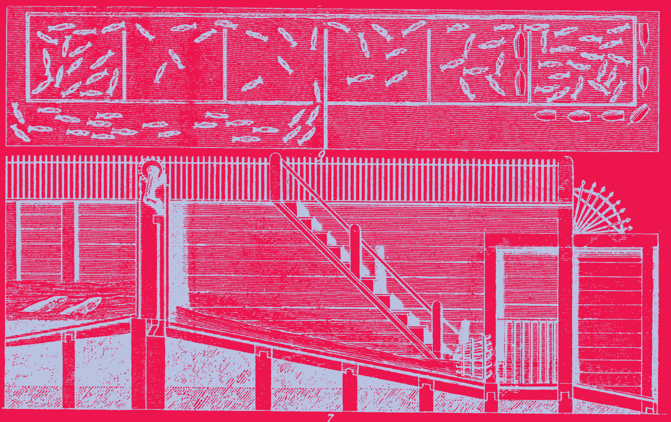
OCTOBER 2005 VOLUME 30 NUMBER 5

;login:

THE USENIX MAGAZINE



USENIX
The Advanced Computing Systems Association

USENIX Upcoming Events

ACM/IFIP/USENIX 6th International Middleware Conference

NOVEMBER 28–DECEMBER 2, 2005, GRENOBLE, FRANCE
http://middleware05.objectweb.org

19th Large Installation System Administration Conference (LISA '05)

Sponsored by USENIX and SAGE

DECEMBER 4–9, 2005, SAN DIEGO, CA, USA
http://www.usenix.org/lisa05

2nd Workshop on Real, Large Distributed Systems (WORLDS '05)

DECEMBER 13, 2005, SAN FRANCISCO, CA, USA
http://www.usenix.org/worlds05

3rd International IEEE Security in Storage Workshop

Sponsored by IEEE Computer Society Task Force on Information Assurance (TFIA) in cooperation with IEEE Mass Storage Systems Technical Committee (MSSTC) and USENIX

DECEMBER 13, 2005, SAN FRANCISCO, CA, USA
http://www.ieeeia.org/sisw/2005

4th USENIX Conference on File and Storage Technologies (FAST '05)

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

DECEMBER 13–16, 2005, SAN FRANCISCO, CA, USA
http://www.usenix.org/fast05

7th IEEE Workshop on Mobile Computing Systems and Applications (WMCSA 2006)

Sponsored by IEEE Computer Society in cooperation with USENIX

APRIL 6–7, 2006, SEMIAHMOO RESORT, WA, USA
http://research.ihost.com/wmcsa2006/
Paper submissions due: October 1, 2005

3rd Symposium on Networked Systems Design and Implementation (NSDI '06)

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

MAY 8–10, 2006, SAN JOSE, CA, USA
http://www.usenix.org/nsdi06
Paper titles and abstracts due: October 10, 2005
Final paper submissions due: October 17, 2005

5th System Administration and Network Engineering Conference (SANE 2006)

Organized by Stichting SANE and co-sponsored by Stichting NLnet, USENIX, and SURFnet

MAY 15–19, 2006, DELFT, THE NETHERLANDS
http://www.sane.nl/sane2006
Paper submissions due: October 24, 2005

2006 USENIX Annual Technical Conference (USENIX '06)

MAY 30–JUNE 3, BOSTON, MA, USA
http://www.usenix.org/usenix06
Paper submissions due: January 17, 2006

15th USENIX Security Symposium (Security '06)

JULY 31–AUGUST 4, VANCOUVER, B.C., CANADA
http://www.usenix.org/sec06
Paper submissions due: February 1, 2006

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

NOVEMBER 6–8, SEATTLE, WA, USA
http://www.usenix.org/osdi06
Paper submissions due: April 24, 2006

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

contents

;login:

VOL. 30, NO. 5, OCTOBER 2005

EDITOR
Rik Farrow
rik@usenix.org

MANAGING EDITOR
Jane-Ellen Long
jel@usenix.org

COPY EDITOR
Steve Gilmartin
proofshop@usenix.org

PRODUCTION
Rob Carroll
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.

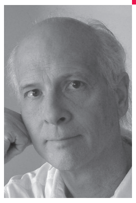
$85 of each member's annual dues is for an annual subscription to *;login:*. Subscriptions for nonmembers are $115 per year.

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 Farrow provides UNIX and Internet security consulting and training. He is the author of *UNIX System Security* and *System Administrator's Guide to System V* and editor of the SAGE Short Topics in System Administration series.

rik@usenix.org

FOR THE PAST COUPLE OF MONTHS, I have become absorbed in operating systems. My quest began with some consulting regarding security features of the current Linux kernel, including SELinux, then plunged even deeper during the HotOS workshop. I do not pretend to be a kernel hacker, but I am very interested in what goes on with the design and implementation of operating system software.

Like some others, I had wondered what had happened to FreeBSD 4's stellar performance when FreeBSD 5 appeared. Instead of getting faster, FreeBSD was slower. If I had bothered digging deeper, I would have learned that this was the result of far-reaching changes in the FreeBSD kernel. Michael W. Lucas explains these changes in his article, which in turn is based on a talk given by Robert Watson about modifying the kernel to support SMP (symmetric multiprocessing). And perhaps by the time you read this column, FreeBSD 6 will have appeared, ready to utilize the new multiprocessor cores that are popping up.

Lucas explains just why the transition from single-threaded to multi-threaded kernel takes so long and is so hard to do right. I first understood the importance of the Big Giant Lock when I was reviewing an early multiprocessing server that used SPARC processors. I ran a simple benchmark that spawned additional processes, each of which ran an integer-intensive program. I tried my benchmark with one processor, then two, three, and, finally, four processors enabled, and the results astounded me (at the time). Adding processors does not in itself linearly improve performance. Enabling the fourth processor barely added a 15% improvement to the results. The Big Giant Lock ensures that only one process (or interrupt handler) can run in kernel space at a time, which devastates performance.

The HotOS workshop (see the summaries in this issue) brought different surprises. I enjoyed the workshop immensely, as much for the free time spent with attendees as for the talks. The lunchtime discussions have sparked one article already, in which Marc Fiuczynski makes a fervent plea for better methods for patching Linux kernels.

You will also find a discussion of the Linux Kernel Developers Summit by Jonathan Corbet. Corbet, who has been summarizing the summits for years, has provided a short overview for *;login:* readers. And Peter Galvin explains Solaris 10 containers. Note that Sun bit the SMP revision bullet years ago, making it a leader in SMP OSes today. The concept of containers adds a powerful twist to Solaris, a useful VM architecture unlike others you may know about.

On the security front, David Malone discusses security features of FreeBSD. As I read this, I found myself wishing that some of these appeared in Linux as well. But BSD-envy is nothing new. Srikanth Kandula explains Kill-Bots, based on a paper he presented at NSDI (see the summaries in the August issue of *;login:*).

You might note that instead of my usual musings, I have acted much more like an editor this time around. I apologize, but this issue is so packed with summaries and articles, I really didn't have the space for me. You can expect that my column will appear much as it has in the past in the December issue of *;login:*, with its focus on security.

Again, I do appreciate your feedback about the changes appearing in *;login:*. You can send me email at login@usenix.org, along with article proposals, letters to the editor, complaints, and praise. If you have books you want to review (or think should be reviewed), try the new bookreviews@usenix.org alias. You can find a more structured approach to making suggestions about *;login:* at https://db.usenix.org/cgi-bin/loginpolls/oct05login/survey.cgi.

To be honest, I am quite happy to be able to approach some of the world's brightest minds and ask them to write articles about those issues I consider most crucial for the advancement of computing systems. But you do need to make sure I don't stray too far.

USENIX FAST 20 05 4th USENIX CONFERENCE ON FILE AND STORAGE TECHNOLOGIES

DEC. 13-16 / 2005 / HOLIDAY INN GOLDEN GATEWAY / SAN FRANCISCO / CA /USA

SAVE THE DATE!

FAST '05

4th USENIX Conference on File and Storage Technologies

December 13–16, 2005, San Francisco, CA

Join us in San Francisco, CA, December 13–16, 2005, for the latest in file and storage technologies. The 4th USENIX Conference on File and Storage Technologies (FAST '05) brings together storage system researchers and practitioners to explore new directions in the design, implementation, evaluation, and deployment of storage systems. Meet with premier storage system researchers and practitioners for a day of advanced tutorials and 2.5 days of ground-breaking file and storage information!

MICHAEL W. LUCAS

FreeBSD 5 SMPng

THE NETWORK STACK

Michael W. Lucas is a network consultant and author of *Absolute BSD, Absolute OpenBSD, Cisco Routers for the Desperate*, and the forthcoming *PGP & GPG*. He has been logging onto UNIX-like systems for twenty years and finds lesser operating systems actively uncomfortable.

mwlucas@blackhelicopters.org

The author wishes to gratefully acknowledge Robert Watson's strong contribution to this article.

FREEBSD IS ONE OF THE GRANDPARENTS of open source operating systems, and FreeBSD version 4 is considered the gold standard of high performance by its user community. In this article we'll discuss the improvements in FreeBSD 5, using the network stack as an example of the particularly heinous problems faced when enhancing multiprocessor operating systems.

FreeBSD 5 had many disruptive new features (such as the GEOM disk layer and ACPI) but also had extremely high ambitions for its new SMP implementation. This SMP infrastructure was necessary for the future growth of FreeBSD, but required massive rewrites in many parts of the system. The new multi-CPU project, dubbed "SMPng," steered FreeBSD in a direction that promised incredible performance enhancements—at the price of a lot of work.

To understand why this was considered worthwhile, we need to consider some basics of multiprocessor computing. What we usually think of as "multiprocessing" is actually "symmetric multiprocessing," or SMP. In SMP you use multiple general purpose processors, all running the same OS. They share memory, I/O, PCI busses, and so on, but each CPU has a private register context, CPU cache, APIC timer, and so on. This is certainly not the only approach: all modern video cards have a Graphics Processing Unit (GPU), which could be considered a special purpose asymmetric multiprocessor. Managing multiple identical general purpose CPUs has become dramatically more important with the advent of multi-core CPUs.

A multiprocessor-capable OS is one that operates correctly on multiprocessor systems. Most operating systems are designed to give the user access to those extra CPUs simply as "more computing power." While many people have implemented alternatives to this, the general idea that more CPU means more horsepower is still what most of us believe.

Additionally, adding processors can't be allowed to change the look-and-feel of our operating system. Managing these processors becomes much simpler if you abandon the current process model, standard APIs, and so on. Many of these APIs and services were designed for systems with only a single CPU and assumed that the hardware had only one processor executing one task at a time. As with so many other hardware evolutions, it would be easier to knock down the house of UNIX and build a strip mall. Instead, the FreeBSD Project has had contractors climbing over the old house to bring it up to today's building code—and gain some extras while we're at it.

Obviously, the goal of adding processors is improving performance. The problem is that "performance" is a very vague term: it depends on the work you're doing, and trade-offs happen everywhere. There's a 100-mpg carburetor that works wonderfully, if you don't mind doing 0 to 60 in about an hour.

The best way to handle performance tuning in SMP is to measure how your system performs under the workload you're interested in, and continue measuring as you add additional CPUs. The SMP implementation needs to not slow down the application in itself, and then it needs to provide features to make it possible to accelerate the application. In an ideal world, your application performance would increase linearly with additional processors—an eight-CPU system would perform eight times as well as a one-CPU system. This simply isn't realistic. The OS and application will be slowed by having to share resources such as memory and bus access, and keeping track of where everything is becomes increasingly complex as the number of CPUs increases. Consistent measuring and benchmarking are vital when embarking on an SMP implementation.

Implementing SMP

Implementing SMP is simple. First, make it run. Then make it run fast. Everything else is just petty details.

The obvious place to begin is in the kernel. Nothing happens until your kernel notices the additional processors. Your OS must be able to power up the processors, send them instructions, and attend to everything that makes it possible to use the hardware.

Then your applications must be able to use the additional CPU. You might find that your important application can only run on one processor at a time, rendering that additional processor almost useless. You can have an application monopolize one CPU while the other CPU handles all the other petty details of keeping the system up, but this is less than ideal.

Your OS libraries and utilities play a vital part in this. Perhaps the most common way to make an application capable of using parallelism is by using threads. Your OS must include a multiprocessor-capable and well-optimized threads library. It can do its thread support in either the kernel or userland. Some applications use more brute-force methods of handling parallelism—the popular Apache daemon forks copies of itself, and those children can automatically run on separate processors.

Once your application can use additional processors, start measuring performance. By observing the system as it handles your work you can identify and address the bottlenecks in your real-world load. Once you fix those bottlenecks, benchmark again to find the new bottleneck. Shuffle your trade-offs until you reach an acceptable average for all of your workloads.

Traditional OS kernels expect that there is only one CPU, and so when the kernel returns to a task, all of the appropriate resources should be right where they were left for that task. When the machine has multiple processors, however, it's entirely possible for multiple kernel threads to access the same data structures simultaneously. Those structures can become corrupted. This makes the kernel angry, and it will take out its feelings on you one way or another.

You must implement a method of maintaining internal consistency and data synchronization, and provide higher level primitives to the rest of the system. Some applications require that certain actions be handled as a whole (known as "atomic operations"). Others simply insist that certain things are done before other things. Your synchronization model must take all of this into account, without breaking the API so badly that you scare off your users. Your choice of locking model affects your system's performance and complexity, and so is very important. For example, let's contrast the locking model used with FreeBSD 3.x/4.x to that used in 5.x and later.

The Big Giant Lock (BGL) model used in FreeBSD 3.x/4.x is the most straightforward way to implement SMP. The kernel is only allowed to execute on one CPU at a time. If a process running on the other CPU needs to access the kernel, it is held off (spinlocks) until the kernel is released by the other process.

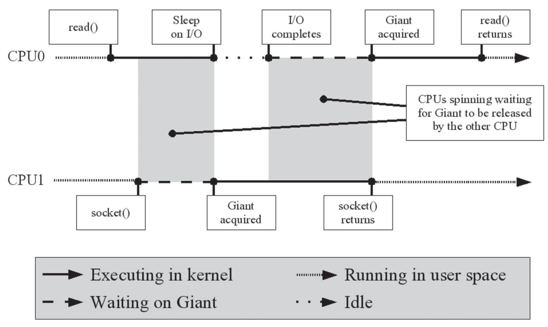
Contention occurs in the BGL model when tasks on multiple CPUs compete to enter the kernel. Think about how many types of workload access the kernel: user threads that do system calls, interrupts or timer driver activity, reading or writing to disk or networks, IPC, scheduler and context switches, and just about everything else. On a two-CPU system this isn't too bad—at least you're doing better than you would with one CPU. It's horrible on a four-CPU system, and unthinkable on an eight-way or bigger.

The locks are obviously necessary for synchronization, but the costs are high. Overall, a dual-CPU system is an improvement over a single processor.

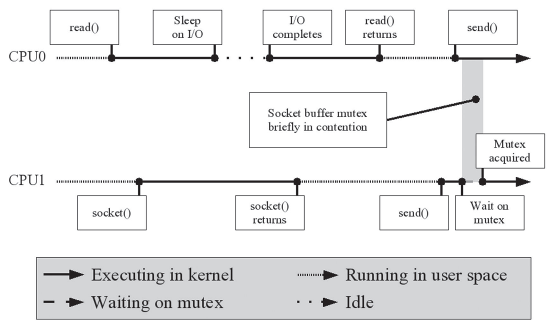
Fine-Grained Locking

FreeBSD 4.x's Big Giant Lock was the main performance bottleneck, and just had to go. That's where fine-grained locking came in. Fine-grained locking is simply smaller kernel locks that contend less. For example, a process that has entered the kernel to write to a file shouldn't block another process from entering the kernel to transmit a packet. The FreeBSD developers implemented this iteratively. First they locked the scheduler and close dependencies such as memo-

Context Switching in a Giant-Locked Kernel



Context Switching in a Finely Locked Kernel



ry allocation and timer events. High-level subsystems followed, such as a generic network stack lock or a file-system lock. They then proceeded to data-based locking. Once they hit this point, it was a simple matter of watching to identify the new bottlenecks and lock them more finely.

Goals along the path include adopting a more threaded architecture and implementing threads where the kernel can work in parallel. Interrupts in particular were permitted to execute as threads. FreeBSD also had to introduce a whole range of synchronization primitives such as mutexes, sx locks, rw locks, and semaphores. Low-level primitives are mapped into higher level programming services. Atomic operations and IPIs are at the bottom, which are used to build mutexes, semaphores, signals, and locks, which, in turn, are assembled into lockless queues and other structures at the very top.

As this was a gradual migration rather than an all-at-once conversion, subsystems that were not yet properly locked during the conversion were allowed to "seize" the Giant Lock. A device driver that had not yet been converted was allowed to scream "OK, everybody out of the kernel, I must process an interrupt!" This was called "holding" the Giant Lock, and it reduced performance to the 3.x/4.x level.

One by one, each system was finely locked, the BGL slid off, and newly exposed problems resolved. This produced a very different contention pattern.

This was complicated, of course, by the fact that FreeBSD is a project in use by millions of people around the world. People even consider selected points along the development version stable enough for production use. If the FreeBSD team could have simply declared, "The development branch of FreeBSD will be utterly unusable for six months," fine-grained locking could have been accomplished more quickly. They would have also alienated many users and commercial sponsors. While commercial companies can get away with this, a project like FreeBSD simply can't. The team did the equivalent of changing a car's oil while said vehicle was barreling down the freeway at 80 miles an hour.

Today, FreeBSD 5.x has fine-grained locking in most major subsystems, except for VFS. The network stack as a whole runs without Giant, although a few network protocols still require it. Some high-end network drivers execute without seizing Giant. FreeBSD 6.0 (which should be out by the time this article reaches print) is almost completely Giant-free. VFS itself runs without Giant, although some file systems do not. (Those of you running high-performance databases on a FAT file store, with a server using those dollar-a-dozen Ethernet cards, might not be pleased with its performance.) A few straggling device drivers require the BGL, but those are slated for conversion or execution before FreeBSD 7.0 is released (probably in 2007). The network stack also runs without the BGL. As locking the network stack was one of the more interesting parts of implementing fine-grained locking, let's take a closer look at it.

Locking the Network

The FreeBSD network stack includes components such as the mbuf memory allocator, network device drivers and interface abstractions, a protocol-independent routing and event model, sockets and socket buffers, and a slew of link-layer protocols and network-layer protocols such as IPv4, IPv6, IPSec, IPX, EtherTalk, ATM, and the popular Netgraph extension framework. Excluding distributed file systems and device drivers, that's about 400,000 lines of code. To complicate things further, FreeBSD's TCP/IP stack has been considered one of the best performers for many years. It's important not to squander that reputation!

Locking the network stack has very real problems. Overhead is vital: a small per-packet cost becomes very large when aggregated over millions of packets per second. TCP is very sensitive to misordering, and interprets reordered packets as requiring fast retransmit. Much like our 100-mpg carburetor, different optimizations conflict (e.g., optimizing for latency can damage throughput).

FreeBSD uses a few general strategies for locking the network stack. Data structures are locked. Also, locks are no finer than that required by the UNIX API—e.g., parallel send and receive on the same socket is useful, but not parallel send on the same socket. References to in-flight packets are locked, not the packets themselves. Layers have their own locks, as objects at different layers have different requirements.

Locking order is vital. Seizing locks incorrectly can cause deadlocks. Driver locks are leaf locks. The network protocol drives most inter-layer activity, so protocol locks are acquired before either driver locks or socket locks. FreeBSD 5 avoids lock problems via deferred dispatch.

Transmission is generally serial, so the work is assigned to a single thread. Reception can be more parallel, so work can be split over multiple threads.

Increasing Parallelism

All of the above is just "making it run." Afterwards came time to "make it run fast." Once the network stack is freed of the Big Giant Lock, pick an interesting workload and see where contention remains. Where was CPU-intensive activity serialized in a single thread, causing unbalanced CPU usage? Identifying natural boundaries in processing, such as protocol hand-offs, layer hand-offs, etc., both restricted and inspired further optimizations. Every trade-off had to be carefully considered and then tested to confirm those ideas. Context switches and locks are expensive, so they had to be made as useful as possible.

All this had its own challenges. The FreeBSD-current mailing list (for those people using the development version) saw many reports of deadlocks, poor performance under edge situations, and any sort of weird issue imaginable. While FreeBSD's sponsors were very generous with donations of test facilities, no test can possibly compare with the absurd range of conditions found in the real world.

One not uncommon problem during development was deadlock. If threads one and two both require locks A and B, and thread one holds A while thread two holds B, the whole system grinds to a halt. This deadly embrace was avoided by a hard lock order on most mutexes and sx locks, disallowing lock cycles, and the WITNESS lock verification tool. There's also a variable, hierarchal lock order. Lock order is a property of data structures, and at any given moment the lock order is defined for that data structure. The lock order can change as the data structure changes. And a master lock serializes simultaneous access to multiple leaf locks. Ordering was vital to avoiding deadlock—but weakening ordering can improve performance in certain cases.

Awareness of locking order and violations is critical throughout this. The WITNESS run-time lock-order monitor tracks lock-order acquisitions, builds a graph reflecting the current lock order, and detects lock-order cycles. It also confirms that you're not recursively locking a non-recursive lock as well as detecting other basic problems. WITNESS uses up a lot of CPU time but is invaluable in debugging.

Every lock is another slice of overhead. FreeBSD 5.x amortizes the cost of locking by avoiding multiple lock operations when possible, and it amortizes the cost of locking over multiple packets. When possible, locks are coalesced or reduced. Combining locks across layers can avoid additional locks. If you lock finely enough you can cause "live lock," where your system is so busy locking and unlocking from interrupts that it does no actual work.

Some workloads handled parallelization better than others. Parts of the network stack, such as TCP, absolutely require serialization to avoid protocol performance problems. Any sort of naive threading violates ordering. FreeBSD uses two sorts of serialization: thread serialization and CPU serialization, which uses per-CPU data structures and pinning/critical sections.

Today's SMPng Network Stack

FreeBSD 5.x and above largely run the network stack without the Big Giant Lock, and 6.x shows substantial improvements over both 4.x and 5.x. The project has progressed from raw functionality to performance tuning. The development team is paying close attention to the performance of popular applications such as MySQL, as well as basic matters such as raw throughput.

Many workloads, such as databases and multi-threaded/multi-process TCP use, show significant improvements. The cost of locking hampers per-packet performance on very specific workloads, such as the packets per second when forwarding and bridging packets. And, compared to the gold standard of 4.x, performance on single-processor systems is sometimes suboptimal. While single-processor performance is being carefully monitored and enhanced, as dual-core systems become the standard even on workstation systems this will be less important. The FreeBSD team is actively working on performance measurement and optimization.

Juggling all these optimizations is hard; it took about five years to get past merely functional to optimal. The oil change at 80 mph is just about done, though, and it's time to floor the accelerator and see what this baby can do.

MARC E. FIUCZYNSKI

better tools for kernel evolution, please!

Dr. Marc E. Fiuczynski is a research scientist in the Computer Science department at Princeton University and a member of the PlanetLab R&D team.

mef@cs.princeton.edu

THIS ARTICLE SUMMARIZES A FORAY into the land of Linux, revealing the soft underbelly of the animal called kernel. This voracious animal is eagerly eating up others, but how much longer can it do so before its belly bursts?

Analogies aside, to many users Linux is great—it is cheap (free in many cases) and often works quite well for the intended purposes. Generally, such users treat Linux as a black box by using an unmodified distribution such as Fedora, SuSE, etc. For users for whom a conventional distribution is insufficient, Linux, as one of the quintessential open source projects, offers unlimited flexibility for customization. Moreover, various kernel extensions released as patch sets, or simply patches, offer unique and useful features not available in the mainline version of the kernel.

Unfortunately, maintaining a customized kernel can be challenging, particularly when it is necessary to keep track of security updates, bug fixes, and general enhancements to the mainline kernel. The reason is that externally developed kernel extensions are often available as patches that typically apply only to the vanilla mainline kernel. While these patches sometimes apply to the latest mainline kernel, often they do not and so require integration work.

While such integration (merging) may be trivial at times, it can quickly get out of hand. Why? Even when these patches apply cleanly to the latest release from kernel.org, they often do not to the latest releases from distributions such as Fedora or SuSE. These distributions, to set themselves apart from others, introduce their own value-added modifications to the kernel and, in some cases, tend to be ahead of the stable 2.6 release by integrating features from the unstable kernel.org releases. Consequently, for those using a customized kernel in a production setting, the job of keeping on top with the latest and greatest kernel can become a tedious one that is pure overhead.

This has been my experience while maintaining the Linux kernel used for PlanetLab (http://www.planet-lab.org). PlanetLab is a geographically distributed overlay platform designed to support the deployment and evaluation of planetary-scale network services. As of August 2005, it consists of over 580 machines at 275 sites in 30 countries, and it has supported over 450 research projects. PlanetLab continues to grow at a rate of approximately five sites and 10 machines per month. Each machine runs a customized version of Linux to support a "virtual private server" (VPS) model, which is used to isolate separate research projects running on a single machine from each other. At one point the kernel for PlanetLab was modified by 28 patches—both externally developed and homegrown. The kernel was so tedious to maintain that at one point it lagged eight minor releases behind the 2.4 mainline kernel release. Upon switching to the 2.6 kernel release, we focused on reducing the patch count to a minimum with the goal of keeping close to the latest mainline kernel release. Nonetheless, our kernel still uses several large patches to support performance isolation and namespace isolation for said VPS support.

At this point, you, likely a Linux user, may be thinking, "Hey man, quit the whining. This problem only affects a small group that have a vested interest in maintaining a highly customized kernel." I humbly disagree. With the rampant success of Linux, this "small" group is growing by leaps and bounds. Besides well-known players like RedHat, SuSE, and IBM, there is a growing number of corporations (e.g., PalmSource, Wind River Systems, Panasonic, NEC, NTT DoCoMo, to name just a few) and government agencies worldwide that are putting a tremendous amount of effort into the Linux kernel. The happy days of the Linux phenomenon probably are numbered. Why? Rather than working toward the general good, corporate kernel programmers will try as hard as possible to push their "modifications" into the Linux kernel in pursuit of their own agendas. It is just a matter of time before the soft underbelly bursts.

When this happens, significant infighting will ensue, leading to fragmentation or who knows what—nothing that's good for any community. What can be done about this? Will Linus Torvalds keep everything under control so that the rest of us can continue to obliviously toil along with Linux? No! To address this problem it is not prudent to rely on a bazaar, cabala, cathedral, benevolent pope, or any sociopolitical model. No one truly understands or can predict the long-term outcome of such models. As engineers we have two choices: (1) blissfully ignore the problem and hope the day of reckoning will never arrive, or (2) turn it into a technological problem that we can attempt to solve—i.e., find better ways to evolve the kernel proper with kernel extensions.

My position paper titled "patch(1) Considered Harmful," presented at this year's Workshop on Hot Topics in Operating Systems , outlines why patch is harmful for the evolution of the kernel (see the HotOS summaries in this issue of *;login:*). In a nutshell, patch is good for localized fixes but bad for kernel extensions that introduce changes which crosscut many files and/or functions. Analysis of patches revealed that kernel extensions can easily cover a hundred existing kernel files, even though it represents a logical unit, expressing a single crosscutting concern. The crux of the problem is that with today's tools (patch, cvs, bk, etc.) these changes need to be integrated into the kernel proper, hence driving kernel programmers to achieve the ultimate integration of their modification into the mainline kernel managed by Torvalds. To avoid this problem altogether, I am working with a team spread across New York University, University of Victoria, and Princeton University on a toolkit called C4, for CrossCutting C Code.

As part of our work on C4, our analysis of various patch sets reveals that kernel extensions primarily make intramodule changes: a coherent collection of modifications encapsulated within an existing module. These changes may modify many of the functions within a particular module, but they do not change the externally visible interface. Clients of the module do not need to change their code and usage patterns. Please note the loose use of the word "module" to mean any collection of components with a well-defined external interface, including kernel subsystems. In particular, a module is not limited to a single kernel source file.

Some kernel extensions also make intermodule changes: modifications that change intermodule interfaces, or the visible semantics of an existing interface, in fundamental ways. For instance, modifications of a function's type or the field makeup of a data structure (e.g., adding, deleting, or changing the type of a field) are intermodule changes. Making such changes can have far-reaching consequences: it can require updating all modules that directly use the changed code. This is prohibitive when the interface changes are in the kernel proper or in the generic device driver framework and trigger corresponding changes in specific device drivers—there might be hundreds. Doing such updates manually is error-prone and time-consuming.

Recognizing that intramodule and intermodule changes are common to new kernel extensions for Linux, our approach with C4 is to make them part of the kernel's architecture by leveraging aspect-oriented programming (AOP) techniques. Whereas object-oriented programming provides linguistic mechanisms for structuring self-contained units of code, AOP provides linguistic mechanisms for structuring concerns that naturally cut across primary modules of a system. More specifically, our approach is to express intramodule changes as semantic patches using aspects, which provide a language-supported methodology for integrating crosscutting concerns with a program.

The benefits of aspects are twofold. First, they provide a well-defined specification of domain-specific features that is separate from baseline functionality, yet can be automatically integrated with the kernel. Second, we believe that aspects will enable tools to perform automatic analysis of the implications of composing several crosscutting concerns and therefore help identify true semantic conflicts, as opposed to the line-by-line conflicts identified by patch.

Thus, a toolkit like C4 provides an opportunity to the growing number of well-funded (and under-funded) kernel developers to have their code included with the latest mainline kernel. The automatic integration of kernel extensions at build time will leave the kernel proper largely unencumbered. In contrast, the existing model litters the kernel proper with unnecessary #include statements and code fragments that do nothing until the corresponding CONFIG option is enabled. More important, Torvalds et al. no longer will need to decide what kernel extensions make it onto the mainline kernel.org release. Rather, since the extensions will be part of the mainline release, the kernel.org folks can declare a preferred composition of these extensions, but others will be truly free to choose what they prefer—i.e., no longer will there be a need to patch in code and resolve merge conflicts between separate extensions.

Building a toolkit to solve this problem is a tall order, and the C4 toolkit is just one approach. There are undoubtedly others. For more information and an initial release of the C4 toolkit, please see http://c4.cs.princeton.edu.

Thanks to USENIX Supporting Members

Addison-Wesley/Prentice Hall PTR	Microsoft Research
AMD	NetApp
Asian Development Bank	Oracle
Cambridge Computer Services, Inc.	OSDL
Delmar Learning	Perfect Order
Electronic Frontier Foundation	Raytheon
Eli Research	Ripe NCC
Hewlett-Packard	Splunk
IBM	Sun Microsystems, Inc.
Intel	Taos
Interhack	Tellme Networks
The Measurement Factory	UUNET Technologies, Inc.

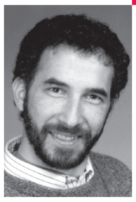
It is with the generous financial support of our supporting members that USENIX is able to fulfill its mission to:

- Foster technical excellence and innovation
- Support and disseminate research with a practical bias
- Provide a neutral forum for discussion of technical issues
- Encourage computing outreach into the community at large

We encourage your organization to become a supporting member. Send email to Catherine Allman, Sales Director, sales@usenix.org, or phone her at 510-528-8649 extension 32. For more information about memberships, see http://www.usenix.org/membership/classes.html.

PETER BAER GALVIN

Solaris 10 containers



Peter Baer Galvin is the chief technologist for Corporate Technologies, Inc., a systems integrator and VAR, and was the systems manager for Brown University's Computer Science Department. He is currently contributing editor for *SysAdmin Magazine*, where he manages the Solaris Corner, and is co-author of the *Operating Systems Concepts* and *Applied Operating Systems Concepts* textbooks.

pbg@petergalvin.info

THE CONCEPT IS SIMPLE: ALLOW multiple copies of Solaris to run within one physical system. Indeed, creating and using Solaris zones is simple for experienced system administrators. But then why are there so many questions surrounding this new Solaris feature? Just what is a container? How do you upgrade it? How can you limit its resource use? Does it work like VMware? And so on. In this article I describe the theory of Solaris 10 containers, and the facts behind creating, managing, and using them.

Overview

Solaris 10 containers are a new feature of Solaris. A container is a virtualized copy of Solaris 10, running within a Solaris 10 system. While this is similar in concept to VMware, for example, it is more of a distant relative than a blood brother. Perhaps its closest relative is BSD jails. Containers have a different purpose from VMware (and other operating system virtualization software, such as Xen). Those tools create a virtual layer on which multiple operating systems can run. Containers run within Solaris 10, act only as Solaris 10 environments, and only run Solaris applications.

Before I delve further into containers, a clarification is needed. Solaris 10 has the concepts of "zones" and "containers." Simply, a container is a zone with resource management (including fair-share scheduling) added. Mostly the terms are used interchangeably, but where needed I will point out differences.

Given that limited Solaris 10 view of virtualization, of what use are containers? Consider the following set of features that containers add to Solaris 10:

- Up to 8192 containers can exist on the same system. The "global zone" is what would normally be called "the operating system." All other containers are referred to as "non-global."
- Containers can share resources with the global zone, including binaries and libraries, to reduce disk-space requirements. An average container takes 60MB of disk space or less. A container sharing files with the global zone is known as a "sparse container" and is created via a sparse install.
- Because binaries are shared (by default), and Solaris optimizes memory use by sharing objects in memory when possible, an average container uses approximately 60MB of memory when booted.

- By default a package (Sun's concept of an installable application) installs in the global zone and all nonglobal zones. Likewise, a patch will by default install in all containers that contain the packages to which the patch refers.
- As mentioned above, a container is a resource-managed zone. The first release of Solaris 10, on which this article is based, includes CPU use as a manageable resource. (Note that there are now three streams of Solaris release: the standard commercial release; the "Express" updates that arrive every month or so, and for all intents are beta releases available to anyone interested; and the Open Solaris community release, which is a periodic snapshot of the internal build of Solaris based on the Open Solaris release, and is the least tested of these. This latter release is the most recent of the builds, but also the most likely to have problems.)

As with other OS-virtualizing technologies, containers are secure from each other, allowing only network access between them. They also have their own network configurations, having their own IP addresses and allowing variations in network membership (subnets and network masks, for instance).

There are also some limits that come with containers (most of these are likely to be removed in future releases of Solaris):

- A container cannot be an NFS server.
- A container cannot be moved to another system (i.e., imported or exported).
- A container must have a pre-set network address (i.e., it cannot be DHCP-configured).
- The only container-manageable resource as of the first release of Solaris is CPU shares. A container could, for example, use all the system's virtual memory.
- Due to the security restriction that a nonglobal container be securely separate from other containers, some features of Solaris 10 do not work in a container. The most limiting is DTrace, the extraordinary Solaris 10 debugging/analysis tool.
- A container cannot run Linux binaries natively (the initial container marketing from Sun to the contrary notwithstanding). Likewise not currently supported, a container cannot have its own firewall configuration. Solaris 10 uses ipfilters as its firewall technology; ipfilters can be configured in the global zone only.
- By definition, a container runs the same operating system release as the global zone. Even kernel patches installed on the system affect all containers. Only application patches or non-kernel operating system patches can vary between containers.

So what we are left with is a very lightweight, easy to manage, but in some ways limited application segregation facility. The same application could be run in multiple containers, use the same network ports, and be unaware of any of its brothers. A container can crash and reboot without affecting any other containers or the global zone. An application can run amok in a container, eating all available CPU but leaving the other containers with their guaranteed fair-share scheduling slices. A user with root access inside of a container may control that container, but cannot escape from the container to read or modify the global zone or any other containers. In fact, a container looks so much like a traditional system that it is a common mistake to think you are using the global zone when you are "contained." An easy navigation solution is found in the command zonename. It displays the name of the current zone. I suggest you have that output displayed as part of your prompt so that you always track the zone you are logging in to.

Creation

The creation of a container is a two-step process. The zonecfg command defines the configuration of a container. It can be used interactively or it can read a configuration file, such as:

```
create -b
set zonepath=/opt/zones/zone00
set autoboot=false
add inherit-pkg-dir
set dir=/lib
end
add inherit-pkg-dir
set dir=/platform
end
add inherit-pkg-dir
set dir=/sbin
end
add inherit-pkg-dir
set dir=/usr
end
add inherit-pkg-dir
set dir=/opt/sfw
end
add net
set address=131.106.62.10
set physical=bcme0
end
add rctl
set name=zone.cpu-shares
add value (priv=privileged,limit=1,action=none)
end
```

In the above example, I create zone00. It will not automatically boot when the system boots. It will be a sparse install, inheriting the binaries, libraries, and other static components from the global zone, for all of the inherit-pkg-dir directories added. It will have the given network address, using the Ethernet port known as bcme0. Finally, it will have a fair-share scheduler share of 1. (See below for more information on the fair-share scheduler.) The command zonecfg –z zone00 –f config-file will read the configuration (from the file named "config-file"). Either a sparse install or a full container install is possible. If there are no inherit-pkg-dir entries, all packages from the global zone are installed in full. Such a container can take 3GB or more of storage but is then less dependent on the global zone. Typically, sparse installation is used.

Several options are available with zonecfg. An important one is fs, which mounts file systems within the container. Not only can file systems from other hosts be mounted via NFS, but loopback mounts can be used to mount directories from the global zone with the container. For example, to mount /usr/local read only as /opt/local in zone00, interactively:

```
# zonecfg –z zone00
zonecfg:zone00> add fs
zonecfg:zone00:fs> set dir=/usr/local
zonecfg:zone00:fs> set special=/opt/local
zonecfg:zone00:fs> set type=lofs
zonecfg:zone00:fs> add options [ro,nodevices]
zonecfg:zone00:fs> end
```

When the zone is rebooted, the mount will be in place.

Next, zoneadm –z zone00 install will verify that the configuration information is complete, and if it is will perform the installation. The installation for the most part consists of package installations of all of the packages in the inherit-pkg-dir directories, but only those parts of the packages that are nonstatic (configuration files, for example). The directory under which the container will be created must exist and must have file mode 700 set. Typically, a container installation takes a few minutes.

Once the container is created, zoneadm –z zone00 boot will boot the container. The first boot will cause a sysidconfig to execute, which by default will prompt for the time zone, root password, name services information, and so on. Rather than answer those questions interactively, a configuration file can do the trick. For example:

```
name_service=DNS
{
domain_name=petergalvin.info
name_server=131.106.56.1
search=arp.com
}
network_interface=PRIMARY
{
hostname=zone00.petergalvin.info
}
timezone=US/Eastern
terminal=vt100
system_locale=C
timeserver=localhost
root_password=bwFOdwea2yBmc
security_policy=NONE
```

Placing this information in the sysidcfg file in the /etc directory of the container (/opt/zones/zone00/root/etc/sysidcfg) before the first boot provides most of the sysifconfig answers. Now the container can be booted. To connect to the container console, as root, you can use zlogin –C zone00. Here you can watch the boot output. Unfortunately, there is still an NFSv4 question to answer for sysidconfig, but once that is done the container is up and running. The first boot also invokes the new Solaris 10 service management facility, which analyzes the container and adds services as prescribed by the installed daemons and startup scripts. Future boots of the container only take a few seconds.

Management

Once a container has been installed and booted, it is fairly self-sufficient. The zoneadm list command will show the status of one or all containers.

I find a script to execute a command against every container is helpful. For example:

```
#!/bin/bash
for z in zone00 zone01 zone02 zone03 zone04
zone05 zone06 zone07 zone08 zone09 zone10
zone11 zone12 zone13 zone14 zone15 zone16
zone17 zone18 zone19 zone20;
do
zoneadm -z $z boot
zlogin -z $z hostname;
done
```

Note that the zlogin command, when run as root in the global zone, can execute a command in a specified zone without a password being given.

Another administrative convenience comes from the direct access the global zone has to the other containers' root file systems. Root in the global zone can copy files into the directory tree of any of the containers via the container's root directory.

By default, any package added to the global zone is added to every container on the system. Likewise, any patch will be added to every container that contains the files to which that patch applies. It is a bit surprising to watch pkgadd boot each installed but nonrunning container, install the packages, and then return the container to its previous state. But that is an example of just how well integrated into Solaris 10 containers are. There are options to the pkgadd and patchadd to override this behavior.

Fair-Share Scheduler

As previously mentioned, the fair-share scheduler can be used in conjunction with zones to make them into containers. A container then may be limited in its share of available CPU scheduling slices. Fair share is a complex scheduling algorithm in which an arbitrary number of shares are assigned to entities within the computer. A given entity (in this case a container, but it could be a collection of processes called a project) then is guaranteed its fair share of the CPU—that is, its share count divided by the total. If some CPU is unused, then anyone needing CPU may use it, but if there is more demand than CPU, all entities are given their share. There is even the notion of scheduling memory, in that an entity that used more than its fair share may get less than its share for a while when some other entity needs CPU.

The system on which a container runs must have the fair-share scheduler enabled. First, the scheduler is loaded and set to be the default via dispadmin –d FSS. The fair-share scheduler is now the default on the system (even surviving reboots), and all new processes are put into that scheduling class. Now all the current processes in the timesharing class (the previous default) can be moved into fair share by priocntl –s –c FSS –i class TS. Finally, I'll give the global zone some shares (in this case, five) so it cannot be starved by the containers: prctl –n zone.cpu-shares –v 5 –r –i zone global. To check the share status of a container (in this case, the global zone), use prctl –n zone.cpu-shares –i zone global. Take note that the prctl command share setting does not survive reboots, so this command should be added to an rc script to make it permanent.

Use

Containers are new, and therefore best practices are still in their infancy. Given the power of containers and the low overhead, I would certainly expect that most Solaris 10 systems will have containers configured. An obvious configuration is that only root users access the global zone, and all other users live in one or more containers. Another likely scenario is that each application be installed in a nonglobal container, or each into its own container, if it only communicates with the other applications on that system via networking. If a set of applications uses shared memory, then they obviously need to be in the same container. Much will depend on the support by software creators and vendors of containers. Early indications are good that vendors will support their applications being installed inside containers.

Some of the design decisions surrounding containers will limit how they are used. For example, it does not make sense to install a development, QA, and production container for a given application on the same system. Development might want to be using a different operating system release than the one currently in use in production, for example. Or QA may need to test a kernel patch. It would be reasonable to create a container for each developer on a development server, however, to keep rogue code from crashing that shared system or hogging the CPU. For those that host applications for others (say, application service providers), containers are a boon for managing and securing the multiple users or companies that use the applications on their servers.

Conclusion

The Solaris 10 container facility is a different take on virtualized operating system environments commonly found on other systems. Only one Solaris release may run on the system, and even kernel patches affect all containers. Beyond that, containers offer a cornucopia of features and functions. They are lightweight, so they may be arbitrarily created and deleted. They are secure, protecting themselves from other containers and protecting the global zone from all containers. And they can be resource managed (although only CPU resources at this point) to avoid a container from starving others. Further, containers are integrated into other aspects of Solaris 10, such as package and patch management. Containers are a welcome addition to Solaris 10 and allow for improved utilization of machines, as well as more security and manageability.

HOBBIT

DNS-based spam rejection

Hobbit has played the "Internet insecurity" game long enough to realize that it's hopeless as long as those bothersome and complacent humans are still in the loop. When he isn't working on heavy-handed approaches to spam control, he's probably outside hacking on the Prius.

hobbit@avian.org

TIRED OF BEING HAMMERED BY spam and virus attacks from spyware-infested ISP customers? Here is an easy and entertaining way to use the providers' own DNS naming schemes against them. In the absence of ISPs really doing anything on their end, we can use a few well-constructed filtering rules at our end to deny email delivery from suspect networks, and still leave a path open for legitimate messages.

It is no secret that a huge flood of spam and malicious email emerges from compromised machines in homes and businesses. The botnets grow larger by the day and have become a profitable underground offering. 0wned machines are used to launder connections and launch all sorts of attacks, using the convenient high-speed bandwidth in the customer infrastructure of the ISPs that connect them. Unfortunately, many major ISPs are behind the curve in preventing this, and obstinately remain so even though what they *should* be doing is common knowledge. They want to retain their neutral status as a carrier, not to be responsible for traffic filtering.

Many ISPs have been forced to block downstream packets to obvious problems such as Windows file-sharing ports, but they have put little effort into upstream filtering, thinking that it would cause too many (more) support complaints. Some, such as UUnet and Concentric, have denied direct SMTP delivery from their own untrusted customer clouds, and they've been successfully running such configurations for years. The result is that spam and virus/trojan attempts rarely, if ever, arrive from their infrastructures. I believe that Comcast began experimenting with filtering in the cable-modem swamps, but that seems to have been undone recently.

What is one to do about the rest of the ISPs that simply let the stuff out? Well, another area that ISPs do seem to pay more attention to is DNS naming within the customer networks, and it turns out we can rely on that much more than their ability to keep a lid on the botnets. Largely automated within turn-key DHCP servers, each address that appears on cable and DSL networks generally receives some kind of valid PTR record in the DNS server authoritative for those .IN-ADDR.ARPA blocks. And since the naming is machine-generated, based on the client's IP or MAC address, it usually follows some recognizable pattern such as these:

```
c-24-128-171-15.hsd1.ma.comcast.net
pcp05184511pcs.plsntv01.nj.comcast.net
```

```
pool-151-203-213-167.bos.east.verizon.net
fl-71-0-153-49.dyn.sprint-hsd.net
15-95.200-68.tampabay.res.rr.com
dsl-67-114-79-114.dsl.lsan03.pacbell.net
CPE0008a10ba047-CM014100000470.cpe.net.cable.rogers.com
```

Almost all of these names imply something about the client and/or the surrounding network infrastructure. They are unlikely to be applied to infrastructure machines for the ISP itself, such as the mail-relay servers customers are expected to use for their outbound mail. Despite the fact that DNS is considered a generally untrustworthy source of data, these PTR names are at least somewhat trustable, since an SMTP server will generally look them up "out of band" via servers that are less likely to be under a spammer's control. If a spammer is intercepting all of your DNS queries, then you've got a much larger problem.

Thus, as *part of* an overall best-practice set of anti-spam measures, we can take advantage of such naming schemes and detect if a generic customer is attempting to deliver mail directly, instead of passing it through the local ISP. We can then deny delivery with a rejection message asking the sender (if it's a human paying attention at all) to please use the ISP's authorized relayer to send the message. I do this within Postfix, and all examples herein are based on the filtering features that Postfix has to offer, but the concepts should easily map to other common SMTP servers. We can hope that the ISP mail relay also applies a few anti-spoofing rules to make sure the headers aren't forged—this is in fact probably not the case in most instances due to the management overhead, but as luck would have it, the mechanism works because most ISP customers get configured by default to drop off their mail at the ISP relay. The few who want to do their own direct sending will see the error and, hopefully, resend the message via the ISP to get it through. (Mildly political rant, below, about the relative merits of each path.)

So, how is this implemented? In Postfix, the SMTP server is able to look up and act upon details about the client connecting to it. Restrictions may be imposed by adding one or more instructions to check client name or IP attributes in main.cf:

```
smtpd_xxx_restrictions =
   ...,
   check_client_access regexp:/etc/postfix/client_acl,
   ...
```

where xxx can indicate one of several stages of SMTP delivery. The most common section is smtpd_recipient_restrictions, which allows collecting as much log detail about the client and the message as possible. The regexp: tag can also be hash: or dbm: or pcre:, depending on which style of data storage is desired and which version of Postfix, but it is likely that only regular-expression-based matching will be useful here. The client access directive is usually found somewhere in a list amidst several other types of checks and special cases—allowing local or authenticated connections, looking up addresses in RBL services, etc.

The client_acl file itself contains an ordered list of expressions to match, actions to take upon match, and optional error-message text for rejections. Here's where the DNS-based magic happens, along with any other network-level checks needed. Rules are matched for both DNS lookups and the ASCII representation of the IP address, which rocks because of the seamless versatility that it lends. Some examples:

```
# *all* of cybermall, 207.0.62.0/23
/^207\.0\.6[23]\./                    550 No Soliciting
# oops, several of yahoo's legit relays keep landing in spamcop
+^216\.155\.201\.[56]+                OK
# block all of Latvia [I don't know anyone there]
/\.lv$/                                     550 Denied
```

```
# random known spam-nests
/cablemas\.net$/                    REJECT
/\.popsite\.net$/                   550 Access denied (spam)
```

Note that it's still all regexp string comparisons, so we can't specify CIDR notation, but that's fine—to deny partial blocks or aggregates, the [set] syntax of address digits gets close enough. I tend to err on the broader side if I'm slamming the door on some podunk ISP that appears to be spammer-friendly. Postfix regular expressions have a couple of very minor differences from normal ones, and by default are case-insensitive unless a modifier is used. Rule processing is normally "at first match, take the action and exit the list."

Inside the set of client rules, two approaches can be taken for filtering a given provider: either whitelisting its legitimate relay servers and denying the rest, or blacklisting its known customer networks using patterns. The approach taken often depends upon how the ISP does its naming, where it locates its mailservers, etc. It takes a bit of digging and actually *reading* the spam carefully to get a clear picture of their infrastructure, but once that's done the rest is easy.

More examples—first, of names that lend themselves to easy one-shot identification:

```
# general classes of dialup/DHCP clients
/[-.]dial/  550 Use your ISP's mail relay
/dial[-inu.]/                       550 Use your ISP's mail relay
/dhcp[-0-9]/                        550 Use your ISP's mail relay
# AOL direct-dialup customers get dropped into this swamp
/ipt\.aol\.com$/                    550 Use your ISP's mail relay
/ipt\.aol\.net$/                    550 Use your ISP's mail relay
# ATTBI dynamics—client-mumbledyfoo.attbi.com
/client.*\.attbi\.com$/             550 Use your ISP's mail relay
# typical comcast client naming
/client.*\.comcast\.net$/           550 Use your ISP's mail relay
/^pc.*\.comcast\.net$/              550 Use your ISP's mail relay
/\.hsd[123]..*\.comcast\.net$/      550 Use your ISP's mail relay
```

An example with special-casing—I have some correspondents in the Albany, NY area who usually send directly, so I let them in ahead of the rest of Verizon's dynamic blocks. Even trying to keep it this tight still lets spammers connect once in a while:

```
/pool-129.*\.alb\.east\.verizon\./  OK
/pool-141.*\.alb\.east\.verizon\./  OK
# all dhcp/pppoe verizon clients seem to match this...
/pool.*verizon\.net$/               550 Use your ISP's mail relay
```

For Roadrunner, looking up their MXes gives a pretty clear picture of which naming classes are likely to deliver legitimate mail—usually from something.mgw.rr.com, but over time I discovered some additional ones:

```
/mgw\.rr\.com$/                     OK
/smtp-.*\.rr\.com$/                 OK
/mx[-0123].*\.rr\.com$/             OK
/\.rr\.com$/                        550 Use your ISP's mail relay
```

An example done purely by IP address—a particular business I was dealing with is unfortunately located within one of Electric Lightwave's netblocks. ELI appears to be a spammer petri dish, so they don't get to talk to me, period. Rather than ask the business to change to a better provider, we can allow mail from a small chunk they're in and then deny the rest of their main /15:

```
## SPL-case: SR-systems sits within an ELI block, but let 'em in
/^208\.187\.213\./                  OK
/^208.18[67]\./                     550 Denied (ELI, spam)
```

Obviously, techniques like this aren't for everyone and are certainly not a solution by themselves, but they do kill quite a bit of spam once the major botnet offenders are added in correctly. It is prudent to research as much as is externally visible about each ISP—using spam we receive through them, legitimate messages we receive, MX lookups, their customer-service Web pages, and possibly even Googling for archived message headers in postings from some of their customers just to read how their mail was delivered. For large IP blocks it may be better to use no-logging firewall rules to completely hide the mailservers from them and avoid piling up big log files of rejections. As jingoistic as it may seem, I have had to occasionally deny huge RIPE/APNIC/LACNIC allocations right up front just to stem the tide.

There are also many other techniques that can be done inside Postfix that are beyond the scope of this article. There are plenty of FAQs on spam-busting kicking around and pointed to from the Postfix.org Web site. A nice feature of using Postfix's own features, such as client parsing, is that it's fast—the regular expression matching runs natively right inside Postfix and doesn't need to farm out to external plug-ins or start up extra processes and Perl interpreters. SpamAssassin and the like seem somewhat less attractive because they sit outside Postfix and chew much more CPU. But if the external packages have the other features you need, then by all means use them.

Some ISP customers, such as small businesses and consultants, may have a legitimate need to handle their own mail and deliver directly. Unfortunately, the ISPs often tar them with the same dynamic-DNS brush, so when the business believes it's acewidgets.com, the outside world sees its SMTP connections coming from c-67-163-134-87.hsd1.ct.comcast.net. The right answer for this is either delegated reverse DNS as described in RFC 2317, or for the ISP to maintain a nominal set of static PTR records for that customer. However, finding anyone within the ISPs who even knows what that means, let alone how to set up and maintain it, is often an insurmountable challenge.

Forcing customers through ISP mailservers is also a political gray area, as well as a certain amount of maintenance headache for the ISP. It doesn't have to be. The relay servers are usually already in place, since most customers appear to get set up to deliver through them by default. But those servers also present a good opportunity for the ISP to enforce anti-spoofing in message envelopes and headers and be a better neighbor to the rest of the Internet, especially with today's nonstop epidemic of forged headers and joe-jobs coming from compromised customer machines. It is highly worthwhile for them to build infrastructure that is sufficiently capable and configurable to support that enforcement but still handle some special cases, such as letting acewidgets.com deliver as acewidgets.com, and then lock down the customer subnets. This of course requires having those same people who are clueless about delegated DNS come to understand how to run a proper mailserver.

If I seem to be down on ISP technical capability in general, it's for very sound historical reasons. Really, I have tried to get through to many of them and have consistently failed to find anyone who knew what I was talking about. Perhaps if you're reading this and work for one of these ISPs, you can take it to management as a wake-up call. The more ISPs realize the extraordinary extent of the problems and put forth the effort to be better Internet neighbors, the less need there will be for kludges like all of the above.

REFERENCES

"Classless IN-ADDR.ARPA delegation": http://rfc-editor.org/rfc/rfc2317.txt

Postfix FAQs: http://www.postfix.org/docs.html

BORIS LOZA

finding trojans for fun and profit

Boris Loza, Ph.D., is a founder of Tego System Inc. and HackerProof Technology, in addition to being a contributor to many industry magazines. He holds several patents and is an expert in computer security.

■ *bloza@hackerproofonline.com*

"THE TROJAN HORSE" ORIGINALLY referred to the ploy used by the ancient Greeks to attack the city of Troy. Today, it's fairly common knowledge that a trojan horse is an application a cheeky hacker tries to install on your hard disk to get easy access to your computer. A trojan can be part of a rootkit while masquerading as a legitimate application such as ls, df, or ps. In this article I will show you how to find rootkits and trojans using handy little utilities and a couple of tricks.

Checking Inodes

One of the ways to find trojan files in a current directory is to check inode numbers. Many rootkits modify the access and modification time of the files they replace, so at a glance a file may appear to be unchanged or even untouched. What remains is to check an inode number of a file in question.

Most installs will install files sequentially. For example, the output below shows inode numbers for files in the /etc directory:

```
$ ls –ai /etc | sort | more
......
183491 TIMEZONE
183492 autopush
183493 cfgadm
183494 clri
183495 crash
183496 cron
......
```

The –i option of ls lists the files' inode numbers. As you can see from the output, most of the inode numbers are in sequence.

A broken number sequence indicates the possibility that those files were installed after the main installation took place. Look for out-of-place entries, either very high or very low. Also look for new groupings, as all the rootkit pieces were probably installed at the same time.

Note: The newfs command uses fsirand(1M) to install random inodes when creating a new file system. Also, if you use fsirand periodically, your system inode numbers will not be in sequence. For this reason, you may want to create a master database of all inode numbers for all your files. You can use something like the following to collect this information into a file:

```
# ls –aiR / > my_inodes
```

Put this database aside and check the inode numbers of files in question against it. Update the database after installing new patches or system upgrades.

Check closely the /usr, /usr/bin, /sbin, /usr/sbin, and your X Window binaries directory, because rootkits are usually hiding in these places.

If an attack was successful, a hacker may install a rootkit. This is a suite of applications that can be used for many nasty things (creating back doors, root shells, etc.). It also helps to hide its own presence by modifying system commands that, for example, list all files in the directory (ls, dir (on Linux)) or find any file (find). Therefore, if you suspect that an attacker is on your system, you may not want to trust the ls or find commands, because they most likely have been replaced. How, then, do you list all files in the directory/s to find the rootkit's files?

Alternative Ways to List Files in a Directory

If a rootkit has been installed on your system, it replaced the ls and find commands with trojan versions that will not show a real list, one that includes the rootkit's files.

If you suspect that the current directory may contain a hidden directory or file, do one of the following. If you use Korn shell (ksh), press "ESC=" to list all files in a directory. For example, on Solaris OS:

```
$ ksh –o vi
$ .<ESC>=
1) ../
2) ./
3) .Xauthority
4) .dt/
5) .dtprofile
6) .hushlogin
7) .netrc
8) .rhosts
9) .sh_history
```

Or:

```
$ *
 1) TT_DB/      12) mnt/
 2) bin/        13) net/
 3) cdrom/      14) opt/
 4) dev/        15) platform/
 5) devices/    16) proc/
 6) etc/        17) sbin/
 7) export/     18) tmp/
 8) home/       19) usr/
 9) kernel/     20) var/
10) lib/        21) vol/
11) lost+found/ 22) xfn/
```

While your ls might be trojaned and not be able to see the hidden files, your Korn shell will see them.

You can also use the echo(1) command, which lists all files in a directory. For example:

```
$ echo *
TT_DB bin cdrom core dev devices downloads etc export home kernel
lib lost+found mnt mynes.txt net nsmail opt patches platform proc proj-
ects sbin test tmp ts1 typescript usr var vol xfn
```

Note: Echo will not show hidden files, such as files starting with ".".

On Linux systems, you may use the less(1) command to display all files in the directory:

```
$ less .
2 drwxr-xr-x    3 boris   other          512  Jun   4     20:59 ./
2 drwxr-xr-x  43 root     sys           1024  May 26    20:05 ../
2 -rw———-    1 boris   other            90  Jun   4    21:14 .bash_profile
...
```

To list all files in the directory you can also use the tar(1) utility. Use the -w option for "wait for user confirmation." Answer "y" for the first entry and "n" for the rest of the list:

```
$ tar cvwf /tmp/bb .
r drwxr-xr-x     1003/1     1024 Jun   1   17:35 2005 .: y
a ./ 0K
r -rw———-     1003/1     3918 Jun   4   20:34 2005 ./.sh_history: n
r -rw-r—r—     1003/1      418 Mar 22  14:43 2004 ./.profile: n ^C
```

Using od and cat Commands

Because a directory is also a file, we will use commands that can be used to look inside files: od and cat. (You cannot use od(1) or cat(1) to display directories on Linux).

First, we will try to get the octal dump (using the od command) of the directory to look for all files. Let's try the following:

```
$ od -c .
0000000  \0  \b 023 337  \0  \f  \0 001   .  \0  \0  \0  \0 013 006   P
0000020  \0  \f  \0 002   .   .  \0  \0  \0  \b 023 340  \0 020  \0 007
0000040   P   r   o   j   e   c   t \0  \0  \b 023 341  \0 024  \0 013
0000060   w   e   b   s   t   a   t   .   l   o   g \0  \0  \b 023 342
0000100 001 304  \0 006   s   t   a   t   u   s \0  \0  \0  \0  \0  \0
0000120  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0
*
0001000
$
```

The od command with -c option displays single-byte characters. Certain nongraphic characters appear as C-language escapes:

null	\0
backspace	\b
form-feed	\f
new-line	\n
return	\r
tab	\t

Others appear as 3-digit octal numbers.

The od -c command starts each line with the number of bytes, in octal, shown since the start of the file. The first line starts at byte 0. The second line starts at byte 20 (that's byte 16 in decimal, the way most of us count). And so on. One can easily find file names on this output (., .., Project, webstat.log, and status).

The cat -v –t –e turns nonprintable characters into a printable form.

A directory usually has some long lines, so it's a good idea to pipe cat's output through fold:

```
$ cat -v -t -e . | fold -62
^@^H^SM-
_^@^L^@^A.^@^@^@^@^K^FP^@^L^@^B..^@^@^@^H^SM-
`^@^P^@^G
Project^@^@^H^SM-a^AM-X^@^Kwebstat.log^@^@^@^@^@^AM-
D^@^Fstatu
s^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@
^@^@^@^@^@^@^@^@^
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
```

```
@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^
@^@^@^@^@^@^@^@^@^
@^@^@^@^@$
```

You may try to filter out nonprintable characters. You can use something like the following (or use your own method):

```
$ cat -v -t -e . | fold | sed "s/[\x00-\x08\x80-\x88\x0B-\x19\
x8B-\x99\x7F\xFF]"//g
```

Or:

```
$ od -c . | sed  "s/[\001-\010\013-\037\177-\377]//g"
```

Note: Be aware that some files shown by od or cat may be files that have already been deleted.

Conclusion

Obviously, if your ls command doesn't show the same files as any of these commands, beware (remember not to count deleted files!). This may be because your ls has been replaced by a hacker. Do not panic! This is not the end of the world. Open your favorite book on incident response and OS recovery.

You can find more hands-on tips and tricks of this kind in my new book *UNIX, Solaris and Linux: A Practical Security Cookbook*, which deals with securing UNIX OS without any of the third-party tools. It can be reviewed at www.amazon.com.

ROBERT HASKINS

ISPadmin

EMBEDDED HARDWARE

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 Renesys Corporation, a leader in real-time Internet connectivity monitoring and reporting. He is lead author of *Slamming Spam: A Guide for System Administrators* (Addison-Wesley, 2005).

rhaskins@usenix.org

IN THIS EDITION OF ISPADMIN, I take a look at the use of embedded systems for ISPs and other networks. There are many embedded platforms and open source software applications available for those interested in deploying wireless access points and firewalls for an ISP or any network. It's not possible to cover all the hardware and software in the space available here, so I've provided overviews and pointers to other resources.

What Are Embedded Systems?

According to Wikipedia, an embedded system is "a special-purpose computer system, which is completely encapsulated by the device it controls" [1]. Embedded systems differ from general-purpose computers (such as PCs) by their programming to accomplish specific tasks. Many different types of embedded systems are in use on large networks, most notably switches and routers. Beyond their specific programming, embedded systems are usually characterized by:

- Low power utilization
- Small size and comparatively lower cost
- Limited use of active cooling devices (fans)
- Limited CPU and memory

To keep the devices in a small package, many design decisions must be made. First of all, the small size dictates a small power supply, which limits the number of components and the speed of the CPU (the faster the CPU, the more power and cooling is used). Less power also means that fewer memory chips can be used. All of these requirements lead to lower cost, which makes the platform attractive for all of the reasons larger devices (such as traditional PCs) are not.

Often, embedded systems are designed to use Compact Flash (CF) or similar types of non-volatile storage rather than hard disk drives. While reducing heat cost and increasing reliability by using fewer moving parts, the lack of a hard drive does make for some added complexity in design. It also requires "stripped down" versions of operating systems to run successfully and can limit log retention due to space and read/write cycles. CF memory is limited in the number of times the memory gates can be switched on or off, so the memory must be replaced after this threshold is exceeded.

Available Embedded Platforms

For the purposes of this discussion, platforms are broken down into two types, general-purpose and re-purposed. Rather than performing any specific function per se, general-purpose platforms are meant to accomplish different functions depending on the software loaded. Re-purposed platforms are commercial products that are loaded with alternate firmware and may or may not be used in the same applications that the original creators intended them for.

There are a number of general-purpose embedded platforms available on the market today. The more common, lower-cost platforms include:

- Soekris Engineering [2]
- PCEngines Wireless Router Application Platform [3]
- PC/104 [4]
- Mini-ITX [5]

The Soekris and PCEngines platforms are each particular to the vendors who created them and are not standards per se. The PC/104 and Mini-ITX platforms are standards in the sense that multiple companies manufacture boards in those formats.

Due to space constraints, only Soekris and Mini-ITX platforms will be covered in this article. However, many of the software packages mentioned run on the other platforms listed above. On the re-purposed platform side, the big target currently is Linksys hardware (and other hardware manufacturers who use similar chipsets). Only the Linksys WRT54G is covered in this article on the re-purposed hardware side.

Soekris Engineering

Soekris boards are one of the more widely used general-purpose embedded platforms available today. The Soekris platform consists of an AMD Geode [6] 100 to 266MHz processor and 16 to 256MB of memory (specifications depend on the model). Flash memory (long-term storage) is provided on board or through a CF socket. All Soekris models come with two or three Ethernet ports, which makes them ideal for firewalls or for aggregating multiple wireless access points/networks into one egress/firewall point. Soekris boards also support power over Ethernet, making them ideal for situations where no traditional 110V AC power is available.

All Soekris boards are available with at least one mini-PCI slot. Some models are available with regular PCI slots, which makes it easy to add functionality, assuming the add-in PCI board consumes little power and runs at 3.3V. However, be aware that the default Soekris case doesn't give external access to the slot, nor is it big enough for a full-size PCI card. These mini-PCI and regular PCI cards are often used for 802.11 wireless network cards.

Mini-ITX

Mini-ITX is a form-factor specification developed by VIA after the purchase of Cyrix from National Semiconductor and Centaur (WinChip) from IDT [7].

Most of the Mini-ITX products are meant to be low-cost PC desktop solutions, as most of the line includes faster processors and active cooling (fans). This is emphasized by the fact that most of the form factors for the Mini-ITX cases are desktop, set-top box, or other hobbyist type of enclosures. However, VIA clearly understands the potential for the embedded market, as it has several boards that do not require cooling fans and that do contain multiple Ethernet ports. These attributes make those VIA product lines ideal for the embedded market.

Embedded System Software

There are many choices when it comes to running software on your embedded device, and no possible way to cover them all here. If you need specialized ISP-type services (e.g., RADIUS, QoS), you must use an environment that allows packages to be added as desired and not a GUI environment such as m0n0wall (see next section). The available software platforms range from build your own from scratch, to flash images that can be loaded directly on flash and then onto the embedded box.

If you want the quickest start, the preprogrammed flash is the way to go. However, using a distribution such as Voyage Linux [8] is only a little more work, but you gain a lot of flexibility. One complication is that Voyage (and similar environments) requires an existing Linux machine to run the install script. If you want to build the Voyage kernel, environment, and/or additional packages, the machine must run a Debian Linux. Pebble [9] is another popular stripped-down Debian Linux environment for embedded/wireless applications.

Numerous BSD-flavored environments are available because the Soekris line of boards seems to have a bias toward this UNIX variant. BSD variants that run on Soekris include m0n0BSD [10] and wifiBSD [11], and there's a nice write-up on installing embedded FreeBSD on Soekris net4801 hardware [12]. Many people seem to roll their own BSD variant for use on embedded hardware [13]. As with Linux, if you want to build your own embedded BSD-based environment, you must first have a BSD-based development system from which to work.

Pre-Packaged Firewalls

If all you require is a firewall, then specialized GUI environments are for you. One of the nicest ones available that runs on Soekris and other PC hardware is m0n0wall [14]. It is a FreeBSD-based system, stripped down, with a very nice GUI; another one is IPcop [15]. However, ease of installation and management comes at the price of flexibility, as you cannot easily add functionality to these firewalls.

Price/Performance Comparison

From a cost/performance standpoint, VIA is clearly the leader—500MHz VIA boards for embedded applications can be purchased for approximately US$100 as of this writing. Add a case, power supply, CF-to-IDE adapter, RAM, and a 128MB CF card and the cost of an embedded Mini-ITX system is comparable to the low-end Soekris models. Obviously, the performance is much greater for the Mini-ITX system; the Soekris boards run at only 100 to 266MHz and have smaller amounts of memory, but they also require very little power (10 watts for a 266MHz net4801 system including a pair of wireless cards).

The inclusion of the typical PC devices on the Mini-ITX platform (VGA adapter, mouse, keyboard, parallel, sound, etc.) does factor in, however. The "wasted" space on the printed circuit board, at least for the embedded application, means that the package for the Mini-ITX form factor can never be as small as the Soekris. The added components will increase power requirements for Mini-ITX as well. The small size of the Soekris (as well as PC/104 and WRAP) form factor is certainly useful in applications where space and power are at an absolute premium.

Re-Purposing Hardware

First, a warning: Re-flashing firmware on Linksys (and similar devices) is going to void your warranty and support agreements, so be sure you know what you are doing before embarking on any project that involves such activity!

On the low-cost embedded hardware side, devices such as the Linksys NSLU2 [16] and Linksys and WRT54G [17] can be re-purposed. The NSLU2 is designed to be a network storage device that allows a USB hard drive to be accessed across the network. It runs proprietary firmware, but this can be changed by installing one of the NSLU2-Linux project software images [18]. One of the firmware releases allows the user to add other hardware devices, turning your NSLU2 into a firewall or otherwise increasing its usefulness.

Most commonly in the ISP market, the WRT54G device is used for enabling wireless access points at low cost. As shipped by Linksys, built-in functions of the WRT54G include:

- Wireless radio
- Five-port Ethernet switch
- Consumer-grade router
- Firewall

Similar chipsets are used by products offered by a number of manufacturers, including Asus, Buffalo, Motorola, and Siemens, among others, according to the OpenWRT Table of Hardware site [19]. The OpenWRT software will run with varying degrees of support on most of these other platforms. The hardware specifications for the Linksys WRT54G version 3.0 device, according to [19], are the following:

- Broadcom 4712 chipset running at 200MHz
- 4MB flash/16MB RAM

At a current street price of US$50 for the WRT54G, that's an excellent price/performance ratio! Products marketed specifically to the service provider with similar functionality start around US$100.

Software for Re-Purposing Devices

ISPs don't use consumer-grade products such as the Linksys WRT54G because the firmware as shipped from Linksys doesn't include authentication methods (such as RADIUS) and other functionality needed for working with their networks. These features are often found in higher-priced devices aimed specifically at the service-provider market, but are lacking in consumer-grade devices.

To rectify this, several people have put out replacement firmware for the WRT54G (and similar) devices. Two of the more widely used ones are:

- Sveasoft Talisman [20]
- OpenWRT [21]

For someone wanting an easy, drop-in network solution, Sveasoft is probably better. It is open source, but a small fee is charged for the latest versions of firmware/support. While OpenWRT is more flexible due to its modular framework for adding packages, more time must be invested to configure the exact image you want. If the package you need isn't available, you may be able to port it to the OpenWRT environment and load it into your custom image.

Conclusion

Embedded hardware is in use at many ISPs and other larger networks. The application for such hardware is usually wireless access points and/or firewalls. There are many embedded hardware solutions available to the ISP or hobbyist today. These include general-purpose platforms such as Soekris Engineering and Mini-ITX, as well as re-purposed hardware such as Linksys WRT54G.

On the software side for general-purpose embedded hardware, stripped-down versions of Linux and BSD variants are in wide use. Images can be readily obtained and loaded via CF and similar media. GUI front ends
are available for firewall applications as well. For re-purposed hardware such as WRT54G, several Linux distributions are available that enable additional functionality for ISPs, including RADIUS and other service provider requirements.

REFERENCES

[1] Wikipedia Embedded Systems definition: http://en.wikipedia.org/wiki/Embedded_systems.

[2] Soekris Engineering: http://www.soekris.com/.

[3] PCEngines WRAP: http://www.pcengines.ch/wrap.htm.

[4] PC104 Consortium: http://www.pc104.org/.

[5] VIA's Mini-ITX page: http://www.via.com.tw/en/initiatives/spearhead/mini-itx/.

[6] AMD Geode: http://www.amd.com/us-en/ConnectivitySolutions/ProductInformation/0,,50_2330_9863,00.html.

[7] For a brief history of Mini-ITX, see http://www.mini-itx.com/hardware/history/.

[8] Voyage Linux: http://www.voyage.hk/software/voyage.html.

[9] Pebble Linux: http://www.nycwireless.net/pebble/.

[10] m0n0BSD: http://www.m0n0.ch/bsd/.

[11] wifiBSD: http://www.wifibsd.org/.

[12] Onlamp embedded BSD article: http://www.onlamp.com/pub/a/bsd/2004/03/11/Big_Scary_Daemons.html.

[13] For a nice description of how to reduce FreeBSD for use in embedded systems, see https://neon1.net/misc/minibsd.html.

[14] m0n0wall: http://www.m0n0.ch/wall/.

[15] IPcop: http://www.ipcop.org/.

[16] Linksys NSLU2 network<==>USB storage link: http://www.linksys.com/servlet/Satellite?childpagename=US%2FLayout&packedargs=c%3DL_Product_C2%26cid%3D1115416906769&pagename=Linksys%2FCommon%2FVisitorWrapper.

[17] Linksys WRT54G: http://www.linksys.com/servlet/Satellite?childpagename=US%2FLayout&packedargs=c%3DL_Product_C2%26cid%3D1115416825557&pagename=Linksys%2FCommon%2FVisitorWrapper.

[18] NSLU2 Linux project: http://www.nslu2-linux.org/.

[19] OpenWRT table of supported hardware: http://openwrt.org/TableOfHardware.

[20] Sveasoft WRT54G replacement firmware: http://www.sveasoft.com/content/view/20/1/.

[21] OpenWRT project: http://openwrt.org/.

DAVID MALONE

security through obscurity

A REVIEW OF A FEW OF FREEBSD'S LESSER-KNOWN SECURITY CAPABILITIES

David is a system administrator at Trinity College, Dublin, a researcher in NUI Maynooth, and a committer on the FreeBSD project. He likes to express himself on technical matters, and so has a Ph.D. in mathematics and is the co-author of *IPv6 Network Administration* (O'Reilly, 2005).

dwmalone@maths.tcd.ie

IN THIS ARTICLE I'M GOING TO LOOK at some less well known security features of FreeBSD. Some of these features are common to all the BSDs. Others are FreeBSD-specific or have been extended in some way in FreeBSD. The features that I will be discussing are available in FreeBSD 5.4.

First, I'll mention some features that I don't plan to cover. FreeBSD jails are like a more powerful version of chroot. Like chroot they are restricted to a subtree of the file system, but users (including root) in a jail are also restricted in terms of networking and system calls. It should be safe to give root access to a jail to an untrusted user, making a jail like virtual system, not unlike some applications of UML [1] or Xen [2] but without running separate kernels for each system. A lot has already been written about jails [3], so I won't dwell on them further here.

Another feature that I don't plan to spend much time on is ACLs. ACLs are an extension of the traditional UNIX permissions system to allow you to specify permissions for users and groups other than the file's owner and group. Since ACLs are familiar to many people from Solaris, Linux, and Windows, to name just a few examples, I'll just refer to [4] for the details on FreeBSD.

File Flags

A lesser-known set of extended permissions is the "file flags" supported by BSD's UFS file system. Of interest to us here are the append-only, immutable, and undeletable flags. These names are reasonably self-explanatory: append-only files can only be appended to, immutable files cannot be changed in any way, and undeletable files cannot be deleted (or have their hard links removed—all names referring to the inode are protected).

Each of these flags comes in two flavors: system and user. System flags can only be set and cleared by root. User flags can be set and cleared by the file's owner and root. The `chflags` command can be used to set them and the `-o` flag to `ls` can be used to display them. With these commands the names used for the system version of the flags are `sappnd`, `schg`, and `sunlnk`, and the user versions are `uappnd`, `uchg`, and `uunlnk`.

For example, it often surprises newcomers to UNIX permissions that a file can be deleted by any user who can write to the directory the file is in. Below, user lmalone has set the undeletable flag, and now user dwmalone cannot remove it.

```
dwmalone@hostname% ls -ldo . normal undeleteable
drwxr-xr-x  19 dwmalone  wheel  -      3584 May 14 09:03 .
-rw-r—r—    1 lmalone   wheel  -        0 May 14 09:03 normal
-rw-r—r—    1 lmalone   wheel  uunlnk    0 May 14 09:03 undeleteable
dwmalone@hostname% rm normal undeleteable
override rw-r—r—  lmalone/wheel for normal? y
override rw-r—r—  lmalone/wheel uunlnk for undeleteable? y
rm: undeleteable: Operation not permitted
```

Even root cannot remove this file, until the flag has been cleared manually:

```
root@hostname# rm undeleteable
rm: undeleteable: Operation not permitted
root@hostname# chflags nouunlnk undeleteable
root@hostname# rm undeleteable
```

There are some obvious applications for these flags. Append-only files can be used as log files, protecting against accidental or malicious truncation. FreeBSD installs a number of important files as system immutable (libc, init) to keep them from being damaged accidentally.

The immutable flag can also be used to prevent people "stealing" a link to an SUID executable. Usually, in any directory for which a person has write permissions, he can make a hard link to any file in that filesystem for which he has read permissions. This means that if someone knows there is a vulnerability to be announced in some SUID executable, he can steal a hard link to it and still have access to the executable after the original file appears to the sysadmin to be deleted. If an immutable flag is set on a file, these sorts of games aren't possible [5].

Note that file flags can only be manipulated locally and cannot be set or cleared over NFS. This means that marking a file on an NFS server as immutable is a good way to keep anyone from changing it.

BSD Secure Level

As I described above, the file system flags provide some useful flexibility, and even some protection against shooting oneself in the foot as root. However, they provide little protection against a malicious root user, who could just clear all the flags before going about their nefarious business.

The BSD operating systems do provide a simple form of protection against a malicious root user in the form of numbered "secure levels," in which the higher the number, the greater the restrictions on what can be done. The secure level can be raised using the sysctl command, but cannot be lowered while the system is running. On FreeBSD the secure level is set to -1 by default, but the secure level for multi-user operation can be set in /etc/rc.conf by adding settings such as:

```
kern_securelevel_enable="YES"
kern_securelevel="2"
```

The restrictions placed on system operation at each secure level are as follows:

If secure level > 0 you can't:

- access hardware from user processes via /dev/mem, /dev/pci, I/O instructions, and so on;
- load or unload kernel modules;
- change system-level file system flags (unlink, immutable, append);
- run a debugger on init;
- or cause /dev/random to perform a reseeding operation by writing to it.

If secure level > 1 you also can't:

- open disks in /dev for writing (including SCSI pass-through devices);

- change firewall rules;
- or run the clock faster than twice its normal speed or turn time backwards.

If secure level > 2 you also can't:

- change certain secondary firewall features such as ipf's NAT and ipfw's dummynet configuration;
- change certain sysctl values (msgbuf_clear, ipport_reserved{high, low}).

At secure level 1, root can't change immutable files. A malicious root might decide to unmount the file system and edit the raw disk, circumventing file permissions, flags, and ACLS. At secure level 2, this isn't possible, as disks cannot be opened for writing. Interestingly, each jail actually has its own secure level, so a jail can run at a higher secure level than the host system.

This means that a careful combination of a high secure level and UFS file flags can prevent an intruder from installing rogue kernels or kernel modules, the sort of trickery described in Rik Farrow's "Musings" column last April [6]. For this to work, *all* the files and directories involved in the boot process need to be immutable—this would include /boot, /sbin, /etc, /bin, /lib, /usr/bin, /usr/lib, etc.

In practice, this isn't often done, as it reduces the amount that can be achieved with online system administration. To update libraries the system must be rebooted and the library installed in single-user mode before the secure level is raised.

I did say that the secure level could not be lowered while the system is running. There is a way around this that I have used. If you have chosen to include kernel debugger support in your kernel, then someone with access to the kernel debugger can reduce the secure level. For example, I can use CTRL+ALT+ESC to get to the debugger on the console of my server:

```
root@hostname# sysctl kern.securelevel=2
kern.securelevel: -1 -> 2
root@hostname# KDB: enter: manual escape to debugger
[thread pid 13 tid 100001 ]
Stopped at      kdb_enter+0x2f: nop
db> write securelevel 0
securelevel              0x2    =    0
db> continue
root@hostname# sysctl kern.securelevel
kern.securelevel: 0
```

This technique allows an administrator to run the system at a high secure level when appropriate but to lower it when needed. It is important to remember that access to the kernel debugger requires physical access to the system (either to the console or via FireWire) and requires debugger support in the kernel.

MAC Framework

The MAC (Mandatory Access Control) framework is part of the excellent work done by the TrustedBSD project [7] to bring new security features to FreeBSD. The MAC framework allows people to develop kernel modules that provide additional checks on what the kernel permits processes to do.

The MAC framework includes a number of sample modules implementing well-known security systems, such as the Biba integrity model and Multi-Level Security (MLS), which I'll just mention here since to do them justice would require many pages. The TrustedBSD project also provides a port of the SELinux [8] policy system. All the MAC modules that are shipped with FreeBSD are documented both in manual pages (man 4 mac) and in the FreeBSD handbook [9].

Along with these well-known modules are included a number of quirkier offerings. I'll mention three of these here: seeotheruids, bsdextended, and portacl.

Note, while some of these modules can be loaded at any time, they all require that the MAC framework be compiled into your kernel by adding options MAC to your kernel config file. The more complex modules must either be loaded at boot time or compiled into your kernel.

SEEOTHERUIDS MAC MODULE

The seeotheruids module prevents users from seeing processes owned by other users. Usually ps, netstat, /proc and top will display the processes and sockets on the system belonging to all users. When the seeotheruids module is enabled, normal users can only see their own processes. Root is not a normal user, so it can see everyone's processes. It is also possible to allow a particular group of users to see the processes of others; for example, we can arrange for users in group wheel (with GID 0) to be able to see everyone's processes:

```
root@hostname# kldload mac_seeotheruids
root@hostname# sysctl security.mac.seeotheruids.enabled=1
security.mac.seeotheruids.enabled: 0 -> 1
root@hostname# sysctl security.mac.seeotheruids.specificgid=0
security.mac.seeotheruids.specificgid: 0 -> 0
root@hostname# sysctl security.mac.seeotheruids.specificgid_enabled=1
security.mac.seeotheruids.specificgid_enabled: 0 -> 1
```

BSDEXTENDED MAC MODULE

The bsdextended MAC module allows you to define more complex relationships between users and what files they are permitted to access. Unlike file permissions, ACLs, and flags, these rules are not attached to particular files but are systemwide.

This is perhaps best explained using another example. Suppose we have a sandbox user "pproxy" whose sole purpose is to run a POP proxy daemon. This user probably only needs read access to a few files on the system but, in fact, has access to all the files that are readable by "others."

```
pproxy@hostname% ./ls -l
total 8068
-r-xr-xr-x  1 pproxy  wheel  4096352 Apr 30 12:07 cat
-r-xr-xr-x  1 pproxy  wheel  4096352 Apr 30 12:07 ls
-rw-r—r—  1 pproxy  wheel        6 Apr 30 12:27 myfile
-rw-r—r—  1 root    wheel        4 Apr 30 12:27 otherfile
pproxy@hostname% ./cat myfile
hello
pproxy@hostname% ./cat otherfile
bye
```

By loading the bsdextended module we can use the ugidfw command to define rules stating which files the pproxy user can access. The following commands set up rules that allow the pproxy user read and execute access to a file and its attributes if it belongs to user pproxy and no access to any other files:

```
root@hostname# kldload mac_bsdextended
root@hostname# ugidfw add subject uid pproxy object uid pproxy mode srx
root@hostname# ugidfw add subject uid pproxy object not uid pproxy mode n
```

The "subject" part of a ugidfw command describes the user or group that the rule applies to. The "object" part describes the files the rule applies to, by specifying their owner (either by UID or GID). The mode describes the permitted operations.

r	normal read access to the file
w	normal write access to the file
x	execute/search access
s	read access to file attributes, such as permissions, owner, etc.

a administrative operations such as chmod, etc.
n no access

The first matching rule is used to decide what access is permitted by mac_bsdextended. Remember that for an action to be permitted it must also be allowed by traditional permissions and any other MAC modules, ACLs, or file flags in force.

After creating these rules with ugidfw, the pproxy user has greatly reduced access to the system. They can no longer read world-readable files (even /etc/passwd and /etc/group are inaccessible), and they cannot write to any files:

```
pproxy@hostname% ./ls -l
ls: otherfile: Permission denied
total 8066
-r-xr-xr-x  1 3007  0  4096352 Apr 30 12:07 cat
-r-xr-xr-x  1 3007  0  4096352 Apr 30 12:07 ls
-rw-r—r—  1 3007  0        6 Apr 30 12:27 myfile
pproxy@hostname% ./cat myfile
hello
pproxy@hostname% ./cat otherfile
cat: otherfile: Permission denied
pproxy@hostname% echo > myfile
myfile: Permission denied
```

Those who read the preceding two examples carefully will have noticed that I used copies of ls and cat belonging to the pproxy user. In fact, I also had to statically link these commands, as the pproxy user cannot read the normal copies of cat, ls, or libc because of the ugidfw rules! This is much the same situation as setting up a chrooted environment, where the correct executables and libraries need to be available to the user.

PORTACL MAC MODULE

The portacl module provides more flexible control of who can use which network ports. The traditional UNIX-style rules controlling who can listen for data on a port are:

- Root can listen anywhere.
- Everyone else can listen on ports > 1023.

Thus certain daemons, such as Web servers and news servers, need to at least begin life running as root in order to get access to the required ports (port 80 and port 119, respectively).

The portacl module allows rules like "user www can bind to port 80," which means that the Web server never needs to run as root. Let's take that as an example:

```
root@www# kldload mac_portacl
root@www# sysctl security.mac.portacl.rules=uid:80:tcp:80,uid:80:tcp:443
security.mac.portacl.rules:  -> uid:80:tcp:80,uid:80:tcp:443
```

Here we're saying that UID 80 (i.e., user www) should be permitted to bind to tcp port 80 and tcp port 443. We do not need to make any other change. Since the constraints enforced by the MAC framework are in addition to the normal constraints enforced by the kernel, we need to tell the kernel to relax its usual restrictions and let portacl do the work.

```
root@www# sysctl net.inet.ip.portrange.reservedlow=0
net.inet.ip.portrange.reservedlow: 0 -> 0
root@www# sysctl net.inet.ip.portrange.reservedhigh=0
net.inet.ip.portrange.reservedhigh: 1023 -> 0
```

Portacl actually has an implicit rule that limits ports 1–1023 to root, unless otherwise permitted by your setting of security.mac.portacl.rules, so we don't need any further rules to have the other ports behave as usual.

We can then start Apache as user www—provided user www can write to the necessary log files. In order to do this you could move the log files to /var/log/www and have that directory owned by user www. The necessary changes to the default Apache conf file look like this:

```
Listen 0.0.0.0:80
LockFile /var/log/www/accept.lock
PidFile /var/log/www/httpd.pid
ErrorLog /var/log/www/httpd-error.log
CustomLog /var/log/www/httpd-access.log combined
```

In some cases, there will be no downside to starting a daemon as a non-root user. However, there are some minor downsides to starting Apache as user www. As the logs are owned by user www, a vulnerability in a CGI or PHP script may give write/truncate access to log files. Of course, file flags could be used to mitigate this.

Unfortunately, there is no equivalent of the `net.inet.ip.portrange.reserved*` sysctls for IPv6, which means that the mac_portacl module is of less use in combination with IPv6. This is why the example uses the wildcard IPv4 address explicitly: to stop Apache using the IPv6 wildcard address. This omission should be fixed in a future release of FreeBSD.

GEOM and Disk Encryption

The last feature of FreeBSD that I'll mention is GEOM. GEOM is a framework for dealing with disk-like objects in the FreeBSD kernel. For example, the physical disks on the system are registered with GEOM. GEOM has classes that understand PC partition tables and FreeBSD disk labels. These classes can examine the disk and make the partitions and subpartitions of the disk available in /dev.

GEOM isn't restricted to just recognizing subpartitions; it can also perform more complex transformations such as striping, network-based disks, and encrypted disks.

A sample module for doing disk-based encryption, called BDE, is included with GEOM. Using BDE is actually quite straightforward. Naturally, you need a spare partition to house the encrypted disk—in this case, we use /dev/ad0s1g. First, we initialize the disk and choose a passphrase—as usual, choosing a good passphrase is essential:

```
root@hostname# gbde init /dev/ad0s1g -L /etc/ad0s1g.lock
Enter new passphrase:
Reenter new passphrase:
```

The lock file specified in the command will have some information about the encrypted disk's "lock sector" written into it. This file should be treated with some care—it needs to be kept backed up, and knowing its contents will make an attacker's life easier. Next, we can attach the disk, create the file system, and check that everything works OK:

```
root@hostname# gbde attach ad0s1g -l /etc/ad0s1g.lock
Enter passphrase:
root@hostname# newfs /dev/ad0s1g.bde
root@hostname# mount /dev/ad0s1g.bde /stuff
root@hostname# df /stuff
Filesystem       1K-blocks Used    Avail Capacity  Mounted on
/dev/ad0s1g.bde  60667770    4 55814346     0%    /stuff
```

Now the file system can be used as usual, but BDE will encrypt each block before it is written to the disk.

The problem with encrypted disks is getting the passphrase to the system when it wants to use the disk. Naturally, you can't store the passphrase on an unencrypted disk; the encryption would be pointless! One option is to manually issue the `gbde attach` command whenever the administrator wants access to use the encrypted disk.

Another option is to require the administrator to enter the passphrase at boot time, when filesystems are mounted. This can be done by creating an appropriate entry in /etc/fstab and /etc/rc.conf:

```
root@hostname% fgrep /stuff /etc/fstab
/dev/ad0s1g.bde       /stuff       ufs    rw          2     2
root@hostname% fgrep gbde /etc/rc.conf
gbde_devices="AUTO"
```

With this configuration, the administrator will be prompted for the passphrase for /dev/ad0s1g at boot time. The boot scripts also support encryption of the swap partition. In this case, the key can be chosen randomly, as the contents of a swap partition aren't required after a reboot:

```
root@hostname% fgrep swap /etc/fstab
/dev/ad0s1b.bde       none         swap   sw          0     0
root@hostname% fgrep gbde /etc/rc.conf
gbde_swap_enable="YES"
```

More details about how to operate the GEOM BDE system, including how to detach and destroy encrypted disks, can be read in the gbde manual page. For a description of BDE internals, see [10]; a discussion of the strengths and weaknesses of its designs can be found at [11]. The GEOM system should also make it easier for FreeBSD to support disk encryption schemes used by other systems, such as NetBSD's CGD [12].

Summary

In this article we've covered some older features (file flags and secure levels) and some newer ones (the MAC and GEOM frameworks). These features basically provide a richer set of choices in the design of a secure system, providing options that aren't available with the plain UNIX security model. As usual, the tricky bit is the care required to use these features correctly.

More features are in the pipeline. In particular, there should be support for event auditing and OpenBSM available in the FreeBSD 6 family of releases. It will also be interesting to see what interesting applications of the MAC and GEOM frameworks people can come up with. Companies and individuals are already developing third-party modules for use in their own environments or in FreeBSD-based products.

REFERENCES

[1] User Mode Linux: http://user-mode-linux.sourceforge.net/.

[2] Xen virtual machines: http://www.cl.cam.ac.uk/Research/SRG/netos/xen/.

[3] The original jail paper is available at http://docs.freebsd.org/44doc/. Many tutorials are also available; see, e.g., http://www.freebsddiary.org/jail.php.

[4] FreeBSD Handbook, "Security," File System Access Control Lists section, http://www.freebsd.org/handbook/; see a tutorial article at http://ezine.daemonnews.org/200310/acl.html.

[5] FreeBSD also provides the sysctls security.bsd.hardlink_check_uid and security.bsd.hardlink_check_gid, which prevent users from making hard links unless their UID/GID matches that of the file. However, these sysctls are only enforced against locally running processes, and so hard links can still be made over NFS.

[6] "Musings," *;login:*, April 2005.

[7] TrustedBSD project: http://www.trustedbsd.org/.

[8] SELinux: http://www.nsa.gov/selinux/.
SEBSD port: http://www.trustedbsd.org/sebsd.html.

[9] FreeBSD Handbook, "Mandatory Access Control," http://www.freebsd.org/handbook/.

[10] gbde—GEOM-Based Disk Encryption: http://phk.freebsd.dk/pubs/bsdcon-03.gbde.paper.pdf.

[11] GBDE discussion on the Cryptography Mailing List threads:
http://www.mail-archive.com/cryptography@metzdowd.com/msg03636.html;
http://www.mail-archive.com/cryptography@metzdowd.com/msg03671.html.

[12] The CryptoGraphic Disk Driver: http://www.imrryr.org/~elric/cgd/.

SRIKANTH KANDULA

surviving DDoS attacks

Srikanth Kandula is a graduate student at the MIT Computer Science and Artificial Intelligence Laboratory in the Networks and Mobile Systems group. His research interests are in networked systems and security.

kandula@MIT.EDU

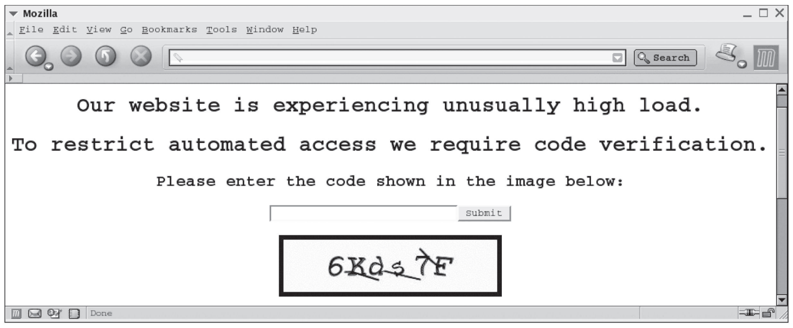
This article is related to research published as "Botz-4-Sale: Surviving Organized DDoS Attacks That Mimic Flash Crowds," *Proceedings of the 2nd Symposium on Networked Systems Design and Implementation (NSDI '05)* (Berkeley: USENIX Association, 2005).

CONSIDER THE FOLLOWING SCENARIO: Alyssa Hacker subverts tens of thousands of machines by using a worm and then uses these zombies to mount a distributed denial of service attack on a Web server. Alyssa's zombies do not launch a SYN flood or issue dummy packets that will only congest the Web server's access link. Instead, the zombies fetch files or query search engine databases at the Web server. From the Web server's perspective, these zombie requests look exactly like legitimate requests, so the server ends up spending a lot of its time serving the zombies, causing legitimate users to be denied service.

Such an attack, which we call CyberSlam, is disconcertingly real. In a recent FBI case, a Massachusetts businessman hired professionals to DDoS his competitor's Web site [1]. Like any other online business, the competitor had a search engine back end. So the professionals used a large botnet to flood the competitor's site with a massive number of queries, bringing it down for almost a week. Several extortion attempts at online gaming and gambling sites used similar attacks [2].

Why CyberSlam? If you think about it, there are some real reasons why CyberSlam attacks happen. First, we know that many large botnets exist. Zombie machines are typically compromised by worms, viruses, or malware, and the zombies are controlled by remote botnet controllers over IRC channels [3, 4, 5]. Second, there is a great incentive to mimic the browsing patterns of legitimate users. It avoids detection by standard filters and intrusion detection boxes that routinely identify and block anomalous traffic. This is especially important for organized DDoS mafia, because for them the botnet is a *reusable* resource that they would like to protect. Finally, in CyberSlam an attacker is doing little while the server does a lot. By sending a single HTTP packet containing a small request, the attacker can make the server reserve sockets, TCP buffers, and an application process, and do significant database processing or congest some other server bottleneck.

So how can a system administrator deal with CyberSlam attacks? Let us look at some existing techniques. First, how about using passwords for authentication? Passwords don't exist for most Web sites (e.g., Google), because they are cumbersome to manage both for the site and for the customers. More im-



Our website is experiencing unusually high load.

To restrict automated access we require code verification.

Please enter the code shown in the image below:

6Kds 7F

FIGURE 1: GRAPHICAL PUZZLE

portant, to check a password the server has to establish a TCP connection with the client, reserve a socket and a server worker process, and search the password database, so an attacker can simply DDoS the password-checking mechanism. Second, computational puzzles make the client do some heavy computation before giving them service. But computation is typically abundant in a botnet, and solving a puzzle for every request slows down all the normal users. So, we need a new approach to counter CyberSlam attacks.

Here is a potential solution. Intuitively, online businesses care more about serving human users; so if the server can quickly identify the human users, it can serve their requests selectively and drop all the others. There are easy ways of distinguishing humans from zombies (e.g., the graphical puzzles used by Yahoo and Hotmail). Humans can solve these puzzles easily; zombies cannot do so at all. So when the server is overloaded, it sends a graphical puzzle, as shown in Fig. 1, to everybody and serves only those who answer correctly.

Challenges

Are we done, then? Unfortunately, the answer is no. There are three main challenges with using graphical puzzles. First, in a typical setup, sending the graphical puzzle allows an unauthenticated client to establish a TCP connection, hog sockets, TCP buffers, and application processes, and force context-switches from the kernel network stack up into application space, so attackers can easily DDoS this authentication mechanism. Second, graphical puzzles have a bias against users who cannot (disabled users) or will not (due to inconvenience) solve the puzzle. By forcing a server to use graphical puzzles, the attacker has already won—she denies access to all such users. Third, a server has to divide its resources between authenticating new users and serving the users it has already authenticated. This is a tricky problem. If the server authenticates every new user, it may run out of resources to serve users who are already authenticated, leading to unnecessary starvation. If, on the other hand, the server authenticates too few new arrivals, then it may not have enough users to serve and may go idle.

Introducing Kill-Bots

We present Kill-Bots, a simple, cheap, and effective software modification to the server's operating system that distinguishes friend from foe. The core principle is simple: do not allow clients to reserve any server resource until they are authenticated. Kill-Bots kicks in whenever a Web site is in danger of being overwhelmed by requests. The software asks requestors to solve a simple graphical puzzle before granting access to server resources such as buffer space. Once a client solves the graphical puzzle, she is given an HTTP cookie so that she can obtain service for some time without having to solve another puzzle. Addresses that repeatedly request access to the server without solving the puzzle are blacklisted automatically. When the load on the Web server decreases, it stops issuing puzzles and accepts requests from non-blacklisted addresses, so even real users who did not solve the puzzle gain access. Finally, Kill-Bots efficiently divides server resources by adapting the probability with which new users are authenticated. A Kill-Bots server neither accepts more users than it can serve nor goes idle by not authenticating enough new users. Fig. 2 shows how these individual pieces fit together in Kill-Bots, and Fig. 3 shows the HTML for the puzzle.

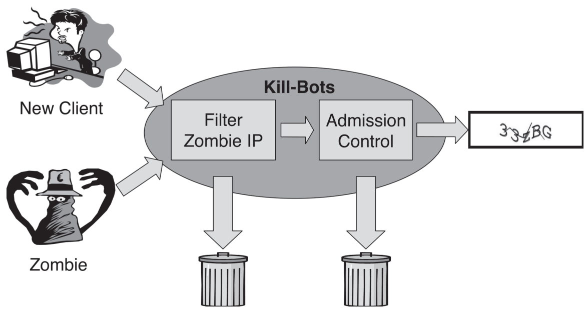


FIGURE 2: HANDLING ZOMBIES WITH KILL-BOTS

```
<html>
   <form method = "GET" action="/validate">
        <img src = "PUZZLE.gif">
        <input type = "password" name = "ANSWER">
        <input type = "hidden" name = "TOKEN" value = "[]">
   </form>
</html>
```

FIGURE 3: HTML SOURCE FOR THE PUZZLE

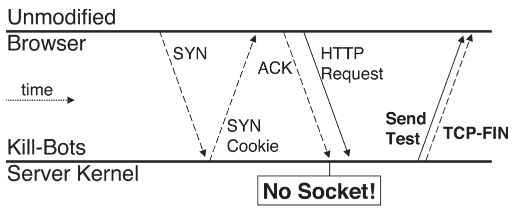


FIGURE 4: KILL-BOTS MODIFIES SERVER'S TCP STACK TO SEND TESTS TO NEW CLIENTS WITHOUT ALLOCATING A SOCKET OR OTHER CONNECTION RESOURCES

Let us briefly look at each of the main components of Kill-Bots. The modified network stack is shown in Fig. 4. When, the client sends a TCP SYN, Kill-Bots responds with a SYN cookie. The SYN cookie is a standard defense mechanism that serves two purposes. First, it filters requests from clients with spoofed IP addresses. Second, it allows the server to continue the TCP handshake without maintaining any state for the half-open connection. Upon receiving the SYN cookie, the client responds with an acknowledgment completing the TCP handshake. At this point, the standard network stack allocates a socket, reserves TCP buffers, and passes the request onto an application-space process. All of this is done for every attacker request and is quite costly. More important, these resources remain allocated until the client responds with a FIN. An attacker can simply hog resources by not sending the FIN. To avoid this, Kill-Bots ignores the acknowledgment. Instead, it looks at the first data packet that contains the same acknowledgment number and peeps into the HTTP header to confirm that this is a previously authenticated client with a valid HTTP cookie. If not, the modified kernel immediately sends the graphical puzzle and a TCP FIN without creating a socket or reserving any resources, as shown in Fig. 4.

Recall the second challenge: Graphical puzzles have a bias against users who cannot or will not answer the puzzle. So whenever an attacker forces the server to use graphical puzzles, these users are denied access. Kill-Bots uses the following observation of client behavior: a human user who does not answer the graphical puzzle will only retry a couple of times to see if he can access the server without answering; attackers, on the other hand, will have to continuously bombard the server with requests and fetch the graphical puzzles or the attack goes away. There is a simple difference between the human users and attackers now; the attackers have many more *unanswered* puzzles than the normal users. Kill-Bots uses a bloom filter to track the number of unanswered puzzles per IP and drops all requests from an IP if its associated bloom counters cross a limit, say, 32.

Recall the third challenge of dividing resources between serving authenticated users and authenticating new users. Kill-Bots deals with this by probabilistically authenticating new users and dropping others. The authentication probability adapts to server conditions. Intuitively, whenever the server is lightly loaded, the authentication probability increases so that Kill-Bots authenticates more new users, and when the server is heavily loaded it decreases. I will defer the specific details of the adaptation, but the tricky parts are *how much can one increase the authentication probability without overloading the server* and *how long it takes to adapt to changing conditions*. Note that small increments would reduce the chance of overshooting, while large increments would react to changing conditions and get you to the optimal operating condition quicker. We reconcile these contradictory preferences by using techniques from control theory.

In experiments, a Kill-Bots-protected Web server successfully endured five times as many hits as an unprotected Web server could tolerate. Not only did the Web server stay online, but protected Web sites also maintained speedy response times, even during the height of the attacks. You might wonder what would happen during a flash crowd, i.e., when the server overload is caused by a large number of legitimate requests. Kill-Bots improves both server hit-rate and response times by using the adaptive authentication probability mechanism to quickly drop new users who cannot be served. This ensures that server resources are not wasted on requests that are going to be dropped at a later time.

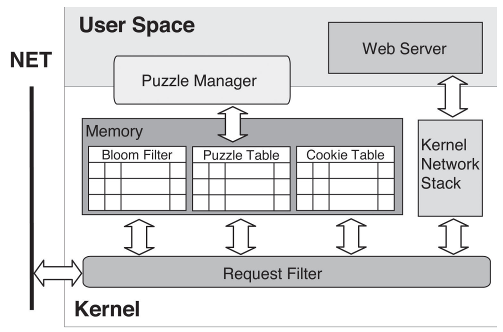


FIGURE 5: A MODULAR REPRESENTATION OF THE KILL-BOTS CODE

Using Kill-Bots

From a practical standpoint, here is how you could use Kill-Bots to protect your own Web server. Fig. 5 is a modular representation of the Web server using Kill-Bots. Kill-Bots doesn't require an extra server; it is a software patch to the operating system of the server kernel. Kill-Bots needs a store of graphical puzzles. Generating the graphical puzzles is relatively easy, for example using the JCAPTCHA software [6], and can done on the server itself during periods of relative inactivity or on a different dedicated machine. Also, puzzles may be purchased from a trusted third party. Kill-Bots has modest overhead; it uses 10MB of RAM to cache graphical puzzles, maintain the bloom filter that blacklists zombie

IPs, and maintain state for each cookie. A kernel thread periodically loads fresh graphical puzzles into memory. The per-request overhead is also quite small. On a 2.4GHz PIV workstation, peeping into HTTP requests costs 8 microseconds of server time, and serving a puzzle costs 31 microseconds.

Why Does Kill-Bots Matter?

Worries over distributed denial-of-service attacks are spreading. It is depressing, yet true, that the future will see many more organized DDoS attacks. Most Web server defenses use authentication procedures that are easily outwitted and require huge excesses in the form of replicated content, multiple CPUs, fancy hardware, and extra bandwidth. Kill-Bots is much cheaper and can be deployed easily; it requires no changes in users' Web browsers and works with the very large number of Web servers running Linux. Although Kill-Bots occasionally misclassifies legitimate users as zombies, it allows Web sites under attack to remain available and so keeps the Web open for business, while barring the way to thieves and vandals.

REFERENCES

[1] K. Poulsen, "FBI Busts Alleged DDoS Mafia," 2004: http://www.securityfocus.com/news/9411.

[2] J. Leyden, "East European Gangs in Online Protection Racket," 2003: http://www.theregister.co.uk/2003/11/12/east_european_gangs_in_online/.

[3] J. Leyden, "The Illicit trade in Compromised PCs," 2004: http://www.theregister.co.uk/2004/04/30/spam_biz/.

[4] E. Hellweg, "When Bot Nets Attack," *MIT Technology Review,* September 2004.

[5] L. Taylor, "Botnets and Botherds": http://sfbay-infragard.org.

[6] JCAPTCHA: http://jcaptcha.sourceforge.net/.

GEOFFREY MAINLAND AND
MATT WELSH

distributed, adaptive resource allocation for sensor networks

Geoffrey Mainland is currently a Ph.D. student at Harvard University and received his A.B. in Physics from Harvard College. His research interests include programming languages, distributed systems, networks, and intelligent control.

mainland@eecs.harvard.edu

Matt Welsh is an assistant professor of Computer science at Harvard University. His research interests encompass sensor networks, operating systems, and distributed systems.

mdw@eecs.harvard.edu

SENSOR NETWORKS HAVE THE POTENTial to revolutionize any number of fields. In the future, sensor networks may allow bridges to immediately report on structural damage following an earthquake, environmental monitoring systems to track pollutants in real time, and emergency response workers to direct limited resources to those who need medical attention most urgently. They will also vastly increase the quantity and quality of research data scientists can collect in the field. Our group at Harvard is working on several projects that apply sensor network technology to problems not traditionally associated with computer science, including CodeBlue, a software and hardware platform for emergency response, and the Volcán Tungurahua project, where we are using sensor networks to monitor eruptions at an active volcano in central Ecuador.

A typical sensor network device is the UC Berkeley Mica2 node, which consists of a 7.3MHz ATmega128L processor, 128KB of code memory, 4KB of data memory, and a Chipcon CC1000 radio capable of 38.4Kbps and an outdoor transmission range of approximately 300 meters. The node measures 5.7cm by 3.1cm by 1.8cm and is typically powered by two AA batteries, with an expected lifetime of days to months, depending on application duty cycle. With such limited computational and communication resources, how can effective applications be developed using this class of device? One might be tempted to assume that in a few years sensor network devices will have more powerful CPUs and better radios. Undoubtedly more powerful devices will appear, but we believe that the majority of sensor network nodes will be similar to the Mica2 in terms of processing power and bandwidth—they'll just be much smaller and cheaper. Far from becoming obsolete, techniques for programming these small devices will become increasingly important.

Consider two commonly cited applications for sensor networks: environmental monitoring [1, 2] and distributed vehicle tracking [3, 4]. Both applications require nodes to collect local sensor data and relay it to a central base station, typically using a multi-hop routing scheme. To reduce bandwidth requirements, nodes may need to aggregate their local sensor data with that of other nodes. Even these simple applications present unique challenges to system implementers. Nodes must individually determine a schedule for sampling, aggregating, and sending data, subject to energy budget constraints. This schedule affects energy usage and, therefore, the overall lifetime of the network, as well as the quality of the data generated by the network. A node's ideal schedule is based on its physical location, position in the routing topology, and changes in the environment. As a result, there is almost never an ideal a priori common schedule for all nodes. Any action-scheduling algorithm must cope with network dynamics, so even full knowledge of a node's network location and capabilities doesn't allow one to choose a schedule ahead of time that will work well for all settings.

Many current applications use a single fixed, common schedule anyway, or try to build in some ad hoc adaptive behavior. For example, an application might selectively activate nodes that are expected to be near some phenomenon of interest. The only current tool that programmers have to address the scheduling issue is manual tuning, which is difficult and error-prone.

We propose an adaptive resource allocation scheme for sensor networks, called "Self-Organizing Resource Allocation" (SORA). Rather than defining a fixed node schedule, SORA causes nodes to individually tune their rate of operation using techniques from reinforcement learning [5]. Nodes receive rewards for taking "useful" actions that contribute to the overall network goal, such as listening for incoming radio messages or taking sensor readings. Each node learns which actions are profitable based on this reward feedback. Network retasking is accomplished by adjusting rewards, rather than pushing new code to sensor nodes, and network lifetime is controlled by constraining nodes to take actions that meet a local energy budget.

Application Example: Vehicle Tracking

As a concrete example of using SORA to manage resource allocation in a realistic sensor network application, we consider tracking a moving vehicle through a field of sensors. Vehicle tracking raises a number of interesting problems in terms of detection accuracy and latency, in-network aggregation, energy management, routing, node specialization, and adaptivity [4, 6, 7]. Vehicle tracking can be seen as a special case of the more general data collection problem also found in applications such as environmental and structural monitoring [2, 8].

In the tracking application, each sensor is equipped with a magnetometer capable of detecting local changes in a magnetic field, which indicates the proximity of the vehicle to the sensor node. One node acts as a fixed base station, which collects readings from the other sensor nodes and computes the approximate location of the vehicle based on the data it receives. The systemwide goal is to track the location of the moving vehicle as accurately as possible while simultaneously maximizing the efficiency of the network's energy use.

Each sensor node can take the following set of actions: *sample* a local sensor reading, *send* data toward the base station, *listen* for incoming radio messages, *sleep* for some interval, and *aggregate* multiple sensor readings into a single value. Each node maintains a fixed-length FIFO buffer of sensor readings, which may be sampled locally or received as a radio message from another node. Each entry in the buffer consists of a tuple containing a vehicle location estimate weighted by a magnetometer reading. The sample action appends a local reading to the buffer, and the listen action may add an entry if the node receives a message from another node during the listen interval.

Each action a has a utility $u(a)$ given by:

$$u(a) = \begin{cases} \beta_a r_a & \text{if the action is } \textit{available} \\ 0 & \text{otherwise} \end{cases}$$

where r_a is the current reward for action a, and β_a is the *estimated probability of payment* for that action, which is learned by nodes as described below. In our work, reward vectors are broadcast by the base station, but they may also be static parameters of the sensor network program. An action may be *unavailable* if either the current energy budget is too low to take the action, or other dependencies have not been met (such as lack of sensor readings to aggregate). This utility function is just the expected reward for taking a given action.

The estimated probability of receiving a reward for an action a, β_a is updated every time that action is taken using an exponentially weighted moving average (EWMA). The equation for this update is:

$$\beta'_a = \begin{cases} \alpha + (1-\alpha)\beta_a & \text{if } a \text{ receives a reward} \\ (1-\alpha)\beta_a & \text{otherwise} \end{cases}$$

where α represents the sensitivity of the EWMA filter. If an action does not produce a reward, the node's estimated probability of receiving a reward decreases, but if the action does produce a reward, the estimated probability increases. Nodes learn the probability of receiving a reward instead of directly learning the expected reward for an action because this allows them to more easily adapt when a new reward vector is injected into the network.

A node chooses an action to perform by examining the utilities of all available actions and picking the action with the largest utility, which is just the expected reward. The expected reward for an action will vary over time due to possible reward adjustments and changing environmental conditions. Therefore, it is important that nodes periodically "take risks" by choosing actions that have a low reward probability β_a. To allow nodes to occasionally explore the action space, we employ an ε-greedy action selection policy. With a small probability, ε, when faced with a decision, a node will choose an available action uniformly at random. With probability 1-ε the node selects the "greedy" action, that is, the action that maximizes the utility $u(a)$. This exploration prevents a node from ever again selecting an action that has been unprofitable in the past.

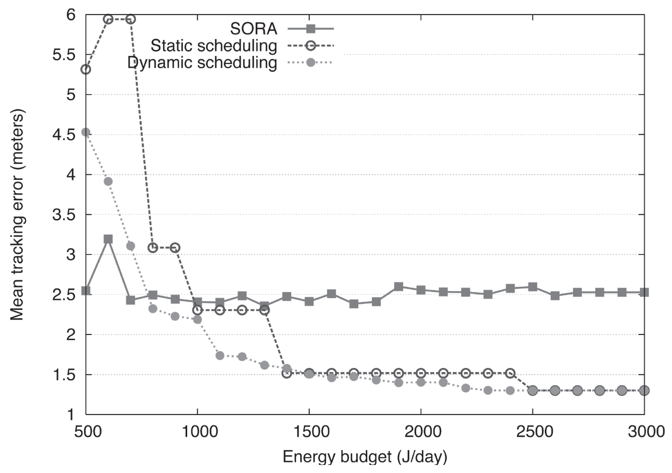
Comparison with Existing Approaches

To compare the use of SORA with more traditional approaches to sensor network scheduling, we implemented three additional versions of the tracking system. The first employs static *scheduling,* in which every node uses a fixed schedule for sampling, aggregating, and transmitting data to the base station. Nodes perform a round of actions: sample, listen, aggregate, and transmit. The node then sleeps for a period of time. Given a daily energy budget, a node determines how long it must sleep between these rounds to meet this energy budget. The same schedule is used for every node in the network, so nodes do not learn which actions they should perform, nor do they adapt their sampling rate to environmental changes such as the approach of the vehicle. This is the typical approach used by current sensor network applications.

The second approach employs *dynamic scheduling,* in which nodes continuously adjust their sleep period based on their current remaining energy. This allows nodes that do not consume energy aggregating or transmitting data to use this conserved energy to increase their sampling rate.

The third and final approach, the *Hoods tracker,* is based on the tracking system implemented using the Hoods communication model [7]. It is largely similar to the dynamically scheduled tracker except in the way that nodes calculate the target location. Each node that detects the vehicle broadcasts its sensor reading to its neighbors. The node then listens for some period of time and, if its own reading is the maximum of those it has heard, computes the centroid of the readings (based on the known locations of neighboring nodes) as the estimated target location. This location estimate is then routed toward the base station. We implemented the Hoods tracker to emulate the behavior of a previously published tracking system for direct comparison with the SORA approach.

For purposes of comparison, we are interested in two metrics: tracking accuracy and energy efficiency. We do not expect SORA to be more accurate than the other scheduling approaches, but if it is to be a realistic solution it should perform similarly. Our goal is to give good performance while maximizing energy efficiency, so we are willing to sacrifice some accuracy for more efficient energy usage.



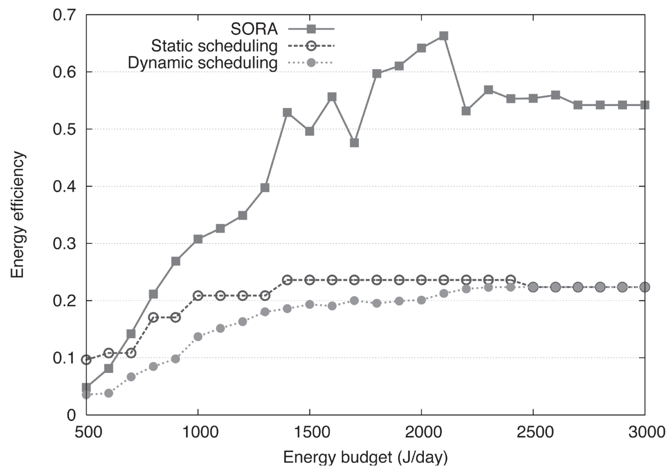


FIGURE 1: TRACKING ACCURACY AND ENERGY EFFICIENCY

Figure 1 summarizes the accuracy and efficiency of each scheduling technique as the energy budget is varied. Each system varies in terms of its overall tracking accuracy as well as in the amount of energy used. While SORA has a somewhat higher rate of tracking error compared to the other scheduling techniques, it demonstrates the highest efficiency, exceeding 66% for a daily energy budget of 2100J. The static and dynamic schedulers achieve an efficiency of only 22%. In SORA, most nodes use far less energy than the budget allows. The ability of SORA to "learn" the duty cycle on a per-node basis is a significant advantage for increasing network lifetimes. The Hood tracker performs poorly due to its different algorithm for collecting and aggregating sensor data, so it is not included in Figure 1.

Conclusions

Current approaches to resource management are often extremely low-level, requiring that the operation of individual sensor nodes be specified manually. With SORA, nodes self-schedule their local actions in response to feedback. This allows nodes to *automatically* adapt to changing conditions and specialize their behavior in response to physical location, routing topology, and environmental changes. While it does require a reformulation of application logic, using SORA is not particularly difficult, and it allows sensor networks to utilize resources much more efficiently than standard, ad hoc manual techniques. The increased complexity of future sensor network applications may make them a bad fit for this simple technique, but SORA is not meant to be universal. We view it not as a final solution, but as an important step that demonstrates the potential power of adaptive techniques in real sensor network systems.

REFERENCES

[1] Alberto Cerpa, Jeremy Elson, Deborah Estrin, Lewis Girod, Michael Hamilton, and Jerry Zhao, "Habitat Monitoring: Application Driver for Wireless Communications Technology," *Proceedings of the Workshop on Data Communications in Latin America and the Caribbean*, 2001.

[2] Alan Mainwaring, Joseph Polastre, Robert Szewczyk, David Culler, and John Anderson, "Wireless Sensor Networks for Habitat Monitoring," *ACM International Workshop on Wireless Sensor Networks and Applications (WSNA'02),* Atlanta, GA, USA, Sept. 28, 2002.

[3] Dan Li, Kerry Wong, Yu Hen Hu, and Akbar Sayeed,"Detection, Classification and Tracking of Targets in Distributed Sensor Networks," *Journal of the IEEE Signal Processing Magazine,* Vol. 19, No. 2, March 2002.

[4] Yingqi Xu and Wang-Chien Lee, "On Localized Prediction for Power Efficient Object Tracking in Sensor Networks," *Proceedings of the 1st International Workshop on Mobile Distributed Computing,* May 2003.

[5] Richard S. Sutton and Andrew G. Barto, *Reinforcement Learning: An Introduction*, MIT Press, Cambridge, MA, 1998.

[6] R. Brooks, P. Ramanathan, and A. Sayeed, "Distributed Target Classification and Tracking in Sensor Networks," *Proceedings of the IEEE,* November 2003.

[7] Kamin Whitehouse, Cory Sharp, Eric Brewer, and David Culler, "Hood: A Neighborhood Abstraction for Sensor Networks," *Proceedings of the International Conference on Mobile Systems, Applications, and Services (MOBISYS '04),* Boston, MA, June 2004.

[8] Venkata A. Kottapalli, Anne S. Kiremidjian, Jerome P. Lynch, Ed Carryer, Thomas W. Kenny, Kincho H. Law, and Ying Lei, "Two-Tiered Wireless Sensor Network Architecture for Structural Health Monitoring," *Proceedings of the SPIE 10th Annual International Symposium on Smart Structures and Materials,* San Diego, CA, March 2000.

book reviews

ELIZABETH ZWICKY

zwicky@greatcircle.com

ADAM TUROFF

ziggy@panix.com

SILENCE ON THE WIRE: A FIELD GUIDE TO PASSIVE RECONNAISSANCE AND INDIRECT ATTACKS

Michal Zalewski

No Starch Press, 2005, 1-59327-046-1, 281 pp.

Reviewed by Elizabeth Zwicky

Zalewski's book is an interesting tour through some security-related stuff. It is not, as the subtitle implies, an overview of passive attacks. It's more of a country ramble than the field guide you might use on such a walk. It wanders through security, and computing as a whole, with some novel information, some basic information you may not have come across before, and some information you already knew and didn't much care about. I, for instance, found the description of how modems actually work irrelevant but amusing, but yawned my way through the state tables that accompany the recapitulation of the entire history of computing, starting with logic itself and ending with modern processor design. Then again, I was fascinated by the section on random number generators; it's a frequently discussed topic in security, but it's still rare to see a lucid discussion of how they work and don't work, and why it matters. Plus, the pictures are very convincing.

If you are looking for practical information of immediate use about passive and indirect attacks, this is not the book for you; then again, it makes no claim to be. If you are a "hacker" type in the old sense of the word, fond of taking things apart to see how they work, and you have any interest in security, you will probably find significant portions of this book intriguing. Try not to be turned off by the initial chapters, which unfortunately are the weakest.

OPTIMIZING LINUX PERFORMANCE: A HANDS-ON GUIDE TO LINUX PERFORMANCE TOOLS

Philip G. Ezolt

Prentice Hall, 2005, 0-13-148682-9, 353 pp.

PERFORMANCE TUNING FOR LINUX SERVERS

Sandra K. Johnson, Gerrit Huizenga, and Badari Pulavarty, eds.

IBM Press, Pearson plc, 2005, 0-13-144753-X, 547 pp.

Reviewed by Elizabeth Zwicky

This batch of books brought me two on Linux performance optimization, an under-served topic, and a difficult one to write about. Performance tuning is difficult, and doing it well involves three things:

1. Excellent detective skills.

2. An intimate understanding of precisely how the system you are tuning works.

3. Familiarity with applicable tools.

1 is extraordinarily difficult to teach at all, let alone in a book. 2 is an immense amount to cover, differs importantly between sub-releases of a single product, and leads into territory best covered by "Internals of" books. That leaves 3, which isn't all that interesting without 1 and 2.

Optimizing Linux Performance does a nice job of negotiating these difficult waters. It's not going to turn anybody into a master performance tuner (no book is), but it should give somebody with normal system administration skills sufficient background in the tools and techniques to start solving problems. Its descriptions of how things work are brief, but deep enough that you can understand what the tools are doing and why you care, and it covers a wide range of tools, with information about how to use their output. Most valuably, it talks about, and demonstrates, the process of performance tuning—how you actually put the tools together to get a desired result.

Performance Tuning for Linux Servers does not succeed so well. It attempts a vast and ambitious task: to cover everything you need to know about Linux internals, down to very precise levels of detail, plus performance characteristics of all the common kinds of server workloads, and case studies, with a tutorial on programming for performance thrown in. I'm prejudiced to start with against books with large numbers of authors, and this one, with 23 authors and three editors, has only strengthened my prejudices. The chapters are of uneven quality and often overlap or even contradict each other. Many of the chapters appear to have originally been written for other purposes and fail to integrate well with the rest of the book.

This book contains an enormous amount of information, but it's not in a form that is particularly usable. I like the chapter on file servers (it's entitled "File and Print Servers," but it doesn't really say much about print servers), and the final "case study" chapter is a genuine case study that ties things together and shows a realistic troubleshooting process. The detailed sections on internals may also be of interest if you are reasonably familiar with kernel internals on other systems and want a technical overview of Linux internals. Although the information in them is of use in performance tuning, they do not in general tell you why or how.

If you are experienced in performance tuning and want to know about Linux specifics, parts of this book may be of use to you. If you are not, the book is unlikely to be useful, and some chapters contain information that is misleading or downright incorrect: "A module is a kernel feature that provides the benefits of a microkernel without a penalty," for instance. Skip it, and get *Optimizing Linux Performance*.

PETER VAN DERLINDEN'S GUIDE TO LINUX

Peter van derLinden

Pearson Education, 2005, 0-13-187284-2, 624 pp.

LEARNING UNIX FOR MAC OS X TIGER

Dave Taylor

O'Reilly, 2005, 0-596-00915-1, 260 pp.

MAC OS X TIGER FOR UNIX GEEKS

Brian Jepson

O'Reilly, 2005, 0-596-00912-7, 395 pp.

Reviewed by Elizabeth Zwicky

Here's a batch of books trying to introduce particular UNIX variants to various audiences. *Peter van derLinden's Guide to Linux* is intended for people converting from Windows; *Learning UNIX for Mac OS X Tiger* is for experienced Macintosh users learning to use the UNIX command line; and the audience for *Mac OS X Tiger for UNIX Geeks* is experienced UNIX developers converting to the Macintosh.

Peter van derLinden's Guide to Linux might possibly have allowed my father to switch to Linux (after years of loathing Windows and using it anyway, my father has converted to a Macintosh, and getting him to use anything else is now a lost cause). But my father is pretty technological to start with. The hypothetical mother in sentences like "Even my mother can use this computer" is not going to use this book. It would be great for your reasonably technology-savvy relatives, the people who are actually running the latest version of Windows, and not because it was installed on the computer when they bought it, either, or the ones who successfully put together their own machine. It's not going to work for the people who think you're a crazy UNIX bigot who hates Microsoft for some stupid political reason of your own. Peter van derLinden tries to be balanced and fair, but he's got a very strong UNIX and open source accent, which I find charming but which will set off alarm bells in people who trust Microsoft and distrust the open source community. I'm not sure it's possible to sell Linux to those people, but I'm sure this book won't do it, although it does try.

The book covers Linux installation, using Linspire, and most day-to-day operations. It's strongly oriented to using free (both open source and non-commercial) tools, so it covers Lphoto and OpenOffice, but doesn't talk about emulation or about commercial software for Linux. It does have thorough coverage of dual boot options.

Learning UNIX for Mac OS X Tiger is what you offer to your geeky relatives with Macintosh systems to get them to use the command line. It is a solid introduction to UNIX basics with a Macintosh accent, including coverage of Macintosh-specific features and difficulties and comparisons with the GUI tools. It moves at a pretty fast clip, as you need to if you're going to get the basics of any UNIX into a mere 245 content pages; so once again, if you have the hypothetical mother people talk about, it's not going to make her happy about firing up a terminal. Your equally hypothetical technology-oriented niece, however, will be writing shell scripts in no time. I found one unfortunate and puzzling error (the statement in Chapter 2 that the shell waits for background processes to finish before returning its prompt—it's not even clear to me what this was intended to say, but it's definitely wrong), but single errors like that can occur in even the best of books.

Max OS X Tiger for UNIX Geeks is a nice introduction to software development on OS X Tiger. It starts with a general introduction to the features of MacOS X that are different from other operating systems, and then dives into what you need to know to build software. I found the general introduction enlightening and useful, although from a system administration point of view it has some shortcomings—I really wanted to know how inetd.conf plays with the other ways of starting programs on demand, for instance. These are minor issues, however, and the book is intended for developers, who can just use an appropriate Tiger method to start their programs. Even for a non-developer, it's a great reference to porting and packaging software for Tiger. I recommend it.

PERL BEST PRACTICES

Damian Conway

O'Reilly, 2005, 0-596-00173-8, 544 pp.

Reviewed by Adam Turoff

Conway's collection of 256 tips for writing better Perl programs is a breath of fresh air, highlighting practices that work well and idioms to be avoided at all costs. All of his advice is prefaced with an implicit zeroth-law of programming: *Always code as if the person who ends up maintaining your code will be a violent psychopath who knows where you live.*

Some advice, such as the chapters on formatting, naming, and control structures, may seem nitpicky and arbitrary. Taken together, they constitute a set of conventions that allow you to do more with less effort. Later chapters focus on more advanced topics where the practices described represent the difference between keeping your sanity and spending extreme amounts of time mired in needless debugging.

I recommend *Perl Best Practices* to anyone who deals with Perl even on a casual basis. This is the first book that simply and concisely captures the lore of what makes a good Perl program good, and how to avoid features that slowly lead to bit-rot. This book is already on my list of books to reread annually.

ADVANCED PERL PROGRAMMING, 2ND ED.

Simon Cozens

O'Reilly, 2005, 0-596-00456-7, 304 pp.

Reviewed by Adam Turoff

The first edition of *Advanced Perl Programming* covered an esoteric grab bag of topics, some useful, some arcane, and some impractical. These represented the vanguard of what "advanced Perl" meant in 1997. Since then, topics such as object-oriented Perl, references, closures, modules, and eval are now widely understood and no longer "advanced" topics.

If you wrote off this title based on experience with the first edition, look again. The second edition of *Advanced Perl Programming* is a complete rewrite, and attempts to address the shortcomings of the first edition. Topics considered advanced eight years ago are now taken as given. The focus is now on advanced *uses* of Perl, not advanced (and oblique) *features* of Perl. The book opens with a discussion of some deep, dark magic-playing with the symbol table and incorporation of useful features first found in other programming languages. From there, the book switches to a Cook's Tour of CPAN, where Cozens compares and contrasts alternative CPAN modules for doing similar tasks. The discussions are useful and informative, but never very deep.

While this book may continue to be mistitled, it is certainly a welcome addition to my Perl bookshelf. I recommend this book to programmers who are starting to delve into advanced topics such as parsing, templating, testing, event-centric programming, Unicode, or database modeling.

WORLDS '05
Second Workshop on Real, Large Distributed Systems
Sponsored by USENIX

December 13, 2005
San Francisco, CA

SAVE THE DATE!

WORLDS '05

Second Workshop on Real, Large Distributed Systems

December 13, 2005, San Francisco, CA

http://www.usenix.org/events/worlds05

Join us in San Francisco, CA, December 13, 2005, for one day of discussion of useful ideas, experience, and research directions in the area of real, large distributed systems. The Second Workshop on Real, Large Distributed Systems will bring together people who are exploring the new challenges of building widely distributed networked systems and who lean toward the "rough consensus and running code" school of systems building. WORLDS is a place to share new ideas, experiences, and work in progress, with an emphasis on systems that actually run in the wide area and the specific challenges they present for designers and researchers.

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

25 YEARS AGO

PETER H. SALUS

peter@usenix.org

It was October 1980.

;login: carried an article headed: "Microsoft Xenix Operating System." With all the to-do over who did what to whom when in UNIX (see http://www.groklaw.net), I thought I'd read it and quote from it.

"The XENIX* Operating System from Microsoft is a microprocessor adaptation of Bell Laboratories' Version 7 UNIX* operating system. With the XENIX operating system, Microsoft is bringing the power of the UNIX OS to microcomputers. The XENIX Operating System will be released in versions that run on the Zilog Z8000, Motorola M68000, Intel 8086, and the DEC PDP-11*."

That's how it began. It continued:

"The XENIX system is an interactive, multi-user, multi-tasking operating system with a flexible user interface. . . .

"Microsoft is committed to supporting the XENIX system as an environment for program development that will be identical on all the popular 16-bit microprocessors. Different CUs, as well as different hardware boards for the same CU, will be supported. . . ."

Interestingly, even 25 years ago, it "sounds" like Microsoft: "backed by Microsoft's experience and expertise in producing quality system software." "Microsoft is committed to having a version of the XENIX system for every popular 16-bit microprocessor by the end of 1981."

Go talk to "Bob Greenberg, XENIX Product Manager," should you have any questions.

Oh. Those asterisks. There's a 4-line footnote about TMs.

WORKSHOP PROCEEDINGS

The October 1985 *;login:* carried seven papers from the December 1984 Graphics Workshop, held in Monterey. In Atlanta, in June 1986, Reidar Bornholt accosted me with questions about the "proceedings" for the 1985 workshop. When I got back to El Cerrito (where the USENIX office was then situated), I asked, and a dusty manila envelope was produced. A photo-offset Proceedings was available by September. It was the first workshop so memorialized.

SAGE UPDATE

CHRIS PALMER

palmer@eecs.harvard.edu

Since the writing of the August SAGE update, many of the transitional arrangements necessary to form a new, independent SAGE have come to pass. First and foremost, we held elections for the new Board. The SAGE Board consists of Geoff Halprin, Trey Harris, Doug Hughes, Andrew Hume, Chris Palmer, David Parter, Tom Perrine, Stephen Potter, and Pat Wilson.

Then, from July 29th to the 31st, the new Board met for a strategic planning session and first Board meeting of the new SAGE. We elected our officers: Tom Perrine is now the president, Pat Wilson the vice president, and Andrew Hume is secretary/treasurer. We then spent nearly all of our time planning for the short and medium term, and doing preliminary long-term brainstorming.

In the short term, our efforts are focused on creating a stable, transparent, and responsible organization. Without a strong foundation, further efforts would be futile. The Interim Board did great work here, providing us with a corporation, bylaws, and legal structure. They also chose an association management company (AMC), and at the meeting we selected an executive director. By the time you read this, we will have likely completed arrangements with the AMC and will have introduced it and our executive director to SAGE.

We continue to work in the short term, with the support of the USENIX Board, on moving SAGE member services to our control. We hope to finish well before the LISA '05 conference. We also formed a number of strategic committees, described in the August Memo to Members, to build various parts of a stable SAGE, such as partnerships, development of sponsors, membership, communications, and leadership.

Leadership is a special focus of the new SAGE. We have formed a highly independent, non-Board Leadership Committee. The Leadership Committee will be the nominating committee for elections, but will stay active between elections to engage and encourage volunteers, helping to develop present and future leaders of SAGE. Greg Rose has graciously consented to serve as chair of the Leadership Committee.

We then went beyond our institutional efforts and examined the mission of SAGE. SAGE is dedicated to individual support and education, advancing our profession as a whole, and serving the wider public interest in system administration. That's a broad mandate, and hard to distill. We aim to provide our membership year-round with the community experience LISA conferences offer. Education, communication, and research all fall under this effort.

Now we have to decide which programs we want to revive or start, in what order, and how we deliver on them. SAGE's work up to and including LISA '05 will aim to answer these questions. We will be presenting our medium-range plans there at the Community Meeting and BoF sessions. We'll want to hear which member services SAGE members want, and how best to deliver them. Board members will be there all week and will welcome input from present or potential SAGE members. Please strongly consider attending LISA this year to see and shape the future of SAGE, as well as for the excellent technical and training programs that LISA offers.

Speaking personally, I have enjoyed serving on the SAGE Board a great deal so far. The engagement and energy—and, honestly, fun times—at the meeting were a hopeful sign of things to come. I hope you join us in building an effective organization focused solely on our profession. Please, if you wish to get involved or have feedback on the future of SAGE, don't hesitate to contact us at board@sage.org.

USACO: THE USA COMPUTING OLYMPIAD

ROB KOLSTAD

USACO Head Coach

You probably know that USENIX is a major sponsor of the USA Computing Olympiad, a 13-year-old organization that serves pre-college computer enthusiasts around the world. The Computing Olympiad has these goals:

- Provide pre-college students with opportunities to sharpen their computer programming skills to enable them to compete successfully at the international level.
- Enhance the quality of pre-college computer education by providing students and teachers with challenging problems, training materials, and competitions that emphasize algorithm development and problem-solving skills.
- Recognize excellent students with outstanding skills in computer science and encourage them to pursue the profession.

- Provide educational, motivational, and competitive materials in the form of programming competitions and Web-based training.

The 2004–05 season has been extraordinarily successful in achieving these goals. The Training Pages are approaching 30,000 logins per month by students in the 90 different countries that utilize the USACO training facilities. The mailing list has over 18,000 members; participation in our monthly contests (held during the academic year) will soon top 1,000 elite programmers per contest.

The most recent achievement was the USA Invitational Computer Olympiad held June 1–9 at Colorado College in Colorado Springs, Colorado. The 15 best U.S. competitors matched skills against the best from Bulgaria, Canada, China, and Poland. Veteran senior Alex Schwendner emerged victorious, edging out silver medalists Eric Price, Filipek Wolski from Poland, and Chuenguag Zhu from China.

Among the innovations this year was the second of six rounds: the speed competition. Unlike most programming contests, where the action cannot be compared favorably to watching paint dry, the speed competition was suspenseful. Competitors were given a problem requiring 5–10 minutes to solve. The first solution earned the most points, with declining points until half the competitors had completed it. At that point, the sub-round closed. Because contestants incur a small penalty for wrong answers, later correct answers may score more points. With almost a dozen tasks and two dozen programmers, and nonstop action, several competitors kept neck-and-neck for the lead!

The Olympiad was also used to choose the four U.S. students who will represent our country at the International Olympiad, to be held in Nowy Sacz, Poland, August 18–25. Veterans Eric Price and Alex Schwendner will be joined by John Pardon and Matt McCutcheon in vying for the gold against 300 students from over 75 other countries.

The USACO continues to strive to accomplish its mission with expanding programs and qualifying events. If you'd like to assist or if your organization would like to support the USACO, please contact Rob Kolstad at kolstad@usenix.org.

THE USENIX ASSOCIATION FINANCIAL REPORT FOR 2004

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. 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, 2004, are available on request.

FINANCIAL STATEMENT SUMMARY

USENIX continues to be a healthy organization. In 2003 and 2004, we managed to have a modest increase in our net assets, which enabled us to replenish the Reserve Fund. At the beginning of the 2004 fiscal year, USENIX produced a balanced budget. However, we incurred a slight deficit of $74K in operations, which was mainly due to lower than projected attendance at the USENIX Annual Technical Conference. This was largely offset by the fourth-quarter performance of the Reserve Fund. USENIX ended the year with an increase in net assets of $218K.

USENIX MEMBERSHIP DUES AND EXPENSES

USENIX averaged 5,700 members in 2004, an 11% drop from the previous year. Of these, half opted for SAGE membership as well.

Chart 1 shows the total USENIX membership dues revenue ($632K) for 2004, 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.

OTHER USENIX PROGRAMS

Chart 3 describes how the money allocated to student and outreach programs and standards activities ($200K) was spent in 2004. Chart 4 shows how the USENIX administrative expenses were allocated. (The category "Misc." covers such items as legal fees, elections, renewals, taxes, licenses, and bank charges.)

SAGE

Chart 5 shows SAGE revenue sources for 2004 (primarily membership dues of $145K and the $114K revenue share from its sponsorship of the LISA conference). Chart 6 shows all SAGE expenses (a total of $226K).

2006 USENIX NOMINATING COMMITTEE

The biennial election of USENIX's Board of Directors will be held in the spring of 2006. The USENIX Board has appointed Kirk McKusick to serve as chairman of the Nominating Committee. The composition of this committee and instructions on how to nominate indiviuals have been sent to USENIX members electronically and published on the USENIX Web site.

USENIX ASSOCIATION
STATEMENTS OF FINANCIAL POSITION
As of December 31, 2004 and 2003

ASSETS	2004	2003
Current Assets		
Cash & cash equivalents	$ 849,131	$ 705,498
Receivables	86,417	91,265
Prepaid expenses	53,316	24,409
Inventory	10,829	16,519
Total current assets	999,693	837,691
Investments at fair market value	5,205,701	4,916,653
Property and Equipment		
Office furniture and equipment	464,569	444,540
Less: accumulated depreciation	(380,152)	(321,667)
Net property and equipment	84,417	122,873
Other assets	328,014	302,785
	$ 6,617,825	$ 6,180,002

LIABILITIES AND NET ASSETS		
Liabilities		
Accrued expenses	$ 312,556	$ 108,036
Deferred Revenue	3,060	13,000
Total liabilities	315,616	121,036
Long-term Liabilities	328,014	302,785
Net Assets		
Temporarily Restricted Assets		
Unrestricted Net Assets	5,974,195	5,756,181
Net Assets	5,974,195	5,756,181
	$ 6,617,825	$ 6,180,002

USENIX ASSOCIATION
STATEMENTS OF ACTIVITIES
For the Years Ended December 31, 2004 and 2003

	2004	2003
REVENUES		
Conference & workshop revenue	$ 3,264,893	$ 3,049,096
Membership dues	632,252	649,079
Product sales	8,825	13,524
SAGE dues & other revenue	145,673	136,119
SAGE Certification	0	1,200
Total revenues	4,051,643	3,849,018
OPERATING EXPENSES		
Conferences & Workshops	2,704,759	2,577,839
Membership; login:	428,755	470,772
Projects & GoodWorks	229,239	148,854
SAGE	208,757	242,773
SAGE Certification	0	6,878
Management and General	441,554	406,727
Fund Raising	112,885	110,840
Total operating expenses	4,125,949	3,964,682
Net operating surplus/(deficit)	(74,306)	(115,664)
NON-OPERATING ACTIVITY		
Donations	50	100
Interest & dividend income	161,436	170,155
Gains & losses on marketable securities	199,495	461,758
Investment fees & other	(68,661)	(59,653)
Net investment income & non-operating expense	292,320	572,360
Increase/(decrease) in net assets	218,014	456,696
Net assets, beginning of year	5,756,181	5,299,485
Net assets, end of year	$ 5,974,195	$ 5,756,181

USENIX ASSOCIATION
STATEMENT OF FUNCTIONAL EXPENSES
For the Years Ended December 31, 2004 and 2003

	Conferences and Workshops	Programs an(Membership	Student Programs, Good Works and Projects	SAGE	Sage Certification	Total Program	Managemen t and general	Fund Raising	Total Support	2004 Total	2003 Total
Operating Expenses											
Conference & workshop-direct *	$ 1,855,355	$	$	$	$	1,855,355	$	$	$ 0	$ 1,855,355	$ 1,631,705
Personnel and related benefits:											
Salaries	509,536	121,306	17,128	114,140		762,110	210,831	75,182	286,013	1,048,123	1,150,706
Payroll taxes	49,385	11,757	1,660	11,063		73,865	20,434	7,287	27,721	101,586	82,928
Employee benefits	101,554	24,177	3,414	22,749		151,893	42,020	14,984	57,004	208,893	215,345
Membership/proceedings		8,254				8,254			0	8,254	7,370
Membership/login:		164,866				164,866			0	164,866	187,520
Membership/e-learning						0			0	0	0
SAGE expenses *				39,630		39,630			0	39,630	68,839
SAGE Certification expenses *						0			0	0	6,878
. Student programs, Good Works, and projects			200,083			200,083			0	200,083	108,019
General and administrative *	188,929	98,394	6,954	21,176		315,453	168,269	15,432	183,701	499,154	505,372
	$ 2,704,759	$ 428,755	$ 229,239	$ 208,757	$	3,571,509	$ 441,554	$ 112,885	554,439	$ 4,125,949	3,964,682

USENIX ASSOCIATION
STATEMENTS OF CASH FLOWS
For the Years Ended December 31, 2004 and 2003

	2004	2003
CASH FLOWS FROM OPERATING ACTIVITIES		
Change in net assets	$ 218,014	$ 456,696
Adjustments to reconcile increase in net assets to net cash provided by/(used for) operating activities:		
Depreciation	58,485	64,728
Net investment income designated for long-term purposes	(89,553)	(108,132)
Realized & unrealized gains on investments	(199,495)	(461,758)
Decr/(Incr) in receivables	4,848	(46,346)
Decr/(Incr) in inventory	5,690	5,526
Decr/(Incr) in prepaid expense	(28,907)	3,415
Incr/(Decr) in accrued expenses	204,520	(247,533)
Incr/(Decr) in deferred revenue	(9,940)	610
Total adjustments	(54,352)	(789,490)
Net cash provided by operating activities	163,662	(332,794)
CASH FLOWS PROVIDED BY/(USED FOR) INVESTING ACTIVITIES:		
Purchase of investments	(2,832,308)	(3,730,254)
Sale of investments	2,832,308	3,730,254
Withdrawals from reserve fund		
Purchase of property & equipment	(20,029)	(11,002)
Net cash used for investing activities	(20,029)	(11,002)
Net change in cash & equivalents	143,633	(343,796)
Cash & equivalents, beginning of year	705,498	1,049,294
Cash & equivalents, end of year	$ 849,131	$ 705,498

Chart 1: USENIX Member Revenue Sources 2004

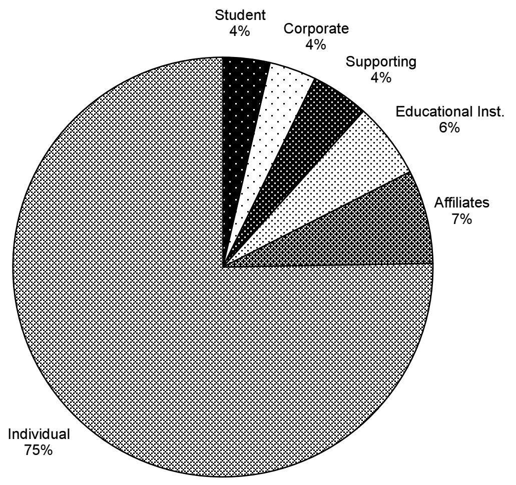


Chart 2: Where Your 2004 Membership Dues Went

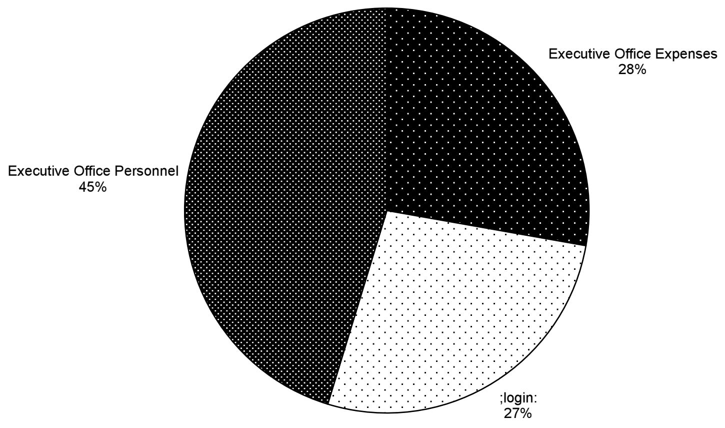


Chart 3: Student & Outreach Program Expenses 2004

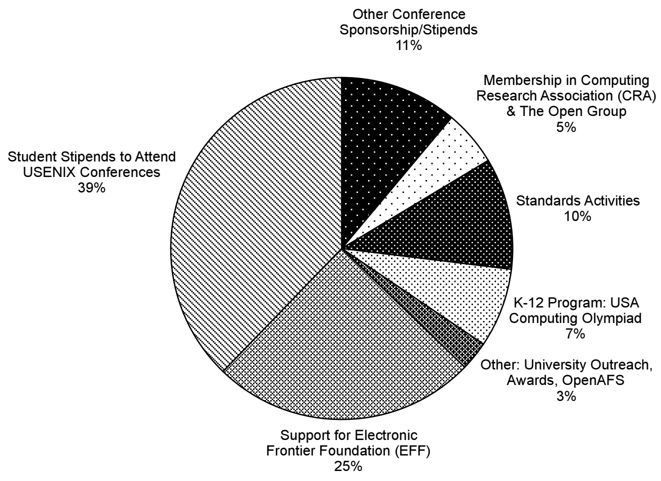


Chart 4: USENIX Administrative Expenses 2004

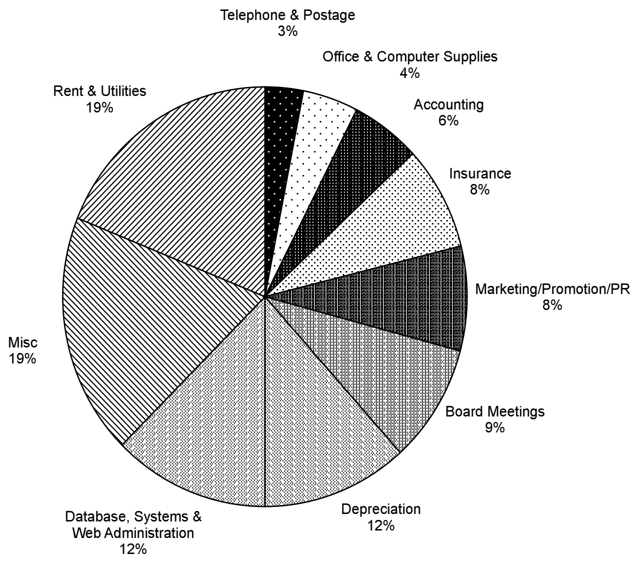


Chart 5: SAGE Revenue Sources 2004

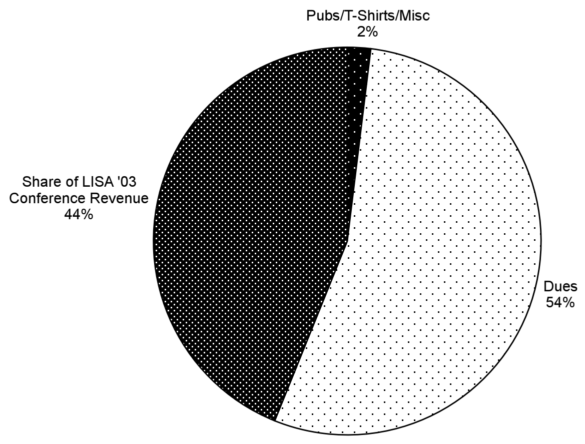
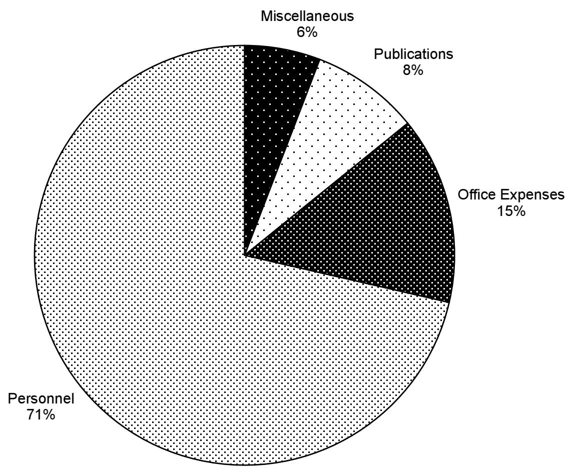


Chart 6: SAGE Expenses 2004



NEW!

;login: Surveys

To Help Us Meet Your Needs

;login: is the benefit you, the members of USENIX, have rated most highly. Please help us make this magazine even better.

Every issue of *;login:* online now offers a brief survey, for you to provide feedback on the articles in *;login:* . Have ideas about authors we should—or shouldn't—include, or topics you'd like to see covered? Let us know. See

http://www.usenix.org/publications/login/2005-10/

or go directly to the survey at

https://db.usenix.org/cgi-bin/loginpolls/oct05login/survey.cgi

Attention, Members:
Are You Getting the Most Out of Your Membership?

Become an active member of the Association. This is your community: get involved!

We are proud of our 30-year history of offering services to the advanced computing systems community. The support and participation of our members make us able to offer some of the most highly respected conferences and publications in the industry.

We have recently added the benefit of a Jobs Board for all members, as well as additional benefits for our Educational, Corporate, and Supporting members. We encourage you either to upgrade your membership or to talk to your employer about an institutional membership with USENIX.

In addition to the great benefits you already enjoy, we are offering these new benefits:

STANDARD MEMBERSHIPS: INDIVIDUAL ($115 PER YEAR) AND STUDENT ($40 PER YEAR)

- The USENIX Jobs Board: Looking for a new job? USENIX members have direct access to offerings from top-notch potential employers. For information on how to post, see http://www.usenix.org/jobs/.

EDUCATIONAL MEMBERSHIP ($250 PER YEAR)

- The USENIX Jobs Board (see above)
- Up to two additional copies of *;login:* per issue (email office@usenix.org with your request)

CORPORATE MEMBERSHIP ($460 PER YEAR)

- The USENIX Jobs Board (see above)
- Up to four additional copies of *;login:* per issue (email office@usenix.org with your request)
- Up to five conference registrations at the USENIX member price for your staff (email conference@usenix.org for a discount code to use in registering)
- Your company name listed on our Corporate Members Web page, http://www.usenix.org/membership/corporate.html.

SUPPORTING MEMBERSHIP ($2500 PER YEAR)

- The USENIX Jobs Board (see above)
- Up to four additional copies of *;login:* per issue (email office@usenix.org with your request)
- Tarballs of any USENIX conference Proceedings from the year *before* your membership term begins (email office@usenix.org with your request)

For a full listing of all benefits or to join online, please see http://www.usenix.org/membership.

conference reports

OUR THANKS TO THE SUMMARIZERS:

2005 LINUX KERNEL DEVELOPERS SUMMIT
Jonathan Corbet

WITMEMO '05
David Blinn
Minkyong Kim
Irfan Sheriff

EESR '05
Ahmad T. Al-Hammouri
Apratim Purakayastha
Himanshu Raj
J. Sairamesh

MOBISYS '05
Mike Blackstock
David Johnson
Neil McCurdy
Xiaoqiao (George) Meng
Himanshu Raj
Ram Kumar Rengaswamy
Ya-Yunn Su
Denitsa Tilkidjieva
Hongwei Zhang

HOTOS X
Prashanth Bungale
Rik Farrow
Alexandra Fedorova
Nikolaos Michalakis
Steve VanDeBogart
Steve Zhang

VEE '05
Xing Fang
Long Fei
Shuo Yang

SRUTI '05
Jayanthkumar Kannan
Balachander Krishnamurthy
Lakshminarayanan Subramanian

2005 Linux Kernel Developers Summit

Jonathan Corbet is a co-founder of LWN.net and the author of its kernel content. He is the lead author of *Linux Device Drivers,* 3rd edition, published by O'Reilly. For the last four years, Jonathan has been on the planning committee for the Kernel Summit.

In the young and fast-moving Linux community, anything that has happened for four years in a row can be called "traditional." Thus, the 2005 Linux Kernel Developers Summit, held on July 18 and 19 (immediately prior to the Ottawa Linux Symposium), is by now a traditional event. For two days each year, this invitation-only crowd of around 70 core kernel developers gather to talk about where kernel development should go over the next year. Few important development decisions were made at the 2005 Summit, but it was an opportunity for developers to catch up on what is happening in areas outside their particular expertise and, of course, to pursue topics of interest in the hallway and pub meetings.

The 2005 Summit opened with a panel of processor architects. This panel has, over the years, served as a forum where manufacturers could share some of their plans and hear about any concerns the kernel developers have. Two themes stood out this year: power management and virtualization. Manufacturers need to reduce the power demands of their chips lest future systems be required to be equipped with cryogenic cooling units; they would like to have help from the kernel developers in designing algorithms (scheduling in particular) that can help with power management. The developers, in return, would like a reliable way to ask the processor what its current power consumption is so that power-related changes can be benchmarked. There is quite a bit of hype around virtualization—running guest operating systems on top of software-implemented virtual machines—and the processor manufacturers are responding by adding better virtualization support to their CPUs.

A session on I/O busses mostly concerned technical details on the best interfaces for DMA operations and dealing with memory-challenged devices and systems. The kernel contains several independent mechanisms for setting up and executing scatter/gather I/O; it was agreed that it might be nice to unify these subsystems at some point, but nobody seems in any real hurry to do the work. There was discussion of a new memory allocation interface that would help drivers work around the memory addressing limitations found in a discouraging number of devices; a patch is expected soon.

The virtual memory management session showed that nobody is itching to make major changes to how the VM subsystem works—in the near future, at least. Instead, attention is currently focused on dealing with memory fragmentation and memory pressure. The most likely short-term solution to fragmentation (where it gets hard for the kernel to allocate multiple, contiguous pages) is a new allocation scheme that would segregate user-space pages (which are easily moved) from kernel memory allocations (which are not). Linux still does not behave as well as anybody would like when memory gets seriously tight; the issue here seems to be finding a good way to throttle memory-intensive processes without creating performance problems.

A brief session discussed some security-related patches that could be merged soon. These include better kernel API checks (finding double-free errors, for example), more address space randomization, making use of recent gcc features, and tightening access to various files in /proc and /dev/mem.

A session dedicated to virtualization had little to say; most of that work is in user space at this point.

There is interest in merging the Xen patches sometime soon. Those patches have been significantly reworked (Xen used to add itself as an entirely new architecture, but that did not go over well with the kernel developers) and should find their way into the kernel before too long.

The final session on Monday was dedicated to the virtual filesystem layer. Some potentially contentious issues (such as the merging of Reiser4 or FUSE) were avoided entirely; instead, the discussion centered on the increasing complexity of the core VFS code. In particular, the mixture of direct and buffered I/O has created a difficult mess that somebody will eventually have to clean out.

Tuesday began with a panel that addressed the frustrations faced by hardware manufacturers who wish to work with the kernel development process. The corporate way of doing things, involving fixed schedules, lengthy internal quality assurance work, and control over the code, does not mix well with the Linux process. Getting code into the kernel is easier if that code is posted at a very early stage so that show-stopper problems can be identified and fixed. But hardware companies would rather just produce a fully functional, tested, and certified driver at the end. That approach can get them sent back to the drawing boards with fundamental problems to fix. The vendors also complained about the constantly changing kernel API, which gives them long-term support problems.

The networking developers had held a summit of their own the week before the Kernel Summit; the outcomes were summarized for the crowd. A great deal is happening in the network community, including the reworking of much old code, the de-bloating of the core sk_buff structure (which represents a packet in the kernel), better cryptographic and security support, and more. We may even see support for hardware TCP offload engines, something that has been resisted by the networking developers for years.

From networking, the discussion turned toward the increasing convergence of the networking and storage subsystems. Storage-area networks, iSCSI, and so on are making networking a crucial part of the block I/O subsystem. This convergence can cause problems when memory gets tight; the block layer needs to write out pages to free memory, but a network-based storage layer must allocate memory to accomplish those writes. There are things that can be done to address this problem, but Linus Torvalds also wants to push back on the manufacturers of these systems. Rather than go through all this trouble to make network-based storage work, wouldn't it be better to just install a local disk? That said, there are real reasons behind these technologies, and Linux will find a way to support them properly.

A brief session on clusters showed that there was not a whole lot to concern the kernel developers; once again, most of the work is now in user space. There will be moves to merge a couple of cluster file systems soon (RedHat's GFS and Oracle's OCFS2); it seems that the two might have agreed to use the same distributed lock manager.

The session on RAS tools was mostly a celebration of the merging of the kexec and kdump patches, which should bring reliable crash dump capability to the mainline kernel. There are still quite a few loose ends to tie down.

Real-time capability for the Linux kernel has been the subject of a great deal of intense discussion over the last year. Most of that intensity failed to show up at the Kernel Summit session dedicated to the topic, though. There was some talk of how the various ways of providing real-time response could be judged, but no time to actually apply those criteria. So real-time can be expected to continue to heat up the mailing lists for a while yet.

The Desktop Developers Conference was happening at the same time as the Kernel Summit; a few delegates came over to give the kernel developers an update. The core of the discussion consisted of grungy details on how to rationalize Linux support for graphics cards. These cards are complex devices, often with secret interfaces. In the past, there has been a great deal of confusion as to whether these cards should be controlled by the kernel or by the X server in user space. These issues are slowly being worked out, and better graphics support should be coming to a screen near you shortly.

The kernel developers heard a report from the power management summit, held two days earlier. Much work remains to be done in the power management area. There are currently two software suspend implementations, neither of which is as solid as its users would like. It was agreed that the external "suspend2" patches would be posted and considered for merging into the mainline. Video adapters are a constant challenge in making suspend work; they are supposed to be reinitialized by the operating system on resume, but the manufacturers will not tell the Linux developers how to do that. So, instead, pressure is now being put on the BIOS vendors to provide that reinitialization support in the firmware.

The final session, traditionally, is devoted to the kernel development process and the ongoing desire to extract hard deadlines from Linus Torvalds. Deadlines were less of an issue this year; instead, the developers were concerned with improving the quality of kernel releases; recent kernels are seen by many as containing too many bugs. Two reasons were identified for this: kernel developers are waiting too long into the release cycle to merge their changes (thus missing out on weeks of testing time), and bugs, even when identified, are not being fixed. An attempt will be made to address the first problem by requiring that new features be merged into the kernel within the first couple of weeks of the cycle. After that, a feature freeze of sorts will be imposed, and only fixes will be merged. Getting developers to actually fix bugs can be a bigger challenge when there is no boss to order them to fix things.

Overall, the 2005 Summit was seen as a successful gathering. Some developers have noted that, over time, the summit is moving away from a forum where issues are debated and decided and is becoming instead a two-day status report. Given that the kernel has grown to a point where nobody can really understand every part of it, such a status report can be important. But if the summit is not a place where decisions are made, some of the developers may stop coming. So there may be changes made in the future to spice things up a bit.

For more detailed reporting from the summit sessions, please see http://lwn.net/Articles/KernelSummit2005/.

International Workshop on Wireless Traffic Measurements and Modeling (WitMeMo '05)

Seattle, WA
June 5, 2005

KEYNOTE ADDRESS

Summarized by Minkyong Kim

■ *Dynamic Adaptation and Mobile Wireless Systems: Experiences and Challenges*

Margaret Martonosi, Princeton University

Mobile computing systems present several challenges. First, mobile computing happens on devices with constrained optimization and highly varying applications. Applications constantly change, and new applications present new constraints. Second, hardware also changes quickly. Instead of abstracting hardware, people tend to make software to fit their needs. Without hardware abstraction, the first challenge becomes more severe. Third, there are always new metrics for success. David Kotz (Dartmouth College) commented that the reason people do not do abstraction is that they want more control.

ZebraNet, a project consisting of a network of mobile sensors designed for animal tracking, started three years ago in response to biologists' desire to track animals over an extended period and long distances. Biologists will use the resulting information to suggest ways to manage land to preserve wildlife.

ZebraNet uses a store-and-forward network to collect sensor data. Each sensor has a radio with a one-kilometer range, though the effective range was only 100–800 meters due to a ground loop caused by the stitching in the collar. Sensor collars put on the necks of seven zebras exchange data every two hours with others within range (i.e., 2 km). The collar, designed from scratch, trades processor cycles to optimize radio transmissions, because computation requires less energy than radio transmissions.

Beyond ZebraNet, there are three challenges: a lack of stable application drivers on which to experiment, a lack of good experimental infrastructure, and a lack of data sharing among researchers. The first two challenges are difficult to change, but the last should be easier. Currently, data sharing takes place more or less exclusively at conferences or workshops. This community needs broader-scale sharing. Martonosi also advocated creating test-beds and simulation environments.

Martonosi concluded her talk with the following research questions: What can we do to tolerate sparse and high-disruption wireless networks? What do we do if a source-to-destination route never exists or exists only rarely? How can we reduce the packet delivery latency for disseminating data? How do we better support infrastructure for real-system wireless measurements?

Maria Papadopouli (University of North Carolina) commented on the difficulty of correlating data from different sensors and also the problem of measurement errors. David Kotz (Dartmouth) asked whether there have been similar problems in space research. The speaker said that it is similar but the distinction is that the events in space operations are relatively well scheduled, whereas zebras are random. She mentioned that the range of control is also different.

http://www.princeton.edu/~mrm/zebranet.html

MORNING PRESENTATIONS

Summarized by Irfan Sheriff

■ *Analysis of a WiFi Hotspot Network*

David P. Blinn, Tristan Henderson, and David Kotz, Dartmouth College

David Blinn began by talking about some large-scale WiFi studies that have been conducted. This study is smaller than many of those and is based on the analysis of data from 312 access points installed at Verizon phone booths in New York. However, this is the first work based on the traces collected from a production 802.11 network. The access points are connected through an ADSL backbone.

The study involved looking at the number of active users and pending users, and the movement of users among access points. It was conducted for five weeks in November–December 2004. Most results were not surprising—diurnal usage, weekly usage patterns, 5% of cards generating about 85% of traffic, and the like. The paper concluded that users were not very mobile and usually stuck to one access point.

Maria Papadopouli suggested that users who were always on the network might be working from home. People were interested in the data and there was discussion of attempts to make the data public.

- ***MobiNet: A Scalable Emulation Infrastructure for Ad Hoc and Wireless Networks***

Priya Mahadevan and Amin Vahdat, University of California, San Diego; Adolfo Rodriguez, IBM and Duke University; David Becker, Duke University

Priya Mahadevan presented an emulation environment for ad hoc wireless networks called the MobiNet. The advantage of this work is that real applications can be simulated on a large scale in the emulator. The drawbacks of the simulator, such as simplified physical layer models and simplified mobility models, remain. This work tries to combine the good features of the simulators (repeatability, efficiency) with that of live deployment (real application usage).

Mahadevan presented the general architecture and evaluation results, which showed that the model is accurate, scalable, and can support applications unmodified. MobiNet performed as well as NS-2, at the same time supporting NS-2.

Ashu Sabharwal said he did not see a big difference between MobiNet and distributed NS-2, because real application traces can also be fed into NS-2. He also noted that with large-scale deployment, the jitter encountered on the Ethernet line between the core and edge nodes could affect the results. Mahadevan countered by saying that this could be solved by limiting the number of applications per edge node and having separate Ethernet lines.

Additional information is available at http://ramp.ucsd.edu/~pmahadevan/publications/Mobinet_techrep.pdf.

- ***An Accurate Technique for Measuring the Wireless Side of Wireless Networks***

Jihwang Yeo, Moustafa Youssef, and Ashok K. Agrawala, University of Maryland; Tristan Henderson, Dartmouth College

Tristan Henderson presented a measurement architecture for 802.11 networks on the wireless side of the network using a set of sniffers. Most earlier measurement work has been on the wired side of the 802.11 network, which cannot capture the full details of the packets, because the access point strips some of the headers before handing over the packets. This work looks at countering the deficiencies of the earlier work.

The setup consisted of three sniffer PCs equipped with prism2 802.11 cards. The tools used included libpcap-based ethereal to capture data. The metric of importance was completeness of data, and three sniffers were found to be sufficient for one access point. There was a discussion of the effect of different drivers on the packet data collected. Maria Papadopouli thought it should not have an effect. However, people agreed that current device drivers handle packets differently and that such handling should be standardized. SNMP seems to be buggy too, as they found a weird packet result count with SNMP.

Ashu Sabharwal asked how intrusions and MAC layer misconfigurations can be detected? Papadopouli pointed out that *some* cases of misconfigured clients and APs can be detected by analyzing SNMP data.

- ***Modeling Users' Mobility Among WiFi Access Points***

Minkyong Kim and David Kotz, Dartmouth College

Minkyong Kim presented a paper on modeling user mobility based on traces collected at Dartmouth College. The tests were conducted from April to May 2003. The Flux model, as it is called, clusters the set of access points that have the common behavior of user distribution during different periods of time.

The model constructs five clusters with a set of access points that have similar peak behavior. Alex wondered why there were just five clusters. Kim responded that the number should probably depend on the environment and the behavior of the users. Ashu Sabharwal asked if there were any surprises in the peak behavior. Kim felt there were no significant surprises. There was a discussion about the need for better models that closely simulate realistic user behavior.

AFTERNOON PRESENTATIONS

Summarized by David Blinn

- ***An Experimental Study of Multimedia Traffic Performance in Mesh Networks***

Yuan Sun, Irfan Sheriff, Elizabeth M. Belding-Royer, and Kevin C. Almeroth, University of California, Santa Barbara

Irfan Sheriff presented experimental results of streaming video and voice traffic testing on a 25-node 802.11b mesh network test-bed (http://moment.cs.ucsb.edu/meshnet/). The network topology consisted of static routes between a sequence of nodes in a four-hop path.

The experimenters observed that fewer simultaneous voice flows could be well supported than video flows and determined that the number of packets per second sent by an application had a greater impact on quality than the size of the packets. Increasing the number of multimedia flows beyond a saturation point caused dramatic jumps in both latency and loss, and increasing the number of hops decreased the saturation point. A greater number of flows also resulted in an increase in latency variation, or jitter.

With multiple video flows, unfairness became an issue, with some flows consuming great amounts of bandwidth leaving little for the other flows. Unfairness was an even bigger problem in voice flows. At the conclusion of the presentation, some members of the audience expressed concern that some of the measured results, such as the effects of RTS/CTS, might be specific to the testing topology and would fail to generalize to other networks.

■ ***The Perils of Simplified Simulation Models for Indoor MANET Evaluation***

Eyal de Lara, University of Toronto

Eyal de Lara opened his invited talk by questioning the value of Mobile Ad Hoc Network (MANET) simulations when the simulations use extremely simplified space and mobility models. MANET simulations frequently assume that the networks are deployed in free space, with no impediments to radio communication, and that users move between random waypoints. These models are inadequate and not robust.

He introduced a detailed model, called attenuation factor (AF), of space and user mobility with an AutoCAD map of a building and waypoints between sensible points in building rooms. In evaluating two routing protocols, DSDV and DSR, using these models, the detailed model predicted similar results to the simplified models for DSDV, but dramatically different results for DSR. He identified that this difference was caused by the effect of frequent link breakages. The AF model observed these breakages, which greatly affected the results.

De Lara stressed that he was not claiming his AF model was perfect, but rather that he wished to show that simplifications in models could produce nonuniform variations, and that it is nontrivial to determine which simplifications are important and which are not.

A paper describing the work is available at http://www.cs.toronto.edu/~delara/papers/secon2004/.

■ ***Measurement Study of Path Capacity in 802.11b-Based Wireless Networks***

Tony Sun, Guang Yang, Ling-Jyh Chen, M.Y. Sanadidi, and Mario Gerla, University of California, Los Angeles

Tony Sun began by remarking that studying wireless path capacity is complicated by the dynamic conditions affecting the links, and presented a scheme to better estimate the maximum achievable data rate in a multi-hop wireless path.

The new scheme, AdHoc Probe, builds upon the previous CapProbe (http://nrl.cs.ucla.edu/CapProbe/) scheme, which uses a packet pair to estimate the capacity of a link. Measuring path capacity in a wireless ad hoc network is more difficult than in a wired network, due to the effects of bottleneck capacity, network topology, interference, the use of 802.11 auto-rate, and RTS/CTS. AdHoc Probe is superior to CapProbe because it uses faster, less interference-prone one-way transmission instead of CapProbe's two-way transmission. Tristan Henderson (Dartmouth) questioned the absence of SIFS/DIFS delays in the researchers' calculations. Sun responded that the aim of the research was only a rough estimation of theoretical link capacity.

Implementation issues in the new scheme include system time synchronization between the links, which can be negated by summing over minimum recorded times, and clock skew caused by clock racing, which can be accounted for by examining trends in clock timing. In an experimental setup of 802.11 access points, experimental results verified the scheme's prediction of C/3 for a three-hop network and C/4 for a network with four hops or more, with C being the single-hop capacity of the links. The presentation concluded with a debate over the precise meaning and usefulness of the notion of wireless path capacity.

PANEL SESSION

Summarized by David Blinn

■ ***Who's Afraid of Wireless Measurements Studies?***

Panelists: Christophe Diot, Intel Research Cambridge; David Kotz, Dartmouth College; Maria Papadopouli, University of North Carolina; Ashu Sabharwal, Rice University

The panelists took turns expressing their hopes and fears for the future of the field of wireless measurement studies. Maria Papadopouli emphasized the need to integrate two directions: measurement and modeling. Benchmarks and metrics should be defined to identify access patterns. David Kotz stressed the need to build a strong measurement community, noting the inconsistencies in definitions and tools. He introduced a new endeavor, CRAWDAD: A Community Resource for Archiving Wireless Data at Dartmouth (http://crawdad.cs.dartmouth.edu), which will include an archive of wireless data sets and a set of tools and will run measurement workshops. Ashu Sabharwal worried that the community might be reinventing the wheel by focusing too much effort on systems that are essentially irrelevant. He introduced the CMC Open Wireless Platform, an upgradable and expandable wireless testbed for testing in all layers of the network stack. Christophe Diot pointed out the need for a strong community and uniform testing equipment along with a common standard to calibrate test-beds. After these introductions, the panel took questions and a spirited discussion followed.

Papadopouli and Sabharwal argued for the need to raise standards within the community, which might require more visible workshops and conferences and a willingness to reject papers that make no attempt to justify their underlying assumptions. The panelists also emphasized the need for multi-disciplinary and interdisciplinary research to support different layers of the systems (from the physical layer all the way through the application), as well as the need for strong ties with statisticians.

James Scott (Intel Cambridge) asked if trace-based experiments could be made as simple as running a simulator. The panel answered that it might be possible to take a measurement and replay it, but someone has to set up actual equipment and find the traces in order to make it easy. People will want scalability in such a setup and will want to be able to tweak parameters of the trace.

Scott also asked how we might judge the success of a simulation or trace when we have to sacrifice realism for reproducibility. David answered that this problem is one encountered in the natural sciences because, like a wireless network, not all elements of nature can be controlled. Following the example of these sciences, the solution is to perform many studies and use statistics to overcome the problem of the huge number of parameters in experiments, for instance in a multi-factor experiment. Simulation can then be used to drive experimentation.

Workshop on End-to-End, Sense-and-Respond Systems, Applications, and Services (EESR '05)

Seattle, WA
June 5, 2005

OPENING REMARKS

Chatschik Bisdikian, of IBM Research, the co-chair of the EESR workshop, explained that the motivation behind the workshop is the context-aware middleware work for pervasive applications. The goal is to examine the role of sensor and actuators and to apply the technology in Internet-scale systems.

Advances in sensor technology and increased intelligence in actuators have provided a rich set of applications. Businesses want to use these Sensor and Actuator Networks (SANETS) to make better decisions and to form true sense-and-respond (S&R) systems. An S&R system has not only sensors and actuators but decision analysis or control components. S&R systems evolve from SANET applications by taking an end-to-end view, where data is sensed, interpreted by the decision-making components, and then acted upon. An example of such a system is the context-aware electricity grid, part of the GridWise project, the subject of the keynote address. In addition to the keynote address, the workshop hosted eight papers, with half covering S&R technologies and half covering applications. The workshop closed with a lively panel discussion.

KEYNOTE ADDRESS

Summarized by Himanshu Raj

- ***The GridWise Project***

Rob Pratt, Pacific Northwest National Laboratory

Rob Pratt presented several problems in existing power grid design. The infrastructure consists of three layers: the generation, the transport (physical wires), and the substations for distribution to end users and businesses. Because the transport layer is exposed to the elements, the whole system is vulnerable to disruptions, with serious consequences to the economy. In addition, the distribution system, from big generation plants to users, does not support third-party, small-scale power producers. One objective of the GridWise project, whose overall goal is to apply sense-and-respond research to the power grid of the future, is to build the "nervous system" for the electric power grid by making a ubiquitous communication infrastructure. This will allow intelligent distribution of power via facilitating cross-level communication.

What causes blackouts? Generally, complete blackouts are caused by a ripple effect in which part of the electric grid goes down and the rest of the system tries to keep up with the load. When the system cannot meet the demand, more of the system shuts down, creating a vicious cycle and eventually resulting in a complete blackout. To address this problem, we need intelligence built into end-user appliances. Such appliances will be able to take actions that are largely unnoticeable to the user and can help the overall system adjust to load shedding. Building such a system, however, requires collaboration between the Grid operators and device manufacturers.

In conclusion, markets and control systems are ultimately going to merge. The ability to optimize at lower granularities than the ones available today will drive future business processes. More information on the GridWise project is available from http://www.gridwise.org.

Most of the questions related to the basic electrical engineering behind power generation, such as how multiple generators are kept in sync in terms of frequency (one generator starts up and works as the heartbeat while others synchronize to it) and whether we can packetize power and treat it like data on the Internet to store and forward (no, but there are some taps in the making that can maintain the flow of power on different lines).

SENSE 'N' RESPOND TECHNOLOGIES

Summarized by Jonathan Munson and Apratim Purakayastha

■ *A File System Abstraction for Sense and Respond Systems*

Sameer Tilak, Kenneth Chiu, and Nael Abu-Ghazaleh, State University of New York (SUNY) at Binghamton; Bhanu Pisupati and Geoffrey Brown, Indiana University

Challenges for wireless sensor networks (WSNs) include heterogeneity in hardware and software, communication models between system elements, the scale of the system, and resource constraints. Previous approaches have examined DB abstractions, but these are too close to the application level. Current abstractions are programming-language– and OS–based. In this presentation, Sameer Tilak proposed a filesystem abstraction. With this approach, the WSN is presented as a distributed file system. The interface is well understood, hides heterogeneity, enables application-specific namespaces, offers structured namespaces, and allows fine-grained control over resources. Tilak then introduced an extended example, a Smart Zoo. The research challenges they face include supporting resource-efficient operation and in-network application-specific processing, the consistency model, and tolerating network unreliability.

A member of the audience asked about the rate of change of the directory structure. Tilak responded that most changes are localized even though the system is dynamic. Weak consistency is good enough. Another member of the audience asked about using a content-driven approach rather than a directory-driven one. He offered TinyDB as an example of a content-driven approach. Tilak responded that a content-driven approach can be realized by application-specific namespaces. The next question was about how you can generate that organization automatically, to which Tilak responded that file systems easily support a dynamic scheme—unlike databases, whose schemas are generally fixed.

■ *Transversal Issues in Real-Time Sense-and-Respond Systems*

Ahmad T. Al-Hammouri, Vincenzo Liberatore, and Huthaifa A. Al-Omari, Case Western Reserve University; Stephen M. Phillips, Arizona State University

Ahmad Al-Hammouri defined real-time sense-and-respond systems as remote control of a physical environment via an S&R system. The contribution of this work is formulating a conceptual framework for real-time S&R issues for networks.

Transversal issues in S&R systems arise because of the necessity of providing quality of service in real-time S&R systems. One issue is adaptability and tolerance to round-trip delays and jitter. They decompose delay into predictable delay and jitter. Jitter is dealt with via playback buffers. Dealing with playback delay is challenging because existing schemes (e.g., from multimedia systems) do not apply since the performance metrics are different. The delays in S&R systems are round-trip rather than one-way, and delays are determined by the controller. The scheme they used in this work is a combination strategy, in which the controller determines the playback delay based on the round-trip time. Al-Hammouri also discussed congestion control and enabling a choice of utility functions for different S&R systems.

During the discussion that followed, the similarities between this work and that of multimedia systems were explored. Al-Hammouri explained that multimedia is concerned about jitter. He also explained that they had examined queue lengths as well as packet-receive rates and output lengths. He presented fixed playback delay, but pointed out that you could focus on avoiding packet losses.

More information is available at http://vorlon.case.edu/~vxl11/NetBots.

■ *M-ECho: A Middleware for Morphable Data-Streaming in Pervasive Systems*

Himanshu Raj, Karsten Schwan, and Ripal Nathuji, Georgia Institute of Technology

Himanshu Raj presented M-ECho, a middleware for system morphing (the continuous and dynamic adaptation of services and systems). The objective is to extend the application's longevity by optimizing power consumption. The application area of interest is robotics.

M-ECho uses an event-driven, pub/sub architecture. Dynamically deployable codelets serve as event handlers and filters. It uses both dynamic code generation and static code repositories. To evaluate the system, the authors integrated it with the Player/Stage robotics framework and measured the power performance of two different mechanisms: code parameterization/substitution and code migration. In the future, Raj expects to examine robotics applications, the use of compiler-assisted techniques, and quantifying the energy cost of dynamic compilation of codelets versus using a static code repository server.

A member of the audience asked whether the optimization is done on each node or across the entire system. Raj responded that the optimization happens across the entire system. Another question was whether the optimization algorithm was centralized. Raj responded that it is currently centralized, but that they plan to use a clustered approach in the future.

- ***The Abstract Task Graph: A Methodology for Architecture-Independent Programming of Networked Sensor Systems***

Amol Bakshi and Viktor K. Prasanna, University of Southern California; Jim Reich and Daniel Larner, Palo Alto Research Center

Sensor networks involve three roles: the end user (usually a domain expert such as a scientist), the application developer, and the system programmer. In his talk, Amol Bakshi presented the Abstract Task Graph (ATaG), a macro-programming model that involves specification of aggregate behavior, not simply node-level programming. ATaG is an application-neutral approach and tries to support reuse. ATaG uses data-driven flow, in which events carry information about the phenomenon. It also uses mixed imperative/declarative specifications in order to separate "what" from "where" from "when."

Feng Shao asked when the binding of abstract tasks occurs. Bakshi responded that some are bound at runtime and others at compile time. Another member of the audience asked about the use of channels. Bakshi responded that the channel is not a communication link but is input/output plus a zone of interest. It is a way to associate a task to a data item.

SENSE 'N' RESPOND SOLUTIONS

Summarized by Ahmad T. Al-Hammouri

- ***A Sensor-Based, Web Service–Enabled, Emergency Medical Response System***

Nada Hashmi and Dan Myung, 10Blade, Inc.; Mark Gaynor and Steve Moulton, Boston University

Steve Moulton, a surgeon at Boston University, presented a scalable emergency response system that combines sensors, mobile databases, Web services, and wireless infrastructure technologies. Steve described current emergency medical services (EMS) problems: patient care reporting is paper-based, poorly captured, untimely, incomplete, and not searchable. The problem is exacerbated in mass casualty events, where there is a need to identify and distinguish each patient's condition so that patients with severe conditions receive immediate care. Also, there is a need for patients to be directed to the care center that can best treat the patient's condition. Consequently, there is a need for an emergency service system with improved communication, documentation, and exchange of information between the pre-hospital and hospital phases of emergency care.

Steve introduced a system that meets the above requirements. He explained the system's components and how data flows between these components. Emergency medical technicians (EMTs) enter patient information on PDAs and tablet PCs equipped with wireless connectivity. A sensor is attached to each patient to keep track of the patient's vital signs and the patient's location (via GPS). Patient sensors in one location form a sensor network. Data from each sensor network, along with data from PDAs and tablet PCs, used by EMTs, is aggregated at a local command center (an ambulance). All such communication occurs over a wireless network, e.g., 802.15.4. Data from different local command centers is then transferred via the Internet and aggregated at a central command center where resources can be managed. Decisions then propagate back to local command centers. The first prototype is built upon several technologies such as TinyOS, 802.15.4 and Zigbee, C# and Java, and Web Services and Grid.

- ***A Rule-Based System for Sense and Respond Telematics Services***

Jonathan Munson, David Wood, Gerry Thompson, and Alan Cole, IBM T. J. Watson Research Center; Sang Woo Lee and DaeRyung Lee, IBM Ubiquitous Computing Laboratory, Korea

Jonathan Munson defined telematics as vehicle-based computers and communication systems. Potential applications include traffic navigation, weather detection, congestion detection, safety vehicles, and distributed multimedia and gaming. Programming challenges arise when developing such applications because of the heterogeneities in data acquisition, data processing, data collection, event detection, and response processing.

Munson presented the Telematics Event Detection Service (TEDS), which offers a rule-based programming model that enables developers to more easily develop a wide range of event-driven telematics services. This project was done in the Ubiquitous Computing Laboratory in Seoul, Korea, jointly created by the government of Korea and IBM.

Rules present low-level events in terms of Boolean expressions or programs with different states. Rule inputs, such as position, velocity, or pressure, are fed from data acquisition devices. Functions defined on rules include spatial functions, temporal functions, and logical functions. A higher-level program can be constructed using the ABLE rule language, a low-level yet flexible language.

In the most recent TEDS prototype, a developer defines a set of rules and the actions for responding to events from the rules; it represents a sense-and-respond framework similar to the Struts framework for developing Web Services applications.

- ***Meteorological Command and Control: An End-to-End Architecture for a Hazardous Weather Detection Sensor Network***

  Michael Zink, David Westbrook, Sherief Abdallah, Bryan Horling, Vijay Lakamraju, Eric Lyons, Victoria Manfredi, and Jim Kurose, University of Massachusetts; Kurt Hondl, National Severe Storms Laboratory

Michael Zink presented the software architecture of the meteorological command and control (MC&C) component of a NetRad prototype. The goal of NetRad is to detect a tornado within 60 seconds of formation and to track its centroid with a temporal error no greater than 60 seconds. The MC&C component forms a closed control loop starting from ingesting data from remote radars (sensors), identifying meteorological features in the data, reporting features to end users, and determining each radar's future scan strategy based on detected features and end-user requirements. All of these steps need to complete within 30 seconds before another cycle starts.

A benchmark based on Nexrad radar data is used to determine whether the total processing time of all steps can fulfill the 30-second deadline. The results show that all these steps have sub-second execution times that make them well-suited for the NetRad system.

- ***Reducing Business Surprises Through Proactive, Real-Time Sensing and Alert Management***

  Mitch Cohen, Jakka Sairamesh, and Mao Chen, IBM T. J. Watson Research Center

This paper describes the system design of a unified semantic event-stream system that continuously monitors diverse data sources and generates alerts based on domain-specific rules. Such a system can enable manufacturers to closely monitor critical business events (reducing surprises) and gather business failures, warranty intelligence, field events, sales transactions, and asset performance. Jakka Sairamesh noted the factors motivating the need for such a system: lack of visibility in current business operations, rising costs for business management, and late reaction to critical events.

The system needed to address several challenges such as real-time sensing and monitoring, complex event management, domain-specific analytic, multi-format streams, and multi-rate streams. Jakka Sairamesh presented the high-level architecture for this system, which is primarily composed of the Event Stream Processor. The Event Stream Processor uses the Event Transformation Service to convert into the standard format, the Event Correlation Service to retrieve a list of metrics that need to get calculated, and the Session Update Service to ensure that the current event is included in future metrics calculations. The Metric Evaluation Service determines what, if any, actions need to be taken based on the newly calculated metrics. These actions are taken with calls to the Action Instantiation Service.

PANEL SESSION

Summarized by J. Sairamesh

- ***Research Challenges of End-to-End Sense 'n' Respond Solutions***

  Panelists: Ron Ambrosio, IBM T. J. Watson Research Center; Malena Mesarina, HP Labs; Vincenzo Liberatore, Case Western Reserve University; Feng Zhao, Microsoft Research

The panel session began with short presentations from each panelist. Feng Zhao focused on challenges in planet-scale S&R systems. He considered application domains such as autonomous vehicle tracking, health care, networked transportation, ubiquitous appliances, and motion sensing for security. He also talked about the role of sensor subnets, protocols for sensor subnet communication, and storage issues in S&R systems, and he considered issues pertaining to two applications in particular: a marine center for air-ground combat sensors and a system for monitoring space in parking lots. He demonstrated the challenge of tracking vehicles using ground-based sensors and the issues in real-time communication and coordination.

Malena Mesarina focused on the role of standards and return on investment (ROI) for sensor-based applications in the industry and physical environment. She talked about applications such as combat field tracking and habitat monitoring, and the role of motes in sensing. She also talked about RFID- and EPC-based sensor applications for the industry as being important and timely. She argued that the ROI is critical to enabling real deployments and that pilots are needed to validate such deployments. She has found that the technology is expensive, hard to program, and requires deep knowledge about the events.

Ron Ambrosio talked about a better closed-loop operations control between the sensor systems at the edge of the network and the operational enterprise systems at the core. He discussed the traditional model of computing, which had a separation between this core and the edge systems. He focused on architectural principles and programming models for scalable Internet control systems that have seamless integration between the edge nodes and the core enterprise nodes. He enumerated many architectural principles. He also defined "real-time" processing as not being simply handling a higher rate of information through sensors but a notion of determinism of the process and events.

Vincenzo Liberatore supported the desire to provide deterministic performance, even at the expense of latency. He supported the definition of "real-time" processing that Ambrosio described. He said that measurement, validation, and metrics for validation are critical in understanding scalable S&R systems.

He talked about a tele-epistemology application and the value of networked S&R in such applications. He argued that almost all of the S&R applications fall into the area of control theory models and that these models need to be applied and validated. He strongly argued the need for real-time S&R benchmarks, simulation, evaluation, metrics, emulation, autonomous operation, determinism in operation, and handling failure situations.

A fundamental question was raised by the audience on whether we were reproducing control theory work. The panel responded that, to some extent, S&R systems are already in the field (e.g., home control systems). However, these systems do not scale up. They also believe that existing control theory is being applied. Questions were raised about event management. The panel talked about event handling, event bus architecture, and standards for handling events.

The panel discussed standards in depth and agreed that standards should focus on accessing information and representing the data to abstractions provided by the sensors. As for what would enable S&R to succeed, the panel felt that the entire community has a role to play. Academia can contribute to application, methodology, and metrics for evaluation. In fact, the NSF has a $40 million program in this area. Business can also contribute, especially focusing on the ROI question.

MobiSys 2005: The Third International Conference on Mobile Systems, Applications, and Services

Seattle, WA
June 5–8, 2005

Tristan Henderson of Dartmouth College conducted a study of the conference WLAN during the conference. Questions regarding the study can be directed to him at tristan@cs.dartmouth.edu.

KEYNOTE ADDRESS

Summarized by Himanshu Raj

■ *Technology's Future: A Crisis in Confidence?*

Rick Rashid, Microsoft Research

Rick Rashid addressed the growing unrest among the CS community about the future of the IT industry and opportunities for research. He then outlined certain trends emerging from ubiquitous and pervasive domains and suggested that we are actually just getting started on research that will impact society at its core.

A terabyte of storage can hold every conversation you take part in for your entire life. It can hold one picture for every minute of your life. It can hold a year of everything you can see, in full-motion video. The SenseCam project at Microsoft Research is examining what you could do with a terabyte of personal storage. They have devised a wearable data and image recorder. The device contains a camera and numerous sensors (e.g., accelerometer, passive IR, light level, temperature, images, etc.). Image capture is triggered by sensing some interesting change in environment (e.g., lighting levels). They are hoping to apply this technology to patients with moderate memory impairment and as an aid for caregivers of patients with severe memory loss. It can also be used for self-reflection.

Rashid also gave examples of ubiquitous I/O, which turns any surface into an interactive computing surface, and of streaming intelligence, such as skyserver.sdss.org and skyquery.net, which allow scientists to do data mining against a large number of databases and has resulted in the discovery of new astronomical phenomena.

Rashid believes the goal of ubiquitous computing should be to bridge the gap between the rich and the poor. More than three billion people pay a poverty premium for basic goods and services. They have little access to important amenities and make decisions affecting their livelihood (e.g., when to plant crops) based on incomplete information. Even in the developed world, people with incomes below $35,000 have a 70% chance of not having good online access. Wiring to every home in the whole world is not realistic. Rashid described the Mesh Networking Project, which is applying wireless networking technology and mesh networks to form a cooperative network to cover the last hop. These networks must be autonomic, in that they should be self-managing, self-healing, and inexpensive. Other projects looking at this problem include the TIER project at Berkeley (http://tier.cs.berkeley.edu/) and the Digital Gangetic Plains project at IIT Kanpur (http://www.iitk.ac.in/mladgp/).

In conclusion, there are ample opportunities for reinvigorating CS research. The emphasis must be on access to information rather than mere access to devices.

During the Q&A session, Ramón Cáceres (IBM Research) asked about how to sustain technology change in rural areas. Rashid responded that you must find the economic value behind the technology and that, unless the person providing the information access points (such as Internet kiosks) makes money, he will not keep providing the service. Mandayam

Raghunath, also of IBM Research, followed up by asking how to get rural areas started. Rashid responded that there have been a number of approaches. One that worked in parts of India and Bangladesh was a bank that supports microloans. Tapan Parikh (University of Washington) pointed out that the positive effects we foresee from technology have to do with the applications that emerge from those technologies. Rashid agreed that this was true and then discussed whether this would cause the field of computer science to lose its identity. He pointed out that this style of research can bring risks from a career and from a funding perspective, but he also pointed out that about 50% of the basic research investment made by Microsoft goes toward projects that funding agencies are not (yet) comfortable with. He expressed hope that funding agencies would eventually view these cross-disciplinary projects as legitimate research areas.

APPLICATIONS ON THE GO

Summarized by Hongwei Zhang

■ ***A Systems Architecture for Ubiquitous Video***

Neil J. McCurdy and William G. Griswold, University of California, San Diego

Neil McCurdy presented the architecture of their ubiquitous video system, RealityFlythrough, which enables users to visually explore a remote area in real time. McCurdy started by discussing the potential applications of the system, which include disaster response, remote monitoring, remote navigation, and virtual shopping.

McCurdy introduced the technical challenges: the low density of camera deployment, the continuous movement of both the object and the camera, and the need for reliable, live, and real-time data delivery. He then elaborated on how he dealt with these technical challenges. RealityFlythrough uses “motion” as a substitute for an infinite number of cameras; it uses dynamic path estimation with smoothened transition between scenes; and it uses dynamic archiving of high-quality images from different locations to reduce the density of camera deployment and the amount of image data that needs to be delivered. He concluded the talk with a discussion of their evaluation. Overall, they found the number of live cameras to be the bottleneck, not the total number of cameras.

After the talk, McCurdy discussed with the audience issues such as how to fetch archival images, how to deal with outdated images, the impact of GPS precision on system performance, and the network challenges posed by the possible use of omnidirectional cameras.

More information is available at http://peanutgallery.homeip.net/drupal/taxonomy/term/1.

■ ***LiveMail: Personalized Avatars for Mobile Entertainment***

Miran Mosmondor, Ericsson Nikola Tesla; Tomislav Kosutic, KATE-KOM; Igor S. Pandzic, Zagreb University

Miran Mosmondor presented LiveMail, a prototype system for communicating personalized images between mobile devices such as cell phones. He briefly introduced the concepts and techniques of 3D modeling, face animation, and 3D graphics on mobile platforms, then described the client-server architecture of LiveMail. The client enables the user to customize the characteristics of face animation, and the server acts as the relay between the sender and the receiver. Mosmondor stressed the importance of performing face animation using parameter adaptation, instead of transmitting raw face images: by transmitting the parameters instead of images, LiveMail only requires a bandwidth of 0.3Kbps, which is available even in resource resource-constrained devices such as cell phones. Mosmondor pointed out that the server of LiveMail spent 98% of the time on parameter adaptation yet only 2% of the time on storage and retrieval.

Mosmondor answered questions regarding the relative importance of bandwidth and processing capability, the necessity of 3D rather than 2D face presentation, and the relation of LiveMail to other efforts. Mosmondor stressed that LiveMail does not require high bandwidth so much as high processing capability, such that the server could be dropped out of the architecture if the client (e.g., cell phone or PDA) is fast enough.

■ ***MediaAlert—A Broadcast Video Monitoring and Alerting System for Mobile Users***

Bin Wei, Bernard Renger, Yih-Farn Chen, Rittwik Jana, Huale Huang, Lee Begeja, David Gibbon, Zhu Liu, and Behzad Shahraray, AT&T Labs—Research

Bin Wei presented MediaAlert, a system for delivering TV news according to the interests of users. Wei discussed the techniques employed in MediaAlert: processing media and extracting descriptive abstractions of the media, and content repurposing to disseminate media to devices with distinct protocols and capabilities. He stressed the importance of aligning unsynchronized images and captions and the scalability of the system to support a large number of users and devices.

Wei walked through a scenario where a user created an identity and topics of interest, and MediaAlert then analyzed and delivered related images to the user. Wei also pointed out that media processing takes longer than dissemination and noted the importance of dealing with false positives in alert.

David Kotz asked whether the user’s location is used in the selection of content. Wei replied that it is and that the news content itself is location-dependent, because different news channels provide different content. Mark Corner asked whether closed captioning was important to this system. Wei replied that closed captioning is very important to the existing system and that this is an area that will evolve. Tristan Henderson asked whether they had done any usability testing. Wei said that they had only used the system within their own team and that more usability testing was necessary. Andre Hesse asked what happens once the user has received the maximum number of alerts for the day and then something really important happens. Wei clarified that the maximum number of alerts per day is for the scheduled alerts and that the system sends the most relevant clips first. Real-time alerts cover emergency events and are not subject to the maximum.

SHAKE 'EM, BUT DON'T CRACK 'EM

Summarized by Neil McCurdy

- *Cracking the Bluetooth PIN*

Yaniv Shaked and Avishai Wool, Tel Aviv University

This paper describes the implementation of an attack on the Bluetooth security mechanism and received considerable press coverage prior to MobiSys.

Shaked presented results which show that a four-digit PIN (the PIN size on many commodity devices) can be cracked in less than 0.3 sec. on an old Pentium III 450MHz computer, and in 0.06 sec. on a Pentium IV 3GHz HT computer. Because the attack uses brute force, the time to crack larger PINs increases by a factor of 10 for each additional digit used in the PIN.

The second contribution of this paper is the re-pairing attack, which forces two Bluetooth devices to rerun the pairing process that had just been described. Because devices usually store the link keys indefinitely, the first attack only works during the short interval when devices first connect to each other. However, an attacker can simulate a lost key by impersonating one device and telling the other device that it has forgotten its key. This action causes the two devices to run the full pairing process the next time they communicate, thus opening the devices to the attack. Shaked suspects that custom hardware would be required, because it may be necessary to spoof the Bluetooth ID and because the timing of the attack is critical.

Shaked concluded the talk by suggesting possible countermeasures to this attack. One consequence of the re-pairing attack is that the user is asked to re-enter the PIN. Users should be wary of such requests. Users should also enter their PINs as infrequently as possible to reduce the risk of an attacker eavesdropping on the pairing process. He also suggests that the hardware manufacturers use the 128-bit PINs that are allowed in the standard to make the brute-force attack less likely to succeed. Even when using a 128-bit PIN, though, a denial of service attack could still be performed by constantly running the re-pairing attack.

- *Shake Them Up! A Movement-Based Pairing Protocol for CPU-Constrained Devices*

Claude Castelluccia, INRIA, France and University of California, Irvine; Pars Mutaf, INRIA, France

This talk complemented the Bluetooth cracking talk. Claude Castelluccia showed how secure authentication between sensor devices could be performed across a clear channel without using a PIN and without using anything as CPU-intensive as a public key. Other goals were: no preconfiguration, no extra cost, no special equipment, and protection from denial of service attacks.

The strategy assumes that an eavesdropper cannot tell the source of a message (source anonymity). Assuming you have two devices, Alice and Bob, Alice can send a message to Bob that says, "I'm Alice." Bob knows this is true, since he's not Alice, so he and Alice can agree that the bit is 1. If Alice instead had said, "I'm Bob," Bob would know that this is false, so they would both agree on the bit being 0. Eve hears these messages, but does not know who was the source and therefore cannot determine the values of the bits.

The challenge is to construct an environment where Eve cannot determine the source of a message. In this talk, Castelluccia discussed three approaches Eve could take to discover the source of a message. She could use timing information, signal power, or frequency. He then described ways that Alice and Bob could protect against each of these techniques, one of which required physically "shaking them up."

Someone suggested that an attacker could intentionally create a collision when Alice sends a message to Bob, so Alice and Bob would not agree. Castelluccia pointed out that this would not work, since 802.11 is reliable: Alice would expect an ACK from Bob. Another questioner wondered whether cameras could be used to record the positions of the devices as they were being shaken. The question was mostly asked in jest, but Castelluccia pointed out that, even if such an attack were feasible, the devices could be obscured from view while being shaken.

MOBILE SERVICES

Summarized by Mike Blackstock

- *Reincarnating PCs with Portable SoulPads*

Ramón Cáceres, Casey Carter, Chandra Narayanaswami, and Mandayam Raghunath, IBM T.J. Watson Research Center

Awarded Best Paper!

Mandayam Raghunath presented SoulPad, a new approach to solving the problem of providing a personal and customized computing environment anywhere. When users move locations, they want to suspend their work and then start up again exactly where they left off. In the SoulPad solution, the PC (body) is separated from the session state and bits (soul). SoulPad uses a pocket-sized USB drive that can be plugged into any PC to allow users to resume their personalized computing environment. The key enablers for SoulPad are fast, small, high-capacity USB 2.0 drives from mass market media players, auto-configuring OSes such as Knoppix, and mature virtualization technologies.

Although the virtual machine and swap space use an encrypted file system, SoulPad is still currently vulnerable to hardware and BIOS attacks. To address reliability, the system supports network backups, though local backup is also possible. To deal with hardware diversity, the system needs to keep many drivers up to date. Some older systems cannot boot from a USB dongle, so a small boot CD could be required.

Raghunath was asked whether there are other issues that need to be addressed to put SoulPads to use. One issue is that certain applications check the hardware and assume that they are pirated when they have been moved to a new PC. There are also legal issues that need to be addressed.

Further information is available at http://www.research.ibm.com/WearableComputing/SoulPad/soulpad.html.

- ***Slingshot: Deploying Stateful Services in Wireless Hotspots***

Ya-Yunn Su and Jason Flinn, University of Michigan

Ya-Yunn Su presented the Slingshot system, which alleviates the bottleneck associated with the back-haul connection to hotspots by replicating application state to surrogate computers closer to wireless access points. She presented the scenario where a user brings a Pocket PC into a coffee shop and uses VNC to execute applications on a server over the Internet. The bandwidth over the Internet is often limited to a T1 (1.5Mbps) and latency is high, leading to poor response to desktop interaction. In a manner similar to cyber-foraging, Slingshot replicates state from the remote (home) computing environment to a VM running on one or more surrogates. Because Slingshot only replicates the state, users can fall back to their home server in the event of surrogate failure. Once a replica is instantiated on one or more surrogates, a proxy broadcasts user interaction to all replicas and the home server. The applications tested are the VNC desktop and speech recognition, though Su only discussed the VNC application in her presentation.

The system was evaluated to show that once the state was transferred, taking 27 minutes, the interaction performance was 2.6 times faster. Using a microdrive to store volatile state and chunk hashes sped things up considerably, requiring only 6 minutes: about 3 minutes to transfer state, another 3 to replay the logs. Overall the system improves performance for low-latency applications and hides surrogate failures by using replicas and broadcasting the interactions.

Ramón Cáceres asked if the system relied on the applications running in a deterministic manner; for example, could a Slingshot application maintain communications from the outside world? Su responded that determinism is required and that outside communication is not yet supported. Another member of the audience asked whether requiring VMware on all surrogates might perhaps be too strong an assumption. Su responded that they do assume that the surrogate is already running VMware, but suggested that perhaps VMware itself could be transferred first in the future.

- ***DeltaCast: Efficient File Reconciliation in Wireless Broadcast Systems***

Julian Chesterfield, University of Cambridge; Pablo Rodriguez, Microsoft Research

Julian Chesterfield presented the DeltaCast system, which very efficiently reconciles two versions of a file between a server and any number of clients with any version of a file using a pure radio broadcast system. DeltaCast uses a hierarchical hashing scheme, combined with decomposable hashes and erasure coding for high efficiency. DeltaCast was compared with file download, flat hash, hierarchical hash schemes using Web pages, and binary data such as software upgrades. From this evaluation, it was determined that the number of hierarchy levels could be dynamic depending on the data type. Compared to hierarchical and single-layer hash systems, the time required to get the data required to update a file on a client is also much lower for Web pages. The amount of data downloaded is the same as in hierarchical hash schemes and less than in single-layer hashing. DeltaCast trades off decoding time, but the overall penalty is low. This system can be applied to not only broadcast radio networks but also IP multicast, overlay networks, and content distribution networks.

One attendee asked why the authors did not consider the use of Turbocodes or other well-known hash algorithms. Apparently these hashes do not require the system to solve linear equations, but use iterative algorithms, so they should be faster. Chesterfield answered that they used a hash that was already available to them and understand that perhaps their system may not be optimal in this area. Maria Ebling asked whether the carousel size of hashes or data on a given channel was fixed. Chesterfield answered that any number of erasure codes can be generated and any of these blocks/codes is useful in regenerating data and hash codes.

POSTERS AND DEMONSTRATIONS

Summarized by Denitsa Tilkidjieva

This session hosted 24 displays of researchers' work in the field of mobile systems, applications, and services. A wide range of applications were on display. We highlight just three of these here:

The pre-hospital patient care system (Hashmi et al.) consisted of a number of pulse sensors, attached to patients' fingers to monitor vital signs. It is most useful when multiple patients are in need of attention, because it can allow quick decisions at critical moments.

The inHand system for ubiquitous personalized interactive content (Bhatti et al.) was shown in both a poster and a demonstration. The researchers demonstrated the inHand device and how it can be used to gather user-customizable information about movies, events, and the like.

If you try to cook a turkey, a duck, and a chicken all in one, you will get a Turducken. Sorber and his colleagues borrowed this name for their system for hierarchical power management, consisting of a laptop, a stripped-down PDA, and a mote. The device always chooses the platform that performed the given task at the lowest energy cost, thus extending the lifetime of the device up to 10 times.

Photos of the posters and demonstrations can be found at http://www.mtholyoke.edu/~dntilkid/mobisys.

PLENARY SESSION

Summarized by Hongwei Zhang

- *Staying Off the Hot Seat with Cool Mobile Systems*

Alfred Spector, IBM Software

Alfred Spector, chief technology officer at IBM Software, gave a visionary talk on the technological and societal implications of the development of mobile pervasive systems, with an emphasis on the robustness of these systems.

Spector first described the trend of mobile systems. On one hand, the ever-growing modality, decreasing form factor, declining cost, and exploding connectivity have been pushing the development of mobile systems. On the other hand, medical informatics and security applications are calling for mobile systems. For instance, the market for health care is about $1.5 trillion per year, which is about as large as the whole IT industry. Nevertheless, in most scenarios of mobile pervasive systems, we are envisioning amalgams of components. The systems are usually so complex that we cannot prove their correctness. In many cases, it is also impossible to anticipate what may go wrong in a system.

Given the complexity of pervasive systems, Spector argued that one key challenge of designing these systems is to guarantee their robustness, e.g., ease of use, evolution, QoS, reliability, security, and fitness to purpose.

After discussing the challenges posed by mobile pervasive systems, Spector analyzed why existing techniques and architectures do not satisfy the need for robustness. To demonstrate design for robustness, Spector described a hierarchical architecture based on a common trusted computing base, a secure hypervisor, and trusted virtual domains. Spector stressed the importance of adopting a top-down approach and the development of trustworthy capabilities (e.g., attestation, privacy services, and authentication).

In particular, Spector stressed the concept of "information provenance (InfoP)," by which the origin of information can be identified with proof. InfoP can provide the basis for law enforcement to play their role in improving the robustness of systems (i.e., punishing actions that are not allowed, such as spam, viruses, etc). But Spector also pointed out the challenges of InfoP: privacy, storage for massive data, digital signatures and CAs, and law enforcement across international boundaries.

Spector discussed the importance of the "science of design" in increasing system robustness. Besides the traditional time and space complexity, Spector specifically noted the need to manage the complexity of usage (i.e., user interface). To this end, he argued that we should pay attention to the meaning and measurement of robustness and of design methodologies. He also emphasized the importance of simple yet flexible architectures, as well as self-healing and self-optimization. Finally, he suggested that the technical community should work with the wider society to address a broader range of robustness issues.

SPEEDY WIRELESS

Summarized by Xiaoqiao (George) Meng

- *Improving TCP Performance over Wireless Networks with Collaborative Multi-Homed Mobile Hosts*

Kyu-Han Kim and Kang G. Shin, University of Michigan

Kyu-Han Kim pointed out that wireless networks have capacity limitations. In practice, a group of nearby hosts may constitute a mobile collaborative community (MC2) where hosts share bandwidth and content. In such a community, each host is multi-homed and data is multiplexed to improve utilization. Accordingly, it is important to allow TCP to achieve high utilization of the aggregate bandwidth over multiple interfaces, but this presents several challenges, from requiring exact link-state information, to coping with dynamic wireless links, to handling out-of-order packet delivery, to controlling congestion.

To cope with these challenges, Kim introduced PRISM, Proxy-based Inverse Multiplexer. PRISM consists of adaptive scheduling, intelligent acknowledgment controlling, network-assisted fast recovery, and IMUX at the proxy's network layer. PRISM architecture has an adaptive scheduler (ADAS) to achieve cheap and adaptive fair-scheduling. PRISM has been implemented in Linux kernel 2.4.20 and netfilter. PRISM-IMUX is a filter at a network layer. The authors also devised a testbed, by which they found that PRISM delivers 95% of the aggregated bandwidth. In a setting with three heterogeneous mobile nodes, PRISM achieved more bandwidth than vanilla TCP.

David Kotz asked how members of the community find one another. Kim answered that they can use the service location protocol to find existing collaborative communities, but that forming communities and dealing with membership dynamics is an area for future work.

■ ***Horde: Separating Network Striping Policy from Mechanism***

Asfandyar Qureshi and John Guttag, MIT Computer Science and AI Laboratory

Asfandyar Qureshi presented Horde, a middleware mechanism that allows multi-stream applications to communicate over multiple channels with widely varying latency and bandwidth. The authors were motivated to build Horde in order to support mobile telemedicine, specifically an application that allows doctors to examine patients in transit to the ER. This application requires the transmission of unidirectional video, bidirectional audio, and low-rate physiological data streams in real time from a moving ambulance. This system requires high throughput, low latency, and the ability to deal with vehicular motion in an urban area.

Network striping in a WWAN environment presents substantial challenges, from coping with limited bandwidth, to dealing with applications having dissimilar needs, to dealing with dissimilar network channels with varying QoSes.

Horde middleware provides a number of services, the most novel of which is QoS modulation. With QoS modulations, applications express stream QoS sensitivities, where the QoS is multi-dimensional. When an application sends data, it receives some utility from the consumption of its data at another host. Applications express QoS "objectives" which define QoS constraints on streams. In summary, the goal for Horde is to build a flexible network striping middleware for WWANs.

David Kotz asked how many applications the authors have examined and how many more they plan to explore. Qureshi expressed concern that the interface may be too rich, in that you may be able to express more than is necessary. They are building the telemedicine application to gain more experience with the system.

■ ***An Overlay MAC Layer for 802.11 Networks***

Ananth Rao and Ion Stoica, University of California, Berkeley

Multi-hop wireless networks are being considered for last-mile broadband connectivity. However, such networks are subject to issues of fairness. This work addresses this issue; specifically, the authors show how to prevent starvation of flows without changing the hardware.

Ananth Rao first described a starvation problem with 802.11 MAC that they identified by using a testbed with six nodes. They found that the cause of this problem was interference and asymmetric carrier sense. Rao pointed out that existing solutions to this problem (e.g., contention-based MACs and TDMA) require hardware modifications but that the starvation problem can also be solved above the MAC layer by limiting how much data each host can send. The proposed solution is called Overlay MAC Layer (OML). OML uses readily available, inexpensive hardware, and can evolve to meet diverse application scenarios/requirements. Rao then described how to implement OML. He summarized the results of their evaluation on both a six-node testbed (based on Click from MIT) and a Qualnet network simulator. They found that the disparity between one-hop and four-hop flows was reduced but that the throughput on the one-hop flows was also greatly reduced.

Himanshu Raj asked whether providing fairness reduced the overall throughput. Rao explained that this approach actually increases the throughput, though it is not shown in the graph because the graph does not account for starved flows. Ramón Cáceres pointed out that with an asymmetric link, one node would never be in the active list, but Rao clarified that a node in the middle can piggyback information about the other node so that it will be added to the active list. Richard Paine encouraged Rao to propose some of these changes to 802.11 because, although 802.11n addresses some of these issues, it would be useful to address the other questions as well. Brian Noble asked what happens if one of the nodes gets greedy, and Rao explained that they currently assume that the box is tamper-proof.

OPERATING SYSTEMS FOR SENSOR NETWORKS

Summarized by David Johnson

■ ***Design and Implementation of a Single-System Image Operating System for Ad Hoc Networks***

Hongzhou Liu, Tom Roeder, Kevin Walsh, Rimon Barr, and Emin Gün Sirer, Cornell University

Hongzhou Liu presented MagnetOS, a distributed operating system designed for use in ad hoc networks. Liu observed that ad hoc network applications are extremely difficult to program, even today. MagnetOS responds to this problem by combining all network nodes into a single, event-based virtual machine; this abstraction eases application development and increases network lifetime. In MagnetOS, synchronous and asynchronous events signal code execution by triggering event handlers. A static partitioning approach converts application class files into event handlers. Migration of event handlers provides improved energy efficiency and saving of computation if loss of power is imminent. MagnetOS provides several migration algorithms that are designed to minimize communication energy overhead.

Liu presented results from evaluating MagnetOS with an application called SenseNet, in which there are a number of fixed sensors and mobile data processing components. They compared a number of algorithms, three that required no communication overhead and two that did. The TopoCenter(1) algorithm, which moves objects using one-hop neighborhood knowledge from each communicating partner, increased system lifetime by a factor of 2.5.

Ahmad Al-Hammouri asked why they did not use aglets, since aglets provide a clean environment for mobile agents. Liu explained that the major complaint about mobile agents is security, because they can execute any code on any node. Doug Terry added that MagnetOS is using a different model; they are breaking their code onto different machines. Terry then asked whether they thought they would need multiple algorithms and adapt between them dynamically or whether one would be sufficient. Liu responded that they currently require the programmer to pick one and that their experience to date suggests that the two Topo algorithms perform well throughout the range.

- ***SOS: A Dynamic Operating System for Sensor Nodes***

Chih-Chieh Han, Ram Kumar, Roy Shea, Eddie Kohler, and Mani Srivastava, University of California, Los Angeles

Ram Kumar presented SOS, an operating system designed for use in sensor network applications. Because sensor networks often manifest in long-term deployments, individual nodes must be flexible to respond to remotely controlled changes. SOS is a modular, application-independent operating system that supports dynamic reprogramming via module updates and replacements. In contrast, TinyOS, the de facto sensor network operating system, produces a static binary composed of both system and application-level functionality and must be recompiled and replaced on each node to effect an upgrade. Another similar system, Maté, provides a virtual machine that can execute small code fragments distributed through the network. Application-level upgrades are possible, but interpreter upgrades must fall back on the TinyOS update system. SOS consists of a static kernel, which provides an abstraction of the hardware, and is installed on all nodes. Dynamically linked modules communicate with various kernel services and device drivers as necessary. Modules can register functions with the kernel, and potential callers may subscribe to provided functions. The kernel provides priority scheduling, dynamic memory and intra-module message passing, as well as safety features.

Kumar discussed the results of an evaluation comparing TinyOS, SOS, and Maté. All ran Surge-like applications (Surge is a well-known multi-hop data-gathering application for TinyOS). They found that CPU activity was on average 1% higher in SOS than in TinyOS. Energy usage during code updates in SOS was an order of magnitude larger than in Maté, because Maté must update only a small chunk of bytecode. However, TinyOS update costs were 400 times as much as for SOS, because updates to TinyOS require the replacement of the entire system binary. Kumar observed that the important metric to observe when comparing update energy costs is the frequency of updates.

Himanshu Raj asked what impact one module can have on another and whether they can crash one another. Kumar responded that the architecture provides no form of memory protection and that a wild pointer could corrupt the data space of another module. Unfortunately, there is no hardware support to correct this problem. Bhanu Pisupati asked about the programming model. Kumar responded that it is described in more detail in the paper, but that all modules are implemented as message handlers and listen on a particular port for messages intended for them. Jason Flinn found the idea of safety checks fairly interesting, but asked for clarification about how this really works. Kumar responded that static checks would require analyzing the entire source code and would not solve the problem. He clarified that the base stations maintain some information about the kind of modules present on the nodes and do some analysis at load time before sending a module.

LOCATION (HERE)

Summarized by Neil McCurdy

- ***A Relative Positioning System for Co-Located Mobile Devices***

Mike Hazas, Christian Kray, Hans Gellersen, Henoc Agbota, and Gerd Kortuem, Lancaster University, U.K.; Albert Krohn, University of Karlsruhe, Germany

Mike Hazas introduced Relate, a compelling approach to determining fine-grained relative locations and orientations between proximate devices. It uses ultrasound peer-to-peer sensing, giving the user relative location accuracy in the 10cm range and eliminating the need for the infrastructure-based location support that is typically needed for such fine-grained accuracy. The Relate team created their own hardware—a USB dongle that has three ultrasound transmitters and sensors. Time of flight of the ultrasound signal determines the range, and the signal strength, as recorded by each of the three sensors, determines the angle of arrival.

To evaluate the system, Hazas described an experiment in which five laptops were placed at various positions on a 2.4 x 1.6m surface. One hundred iterations of this experiment were performed, with half of them ensuring that the dongles had line-of-sight to one another. With good line-of-sight, one can expect roughly 9 cm, 33° accuracy. With limited line of sight, one can expect approximately 11cm, 48° accuracy. Hazas closed by discussing some of the issues with this approach. First, he posited that hardware that is better equipped to do signal processing may be able to get the accuracy to as low as 2 or 3cm. He also pointed out that there is a limit to the number of devices that can be handled, since only one device can communicate at a time.

The Relate system was demonstrated on Monday night, and Hazas was asked how his system differed from Cricket. Relate dongles have three transducers, are optimized for co-planar calculations, and can calculate the angle of arrival to determine the orientation of the devices. The Relate system also does not require any calibration.

One questioner pointed out that location accuracy is more important as the devices get closer to one another. For example, 10cm accuracy may not be adequate when the devices are only 2cm apart. Hazas agreed. There was also a question about whether the signal processing could be done in software. Hazas did not see any reason why not.

- ***WALRUS: Wireless Acoustic Location with Room-Level Resolution Using Ultrasound***

Gaetano Borriello, University of Washington and Intel Research; Alan Liu, Tony Offer, and Christopher Palistrant, University of Washington; Richard Sharp, Intel Research Cambridge, UK

Gaetano Borriello presented a system that provides room-level granularity by using a combination of wireless and ultrasound. Borriello began by explaining the importance of room-level localization to usability. The goals of the system included low deployment cost, low support cost, and a system that was incrementally useful and deployable. The system should also approach 100% accuracy while maintaining privacy.

Borriello described WALRUS as being inspired by lightning and thunder, with WiFi (through an access point broadcast of information) acting as the lightning and ultrasound (through commodity speakers) acting as the thunder. Clients receiving the WiFi packet keep the packet only if they also detect the sound. The system was tested in two environments under many different conditions (music playing, conversations ongoing, keys jangling, doors slammed, etc.), and the system proved largely immune to extraneous noise. Borriello concluded by discussing some of the limitations in the existing implementation and presenting a vision of future prototypes that might one day be located in a store near you: a wristwatch receiving device, with light bulb and air freshener transmitting devices.

Lin Zhong asked whether they had considered listening to the ultrasound signal first and then the WiFi. Borriello responded that this approach might be perfectly reasonable, but that they haven't tried it. Ed Nightingale asked whether they had considered rooms that were changing. Borriello responded that you could put arbitrary information in the packets. David Kotz commented that one of the unique characteristics of this work was that they were using existing devices. He then asked why they decided to move toward custom hardware. Borriello replied that one thing to consider is how much optimization each device could have. Robert Hall asked whether they had considered the health implications of ultrasound. Borriello said that they are not boosting the speaker output and that one of the reasons to move toward custom hardware was to move further away from the audible range. Another member of the audience asked whether they had considered placing microphones in the room to measure the volume levels and then including those volume levels in the WiFi packets so that clients would know what volume to expect. Borriello responded that they had walked around the room and measured the volume levels, but they found that this was not important because their detection algorithm relied on a relative measure of strength.

WORK-IN-PROGRESS REPORTS (WIPS)

Summarized by Ya-Yunn Su

- ***CAM: Architecture for Automating Paper-Based Processes in the Developing World***

Tapan Parikh, University of Washington

Tapan Parikh said that in developing countries paper-based processes are inefficient, but cell phone penetration is growing, and most cell phones have built-in cameras. He proposed using such mobile phones to digitize manual, paper-driven processes. A user could take a picture of a paper document, transfer the data to a remote machine, and propagate it to the appropriate recipient. The receiver could then print the document.

http://www.cs.washington.edu/homes/tapan

- ***Smart Attire–The Digital Diary***

  Tarek Abdelzaher, University of Virginia

  The author proposed building sensors into clothing to record user activity. The smart attire includes accelerometers, magnetometers, and temperature, sound, light, and GPS sensors. Possible applications are personal digital diaries, and medical monitors. The prototype system is a winter jacket with five motes and a GPS tracking device. The motes on the jacket record activities and upload the data when near an access station using 802.15.4. One example shows that the author can infer the user's activity (e.g., typing, walking) from the data collected by accelerometers. Another example shows the user's path or stillness by data from the GPS device. The user may use data mining techniques to understand personal life patterns (e.g., Is my social life declining?) and query the history to keep track of health or financial activities (e.g., When is the last time I visited my dentist?). The prototype system shows that power management is important for smart attire.

- ***Extracting a Mobility Model from Network Traces***

  Minkyong Kim, Dartmouth College

  Researchers need a realistic mobility model to simulate the effectiveness of new algorithms on wireless networks. There are two ways to generate network traces: syslog and GPS. Syslog data collected on access points contains client events, including time and action. It has the advantage of availability of large data sets, but it is often hard to estimate a user's location from an AP location, perhaps due to the device not being associated to the closest AP or perhaps because of some device-specific implementation. An alternative is GPS. Unfortunately, there is no GPS data set publicly available, and it does not work indoors. To address these limitations, Kim extracts the user's path from time and AP location from syslog data, extracts mobility characteristics from the user's path, and extracts speed/duration/pause to generate the mobility model.

- ***Emulab Unleashed! Into the WiFi and Mobile Wireless Dimensions***

  David Johnson, University of Utah

  David Johnson presented Emulab, a network emulation testbed. Due to the complex nature of wireless networks, simulation is not enough in testing and validating new research ideas. A real testbed like Emulab would be a better way. Emulab has added three wireless testbeds including a building-scale WiFi testbed, a fixed sensor net, and a mobile WiFi network. The WiFi testbed enables remote hardware reset functionality, since it is difficult to find the real location and reboot the machine manually. There are also fully programmable motes and mobile robots with a tracking system accurate to 1cm. Emulab can be remotely accessed and allows multiple users. It provides a realistic and repeatable wireless environment. Future work includes automated rechargeable stations and power monitoring. More information can be found at http://www.emulab.net.

- ***Content Management for Mobile and Pervasive Experiences***

  Nigel Davies, Lancaster University

  Nigel Davies emphasized the importance of content in a ubicom/pervasive computing environment based on experience in GUIDE deployment. Their solution was to assign many students to work on content and the user interface. Other projects, such as Can You See Me Now? (http://www.canyouseemenow.co.uk) and Equator (http://www.equator.ac.uk/) reached similar conclusions. In the e-Campus project, content can come from users, be automatically generated, etc. The challenge is how to manage multimedia content in future mobile and ubiquitous computing environments. More information on the GUIDE project is located at http://www.guide.lancs.ac.uk/.

- ***Invisible Agents***

  Nobuo Kawaguchi and Negishi Yuuya, Nagoya University

  Nobuo Kawaguchi pointed out that in a world with many computers (PC, laptop, PDAs, etc.) and smart appliances (music players, TiVo), we do not have a good interface to control and aggregate the interfaces of all the devices. They built invisible agents to solve this problem. The example is a conference room with a projector, multiple monitors, ceiling lights, and similar equipment. They can combine brightness with a human sensor to control the ceiling light, or a human sensor to activate the projector. On each of the target devices, they run a VNC server. The master device connects to all the target devices.

- ***A Social Networking Web Site for the Research Community***

  James Scott and Richard Sharp, Intel Cambridge

  James Scott said that the population of researchers is huge and growing. There are many types of relationships between researchers (e.g., as co-authors, work colleagues, conference attendees), but communities that overlap might also be isolated (e.g., SIGMOBILE, Pervasive/UbiComp, SIGCOMM). One useful feature of a social networking Web site would be an easy home-page generator containing basic information (e.g., bio, publications, photos). To prevent spamming, the Web site could be semi-exclusive or could follow Gmail's invitation-based model. The Web site would encourage people to register and verify their information.

- ***Crawdad—A Community Resource for Archiving Wireless Data at Dartmouth***

  David Kotz, Dartmouth College

  David Kotz pointed out that there is little real wireless network traffic available for researchers. They initiated Crawdad as a facility for storing data sets collected from real wireless networks. They already had a campuswide wireless infrastructure for collecting traces. The challenges include developing a common data format, importing existing data, and anonymizing data. They hope to coordinate with other communities to develop network trace formats and tools. The tools and data can also be used in course projects. More information is available at http://crawdad.cs.dartmouth.edu/.

- ***Secure Mobile Architecture***

  Richard Paine, Boeing Technology

  Richard Paine proposed a secure mobile architecture that can cryptographically identify each packet. They can support mobility by transparently changing the address for the user and application. This framework improves their enterprise network by reducing operational cost and complexity. They use four techniques to achieve their goal:

  - 1. PKI: Public key infrastructure. Each client uses his/her badge for client authentication.
  - 2. HIP: Host mobility protocol. Communications are based on IPSec, therefore each host is identified by a security parameter index (SPI) rather than IP. Each host is further identified by a host identity tag (HIT), which is SHA-1 of the public key.
  - 3. NDS: Network directory services. The client goes through an identification process before using LDAP.
  - 4. LENS: Location-enabled network services: see http://www.opengroup.org/bookstore/catalog/select.tpl?text=secure+mobile+architecture.

- ***GSM War Drive***

  Mike Chen, Intel Seattle

  Mike Chen presented their goal of providing a playground for location-based services. Cell phones are the location devices people already carry every day. Some current location technologies are GPS and WiFi. GPS devices are not accurate and have limited coverage. WiFi can only be used on limited devices. Based on calculations from GSM tower signal strengths, the cell phone can provide accuracy within 30cm outdoors and 4m accuracy with a 1m grid. They plan to make the trace publicly available.

LOCATION (THERE)

Summarized by Mike Blackstock

- ***The Horus WLAN Location Determination System***

  Moustafa Youssef and Ashok Agrawala, University of Maryland

  Moustafa Youssef presented the Horus system used to determine indoor locations to an accuracy of less than 2m by using existing wireless LAN infrastructure. Horus, like other WLAN-based locating systems, uses APs as reference points and the observed signal strength to these APs to estimate distance via triangulation. However, when indoors, the observed signal strength readers can differ by 15dBm for a given distance. Like Radar, Horus uses a radio map to characterize the area to counter these effects. Unlike Radar, however, Horus is a probabilistic system rather than deterministic.

  The goals of this system were high accuracy, low computational requirements, energy efficiency, and scalability (both in number of users and in the area covered). The Horus techniques accounted for a 25% reduction in the average distance error. The Horus system is shown to have higher accuracy on average than Radar by more than 82% and better than 27% for the probabilistic system, and is more computationally efficient by an order of magnitude. The authors then applied the Horus ideas in Radar and showed that these ideas could reduce Radar's average distance error by more than 58% and decrease the worst-case error by more than 78%.

  Robert Harle asked what the test environments were like. Youssef indicated that it was a typical CS department with offline measurements taken at night and used during the day to capture environment variations in a typical deployment. Mark Corner asked how the new dynamic power control feature would affect location determination using the new WLAN systems. Youssef thought that their ongoing work in automatically generating the radio maps based on environment changes may be effective in dealing with this problem.

- ***Deploying and Evaluating a Location-Aware System***

  R.K. Harle and A. Hopper, University of Cambridge, U.K.

  Robert Harle presented their experience with deploying and using the Active Bat system at Cambridge. The Bat system is accurate to 3 to 5cm in three dimensions. The data on which this study is based was collected over a period of more than two years from a deployment that covers about 500 square meters in their new building.

  Harle found that the killer application for this type of system was allowing companies to learn more about how office building space is used, to encourage people to work more effectively. Surveys showed that privacy was not an issue, but that result may not extend beyond small communities such as theirs. A more meaningful study of such systems would require deployment in a corporate office or perhaps a hospital where there is more overlap in working hours and collaboration is more common.

  James Scott thought it was interesting that the corporate space usage was the best application rather than an application that benefited end users. Harle indicated that in fact one company expected to get a return on their investment in one year of data collection, without using any of the software that could benefit the end user. Further, they were not interested in deploying end-user applications until they had reaped that ROI. Minkyong Kim asked whether providing more information to the end user regarding the accuracy of a location reading would increase system trust. Harle indicated that they experimented with a five-bar system ranking and found it useful, but that this ranking system did not help this issue significantly. Guanling Chen asked whether the high accuracy of the Bat system was necessary for the applications in this deployment. Harle indicated that although it was not needed for this type of application, there are classes of applications, such as those that use position clicking and pointing, that require it. He agreed that the most important thing is highly accurate room-level granularity.

- ***Accuracy Characterization for Metropolitan-Scale WiFi Localization***

Yu-Chung Cheng, Intel Research Seattle and University of California, San Diego; Yatin Chawathe and Anthony LaMarca, Intel Research Seattle; John Krumm, Microsoft Research

While an intern at Intel Research, Yu-Chung Cheng and his colleagues worked to characterize the accuracy of the Place Lab WLAN location determination systems for use outdoors. Unlike GPS, their system works in urban canyons and indoor environments, and it relies on more ubiquitous technology (check out the density map for Manhattan at http://www.wigle.net). Unlike other WLAN systems, their system requires less configuration time (1km^2 area/1 hour) using war driving. Although it is consequently less accurate (13–40m), this is not an issue for applications such as a location-based Web search.

For their experiment they gathered data in three neighborhoods—downtown Seattle, an urban residential area, and an area in Kirkland (a less dense suburb where homes are 15 to 20 feet apart)—and compared their location estimates with GPS readings.

They then applied three different algorithms. Their baseline tests found that the specific algorithm used did not matter much, though fingerprints performed poorly with only one AP. Interestingly, errors were higher in the more dense downtown area, probably owing to the fact that many APs are higher up and not contributing to making measurements more accurate. They concluded that it is possible to get acceptable accuracy of 13–20m in high-density areas, and around 40m in lower-density areas, with about 30 to 60 minutes of calibration for the neighborhood.

David Kotz asked about the data corresponding to Figure 4, where the Y axis is labeled "% of Time"; Cheng clarified that it actually represents "% of records." Another member of the audience was asked whether the urban results were affected by the GPS noise in urban canyons. Cheng said that they countered this possible effect by checking that readings were correlated by three GPS units and only using those that were consistent. David Kotz then asked whether it would be possible to improve accuracy with the indoor Horus techniques. Youssef and Cheng discussed this possibility offline. Youssef later reported that they concluded that the Horus techniques would be useful when there is more information available for the localization algorithm. For example, when the number of APs per scan increases, Horus techniques can give a significant advantage. However, where you have just one AP per scan, there is not enough information to notice a difference between the techniques.

The traces and source code for this paper are available at http://www.placelab.org/.

MORE POWER TO YOU

Summarized by Ram Kumar Rengaswamy

- ***Energy Efficiency of Handheld Computer Interfaces: Limits, Characterization, and Practice***

Lin Zhong and Niraj K. Jha, Princeton University

Lin Zhong noted that the role of the user interface has often been ignored in the design of energy-efficient systems. The speed of the human interaction is significantly lower than the speed of the computer. The computer ends up spending significant time waiting for user inputs. The energy consumed by the computer in these idle periods can be eliminated through better system design.

Zhong and Jha compared two forms of user input, speech and handwriting, using energy efficiency as the metric. Energy efficiency is a combination of the speed of the input and its power efficiency. The experiments showed that speech is more energy efficient than handwriting. Zhong also designed a wireless wrist-watch to be used as a low-power, low-cost cache device. The cache device is a slave to the main computer. It collects user input while the main computer is put to sleep. This results in system-level power savings.

The results evoked a lot of interest from the audience. Brian Noble raised an interesting point about the usage scenarios of the input techniques. Listening and writing are concurrent operations while speaking and listening are not. Hence, from a holistic viewpoint, it might be more energy efficient to listen and write than to listen and speak.

- ***Turducken: Hierarchical Power Management for Mobile Devices***

  Jacob Sorber, Nilanjan Banerjee, and Mark D. Corner, University of Massachusetts; Sami Rollins, Mount Holyoke College

  Jacob Sorber explained that the main principle of hierarchical power management is to pick the most energy-efficient component of the system for a task. The challenge is in partitioning a given task into a set of subtasks and assigning each to the most efficient component. Such an approach automatically maximizes system lifetime. It is desirable to have minimum user intervention in such a system.

  The authors developed the Turducken system for hierarchical power management in laptops. Turducken consists of a laptop attached to a PDA and a mote sensor node. The role of the mote is to maintain clock synchronization with a time server. The laptop and the PDA derive their clock from the mote upon their wakeup. The PDA is responsible for caching Web pages and waking up the laptop to display the pages once they are fully loaded. The user interacts directly with the laptop. The laptop responds to user queries (e.g., email retrieval). Turducken significantly lowers average power consumption compared to conventional systems.

  The audience provided some very interesting comments and suggestions. The power supply design of such a system was discussed. It was concluded that it would be most energy efficient to have separate batteries for each system component. Usage of the lower-tier components during the transition interval of a high-tier device from one state to another was considered.

HotOS X: Tenth Workshop on Hot Topics in Operating Systems

Sante Fe, New Mexico
June 12–15, 2005

RELIGIOUS WARS

Summarized by Alexandra Fedorova

- ***Are Virtual Machine Monitors Microkernels Done Right?***

  Steven Hand, Andrew Warfield, Keir Fraser, Evangelos Kotsovinos, and Dan Magenheimer, University of Cambridge Computer Laboratory and HP Labs

  Steven Hard argued that microkernels and virtual machine monitors (VMMs) both emerged to achieve isolation of the software system from the underlying hardware, but used different means to do so. He highlighted similarities and differences among VMMs and microkernels, talked about architectural lessons learned with these systems, and suggested that it is best to design architectures that borrow the best from VMMs and microkernels, as opposed to sticking to a particular architecture.

  He summarized the differences as follows: Whereas VMMs multiplex entire operating systems, microkernels multiplex many small tasks as threads. VMMs are closely aligned with hardware, so the interface looks like hardware; microkernels expose a higher-level interface, and tasks communicate using synchronous IPC. VMMs offer one address space per scheduled entity; microkernels offer multiple scheduled entities per address space. Architectural lessons learned from research with these systems are:

  - Avoid liability inversion. Moving trusted system code to the user level, as is done in microkernels, involves having to trust user code to perform essential system functions (e.g., user-level page in Mach).
  - Make synchronous IPC irrelevant. Synchronous IPC in microkernels is expensive. But as we learned from Xen, IPC does not need to be on a critical path.
  - OS is a reusable component. VMMs have achieved complete component reusability, because they treat OS as the reusable component.

  During the Q&A session, the most fire came from Gernot Heiser, who disagreed with the analysis. Gernot argued that liability inversion is not inherent to microkernels (in UNIX you have system daemons implemented as user processes that run in privileged mode). He also insisted that IPC in microkernels is not a problem: New fast hardware allows cheap implementation. Besides, Liedtke, using L4 as an example, demonstrated that microkernels can be fast.

- ***OS Verification—Now!***

  Harvey Tuch, Gerwin Klein, and Gernot Heiser, National ICT Australia

  Harvey Tuch argued that there is an urgent need to develop practical formal verification tools that can be used in high-performance industrial operating systems. This is a challenging task, and requires the right OS architecture: basically, a microkernel. VMMs won't work because they increase the size of the TCB.

  The rest of the talk was a survey of available formal verification methods. Formal verification is done by constructing a system model, a system specification, and a verification tool that checks for desired properties using the model and specification as input. Examples of formal specifications are HOL temporal and Bayer-Moore logic, microkernel APIs. Model checking is usually done by automatic reachable state-space exploration or by theorem proving. They have already implemented some preliminary verification tools for the L4 microkernel.

  During the Q&A, Aaron Brown pointed out that real customers are using OSes as giant application servers; they run databases and application servers on them, and it is not clear how much leverage you are going to get from verifying just the OS kernel. Several other attendees, including Eddie Kohler and Rik Farrow, asked how they were going to deal with changing operating systems. Harvey responded that there are two types of changes, implementation changes and API changes; they have a way of dealing with implementation changes, but API changes are more difficult.

- ***Making Events Less Slippery with eel***

Ryan Cunningham and Eddie Kohler, University of California, Los Angeles

This talk continued the old debate concerning threads and events. Events are fast but difficult to program. The speaker described eel, a programming tool designed to simplify programming with events. eel uses program analysis techniques to improve programmability while preserving the event model, and provides the libeel event library, visualization, and debugging tools.

Programming events is hard because it is difficult to understand the flow of the program and difficult to debug. eel's visualization tools make it easy to visualize the control flow, and the debugger allows you to step through each program flow separately; you follow the callbacks related to the same connection, so each flow of related events appears sequential. The speaker concluded that events don't need to be hard to program. You can use simple tools to extract program control flow and to help with visualization, verification, and debugging.

Margo Seltzer pointed out that while eel helps clarify the event-based program after it's written, it does not appear to help with the actual writing of the program. The speaker responded that by allowing the programmer to visualize newly written code, eel does make writing easier. Margo was not convinced that using events is worth all this trouble. The speaker said that using events is preferable because they are much faster. Pei Cao responded that based on her experience of designing large software projects people use both threads and events. Events are a pain, so people use them only when performance is absolutely critical, which does not happen very often.

STORAGE

Summarized by Steve VanDeBogart

- ***Parallax: Managing Storage for a Million Machines***

Andrew Warfield, Russ Ross, Keir Fraser, Christian Limpach, and Steven Hand, University of Cambridge Computer Laboratory

Andrew Warfield spoke about Parallax, a system for dealing with the increased demand on storage systems created by the use of virtual machines. With the adoption of virtual machines in cluster environments, the number of system images per disk has increased by a factor of 10 to 100. Some organizations expect that within the next few years they will be supporting one million live system images in a single data center. Furthermore, techniques that take advantage of historical versions of VM state (e.g., for intrusion detection or configuration debugging) imply additional demand on storage systems. Parallax is designed to solve these problems by being able to scale to many images while supporting fast and frequent snapshots.

The key insight behind Parallax is that block-level write sharing should not be on the critical path. The common case is that all active system images started from a small number of base images and then diverged. While it is possible that distinct systems have the same software added on top of the base image, this is not the major source of shared blocks. Parallax handles this by maintaining a radix tree of the blocks in the active image.

Armando Fox and Mary Baker asked about the fault tolerance of the system. Andrew responded that nodes are monitored for failure (both loss of connectivity and deadline failure) and that blocks are sent to at least three nodes to reduce the chance of losing information.

- ***Stupid File Systems Are Better***

Lex Stein, Harvard University

Lex Stein discussed some experiments he conducted to validate the file system speed tricks accumulated over the last 30 years. The classic assumption in file systems is that the closer the block numbers, the faster it will be to access blocks. However, with virtualized block numbers on modern storage systems, this is not necessarily the case. To determine how much the smart allocation of blocks affects performance on a modern storage system Lex removed the smartness by randomizing the block values in a trace.

Two traces were used to conduct tests, one taken during the compilation of a Linux kernel, the other under the Postmark benchmark. The block numbers in these traces were then randomly permuted. The modified and unmodified traces were played back in a disk-accurate simulator with a varying number of disks used as JBOD (Just a Bunch of Disks), RAID 4, and RAID 5. The simulation results show that with a modest number of disks the randomized traces started to perform better than the smart traces.

John DeTreville suggested that maybe the smart traces were trying to schedule the blocks into too short a time frame and thus not taking advantage of the parallel seek capacity of the multiple disks. Lex replied that maybe there is a point where things should get parallelized and that this might be a new trick to improve the performance of file systems. Russ Cox added that maybe instead of dumbing down the file system, since it still does help performance on a single disk, the volume manager should do a better job of distributing blocks. Petros Maniatis suggested adding traces that evenly distributed the work among the disks for comparison: maybe performance is improved, not by randomness per se, but through hot-spots eliminated in the random trace.

- ***Aggressive Prefetching: An Idea Whose Time Has Come***

  Athanasios Papathanasiou and Michael Scott, University of Rochester

  Athanasios Papathanasiou put forth the idea that the performance characteristics of common memory and storage architectures have changed enough that in order to get good performance (where performance may affect energy consumption), aggressive prefetching must be done. There have been drastic improvements in CPU power and storage capacity with moderate improvements in I/O bandwidth, but I/O latency has hardly improved at all. This means there is decreased risk and increased need to do prefetching.

  Bandwidth has increased 40% per year, but latency has only increased 10%. In order to lower the latency to the same level as before, increased prefetching is required. Furthermore, the amount of memory available to do speculative operations has increased. Memory slack has increased by a factor of 100 over the past 15 years.

  Having pitched the idea of aggressive prefetching, Athanasios presented some research challenges that would have to be met in order to make aggressive prefetching worthwhile: device-aware prefetching; characterization of an application's I/O demand; coordinating I/O requests with device power states; speculative predictors that provide sufficient data coverage; and better metrics to account for the true cost of cache hits and misses.

  Several attendees pointed out that good access models are needed in order to prefetch the right data. Athanasios agreed, adding that we need to come up with some generic models with parameters that can be tuned for specific devices or applications.

OUTSIDE THE COMFORT ZONE

Summarized by Steve Zhang

- ***Why Markets Could (But Don't Currently) Solve Resource Allocation Problems in Systems***

  Jeffrey Shneidman, Chaki Ng, and David Parkes, Harvard University; Alvin AuYoung, Alex Snoeren, and Amin Vahdat, University of California, San Diego; Brent Chun, Intel Berkeley Research Lab

  Jeffrey Shneidman argued for using markets to solve the resource allocation problem in large-scale systems. Utilization data from systems like PlanetLab shows how demand exceeds supply, especially during peak times, and demonstrates the need for an allocation policy. Using markets would allow for an efficient policy where those who value the resources the most receive them.

  He outlined some problems that would face any allocation policy. First, it would have to be selected and supported by the users. In addition, it would be difficult to divide resources, since different resources in a system may be connected in subtle ways. Consumers would also need to predict needs well and accurately value their needs based on the currency chosen. Finally, the implementation of a market policy requires a method for expressing bids easily and efficiently. Despite these problems, and although using market economies have been studied before, he believes today's environ-ment (e.g., demand much higher than supply, a semi-cooperative user base, repeated large resource allocation, improved OS support for resource isolation) could make revisiting this old idea fruitful.

  Several people expressed doubt that demand outpacing supply is only a problem for free usage testbed systems. Others noted the difficulty in choosing a viable currency for such a market economy. Although history has shown that artificial currencies tend to work poorly, using real money is generally unpopular with user bases, since users would not start on an equal footing. Finally, real economies have problems that are generally addressed by government-controlled regulatory agencies, and an analogous solution would need to be found for a market-based allocation policy.

- ***Operating Systems Should Support Business Change***

  Jeff Mogul, HP Labs

  Jeff Mogul expounded on why building systems with business changes in mind (e.g., starting or changing services, meeting new regulations, mergers, and spinoffs) is extremely important for most enterprises. AT&T wireless's downfall, HP's SCM rollout, and Comair's crew-scheduling fiasco were mentioned as anecdotal evidence of the difficulties in adapting to business changes and the high costs of being unable to do so efficiently. Agile businesses tend to dominate their markets. However, most business systems are so complex that changing or replacing them is extremely costly, and with most IT departments' current focus on cost-cutting, change is never easy or quick.

  Jeff believes that research should focus on allowing application-level flexibility, which requires standardization at the lower layers at enterprise scales. OS-level flexibility, which has been the historical focus of many researchers (e.g., microkernels, extensible OS), actually undermines standardization. Although formal verification may be unrealistic for years to come, OS conformance testing should be a first-class research topic. Also, capturing an accurate snapshot of an IT infrastructure at many levels and quantifying the value of IT (how much money a machine or a system is making) would be vital for management to gauge the costs and benefits of system changes accurately. Finally, with the prevalence of outsourcing, auditability of systems becomes critical for third-party auditors to verify that a customer's requirements are being met by the supplier.

Researching these areas is challenging, however, because enterprise applications tend not to be open source and may require millions of dollars and several person-years to install and configure. Studying controlled testbed systems in a lab environment is not a replacement for looking at real deployed systems. In addition, it is not clear how to measure the success of any method in addressing these concerns.

IT'S NOT AI, IT'S SYSTEMS

Summarized by Steve Zhang

- ***Designing Controllable Computer Systems***

Christos Karamanolis, Magnus Karlsson, and Xiaoyun Zhu, HP Labs

As a prelude to Christos Karamanolis's talk, Elizabeth Bradley of the University of Colorado, Boulder, gave a brief introduction to control theory. She talked about simple methods and tools that work for linear time-invariant systems. However, operating system problems tend to be nonlinear and time-varying, the solutions for which are usually ad hoc. Although it's possible to simplify many of these cases to work with a linear system-based approach, it's very important to be mindful of the assumptions that must be made for these solutions to be valid.

Christos Karamanolis talked about his experience in applying control theory to systems management. Recent surveys have shown that 75%–80% of IT costs go into managing existing systems. A feedback-based control system would allow humans to be taken out of the loop and thus cut costs immensely. However, because computer systems change quite frequently, adaptive control is needed to dynamically estimate models to be used by the controller.

The authors experimented with using an automated controller as a scheduler that handles different classes of clients, and attempted to have the system maintain a consistent throughput based on a service-level agreement. Some important lessons were learned: First, more recent controller actions must have higher impact on current measurements than earlier actions. Second, the action-measurement relationship must be close to linear; where that is not the case, different actions and/or measurements must be tried until a linear pair is found. More properties from control theory were translated into system requirements in order to help system architects design systems more amenable to automated control, and these are listed in the paper. During the Q&A session, it was established that although control theory is at least four decades old, researchers have only recently explored taking formal control-theory approaches to systems issues. However, informal ad hoc methods based on control-theory principles have been used for quite some time.

- ***Three Research Challenges at the Intersection of Machine Learning, Statistical Induction, and Systems***

Moises Goldszmidt and Ira Cohen, HP Labs; Steve Zhang and Armando Fox, Stanford University

Terran Lane from the University of New Mexico set the stage for this talk by providing an overview of machine learning, focusing on supervised learning and reinforcement learning. In supervised learning, the goal is to build a model that can predict system output from sensor values, whereas in reinforcement learning the goal is a model that can control the system according to sensor values to achieve some desired behavior. For supervised learning, there are well-established techniques that can handle high-dimensional data, but reinforcement learning techniques work best for low- (5–10) dimensional data. The speaker only briefly touched upon unsupervised learning but believes that many system problems may require such techniques.

Moises Goldszmidt then talked about the challenges of applying any statistical learning technique to systems. More specifically, he related the problem to his work with correlating low-level system metrics with higher-level objectives. Finding such correlations not only would help novice system administrators deal with simple problems, but would also be useful to experts by providing better visibility about the behavior of large-scale systems.

Although there is generally plenty of raw data available for problems in this arena, thanks to mature measurement and monitoring tools, there is a lack of labeled data that would aid the evaluation and comparison of different approaches. Another challenge is that learning schemes must be adapted to handle online streams of data. Although the machine-learning community has yet to provide any general solutions, it is possible to use domain-specific knowledge to construct efficient algorithms for the systems area. The speaker also noted that in most cases, obtaining true root-cause analysis is neither practical nor necessary. Instead, one should strive for diagnosis that can easily map to possible repair actions.

■ ***Panel: Control Theory/Machine Learning***

Christos Karamanolis, Elizabeth Bradley, Terran Lane, Moises Goldszmidt

This panel focused on the feasibility of applying control theory and machine-learning techniques to large-scale distributed systems. The consensus seemed to be that without centralizing data from individual nodes, it would be very difficult if not impossible for these approaches to effect conformance with systemwide high-level policy. It was noted, however, that while a globally optimal solution may be impossible for distributed systems, locally stable methods (e.g., TCP/IP) are often practicable.

This led to a reference to biological systems and how nature has solved the problem of achieving global goals from local control. Terran Lane responded that Mother Nature has reached such solutions through billions of years of experiments on an infinitely parallel supercomputer. He added that if one had a problem that had an accurate analogy in nature, using nature's solution would be feasible. However, most problems do not fit such a model.

CLEANING UP THE MESS WE'VE MADE

Summarized by Alexandra Fedorova

■ ***Making System Configuration More Declarative***

John DeTreville, Microsoft Research

System configuration is hard. A huge fraction of every user's time is spent futzing in a computer system. This is the biggest performance bottleneck. The issue is that a configuration is a shared mutable state. We update the state in place when we install and uninstall. A system's correctness depends on every install and uninstall we have ever done. The proposal is to use declarative configuration: record everything that describes system configuration, all files, all system variables. Then we have a chance of checking whether the configuration is correct. But in order to do this, we need a system model.

The system model can incorporate submodels. Programmers, publishers, and remote administrators can write these submodels. Models express rules for composing the programs into systems. The expectation is that system models will be easier to compose from submodels. As a result, no sequence of installs and uninstalls can result in a badly formed system instance. The drawback of this approach is that system models/policies may be too difficult to express. In conclusion, John pointed out that earlier efforts at declarative configuration were not widely adopted, because they were targeted at programmers.

Joe Hellerstein asked how to deal with distributed applications. John said that this is a hard problem but that improving local system administration is a good start. Jay Lepreau pointed out that standard program installers are already designed to handle program interdependencies. John responded that they have the right mechanisms, but in practice they do not handle these interdependencies properly.

■ ***Reducing the Cost of IT Operations—Is Automation Always the Answer?***

Aaron Brown and Joseph Hellerstein, IBM T.J. Watson Research Center

Aaron Brown said that costs of IT operations are quickly outpacing server spending. A common solution is automation, but expenses involved in developing and deploying an automated solution often outweigh the savings it provides. Aaron provided a case study where automated solution did not help to cut costs. Then he offered a mathematical framework for evaluating the cost-effectiveness of a solution.

Basic cost model: There are fixed costs for setup and maintenance and variable costs for automated inner loop and per-instance tasks. Because there are fixed costs involved, you have to look at the lifetime of automation, to amortize fixed costs over time. Automation lifetime can be very short, because the software package or the installer might change, for example. Apart from cost, there are the issues of trust and adoption. Automation is a disruptive force for IT systems managers. Using an incremental transition path from manual to automatic may be the right approach.

Margo Seltzer suggested that it may be difficult to know the lifetime of automation. Aaron agreed and said that they would like to be able to predict it. Christos Karamanolis wondered how they can quantify fixed costs for the model. Aaron responded that this is challenging and that user studies are needed to evaluate the costs of complication that come with automation. However, they do have a way of modeling such costs.

■ ***Human-Aware Computer System Design***

Ricardo Bianchini, Richard Martin, Kiran Nagaraju, Thu Nguyen, and Fabio Oliveira, Rutgers University

Thu Nguyen began his talk by encouraging the community to consider the human as a first-class entity in computer system design. He argued that systems designers should use human-aware principles, such as:

- making systems more robust to human mistakes;
- understanding human actions and mistakes;
- developing techniques and infrastructure to increase human understanding of systems to prevent, hide, and undo mistakes;
- designing new metrics and benchmarks to measure system improvements.

He then described their work on understanding operator actions and mistakes in configuring Internet services. The results of this study could be used to fix and prevent user mistakes. Thu concluded by admitting that designing good human-factor benchmarks is hard and recruiting human study subjects with the necessary background is even harder.

Pei Cao suggested that the reason why networking guys do a lot more online testing is because Cisco routers are specifically designed for online testing. If people built better software, maybe we would not have such issues with installation complexity. If you build complex software, you have to have a user model in mind.

APPROACHES TO OS RESEARCH

Summarized by Prashanth Bungale

- ***Thirty Years Is Long Enough: Getting Beyond C***

  Eric Brewer, Jeremy Condit, Bill McCloskey, and Feng Zhou, University of California, Berkeley

  Bill McCloskey explained that the point of this talk was to get people to think of C as a bad habit and to stop using it. Safety and security have now become more important than any other factors. Low-level systems are still unsafe and insecure, and C is the main problem. Java has failed mainly because it is not expressive enough and because porting applications is expensive. The authors believe that it's possible to design a systems language that is safer and better than C, Java, or C#. Areas for improvement include memory management (GC is not good enough), concurrency (manual locking is too error-prone), data layout (the programmer should have bit-level control on data storage in memory), and API adherence (compiler checks should be automatic).

  Their proposal for a new language, Ivy, guarantees that the following classes of errors are eliminated: buffer overflows (via bounds checking), dangling pointers (via checked memory management policies), race conditions and deadlocks (via use of atomic sections instead of explicit locking), API violations (via type qualifiers), resource leaks (via computation stacks), and macro errors (via a safer and newer preprocessor). In conclusion, Bill pointed out the problem of having two conflicting goals: starting out with a safe, clean foundation, or keeping existing code relevant.

  An audience member asked about what linguistic features were novel in Ivy, and why. The speaker said that this was future work. Prashanth Bungale asked why, when previous attempts such as Cyclone failed for this reason, we should believe that Ivy won't end up involving enormous amounts of human intervention. Bill replied that it is a fundamental goal of Ivy to reduce manual intervention as much as possible, but it remains to be seen how much this can be done. Jay Lepreau said that Cyclone on TCP/IP (which had millions of lines of code) actually hadn't required very much manual intervention, and that perhaps what kept Cyclone from being widely adopted were things like resistance and inertia.

- ***Broad New OS Research: Challenges and Opportunities***

  Galen Hunt, James Larus, David Tarditi, and Ted Wobber, Microsoft Research

  Galen Hunt described his working definition of OS research as research into the base abstractions provided for computation and into the practical implementations of those abstractions.

  Singularity is a research project with the following hypothesis: Sound verification techniques can be combined with new OS abstractions to provide dependability, reliability, and security (though sometimes at the expense of performance). It has focused on the following:

  - Configuration and manageability: The OS knows a lot about the hardware but next to nothing about the applications. Where's plug 'n' play for software applications?
  - Safe system extension: Extensions add new value to applications from the user's perspective (e.g., Google toolbar) but are unsafe. If we look at the last few SOSP submissions, safe OS extension has been an active area of research. But what about safe application extension?
  - Multi-processor cores: We can expect up to 256-processor cores in the foreseeable future, and hence need better OS support.

  The Singularity architecture includes a VMM for abstract hardware (with the abstract instruction set being type-safe, memory-safe MSIL), closed processes (i.e., no shared memory, no dynamic code loading, no dynamic code-generation), and IPC channels with contract guarantees.

  One key feature of Singularity is the concept of software-isolated processes (SIPs), where each process has its own garbage collector and its own garbage collection domain, exclusive ownership of its address space, and no pointers outside its address space (guaranteed because of type-safety). Therefore, to exploit this situation for performance reasons, the entire system is run in ring 0. According to their IPC micro-performance results, Singularity is an order of magnitude faster than hardware-protected systems, because there is no hardware protection domain change, and an order of magnitude slower than an in-process procedure call, because the IPC involves a GC domain change.

  Gernot Heiser asked if they also had a new hardware model; otherwise, how would type-safety help device drivers? Galen responded that their hardware model already incorporates simple I/O ports, for example, but it is not yet sophisticated. Armando Fox was concerned that while the changes regarding the hardware protection domain (e.g., entire system running in ring 0) may only change the performance from being "really fast" to being "really, really fast," why get rid of the mechanical intellect just for this reason? His concern was mainly that historically, checking through software has been hard to get right. Jay Lepreau pointed out that Andrew Appel broke type-safety in secure chips. Galen responded that if you care a lot, you can always put the process in a higher ring; an option can be provided to say, "OK, I don't trust this code. Use a separate protection domain." Robert Grimm commented that using an abstract machine as an executable platform made it hard to predict . . . The speaker interrupted him and immediately responded that everything is *compiled* (no more JIT).

■ ***patch (1) Considered Harmful***

Marc Fiuczynski and David Walker, Princeton University; Robert Grimm, New York University; Yvonne Coady, University of Victoria

Marc Fiuczynski said that the key lesson from his talk is that we need better tools to improve OS evolution. The open source model is used everywhere—embedded systems, servers, clusters, HPC—and fosters community development. Updates are performed through patches, where the changes can correspond to intraprocedural changes (modifications to the internal logic of a function), intermodule changes (changes to a function's signature or a data structure's field-makeup), or behavior changes (changes to the semantics of interfaces). Through examples of Linux kernel patches containing separate concerns, Marc showed how an extension could easily cover a hundred existing kernel files, even though it represents a logical unit expressing a single, cross-cutting concern. Current practice makes OS evolution hard and dirty: Since it is hard to understand implementation of a concern, composition of separate concerns is very hard, maintenance is generally hard (or very annoying), and new concerns bloat and dirty the mainline code base.

Marc then presented C4, a toolkit to improve OS evolution. The problem with the existing approach of patch (1) is that it operates at the lexical level. By contrast, C4 functions at the semantic level so that we can build better tools to manage complexity and analyze interference semantically. Their approach is to use aspect-oriented software development (AOSD) techniques. C4 provides a semantic patch compiler through which one can express behavioral changes as semantic patches using aspects, which provide a language-supported methodology for integrating crosscutting concerns with a program. The C4 toolkit consists of an unweaver and a weaver, which are analogous to diff and patch, respectively. The main difference is that the unweaver actually removes the code belonging to an aspect from the baseline code and the weaver puts it back in the right place, using knowledge of C's abstract syntax to avoid merge conflicts.

Their current focus is the engineering effort needed to get the weaver and unweaver working. Future work enabled by C4 includes program analysis tools, identifying data structure changes, identifying memory safety (or lack thereof), and capturing programmer intentions via declarative frameworks.

Steven Hand commented that a lot of the problem simply lies in really crappy open source code. Margo Seltzer asserted that they are still distributing patches and asked what they distribute exactly. Marc responded that they distribute C4 files. Chris Small asked how they are going to get people to use these tools. Marc replied that whether elegant or not, they reduce the pain for the user, and that is what is going to get people to use their tools. Kirk McKusick asked why not use CVS. Marc responded that even then one would have to deal with merge conflicts. Jay Lepreau commented that the problem is that Linux has a pope model—there's only one integrator.

■ ***Panel: Do We Work Within Existing Frameworks or Start from Scratch?***

Bill McCloskey, Galen Hunt, Marc Fiuczynski, Robert Grimm, Russ Cox, and Eric Brewer

Chris Small: Mike Jones once said to me, "When you're on an exponential path, nothing you do now matters." So, why waste your time on Ivy? Why not build a better language instead?

Eric Brewer: Do both. We're not going to get the language right either; but Ivy is extensible.

Robert Grimm: Every year, the programming language community publishes work and more work on Java + delta. What if you want Java + delta + delta? Extensions are the answer.

Margo Seltzer: Galen's talk seemed so far removed from helping the end user.

Galen Hunt: Extensible applications would help the end user to a great extent.

Andrew Hume: Saying "The language we're programming in is the problem" is a delusion.

Galen Hunt: The language directly affects what we think.

Andrew Hume: That's balls.

Eric Brewer: I don't believe any of that.

Phil Lewis: Education—what languages do you learn? Let's not teach people C! Ten, twenty years from now, the problems will go away.

John DeTreville: This is not about languages. Back in the '70s, people wrote their own OS, compiler, etc. But now it's impossible to write your own OS.

Michael Scott: Regarding the panel question, look at past examples. Two and a half models for success: (1) Exponential curve: Java, Perl, HTML, etc.: there wasn't a market for it. (2) Migration path: C++ (from C), XHTML (from HTML), Opteron (30 years of x86). (2.5) In between: MESA, Smalltalk, VKernel, maybe Plan9. So which model should we be looking at now?

Galen Hunt: My answer is a definitive it depends. Will I ship Singularity as a product? No, we will learn ideas.

Pei Cao: Galen's presentation's problem and solution seemed totally far off.

Galen Hunt: Then you young people should go work on solving the interesting problems.

Margo Seltzer: What lessons can the OS community learn from each of your projects, assuming that you're wildly successful?

Robert Grimm: Languages are very important for reliability and security.

Galen Hunt: Sound static analysis and verification can dramatically impact our ability to test (9,000 testers testing Windows currently, and you know the result).

Eric Brewer: We do want to support legacy drivers, etc.

Galen Hunt: The OS knows nothing about the application. We still have a 1970 model of what a program is: a.out, stdin, stdout, stderr model.

Andrew Hume: Sometimes you just get it right! Clarity and economy of expression . . . Are we ever going to have "the model" of an OS, or are we going to continually have periodic purging?

Galen Hunt: I don't know!

DISTRIBUTION

Summarized by Nikolaos Michalakis

■ ***WiDS: An Integrated Toolkit for Distributed System Development***

Shiding Lin, Aimin Pan, and Zheng Zhang, Microsoft Research Asia; Rui Guo, Beijing University of Aeronautics and Astronautics; Zhenyu Guo, Tsinghua University

Zheng Zhang started by explaining that today's distributed system development process is unscalable for humans. Debugging is painful, and there is code divergence between simulation and implementation. WiDS is designed to maintain one code for both simulation and implementation, simplify debugging by allowing debugging in a single address space as much as possible, and support large-scale performance studies of the system in design. To achieve these goals, WiDS essentially lets programmers link their code to different libraries according to their needs (single-node simulation, parallel simulation, network execution). Verification and debugging of protocol implementations can be done through a model checker in simulation.

Experience with WiDS shows that distributed system development and deployment can be greatly simplified both in building complete systems such as BitVault, a data retention system, and in the large scale, such as the RNRP protocol on two million nodes. Research in progress hopes to include playback of message logs in simulation mode to find network-related bugs. In terms of extending APIs, whether to use events or threads must be a programmability-centric decision. Zheng's conclusion was that distributed system and tool development should go together.

Ion Stoica noted that in reality there are problems due to connectivity asymmetries, congestion, and unbounded packet delivery, and asked how many of those violations WiDS included in their simulation. Zheng replied that they have end-to-end connectivity simulation and packet drops and that the best way to incorporate newly discovered violations is to use the model checker and update the protocol model. Andrew Hume asked if Zheng had found anything not covered by the protocol checker. He replied that this could happen if, for example, the protocol specification was implemented incorrectly or if the model was wrong.

■ ***Causeway: Operating System Support for Controlling and Analyzing the Execution of Distributed Programs***

Anupam Chanda, Khaled Elmeleegy, and Alan Cox, Rice University; Willy Zwaenepoel, School of Computer and Communication Sciences, EPFL

Anupam Chanda began by sketching the execution flow of a multi-tier program composed of a Web and database server. Execution steps are performed by "actors," such as system calls, over "channels," such as sockets. As he noted, it is sometimes useful to write meta-applications to control and analyze the execution flow of such multi-tier programs. Meta-applications can be categorized as "log-based" (e.g., Magpie) or "metadata passing" (e.g., Pinpoint). Unlike log-based approaches, metadata passing across actors allows online control of multi-tier programs and is the approach chosen by Causeway, a framework that provides OS support for building meta-applications.

The framework is placed at the level of the OS, since placing it at either the application or middleware level might lead some components of the multi-tier program to be oblivious to metadata passing, might require modifications to all applications, and, in the case of middleware, might not support all legacy protocols. Causeway associates metadata with an actor upon a write on a channel and propagates metadata when the actor at the other end of the channel performs a read, thus making metadata passing follow the program flow. Causeway invokes a meta-application through callbacks. The authors implemented a priority scheduler for a Web server application in only 150 lines of code, making a convincing argument for the feasibility of building meta-applications using OS support. A concern to be addressed in the future, however, is security and, in particular, the illegal modification of metadata by the running program.

Petros Maniatis suggested that Causeway could benefit from the use of both metadata passing and log-based analysis, since logs are streams and could be mined as they go by. Anupam didn't find the idea feasible, however. Doug Terry wondered what Causeway could do in the OS layer that it couldn't do in the middleware layer. Anupam clarified that by OS support he meant modifications to the OS as well as system libraries, but failed to answer the question exactly.

- ***Treating Bugs as Allergies: A Safe Method for Surviving Software Failures***

Feng Qin, Joseph Tucek, and Yuanyuan Zhou, University of Illinois, Urbana-Champaign

Bugs are inevitable, and they lead to system failures. Existing solutions such as rebooting, checkpoint-recovery, application-specific recovery, and failure-oblivious computing cannot recover from deterministic bugs. Yuanyuan Zhou offered a different approach to this problem: Since deterministic bugs are hard to cure, then "run away" from them, essentially treating them as allergies. This is achieved by changing the execution environment on demand upon soft failures. Essentially, the program is rolled back after each change is applied to the execution environment until the bug disappears.

This method is developed by the Rx system in a comprehensive, safe, noninvasive, efficient, and informative manner. Sensors detect bugs before the program crashes, and changes include padding allocated memory to avoid overflows, allocating memory in an isolated location to protect against memory corruption, and, in the worst case, dropping user requests. However, all changes respect the application's API. While preliminary results on escaping deterministic bugs are more than encouraging, there are still several challenges for Rx, such as committing on the program's output (one reply to user) and the need for more powerful bug sensors. Yuanyuan emphasized that to be successful against deterministic bugs it is necessary to make the system nondeterministic.

Aaron Brown asked how Rx handles a bug that is detected after output is sent to the user. Yuanyuan said that once the output reaches the user the problem is hard, but before the output reaches the user, techniques such as data mining could help prevent this. Aaron then asked how Rx handles concurrent requests. The reply was that requests are not serialized, but replays after checkpointing are. Brett Fleisch asked whether by "non-deterministic" she meant increasing the percentage of hidden bugs compared to deterministic bugs, and she agreed. Petros Maniatis (Intel Research) pointed out that changes by Rx might break down programmers' optimizations and asked how Rx would cope with that and ensure safety without programmer feedback. Yuanyuan's reply was that future work will include identifying common assumptions made by programmers and incorporating them into Rx. Armando Fox asked how Rx compared to failure-oblivious computing. The answer was that Rx is more general. Dug Terry asked how Rx prioritizes changes to the execution environment so that the effects are maximized. The answer was that, when possible, multiple changes are made simultaneously. Machine learning could be of further help there. Mary Baker asked how many rollbacks were necessary for avoiding bugs in general. The answer was not more than four.

SECURITY

Summarized by Nikolaos Michalakis

- ***When Virtual Is Harder than Real: Security Challenges in Virtual Machine-Based Computing Environments***

Tal Garfinkel and Mendel Rosenblum, Stanford University

Tal Garfinkel began by presenting functional differences between virtual and real (traditional) machines to support the hypothesis that such differences break existing security management approaches, so we have to rethink VM security. More specifically, traditional machines scale slowly and predictably while virtual machines do so rapidly, and traditional machines enforce homogeneity but virtual ones encourage diversity. In addition, traditional machines support stable populations, but virtual machines support highly transient ones, and the difference is more acute since virtual machines allow increased mobility, making it harder to link the VM to its owner.

The solution proposed is to move security-related functionality out of the guest OS and into a ubiquitous virtualization layer. Such an approach will help decouple security and management from the structure of the guest OS.

Margo Seltzer observed that this architecture resembles a microkernel, where the Trusted Computing Base is pulled out of the VM. Tal replied that microkernels were cool, and he didn't find anything wrong with that. Edward Wobber asked whether updating the VM state from outside the VM could be useful. The reply was that, depending on the OS, it could be. Jay Lepreau followed up on Margo's remark, saying that there is no problem in separating the security from the VM. Tal added that such separation can give more control to administrators and more flexibility to users. Rik Farrow asked whether the security layer would look like an additional virtual layer. Tal mentioned that the platform needs to be beside the VM for security, but it is an interesting question what the actual architecture will look like.

■ ***Make Least Privilege a Right (Not a Privilege)***

Maxwell Krohn, Cliff Frey, Frans Kaashoek, and David Ziegler, MIT; Petros Efstathopoulos, David Mazières and Steve VanDeBogart, University of California, Los Angeles; Michelle Osborne, New York University

Max Krohn said that a problem faced today by servers is that process boundaries do not always align with an application's security goals. Alice can steal Bob's data via buffer overruns, trojans, SQL injection, or even bad access control policies. Max presented a set of such scenarios based on Alice and Bob accessing the same Web site.

To avoid these issues, Asbestos OS uses Mandatory Access Control (MAC). Asbestos uses compartments to track and control data flow. Unlike other systems, compartments are introduced not only by the kernel, but by applications as well. The data tagger, which is a small component that has no privileges, tags data based on users. When running an Asbestos Web server, data flow is tagged; the more the components of the application/system are touched by data without conflicting tags, the more a compartment grows, independently of the processes involved. Compartments are tagged upon reading data, and the more elements that are touched (e.g., processes, sockets, virtual memory pages), the more a compartment grows. If a compartment that is already tagged by user A is touched upon a read by data from a new user B, the compartment is tagged anew with a third tag, AB. This prevents data from being written out to compartments having tags A or B, thus protecting users' data from each other. When an operation is done, the tags on a compartment are removed and the components restored. The system finally uses trusted declassifiers that can act on behalf of multiple users and traverse subprocess boundaries.

Philip Levis asked how Asbestos deals with database security. The answer was to have user data on different pages (serving as compartments) and a trusted index server. Jay Lepreau wanted to know how Asbestos differs from Flask in doing MAC in a distributed way. The answer was that applications, not only the kernel, can introduce compartments. However, as Jay noted, a trojan might contaminate a declassifier and get access to other users' data. Max agreed that they need to be careful with declassification.

Pei Cao asked whether an attacker could trick the process into restoring a component. The answer was that only the kernel enforces the restore. John DeTreville asked whether this fine-grained control is better. Max said, "The more fine-grained, the better." Alex Snoeren asked whether a compartment tag could be illegally changed in the case of multi-threaded or event-driven applications. The answer was that once a compartment has been labeled it cannot be accessed by a flow of different tag. Margo Seltzer asked what happens with other data, such as registers. The answer was the registers are flushed similarly to a context switch. Peter Druschel asked what happens if user-specific data gets into a stack and another user finds it. One way to deal with this problem is to wipe the stack out when restore is issued.

■ ***Access Control in a World of Software Diversity***

Martin Abadi, University of California, Santa Cruz; Andrew Birrell and Ted Wobber, Microsoft Research

Andrew Birrell described the first steps of a design for authentication and access control as part of Microsoft's Singularity operating system project. In actual operating systems the facts that principals are bound to either users or "logged-in" users and that ACLs are flat lists of principals are inadequate for making flexible access control decisions.

The new design is based on three components: the naming tree, which records decisions the administrator or implementer has made (e.g., when installing a program), thus separating static policy decisions from online control ones; a compound principal mechanism; and a pattern recognizer for accessing control lists. In the Singularity design, the principal is just a string constructed by a path in the naming tree and logical operators. Since applications are part of principal names, they are described in the naming tree in the form of manifests. Andrew argued that enumerating principals in a control list will not give the desired flexibility. Instead, given a list and a principal, it must be determined whether the principal string is contained in the list string. The right approach, therefore, is to do pattern recognition, and for that reason regular expressions are used.

Armando Fox asked whether time expiration is included in the design. The reply was that this might be useful but no compelling need was found for that yet. Margo Seltzer noted that regular expressions create a disconnect between flexibility of expression and usability, because they are not understood by mere mortals. Andrew replied that the ACLs will be created mostly by installation programs, not humans. Removing regular expressions does not solve the problem. The goal is to be expressive. Michael Jones asked how reputation-based access control could be incorporated. Andrew replied that he would not like to add more complexity than that of the naming tree. Reputation makes him nervous. Michael Scott suggested that regular expressions could be used to find bad access control rules, something that Andrew agreed to look into. Alex Snoeren noted that since semantic value is put on the strings in the naming tree, there is a danger that if the tree is changed it will hurt the system. Andrew agreed that they better get these names right; relative paths would help there. Petros Maniatis asked whether they could do combinations of authentication methods in a scalable manner. Andrew replied that it was not possible in regular expressions; they would need to enhance their language.

SENSOR NETS

Summarized by Steve VanDeBogart

- ***PRESTO: A Predictive Storage Architecture for Sensor Networks***

Peter Desnoyers, Deepak Ganesan, Huan Li, Ming Li, and Prashant Shenoy, University of Massachusetts

Deepak Ganesan presented the ideas and motivation for PRESTO, a query architecture for sensor networks. Desirable features include low latency, low power utilization, the ability to query archival data, and the ability to formulate new queries after events have already occurred. PRESTO tries to provide all these features by taking advantage of the decreasing cost of storage as well as suppressing communications that report no new information.

Many events that sensor networks are currently being used to monitor have domain-specific models. For instance, temperature variation is easily predicted from time of day and season. Based on previous data, the PRESTO proxy can send models to the sensor nodes. The sensor nodes can then only report events that violate the model. The proxy can then either send a new model or note the violation as an aberration. This technique is more energy efficient than a push model. It also allows the proxy to answer queries immediately, since the proxy is notified whenever an abnormal event occurs. If the query requires more accuracy than the model provides, the proxy will first examine its cache to see if it has already retrieved the needed information. If not, it will poll the relevant nodes. This information may be available at a lower tier in a multi-tiered network, possibly sparing the energy-scarce motes at the bottom from answering the query directly; if not, the data is cached at each tier on the way up, preventing further direct queries of the low-level node if the data is needed again. These techniques provide a middle ground between streaming out all the data, which is energy expensive, and querying nodes directly, which is slow.

- ***Towards a Sensor Network Architecture: Lowering the Waistline***

David Culler, Prabal Dutta, Cheng Tien Ee, Rodrigo Fonseca, Jonathan Hui, Philip Levis, Joseph Polastre, Scott Shenker, Ion Stoica, Gilman Tolle, and Jerry Zhao, University of California, Berkeley

Philip Levis began by saying that sensor network research today is a mess. There are a lot of different options for solving a given problem, but each solution is vertically integrated into a complete stack of solutions and each component in a stack is incompatible with any other stack. Therefore, if you want to use modules from more than one stack you have to build a totally new stack that integrates the components you need. This lack of capability is a limiting factor to the advancement of sensor networks.

This wasn't a problem for the Internet because there was a well-defined protocol, IP, midway through the network stack that allowed work to proceed in parallel above and below that point. Can we just use IP in sensor networks? No, it isn't appropriate, but we should develop something for sensor networks that serves the same purpose that IP does for the Internet.

Philip went on to argue that SP, the Sensor Protocol, should be a single-hop protocol that provides a richer interface than just send and receive. It should not specify the wire protocol, because the underlying link layer varies too much. It should work both for address-free protocols, such as flooding and tree collection, and for name-based protocols. There should be an interface for the layers above to specify a forwarding predicate. Furthermore, it is important that it provide interfaces for things that have to cross layers, such as power management, timing, and security.

Petros Maniatis asked if SP will help in the wild or if it's an academic exercise. Philip responded that it will take place in both domains. In an academic sense SP will help us understand things at a deep level, but at the same time it will facilitate real code. Additionally, because sensor networks still exist in an isolated administrative domain, we may be able to iterate the design, unlike IP.

- ***Breakout Sessions***

Summarized by Rik Farrow

During the last portion of the HotOS workshop, the attendees split into groups that had been arranged by Margo Seltzer. The assignment for each group was to design and present a paper on a particular topic in one hour. As it turned out, all papers that included a PowerPoint presentation were accepted, and the one group that failed to reach that point had its topic rejected.

Completed papers will become part of the HotOS 2005 proceedings.

VEE '05: First ACM/USENIX Conference on Virtual Execution Environments

June 11–12, 2005

KEYNOTE ADDRESS

Summarized by Shuo Yang, Long Fei, and Xing Fang

■ *A Unified View of Virtualization*

James E. Smith, University of Wisconsin

James Smith started his keynote by discussing why virtualization techniques are interesting: they enable transcending interfaces, flexible innovation, adaptation to other software/hardware, networked computing, and enhanced security. He views virtualization as the fourth pillar in computer systems besides hardware, system software and application software.

He reviewed the domains and examples of virtual machines, and the origins of virtual machine concepts. He said that different interest groups, such as OS developers, compiler developers, and application programmers have different perspectives on virtual machines. System virtual machines provide a system environment that is constructed at the ISA level. Process virtual machines are constructed at the ABI level. High-level-language virtual machines (HLL VMs) provide APIs, i.e., they raise the level of abstraction.

He pointed out that many of the existing VM techniques were invented on an *ad hoc* basis, and he asked whether there might exist something more fundamental than a collection of techniques. He then discussed his idea of how to establish uniform concepts. He also reviewed the solutions and challenges of VM techniques. From there, he put forward the question of how to enhance virtual machine performance. For example, the increase of system layer complexity leads to inefficiency. The Intel VT-x solution is to add another set of layers. He then presented some performance primitives, such as code cache support, visibility of micro-ops, and so forth.

Smith reminded the audience that killer applications motivate virtualization techniques. One of the killer apps for virtualization is security. HLL VMs, with security features built in; process VMs, whose code can be inspected before being executed; and system VMs, which provide isolation with simple VMMs, can all be used to support secure networked computing.

Smith also discussed the education curriculum about VMs. He proposed an outline of what to teach in a VM course, for example, putting together virtually all levels of computer system hardware and software. He believes it is a challenging and necessary task.

Smith concluded the talk with the following points: the common framework and terms should be resolved; virtualization needs higher concepts, rather than a bag of terms and techniques; and a unified conference to exchange ideas from different communities is needed, and VEE serves that goal perfectly.

SCALABILITY, PERFORMANCE, AND REAL-TIME

Summarized by Shuo Yang

■ *Friendly Virtual Machines: Leveraging a Feedback-Control Model for Application Adaptation*

Yuting Zhang, Azer Bestavros, Mina Guirguis, Ibrahim Matta, and Richard West

Azer Bestavros presented a Friendly Virtual Machines (FVM) framework for efficient and fair resource allocation when sharing an underlying host system. Bestavros first discussed the background of virtual machine adaptation: the trend of VM techniques being increasingly adapted to support applications running on third-party hosts, the need to isolate independent constituents, and the emergence of VM abstractions. He then said that the motivation of the research is to ensure fairness and efficiency in the underlying host resource allocation. He advocated the use of self-adaptation in the guest VMs themselves, based on feedback about resource usage and availability.

Bestavros defined a virtual machine that fairly and efficiently adjusts its demand for system resources as a Friendly VM (FVM) and proposed a resource-sharing technique that is applicable to any application whose execution is "friendly" to other applications sharing the same underlying resources. The friendliness feature of FVM applies the classical end-to-end argument to the problem of multi-resource allocation across a set of applications sharing the same infrastructure.

Bestavros said that the host in FVM needs to provide unbiased on-demand resource allocation and VMs, and he mentioned the pricing issues enabled by the FVM framework. Bestavros showed the experimental result of resource utilization using the FVM system with "made-up" benchmarks and real Web server benchmarks. The results showed that FVM successfully achieves fairness and efficiency in sharing common hosting resources.

■ *Diagnosing Performance Overheads in the Xen Virtual Machine Environment*

Aravind Menon, Jose Santos, Yoshio Turner, G. (John) Janakiraman, and Willy Zwaenepoel

Yoshio Turner presented Xenoprof, a systemwide statistical profiling toolkit for the Xen virtual machine environment. Turner first discussed how the increased adaptation of virtualization techniques can affect application performance in unexpected ways. He then presented an example of Web server performance degradation under the Xen system, which motivated their Xenoprof project. An application's performance in a virtual machine environment can differ markedly from its performance in a non-virtualized environment, because of interactions with the underlying VMM. Xenoprof is designed to enable coordinated profiling of multiple VMs in a system to obtain the distribution of hardware events. He introduced the Xenoprof design: a paravirtualized interface to support domain-level profilers. OProfile has been ported to Xen environment as a domain-specific profiler.

Turner presented an example of Xenoprof use, showing how they found a TCP-receive performance anomaly under XenoLinux. After that, he presented the work done with Xenoprof to analyze several performance problems observed under different Xen configurations for receiver, sender, and Web server applications. Turner concluded that Xenoprof is a useful tool to identify major overhead in Xen. Xenoprof will be included in official Xen and OProfile releases.

- ***Supporting Per-Processor Local-Allocation Buffers Using Lightweight User-Level Preemption Notification***

Alex Garthwaite, Dave Dice, and Derek White

Alex Garthwaite presented a local-allocation buffers (LAB) management technique that supports local buffer allocation with regard to processors instead of threads.

Garthwaite first gave a performance comparison under different LAB-size policies for the VlanoMark benchmark. He made the point that garbage collection is a hard problem when there are more threads than processors and high preemption rates.

He then presented a processor-local allocation buffers (PLABs) strategy that associates local allocation buffers (LABs) with processors and with buffers for each thread allocated from its processor's LAB. Multi-processor restartable critical sections (MP-RCS) techniques implement such a buffer allocation strategy. He then introduced the challenges of PLABs, such as the cost of dynamic checks and the need for expression translation in C. There is an overhead for using PLABs over thread-local allocation buffers (TLABs) when the number of threads is less than the number of processors, or when threads are entirely compute-bound. PLABs are much better than TLABs when the number of threads is larger than the number of processors. He showed that their mechanism can combine the PLAB and TLAB adaptively. Garthwaite said that their mechanism can easily be implemented in x86 architecture. He then presented the experimental results of their techniques and concluded that the simple MP-RCS mechanism is independent of the threading/scheduling model and is applicable to many platforms.

- ***A Programmable Microkernel for Real-Time Systems***

Christoph Kirsch, Marco Sanvido, and Thomas Henzinger

Marco Sanvido presented the microkernel system architecture for hard real-time applications. He first presented the reactive (response to environment) and the proactive (task scheduling in platforms) requirement in embedded systems, and introduced the concept and architecture of their solution's design. The E (embedded) machine is a virtual machine that triggers the execution of software tasks with respect to events. The S (scheduling) machine is a virtual machine that orders the execution of software tasks, and together their E+S machines equal the microkernel model. Their model represents the abstraction of the interaction between the hardware platform, reactively constrained by the E machine and proactively constrained by the S machine.

Sanvido talked about the time-safety requirement of hard real-time applications and presented a proof that the time-safety requirements were fulfilled with schedule-carry code in their system. He introduced the implementation issues of the E+S machine, which has been implemented using the StrongARM processor and integrated into HelyOS. In conclusion, Sanvido said that their work has adopted microkernel architecture for the real-time application domain.

OBJECTS AND THEIR COLLECTION

Summarized by Long Fei

- ***The Pauseless GC Algorithm***

Cliff Click, Gil Tene, and Michael Wolf

Cliff Click began by pointing out that garbage collection response time has become an important problem for applications that contain response-time-sensitive components. He described a system, including CPU, chip, board, and OS, built by Azul Systems to run garbage-collected virtual machines. The hardware supports fast user-mode trap handlers. The hardware TLB supports an additional privilege level, GC mode, which lies between the usual user and kernel modes. TLB violations on GC-protected pages generate fast user-level traps instead of OS-level exceptions. The CPU supports a read barrier instruction, which resembles a standard load instruction except that if it refers to a GC-protected page a fast user-mode trap (GC-trap) handler is invoked.

The GC algorithm is highly concurrent, parallel, and compact. The algorithm is divided into three phases, Mark, Relocate, and Remap. The Mark phase is responsible for periodically marking the live and dead objects. The Relocate phase finds pages with little live data, to GC-protect, relocate, and compact them and to free the backing physical memory. The Remap phase updates every relocated pointer in the heap. During the relocating phase, if a mutator's read-barrier GC-traps, the GC-trap handler looks up the forwarding pointer and places the correct value both in the register and in memory.

The authors backed up their design with experiments using a modified version of SpecJBB benchmark. The authors also state that the read barrier behavior can be emulated on standard hardware at some cost.

- ***Use Page Residency to Balance Tradeoffs in Tracing Garbage Collection***

Daniel Spoonhower, Guy Blelloch, and Robert Harper

Daniel Spoonhower presented this paper. The key innovation of the paper is a mechanism that allows the collector to dynamically balance the tradeoffs of copying and non-copying collection for each page based on page residency, a measure of the density of reachable objects on a page. If the residency of a page is sufficiently high, the page should be promoted, otherwise it should be copied.

Measuring the residency of even a single page requires a traversal of the entire heap. To avoid this overhead, the authors devised several residency-prediction heuristics and recovery mechanisms to handle poor predictions. The authors also identified a continuous range of tracing collectors and showed that classic GC algorithms can be considered special cases of the proposed GC algorithm with extreme residency assumptions.

Their experiments revealed the impact of heap size and configuration thresholds on the performance of GC algorithms. Mark-sweep performs better when the heap size is small, whereas semi-space performs better when the heap size is large. In both cases, their algorithm yields a performance close to the better of these. Experiments show that this new algorithm has only a small variation in performance under six different configurations.

- ***Exploiting Frequent Field Values in Java Objects for Reducing Heap Memory Requirements***

Guangyu Chen, Mahmut Kandemir, and Mary J. Irwin

Guangyu Chen said that the capabilities of applications executing on embedded and mobile devices are strongly influenced by memory size limitations. The authors use object compression to improve memory space utilization in an embedded Java environment. The compression is based on the observation that a small set of values appears frequently in heap-allocated objects.

Their approach uses profile information to categorize the object fields into three levels: level-0 (the field does not have a dominant frequent value), level-1 (the field has a non-zero or non-null frequent value), level-2 (the field has a frequent value that is zero or null). They propose two compression schemes. The first one divides an object into primary part and secondary part (containing level-2 fields). The secondary part is eliminated if all the level-2 fields are zero or null. The second scheme shares level-1 fields among multiple objects. Experimental results showed that these compression schemes can reduce the heap size significantly with little performance impact.

GOING NATIVE

Summarized by Long Fei

- ***An Efficient and Generic Reversible Debugger using the Virtual Machine based Approach***

Toshihiko Koju, Shingo Takada, and Norihisa Doi

Toshihiko Koju began with the statement that reverse execution is very useful for locating the cause of software failures. He described a novel reversible debugger that uses a virtual machine based approach. This debugger provides compatibility and efficiency. In addition, it provides two execution modes: native mode, where the debugger is directly executed on a real CPU, and the virtual machine mode, where the debugger is executed on a virtual machine.

In order to provide compatibility and efficiency, the debugger uses native machine code as its target. The virtual machine translates the native machine code of the target program by inserting code to save states that are changed during execution. The debugger is capable of switching between the native and the VM mode. In the native mode, users cannot reverse-execute the target program; in the VM mode, the users can use reverse execution. Some basic debugging functionality (e.g., breakpoint, step) is supported in both modes. The debugger allows four types of settings, to allow trade-offs between granularity, accuracy, overhead, and memory requirements of reverse execution. The user can choose the appropriate setting by designating the proper reverse execution unit (line or procedure) and optimization flag (enable or disable).

- ***Module-aware Translation for Real-life Desktop Applications***

Jianhui Li, Peng Zhang, and Orna Etzion

Jianhui Li explained that a dynamic binary translator is a just-in-time compiler that translates source architecture binaries into target architecture binaries on the fly. When hot modules are loaded and unloaded repeatedly, traditional dynamic translators spend a significant amount of time on repeatedly translating these modules. He proposed a translation reuse engine that uses a novel verification method and a module-aware memory management mechanism.

There are three stages to accomplish translation reuse: translation reservation, source binary verification, and translation revivification. In this framework, when a translated code block is invalidated, it is preserved by the reuse engine and saved by the execution engine. In order to verify that the saved translation is exactly the expected translation for a code block, the translation engine uses a save-and-comparison scheme. If the reuse engine decides to reuse the translation for a piece of the source binary, it saves a minimum set of source binaries that determine the semantics of translation. Before the translation engine translates a piece of the source binary, it requests the reuse engine to compare the saved partial source binaries with their counterparts in the current binary image. The saved translation is used if they are the same.

The authors propose a module-aware memory management mechanism, which organizes the translation code blocks of different modules into different pools (module-private page pool and general page pool). When a hot module is unloaded, its private code pages are reserved for future reuse (unless it's identified as not reusable). Experiments with real-life desktop applications show that this new translation-reuse technique can significantly improve the performance of four real-life desktop applications.

- ***Planning for Code Buffer Management in Distributed Virtual Execution Environments***

  Shukang Zhou, Bruce R. Childers, and Mary Lou Soffa

Many devices in a distributed computing environment have tight memory constraints. One approach is to download code partitions on demand from a server and to cache the partitions in the client. Shukang Zhou and his colleagues addressed the problem of intelligently managing the code buffer to minimize the overhead of code buffer misses. They propose to move the code buffer management to the server, where sophisticated schemes can be employed.

A program is first divided into code partitions, which are then stored in a code server connected to the client. Profiling is used to capture the hotness of code partitions. The client's code buffer is partitioned into multiple sub-buffers. The sub-buffers are ordered by the hotness of partitions assigned to them. One sub-buffer holds very hot code, while another may hold infrequently executed code. This approach is based on the fact that most programs spend a large part of their execution in a small portion of code.

The authors discuss the overall strategy of CB memory planning and then describe two particular schemes. In the fixed scheme, code partitions are always housed in the same sub-buffer during execution. In the adaptive scheme, partitions are cached in sub-buffers based on a program's run-time behavior. The authors also introduce a heuristic called density, which is defined as a partition's execution frequency divided by its size, to measure the priority of code partitions to reside in CB. Experiments show that these schemes have fewer CB misses, which translates to a significant speedup.

KEYNOTE ADDRESS

Summarized by Shuo Yang, Xing Fang, and Long Fei

- ***Application Servers as Virtualization Environments***

  Martin Nally, CTO, IBM Rational

Martin Nally first gave a broad overview of virtualization services. Virtualization techniques serve as tools of software development synergy between software and the execution environment. According to IDC data, the worldwide application server software platform revenue is increasing dramatically. He then introduced an example of writing Web server applications without knowing the specific target application servers, and showed the necessity for the Web server application to be compatible with different OS and hardware platforms.

He introduced various approaches to building Web server applications, such as JVM (J2EE), CGI-BIN, etc., and discussed the pros and cons of each approach. Then he discussed different levels of virtualization—OS, JVM, and Web server—with respect to their generality. OS virtualization provides important and very general concepts to support application execution. JVM virtualization provides platform independence. Application server virtualization targets certain classes of applications.

He said that application servers are usually thought of as containers and pointed out the important role of virtualization. First, the virtual environment serves as a logic wrapper. He gave a Web server application as an example of this view. He then discussed Web application containers as a virtual secure environment and talked about the scalability and performance challenges of application servers.

He presented cluster and workload management and hardware virtualization. One of the key features of application servers is scaling. There are a wide variety of configurations to virtualize hardware. He introduced how Web cluster failover is handled and recovered. Web clusters allow transparent application updates and enable continuous availability of service. WebSphere XD is the next level of sophistication of virtualizing hardware. He discussed the challenges of an on-demand operating environment for a large financial company: for example, the underutilization of servers and the inability to share. He believes virtualization, such as WebSphere XD's automatic management, provides a better solution than the conventional approaches in this case.

Nally concluded his talk with the following points. Application servers provide a virtual environment for executing Internet and intranet applications. Application servers present a simple virtual environment to application programmers. Application servers virtualize across many physically individual computers that may be running different OS application servers, even using some virtualization techniques that are not seen at the OS level.

He raised some questions at the end of his talk. J2EE applications run on JVMs which run on OSes, and each of these layers is performing virtualization—is this working? Could some of the redundancy be removed? Could these layers work better together or even be coalesced?

DYNAMIC COMPILATION TECHNIQUES

Summarized by Xing Fang

- ***Escape Analysis in the Context of Dynamic Compilation and Deoptimization***

Thomas Kotzmann and Hanspeter Mössenböck

Thomas Kotzmann presented an intra- and interprocedural escape analysis for a dynamic compiler. Escape analysis determines, for each object, whether it is accessible from within a single method, or one thread, or multiple threads. Method-local objects are eliminated and replaced with scalar variables. Thread-local objects are stack-allocated, and synchronization on them is removed.

The analyses and optimizations are implemented in Sun's Java HotSpot Client VM, which has a front end that operates on an SSA-based High-level Intermediate Representation (HIR). Escape analysis and scalar replacement are performed in parallel with the construction of the HIR. A state object containing a locals array is maintained by the compiler, to track the values most recently assigned to local variables. The state object also has a fields array which stores the current values of all fields.

An object is represented by its allocation instruction. The intraprocedural analysis parses instructions that might cause an object to escape, and updates the escape state of the instruction representing the object. Effects of various instructions on escape states were discussed. With SSA form, control flow is captured in phi-functions at the control merge points.

Test results about compilation time and machine code quality were performed and analyzed. The benefit is very evident for some benchmarks, most notably mtrt in SPECjvm98 and Monte Carlo in SciMark.

- ***Inlining Java Native Calls at Runtime***

Levon Stepanian, Angela Demke Brown, Allan Kielstra, Gita Koblents, and Kevin Stoodley

Levon Stepanian started out by observing that Java native calls are pervasive because they allow legacy, high-performance, or architecture-dependent native code to be integrated with Java applications. However, cross-language calls usually incur large time and space overheads, and this is true with JNI.

To reduce these overheads, the authors propose inlining JNI calls into Java applications with a JIT compiler. Both callouts (Java calls to functions implemented in external languages) and callbacks (external code accessing and modifying data and services from a running JVM) can be inlined. Work was done on the IBM TR JIT compiler, and the native functions exist in the form of W-code, the mature stack-based bytecode-like representation generated by IBM compiler front-ends.

Entire native functions are inlined at their call sites. For small native functions, removing the overhead of callouts can be a significant benefit. The benefit of transforming callbacks is even higher, with a minimum achieved speedup of nearly 12X in the micro-benchmark test cases. Inlining also appears to reduce the need for conservative assumptions about the behavior of native code in the JIT optimizer.

- ***Optimized Interval Splitting in a Linear Scan Register Allocator***

Christian Wimmer and Hanspeter Mössenböck

Linear scan register allocation is very suitable for JIT compilers because it is much faster than graph coloring and is nearly as effective. For each virtual register, lifetime intervals store the range of instructions where a value is active. Two intersecting intervals must not have the same register assigned. The algorithm assigns registers to values in a single linear pass over all intervals.

Three optimizations are proposed and implemented. Split positions are optimized to reduce the number of spill loads and stores at runtime. They are moved out of loops and into block boundaries. When two intervals are connected only by a move instruction, the interval of the move target stores the source of the move as its register hint. When possible, the target gets the same register allocated as the source, eliminating the register-to-register move. In two common cases that cover most of the intervals, moves inserted for spill stores can be removed.

Christian Wimmer presented the flow of the algorithm and used detailed examples to illustrate it. The algorithm is implemented for Sun's Java HotSpot Client VM. Results prove the efficiency of the optimized algorithm: the compilation time of the algorithm is nearly linear, and it is even faster than the original register allocation algorithm in the client compiler.

LANGUAGE REPRESENTATIONS

Summarized by Xing Fang

An Execution Layer for Aspect-Oriented Programming Languages

Michael Haupt, Mira Mezini, Christoph Bockisch, Tom Dinkelaker, Michael Eichberg, and Michael Krebs

Michael Haupt introduced the key concepts of the pointcut-and-advice (PA) flavor of Aspect Oriented Programming (AOP): join points, pointcuts, and advice. A join point is a point in the execution of a program, a pointcut is a query that quantifies over join points, defining related sets of join points, and an advice is a piece of functionality that can be attached to pointcuts, taking semantic effect when the respective pointcuts match.

According to Haupt, AOP language mechanisms, like OOP mechanisms, deserve implementation effort. AOP features have not gained sufficient support.

Currently, dispatching logic is inserted into application logic at compile or load time. This gap in semantics confuses debug efforts and incurs performance drawbacks. The contribution of the work is the integration of both the JPRM and the WM into the VM for supporting AspectJ's dynamic point model. The authors developed Steamloom, an extension to IBM's Jikes RVM that provides AOP functionality at the VM level, assessable through a Java API.

Evaluations show that the overhead incurred by the modification to implement Steamloom is practically zero, for hot runs. Class loading and method compilation overhead is about 7.8%. Other results show that using an AOP-enabled infrastructure does not in itself mean that execution is slowed down. Applied modifications of the original VM do not critically interfere with other subsystems. AOP-related functionality is more efficiently realizable at VM level.

Virtual Machine Showdown: Stack Versus Registers

Yunhe Shi, David Gregg, Andrew Beatty, and M. Anton Ertl,

A long-running question in the design of Virtual Machines is whether stack architecture or register architecture can be implemented more efficiently with an interpreter. David Gregg started off by noting that stack machines were more popular, because of the small code size and the ease of building stack machines. The JVMs and PERL 5 interpreter took this approach. But PERL 6 used a register machine instead, because register code was perceived as faster to interpret.

Execution of a VM instruction could be broken down into instruction dispatch, operand access, and actual computation times. Register code incurs less dispatch time than stack code, because of its fewer number of instructions. However, register code needs to access its operands explicitly so the code size is usually larger, resulting in more memory fetches for the code. Actual computation time is about the same for register and stack codes. Instruction dispatch is more costly than code fetching, so register code has the potential to be faster.

The authors built a much more sophisticated translator from the stack code to the register code. Results showed that the register code has 47% fewer instructions at a cost of a 25% increase in code size. Optimizations on the register code were very effective. 43.47% of static VM instructions were eliminated, as well as 47.21% of the dynamic VM code. On a Pentium 4, the register machine requires 32.3% less time than stack machine, with a less than perfect dispatch scheme. With a better dispatch, the reduction in execution time was still 26.5%. It is a very strong indication that the register architecture is superior to the stack architecture for implementing interpreter-based VMs.

Instrumenting Annotated Programs

Marina Biberstein, Vugranam C. Sreedhar, Bilha Mendelson, Daniel Citron, and Alberto Giammaria

Instrumentation is commonly used to collect program profile information. It is a spectative (i.e., one that observes the program behavior) program transformation and must maintain the program structure and functionality. Program annotation enables developers and tools to pass extra information to later stages of software development and execution. It is widely used in the CLR platform and has been adopted into the Java 1.5 standard.

Marina Biberstein gave a motivating example of two annotations that were both interfered with, although differently, by instrumentation. To solve the problem, instrumentation must handle different annotations in different ways. There must be active cooperation between the two.

The proposed solution takes an instrumentation-driven approach. Annotations are classified according to their behavior, and annotation writers would, for each annotation type, provide its description, in the form of meta-annotations. The descriptions provide information about the stage and lifetime of the annotation, its scope, sensitivity to instrumentation, and whether the annotation can be removed or healed. This information is passed on to instrumentation, which then bases its decisions on the information provided.

The authors proposed a taxonomy of annotations based on their study of more than two hundred live examples, which they used to classify annotations. They demonstrated their solution on a set of sample annotations.

DISTRIBUTED VEES

Summarized by Shuo Yang

- *PDS: A Virtual Execution Environment for Software Deployment*

Bowen Alpern, Joshua Auerbach, Vasanth Bala, Thomas Frauenhofer, Todd Mummert, and Michael Pigott

Joshua Auerbach presented a virtual machine solution, the Progressive Deployment System (PDS), to manage complex software deployment. The idea of PDS is to have software packaged, to provision, deploy, and execute software on customer machines, and to share software updates with many others (customers).

Auerbach gave an overview of PDS by introducing a working prototype under Windows XP: assets to be delivered include Eclipse, JBoss, and WebSphere. Using this example, he presented the architecture components of PDS and showed that PDS's virtual environment makes software look locally installed and resolves environment conflicts.

He talked about PDS's virtualizer. A process VM virtualizes the application binary interface (ABI), not hardware. Processes derived from the same asset are in the same VM. PDS uses a selective virtualization and only virtualizes OS calls that access the assets' virtually installed image. Auerbach concluded his talk by comparing the related work and presenting the differences with PDS and saying that PDS provides a feasible and convenient approach to deploying software packages.

- *The Entropia Virtual Machine for Desktop Grids*

Brad Calder, Andrew Chien, Ju Wang, and Don Yang

Brad Calder introduced the background of desktop distributed computing. The customers in this venue require the following features: desktop security, a clean execution environment, unobtrusiveness, application security, ease of application integration, lightweight VM installation, and low performance overhead.

He gave a system-architecture overview of the Entropia desktop computing system, which includes job management, resource management, and physical node management. Entropia virtual machines (EVMs) consist of: (1) a desktop controller to guarantee unobtrusiveness; and (2) a sandbox execution layer to provide security features. The sandbox layer takes two approaches to guarantee security: device driver mediation (with relatively high overhead) and binary interception (with low overhead). The sandboxing execution layer provides file virtualization (all files are accessed through a confined virtual file system located in an Entropia directory); file I/O throttling and automated file encryption; registry virtualization; GUI virtualization; network virtualization; and network I/O throttling.

Finally, he discussed the performance of jobs running on EVM. The talk concluded with an interesting discussion of market and customer demands.

- *HyperSpector: Virtual Distributed Monitoring Environments for Secure Intrusion Detection*

Kenichi Kourai and Shigeru Chiba

Kenichi Kourai presented a virtual distributed monitoring environment called HyperSpector, whose goal is to achieve secure intrusion detection in distributed computer systems.

He introduced the distributed intrusion detection systems (DIDs) and threats against DIDs by describing the behaviors and actions of active attacks and passive attacks. He then talked about the traditional approach to solving these challenges to DIDs—isolated monitoring—which is secure but needs additional hardware and supports only network-based IDSes (NIDSes).

He discussed the design of HyperSpector: it runs IDSes and server applications on separate VMs, and it builds a virtual network across the IDS VMs. HyperSpector provides three mechanisms: software port mirroring (packet capturing), inter-VM mounting (filesystem checking), and inter-VM mapping (process checking).

HyperSpector has been implemented in their Persona operating system, which is based on the FreeBSD 4.9 kernel. Their experimental results showed the effectiveness of their HyperSpector system design in terms of both security and overhead.

Steps to Reducing Unwanted Traffic on the Internet Workshop (SRUTI '05)

July 7, 2005, Cambridge, MA

Summarized by Jayanthkumar Kannan and Lakshminarayanan Subramanian, and edited by Balachander Krishnamurthy

SRUTI, a first-time USENIX workshop, sponsored by AT&T Labs, Cisco Systems, and the Department of Homeland Security, was attended by 55 people, and 13 peer-reviewed papers were presented.

DDOS AND WORMS

- *Using Routing and Tunneling to Combat DoS Attacks*

Adam Greenhalgh, Mark Handley, and Felipe Huici, University College London

The first session of the SRUTI workshop focused on different forms of network-level filtering mechanisms to defend against DDoS and worm attacks. The first paper argues that while many existing DoS defense mechanisms are hard to deploy, one can use a combination of routing and tunneling techniques to obtain a deployable DoS defense. The basic idea is to tunnel the traffic bound to a server across a fixed set of control points (edge routers in ISPs), which act as IP-level filtering gateways and use underlying routing protocols (e.g., I-BGP, E-BGP, OSPF) to signal information across different control points. The concept of using naming and path information as separate entities to force inspection at different control points is potentially applicable in other network security mechanisms.

- ***Reducing Unwanted Traffic in a Backbone Network***

Kuai Xu and Zhi-Li Zhang, University of Minnesota; Supratik Bhattacharyya, Sprint ATL

This paper shows how one can observe the communication patterns of end-hosts and use this information to determine unwanted traffic within the backbone of an ISP. The goal is to use the behavioral profile of each end-host based on IP header information and the Zipf-like nature of traffic characteristics to identify and filter the large sources of unwanted traffic. One open question remains: Under what constraints can good traffic be separated from bad traffic based only on observing the IP header information?

- ***Analyzing Cooperative Containment of Fast Scanning Worms***

Jayanthkumar Kannan, Lakshminarayanan Subramanian, Ion Stoica, and Randy H. Katz, University of California, Berkeley

The final paper in this session focused on analyzing the effectiveness of different cooperative strategies for worm containment, specifically, on the relationship between the type of signaling between firewalls and the level of containment. This paper illustrates that the signaling strategy essential for good containment depends on various factors, including the reproduction rate of the worm (i.e., the number of new hosts one vulnerable host affects), the level of malice, and the extent of deployment. How to generate robust and succinct worm filters with a low false-positive probability remains a goal for future work.

SPAM-1

- ***Push vs. Pull: Implications of Protocol Design on Controlling Unwanted Traffic***

Zhenhai Duan and Kartik Gopalan, Florida State University; Yingfei Dong, University of Hawaii

The second session was the first of two that focused on spam evasion and detection. This paper proposes a simple design principle for communication protocols that help participants avoid unwanted traffic. The main observation is that a receiver-pull approach is superior to a sender-push approach in the degree of control offered to a recipient. However, in some applications, such as email, a pure receiver-pull approach is not possible, since communication is initiated by the sender. For such applications, a sender-intent receiver-pull approach is proposed, where the sender first sends a short intent-to-send message, on the basis of which the receiver makes the decision to accept or reject the message. The principal advantage of this approach is the potential bandwidth savings, since the receiver does not need to download the entire message. As pointed out by one workshop participant, this basic idea has been proposed before, but this paper suggests a way of implementing it using simple extensions to SMTP.

- ***Detecting Spam in VoIP Networks***

Ram Dantu and Prakash Kolan, University of North Texas, Denton

This paper deals with the problem of spam detection in VoIP networks. VoIP spam is likely to be more irritating to users than email spam, since VoIP is synchronous. In VoIP spam detection, the decision about spam potential has to be made using only the initial context of the message and cannot be dependent on the content of the entire message. The paper proposes a multi-stage VoIP spam identification mechanism that involves several building blocks such as Bayesian detection, rate limiting, and blacklisting. This mechanism also leverages the social network of caller-callee relationships in deducing the reputation of a caller.

BOTS AND SPOOFED SOURCES

- ***The Zombie Roundup: Understanding, Detecting, and Disrupting Botnets***

Evan Cooke and Farnam Jahanian, University of Michigan; Danny McPherson, Arbor Networks

A common message from this session was the need for security developers to share information in order to keep pace with the growing sophistication of Internet attacks.

The first paper illustrates the prevalence of bots on the Internet, where thousands of new bots show up on a daily basis, and it describes different techniques for detecting and disrupting botnets. Among the different detection strategies, this paper stresses the need for a behavioral methodology for analyzing IRC traffic from end-hosts to detect bot communication. One challenge in measuring the prevalence of bots is that one needs to be part of several botnets to perform such measurements, raising legal issues.

- ***An Architecture for Developing Behavioral History***

Mark Allman and Vern Paxson, International Computer Science Institute; Ethan Blanton, Purdue University

This paper examines how architecture can aid in determining the sources of unwanted traffic where the identity of a source can be in different granularities (e.g., email, end-host). The grand vision is to build a repository that consists of the sources of different forms of malicious traffic, with the challenges that the architecture be scalable, open system, distributed, robust, abe to handle various types of traffic, and policy neutral. To detect bogus information in this repository, one would need audit trails for evidence and the ability to assess the reputation of reporters and corroborate different entries for correctness in the system.

- ***The Spoofer Project: Inferring the Extent of Internet Source Address Filtering on the Internet***

  Robert Beverly and Steve Bauer, MIT

  The final paper in this session describes a measurement study to quantify the extent and nature of source address filtering. Among the important findings are that a significant number of netblocks allow some form of spoofing, filtering is applied inconsistently, filtering policies correspond to netblocks in BGP, and no specific geographic patterns abound in spoofing.

ADAPTIVE DEFENSE SYSTEMS

- ***Stress Testing Traffic to Infer Its Legitimacy***

  Nick Duffield and Balachander Krishnamurthy, AT&T Labs—Research

  This paper proposes stress testing as a general approach to distinguish between legitimate and malicious traffic. By inducing artificial impediments to traffic and examining the reaction of the sender, one can deduce whether the traffic is malicious. This idea is predicated on two points: (1) differentiation: response to impairment differs between malicious traffic and legitimate traffic; and (2) recovery: legitimate traffic can deal with impairments. They examine the applicability of these principles in different domains, such as TCP, HTTP, UDP, SMTP, and BGP. The extent to which a TCP sender backs off in response to an induced loss can be used as a metric of the malice of the sender. The frequency of HTTP connection establishment in response to a "Service unavailable" message can be used similarly. The authors are also working on evaluation of these techniques over normal traffic. One comment by an audience member was that stress testing may lead to an increase in the total traffic under certain conditions.

- ***Adaptive Defense Against Various Network Attacks***

  Cliff C. Zou, University of Massachusetts; Nick Duffield, AT&T Labs—Research; Don Towsley and Weibo Gong, University of Massachusetts

  This paper proposes a general way to adaptively tune an attack detection mechanism in response to the volume of attack traffic. The basic idea is to periodically vary parameters in a detection mechanism so as to optimize an objective function that includes penalties for missed attacks (false negatives) and incorrect alarms (false positives). This is based on the intuitive observation that a higher false positive probability is tolerable during periods of high attack. This technique is applied to two detection mechanisms known in literature: the hop-count filtering method for detecting spoofed SYN flood attacks, and the threshold random walk for defending against worms. This paper provoked considerable discussion among attendees regarding the pros and cons of such adaptive defense techniques. While it is clear that smart attacks (such as pulsed DoS attacks) are still viable against adaptive defense mechanisms, it was generally agreed that adaptive defense would reduce the impact of the attack.

SPAM-2 AND ENCRYPTION

- ***HoneySpam: Honeypots Fighting Spam at the Source***

  Mauro Andreolini, Alessandro Bulgarelli, Michele Colajanni, Francesca Mazzoni, and Luca Messori, Università di Modena e Reggio Emilia

  The final session dealt with email spam detection and encryption mechanisms. The first paper described the architecture of HoneySpam, a honeypot implementation to reduce spam. The goal is counter-cultural in that it encourages spammers to use the system to send spam so that HoneySpam can then identify the spammers, traffic-shape them, and provide them with incorrect information to hinder their progress. The challenge is to hide the identity and location of the HoneySpam system.

- ***Improving Spam Detection Based on Structural Similarity***

  Luiz H. Gomes, Fernando D.O. Castro, Virgilio A.F. Almeida, Jussara M. Almeida, and Rodrigo B. Almeida, Universidade Federal de Minas Gerais; Luis M.A. Bettencourt, Los Alamos National Laboratory

  This paper deals with improving traditional spam detection algorithms using information regarding the social networks of the sender and the recipient. All senders are grouped into clusters based on the similarity of the recipients they send mail to. Similarly, receivers are grouped into clusters based on the senders who have contacted them in the past. The probability that a particular email is spam is computed based on the extent to which the sender's (recipient's) cluster have sent (received) spam in the past. This decision is used to augment a Bayesian classifier, and the results demonstrate that false positives are reduced, but not by a significant amount. A question on the scalability of the system to several thousands of senders/receivers was raised, and the author suggested schemes like LRU aging to deal with this issue.

- ***Lightweight Encryption for Email***

  Ben Adida, Susan Hohenberger, and Ronald L. Rivest, MIT

  The final paper leverages identity-based encryption (IBE) techniques for easing the use of encrypted email. The basic idea is to leverage DNS as a distribution mechanism for public keys at the domain level. In IBE, a sender can use the recipient's email address along with a master public key (MPK) to derive the recipient's public key. The paper suggests that each email domain should designate a set of key servers that would generate an MPK jointly and distribute it via DNS. These key servers would communicate the secret key for an email address in their domain by simply sending it via email. For additional security, a recipient could also publish a second public key on a broadcast channel. The security of this scheme is dependent on that of DNS and the channel between the key server and the recipient.

One comment raised was that the ease of deriving the public key for a particular recipient might also allow a spammer to encrypt messages and render them unreadable by spam filters.

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.

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 must grant permission for any third party to 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 will 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.

PROFESSORS, CAMPUS STAFF, AND STUDENTS— DO YOU HAVE A USENIX REPRESENTATIVE ON YOUR CAMPUS? IF NOT, USENIX IS INTERESTED IN HAVING ONE AT YOUR UNIVERSITY!

The USENIX University Outreach Program is a network of representatives at campuses around the world who provide Association information to students, and encourage student involvement in USENIX. This is a volunteer program, for which USENIX is always looking for academics to participate. The program is designed for faculty who directly interact with students. We fund one representative from a campus at a time. In return for service as a campus representative, we offer a complimentary membership and other benefits.

A liaison's responsibilities include:

- Maintaining a library (online and in print) of USENIX publications at your university for student use
- Distributing calls for papers and upcoming event brochures, and re-distributing informational emails from USENIX
- Encouraging students to apply for travel stipends to conferences
- Providing students who wish to join USENIX with information and applications
- Helping students to submit research papers to relevant USENIX conferences
- Providing USENIX with feedback and suggestions on how the organization can better serve students

In return for being our "eyes and ears" on campus, liaisons receive a complimentary membership in USENIX with all membership benefits (except voting rights), and a free conference registration once a year (after one full year of service as a campus liaison).

To qualify as a campus representative, you must:

- Be full-time faculty or staff at a four year accredited university
- Have been a dues- paying member of USENIX for at least one full year in the past

For more information about our Student Programs, see
http://www.usenix.org/students

USENIX contact: Tara Mulligan, Scholastic Programs Manager, tara@usenix.org

Announcement and Call for Papers USENIX

2006 USENIX Annual Technical Conference: Systems Practice & Experience Track

Formerly the General Session Refereed Papers Track

http://www.usenix.org/usenix06

Training Program: Tuesday–Saturday, May 30–June 3, 2006
Technical Sessions: Thursday–Saturday, June 1–3, 2006

Boston, Massachusetts, USA

Important Dates

Paper submissions due: *Tuesday, January 17, 2006 (hard deadline)*
Notification to authors: *Monday, February 27, 2006*
Final papers due: *Monday, April 17, 2006*
Poster submissions due: *Monday, April 24, 2006*

Program Committee

Program Co-Chairs
Atul Adya, *Microsoft*
Erich Nahum, *IBM T.J. Watson Research Center*

Program Committee
Steven Bellovin, *Columbia University*
Ranjita Bhagwan, *IBM T.J. Watson Research Center*
Jeff Chase, *Duke University*
Mike Chen, *Intel Research, Seattle*
Jason Flinn, *University of Michigan*
Steven Hand, *University of Cambridge*
Gernot Heiser, *University of New South Wales and National ICT Australia*
Kim Keeton, *Hewlett-Packard*
Dejan Kostic, *EPFL*
Jay Lepreau, *University of Utah*
Barbara Liskov, *Massachusetts Institute of Technology*
Jason Nieh, *Columbia University*
Vivek Pai, *Princeton University*
Dave Presotto, *Google*
John Reumann, *Google*
Mendel Rosenblum, *Stanford University*
Stefan Saroiu, *University of Toronto*
Geoff Voelker, *University of California, San Diego*
Alec Wolman, *Microsoft Research*
Yuanyuan Zhou, *University of Illinois at Urbana-Champaign*

Overview

Authors are invited to submit original and innovative papers to the Systems Practice & Experience Track (formerly the General Session Refereed Papers Track) of the 2006 USENIX Annual Technical Conference. We seek high-quality submissions that further the knowledge and understanding of modern computing systems, with an emphasis on practical implementations and experimental results. We encourage papers that break new ground or present insightful results based on experience with computer systems. The USENIX conference has a broad scope, and we encourage papers in a wide range of topics in systems

Topics

Specific topics of interest include but are not limited to:

- Architectural interaction
- Benchmarking
- Deployment experience
- Distributed and parallel systems
- Embedded systems
- Energy/power management
- File and storage systems
- Networking and network services
- Operating systems
- Reliability, availability, and scalability
- Security, privacy, and trust
- Self-managing systems
- Usage studies and workload characterization
- Virtualization
- Web technology
- Wireless and mobile systems

Best Paper Awards

Cash prizes will be awarded to the best papers at the conference. Please see http://www.usenix.org /publications/library/proceedings/best_papers.html for examples of Best Papers from previous years.

How to Submit

Authors are required to submit full papers by 11:59 p.m. PDT, Tuesday, January 17, 2006. *This is a hard deadline; absolutely no extensions will be given.*

All submissions for USENIX '06 will be electronic, in PDF format, via a Web form on the conference Web site. Authors will be notified of receipt of submission via email. USENIX '06 will accept two types of papers:

- **Regular Papers:** Submitted papers must be no longer than 14 single-spaced pages, including figures, tables, and references, using 10 point font or larger. The first page of the paper should include the paper title and author name(s); reviewing is not blind. Papers longer than 14 pages will not be reviewed.
- **Short Papers:** Authors may submit short papers, at most 6 pages long. These will be reviewed, accepted submissions will be included in the Proceedings, and time will be provided in the Short Papers Sessions for brief presentations of these papers. We expect that this format will appeal to authors who wish to publicize early ideas, convey results that do not require a full-length paper, or advocate new positions.

In addition, the program committee may accept some standard submissions as 6-page short papers if they feel the submission is interesting but does not meet the criteria of a full-length paper. Please indicate explicitly if you do not wish your full-length paper to be considered for the Short Papers Sessions. Papers accepted for the Short Papers Sessions will automatically be included in the Poster Session.

Specific questions about submissions may be sent to usenix06chairs@usenix.org.

In a good paper, the authors will have:

- attacked a significant problem
- devised an interesting and practical solution
- clearly described what they have and have not implemented
- demonstrated the benefits of their solution
- articulated the advances beyond previous work
- drawn appropriate conclusions

Simultaneous submission of the same work to multiple venues, submission of previously published work, and plagiarism constitute dishonesty or fraud. The USENIX Annual Technical Conference, 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, the program committee 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.

Papers accompanied by nondisclosure agreements cannot be accepted. All submissions are held in the highest confidentiality prior to publication in the Proceedings, both as a matter of policy and in accord with the U.S. Copyright Act of 1976.

Authors will be notified of paper acceptance or rejection by Monday, February 27, 2006. Accepted papers will be shepherded by a program committee member. Final papers must be no longer than 14 pages, formatted in 2 columns, using 10 point Times Roman type on 12 point leading, in a text block of 6.5" by 9".

Note regarding registration: One author per paper will receive a registration discount of $200. USENIX will offer a complimentary registration upon request.

Poster Session

A poster session will also be held, in conjunction with the reception, where researchers can present recent and ongoing projects. The poster session will be a good forum to discuss new ideas and get useful feedback from the community. The poster submissions should include a brief description of the research idea(s); the submission must not exceed 2 pages. Accepted posters will be put on the conference Web site; however, they will not be printed in the conference Proceedings. Send poster submissions to usenix06posters@usenix.org by Monday, April 24, 2006.

Announcement and Call for Papers USENIX

15th USENIX Security Symposium

http://www.usenix.org/sec06

July 31–August 4, 2006 **Vancouver, B.C., Canada**

Important Dates

Paper submissions due: *February 1, 2006, 11:59 p.m. PST*
Panel proposals due: *March 29, 2006*
Notification to authors: *April 3, 2006*
Final papers due: *May 11, 2006*

Symposium Organizers

Program Chair
Angelos D. Keromytis, *Columbia University*

Program Committee
William Arbaugh, *University of Maryland*
Lee Badger, *DARPA*
Peter Chen, *University of Michigan*
Bill Cheswick, *Lumeta*
Marc Dacier, *Eurecom, France*
Ed Felten, *Princeton University*
Virgil Gligor, *University of Maryland*
John Ioannidis, *Columbia University*
Trent Jaeger, *Pennsylvania State University*
Somesh Jha, *University of Wisconsin*
Louis Kruger, *University of Wisconsin*
Wenke Lee, *Georgia Institute of Technology*
Fabian Monrose, *Johns Hopkins University*
Andrew Myers, *Cornell University*
Vassilis Prevelakis, *Drexel University*
Niels Provos, *Google*
Michael Reiter, *Carnegie Mellon University*
Michael Roe, *Microsoft Research, UK*
R. Sekar, *Stony Brook University*
Anil Somayaji, *Carleton University*
Jessica Staddon, *PARC*
Salvatore Stolfo, *Columbia University*
David Wagner, *University of California, Berkeley*
Brian Weis, *Cisco*
Tara Whalen, *Dalhousie University*

Invited Talks Co-Chairs
Patrick McDaniel, *Pennsylvania State University*
Gary McGraw, *Cigital*

Work-in-Progress Session Chair
Doug Szajda, *University of Richmond*

Symposium Overview

The USENIX Security Symposium brings together research-ers, practitioners, system administrators, system programmers, and others interested in the latest advances in the security of computer systems and networks. The 15th USENIX Security Symposium will be held July 31–August 4, 2006, in Vancouver, B.C., Canada.

All researchers are encouraged to submit papers covering novel and scientifically significant practical works in security or applied cryptography. Submissions are due on February 1, 2006, 11:59 p.m. PST. The Symposium will span five days: a training program will be followed by a two and one-half day technical program, which will include refereed papers, invited talks, Work-in-Progress reports, panel discussions, and Birds-of-a-Feather sessions.

New in 2006, a workshop, titled Hot Topics in Security (HotSec '06), will be held in conjunction with the main conference. More details will be announced soon on the USENIX Web site, http://www.usenix.org.

Symposium Topics

Refereed paper submissions are solicited in all areas relating to systems and network security, including:

- Adaptive security and system management
- Analysis of network and security protocols
- Applications of cryptographic techniques
- Attacks against networks and machines
- Authentication and authorization of users, systems, and applications
- Automated tools for source code analysis
- Cryptographic implementation analysis and construction
- Defenses against malicious code (worms, viruses, trojans, spyware, etc.)
- Denial-of-service attacks and countermeasures
- File and filesystem security
- Firewall technologies
- Forensics and diagnostics for security
- Intrusion and anomaly detection and prevention
- Network infrastructure security
- Operating system security
- Privacy-preserving (and compromising) systems
- Public key infrastructure
- Rights management and copyright protection
- Security of agents and mobile code
- Security architectures
- Security in heterogeneous and large-scale environments
- Security policy
- Self-protecting and healing systems
- Techniques for developing secure systems
- Voting systems analysis and security
- Wireless and pervasive/ubiquitous computing security
- World Wide Web security

Note that the USENIX Security Symposium is primarily a systems security conference. Papers whose contributions are primarily new cryptographic algorithms or protocols, crypt-analysis, electronic commerce primitives, etc., may not be appropriate for this conference.

Refereed Papers & Awards

Papers which have been formally reviewed and accepted will be presented during the Symposium and published in the Symposium Proceedings. It is expected that one of the paper authors will attend the conference and present the work. It is the responsibility of the authors to find a suitable replacement presenter for their work, if the need arises.

The Proceedings will be distributed to attendees and, following the Symposium, will be available online to USENIX members and for purchase.

One author per paper will receive a registration discount of $200. USENIX will offer a complimentary registration upon request.

Awards may be given at the conference for the best overall paper and for the best paper for which a student is the lead author. Papers by program committee members are not eligible for these awards.

Training Program, Invited Talks, Panels, WiPs, and BoFs

In addition to the refereed papers and the keynote presentation, the Symposium will include a training program, invited talks, panel discussions, Work-in-Progress reports (WiPs), and Birds-of-a-Feather sessions (BoFs). You are invited to make suggestions regarding topics or speakers in any of these sessions via email to the contacts listed below or to the program chair at sec06chair@usenix.org.

Training Program

Tutorials for both technical staff and managers will provide immediately useful, practical information on topics such as local and network security precautions, what cryptography can and cannot do, security mechanisms and policies, firewalls, and monitoring systems. If you are interested in proposing a tutorial or suggesting a topic, contact the USENIX Training Program Coordinator, Dan Klein, by email to dvk@usenix.org.

Invited Talks

There will be several outstanding invited talks in parallel with the refereed papers. Please submit topic suggestions and talk proposals via email to sec06it@usenix.org.

Panel Discussions

The technical sessions may include topical panel discussions. Please send topic suggestions and proposals to sec06chair@usenix.org. The deadline for panel proposals is March 29, 2006.

Work-in-Progress Reports (WiPs)

The last session of the Symposium will consist of Work-in-Progress reports (WiPs). This session offers short presentations about work in progress, new results, or timely topics. Speakers should submit a one- or two-paragraph abstract to sec06wips@usenix.org by 6:00 p.m. PDT on Wednesday, August 2, 2006. Make sure to include your name, your affiliation, and the title of your talk. The accepted abstracts and session schedule will be posted on the conference Web site. The time available will be distributed among the presenters, with each speaker allocated between 5 and 10 minutes. The time limit will be strictly enforced.

Birds-of-a-Feather Sessions (BoFs)

Birds-of-a-Feather sessions (BoFs) will be held Tuesday, Wednesday, and Thursday evenings. Birds-of-a-Feather sessions are informal gatherings of persons interested in a particular topic. BoFs often feature a presentation or a demonstration followed by discussion, announcements, and the sharing of strategies. BoFs can be scheduled onsite or in advance. To preschedule a BoF, please send email to bofs@usenix.org with the title and a brief description of the BoF; the name, title, affiliation, and email address of the facilitator; and your preference of date and time.

Paper Submission Instructions

Papers are due by February 1, 2006, 11:59 p.m. PST. All submissions will be made online, and details of the submissions process will be made available on the conference Web site, http://www.usenix.org/events/sec06/cfp, well in advance of the deadline. Submissions should be finished, complete papers. Paper submissions should be about 10 to a maximum of 20 typeset pages, formatted in a single column, using 11 point Times Roman type on 12 point leading, in a text block of 6.5" by 9" (default LaTeX 11 point single-column article format is acceptable). Reviewers may not take into consideration any portion of a submission that is over the stated limit. Once accepted, papers must be reformatted to be about 8 to a maximum of 16 typeset pages, formatted in 2 columns, using 10 point Times Roman type on 12 point leading, in a text block of 6.5" by 9".

Paper submissions ***must not*** be anonymized.

Submissions must be in PDF format. Please make sure your submission can be opened using Adobe Acrobat 4.0. For more details on the submission process, consult the detailed author guidelines.

To insure that we can read your PDF file, authors are urged to follow the NSF "Fastlane" guidelines for document preparation and to pay special attention to unusual fonts. For more details, see:

- https://www.fastlane.nsf.gov/documents/pdf_create/pdfcreate_01.jsp
- https://www.fastlane.nsf.gov/documents/tex/tex_01.jsp

All submissions will be judged on originality, relevance, correctness, and clarity. The USENIX Security Symposium, like most conferences and journals, requires that papers not be submitted simultaneously to another conference or publication and that submitted papers not be previously published elsewhere, or subsequently published within 12 months of acceptance at the Symposium. In addition to citing relevant, published work, authors should relate their submission to any other relevant submission of theirs in other venues that are under review at the same time as their submissions to the Symposium.

We may share information about submissions with the program chairs of other conferences considering papers during the review period. In case of a double submission or a similar violation, we will automatically reject the specific submission as well as any other submissions by the authors of the offending paper, and reserve the right to take further action.

Papers accompanied by nondisclosure agreement forms will not be considered. All submissions are treated as confidential, both as a matter of policy and in accordance with the U.S. Copyright Act of 1976.

Authors will be notified of acceptance by April 3, 2006. The final paper due date is May 11, 2006. Each accepted submission may be assigned a member of the program committee to act as its shepherd through the preparation of the final paper. The assigned member will act as a conduit for feedback from the committee to the authors.

Specific questions about submissions may be sent via email to the program chair at sec06chair@usenix.org.

Program and Registration Information

Complete program and registration information will be available in May 2006 on the Symposium Web site, both as HTML and as a printable PDF file. If you would like to receive the latest USENIX conference information, please join our mailing list at http://www.usenix.org/about/mailing.html.

LISA'05

19TH LARGE INSTALLATION SYSTEM ADMINISTRATION CONFERENCE

DEC. 4–9, 2005
SAN DIEGO, CA

www.usenix.org/lisa05

FIND THE MISSING PIECES
TO YOUR TOUGHEST PUZZLES.

6 DAYS OF TRAINING
by industry experts, including:

- Rik Farrow on Hands-on Linux Security
- Don Bailey on 802.11 Wireless Network Penetration Testing
- Richard Bejtlich on Network Incident Response
- Jacob Farmer on Disk-to-Disk Backup

⇨ **LISA '05 offers the most in-depth, real-world system administration training available!**

3-DAY TECHNICAL PROGRAM

KEYNOTE: Qi Lu, Vice President of Engineering, Yahoo! Inc., on "Scaling Search Beyond the Public Web"

2 TRACKS OF INVITED TALKS, INCLUDING:

- Matt Blaze, University of Pennsylvania: "Picking Locks with Cryptology"
- Kevin Bankston, EFF: "How Sysadmins Can Protect Free Speech and Privacy on the Electronic Frontier"

NEW! Hit the Ground Running Track

Refereed Papers, Guru Is In Sessions, Vendor Exhibition, BoFs, WiPs, and more!

Sponsored by USENIX and SAGE

USENIX 30th ANNIVERSARY

Check out the Web site for more information!
http://www.usenix.org/lisa05

;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