. Conditionally 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, fast07chairs@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 see the Call for Papers Web site, http://www.usenix.org/fast07/cfp, for details.

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 2006 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.

Preliminary Announcement and Call for Papers USENIX

4th Symposium on Networked Systems Design & Implementation (NSDI '07)

Sponsored by USENIX in cooperation with ACM SIGCOMM and ACM SIGOPS

http://www.usenix.org/nsdi07

April 11–13, 2007 **Cambridge, MA, USA**

Important Dates

Paper titles and abstracts due: *October 2, 2006, 11:59 p.m. GMT*
Complete paper submissions due: *October 9, 2006, 11:59 p.m. GMT*
Notification of acceptance: *December 22, 2006*
Papers due for shepherding: *January 26, 2007*
Final papers due: *February 20, 2007*

Conference Organizers

Program Chairs

Hari Balakrishnan, *Massachusetts Institute of Technology*
Peter Druschel, *Max Planck Institute for Software Systems*

Program Committee

TBA

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 of large-scale networked and distributed computer systems. Systems as diverse as Internet routing, peer-to-peer and overlay networks, sensor networks, Web-based systems, and network 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 computer networks, distributed systems, and operating systems—to foster a broad approach to addressing our common research chal-

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:

- Highly available and reliable networked systems
- Security and robustness of networked systems
- Novel architectures for networked systems (e.g., for specific application domains)
- Overlay networks and peer-to-peer systems
- Mobile, wireless, and sensor network systems
- Network measurements, workload, and topology characterization
- Autonomous and self-configuring networked systems
- Managing, debugging, and diagnosing problems in networked systems
- Resource management and virtualization for networked systems
- Distributed storage, caching, and query processing
- Practical protocols and algorithms for networked systems
- Application experiences based on networked systems
- Novel operating system support for 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 now single- rather than double-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 11:59 p.m. GMT on October 2, 2006, and the corresponding full paper is due by 11:59 p.m. GMT on October 9, 2006. 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 26, 2007. 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 20, 2007, for the conference Proceedings.

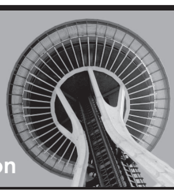
Simultaneous submission of the same work to multiple venues, submission of previously published work, and plagiarism constitute dishonesty or fraud. USENIX, like other scientific and technical conferences and journals, prohibits these practices and may, on the recommendation of a program chair, take 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.

Authors uncertain whether their submission meets USENIX's guidelines should contact the Program Chairs, nsdi07chairs@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.

OSDI | 06
7th Symposium
on Operating Systems
Design and Implementation



Nov. 6–8, 2006,
Seattle, WA

Sponsored by USENIX,
The Advanced Computing
Systems Association,
in Cooperation with
ACM SIGOPS

Save the Date!

7th Symposium on Operating Systems Design and Implementation

November 6–8, 2006, Seattle, WA, USA

http://www.usenix.org/osdi06

Join us in Seattle, WA, November 6–8, 2006, for innovative, exciting work in the systems area. The 7th Symposium on Operating Systems Design and Implementation (OSDI '06) brings together professionals from academic and industrial backgrounds in what has become a premier forum for discussing the design, implementation, and implications of systems software. OSDI emphasizes both innovative research and quantified or illuminating experience.

OSDI '06 is co-located with the 3rd Workshop on Real, Large Distributed Systems (WORLDS '06), which will take place on November 5. Paper submissions are due July 7, 2006. See http://www.usenix.org/worlds06 for details.

The Second Workshop on Hot Topics in System Dependability (HotDep '06) will be held on November 8, immediately following OSDI '06. Paper submissions are due July 15, 2006. See http://www.usenix.org/hotdep06 for details.

HPC wire
GRID today

HPCwire & GRIDtoday

HPCwire is the Leading Source for Global News, Information, and Events on High Performance Computing

GRIDtoday is the Leading Source for Global News, Information, and Events on Grid and Service-Oriented IT

HPCwire and GRIDtoday are the leading publications in their respective fields. Both online publications, are produced by Internet publishing pioneer, Tabor Communications Inc., and provide timely news, authoritative commentary and feature interviews with leaders in the fields of high-end, data-intensive computing.

Free One-Year Subscription

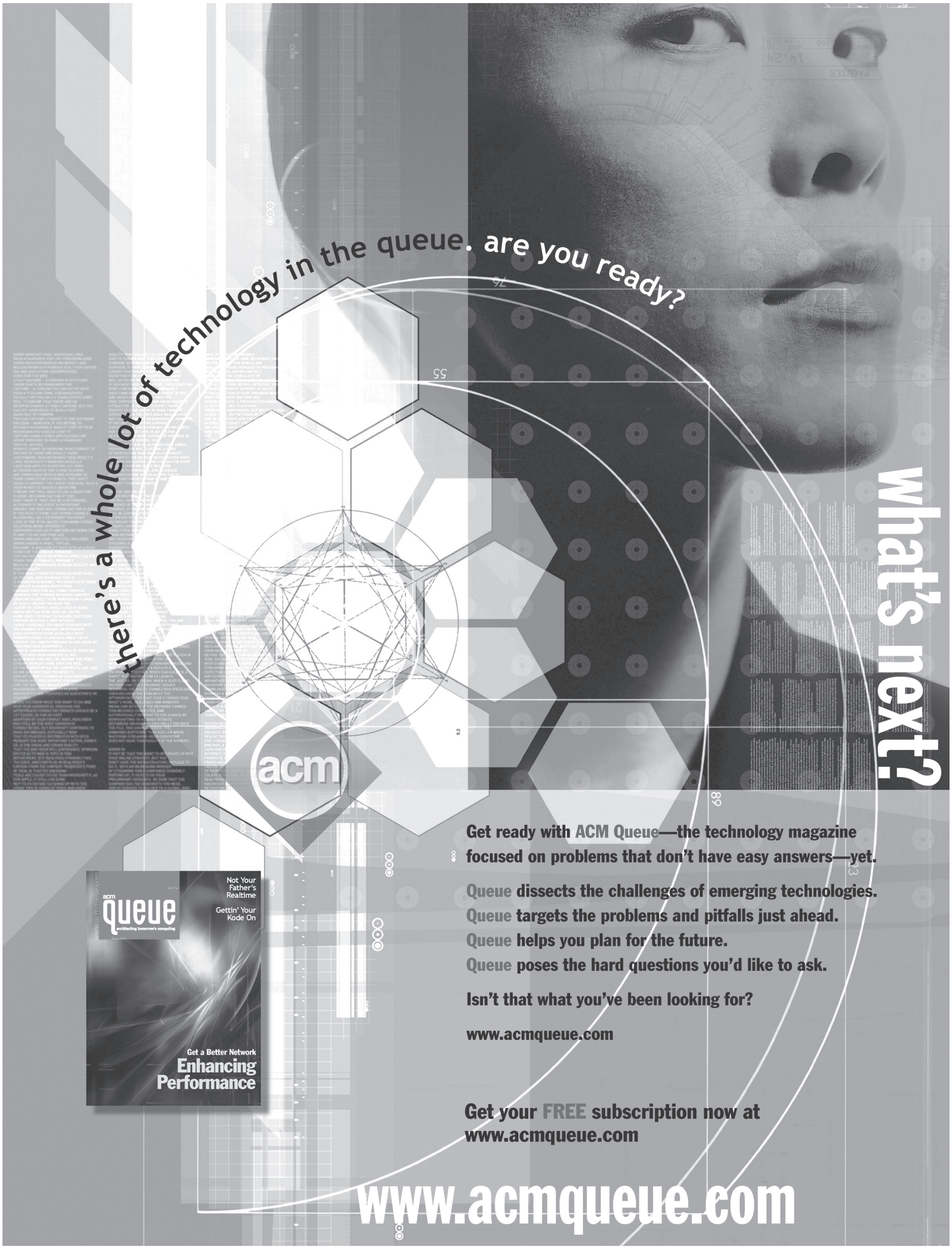
For your complimentary subscription go to
www.hpcwire.com
www.gridtoday.com

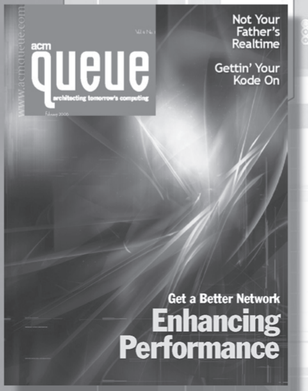
You Won't Want to Miss a Single Issue!

What Our Readers Say About Us

"I have read HPCwire almost every Thursday night since its inception. I find it an extremely useful method of maintaining an awareness of the activities in the HPCC community."

"I now totally depend on it for providing me with late breaking information... In fact, I am not quite sure how I did my job as Director of a national supercomputer center without it."


there's a whole lot of technology in the queue. are you ready?
what's next?
acm



Get ready with ACM Queue—the technology magazine focused on problems that don't have easy answers—yet.

Queue dissects the challenges of emerging technologies.
Queue targets the problems and pitfalls just ahead.
Queue helps you plan for the future.
Queue poses the hard questions you'd like to ask.

Isn't that what you've been looking for?

www.acmqueue.com

Get your FREE subscription now at www.acmqueue.com

www.acmqueue.com

LISA '06

20TH LARGE INSTALLATION
SYSTEM ADMINISTRATION CONFERENCE

A **Blueprint** for Real World
System Administration

DECEMBER 3–8, 2006 | WASHINGTON, D.C.

6 days of training by experts in their fields

3-day technical program:

- Keynote Address by Cory Doctorow, science fiction writer, co-editor of "Boing Boing," and Senior Fellow, USC Annenberg Center for Communication
- Invited Talks by industry leaders
- Refereed Papers, Guru Is In Sessions, and WiPs

Vendor Exhibition

And more!

Online registration opens in late summer at www.usenix.org/lisa06

Sponsored by
USENIX [sage]

Register by November 10 and save! **www.usenix.org/lisa06**

;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
RIDE ALONG ENCLOSED