



OPINION

Musings
RIK FARROW

PROGRAMMING

Algorithms for the 21st Century
STEPHEN C. JOHNSON
Maybe You Should Use Python
MIKE HOWARD

SYSADMIN

Configuration Management: Models and Myths, Part 2
MARK BURGESS
It's About Time ...
BRAD KNOWLES

TECHNOLOGY

Multi-Core Microprocessors Are Here
RICHARD MCDUGALL AND JAMES LAUDON
Measuring Performance of FreeBSD Disk Encryption
TIMO SIVONEN

NETWORK

Automating Server Selection with OASIS
MICHAEL J. FREEDMAN
WISPER: Open Source, Long-Distance Wireless
RIK FARROW

COLUMNS

Practical Perl Tools: Tie Me Up, Tie Me Down (Part 2)
DAVID BLANK-EDELMAN
ISPadmin: Anti-Spam Roundup
ROBERT HASKINS
Echo in VoIP Systems
HEISON CHAK
/dev/random
ROBERT G. FERRELL

BOOK REVIEWS

Book Reviews
ELIZABETH ZWICKY ET AL.

STANDARDS

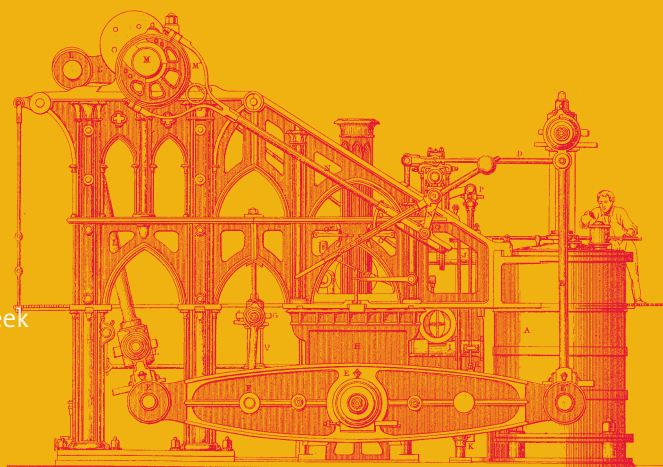
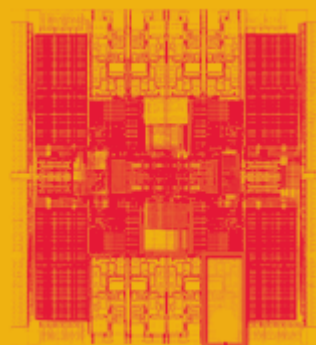
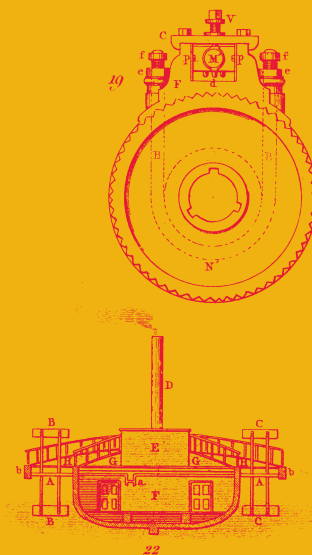
An Update on Standards: Diary of a Standard Geek
NICK STOUGHTON
Why Get Involved in POSIX?
ANDREW JOSEY

USENIX NOTES

USACO News
ROB KOLSTAD
SAGE Update
STRATA CHALUP

CONFERENCES

Annual Tech '06; SRUTI '06; EVT '06



USENIX

The Advanced Computing Systems Association

USENIX

Upcoming Events



INTERNET MEASUREMENT CONFERENCE 2006 (IMC 2006)

Sponsored by ACM SIGCOMM in cooperation with USENIX
OCTOBER 25–27, 2006, RIO DE JANEIRO, BRAZIL
<http://www.imconf.net/imc-2006/>

3RD WORKSHOP ON REAL, LARGE DISTRIBUTED SYSTEMS (WORLDS '06)

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

7TH USENIX SYMPOSIUM ON OPERATING SYSTEMS DESIGN AND IMPLEMENTATION (OSDI '06)

Sponsored by USENIX in cooperation with ACM SIGOPS
NOVEMBER 6–8, 2006, SEATTLE, WA, USA
<http://www.usenix.org/osdi06>

SECOND WORKSHOP ON HOT TOPICS IN SYSTEM DEPENDABILITY (HOTDEP '06)

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

ACM/IFIP/USENIX 7TH INTERNATIONAL MIDDLEWARE CONFERENCE

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

20TH LARGE INSTALLATION SYSTEM ADMINISTRATION CONFERENCE (LISA '06)

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

5TH USENIX CONFERENCE ON FILE AND STORAGE TECHNOLOGIES (FAST '07)

Sponsored by USENIX in cooperation with ACM SIGOPS, IEEE Mass Storage Systems Technical Committee, and IEEE TCOS
FEBRUARY 13–16, 2007, SAN JOSE, CA, USA
<http://www.usenix.org/fast07>

1ST SYMPOSIUM ON COMPUTER HUMAN INTERACTION FOR MANAGEMENT OF INFORMATION TECHNOLOGY (CHIMIT '07)

Sponsored by ACM in cooperation with USENIX
MARCH 30–31, 2007, CAMBRIDGE, MA, USA
<http://chimit.cs.tufts.edu>

SECOND WORKSHOP ON TACKLING COMPUTER SYSTEMS PROBLEMS WITH MACHINE LEARNING TECHNIQUES (SYSML '07)

Co-located with NSDI '07
APRIL 10, 2007, CAMBRIDGE, MA, USA
<http://www.usenix.org/usenix07>
Paper submissions due: November 20, 2006

4TH SYMPOSIUM ON NETWORKED SYSTEMS DESIGN AND IMPLEMENTATION (NSDI '07)

Sponsored by USENIX in cooperation with ACM SIGCOMM and ACM SIGOPS
APRIL 11–13, 2007, CAMBRIDGE, MA, USA
<http://www.usenix.org/nsdi07>
Paper submissions due: October 9, 2006

11TH WORKSHOP ON HOT TOPICS IN OPERATING SYSTEMS (HOTOS XI)

Sponsored by USENIX in cooperation with the IEEE Technical Committee on Operating Systems (TCOS)
MAY 7–9, 2007, SAN DIEGO, CA, USA
<http://www.usenix.org/hotos07>
Paper submissions due: January 4, 2007

THIRD INTERNATIONAL ACM SIGPLAN/SIGOPS CONFERENCE ON VIRTUAL EXECUTION ENVIRONMENTS (VEE '07)

Sponsored by ACM SIGPLAN and SIGOPS in cooperation with USENIX
JUNE 13–15, 2007, SAN DIEGO, CA, USA
<http://vee07.cs.ucsb.edu>
Paper submissions due: February 5, 2007

2007 USENIX ANNUAL TECHNICAL CONFERENCE

JUNE 17–22, 2007, SANTA CLARA, CA, USA
<http://www.usenix.org/usenix07>
Paper submissions due: January 9, 2007

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

contents



VOL. 31, #5, OCTOBER 2006

EDITOR

Rik Farrow
rik@usenix.org

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

MANAGING EDITOR

Jane-Ellen Long
jel@usenix.org

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

COPY EDITOR

David Couzens
proofshop@usenix.org

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

PRODUCTION

Lisa Camp de Avalos
Casey Henderson

TYPESETTER

Star Type
startype@comcast.net

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

USENIX ASSOCIATION

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

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

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

©2006 USENIX Association.

USENIX is a registered trademark of the USENIX Association. Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. USENIX acknowledges all trademarks herein. Where those designations appear in this publication and USENIX is aware of a trademark claim, the designations have been printed in caps or initial caps.

OPINION

2 Musings
RIK FARROW

PROGRAMMING

6 Algorithms for the 21st Century
STEPHEN C. JOHNSON

13 Maybe You Should Use Python
MIKE HOWARD

SYSADMIN

18 Configuration Management: Models and Myths. Part 2: Babel, Babble, Toil, and Grammar
MARK BURGESS

25 It's About Time . . .
BRAD KNOWLES

TECHNOLOGY

32 Multi-Core Microprocessors Are Here
RICHARD MCDUGALL AND JAMES LAUDON

40 Measuring Performance of FreeBSD Disk Encryption
TIMO SIVONEN

NETWORK

46 Automating Server Selection with OASIS
MICHAEL J. FREDMAN

53 WISPER: Open Source, Long-Distance Wireless
RIK FARROW

COLUMNS

57 Practical Perl Tools: Tie Me Up, Tie Me Down (Part 2)
DAVID BLANK-EDELMAN

65 ISPadmin: Anti-Spam Roundup
ROBERT HASKINS

69 Echo in VoIP Systems
HEISON CHAK

72 /dev/random
ROBERT G. FERRELL

BOOK REVIEWS

75 Book Reviews
ELIZABETH ZWICKY ET AL.

STANDARDS REPORTS

78 An Update on Standards: Diary of a Standard Geek
NICK STOUGHTON

80 Why Get Involved in POSIX?
ANDREW JOSEY

USENIX NOTES

82 USACO News
ROB KOLSTAD

83 SAGE Update
STRATA CHALUP

84 Writing for *;*login:

CONFERENCE REPORTS

85 Annual Tech '06

104 SRUTI '06

107 EVT '06

RIK FARROW

musings



rik@usenix.org

SOME GUY FROM QUALCOMM TOLD

me, during the USENIX Security program committee party in Vancouver, that he was amazed that I could come up with so many columns. Sometimes I wonder about that too, as I strive not to repeat myself or to travel down already well-worn paths. Just like following a rutted road, it is all too easy to follow the groove.

I've just returned from the USENIX Security conference in Vancouver, the fifteenth such conference, and, as usual, I find myself depressed. Not just because the conference, which itself was a lot of fun, is over, but because not enough has changed.

True, some students showed how they could build a simple device that was able to monitor the keyboard serial cable for possible username/password combinations. The students came up with a scheme that encoded passwords by varying the time between keystrokes by 20 milliseconds, adding some framing, and repeating the same sequence of timings. If those keystrokes travel across a network, even as part of an encrypted SSH connection, they will expose the password.

Isn't anything safe? No, not really, but there certainly are ways that we could make things better. Sitting in on the 2006 USENIX/ACCURATE Electronic Voting Technology Workshop (www.usenix.org/events/evt06), I learned that cryptographic techniques which would make voting more accurate have existed for many years. In his presentation, Josh Benaloh of Microsoft Research said that the hard problems in voting, such as accurately recording votes, accurately counting those votes, and providing the voter with proof that her vote was recorded as cast without revealing the way she voted, are manna to cryptographers. Give them a difficult problem, and they eat it up. So now we have a choice of solutions, not just one. Are we using them? Not in the U.S.

Joseph Lorenzo Hall, a lawyer at UC Berkeley, suggested that we need to add a new category to open source licensing—"open disclosure." "Open disclosure" means that vendors maintain both control of and the license to their software, but openly disclose the code so that it can be checked for correctness. This sounds like a great idea, especially in countries where everything gets done for a profit, from prisons to hospitals, and even voting machines. But there are flies in this oint-

ment, as some voting vendors' code is so ugly that they would likely drop out of the business sooner than reveal their code. As he said this, I heard mumblings in the room from people who had actually had permission to audit DRE (Direct Recording Election) machine code, quietly agreeing with him about the crappiness of the code used in popular voting machines.

Another talk really knocked my socks off (yes, I was wearing shoes). Ka-Ping Yee, also of UC Berkeley, talked about work he had done with David Wagner, Marti Hearst, and Steve Bellovin to create voting software. They had split the voting code into two parts: the set of images that represent the ballots and screens that make up the user interface, and the code that interprets the voter's input. Each image, representing a particular contest for, say, corporate commissioners, can be rendered in advance, tested, approved, and digitally signed. The logic that ties the images together, creating a flow from one screen to the next, can also be verified through testing. Sounds pretty simple, and it actually is. Instead of the 31,000+ lines of C++ code sitting on top of the Microsoft Foundation Libraries and Windows CE that's found in Diebold's DRE system, Ping and his friends wrote their functional e-voting software in 293 lines of Python and just a couple of libraries. Hmmm, seems like that just might be short enough to audit.

There was much more going on during EVT 2006: read the summaries in this issue of *;login:*. I left knowing that it was definitely possible to create trustworthy voting systems and that Europe, Australia, India, and other countries had successfully worked with e-voting systems, but my own country, the U.S., had decided to stick with systems known to be broken or ones still sealed in secrecy.

illumination

There were certainly bright notes during the Security conference: for instance, Andy Ozment presented a paper showing that code, at least OpenBSD code, actually has been getting better. He and Stuart Schechter showed that most of the security problems found in OpenBSD came from code inherited when OpenBSD was forked from NetBSD. These bugs continued to be found for many years, while a much smaller amount of new code was found to contain bugs as well. Finally, a ray of hope, and one that might apply to other code bases as well.

But I'm still depressed. Unsurprisingly, no new technique for guaranteeing the security of any computer system appeared. Instead, we are besieged by growing complexity. Sure, you can still strip down a UNIX, Linux, or BSD system to its bare minimum and reduce your risk factor enormously. The version of Linux to be used in the One Laptop per Child (laptop.org) project will be extremely stripped-down, as befits an OS designed for a single hardware target (no extra device drivers) to be used by children, with no system administration needed. There are still some floppy-disk versions of UNIX-like operating systems around, and many more that fit into small (16MB) flash devices. These come close.

But when I run a process listing on a Linux, Mac, or Windows system, I am astounded at the number of processes and dismayed at the number of processes I don't recall enabling or needing. The complexity of desktop systems has grown along with the apparent need to coddle the user. Oh, I guess I should write "improve the user's experience." While it is true that

automounting a CD and popping open a file browser or music player is a nice feature, it still represents complexity and provides an attacker with the ability to execute commands as the currently logged-in user. If you wonder if this ever happens, just consider the Sony BMG debacle of late 2005, when it was disclosed that millions of Windows systems had a file-hiding rootkit installed in the name of digital rights management (DRM). Ed Felten of Princeton provided a great synopsis of why many vendors do the wrong thing, while Apple has managed to do the right thing with their DRM. And Felten's student, Alex Halderman, gave us the details of the rootkit.

Please do not think that because you use Linux, BSD, or Mac OS X you are not vulnerable. During Black Hat, a conference occurring simultaneously with Security '06 (too bad), two researchers displayed a wireless hack that affects most Wi-Fi devices based on Atheros chips and demonstrated it on Mac OS X Tiger—just because of the perceived security of that OS. No OS is immune to faults, but some systems will have fewer faults than others.

Lineup

In this issue, we lead off with Steve Johnson. In the June '06 issue of *;*login**: I mentioned that Steve had found some surprising performance results when he experimented with data locality. Keeping often-referenced elements close together, in structures, has become a mantra in modern programming (see Spinellis in the April 2006 *;*login**:), but Steve clearly shows that this may not be the best strategy. Programmers (and system designers), take note!

Mike Howard sounds off next with a response to Luke Kanies's article (*;*login**:, April 2006) about Ruby. Mike considers Python nearly as object-oriented (completely so in v2) as Ruby, with many of the same features as Ruby but more maturity.

In the Sysadmin section, Mark Burgess continues his series on configuration management. He is followed by Brad Knowles, who discusses in detail the state of NTP, providing excellent advice about the appropriate use of NTP and stratum one and two timeservers.

The Technology section brings something for which I have been searching: a discussion of CPU and system technology designed to deal with the large difference between CPU and memory speed. Richard McDougall and James Laudon write about Sun's new T1 CPU architecture. In a computing world dominated by Intel and AMD, Sun has taken a very different tack, reverting to an earlier processor design, then building an eight-core, multi-threaded system. The T1 architecture provides some very significant throughput gains in applications that already have many threads, such as Apache and Oracle, while using much less power (and producing less waste heat) than their powerful competitors' chips. People whose applications match this processor's strengths owe it to themselves (and their energy budgets) to take a close look at this technology.

Timo Sivonen then explores two techniques for using encrypted file systems within FreeBSD. Timo explains how he tried two GEOM encryption facilities, GBDE and GELI, and compared their performance, including the use of different encryption algorithms.

In the Network section, Mike Freedman writes about OASIS, a system for automating server selection. Like two articles that appeared in the June '06

issue of *login*., this article is based on a paper that appeared at NSDI '06. The OASIS system provides ways that clients can be directed to the most appropriate server. Most systems for choosing the best of a set of replicated servers choose the closest server, but the OASIS algorithm also takes into account the current load reported by each server.

I wrote the next article in response to a request from Teus Hagen of NLnet. Teus thinks that the world needs a project to create a low-cost networking infrastructure. As I dug into this topic, I could see that while there are some projects dancing around the edges of this issue, what Teus has in mind goes much further. This article explores some issues in wireless technology, using RoofNet as an example, then ends with Teus's wish list for this new technology.

David Blank-Edelman has written the second half of column about tie(), outdoing himself (as usual), while Robert Haskins takes a look at the world of anti-spam solutions. Heison Chak considers how echo arises in VoIP. Robert Ferrell is back with another */dev/random* column, to be taken not seriously but certainly thoughtfully. After an excellent selection of book reviews, two articles about the standards process, and USENIX Notes, this issue ends with an array of conference reports: 2006 USENIX Annual Tech, SRUTI '06, and EVT '06.

The mention of standards reminds me of something I wanted to include in this Musings. Nick Stoughton's description of work on the ISO-C committee sent me to an article by Dennis Ritchie on the birth and early life of the C programming language. I enjoyed reading Nick's article, but I equally appreciated Dennis's viewpoint on the development of C (and reading about Steve Johnson's part in this). You can find this article at <http://cm.bell-labs.com/cm/cs/who/dmr/chist.html>.

I will confess that I see some features of the C language very differently from that of one of its creators. I heard this echoed during USENIX Security, where one panelist described C as "the best macro-assembler ever written." My own view of C is similarly colored, as I found it wonderfully close to the assembly I was using when I first learned C. But I also like to call C the programming language for people who write operating systems, and I hope that the many programmers who don't write operating systems will consider writing in strongly typed languages that have bounded arrays and don't allow manipulation of pointers. The computing world would be a much safer place if they did so.

STEPHEN C. JOHNSON

algorithms for the 21st century



PRESENTED AT THE 2006
USENIX ANNUAL TECHNICAL
CONFERENCE, BOSTON, MA

Steve Johnson spent nearly 20 years at Bell Labs, where he wrote Yacc, Lint, and the Portable C Compiler. He served on the USENIX board for 10 years, 4 of them as president.

scj@yaccman.com

THE ALGORITHMS TAUGHT TO COMPUTER science students haven't changed all that much in the past several decades, but the machines these algorithms run on have changed a great deal. Each of these algorithms is analyzed and justified based on a model of the machine running the algorithm. The analysis is often in terms of asymptotic behavior (usually described as the behavior on large problems).

This article claims that the machines we run today do not resemble, in performance, the models being used to justify traditional algorithms. In fact, today's caches and memory systems seem to reward sequential memory access, but they may actually penalize memory patterns that seem to have considerable locality of reference. This behavior is especially noticeable for the kinds of very large problems that are driving us to 64-bit architectures.

Traditional Assumptions

Traditional classes that teach analysis of algorithms and data structures use a model, often an implicit one, of how an algorithm performs on a computer. This model often has a uniform memory model, where loads all take the same time and stores all take the same time. The cost of making function calls, allocating memory, doing indexing computations, and loading and storing values is often ignored or dismissed as unimportant.

A couple of decades ago, numerical algorithms were analyzed in terms of FLOPs (floating point operations). Various schemes were used; some counted loads and stores, and some treated divide as more expensive than multiply. Over time, it became clear that the FLOP count for an algorithm had only the most tenuous connection with the running time of the algorithm, and the practice fell into disuse.

I hope to awaken a doubt in you that such traditional techniques as linked lists, structures, binary trees, and "divide and conquer" algorithms are always good for large problems on today's machines. Let's start with some simple measurements. You are encouraged to try this at home on your own computer.

But First, a Word About Time

Most modern computers have CPU cycle counters. These have the advantage that, for desktop machines, they can produce extremely accurate and repeatable timings. The times in this paper are all obtained using cycle counters.

However, there are disadvantages. There appears to be no portable way of turning cycle counts into clock time (e.g., determining the clock speed of your computer), or even getting at the cycle timer itself. In the case of laptops, the situation is quite bizarre—most laptops run faster when plugged into the wall than they do when running on batteries. Also, laptops tend to slow down when they get hot (i.e., when they are doing work!). So running tests on laptops can be misleading and the data can be quite noisy. All the data in this paper was gathered from desktop machines.

So please try this at home, but preferably not on a laptop. This article gives all the code you will need to replicate this data on an Intel-based Linux system using gcc.

I used a simple C++ class to do the basic timing. There are two methods of interest: *tic* and *toc*. Calling *tic* reads the cycle counter and saves the value; calling *toc* reads the counter again and returns the difference. The CPU timer class is:

```
class CYCLES
{
    long long var;
public:
    CY(void){};
    ~CY(void){};
    void tic(void);
    long long toc(void);
};
static long long int cycle_time;

static void tsc(void)
{
    __asm__ volatile ("rdtsc" : "=A"(cycle_time));
}

void CYCLES::tic(void)
{
    tsc();
    var = cycle_time;
}

long long CYCLES::toc(void)
{
    tsc();
    return( cycle_time - var );
}
```

Summing a Million Elements

The first program examines how the process of summing a million double-precision numbers is affected by the order in which we do the summation. We can add the numbers sequentially through memory. Or we can add every other number, then come back and get the numbers we missed on a

second pass. Or we can add every third number, and then make two additional passes to get the ones we miss. The relevant part of the program is

```

CYCLES c;           // cycle counter
#define N 1000000
double a[N];       // the array to be summed
// initialize a
for( int i=0; i<N; ++i )
    a[i] = 1.0;
double S = 0.;
long long t;       // the cycle count
// time a sum of stride s
c.tic();
for( int i=0; i<s; ++i )
    for( j=i; j<N; j += s )
        S += a[j];
t = c.toc();

```

In fact, the data to be presented are the average of 10 runs, covering strides from 1 to 1040. The cycle counts are normalized so that the stride 1 case is 1.0.

This example is not as contrived as it may appear to be, since it simulates array access patterns in large two-dimensional arrays. For example, stride 1000 simulates the reference pattern in a 1000x1000 double-precision array where the “bad” dimension varies most rapidly (the “bad” dimension in C is the first one; in FORTRAN and MATLAB it is the second one). Figure 1 shows the data for an AMD 64-bit processor, when the program is compiled unoptimized.

Notice that stride 1 is the fastest, as we might expect. But beyond that, there are some unintuitive features of this graph:

- There are periodic “spikes” where the time is 5x or more worse than unit stride.
- Even small strides are several times worse than unit stride.
- The performance gets rapidly worse for small strides, then improves for much larger ones.

Actually, the spikes, although striking, are probably the feature of these graphs that is easiest to understand. They probably arise from the way caches are designed in most modern CPUs. When an address reference is made, some bits from the middle of that address are used to select a portion of the cache to search for a match, to save time and power. Unfortunately, this means that when the stride is close to a high power of 2, only a small portion of the available cache space is being used. It is as if the effective cache size is a tiny fraction of that available in the unit stride case. This effect happens, with somewhat different numerology, for each of the caches (with modern systems having two or three).

What is surprising, especially in the later data, is the magnitude of this effect.

The graph in Figure 1 involved unoptimized code. If we optimize (gcc -O4), we get the graph shown in Figure 2.

Optimization does not change the essential shape or properties of the curve, although the spikes are a bit higher. This effect is largely the result of the code for unit stride being a bit faster (recall that the graphs are normalized so that unit stride is 1.0).

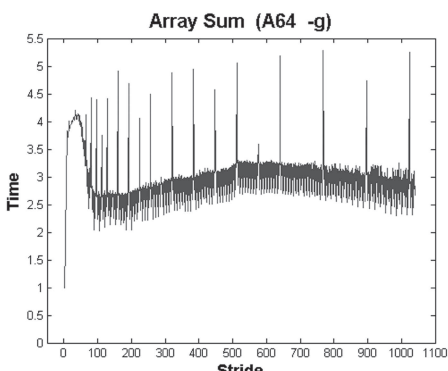


FIGURE 1

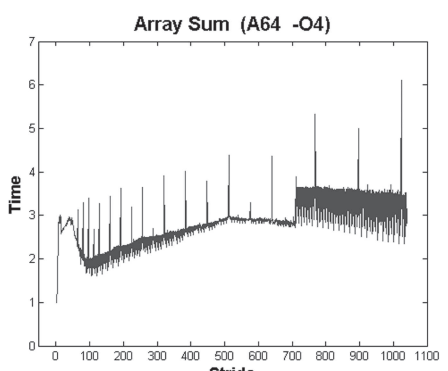


FIGURE 2

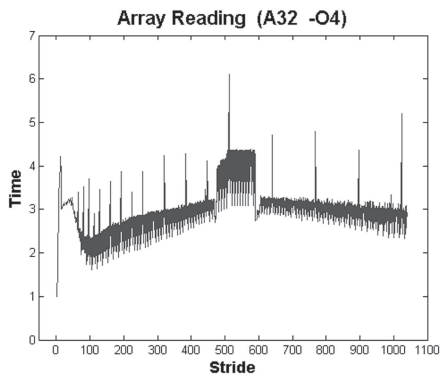


FIGURE 3

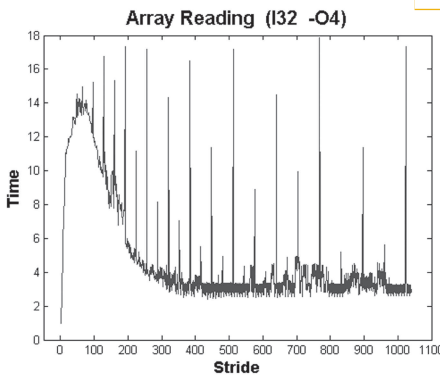


FIGURE 4

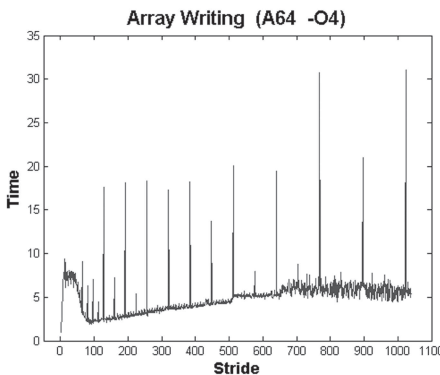


FIGURE 5

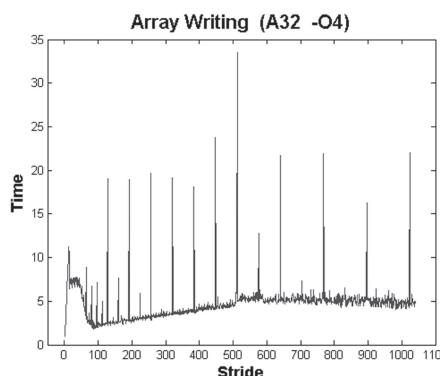


FIGURE 6

We can also collect data on a 32-bit AMD processor (see Figure 3).

Notice that the shape of the curve is similar, but the spikes are closer together. There is also a strange “hump” around 512, which appeared on multiple runs (which doesn’t preclude it from being an artifact!). The unoptimized version of this test on the 32-bit AMD system also had a hump that was lower and broader. The 64-bit AMD data may show signs of a smaller hump centered around 1024.

Figure 4 displays the curve for an Intel 32-bit system.

Note that the behavior for small strides is much worse than that of the AMD machines, but there is no sign of the hump. The spikes are closer together, probably because the caches are smaller.

Writing Data

We can run a similar test on writing data. In fact, we do not need to initialize the array, so the code is simpler:

```

CYCLES c;      // cycle counter
#define N 1000000
double a[N];   // the array to be written
long long t;   // the cycle count
// time writing N elements with stride s
c.tic();
for( int i=0; i<s; ++i )
    for( j=i; j<N; j += s )
        a[j] = 1.0;
t = c.toc();

```

The results for a 64-bit AMD machine are shown in Figure 5.

At first glance, the data appears smoother (except for the spikes), but this is an illusion, because the scale is much larger. In this case, the worst peaks are up to 30x the unit stride times. Once again, the peaks appear at strides that are powers of 2.

The 32-bit AMD data is shown in Figure 6.

Again the peaks appear at powers of 2, and again they are up to 30x worse than unit stride. The Intel 32-bit graphs for reading and writing are quite similar.

Writing Data Repeatedly

The programs for summing and writing data are worst-case examples for cache behavior, because we touch each data element exactly once. We can also examine what happens when we write data repeatedly. By modifying our test case slightly, we can write only 1000 elements out of the million-element array but write each element 1000 times. Once again, we vary the strides of the 1000 elements. Note that for all strides, only 8000 bytes are written. The program looks like:

```

CYCLES c;      // cycle counter
#define N 1000000
double a[N];   // the array to be written
long long t;   // the cycle count
// time writing N elements with stride s
// note: N must be bigger than 999*s+1

```

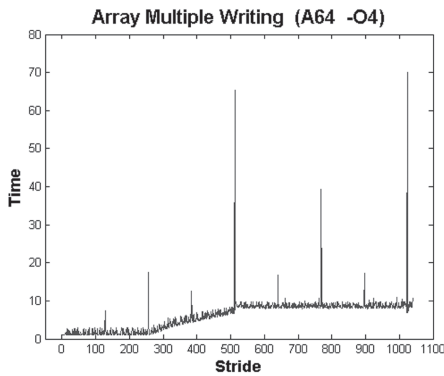


FIGURE 7

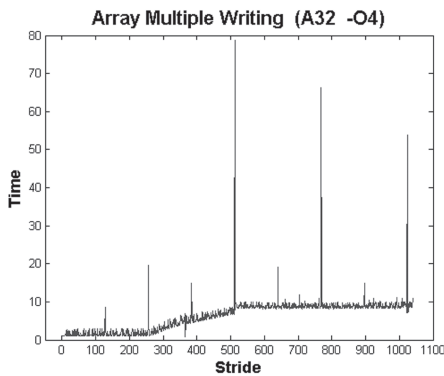


FIGURE 8

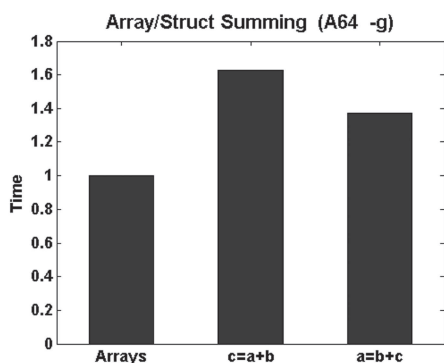


FIGURE 9

```
c.tic();
for( int i=0; i<1000; ++i )
    for( j=k=0; k<1000; j += s, ++k )
        a[j] = 1.0;

t = c.toc();
```

We can be forgiven for having hoped that this amount of data could fit comfortably into the caches of all modern machines, but Figure 7 shows the 64-bit AMD results, and Figure 8 shows the 32-bit AMD results.

Unfortunately, the peaks are still present. Large strides are still worse than small strides by nearly an order of magnitude. And the size of the peaks is astonishing, up to 70x.

Data Layout Issues

This data suggests that modern memory systems don't actually do much to improve local references to data unless those references are in fact sequential. Even rather small strides show significant degradation over the unit stride case. This rather contradicts the trend in language design to support structures that place related data together. We can measure the magnitude of this effect of structures with a similar test. Suppose we wish to do a million additions of related elements. We can create three million-element arrays, and add the corresponding elements. Or we can create a structure with three elements in it, make a million-element structure array, and loop through it by doing the additions for each structure in the array. The inner loop of the programs looks like:

```
CYCLES c;
#define N 1000000
double a[N], b[N], c[N];
long long t;
for( int i=0; i<N; ++i )
    a[i] = b[i] = c[i] = 1.0; // initialize
c.tic();
for( int i=0; i<N; ++i )
    a[i] = b[i] + c[i];
t = c.toc();
```

for the case of three arrays, and

```
CYCLES c;
#define N 1000000
struct three { double a, b, c; } A[N], *p;
long long t;
int i;
for( i=0, p=A; i<N; ++i, ++p )
    p->a = p->b = p->c = 1.0; // initialize
c.tic();
for( i=0, p=A; i<N; ++i, ++p )
    p->a = p->b + p->c;
t = c.toc();
```

for the structure case. Just to see whether the order matters, we can also measure

```
p->c = p->a + p->b;
```

Figure 9 displays the results for the AMD 64-bit machine, with the programs compiled unoptimized.

Note that using unit stride with separate arrays is significantly faster than for the structure cases, by tens of percents. Note also that there is a significant difference between the two structure cases, depending on the data ordering in the structure. If we optimize, we get the results shown in Figure 10.

Once again, using separate arrays is significantly faster than using structures. The order of the data in the structure is much less important when the program is optimized.

Discussion

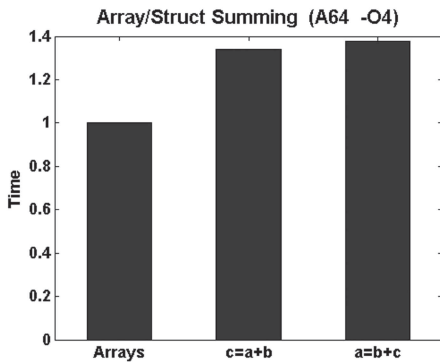


FIGURE 10

I have collected too much wrong performance data in my career not to warn that these data may contain artifacts and noise caused by operating system tasks and other background computing. More seriously, with just a few tests we are far from understanding the effect of CPU speed, cache size and architecture, and memory system architecture on the performance of even these simple programs. There is enough data, however, to strongly suggest that modern computer cache/memory systems do *not* reward locality of reference, but rather they reward sequential access to data. The data also suggests that access patterns that jump by powers of 2 can pay a surprisingly large penalty. Those doing two-dimensional fast Fourier transforms (FFTs), for example, where powers of 2 have long been touted as more efficient than other sizes, may wish to take notice.

I am not trying to suggest that computers have not been designed well for the typical tasks they perform (e.g., running Apache, Firefox, and Microsoft Office). However, with 64-bit computers and terabyte datasets becoming common, computation on datasets that greatly exceed the cache size is becoming a frequent experience. It is unclear how such data should be organized for efficient computation, even on single-processor machines. With multi-core upon us, designing for large datasets gets even more murky.

It is tempting to think that there is *some* way to organize data to be efficient on these machines. But this would imply that the system designers were aware of these issues when the machines were designed. Unfortunately, that may well not have been the case. History shows that computing systems are often designed by engineers more motivated by cost, chip and board area, cooling, and other considerations than programmability. Future data structure design, especially for large datasets, may well end up depending on the cache and memory sizes, the number of cores, and the compiler technology available on the target system. “Trial and error” may have to prevail when designing data structures and algorithms for large-data applications. The old rules no longer apply.

We can speculate that “large dataset computing” could become a niche market, similar to the markets for servers and low-power systems. Perhaps we can work with hardware manufacturers to develop techniques for algorithm and data-structure design that software designers can follow and hardware manufacturers can efficiently support. Meanwhile, try this at home, and welcome to a brave new world.

REFERENCES

There is an interesting book by David Loshin, *Efficient Memory Programming*, that has a lot of material on how caches and memory systems work

(even though the book dates from 1998). Unfortunately, there's little empirical data, and he repeats the old saws about locality of reference.

There is also a field of algorithm design called *cache-aware algorithms*. The idea is to develop a family of algorithms to solve a particular problem, and then choose one that best fits the machine you are running on. Although this is an effective technique, it begs the question of how we design data structures to optimize performance for today's machines. Google "cache aware algorithm" to learn more than you want to know about this field.

It's worth pointing out that similar issues arose once before in the vector machine era (1975 to 1990 or so). Vector machines so preferred unit stride that many powerful compiler techniques were developed to favor unit stride. It is also notable that most vector machines did not have caches, since reading and writing long vectors can "blow out" a conventional cache while getting little benefit thereby.

Here is the detailed information about the machines I used to collect this data:

- The AMD 64-bit data was collected on a dual-processor 2.2 GHz Athlon 248 system with 1 MB of cache and 2 GB of main memory. The gcc version was 3.4.5.
- The AMD 32-bit data was collected on a three-processor AMD Opteron 250 system running at 1 GHz with 1 MB caches. The gcc version was 3.2.3.
- The Intel 32-bit data was collected on a four-processor Xeon system—each system ran at 3.2 GHz and had a 512K cache. The gcc version was 3.2.3.

MIKE HOWARD

maybe you should use Python



Mike Howard came into programming from Systems Engineering and has been stuck there. He currently makes his living doing custom software and system administration for a few small companies.

mike@clove.com

LUKE KANIES' ARTICLE "WHY YOU

should use Ruby" in the April *;login*: [1] makes some really good points in Ruby's favor. While reading the article, I noticed I could make all the same points for Python.

Before getting started, I need to make two things clear.

First, this is not a criticism of Ruby. I'm not sure which is better, if either is. The important thing to me is that the features Luke talked about make programming easier to do and maintain. They should be part of *any* modern language.

I also should explain my feelings about Ruby and Python: I kind of like Ruby, but don't plan to do much coding in it. I've been writing code in Python for about five or six years now—beginning with Python 1.5.2. I like Python's terse, clear style. One thing that attracted me to it to begin with is the thing that bothers Luke—indentation is mandatory and syntactically significant. However, I'm not a Python guru—I've only written around 50,000 lines or so.

I was attracted to Ruby because of Rails. I haven't written much more than a few thousand lines, but from what I have seen, it's a nice language with a few more rough edges than Python.

The primary difference I see between the languages are:

- Their age: Python is a few years older and so has had more time to be cleaned up.
- The approach of the designers: Python is aimed at succinct code that tends to have only one method to do any given task; Ruby is more of a kitchen-sink language, and so programmers have many equivalent options.

I think the two languages are converging toward much the same feature set, but with stylistic differences. Python is gradually and carefully adding things, whereas Ruby is slowly discarding things that are redundant.

Now let's get to the points Luke Kanies brought up.

Point 1: In Ruby, Everything Is an Object

Python 1.x had two kinds of things: primitives and objects. Python 2 introduced the "new style class" and has relentlessly driven the language to the point where "everything is an object."

Within Python 2.x, old style classes still exist, even though for later releases primitives such as

integers and strings are now objects. We are told that the journey will be complete in Python 3.x.

Luke provides this example of how easy it is to get information about an object in Ruby:

```
[Class, "a string", 15, File.open("/etc/passwd").each { |obj|
  puts "'%s' is of type %s" % [obj, obj.class]
}]
```

Here is essentially the same code in Python:

```
for x in [object, "string", 15, file('/etc/passwd')]:
    print "%s is a %s" % (repr(x), x.__class__)
```

which yields

```
<type 'object'> is a <type 'type'>
'string' is a <type 'str'>
15 is a <type 'int'>
<open file '/etc/passwd', mode 'r' at 0xb755f4a0> is a <type 'file'>
```

Point 2: According to Luke, in Ruby There Are No Operators: All Operations Are Defined by Functions Associated with the Objects Involved

This was not true of Python 1.x, but it is pretty much true in Python 2.4 and above. As everything becomes “an object,” it will be uniformly true, in the sense it is in Ruby. That is, all operators in the language are implemented using special methods attached to the operands and if the method doesn't exist, the interpreter throws an exception.

For example, in Python, $x + y$ is executed as `x.__add__(y)` or `y.__radd__(x)`.

Point 3: Introspection

Introspection allows you to find information about objects as they are running. This contrasts sharply with languages in which programs must be (almost) completely specified at compile time. Both Ruby and Python have extensive support for inspecting the visible state of everything.

I'll only mention two Python features here: the builtin function `dir()` and the `__doc__` attribute.

`dir(foo)` returns a list of all attributes and methods attached to its argument. That's all there is to it:

```
dir(1.0)
['__abs__', '__add__', '__class__', '__coerce__', '__delattr__', '__div__', '__divmod__', '__doc__', '__eq__', '__float__', '__floordiv__', '__ge__', '__getattr__', '__getnewargs__', '__gt__', '__hash__', '__init__', '__int__', '__le__', '__long__', '__lt__', '__mod__', '__mul__', '__ne__', '__neg__', '__new__', '__nonzero__', '__pos__', '__pow__', '__radd__', '__rdiv__', '__rdivmod__', '__reduce__', '__reduce_ex__', '__repr__', '__rfloordiv__', '__rmod__', '__rmul__', '__rpow__', '__rsub__', '__rtruediv__', '__setattr__', '__str__', '__sub__', '__truediv__']
```

The `__doc__` attribute contains printable documentation about the object:

```
print 1.0.__doc__
float(x) -> floating point number
```

Convert a string or number to a floating point number, if possible.

Point 4: In Ruby, Many Objects Know How to Iterate Over Themselves

Rather than writing a conventional, imperative programming style loop, you can say something like this:

```
object.each { |x| do something with x }
```

This is a good thing and such facility has been added to the 2.x versions of Python, with capabilities increasing with each point release. This is a very light survey of what Python provides.

Python uses the existing for loop syntax similarly to the way that Ruby uses the each method. The Python for loop looks roughly like this:

```
for <list of variables> in <arbitrary iterator>:  
    stuff to do
```

which is equivalent to calling a block of code with parameters set to the variables in the list. So the Python for is equivalent to Luke's Ruby code.

Initially, the <arbitrary iterator> was any sequence—a list, tuple, or string. This has been extended in Python 2.x to anything that satisfies the “iterator protocol” (see below).

In addition, several interesting additions to Python make it easier to use this construction in more compact, yet clear ways.

LIST COMPREHENSIONS

List Comprehensions, added in 2.1, are lists defined by one or more sequences. List Comprehensions generalized the ideas of `map()`, `zip()`, and friends. For example,

```
[x*x/2.0 for x in range(1,1000) if x%3 == 0]
```

GENERATORS

Generators were added in 2.2. A generator looks like a function except that it contains the keyword `yield` instead of `return`. A generator returns an object that has a `next()` method. Calling the `next` method either returns a value or raises the `StopIteration` exception. (`StopIteration` is what now stops the for loop in Python.)

```
def myrange(bot, top):  
    while bot < top:  
        yield bot  
        bot += 1  
  
for x in myrange(1,20):  
    whatever
```

GENERATOR EXPRESSIONS

Generator Expressions, added in 2.4, are essentially lazy list comprehensions. That is, a list comprehension realizes the entire list, but a generator expression just returns the next element. This allows simple iteration over infinite sequences. To write one, replace the square brackets with parentheses:

```
(x*x/2.0 for x in range(1,1000) if x%3 == 0)  
(line for x in file('/etc/passwd') if x[0] != '#')
```

ITERATOR PROTOCOL

Iterator Protocol, added in 2.2, is a general method for objects to become iterators. This provides functionality similar to Ruby's `each` method. In brief, if an object defines two special methods, it can replace the list component of a `for` loop. These methods are:

- `__iter__` (self), which returns a function that acts as an iterator
- `next`(self), which returns the next item from the object or raises the `StopIteration` exception

The imminent release of Python 2.5 will continue this trend and promises support for co-routines by expanding the capabilities of generators and unifying Python's exception-handling code.

Iterators are really cool. They make code both compact *and* easy to understand.

Point 5: Ruby Has Code Blocks

Ruby code blocks are similar to anonymous functions that can be passed to methods for execution. I say they are similar because the parameters of a Ruby code block are existing local variables; those local variables are then used within the block and their values are changed as a result of the block's execution [2]. Ruby code blocks can be arguments of methods, can be assigned to variables, etc.

Python functions can be manipulated similarly. They can be passed to and returned from functions. They can be assigned to variables. And they can be executed in loops that are controlled by iterators. Python function parameters are always local variables, so they are less prone to unintended side effects.

In Python, `def` (function definition) is an executable statement that returns a function. This makes it easy to write closures:


```
>>> def foo(x):
...     def tmp(y):
...         return y < x
...     return tmp
...
>>> a = foo(10)
>>> a
<function tmp at 0xb7562b1c>
>>> a(4)
True
>>> a(12)
False
```

Summary

I agree strongly with Luke that the features he outlined in [1] are excellent features that should be in all modern programming languages. I also agree that they are good reasons to use Ruby or Python.

REFERENCES

- [1] Luke Kanies, “Why You Should Use Ruby,” *;login*: (April 2006).
- [2] Dave Thomas, *Programming Ruby: The Pragmatic Programmers Guide*, 2nd ed. (Pragmatic Programmers, 2005), p. 51. This scope issue is being debated within the Ruby community and will probably change in some subsequent release.
- [3] See www.python.org for documentation for all releases, as well as the code. Many features (implemented and proposed) are described in the PEPs.
- [4] David M. Beazley, *Python, Essential Reference*, 3rd ed. (Sams Publishing, 2006).



Sponsored by USENIX,
The Advanced Computing Systems Association,
in cooperation with ACM SIGOPS

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

Join us in Seattle, WA, November 6–8, 2006, for the 7th USENIX Symposium on Operating Systems Design and Implementation. OSDI brings together professionals from academic and industrial backgrounds in what has become a premier forum for discussing the design, implementation, and implications of systems software. Sessions include:

- Runtime Reliability Mechanisms
- OS Implementation Strategies
- Distributed Storage and Locking
- Program Analysis Techniques
- And more

See the full program at www.usenix.org/osdi06. Don't miss an outstanding program covering the best systems software research and practice. Register online by October 16, 2006, and save!

OSDI '06 is co-located with the 3rd USENIX Workshop on Real, Large Distributed Systems (WORLDS '06), which will take place on November 5. The Second Workshop on Hot Topics in System Dependability (HotDep '06) will be held on November 8, immediately following OSDI '06. See www.usenix.org/events for more information and to register online.

Nov. 6–8, 2006,
Seattle, WA

www.usenix.org/osdi06

MARK BURGESS

configuration management: models and myths



PART 2: BABEL, BABBLE, TOIL, AND GRAMMAR

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

Mark.Burgess@iu.hio.no

TIME TO PUT THE ADMINISTRATIVE house in order? Then you are going to need a way of describing that house. Configuration management, as discovered in part 1 of this series, is the management of resource patterns. If you can't communicate a configuration pattern, you certainly can't have someone create it, verify it, or maintain it. So, although there are clearly many ways to build your house of cards, you will need to learn the language of patterns if you want to make a bunch of them exactly alike.

Parentheses (and More Parentheses)

Call me not a linguist or a poet by training; my roots were nurtured in that country on Earth with surely the worst reputation for knowing foreign languages (worse even than the United States). Still, I am uncharacteristically both intrigued and betaken by language.

(What? "Betaken"? Not a word, you say?
Hmmm . . . stay tuned!)

These days I live in Oslo, in the southern part of Norway, but I started my life in the northwest of England. Ironically, this is the part of England whose culture and language were "impregnated and borrowed" by Vikings from Norway in early A.D. The inheritance of that invasion is still there, to the observant eye and ear.

I lived not far from a charming creek called Beckinsdale (in Modern Norwegian, *Bekk i dal*, meaning "stream in valley"). People still call their children "bairns" in that part of the world (in Modern Norwegian, "barn" means "child"). There are many examples of such glossal cross-pollination. In fact, the languages of Old English and Old Norse were so alike that the Vikings and their victims probably understood each other quite easily, and today language scholars are often at pains to determine from which of them certain words came.

In dialect, I recall verb endings that sounded perfectly natural to me: "we've meeten, eaten, beaten, moven, proven." Surely, these are older forms of verb endings than in the modern English "we've met, beaten, moved, proved." (The endings sound somehow more Germanic, though I am just guessing; I was reminded of them on playing Deep Purple's song "Space Truckin'," where Ian Gillan sings, "We've meeten all the groovy people . . .")

It is odd that “eaten” alone has survived in the U.K. (as has, occasionally, “proven”; “gotten,” however, which has survived in the U.S., is strictly forbidden in the U.K. and yet the derivatives “forgotten” and “begotten” are standard.). Clearly, the rumors of English grammar have been greatly exaggerated.

What of “betaken”? Why is this not a word? It clearly fits the grammatical forms. One even says idiomatically “I am rather taken by that” (preferably with a butler-like inflection) and, of course, there is a similar word “betrothed,” which is in the dictionary. In Modern Norwegian it is indeed a word (“betatt”) and it means exactly “rather taken by,” so I hereby define the word “betaken.” And who can stop me?

Indeed, language changes regularly and we are inventing new words all the time, using recognizable patterns. It tends to move from complicated constructions toward simple regular patterns. If you examine which verbs are still irregular (or strong) in language, it is those verbs that are most commonly used (e.g., “to be”). There is a simple reason for this: We only remember the irregularities if we are using them all the time and they are strong enough to resist change. In other cases we forget the “correct” forms and (regularize|regularise) them according to some simple syntactic pattern. Anyone who has seen the British TV show “Ali G” will know from his parodical dialect that there are parts of the U.K. where even the verb “to be” is losing to regularization: “I is, you is, he is . . . , innit.” (Prizes will be awarded for guessing the last word’s origins.)

In fact we add and change word endings willy-nilly: In the U.S. my least favorite word at the moment is “provisioning” (which I like to call “provisionizationing”) although “de-plane” is way up there (and it surely means picking passenger aircraft out of the fur of a cat). These are particularly nasty examples of “verbing” and “nouncing,” especially American phenomenonizationings. In the U.K., people have an odd habit of saying “orientated” instead of “oriented,” fearing possibly that the latter has something to do with a cultural revolution of cheap shoes, or harks of a country they never managed to “civilise.” Or, perhaps they are simply so orientitillated that they feel they must.

At any rate, although there are definite patterns to be seen, clearly human language is driven by populism and natural selection, not by total logic or design.

The Chomsky Hierarchy

So much for human language. It seems to have very little to do with structure or reliability—qualities we are certainly looking for in system administration. So let’s get formal.

In the passages in the previous section, I broke several rules of writing [although ;login:’s copyeditor may have unwittingly “corrected” some of the more egregious abuses—*copy ed.*] and made you (the reader) work harder than is generally allowed in modern literature. I served a plethora of parenthetical remarks and digressions. I am guessing that you have noticed these (and that you had no trouble in parsing them) but that they were a little annoying, since you had to work slightly harder to understand what I have written. Of course, I was making a point.

The theory of discrete patterns, such as houses of cards or flowerbeds, is the theory of languages, as initiated by researchers including Noam Chomsky in the late 1950s and 1960s. For discrete patterns, with symbolic

content, it makes intuitive sense that discrete words and their patterns might be a good method of description; but when we get to continuous patterns, such as the curving of a landscape, what words describe the exact shapes and infinite variations of form? For that we need a different language: continuous (differential) mathematics, which we shall not have time to mention in this episode.

The theory of formal languages assumes that a discrete pattern is formed from an alphabet of symbols, shapes, colors, etc., much like a pack of cards; patterns are then classified by measuring the complexity of the simplest mechanism or computer program that could generate the pattern. The classes of patterns are called formal grammars. Their classifications and corresponding state-machines are as follows:

- Regular languages (finite automata, or finite state machines)
- Context-free languages (push-down automata)
- Context-sensitive languages (nondeterministic linear bounded automata)
- Recursively enumerable languages (Turing machine)

The syntax of a language is a list of all legal sentences in the language. Lists are not very helpful to us, though: We have trouble remembering things by brute force, so we try to identify the repeated patterns and turn them into rules. These pattern-rule templates are called grammars. The simplest grammars are the regular grammars, and the patterns they represent can be modeled by a simple pattern-matching language: regular expressions.

Regular Expressions

All UNIX users have met (or meeten) regular expressions. They are a well-known and powerful way of matching text strings. The implementations we know are stylized enhancements of the regular expressions of language theory.

A language is said to be regular if it can be constructed from some alphabet of symbols and satisfies a few basic rules. Let us suppose that we have an alphabet, A , which contains a finite number of symbols. Those symbols could be alphabetic, alphanumeric, numeric, glyphs, flowers (as in part 1), or any arbitrary collection of denumerable symbols. The rules are these:

- The empty string and each symbol in the alphabet are regular expressions.
- If E_1 and E_2 are regular expressions, then so is E_1E_2 , i.e., the concatenation of the two (e.g., expressions “b,” “e,” “be,” “taken,” and “betaken”).
- If E_1 and E_2 are regular expressions, then so is the union of the two (i.e., we allow alternate expressions to be combined in no particular order). This is written with the vertical bar “|” in most implementations (e.g., we have (met|meet)).
- If E is a regular expression then so is E^* (repeated instances). Hence we have acceptable expressions “provision,” “ization,” and “ing” generating “provisionizationingizationingingization,” etc. ad lib.
- Nothing else is a regular expression.

The Kleene star ($*$) is a shorthand for the concatenation zero or more instances of members of a set or expression. This is the parsimonious form of regular expressions. We'll not delve into implementations for now.

Languages in Configurations

There has been a lot of talk about “configuration languages” as tools for sorting out UNIX systems: cfengine, LCFG, now Puppet, etc. Rumor has it, I wrote one of these myself. But don't let this talk of language trick you back into thinking about these tools. Rather, notice that the very problem of configuration itself involves language—because it is about describable patterns. For example, UNIX file permissions form the simplest kind of regular language. If we take the octal representation, they consist of scalar states of constant length and a fixed alphabet consisting of the following “symbols”:

$$Q = \{0,1,2,3,4,5,6,7\}$$

It is easy to represent this as a language. It is simply the union of each of the symbols. That is, if we ignore the foibles of UNIX syntax, then the entire language is simply written

```
000|001|002|003|004|...|776|777
```

This is all very well, but so what?

The significance of regular expressions for configuration policy is that there is a provable equivalence between regular languages and finite state machines, i.e., the simplest kind of algorithms, using a fixed amount of memory. This means that regular strings are relatively easy to parse, identify, and understand. This, at least partly, accounts for their ubiquity in computer software where pattern matching is required.

Regular expressions occur in editors, where searching and replacing is required, in intrusion-detection and spam-detection software, in all kinds of policy languages, and on the UNIX command shell (as “globbing”). They are a central part of Perl, a language designed for pattern extraction (though Perl is not a regular language). Today, no computer toolbox is complete without a regular expression library.

Bring on the Toil (Parentheses Again)

In spite of the multifarious uses for regular expressions, they are only the lowest level of sophistication in the Chomsky hierarchy. The computer languages we are most familiar with for programming or markup are almost all context-free languages. Such languages can only be approximated with finite memory. They contain nested parenthetical structures that require an extensible stack to process. Here, for instance, are some examples of languages that use parentheses to identify information by type:

1. `<account>`
 `<uname>User1</uname>`
 `<passwd>x7hsk.djt</passwd>`
 `<uid> 100 </uid> ... </account>`
2. `(account (uname User1) (passwd x7hsk.djt) ...)`

If the level of parenthetical nesting in a grammar is not large, we can simulate common cases of context-free languages by treating fragments as regular expressions with balanced pairs of symbols (as anyone who has written a simple parser will know). This is useful because it means that a simple finite state machine can make a good attempt at interpreting the string and this is cheap.

However, to ensure full generality one must go beyond regular language tools and enter the realm of stack-based tools such as Yacc and Bison for

context-free grammars. Each level of the Chomsky hierarchy grows in its computational complexity (costing us more to parse parenthetical remarks (as you (no doubt) experienced in my introduction)). The most general patterns require a full Turing machine (a computer with infinite memory) to solve.

The trouble with this next level of computation is that it is a drastic step. It requires a whole new level of sophistication and toil in modeling, describing, and understanding to master. We want to use higher-grammatical patterns to design, classify, and maintain structures that are context free. Worse yet, the structures might be inside files, in packet streams, distributed around a network, or inside a database. The difficulty of going beyond finite state automata partly explains why pattern-recognition systems (such as network intrusion detection systems), which obviously need to deal with parentheses (e.g., TCP-SYN, TCP_FIN), generally do not record such state, but rather rely on regular expression rules applied as fragments. This is “doable,” if not optimal.

Data Types and Bases

In configuration management we meet information in a variety of forms. Lists of values are common. Line-based configuration files are ubiquitous in UNIX. Windows has a simple key database in its registry. What kinds of languages do these data form?

- Scalar permissions are regular languages.
- Lists of regular objects are also regular.
- A line-based text file is a list and hence is regular.
- Text files containing higher grammars such as XML are context free.

Relational databases have been used to store data almost since computing began. They add a new twist to the idea of languages, namely that the words one forms from the basic alphabet of a language (and sometimes even the symbols of the alphabet) can be classified into types. Consider Figure 1.

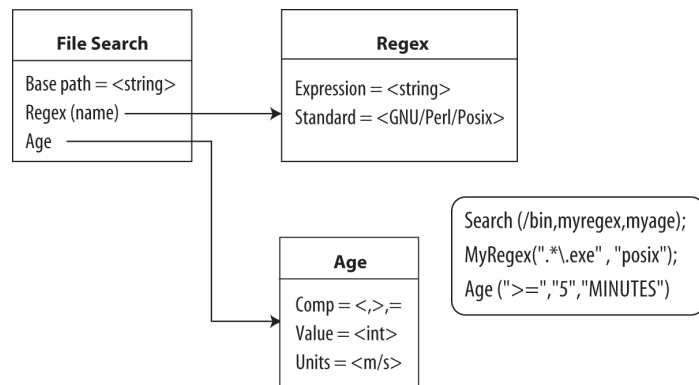


FIGURE 1: SOME TABLES IN A RELATIONAL DATABASE

The figure shows the basic idea of a relational database. Certain types of data are grouped together in tables or records. Such data structures have eventually ended up in programming languages too, in the form of records, structs, and now even object-oriented “classes.” The main point of putting information into a predictable structure is that one imposes a linguistic discipline on the data. The tables are simple parentheses around a number of regular language items that are given names. In the first table we have a string (which is a regular object) with a name “base path,” a “regex,”

which is a new kind of table or parenthetic grouping, and an age, which is yet another parenthetic grouping. The “regex” has two regular members: a regular expression (which is a string and is hence also a regular object) and a label (string or number), which is regular. Similarly, “Age” consists of a list of three regular objects.

A relational database is therefore a context-free language. SQL is a query language that uses regular expressions embedded in a table model to locate data in the database (which has its own context-free language pattern). We cannot escape from languages or these basic pattern ideas in configuration management. They recur at all levels.

Data types are a powerful idea. They allow us to distinguish among seemingly equivalent patterns of data and therefore open up a range of flavors or colors to the flowers in our garden. This is the purpose of having tables in relational databases: We can group together objects into comparable clusters. Syntactically, all objects of the same type have the same basic structure and are therefore comparable, i.e., they form the same subpattern.

Markup

The trouble with databases is that they are not very transparent—they can only be read with special tools, so it is hard to see the structures in data in an intuitive way. This is less of a problem in computer programming languages where class hierarchies are written down in ASCII form. For many, the answer to this problem has been to adopt XML, a generic markup representation for a context-free data structure, which adopts the best of both worlds. Not only does XML offer a standardized encoding of a context-free structure, it claims to make it parsable by humans as well as machines. (Let us say that the rumors of its human-readability have been greatly exaggerated.)

Every pair of tags in a markup language such as HTML or XML makes a data type out of the parenthesized region. For example:

```
The <adj>quick</adj> brown <noun>fox</noun> <verb>jumps</verb>
over the lazy dog.
```

The current adoration of XML has no real significance as far as problem-solving goes, but it is interesting that the trend in system design is to move away from regular line-based data, as is traditional in UNIX and DOS, toward context-free data. This opens the door to much greater complexity, with attendant consequences that we shall consider as the series progresses.

Revolution or Regex?

Toil, work, and difficulty relate to grammars or patterns rather than to symbols. Noah Webster, as a slap in the face to the British, rewrote the spelling of the American English as a political act after the revolution. (No doubt my own spellings “colour,” “flavour,” etc., have been magically transformed into American “color” and “flavor” by the copy editor. [Indeed—*copy ed.*]) The adaptation has almost no consequence (except to annoy self-righteous Brits immensely); many readers hardly even notice this change. Had Webster altered the grammar of the language, there would have been serious trouble. But the fact is that, although he obscured some of its etymology, the basic patterns of the language did not change, and therefore even the most obtuse of colonialists can still read American (although Canadians seem totally confused about how they are supposed to spell).

The patterns that we are able to discuss and represent are key to mastering the problem of configuration management. Many system administration and management tools try to force users into doing either what the tools can do or what is considered manageable. By asking users to limit the complexity of their configurations they plump for a reasonable strategy that strives for predictability. This might be all right in practice, for the time being, but if we are going to fully understand the problem, we must go beyond quick fixes. The challenge for any theory of configuration lies in describing what people really do, not in trying to force people to do something that is easy to understand.

In the next part of this series, I would like to run through some of the data models that have been applied to the problem of system management. We shall ask the following: How can we measure their complexity, and why are none of them ever really used?

Save the Date!

www.usenix.org/fast07



**5th USENIX Conference on File
and Storage Technologies**
February 13–16, 2007 San Jose, CA

Join us in San Jose, CA, February 13–16, 2007, for the latest in file and storage technologies. The 5th USENIX Conference on File and Storage Technologies (FAST '07) brings together storage system researchers and practitioners to explore new directions in the design, implementation, evaluation, and deployment of storage systems. Meet with premier storage system researchers and practitioners for 2.5 days of ground-breaking file and storage information!

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

USENIX

BRAD KNOWLES

it's about time . . .



Brad has been using UNIX and the Internet for over 22 years, doing UNIX and Internet administration for over 16, and specializing in Internet email and DNS administration for more than a decade, and he now considers NTP and the NTP PSP to be his third principal area of specialty. He has spoken at a number of major conferences; was on the program committee for SANE 2000 and SANE 2002; was a reviewer for the second editions of *Sendmail* (O'Reilly, 1997), *DNS and BIND* (O'Reilly, 1997), and *Sendmail Performance Tuning* (Pearson Education, 2002); is currently involved in writing his own book; is co-authoring a booklet in the SAGE Short Topics series; and has been asked to be a reviewer of at least one other technical book in the field.

brad@stop.mail-abuse.org

AFTER 25 YEARS OF DEVELOPMENT

[41], the Network Time Protocol (NTP) is now firmly established as the standard cross-platform way to set and maintain computer clocks on the Internet. Most modern OSes ship out-of-the-box with clients for NTP, and many of those are turned on by default. Most network devices have NTP clients built-in, and even many Small Office/Home Office (SoHo) DSL/cable modems/routers have them turned on by default. Unfortunately, as adoption spreads, misconfiguration is becoming more common, especially vendor misconfiguration. With misconfiguration comes bad or no clock synchronization and abuse or even “vandalism” of a surprisingly small number of time servers on the Internet.

The purpose of this article is to give you an update on the status of the protocol itself, the NTP Public Services Project (where you can get support for questions you may have regarding NTP), books and documentation related to NTP, the Top Five Most Common Problems, lists of publicly accessible time servers (including the NTP Server Pool project), time synchronization “state of practice” on the Internet, the release of updated “Reference Implementation” code, and recent developments on NTP server abuse (following David Malone’s article from the April 2006 issue of *;login:*). Footnotes used will be in the *asr* (alt.sysadmin.recovery) tradition [32, 33].

NTP Working Group

The IETF is in the process of updating the NTP-related RFCs, specifically working toward an official specification for version 4 of the NTP protocol (RFC 1035 was published in 1992 and covered NTPv3).

Toward this end, they have set up an NTP Working Group (NTPWG) [14]. The mailing lists are being hosted [15] by the NTP Public Services Project [26], as well as a TWiki[40]. The NTPWG page [14] tells us:

A number of topics have been raised as potential work items for an update to NTP including support for IPv6, security considerations including authentication, automatic

configuration including possible requirements for DHCP, and algorithm improvements.

If you're interested in helping to shape the future of the NTP protocol or the NTP implementations, please join the group and give us the benefit of your experience and views.

NTP Public Services Project

For years, the main site for most things related to NTP was at www.ntp.org. The ntp.org domain is owned by Dr. David Mills [34, 7], the Web site is maintained by his students at the University of Delaware and various members of the “volunteer corps,” and the hardware is managed by the UDel staff. However, the support services eventually outgrew the hardware resources available at the host institution and, unfortunately, began to conflict with their policies.

Thanks to support from the Internet Systems Consortium, most of the NTP support services have been migrated to the NTP Public Services Project (NTP PSP) [26], as part of the “Hosted@ISC” programme, which includes Apache, FreeBSD, KDE, Mozilla, OpenBSD, OpenDarwin, OpenLDAP, OpenOffice, PostgreSQL, XFree86, kernel.org, and many others. Remaining at the ntp.org Web site are the main NTP home page, the NTP FAQ, the official download site for the source code, and tarballs of the “Reference Implementation,” as well as continued research and development of the protocol and code by Dr. Mills and the “volunteer corps.”

We are very grateful for all the assistance and hardware provided by ISC, and we'd like to thank all of our other donors as well [30]. A page for donating to the project has been set up [29], which includes information on how you can make tax-deductible donations through ISC and links to information on various pieces of hardware that we lack and hope to be able to obtain [31].

NTP Documentation

Dr. Mills is the person who originally created the NTP protocol and is sometimes called “Father Time” [7, 34]. He has recently published his book on the subject, *Computer Network Time Synchronization: The Network Time Protocol*, published by CRC Press (2006). Of course, it's already been reviewed on slashdot [8].

The only other dead-tree publication to cover this topic (so far) is *Expert Network Time Protocol: An Experience in Time with NTP* by Peter Rybaczyk (published by Apress, 2005, and reviewed by slashdot [9]).

For online documentation, there are the official pages written and maintained by Dr. Mills [10], the NTP FAQ [11], the Community Supported Documentation (CSD) [12], and many other pages linked from the NTP PSP Documentation page [13].

Top Five Most Common Problems with NTP (a.k.a. NTP Mini-FAQ)

One of the most common issues I've seen has been something along the lines of “I've done everything I'm supposed to, and it still doesn't work!” Here's a run-down of common causes:

1. They haven't punched a hole in their network firewall or host firewall software for bi-directional traffic on UDP port 123.

If you can't open port 123 for UDP in both directions, then you can't use the NTP daemon. The `ntpd` program can be used with a `-u` option to tell it to bind to a high-numbered port, which may be allowed by the firewall configuration, but this sort of option is not (yet) supported by `ntpd`.

2. They have unknowingly configured the software to ignore all responses that are not cryptographically signed.
Hint: The meaning of "notrust" changed between 4.1.x and 4.2.0. Disable "restrict notrust" unless you really understand what it's doing.
3. They may be running with SELinux enabled and not configured to allow the NTP software to update the system clock, etc.
4. They chose a set of upstream time servers that is not sufficient to allow the NTP algorithms to work correctly.
Hint: Use either just one or at least three or more, because the person with two clocks never knows what time it is [35].
In fact, you should use at least four or five upstream clocks if you want to be able to have one or more of them die or go insane, while your clock continues to function correctly. More information can be found in Section 5.3 of the CSD [16].
5. Their time zone is not correctly configured or is not properly displaying daylight savings time. The machine may be doing an adequate job in synchronizing the system clock to the upstream servers, but the presentation of this information is not correct.
This is not an NTP problem, since NTP operates exclusively in Universal Coordinated Time (UTC). The conversion from UTC to the local time zone is considered to be a representation issue for the OS and is outside the control of the NTP programs.
Make sure your time zone settings are correct in your `/etc/localtime` file, the `$TZ` environment variable, or otherwise as appropriate for your OS.

If you're having problems with NTP, we've got a whole section of the CSD devoted to troubleshooting [17] and describing common issues that people have, especially Section 9.1 on common hardware problems [18] and Section 9.2 on OS trouble [19]. If you've gone through all the documentation and you're still having problems, feel free to post on `comp.protocols.time.ntp` (which is gatewayed to the mailing list `questions@ntp.isc.org` [36]), or come see us on irc at `#ntp` on `irc.freenode.net`.

If you decide to use the irc channel, please be aware that there aren't many of us in the project and who monitor the channel on a regular basis, so you might need to wait a while for a response—perhaps several hours, or even a day or more. The mailing list/newsgroup is probably a better choice, unless you have a strong requirement for interactive support and you can afford to wait for it.

Also, if you see anything that could be improved in the CSD, or needs clarification, please feel free to sign up for a TWiki account and then dive right in to make the changes yourself. There's no way we can maintain all this information all by ourselves (in our nonexistent free time), which is why we created the CSD pages—so that everyone in the community would have the ability to contribute and correct information found there.

NTP Server Pool

If you are configuring your own NTP clients (or local NTP servers, from which your clients will be served), you should read the "Rules of

Engagement” [37], but you should also be aware of three different but related sets of public time servers. There is the list of public Stratum 1 time servers [20], which should only be used if you are setting up your own local NTP server(s) and are going to be serving local clients from it (them). There is the list of public Stratum 2 time servers [21], which can be used by individual clients as well as local time servers that will be redistributing time to their own local clients. Then there is the set of servers that comprise the “NTP Server Pool.” See the NTP PSP Pool Servers page [22] and the NTP Server Pool Web site [23] for more information on how the pool works.

The purpose of the NTP Server Pool is not to give you the best possible time, but instead to help you fill out your list of upstream servers that should be able to give you a reasonable baseline, from which your client/server can pick out the best available source.

There are now over 600 public time servers currently available through the NTP Server Pool (out of more than 700 total defined in the database, about 100 of which are currently not being advertised owing to various problems), with about 400 (total) in Europe and over 200 (total) in the United States. However, there is still a desperate need for additional time servers in the pool for other zones.

As of 24 April 2006, Ask Bjørn Hansen (the current maintainer of the NTP Server Pool) estimates that there are somewhere between two and six million client systems that are using the pool. You can help the project stay alive by contributing to the pool, if you have a static IP address [24].

Time Synchronization State of Practice

A question came up on the sage-members mailing list about the state of practice of time synchronization, and wondering why this doesn't seem to be more universally deployed at the server and client level.

I can't speak for the operational practices for most organizations, but I can say that more and more vendors are enabling NTP or SNTP code out-of-the-box. With recent versions of Windows, Microsoft ships an SNTP client, and they provide their own time servers for those clients to connect to. Apple has provided an NTP client in MacOS X for quite some time, making it easy to enable and configure and also providing time servers for those clients to connect to.

FreeBSD, NetBSD, and most other *BSD implementations not only ship NTP clients out-of-the-box, but they also enable them by default. Many Linux distributions are doing the same.

For vendors that configure their clients to use NTP by default, the practice within the community has been to encourage those vendors to also supply some time servers for those clients to use, or to configure their DNS in a particular way to allow them to make use of the servers provided through the NTP Server Pool project in a way that will minimize negative impact [38].

However, not all free/libre/open-source systems (FLOSS) platforms have felt that they have the ability to provide servers directly. Instead, some FLOSS platforms are actively encouraging their members to help provide additional machines for the NTP Server Pool. Debian is probably the best known in this regard, but Red Hat is providing their own Stratum 1 and Stratum 2 servers (see the aforementioned lists), as well as listing these machines in the pool.

Poul-Henning Kamp (from the FreeBSD project) runs a couple of restricted-access Stratum 1 time servers, and thanks to donations of GPS reference clock hardware from Meinberg, the NTP PSP also hopes to make available at least two Stratum 1 time servers of their own.

In addition, more vendors are shipping embedded hardware with NTP enabled, even though some of them make mistakes and misconfigure the firmware in their devices.

Most dedicated network devices (especially routers) come with NTP clients built in and can even act as NTP servers (although this may not be a good idea [25]; see “Sidebar,” p. 30).

At this point, the only observation I can make is that we must tend to get one of two situations:

1. Many people apparently configure this stuff and do so with relative ease and don't feel the need to tell anyone. Thus we don't hear about the positive cases.
2. Many other people probably still don't see the need to have good time sync on their machines. Thus we don't even know about the cases where we never even got considered.

But there were a surprising number of people on the sage-members list who spoke up and said that, based on their personal experience, network-wide NTP time synchronization was a much more common thing than you (or I) might think.

Updated Code

As of the time of this writing, the NTP PSP has recently released version 4.2.2 of the “Reference Implementation” of the NTP protocol [27]. By the time you see this article, we hope that many vendors will already have picked up this greatly improved code and incorporated it into the software they are shipping.

NTP 4.2.0 was released on 15 Oct 2003. Version 4.2.1 has been in development since, with many improvements made over the years. Unfortunately, many vendors have stuck with the “stable” 4.2.0 codebase, instead of tracking the improvements that have been made in the 4.2.1 development tree.

This has left many in the community with various known bugs and weaknesses that have already been fixed in the source tree, and they have found themselves in the uncomfortable position of either having to remove the vendor-provided code and replace that with code based on the source tarballs available from the NTP PSP download page [27] or waiting for someone to create a binary packaged version for them to download and install. Both approaches cause configuration management problems.

With the advent of version 4.2.2, we're going to be changing our release numbering scheme slightly [28], and we hope to be able to release new versions much more frequently than every few years. For now, we're targeting at least two new releases per year.

We are now also creating cryptographic hashes for the source tarballs, and we hope to start PGP-signing the announcements so that you can be reasonably sure that the code you're downloading is actually the code we released.

You may be interested to know that we also provide an RSS 2.0 feed of our current tarball information [39].

Sidebar: Time Server Abuse

In the April 2006 issue of *;login*: you may have read David Malone's article "Unwanted HTTP: Who Has The Time?" [42]. To summarize: there were thousands of clients worldwide running a program called Tardis, connecting to his server and obtaining a timestamp via HTTP. These clients were connecting as frequently as once an hour or even once a minute. Traffic volume was estimated at 30 GB/month, based on the initial data collected after enabling increased logging.

Although these clients were connecting via HTTP, this is a classic case of time server abuse by misconfigured clients. Unfortunately, it's not the only case, or even the most recent one.

A better-known case is found at the University of Wisconsin [1], where:

[They were] the recipient of a continuous large scale flood of inbound Internet traffic destined for one of the campus' public Network Time Protocol (NTP) servers. The flood traffic rate was hundreds-of-thousands of packets-per-second, and hundreds of megabits-per-second.

Ultimately, all this traffic was discovered to be the fault of misconfigured NetGear cable/DSL routers with embedded IP addresses as their set of pre-defined (and nonoverrideable) NTP time servers. At least NetGear was willing to work with UWisc and Dave Plonka to try to resolve the problem as well as possible, and the company has made a donation to the university for their help in locating and helping to get this problem fixed [2].

Just after the April 2006 issue of *;login*: came out, another instance of time server abuse came to the forefront. This time, it was the time server run by Poul-Henning Kamp at the Danish Internet Exchange, for the benefit of network providers in Denmark and their customers. Again, the fault lay with a commercial product with a bad default configuration (in this case, D-Link cable/DSL routers). However, this time the company took the notice by Poul-Henning to be an act of extortion, sending their lawyers after him.

The issue is now supposedly settled [3], so Poul-Henning has taken down the original notice, but you can still read about the story on other Web sites [4, 5, 6].

Wikipedia also has a good page on the subject of time server abuse [6], including a reference to a similar abuse problem that occurred between SMC and the CSIRO in Australia.

When all is said and done, one question you have to ask yourself is whether or not you want to be using hardware from a company that acknowledges the problems that they may accidentally create for others and works with you to try to resolve them.

What happens when you're on the other end of that pointy stick and your servers are being nuked off the Internet? What kind of response do you want to see from the company that is responsible?

REFERENCES

[1] <http://www.cs.wisc.edu/~plonka/netgear-sntp/>.

[2] <http://www.doit.wisc.edu/news/story.asp?filename=322>.

[3] <http://people.freebsd.org/~phk/dlink/>.

- [4] <http://yro.slashdot.org/article.pl?sid=06/04/07/130209>.
- [5] <http://www.lightbluetouchpaper.org/2006/04/07/when-firmware-attacks-ddos-by-d-link/>.
- [6] http://en.wikipedia.org/wiki/NTP_vandalism.
- [7] <http://www.udel.edu/PR/Messenger/02/1/where.html>.
- [8] <http://books.slashdot.org/article.pl?sid=06/05/15/143251>.
- [9] <http://books.slashdot.org/article.pl?sid=05/08/16/0344212>.
- [10] <http://www.eecis.udel.edu/~mills/ntp/html/index.html>.
- [11] <http://www.ntp.org/ntpfaq/NTP-a-faq.htm>.
- [12] <http://ntp.isc.org/support>.
- [13] <http://ntp.isc.org/doc>.
- [14] <http://www.ietf.org/html.charters/ntp-charter.html>.
- [15] <https://lists.ntp.isc.org/mailman/listinfo/ntpwg>.
- [16] <http://ntp.isc.org/bin/view/Support/SelectingOffsiteNTPServers>.
- [17] <http://ntp.isc.org/bin/view/Support/TroubleshootingNTP>.
- [18] <http://ntp.isc.org/bin/view/Support/KnownHardwareIssues>.
- [19] <http://ntp.isc.org/bin/view/Support/KnownOsIssues>.
- [20] <http://ntp.isc.org/s1>.
- [21] <http://ntp.isc.org/s2>.
- [22] <http://ntp.isc.org/pool>.
- [23] <http://www.pool.ntp.org/>.
- [24] <http://www.pool.ntp.org/join.html>.
- [25] http://ntp.isc.org/bin/view/Support/DesigningYourNTPNetwork#Section_5.6.
- [26] <http://ntp.isc.org/>.
- [27] <http://ntp.isc.org/download>.
- [28] <http://ntp.isc.org/bin/view/Main/ReleaseNumberingScheme>.
- [29] <http://ntp.isc.org/bin/view/Main/DonatingToTheProject>.
- [30] <http://ntp.isc.org/bin/view/Main/OurDonors>.
- [31] <http://ntp.isc.org/donat>.
- [32] <news:alt.sysadmin.recovery>.
- [33] <http://www.faqs.org/faqs/sysadmin-recovery/index.html>.
- [34] <http://www.eecis.udel.edu/~mills/bio.html>.
- [35] http://www.quotationspage.com/quotes/Segal's_Law.
- [36] <https://lists.ntp.isc.org/mailman/listinfo/questions>.
- [37] <http://ntp.isc.org/bin/view/Servers/RulesOfEngagement>.
- [38] <http://www.pool.ntp.org/vendors.html>.
- [39] <http://ntp.isc.org/rss/releases.xml>.
- [40] <http://ntp.isc.org/ietf>.
- [41] <http://www.eecis.udel.edu/~mills/database/papers/history.pdf>.
- [42] <http://www.usenix.org/publications/login/2006-04/pdfs/malone.pdf>.

RICHARD MCDUGALL AND
JAMES LAUDON

multi-core microprocessors are here



Richard McDougall is a Distinguished Engineer at Sun Microsystems, specializing in operating systems technology and systems performance.

richard.mcdougall@sun.com



James Laudon is a Distinguished Engineer and a Niagara processor line architect at Sun Microsystems. His specialties are hardware multi-threading, multi-processing, and performance modeling.

james.laudon@sun.com

THE ADVENT OF SYMMETRIC MULTI-Processing (SMP) added a new degree of scalability to computer systems. Rather than deriving additional performance from an incrementally faster microprocessor, an SMP system leverages multiple processors to obtain large gains in total system performance. Parallelism in software allows multiple jobs to execute concurrently on the system, increasing system throughput accordingly. Given sufficient software parallelism, these systems have proved to scale to several hundred processors.

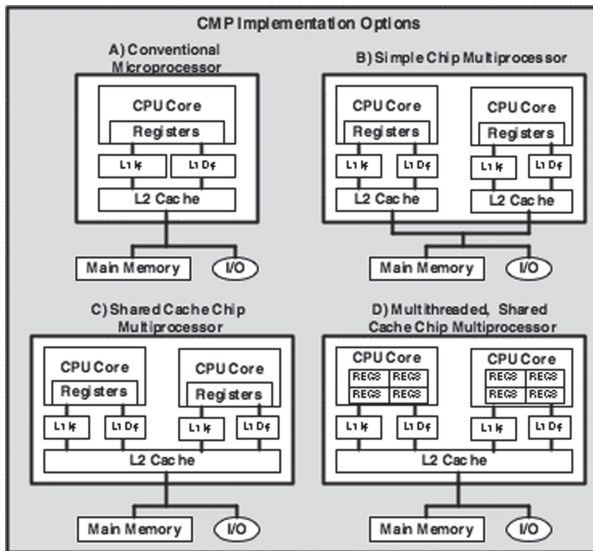
More recently, a similar phenomenon is occurring at the chip level. Rather than pursue diminishing returns by increasing individual processor performance, manufacturers are producing chips with multiple processor cores on a single die. For example, the AMD Opteron and UltraSPARC IV now provide two entire processor cores per die, providing almost double the performance of a single-core chip. The Sun UltraSPARC T1 (Niagara) processor packs eight cores onto a single die and can provide up to eight times the performance of the dual-core UltraSPARC processors.

There are three main types of multi-core processors:

- Simple multi-core processors have two complete processors placed on the same die or package (e.g., the dual-core AMD Opteron processor).
- Shared-cache multi-core processors consist of two complete cores, sharing some levels of cache, typically Level 2 (L2) or Level 3 (L3) (e.g., the Sun UltraSPARC IV+ and Intel Woodcrest processors, which share caches between two cores).
- Multi-threaded multi-core processors have multiple cores, with multiple threads within each core (e.g., the Sun UltraSPARC T1).

As processor frequency increases, the amount of time spent waiting for memory stalls increases. This means that placing multiple cores on a die can increase performance, but ultimately multi-threading in the CPU is critical to overcoming memory latency stalls. Implementations that use multi-cores *plus* hardware threading have recently proven to give superior performance at much lower power consumption.

These new multi-core processors are bringing what was once a large multiprocessor system



**FIGURE 1: FOUR TYPES OF
CHIP-LEVEL MULTIPROCESSING**

down to the chip level, providing a significant level of throughput in a small package, with extremely low power consumption. In the case of the Sun UltraSPARC T1 processor with eight cores and four threads per core, power consumption of the chip is less than 70 watts. We have effectively reduced the equivalent throughput of a refrigerator-sized server (such as the Sun E6000 30-way, circa 2000) into a 1U single-processor machine, using less than 180 watts.

In this article, we'll contrast the different types of multi-core approaches and look at the performance advantages and tradeoffs. We will also discuss the potential implications for systems and application software.

Multi-Core Processors

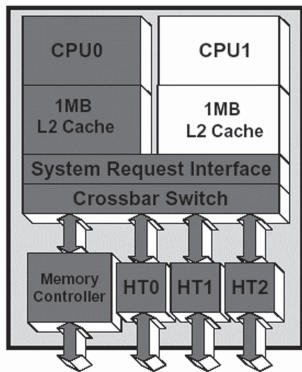


FIGURE 2: THE AMD OPTERON USES SIMPLE CHIP MULTI-PROCESSING, WITH THE TWO CORES SHARING ACCESS TO MEMORY AND TO OTHER BUSES (HYPERTRANSPORT)

The latest dual-core AMD Opteron is an example of a multi-core design. The chip has two complete processor cores, sharing a bus to memory. As shown in the left of Figure 2, it is almost identical to its single-core predecessor; the second core is a complete duplication of the first, including its pipeline and caches. From a performance perspective, the chip behaves much like a dual-processor SMP machine, albeit with some potential contention for memory bandwidth through the shared path to memory. From a software perspective, the chip appears almost indistinguishable from a dual-processor system. Software threads are scheduled onto the processor cores by the operating system—at least two threads are required to keep both cores busy.

Multi-Threading

Processor designers have found that since most microprocessors spend a significant amount of time idly waiting for memory, software parallelism can be leveraged to hide memory latency. Since memory stalls typically take on the order of 100 processor cycles, a processor pipeline is idle for a significant amount of time. Table 1 shows the amount of time spent waiting for memory in some typical applications, on 2 GHz processors. For example, we can see that for a workload such as a Web server, there are sufficient memory stalls such that the average number of machine cycles is 1.5—2.5 per instruction, resulting in the pipeline waiting for memory up to 50% of the time.

Waiting Application	Typical Number of Cycles per Instruction	Percent of Time for Memory Stalls
Transaction database	3–6	>75%
Web server	1.5–2.5	~50%
Decision support database	1–1.5	~10–50%

TABLE 1: EXAMPLES OF THE PERCENTAGE OF TIME SPENT WAITING FOR MEMORY ACCESS TO COMPLETE (STALLS OR WAIT STATES)

In Figure 3, we can see that less than 50% of the processor's pipeline is actually being used to process instructions; the remainder is spent waiting for memory. By providing additional sets of registers per processor pipeline, multiple software jobs can be multiplexed onto the pipeline, a technique known as simultaneous multi-threading (SMT). Threads are switched on to the pipeline when another blocks or waits on memory, thus allowing the pipeline to be utilized potentially to its maximum. Figure 4 shows an example with four threads per core. In each core, when a memo-

ry stall occurs, the pipeline switches to another thread, making good use of the pipeline while the previous memory stall is fulfilled. The tradeoff is latency for bandwidth; with enough threads, we can completely hide memory latency, provided there is enough memory bandwidth for the added requests. Successful SMT systems typically allow for very high memory bandwidth from DRAM, as part of their balanced architecture.

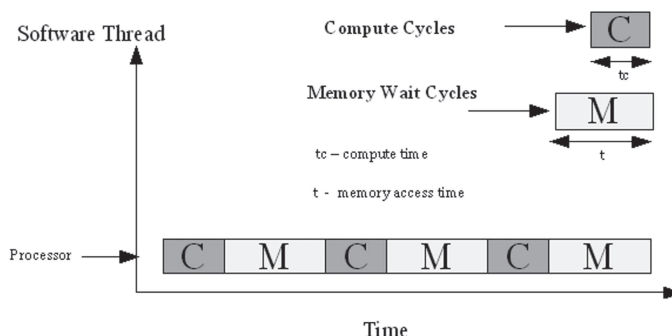


FIGURE 3: IN SINGLE-THREADED PROCESSOR DESIGNS, MORE TIME IS SPENT WAITING FOR MEMORY THAN ACTUALLY PROCESSING DATA

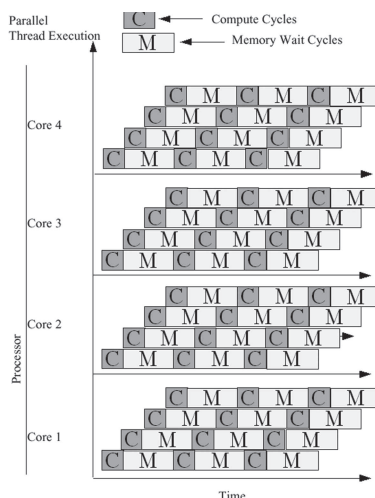


FIGURE 4: SIMULTANEOUS MULTI-THREADING DESIGNS ALLOW THE PROCESSOR TO CONTINUE WITHOUT WAITING FOR MEMORY BY QUICKLY SWITCHING BETWEEN THREADS

SMT has a high return on performance in relation to additional transistor count. For example, a 50% performance gain may be realized by adding just 10% more transistors with an SMT approach, in contrast to making the pipeline more complex, which typically affords a 10% performance gain for a 100% increase in transistors. Also, implementing multi-core alone doesn't yield optimal performance—the best design is typically a balance of multi-core and SMT.

Contrasting the Different Types of Threading

There are three main ways to multi-thread a processor: *coarse-grain*, *vertical*, and *simultaneous* [1].

With coarse-grain threading, a single thread occupies the full resources of the processor until a long-latency event such as a primary cache miss is

encountered, as shown in Figure 3. At that point, the pipeline is flushed and another thread starts executing, using the full pipeline resources. When that new thread hits a long-latency event, it will yield the processor to either another thread (if more than two are implemented in hardware) or the first thread (assuming its long-latency event has been satisfied). Coarse-grain threading has the advantage that it is less of an integral part of the processor pipeline than either vertical or simultaneous multi-threading and can more easily be added to existing pipelines. However, coarse-grain threading has a big disadvantage: the high cost of switching between threads. When a long-latency event such as a cache miss is encountered, all the instructions in the pipeline behind the cache miss must be flushed from the pipeline and execution of the new thread starts filling the pipeline. Given the pipeline depth of modern processors—as many as 16 instructions in Intel-styled processors—this means a thread switch cost in the tens of processor cycles. This high switch cost means that coarse-grain threading cannot be used to hide the effects of short pipeline stalls owing to dependencies between instructions and even means that the thread-switching latency will occupy much of the latency of a primary cache miss/secondary cache hit. As a result, coarse-grain multi-threading has been primarily used when existing, single-threaded processor designs are extended to include multi-threading.

The two remaining techniques for threading, vertical threading (VT) and SMT, switch threads on a much finer granularity (and not surprisingly are referred to as fine-grained multi-threading). On a processor capable of multiple instruction issue, an SMT processor can issue instructions from multiple threads during the same cycle, whereas a VT processor limits itself to issuing instructions from only one thread each cycle (see Figure 4). On a single-issue processor there is no difference between VT and SMT, as only one instruction can be issued per cycle, but since there is no issue of instructions from different threads in the same cycle, single-issue, fine-grained multi-threaded processors are labeled VT.

Both SMT and VT solve the thread switch latency problem by making the thread switch decision part of the pipeline. The threading decision is folded in with the instruction issue logic. Since the issue logic is simply trying to fill the pipeline with instructions from all of the hardware threads, there is no penalty associated with switching between threads. However, a little extra complexity gets added to the issue logic, as it now needs to pick instructions from multiple ready threads. This additional issue logic complexity is fairly small (certainly much smaller than all the other issue-related complexity that is present in a modern superscalar processor) and well worth it in terms of performance. The advantages of SMT and VT are that very short pipeline latencies (all the way down to a single cycle) can be tolerated by executing instructions from other threads between the instructions with the pipeline dependency.

The ability to switch threads at no cost is the key to enabling the impressive performance of the new processors.

Chip-Level Multi-Threading

A new generation of processors that use Chip-Level Multi-Threading (CMT) combine multi-core with SMT, thereby providing a large core count and the ability to extract the maximum performance from each core. The UltraSPARC T1 is an example of a CMT processor design.

Most people are familiar with the hyperthreaded Intel processors, which employ SMT. They support two threads in hardware and show modest gains on some parallel workloads. Given that SMT is the most aggressive of the three threading schemes, one would expect SMT to deliver the highest performance, but in general the performance gains seen from hyperthreading are small (and sometimes hyperthreading actually leads to performance losses). However, the gains seen from hyperthreading are not limited by the SMT but more by the memory system, and unfortunately the Intel hyperthreading implementation delivers a misleading message about the performance to be gained from fine-grained multi-threading.

The UltraSPARC T1, in contrast, was built from the ground up as a multi-threaded chip multiprocessor, and each of the eight pipelines employs vertical threading of four hardware threads. The eight pipelines in the UltraSPARC T1 are short (six stages), and one might be tempted to employ the slightly simpler coarse-grain threading. However, even on the UltraSPARC T1, the gains from vertical threading over coarse-grained multi-threading ended up being substantial. In fact, the very earliest proposals for what became the UltraSPARC T1 employed coarse-grain threading. Rather quickly, the modest additional complexity of vertical threading was traded off against its performance gains and the switch to vertical threading was made. The performance and performance/watt numbers from the UltraSPARC T1 show that it's been worth it!

The UltraSPARC T1 processor uses eight cores on a single die. Each core has four threads sharing a pipeline, an L1 instruction cache, a data cache, and a memory management unit (MMU).

The UltraSPARC T1 architecture has the following characteristics:

- Eight cores, or individual execution pipelines, per chip.
- Four hardware threads (strands) or active thread contexts that share a pipeline in each core. Each cycle of a different hardware strand is scheduled on the pipeline in round-robin order.
- A total of 32 threads per UltraSPARC T1 processor.
- A strand that stalls for any reason is switched out and its slot on the pipeline is given to the next strand automatically. The stalled strand is inserted back in the queue when the stall is complete.
- Cores that are connected by a high-speed, low-latency crossbar in silicon. An UltraSPARC T1 processor can be considered SMP on a chip.
- Hardware strands that are presented by the operating system as a processor. For example, Solaris and Linux see each thread as a separate processor.
- Cores that have an instruction cache, a data cache, an instruction translation-lookaside buffer (iTLB), and a data TLB (dTLB) shared by the four strands.
- Strands defined by a set of unique registers and logic to control state.
- A 12-way associative unified L2 on-chip cache. Each hardware strand shares the entire L2 cache. Historically, the level of associativity we typically see is around 4 for a non-CMT core, but with 32 strands sharing the L2, larger associativity is critical.
- Low-latency Double Data Rate 2 (DDR2) memory to reduce stalls. Four on-chip memory controllers provide high memory bandwidth (with a theoretical maximum of 25 gigabytes per second).
- An operating system scheduler that schedules LWPs on UltraSPARC T1 hardware strands. It is the task of the hardware to schedule strands in the core.

- A modular arithmetic unit (MAU) for each core that supports modular multiplication and exponentiation to help accelerate Secure Sockets Layer (SSL) processing.

The layout of a system implemented with the UltraSPARC T1 processor is shown in Figure 5 (which, for clarity, does not show the four memory controllers between the L2 cache and the four banks of SDRAM).

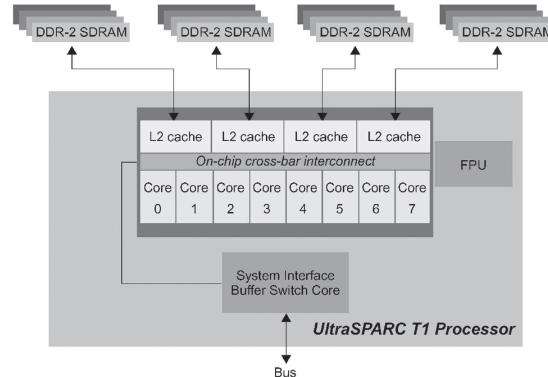


FIGURE 5: ULTRASPARC T1 CHIP LAYOUT

The Power Advantages of CMT

More complex pipelines use a significantly larger amount of power for little gain in performance. For example, when the clock rate of the Intel Celeron went from 1.2 GHz to the Pentium 4 at 2.2 GHz, power increased from 29 to 55 watts, but there was only a 20% performance improvement (measured using the Business Winstone Benchmark 2001).

Keeping the pipeline simple significantly reduces power consumption. In aggressive CMT architectures, such as the UltraSPARC T1, power per core is as low as 8 watts. This is achieved by keeping the pipeline simple, for example using a single-issue pipeline and eliminating many of the nonessential pipeline features, such as memory prefetching.

As an example, we can look at the throughput and power consumption of a typical Web workload, represented by the SPECweb2005 benchmark (see Table 2). By measuring the ratio of performance against watts consumed, we can contrast the performance/power efficiency of the system.

* From sun.com: "Sun Fire T1000 (8 cores, 1 chip) compared to Dell PowerEdge 2850 (4 cores, 2 chips). Results from www.spec.org as of May 30th 2006. Dell power measurements taken from the Dell Power Calculator, 03/06/06, posted: http://www1.us.dell.com/content/topics/topic.aspx/global/products/pedge/topics/en/config_calculator?c=us&cs=555&l=en&s=biz. System configured with 2 x Dual Core 2.8GHz processors, 16GB RAM, 2 x USCSI disks. Sun Fire T1000 server power consumption taken from measurements made during the benchmark run."

	UltraSPARC T1	2x Dual Core @ 2.8 GHz
Space (rack units)	1	2
Watts (system)	188	450
Performance	10,466	4850
Performance per watt	55.7	10.8

TABLE 2: ULTRASPARC T1 PERFORMANCE CHARACTERISTICS*

In this example, the CMT design provides roughly twice the throughput of the nonthreaded system, at half the power [2].

The Software View of CMT

A very simplistic view of a CMT system is that its software performance is much like that of an SMP system with the number of processors equal to

the number of strands in the chip, each with slightly reduced processing capability. Software threads are scheduled by the operating system onto individual hardware threads, and strands are scheduled onto the pipeline by a hardware scheduler. The number of software threads required to keep the core busy varies from one to many, depending on the ratio of memory stalls to compute events.

SINGLE-THREAD PERFORMANCE

Since each hardware thread is sharing the resources of a single processor core, each thread has some fraction of the core's overall performance. Thus, an eight-core chip with thirty-two hardware threads running at 1 GHz may be somewhat crudely approximated as an SMP system with thirty-two 250 MHz processors. Applications which are single-threaded will see lower performance than that of a processor with a more complex pipeline. The reduction in performance for single-threaded applications will depend on whether the application is more memory- or compute-bound, with compute-bound applications showing the largest difference in performance.

CMT LOVES "THROUGHPUT WORKLOADS"

To achieve per-thread performance with a significant increase of throughput and a reduction in power requires concurrency in the software. Applications that are server-oriented typically have bountiful amounts of concurrency. Typically these types of throughput applications are driven by a large number of connections or users, meaning there is enough natural concurrency to exploit SMP or CMT systems.

For a throughput-oriented workload with many concurrent requests (such as a Web server), the marginal increase in response time is virtually negligible, but the increase in system throughput is an order of magnitude over a non-CMT processor of the same clock speed.

A number of classes of applications benefit directly from the ability to scale throughput with CMT processors:

- Multi-threaded native applications: Multi-threaded applications are characterized by having a small number of highly threaded processes. Examples of threaded applications include Lotus Domino or Siebel CRM.
- Multi-process applications: Multi-process applications are characterized by the presence of many single-threaded processes. Examples of multi-process applications include the Oracle database, SAP, and PeopleSoft.
- Java applications: Java applications embrace threading in a fundamental way. Not only does the Java language greatly facilitate multi-threaded applications, but the Java Virtual Machine is a multi-threaded process that provides scheduling and memory management for Java applications. Java applications that can benefit directly from CMT resources include application servers such as Sun's Java Application Server, BEA's Weblogic, IBM's Websphere, and the open-source Tomcat application server. All applications that use a Java 2 Platform, Enterprise Edition (J2EE platform) application server can immediately benefit from CMT technology.
- Multi-instance applications: Even if an individual application does not scale to take advantage of a large number of threads, it is still possible to gain from CMT architecture by running multiple instances of the application in parallel. If multiple application instances require some

degree of isolation, virtualization technology (for the hardware of the operating system) can be used to provide each of them with its own separate and secure environment.

We've spent a great deal of time evaluating server application performance of CMT architectures; my blog [3] contains a good starting summary of the results we've had.

THOUGHTS ABOUT SOFTWARE SCALING

On a multi-threaded microprocessor, each hardware thread appears to the operating system as an individual processor. The ability of system and application software to exploit multiple processors or threads simultaneously is becoming more important than ever. As CMT hardware progresses, software is required to scale accordingly to fully exploit the parallelism of the chip.

Thus, bringing this degree of parallelism down to the chip level represents a significant change to the way we think about scaling. Since the cost of a CMT system is close to that of recent low-end uniprocessor systems, it's inevitable that even the cheapest desktops and servers will be highly threaded. Techniques used to scale application and system software on large enterprise-level SMP systems will now frequently be leveraged to provide scalability even for single-chip systems. We need to consider the effects of the change in the degree of scaling at the low end on the way we design applications, on which operating system we choose, and on the techniques we use to deploy applications.

Conclusion

In today's data centers, power and space are valuable resources. The advantages brought about by CMT are inevitable for optimizing these resources. The aggressiveness of CMT varies with different system designs; we expect to see four-core systems from AMD in the near future, UltraSPARC follow-ons are expected to increase the thread count, and Intel is discussing some radical multi-core designs. The interesting debate will be about the number of cores to have and to what degree each approach will utilize vertical threading within each core to hide memory latency. It's going to be a fun time in this space. Stay tuned!

ACKNOWLEDGMENTS

Thanks are owed to Denis Sheahan, Performance Specialist in the UltraSPARC T1 group for the UltraSPARC T1 specifications, and the PAE performance group at Sun Microsystems for providing the performance characterization data of workloads on CMT.

REFERENCES

- [1] Jim Laudon's blog: <http://blogs.sun.com/jlaudon>.
- [2] T1000 Server Benchmarks:
<http://www.sun.com/servers/coolthreads/t1000/benchmarks.jsp>.
- [3] Richard's Blog: <http://blogs.sun.com/rmc>.

TIMO SIVONEN

measuring performance of FreeBSD disk encryption



Timo Sivonen is a Senior Consultant at International Network Services (INS). He lives in the U.K. with his wife and son.

timo.sivonen@ins.com

DISK ENCRYPTION IS ONE OF THOSE services absolutely invaluable to laptop owners and possibly even suspicious if used by others. Yet, as with many other security services, you have to ask yourself what is the threat you are trying to protect against or the problem you are trying to solve.

Low-level disk encryption, which encrypts everything on the raw partition including the file system, does keep your files secure if the passphrase to open the partition is not known. Yet, if the encryption layer lies below the file system, almost the only way to create an encrypted backup of an encrypted file system is to copy the partition to an image file with `dd(1)`, which is both cumbersome and inefficient.

In my day job I have to store confidential client data on my laptop and I have to back the laptop up regularly. However, data encrypted on the hard disk must remain encrypted on backups. Furthermore, I prefer multiple 650 MB encrypted file systems to one 4 GB volume, since I can back the smaller volumes up on individual CDs as opposed to using more cumbersome tapes. Like everyone else, I also use a flash drive, but instead of carrying around several flash drives for different operating systems, I wanted to consolidate my flash drives to one and copy files between UNIX boxes using an encrypted file system without losing UNIX/Windows interoperability offered by FAT32.

Since version 5, FreeBSD has featured GEOM-Based Disk Encryption (GBDE). In brief, GBDE is a software-only disk-encryption service that uses AES-128 to encrypt the contents of the designated raw device and the master encryption key is stored under AES-256. GBDE seemed to offer exactly what I was looking for except for the fact that all documentation pointed toward encrypting raw partitions. However, FreeBSD also has a facility called `md(4)`, or memory disk, which, among other things, allows you to read and write image files as raw devices. Combined with `newfs`, one is able to write a file system on the image and mount it like any other disk device. All that is required is to set up the GBDE layer on top of `md`, write a file system on the GBDE device, and mount the device.

Setting It Up

The initialization of a new encrypted memory disk is a relatively straightforward operation, although a few caveats do exist. (Allocate the image file, create the memory disk, label the device, initialize the encrypted device, attach the encrypted device, create the file system on the encrypted device, and mount it.) The most critical advice is to *not* enable UFS soft updates, as these may cause devfs to lock the (GBDE-encrypted) file system. In other words, a locked file system is permanently busy and can't be unmounted even in system shutdown, which ultimately may corrupt the file system. This may not happen if running FreeBSD normally, on bare metal hardware, but it is certainly possible under VMware. Later in this text we will discover that not using soft updates has little impact on file system performance.

A GBDE-encrypted file system is created on a memory disk in seven steps:

1. # dd if=/dev/random of=/home/user/gbdeimg bs=1m count=650
2. # mdconfig -a -t vnode -f /home/user/gbdeimg
md1
3. # bsdlabel -w md1 auto
4. # gbde init /dev/md1c -L /etc/gbde/gbdeimg.key
Enter new passphrase:
Reenter new passphrase:
5. # gbde attach /dev/md1c -l /etc/gbde/gbdeimg.key
Enter passphrase:
6. # newfs /dev/md1c.bde
7. # mount /dev/md1c.bde /mnt

The explanation of each step is as follows:

1. Allocate the disk space by writing 650 MB of random data from /dev/random to the designated image file.
2. Create the memory disk, through which the disk image will be accessed. Note that mdconfig(8) will print out the assigned md device, unless explicitly told which device to use.
3. Label the newly created memory disk to enable creation of the encryption layer and the file system at a later stage.
4. Initialize the encryption layer. Since the encryption key is specified separately from the disk image, those really concerned about their laptop security can save the encryption key(s) on a separate flash drive, which must be attached and mounted in order to prepare and mount encrypted partitions.
5. Once the encrypted device has been initialized, it has to be attached in order to write the file system on the encrypted device. This operation will also create a new instance of the disk device with the .bde suffix to denote GBDE encryption. Hence, all file system operations must be made with the .bde device.
6. Write the file system on the encrypting device (i.e., md1c.bde in the example that follows). Remember to not enable soft updates, since devfs may become upset by this. Furthermore, it was discovered that there may not be any significant performance improvement over UFS file systems that do not use soft updates.
7. Once a file system has been created, the encrypting device can be mounted normally. All updates to the file system are written to the image file, which can be backed up using tar or cpio or burned on a CD when unmounted and taken off-line.

One unhappy discovery in this journey was that an encrypted file system cannot be attached from read-only media. You can create a memory disk from a read-only image, and you can mount UFS file systems read-only, but you cannot attach an encrypted file system if its disk image resides on, for example, a CD-ROM. This discovery was a slight setback but not critical: After all, PGP disks formatted with NTFS must be copied from the backup media to a disk first, since NTFS cannot be mounted read-only either.

Choices and Performance

FreeBSD 6 introduced a new encrypted file system, GELI (GEOM_ELI cryptographic GEOM class). Unlike GBDE, which is a software-only facility, GELI utilizes the `crypto(4)` framework and is able to use encryption hardware if available.

GELI also gives more choice in algorithm selection and key length. Whereas GBDE only uses AES-128 for encrypting the disk contents, the users of GELI can choose from 3DES, AES, and Blowfish with key lengths of 128 or 256 bits. 3DES has a fixed key length of 192 bits. With this kind of selection it may seem difficult to choose the right encryption algorithm and balance security with performance.

To answer these questions and to be able to make educated decisions on which disk encryption to use, or which algorithm and how long a key to select, I devised a test plan to give some insight into the performance of FreeBSD disk encryption. Understanding that the results would be affected by the type of hardware available, even when executing tests on an otherwise idle system, one has to accept a certain distribution in results since, after all, UNIX is a time-sharing operating system and does not guarantee any throughput time for a command or a system call. My plan was to measure processing times for write and read operations when writing and reading a single 100 MB file or writing and reading 100 1-MB files. The files would only contain random binary data as read from `/dev/random`. Hence, the following tests were conceived:

- Writing and reading a 100 MB file and 100 1-MB files using GBDE and GELI on a memory disk.
- Writing a 100 MB file and 100 1-MB files using GBDE and GELI/AES-128 on a raw disk partition. The purpose of this test was to establish the baseline on what type of throughput one may expect from encrypting file systems. Without the `md` layer in the way, one would expect a visible improvement in performance.
- Writing a 100 MB file and 100 1-MB files using GELI on a memory disk, with a variety of ciphers. The purpose of this test was to compare the performance of GELI when a cipher other than the default AES-128 is used.

Each write and read test was repeated three times and an average was calculated to get an approximate time of how long it would take to process an operation. Since the absolute processing times depend on the underlying hardware, one should not measure absolute seconds but, instead, compare approximate processing times among different disk-encryption methods.

The Results

The test system was an IBM ThinkPad T21 laptop with an 850 MHz clock. The boot disk was a 40 GB Fujitsu MHV2040AH and the test disk was a 6 GB IBM DARA206000 running at 4,200 rpm. The tests were conducted in

multi-user mode by writing files from the boot disk to the test disk and vice versa. There were no encrypted file systems active on the boot disk. The tests were made using FreeBSD 6.1.

The most interesting test was to write a 100 MB file of random data and 100 files of 1 MB of random data to a directory. This test was conducted for GBDE, GELI, and a plain UFS file system on a memory disk. The test results are illustrated in Figure 1.

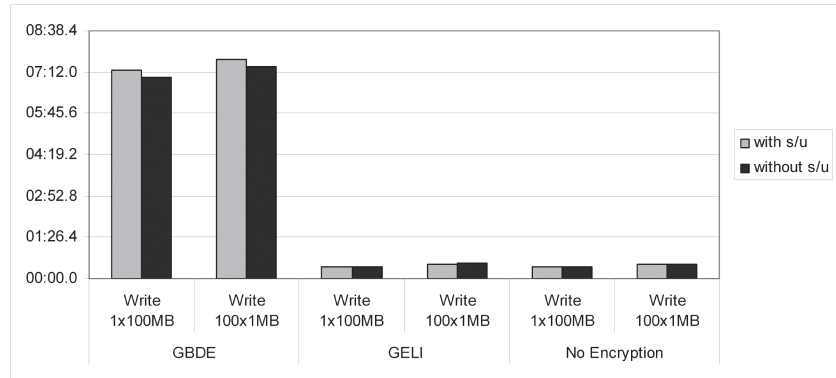


FIGURE 1: WRITING TO AN MD DEVICE

According to these results, GBDE does not perform very well at all when writing on a memory disk. In comparison, write performance of GELI is roughly equivalent to plain UFS on a memory disk. One should also note that enabling or disabling soft updates makes practically no difference to performance.

Since results on a memory disk showed a visible difference in performance between GBDE and GELI, it was measured whether there is any difference in performance for these two disk-encryption systems when writing on a raw partition, without using a memory disk layer. The results can be found in Figure 2.

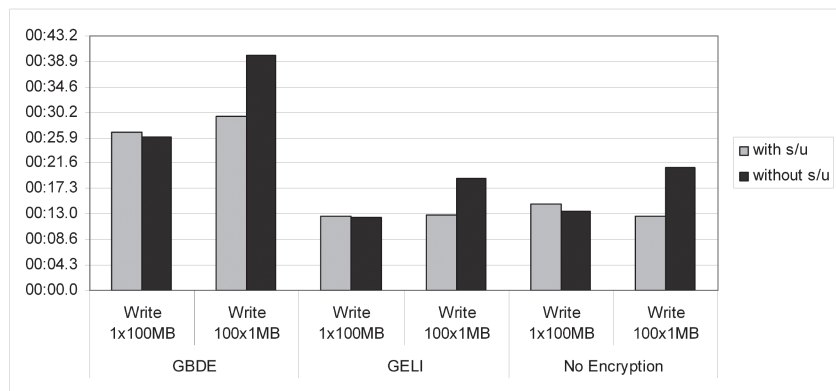


FIGURE 2: WRITING TO A DISK DEVICE

The results show that GBDE is still visibly slower than GELI but the difference between the two is no longer as dramatic. However, I am unable to explain why GBDE performs so poorly with an md device, whereas the performance difference with GELI on a raw partition is not that significant.

As one would expect, reading from a memory disk is much faster than writing (Figure 3, next page). There is little difference between GBDE and GELI, although GBDE may be slightly slower than GELI or a plain memory disk. In fact, the results show that on a moderately fast processor the time spent encrypting or decrypting is negligible compared to the disk transfer rates.

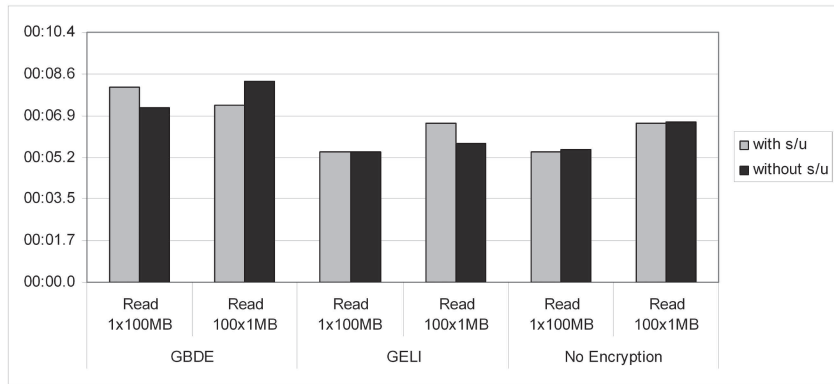


FIGURE 3: READING FROM AN MD DEVICE

The final test was to measure whether different encryption algorithms or key lengths would affect the performance of GELI in any way. The presumption was that AES and Blowfish would probably perform better than 3DES when encryption operations are performed in software. However, since no crypto hardware was available for the testing, 3DES was likely to be the worst performer.

The first presumption of performance seemed to be correct when write times using different ciphers were measured (Figure 4). 3DES-192 was clearly slower than AES-256 or AES-128 when writing a single 100 MB file or 100 1-MB files. This result was completely expected, since 3DES has to do three crypto operations (i.e., encrypt, decrypt, and encrypt again), whereas AES does only one.

A more interesting observation was the relative performance of AES-128 and AES-256. The processing times of these two were practically the same, which leads to the conclusion that one might as well use AES-256 with GELI, since the security benefits of a longer key are much bigger than the insignificant processing impact the longer key might have.

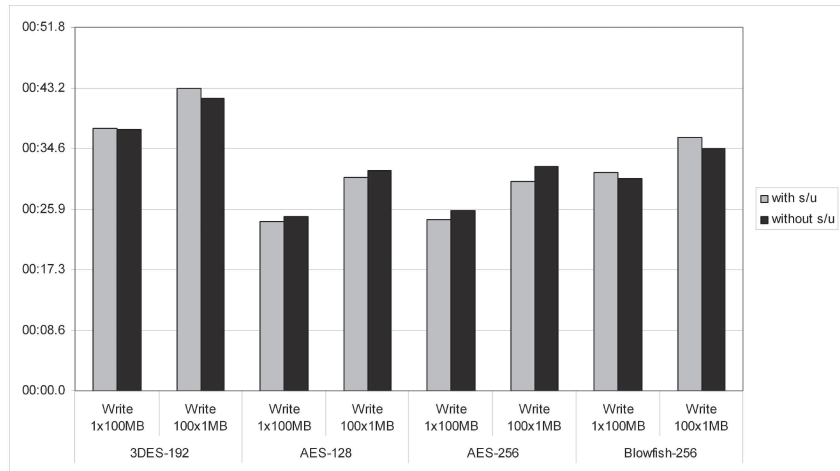


FIGURE 4: WRITING TO AN MD DEVICE WITH DIFFERENT CIPHERS

The other surprise was the performance of Blowfish 256. I would have expected Blowfish to be on a par with AES but, interestingly enough, it seems to fall between 3DES 192 and AES. Unless Blowfish has cryptographic properties that AES does not have, one would be tempted to prefer AES to Blowfish. However, a test with a larger sample may be required to determine whether AES actually is faster than Blowfish or whether the result was only a fluke.

Conclusions

An approach using GBDE- and GELI-encrypted file systems with a memory disk, md(4), was presented. The advantage of this method is the ability to create several encrypted file systems of different sizes on a normal UFS file system. With an encrypted file system of 650 MB it is possible to back up the encrypted image on a CD and save the contents of the file system in an encrypted format. Furthermore, assuming that the UFS file system is large enough, it is possible to create new encrypted file systems and mount them on an ad-hoc basis.

The relative performance of GBDE and GELI was also discussed. It was discovered that GBDE is significantly slower than GELI using AES 128 on a memory disk. GBDE was slightly slower than GELI on a raw disk partition but there is no major performance difference between the two. This leads to the conclusion that GBDE is unsuitable for use on a memory disk and, if used, its use should be limited to removable devices such as flash drives and floppies.

Since GELI uses the crypto(4) framework and has multiple ciphers, the relative performance of different ciphers was also measured. No crypto hardware was available for these tests. It was discovered that AES with a 256-bit encryption key performed as well as AES using a 128-bit key, thus leading to the conclusion that one should be using the longer key because of its stronger security.

It was also discovered that AES performed better than Blowfish or 3DES, thus making it the cipher of choice. This observation may have resulted from the small sample, and further investigations may be called for.

MICHAEL J. FREEDMAN

automating server selection with OASIS



Michael J. Freedman is a doctoral student at NYU, currently visiting Stanford University, and received his M.Eng. and S.B. degrees from MIT. His research interests are security, distributed systems, and cryptography. He is the author of the Coral Content Distribution Network (<http://www.coralcdn.org/>) and OASIS (<http://oasis.coralcdn.org>).

mfreed@cs.nyu.edu

OASIS PROVIDES A PUBLICLY AVAILABLE, locality-aware server-selection infrastructure. Replicated servers adopting OASIS each run a small application that communicates with the OASIS infrastructure, sharing information about their current load levels and their measured network response times to selected IP addresses. OASIS in turn can redirect unmodified clients to nearby and/or lightly loaded live replica servers. In this article, I explain how OASIS works, provide some performance analysis, and describe how services can start using OASIS.

OASIS (Overlay Anycast Service Infrastructure) has been publicly deployed since November 2005 on PlanetLab [10], a distributed testbed running at over 300 academic and industry sites. It has already been adopted by a number of services [1, 2, 4, 7, 8, 11, 12]. OASIS supports a variety of protocols—currently DNS, HTTP, and RPC—that redirect unmodified clients for server selection and that expose its geolocation and distance-estimation functionality to OASIS-aware hosts.

Why This Matters

High-volume Web sites are typically replicated at multiple locations for performance and availability. Content distribution networks amplify a Web site's capacity by serving clients through a large network of Web proxies. File-sharing, instant messaging, and VoIP systems use rendezvous servers to bridge hosts behind NATs.

In all of these examples, system designers must tackle the problem of *server selection*: After deploying servers at various locations throughout the Internet, they must now direct clients to ones that are appropriately nearby or unloaded. Or, posed more concretely, when accessing a replicated Web site, from which mirror should a user download?

This server-selection problem is not merely academic: The performance and cost of such systems depend highly on clients' choice of servers. File download times can vary greatly based on the locality and load of the chosen replica. A service provider's costs may depend on the load spikes that the selection mechanism produces, as many

data centers charge customers based on the 95th-percentile bandwidth usage over all five-minute periods in a month.

Unfortunately, common techniques for replica selection produce suboptimal results. Asking human users to select the best replica is both inconvenient and inaccurate. Round-robin and other primitive DNS techniques spread load, but do little for network locality.

OASIS, however, can automate the process of selecting nearby and/or lightly loaded servers, yet it remains easy to integrate into existing applications.

Architecture

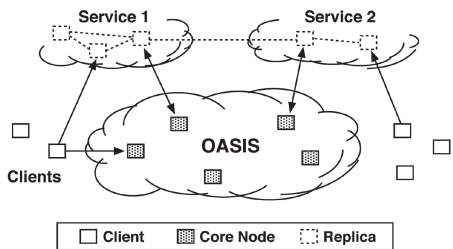


FIGURE 1. OASIS SYSTEM COMPONENTS

Figure 1 shows OASIS's two-tier architecture. The system consists of a network of *core* nodes that help *clients* select appropriate *replicas* of various services. All services employ the same core nodes (which we run as a public service); we intend this set of infrastructure nodes to be small enough and sufficiently reliable so that every core node can know most of the others. Replicas (which are deployed by service operators seeking to use OASIS for server selection) also run OASIS-specific code, both to report their own load and liveness information to the core and to assist the core with network measurements. Clients need not run any special code to use OASIS, because the core nodes provide DNS- and HTTP-based redirection services.

The primary function of the OASIS core is to return a suitable service replica to a server-selection request. Given that a request only provides the client's IP address and the service name (encoded in the domain name being resolved when using DNS), how does OASIS determine a client's location, which is needed to discover nearby replicas?

GEOLOCATING IP ADDRESSES

To discover the location of clients, OASIS probes Internet destinations using the replica servers as vantage points and, in doing so, finds the closest replica. One of OASIS's main contributions is a set of techniques that make it practical to measure the entire Internet in advance and therefore eliminate on-demand probing when clients make requests.

OASIS minimizes probing and reduces its susceptibility to network peculiarities by exploiting *geographic coordinates* as a basis for locality and by leveraging the locality of the IP prefixes [6] (e.g., NYU has IP prefix 216.165.0.0/17). We assume that every replica knows its own latitude and longitude, which already provides some information about locality before any network measurement. Then, in the background, OASIS uses service replicas as vantage points to probe each IP prefix to discover the replica with lowest round-trip-time (yet still does so in a manner that minimizes probing [13]). Finally, OASIS stores the geographic coordinates of the replica closest to each prefix it maps.

Because the physical location of IP prefixes rarely changes, an accurately pinpointed network can be safely reprobbed infrequently (as rarely as once a week). Additionally, this approach amortizes bandwidth costs across the multiple services using OASIS, resulting in an acceptable per-node cost that only decreases as more services adopt OASIS. Such infrequent, background probing both reduces the risk of abuse complaints and allows the system to respond quickly to requests, with no need for on-demand probing.

RESOLVING SERVER-SELECTION REQUESTS

What happens when a client makes a selection request to a core node? First, a core node maps the client's IP address to an IP prefix of appropriate granularity to capture locality properties. It then attempts to map the IP prefix to geographic coordinates. If successful, OASIS returns the closest service replicas to that location (unless load-balancing requires further consideration of load as a primary selection metric). Otherwise, if it cannot determine the client's location, it returns random service replicas.

This server-selection process relies on four databases maintained in a distributed manner by the core: (1) A *service table* lists all services using OASIS (and records policy information for each service), (2) a *bucketing table* maps IP addresses to prefixes, (3) a *proximity table* maps prefixes to coordinates, and (4) one *liveness table per service* includes all live replicas belonging to the service and their corresponding information (i.e., coordinates, load, and capacity).

How are these tables managed in a distributed manner? OASIS optimizes response time by heavily replicating most information. Service, bucketing, and proximity information need only be weakly consistent; stale information only affects system performance, not its correctness. Thus, OASIS uses gossiping to efficiently disseminate such state—as well as for failure notifications regarding core nodes—throughout the network.

Replica liveness information, however, must be fresher: DNS resolvers and Web browsers deal poorly with unavailable replicas, since such client applications cache stale addresses longer than appropriate. To tolerate replica failures robustly, replica information is maintained using soft-state: Replicas periodically send registration messages to core nodes (currently, every 60 seconds). This replica process also regularly connects to the local application seeking OASIS service to verify its liveness (i.e., every 15 seconds). These communications are shown in Figure 2.

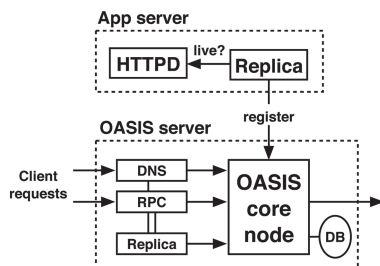


FIGURE 2: OASIS COMMUNICATION DIAGRAM

OASIS must know most replicas belonging to a service to answer corresponding selection requests. Therefore, OASIS aggregates replica liveness information for each particular service at a few core nodes known as *service rendezvous nodes*. To provide self-organizing properties within the core, different sets of core nodes are chosen via consistent hashing [9] to play the role of rendezvous nodes for each service.

Although all core nodes can map a client's IP address to geographic coordinates and determine the relevant service policy, when a core node receives a service request for which it does not play the role of rendezvous node, it must also send an RPC query to one of the requested service's rendezvous nodes. This rendezvous node uses its aggregated list of known replicas to determine the best-suited replicas for the client. In [5], I describe a variety of additional optimizations to reduce the load on a service's rendezvous nodes for increased scalability.

Evaluation

I now briefly present some wide-area measurements of OASIS on PlanetLab [10]. This section is meant simply to demonstrate that OASIS can greatly improve end-to-end latencies and load-balancing for replicated systems. For a full evaluation of OASIS and a complete explanation of the experiments, please see [5].

Figure 3 shows the end-to-end time for clients to download a Web page from a domain name being served by OASIS. This time includes a DNS

lookup and the subsequent TCP transfers. Using 250 PlanetLab hosts, we compare OASIS to a variety of other state-of-the-art and simplistic server-selection schemes, include using Meridian for on-demand probing [13], Vivaldi for virtual coordinates [3], and round-robin selection. The median response time for OASIS is 290% faster than Meridian and 500% faster than simple round-robin systems. These end-to-end measurements underscore OASIS's true performance benefit, coupling fast DNS response time (by using cached information) with accurate server selection.

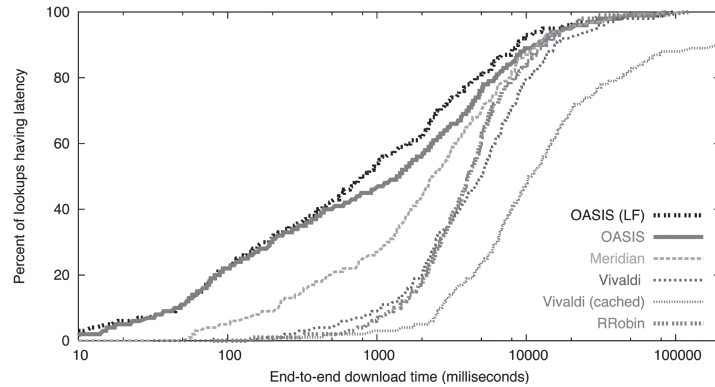


FIGURE 3: GRAPH (CDF) OF LOOKUP LATENCY VERSUS DOWNLOAD TIME

Table 1 shows how OASIS can reduce bandwidth costs associated with 95th-percentile billing. When multiple co-located clients (here, all in California) make requests against our four distributed Web servers, OASIS's load balancing ensures that 95% peak load remains evenly balanced. Purely locality-based selection, in contrast, yields a traffic spike at the nearest Web server.

Metric	California	Texas	New York	Germany
Latency	23.3	0.0	0.0	0.0
Load	9.0	11.3	9.6	9.2

TABLE 1: 95TH-PERCENTILE BANDWIDTH USAGE (IN MB PER MINUTE)

I next describe how services can adopt OASIS to enjoy similar performance benefits.

Using OASIS

Figure 4 shows various ways in which legacy clients and services can use OASIS to access a service. In our usage scenarios I use CoralCDN [4], an open content distribution network we have been running since early 2004. CoralCDN receives about 25 million requests daily from over 1 million clients; in fact, it motivated us to build OASIS in the first place to provide better proxy selection.

A CLIENT'S STEP-BY-STEP BEHAVIOR

The top diagram of Figure 4 shows how to make legacy clients select replicas using DNS redirection. The service provider advertises a domain name served by OASIS (e.g., coralcdn.nyuld.net). (OASIS currently uses the domain name .nyuld.net for its core nodes.) When a client looks up that domain name (Step 1), OASIS first redirects the client's resolver to a nearby

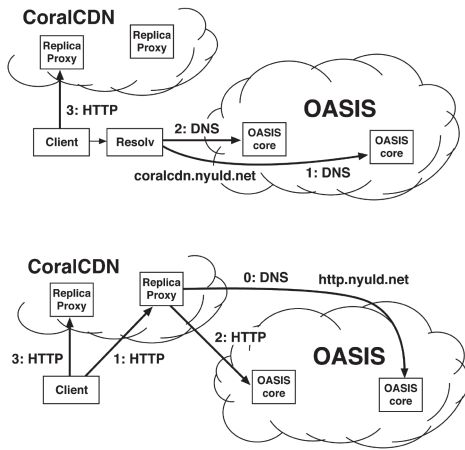


FIGURE 4: USES OF OASIS FOR ACCESSING A SERVER

OASIS nameserver (by resolving `dns.nyuld.net` with respect to the client's IP address and returning the results as NS records). The client's resolver caches these nameservers for future accesses. Then the resolver queries this nearby nameserver (Step 2) to determine the address of nearby, unloaded CoralCDN Web proxies (returned as the domain's A records). This approach can be accurate, provided that clients are near their resolvers.

The bottom diagram shows an alternative based on application-level HTTP redirection. Here, the CoralCDN replicas are also clients from OASIS's point of view. Each replica connects to a nearby OASIS core node that provides HTTP service, as selected by DNS redirection for `http.nyuld.net` (Step 0). When a client connects to a replica (Step 1), that replica queries OASIS to find a better replica (Step 2), now asking for service by explicitly specifying the client's IP address in an HTTP query string. Finally, an HTTP redirect is returned to the client, causing it to contact the selected replica for service (Step 3). Such an approach does not require that clients be located near their resolvers in order to achieve accurate locality.

In fact, OASIS supports several variations on this same theme: The CoralCDN replica can query the OASIS core using RPC, instead of HTTP. Alternatively, the replica's query can simply ask for the estimated distance between two IP addresses, which only uses the core's location database and does not require that it maintain a service-specific replica state (although the service itself would then need to maintain liveness information). Furthermore, the HTTP server on core nodes can perform HTTP redirection for clients themselves, avoiding the need for clients to contact the initial replica proxy (Step 1).

The rest of this section is devoted to the concrete steps a service operator needs to perform in order to integrate OASIS into their distributed system.

REGISTERING A SERVICE

A service policy must be registered with the core so that OASIS can handle its server selection. This policy currently includes a service's name (e.g., `coralcdn`), the number and expiration time of replica addresses returned per request, and the selection criteria. By default, OASIS selects replicas based on locality, unless the nearer replica's load exceeds its capacity. Other policies support pure locality-based selection or a load-balancing algorithm meant to reduce costs associated with 95th-percentile billing.

To enable sites to publish their own top-level domain names, OASIS supports aliases. Thus, in the context of CoralCDN, requests to `nyuld.net` will be interpreted as `coralcdn.nyuld.net` or, in the case of the OverCite service [12], `overcite.org` gets interpreted as `overcite.nyuld.net`. To support this aliasing, however, a server operator must also point the nameserver records for their top-level domain to some subset of OASIS's nameservers.

DEPLOYING REPLICAS AND INTEGRATING APPLICATIONS

On every host running a service application—such as a CoralCDN Web proxy—the service's administrator should deploy an OASIS replica. (The source code is released under the GPLv2 and is available from <http://oasis.coralcdn.org/>.) Service replicas should be configured with their geographic coordinates and, in order to monitor its liveness, the service name and listening port of their local application.

On the application side, the application (or some stand-alone daemon monitoring it) simply needs to listen on a TCP server socket on the config-

ured port. Then, when the local OASIS replica connects to the application (every 15 seconds by default), the application should simply accept the connection, respond with its application status (a shared secret code for verification, its current load, and its maximum capacity), and then close the connection.

We already run OASIS replicas on most PlanetLab hosts [10] as a public service to the PlanetLab community. Thus, system developers seeking to deploy their services on PlanetLab need only configure their application to respond to our local liveness checks and need not deploy replicas themselves, as a single OASIS replica can monitor multiple local services and their applications.

ACCESSING THE SERVER-SELECTION AND GEOLOCATION SERVICE

Once a service's policy and some of its replicas are registered with the OASIS core, core nodes can immediately respond to client server-selection requests. OASIS currently provides DNS, HTTP, and RPC interfaces for server selection, as shown in Figure 4, above. To access a CoralCDN Web proxy via DNS redirection, for example, a client need only connect to the hostname `coralcdn.nyuld.net`. To use HTTP redirection, the client simply accesses the URL `http://http.nyuld.net:8096/redirect.html?pol=coralcdn&ip=<ip>`, which causes the client first to discover a nearby core node running an HTTP proxy (via DNS), then to ask that HTTP proxy for a nearby CoralCDN replica. The optional query string `<ip>` performs the request with respect to that specified IP address, as opposed to the client's own IP address.

OASIS exposes additional information through its HTTP and RPC interfaces. For example, a client can query OASIS for the geographic coordinates of a particular IP address or the distance between any two such addresses: `http://http.nyuld.net:8096/distance.xml?src=<ip1>&dst=<ip2>`.

Output from the HTTP proxy can be either in HTML or XML, the latter allowing for simple integration with third-party Web services.

Conclusion

OASIS is a global, distributed, server-selection system that allows legacy clients to find nearby or unloaded replicas of distributed services. Two main features distinguish OASIS from prior systems. First, OASIS allows multiple application services to share the selection service. Second, OASIS avoids any on-demand probing when clients initiate requests by geolocating all IP prefixes in advance.

Publicly deployed since November 2005, OASIS has already been adopted by a number of distributed services [1, 2, 4, 7, 8, 11, 12]. Experimental measurements and third-party experiences suggest that OASIS produces highly accurate results, ensures server liveness, and provides simple system integration and client use. For more technical information on OASIS's design, evaluation, and integration, as well as relevant source code, please visit <http://oasis.coralcdn.org/>.

REFERENCES

- [1] F. Annexstein, K. Berman, S. Strunjas, and C. Yoshikawa, "Adaptive Client-Server Load Balancing Using Persistent Demands," Technical Report ECECS-TR-2006-06, University of Cincinnati, July 2006.

- [2] B.-G. Chun, P. Wu, H. Weatherspoon, and J. Kubiawicz, "ChunkCast: An Anycast Service for Large Content Distribution," *Proceedings of the 5th International Workshop on Peer-to-Peer Systems* (February 2006).
- [3] F. Dabek, R. Cox, F. Kaashoek, and R. Morris, "Vivaldi: A Decentralized Network Coordinate System," *Proceedings of SIGCOMM* (August 2004).
- [4] M. J. Freedman, E. Freudenthal, and D. Mazières, "Democratizing Content Publication with Coral," *Proceedings of the First Symposium on Networked Systems Design and Implementation* (March 2004).
- [5] M. J. Freedman, K. Lakshminarayanan, and D. Mazières, "OASIS: Anycast for Any Service," *Proceedings of the 3rd Symposium on Networked Systems Design and Implementation* (May 2006).
- [6] M. J. Freedman, M. Vutukuru, N. Feamster, and H. Balakrishnan, "Geographic Locality of IP Prefixes," *Proceedings of the Internet Measurement Conference* (October 2005).
- [7] R. Grimm, G. Lichtman, N. Michalakis, A. Elliston, A. Kravetz, J. Miller, and S. Raza, "Na Kika: Secure Service Execution and Composition in an Open Edge-side Computing Network," *Proceedings of the 3rd Symposium on Networked Systems Design and Implementation* (May 2006).
- [8] D. Joseph, J. Kannan, A. Kubota, K. Lakshminarayanan, I. Stoica, and K. Wehrle, "OCALA: An Architecture for Supporting Legacy Applications over Overlays," *Proceedings of the 3rd Symposium on Networked Systems Design and Implementation* (May 2006).
- [9] D. Karger, E. Lehman, F. Lighton, M. Levine, D. Lewin, and R. Panigrahy, "Consistent Hashing and Random Trees: Distributed Caching Protocols for Relieving Hot Spots on the World Wide Web," in *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing* (May 1997).
- [10] PlanetLab: <http://www.planet-lab.org/>.
- [11] S. Rhea, B. Godfrey, B. Karp, J. Kubiawicz, S. Ratnasamy, S. Shenker, I. Stoica, and H. Yu, "OpenDHT: A Public DHT Service and Its Uses," *Proceedings of SIGCOMM* (August 2005).
- [12] J. Stribling, J. Li, I. Councill, M. F. Kaashoek, and R. Morris, "Over-Cite: A Cooperative Digital Research Library," *Proceedings of the 3rd Symposium on Networked Systems Design and Implementation* (May 2006).
- [13] B. Wong, A. Slivkins, and E. G. Sirer, "Meridian: A Lightweight Network Location Service without Virtual Coordinates," *Proceedings of SIGCOMM* (August 2005).

RIK FARROW

WISPER: open source, long-distance wireless



rik@usenix.org

AS THE DEVELOPED WORLD BECOMES ever more connected, the developing world falls ever farther behind. Projects such as One Laptop per Child (OLPC) seek to bring low-cost technology to aid in education to all parts of the world, but you might have noticed that there is a problem with wireless laptops in, say, central Africa. Where is the equally low-cost networking infrastructure required to connect these laptops to the global Internet?

I had considered this question, even as I mused about the reality of a \$100 laptop (laptop.org). But it wasn't until Teus Hagen, Director of NLnet, caught up with me during USENIX Annual Tech '06 in Boston that I began to research the issue. Hagen had become infused with missionary zeal to create a new open source project that will, at the least, look into the prospects of creating software and related hardware platforms that will support low-cost networking.

Although the biggest focus on wireless technology has been for use in cities, wireless means that less densely populated areas might quickly become connected without large expenditures. And the research into adding free (or nearly free) wireless in large cities (e.g., projects such as RoofNet in Cambridge, MA) provide a giant step forward into creating actual software and hardware designs for expanding connectivity and communications into far-flung locations.

The Big Idea

Hagen has labeled his nascent project WISPER, an obvious play on the English word "whisper." Just like the low-powered radios used in Wi-Fi access, a network of low-powered devices that can be mass manufactured just might support the development of the communications infrastructures in much of the world. The OLPC project relies on the same economies of scale, that is, creation of a single design that can be mass-produced, bringing down the cost through economies of scale. The OLPC design does include Wi-Fi that will always be on (as long as the batteries provide power), holding out the possibility of a continuous, but purely local, network. But this type of network scales poorly. Clearly, something else is needed.

Hagen believes that the WISPER project could provide the next level of connectivity. While I can imagine that not all governments are interested in

unfettered Internet connectivity, the ability to move beyond a local-only information base to a global one seems a logical next step—not an easy step, but certainly one worth exploring.

RoofNet

Perhaps the closest research project to WISPER that I've found so far is the RoofNet research network [1]. RoofNet involved a small number of systems (37) set up in a relatively small (and affluent by world standards) part of Cambridge, MA. Students installed Dell PCs running the RoofNet software package in the homes or apartments of participants. Installation included, in most cases, running a coax cable up to an omnidirectional roof-mounted antenna. In an area of mostly three- to four-story buildings, these antennas provided good coverage and were much easier to set up than directional antennas. Most systems were less than half a kilometer away from other systems, although there were outliers over a kilometer away.

RoofNet connected to the Internet via five gateways. As with the selection of RoofNet participants, the locations of gateways were more random than planned, so as a research platform, RoofNet better resembled an organically grown, rather than a carefully laid-out, wireless network. RoofNet not only worked but provided a mean bandwidth of 627 kbs and a median of 400 kbs, values that are not bad for an unplanned network using 802.11b as its wireless base.

The big story in RoofNet was not the hardware, or even that it was done, but the research that came out of the project. Some findings were not surprising, given prior work. For example, as the number of routing hops in a mesh network increases, the available bandwidth decreases. If you consider that someone who is four hops away from a gateway must share the bandwidth of intermediate systems with an ever-increasing amount of traffic, this finding just makes sense. But it makes the notion of large, unplanned mesh networks sound unworkable if it means that the number of hops between a client and a gateway must be, say, four or fewer [2].

RoofNet simply avoided the problem of the decrease in bandwidth with the increase in hop count, because no node was more than five hops from a gateway. RoofNet did, however, successfully deal with related issues. For example, traditional routing is based on following the best calculated routing metrics. Routers typically exchange reachability and response-time data among themselves and use those metrics to choose the next router to forward packets to. In RoofNet, a hybrid system, ExOR, was used instead for large file transfers.

All RoofNet Web traffic went through proxies, and these proxies would use ExOR for transferring the first 90% of large files and traditional routing for the last 10%. And using ExOR resulted in increases in transfer rates from 40 to 300%, even over single-hop routes. The key to ExOR was that it broadcast collections of packets, rather than unicasting the packets to the next hop, and included in this broadcast a manifest of the packets sent [3]. This approach takes advantage of the vagaries of Wi-Fi, where some packets might skip what would in traditional routing be the first or even later forwarders, and thus reach the destination system more quickly. ExOR also increases throughput in single hops, because it does not use RTS/CTS and has a different method for handling retransmissions of lost or damaged packets.

RoofNet also used Click [4] for assignment of addresses. Each PC was assigned an address based on the lower 24 bits of the wireless card's MAC address, with the higher 8 bits of the IP address remaining fixed. But each PC could also act as a router with NAT, so that locally connected systems

would be given an address using DHCP. Because local systems were assigned private network addresses, participants had access to the Internet, but not to each other's systems.

The success of RoofNet is a relevant one, as it used innovative techniques to get beyond some of the limitations found in 802.11 hardware and standard IP software. But RoofNet itself is not a model for WISPER, where there might be many hops between a client and an Internet gateway. RoofNet was an experimental testbed.

Out of RoofNet

RoofNet has already produced a spin-off, a new company called Meraki Networks (meraki.net), which has already designed and produced the Meraki Mini, a \$100 device designed for relaying Wi-Fi traffic. The Mini includes an Atheros chip that supports 802.11b/g Wi-Fi, an Ethernet port, a USB port, built-in encryption support, and a MIPS processor that runs a 2.4 Linux kernel. With the addition of some flash memory and SDRAM, the Mini is a self-contained system (just requiring a power supply) that can serve as a development platform for more mesh-style networking research.

I talked to John Bicket, the CTO and co-founder of Meraki Networks, and one of the students who was involved in the RoofNet research, about the Mini. Although the Mini sounds like the perfect hardware platform for WISPER, the match is actually far from perfect. Mini is designed to be used indoors, for example, making it a poor candidate for the environments found in developing countries. Meraki has aimed the Mini at commercial really low-cost wireless deployment in a building, such as an apartment building. Bicket said that the Mini is also designed with use by researchers in mind, as full documentation and sample device drivers are available. At \$100 per box, the Mini costs just 10% of what each RoofNet node had cost, and less than many less capable devices used in mesh-network research.

There have been other network devices, such as the Meshcube (meshcube.org), intended for use in urban mesh networks, but these devices cost considerably more than the Mini. And none of the existing devices, or software, really satisfies Teus Hagen's list of goals.

WISPER

I must confess that it took me a while to understand what Hagen wanted from a WISPER project. At first, it seemed like all the parts were already in place—but they really are not, at least not yet. And Hagen has higher ambitions than a deployment in places where a RoofNet is just an alternative to several other means of having Internet connectivity. WISPER is really intended for places where the Internet has not yet penetrated—and not just physical locations.

Hagen wants WISPER to be a project not only for developing wireless networking but also for advancing IPv6. The very addressing issues that RoofNet neatly dodges need to be attacked head-on, Hagen believes, and a project like WISPER might be just the ticket. IPv6 was developed to deal with the shortcomings of IPv4, including the limited address space. But large-scale deployments of IPv6 remain scarce. And unplanned deployments, such as an ad hoc arrangement of wireless access points in a type of a mesh, really require a new way of thinking about addressing.

Hagen's complete list of goals for WISPER includes:

- IP address space solution, IPV6

- Multi-mesh scaling and answers to throughput issues
- Stability
- Operating systems other than 2.4 Linux (OpenWRT), such as 2.6, BSD, or even Minix 3
- Configuration and simplicity
- Autodynamic configuration
- Wi-Fi long distance: a cheaper WIMAX
- Standard hardware
- Open source code, to allow cooperative development
- Free access and free availability
- VoIP integration (Asterisk)
- Misuse measurements
- Security aspects (node and user authentication)
- Privacy (existing mesh solutions offer no privacy)
- A mobile mesh to allow Wi-Fi VoIP as an alternative to cellular phones

Hagen's wish list is certainly optimistic. But some items, such as auto-configuration, are a must for a device that will be installed and used by a very nontechnical population. Others, such as VoIP, may at first seem whimsical, but for parts of the world with little or no communications infrastructure and high illiteracy rates, the capacity to support voice communications will actually be very important.

Hagen turns out to be ideally placed for supporting an optimistic and open-ended project such as WISPER. Hagen is the Director of NLnet, an organization that came about when early Internet infrastructure in The Netherlands was sold to a commercial company. NLnet can invest millions of U.S. dollars per year in launching network projects, projects designed to become part of the public domain and be totally open source.

At one point in our many email exchanges, Hagen suggested that I liken WISPER to the old UUCP mail exchange networks. These networks were very ad hoc, relying on people using their own phone lines and systems to relay other people's email messages, files, and USENET postings. The Internet itself sprang from this very primitive dial-up network. In the same way, Hagen hopes that WISPER will help spawn the next level of global connectivity.

If WISPER excites you and you want to participate (or perhaps just comment on what may appear to you to be a "wild-assed scheme"), you can visit the Wiki at <http://www.nlnet.nl/wifi>.

REFERENCES

- [1] Papers about the MIT CSAIL RoofNet research project: <http://pdos.csail.mit.edu/roofnet/doku.php?id=publications>.
- [2] Some comments about mesh networks, including a geometric decrease in bandwidth as hop counts increase: <http://www.oreillynet.com/pub/a/wireless/2004/01/22/wirelessmesh.html>.
- [3] Sanjit Biswas and Robert Morris, "Opportunistic Routing in Multi-Hop Wireless Networks": <http://pdos.csail.mit.edu/papers/roofnet:exor-sigcomm05/>.
- [4] Eddie Kohler, Robert Morris, and Massimiliano Poletto, "Modular Components for Network Address Translation": <http://pdos.csail.mit.edu/papers/click-rewriter/>.

See also "Wireless Networking in the Developing World," with handy how-to info on boosting a Wi-Fi device: <http://www.wndw.net/>.

DAVID BLANK-EDELMAN

practical Perl tools: tie me up, tie me down (part 2)



David N. Blank-Edelman is the Director of Technology at the Northeastern University College of Computer and Information Science and the author of the book *Perl for System Administration* (O'Reilly, 2000). He has spent the past 20 years as a system/network administrator in large multi-platform environments, including Brandeis University, Cambridge Technology Group, and the MIT Media Laboratory. He was the program chair of the LISA '05 conference and is one of the LISA '06 Invited Talks co-chairs.

dnb@ccs.neu.edu

WHEN WE LEFT OFF IN THE PREVIOUS column, I was standing over an anonymous hash holding a whip. OK, maybe not, but it did get you to check your back issues of `;login;`, no? Actually, we left off with something much more titillating: the ability to modify the fundamental nature of how Perl variables work using modules based on Perl's `tie()` functionality.

At the end of the last column we had just begun to contemplate the following list of things we wished a Perl hash could do:

- have elements that would automatically expire after a certain amount of time had elapsed
- keep a history of all changes made to it over time
- restrict the updates that are possible
- always keep track of the top *N* values or the rank of the values stored in it
- always return the keys in a sorted order based on the values in that hash
- transparently encrypt and decrypt itself
- easily store and retrieve multiple values per key

Let's take some time to make those wishes (and more that you didn't even know you had) come true, and then we'll end by discussing how to create our own `tie()`-based code.

Expiring Hashes

Hashes with entries that disappear after a certain time period are easy to construct, thanks to `Tie::Hash::Expire`:

```
use Tie::Hash::Expire;
tie my %hash, 'Tie::Hash::Expire', { 'expire_ seconds' => 5};
$hash{'magician'} = 'Erik Weis';
$hash{'musicians'} = ['Jalil', 'Ecstasy', 'Grandmaster Dee'];
$hash{'software'} = 'Side Effects Software';

print scalar keys %hash; # prints '3'
# do something for 6 seconds...
print scalar keys %hash; # prints '0'
```

An interesting twist on this module is `Tie::Hash::Cannabinol`, which describes itself as a "Perl extension for creating hashes that forget things." Specifically, the doc says:

Once a hash has been tied to `Tie::Hash::Cannabinol`, there is a 25% chance that it will forget anything that you tell it immediately and a further 25% chance

that it won't be able to retrieve any information you ask it for. Any information that it does return will be pulled at random from its keys.

Oh, and the return value from `exists` isn't to be trusted, either.

To get back on a slightly more even keel, it should be mentioned that `Tie::Scalar::Timeout` or `Tie::Scalar::Decay` can do similar expiration magic for scalar variables.

Hashes with a Sense of History

There are two modules that give a hash variable the ability to remember all changes made over time. Usually when you set a value for a key in a hash, that value replaces any previous value with no record of there ever having been a previous value. With `Tie::History` or `Tie::HashHistory`, magic takes place in the background, making it possible to access previous values. For example, let's assume we were tracking the price of corned beef. We could write code that looked like this:

```
use Tie::History;

my $hobj = tie my %historyhash, 'Tie::History';

$historyhash{'cornbeef'} = 1.28;
$hobj->commit;

$historyhash{'cornbeef'} = 1.35;
$hobj->commit;

$historyhash{'cornbeef'} = 1.25;
$hobj->commit;
```

The initial `tie()` line creates a special hash called `%historyhash` and then returns an object through which the history methods for that hash are controlled. Values for the key `cornbeef` are set using the standard notation. Each time we want to remember the state of the hash, we use the control object (`$hobj`) to commit it. At this point in the execution of our program, if we call the `getall()` method for the hash control object, we'd see:

```
DB<1> x $hobj->getall
0 ARRAY(0x50bcac)
0 HASH(0x5248cc)
  'cornbeef' => 1.28
1 HASH(0x53e624)
  'cornbeef' => 1.35
2 HASH(0x53e660)
  'cornbeef' => 1.25
```

There are other methods such as `previous()`, `current()`, and `get()` that allow you to retrieve the state of `%historyhash` at a specific point in the running history. `Revert()` will actually revert the hash to a specified position in the history. `Tie::History` is pretty spiffy, because it also works for scalars and arrays.

To keep things moving along, I won't show you an example for `Tie::HashHistory`, but I do want to mention one way that it differs from `Tie::History`. `Tie::HashHistory` is meant to augment other `tie()` modules and give them history superpowers. For example, if you were using the `Tie::DBI` module talked about in the last column, it might be handy for debugging purposes to keep a running history of all changes made to a `Tie::DBI`'d hash. `Tie::HashHistory` makes that easy.

Restrictive Hashes

Hashes are fabulous data structures. This notion of a collection of information in key/value pairs is both very powerful and very easy to use. But hashes can be a bit of a pushover. They are happy to store anything you throw at them, even mistakes. For example, a hash has no way to know that the key in:

```
$authors{'Charles Dickens'} = "Hard Times";
```

is a typo (after all, it could be the well-known Dutch author). Just as the `use strict pragma` offers a good way to avoid variable name typos, the module `Tie::StrictHash` will do the same for hash keys:

```
use Tie::StrictHash;
my $hobj = tie my %authors, 'Tie::StrictHash';
```

Now we have two things: a hash (called `%authors`), which will behave in an unusual way, and a hash control object (`$hobj`) that will be used to make changes to that object. `%authors` is now unusual because, as the documentation says:

- No new keys may be added to the hash except through the `add` method of the hash control object.
- No keys may be deleted except through the `delete` method of the hash control object.
- The hash cannot be reinitialized (cleared) except through the `clear` method of the hash control object.
- Attempting to retrieve the value for a key that doesn't exist is a fatal error.
- Attempting to store a value for a key that doesn't exist is a fatal error.

So if we wanted to add a new key to the hash, we would call:

```
$hobj->add('Charles Dickens' => 'Hard Times');
```

Once in place, existing keys are changed just as one would expect:

```
$authors{'Charles Dickens'} = 'Great Expectations';
```

In the interests of full disclosure (since this isn't `tie()`-related), Perl 5.8.x versions implement something known as “restricted hashes” that have very similar properties. Using the `Hash::Util` module that ships with Perl, it is possible to lock down a hash or even individual keys in a hash. You don't get all of the `Tie::StrictHash` functionality or its very clear semantics (i.e., method calls for making changes), but it doesn't require installing a separate module.

Automatic Ranking

It's fairly common to write code that reads in a set of values and then has to report back the rank of each value in the whole list. `Tie::Hash::Rank` makes it easy to determine where a particular value stands in the ranking by constructing magical hashes that return a rank for each key stored in them instead of the associated value. For example:

```
use Tie::Hash::Rank;

tie my %rhash, 'Tie::Hash::Rank';

# grams of sugar
%rhash = (
    'countchocula' => 12,
    'booberry'    => 15,
```



```

        'trix'      => 13,
        'cheerios' => 1,
    );

    print $rhash{'countchocula'}, "\n"; # prints 3
    print $rhash{'booberry'},      "\n"; # prints 1
    print $rhash{'trix'},          "\n"; # prints 2
    print $rhash{'cheerios'},      "\n"; # prints 4

```

There are other similar modules that make it easy to keep track of the top *N* values in the hash (e.g., `Tie::CacheHash`).

Automatic Sorting

There are a few modules that handle keeping a hash's elements in a sorted order; these include `Tie::Hash::Sorted` and `Tie::IxHash`. If you find yourself repeatedly sorting and resorting your hash keys before processing, this can be a big win (especially when dealing with large data sets).

In a related vein, `Tie::Array::Sorted` helps you maintain an array whose elements stay (or appear to stay) sorted even in the face of element additions and deletions. I say “appear” because there is also a `Tie::Array::Sorted::Lazy` module in the package that is smart enough to only re-sort the array when its contents are retrieved. This works well for cases where you have an array that will be modified quite a bit before being read.

Encrypted Hashes and Multi-Valued Storage

We're getting close to the end of our wish list, so let's take a quick look at the last two items on the list so we can move on to creating our own `tie()`-based code. The first is the ability to transparently encrypt and decrypt data in a hash. `Tie::EncryptedHash` is an excellent module for this purpose if it fits your program's model. `Tie::EncryptedHash` is good for those cases where you want to create a collection of information that needs to be encrypted when not in active use.

A hash tied using this module can contain both normal key/value pairs and “encrypting fields.” Encrypting fields are those key/value pairs that begin with an underscore (e.g., `$hash{'_name'}`, `$hash{'_socsecur'}`, etc.). The hash itself is kept either in transparent/unencrypted or opaque/encrypted mode. The mode designates whether the encrypting fields found in that hash are encrypted or not.

To read or modify the encrypting fields in the hash, you unlock it with a special `__password` key; deleting this password will lock it again. In locked mode, you can safely drop the contents of the hash to disk or copy it over a network without fear of those fields being disclosed (the normal key/value pairs will continue to stay in plaintext). This is really simple in practice:

```

    use Tie::EncryptedHash;
    use Data::Dumper;

    tie my %eh, 'Tie::EncryptedHash';

    $eh{'normal'} = 'no magic here';

    # let's unlock the hash

    $eh{'__password'} = 'supersecretsquirrel';

    # and store an encrypting field
    $eh{'_encrypting'} = 'now you see it ...';

```

```

print "transparent: " . Dumper( \%eh ) . "\n";

# lock the hash
delete $eh{ '__password' };

print "opaque: " . Dumper( \%eh );

```

The output of this program is:

```

transparent: $VAR1 = {
    'normal' => 'no magic here',
    '_encrypting' => 'now you see it ...'
};
opaque: $VAR1 = {
    'normal' => 'no magic here',
    '_encrypting' => 'Blowfish FHt07w3l/xyfd1/c4hskvQ
53616c7465645f5f4e8203d51070213d75fdf19b4c26b13435bc375600c49f
27b07e21be89f631df'
};

```

The last item on the wish list is one that makes a programmer's life easier. `Tie::Hash::MultiValue` is helpful in those cases where you want to store multiple values in a single hash key. The standard way to handle this situation is to store a reference to an anonymous array for that hash key (the Hash of Lists idea). `Tie::Hash::MultiValue` actually does this, but it makes the process a little easier. For example, instead of having to write something like this:

```

$mvh{ 'mike' } = [qw(greg peter bobby)];

# add a new element to the list, not the most pleasant syntax
push(@{$mvh{ 'mike' }}, 'tiger');

```

you can write:

```

use Tie::Hash::MultiValue;

tie my %mvh, 'Tie::Hash::MultiValue';

$mvh{ 'mike' } = "greg";
$mvh{ 'mike' } = "peter";
$mvh{ 'mike' } = "bobby";
$mvh{ 'mike' } = "tiger";

```

to get the same result. The module will also make sure that only unique values are stored, so that:

```

$mvh{ 'mike' } = "alice";
$mvh{ 'mike' } = "alice";

```

only stores "alice" once in the anonymous array associated with the key "mike." (A quick warning: The doc for `Tie::Hash::MultiValue` says that it is possible to assign multiple values at a time. This unfortunately does not work in the current version available on CPAN.)

That's the last of the items on our wish list, but there are still many impressive `tie()`-based modules available on CPAN that we could talk about. I could continue to blather on about modules such as `Tie::File` (which reads and writes lines in a file as if it were an array), `Tie::HashVivify` (in which you call your own subroutine every time you attempt to read a key in a hash that doesn't exist), or `Tie::RemoteVar` (which implements a client/server model that allows you to share the same variables between programs running on different machines).

Instead, let's move on to creating our own `tie()`-based code.

Don't Do It

As I mentioned in the first part of this series, there are a number of valid objections to writing `tie()`-based code. They include a couple of concerns:

- Performance: Tied variables can be quite slow compared to other approaches, because of all of the overhead.
- Maintainability: Without seeing the `tie()` call, other programmers can't know whether a tied variable will go "oogah-boogah" every time it is accessed, instead of exhibiting the usual variable behavior.

Both of these are perfectly reasonable concerns, so let me quickly state the alternative to `tie()`-based code: Write your code using standard OOP practices, create objects instead of `tie()`'d variables, and call the methods of those objects explicitly. Instead of:

```
$special{'key'} = 'value'; # prints "oogah-boogah"
```

use something like:

```
$special->oogahboogah('key','value')
```

instead. Although this is not the most glamorous alternative, it does help with both performance and maintainability.

No, Really. Tie() Me Up, Tie() Me Down

If you've determined you do want to write code for `tie()` there are a few ways to go about it. In the interests of space and time, I'm going to show you only one way, using a very simple example that makes SNMP (Simple Network Management Protocol) queries using hash semantics. If you are not familiar with how SNMP works or the `Net::SNMP` module (perhaps a future column topic), the short version is that it is a protocol for querying management information from a device (e.g., a router).

The standard way to construct a `tie()`-based module requires a bit of Perl OOP knowledge. If you don't have that knowledge or just want something quick and dirty you can use the `Tie::Simple` module to hide the details for you. These details are found in the `_perltie_` manual page (`perldoc perltie`) and are the subject of Chapter 9 in Damian Conway's *Object Oriented Perl*.

Here's how the code you are about to see works. To create `tie()` code, you build an OOP-based package. This package creates objects with methods that implement all of the standard variable operations (`fetch`, `store`, `exists`, `delete`, etc.) required of that variable type. Code for `tie()`-ing scalars needs to contain 4 subroutines to cover all of the operations; for hashes the number is 9, and for arrays it goes to 13. To avoid having to write all that code for this example, we're going to inherit a set of default subroutines from `Tie::StdHash` module in the `Tie::Hash` package. These subroutines mimic the standard hash behavior, leaving us free to redefine just the operations that suit our purpose. In the example that follows, we redefine only the `TIEHASH` operation, called when the `tie()` function is executed, and `FETCH`, called when a key is looked up in a hash.

Here's the code, with explanation to follow:

```
package SNMHash;
require Tie::Hash;
use Net::SNMP;

@ISA = (Tie::StdHash);

sub TIEHASH {
    my ( $class, $arghash ) = @_;
```

```

# create the object
my $self = {};
bless $self, $class;

# create an SNMP session and store it in the object
my ( $session, $error ) = Net::SNMP->session( %{$sarghash} );
die "Could not establish SNMP session: $error" unless defined $session;
$self{'session'} = $session;

return $self; # return the object
}

sub FETCH {
    my $self = shift;
    my $key = shift;

    # do the actual SNMP lookup
    my $result = $self{'session'}->get_request( -varbindlist => [$key] );
    return $result->{$key};
}

1; # to allow for loading as a module

```

Here's a very brief tour of the code: We start by declaring the name of the package (which will become the class of the object created). After the usual loading of modules we declare that we'll be inheriting from the `Tie::StdHash` module. This gives us the freedom to redefine two operations, `TIEHASH` and `FETCH`.

For `TIEHASH`, we create an empty object, initialize an SNMP session object based on the arguments in the `tie()` statement, and store a reference to this session in the object for later use. That use happens in the very next subroutine when we define what should happen upon key lookup. In this case, we take the key, turn it into a standard SNMP `_get_request`, and then return the answer. When this happens it looks like the `tie()`'d hash has magical keys consisting of SNMP variables (in OID form), which can be queried to see the live data.

How does this get used?

```

# assumes we saved the previous example in a file called SNMPHash.pm
# someplace Perl can find it
use "SNMPHash";
tie my %snmhash, 'SNMPHash',
    { '-hostname' => 'router', '-community' => 'public' };
# this long string of numbers is just a way of referencing (in SNMP
# OID form) the SNMP variable that holds the description for a system
# (i.e. sysDescr.0). See Elizabeth Zwicky's article at
# http://www.usenix.org/publications/login/1998-12/snmp.html or
# another SNMP tutorial for more info
print $snmhash{'1.3.6.1.2.1.1.1.0'}, "\n";

```

This yields something like:

```

Cisco Internetwork Operating System Software
IOS (tm) s72033_rp Software (s72033_rp-PK9S-M), Version 12.2(18)SXD1,
RELEASE SOFTWARE (fc1)
Technical Support: http://www.cisco.com/techsupport
Copyright (c) 1986-2004 by Cisco Systems, Inc.
Compiled Wed

```

There's a ton of things missing from this sample code; it is just meant to be a snippet to start your motor running. There's virtually no error checking. Most of the hash operations aren't implemented. (It isn't exactly clear just

how all of the SNMP operations should map onto hash operations. Some are obvious, but the others deserve some head scratching.) Still, you've now received a taste of what it takes to write your own tie()-based code. And with that, we need to wrap up this issue's column. Take care, and I'll see you next time.

Save the Date!

www.usenix.org/nsdi07

NSDI '07

**4th USENIX Symposium on Networked
Systems Design & Implementation**
April 11–13, 2007 Cambridge, MA

Join us in Cambridge, MA, April 11–13, 2007, for NSDI '07, which will focus on the design principles of large-scale networks and distributed systems. Join researchers from across the networking and systems community—including computer networking, distributed systems, and operating systems—in fostering cross-disciplinary approaches and addressing shared research challenges.

Sponsored by USENIX in cooperation with ACM SIGCOMM and ACM SIGOPS

USENIX

ROBERT HASKINS

ISPadmin: anti-spam roundup



Robert Haskins has been a UNIX system administrator since graduating from the University of Maine with a B.A. in computer science. Robert is employed by Shentel, a fast-growing network services provider based in Edinburg, Virginia. He is lead author of *Slamming Spam: A Guide for System Administrators* (Addison-Wesley, 2004).

haskins@usenix.org

IN THIS EDITION OF ISPADMIN, I TAKE a look at a few relatively current events in the area of anti-spam. Although not directly service-provider related, spam is certainly an area that is near and dear to the network service provider's operations.

The areas that I look at in this column include:

- Reputation
- Image spam
- SpamHINTS
- Sophos "top 12"
- Dlink SECURESPOT

Reputation

The area of reputation as it applies to anti-spam continues to mature. Although "reputation" can mean many different things to different people, I use "reputation" here to mean the likelihood a particular IP has emitted spam in the past. A simple reputation service would be a DNS blacklist service such as Spamhaus SBL [1] or DSBL [2]. However, these systems are simple "on" or "off" reputation systems; the systems described below assign a likelihood-of-spam probability in much the same way a personal credit scoring system assigns probabilities.

OPEN SOURCE SOFTWARE

Unfortunately, there has been little progress in the open source arena on anti-spam reputation-based systems. GOSSIP [3] appears to be at a standstill. I guess this means that a viable reputation service needs to have a commercial entity behind it (Trend Micro, Symantec, etc.). However, similar open source-like data-sharing anti-spam schemes currently exist in the form of the Distributed Checksum Clearinghouse (DCC) [4] and Vipul's Razor (written by Vipul Prakash, a founder of Cloudmark). Perhaps some would consider DCC commercial in the sense that it is backed by the commercial entity Rhyolite Software, but as far as I am aware, it doesn't generate revenue directly from running the DCC service.

COMMERCIAL SOLUTIONS

In the commercial arena, the IP reputation-based solutions keep on coming. One of the newer ones is from Simplicita Software [5]. The Simplicita Reputation Knowledge Server (RKS) can take IP

feeds from many different sources (MTA logs, firewalls, DNS-based blacklists, etc.) and allow network operators to block inbound messages from IPs determined to be bad. The RKS solution allows easy manipulation of these lists, adding and expiring IPs using many different schemes.

Another approach to the IP reputation is the Symantec 8100 series appliance [6], formerly known as Turntide. (Disclaimer: My employer is a Symantec customer and user of the 8100 and SBAS products mentioned here.) In the most common implementations, you simply put the device “in line” in front of your mail transfer agents (MTAs), behind either switch(es) or router(s). This solution works at the TCP level to block SMTP connections from IPs that have sent spam messages to your email infrastructure. The most egregious spamming IPs are “ratcheted down” in the rate at which they can send messages to the inbound or outbound MTAs sitting behind the 8100 appliance.

The good news about the 8100 device is that deploying them can significantly reduce the need for additional MTAs in your email architecture and they can reduce the number of spam messages hitting your MTA. However, the appliance isn’t a perfect solution, as it does not eliminate the need for additional filtering capability, requiring an MTA-level anti-spam filter such as Symantec Brightmail AntiSpam [7].

The other bit of bad news is that the appliance often blocks legitimate email from hosted domain forwarders, resulting in complaints from customers who have email forwarded to local accounts that come from such domain email forwarders. Of course, the 8100 device allows the administrator to whitelist IP address space, but the problem is where to draw the line. If you whitelist too many senders, then there isn’t much point in having the device to begin with.

The 8100 device does allow placement of IPs into a number of “classifications,” ranging from essentially unlimited access to a very slow rate of accepting SMTP connections. One solution is to “lock” the domain-hosting IP into a specific classification, not allowing unrestricted access, but slowing it down somewhat. It’s not a perfect solution, but it’s better than the alternatives.

One resource I have found to be particularly useful in managing the 8100 devices is Ironport’s Senderbase [8]. When one needs to whitelist an email service provider and the provider is unwilling or unable to tell you what their outbound servers are, the Senderbase data can give you a good idea of what IPs to start with. Although not perfect, it gives one a place to start.

Image Spam

One of the newer spamming techniques is the use of embedded images in email messages. This is a particularly difficult method of spam to deal with. (I personally have been receiving image spam for at least two years now.) Some commercial anti-spam vendors recommend not allowing images as a solution to this problem. This isn’t very practical, as this would only work for those that don’t regularly receive images. I would guess that isn’t very many users. Barracuda Networks’ [9] solution is to perform optical character recognition on image attachments [10]. This sounds interesting, but I wonder how well it works. Pretty soon, we’ll be seeing watermarks in spam images to throw off the automated detection of spam via OCR!

SpamHINTS

SpamHINTS [11] is a research project by Richard Clayton at the University of Cambridge. His approach is to look at network traffic patterns and glean spamming IPs from the changes in patterns over time. This strategy will be very interesting if it works. The problem I see with this approach is that if the spammers can make their traffic look just like legitimate traffic, then there won't be anything to find. I suspect that a traffic-based approach will work for the high-volume spammers who send their junk via a relatively small number of IP addresses, but detecting small volumes spread out over widely disparate networks will prove difficult to identify using this method.

Sophos "Top 12"

I cringe every time I see a lot of press coverage of the "biggest spammer" lists such as that recently published by Sophos [12]. It's not that I have anything against such lists; but I have reservations about how accurate the data really is and what it really means. But first, does it really matter that North America allegedly originates 23.1% of spam? I think that the SpamCop stats [13], which show spammers by net blocks, are much more useful. With per-network spam information, we (Internet users) can put pressure on the egregious spammers with the SpamCop information. Back on the geographic lists, do we really care what country (or continent, for that matter) originates the most spam? What are we going to do, complain to George Bush because the United States originates the most spam?

Regarding the accuracy of the data, I suspect that the geolocation data has gotten better over time, but I still wonder how accurate it really is, with VPNs, inexpensive bandwidth, and all the other little details that make geolocation difficult.

Dlink SECURESPOT

The Dlink SECURESPOT [14] takes the security appliance to the extreme, for the consumer (SOHO) market. In a box half the size of a deck of cards, the device performs a whole host of security-related functions, including:

- Parental control
- Pop-up blocking
- Virus protection
- Spam blocking
- Spyware protection
- Identity protection
- Firewall protection
- Network reporting

SECURESPOT was designed by Bsecure Technologies [15], who handle the "service" side of the product, updating firmware and maintaining the lists of "bad guys" to block as part of the solution. Like any new product, it will take time for the support issues to be straightened out. I would imagine that this product would take some tweaking to work within the existing user's PC environment, given software firewalls, anti-spam, anti-virus, and other security-related software already running on the PC.

Of course, it remains to be seen how well this device works when compared to software equivalents such as Symantec's Norton Internet Security

[16]. The best design would be to integrate this functionality into the existing SOHO firewall/router and not have an extra box, but I suppose you have to start somewhere. I suspect that if this product takes off, Dlink will probably integrate the SECURESPOT functionality into some of their other products.

REFERENCES

- [1] Spamhaus SBL: <http://www.spamhaus.org/sbl/index.lasso>.
- [2] DSBL: <http://dsbl.org/main>.
- [3] GOSSIP: <http://gossip-project.sourceforge.net/>.
- [4] DCC: <http://www.rhyolite.com/anti-spam/dcc/>.
- [5] Simplicita: <http://www.simplicita.com/>.
- [6] Symantec Mail Security 8100:
<http://www.symantec.com/Products/enterprise?c=prodinfo&refId=852>.
- [7] Symantec Brightmail AntiSpam:
<http://www.symantec.com/Products/enterprise?c=prodinfo&refId=835&cid=1008>.
- [8] Ironport's database of sending IPs: <http://www.senderbase.org/>.
- [9] Barracuda Networks: <http://www.barracudanetworks.com>.
- [10] <http://www.networkworld.com/news/2006/071906-barracuda.html?fsrc=rss-security>.
- [11] SpamHINTS: <http://www.spamhints.org/>.
- [12] Sophos top 12: <http://www.sophos.com/pressoffice/news/articles/2006/04/dirtydozapr06.html>.
- [13] SpamCop stats: <http://www.spamcop.net/spamstats.shtml>.
- [14] Dlink SECURESPOT:
<http://www.dlink.com/products/?sec=0&pid=486>.
- [15] Bsecure Technologies: <http://www.bsecure.com/>.
- [16] Symantec Norton Internet Security:
http://www.symantec.com/home_homeoffice/products/overview.jsp?pcid=is&pvid=nis2006.

HEISON CHAK

echo in VoIP systems



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

heison@chak.ca

HELLO, HELLO, HELLO, HELLO, . . .

Many of us have had experience with echo, whether it was speaking loudly in an empty stadium, shouting in the great outdoors in front of a mountain, hearing your own voice while talking on a telephone, or even singing Karaoke with the echo feature enabled on a mixer. It all boils down to the persistence of a sound after its source has stopped.

Although there are two main types of echo, acoustic and electronic, both can be generalized as reflection points being introduced that cause the audible echo.

In the case of an acoustic echo, sound waves get bounced against objects. Since these objects may have very different physical characteristics and can be varying distances from the source of the sound, the resulting sound arrives at the speaker's ears at different times and amplitudes. It is important to understand that echo may only be heard by the speaker. Consider a speaker standing at one end of a tunnel or an empty stadium, shouting out to his listener. Only the speaker will hear his own voice being reflected by the surroundings. Similarly, if a climber is shouting out to her friend on top of a mountain, the friend may be able to hear the message clearly with no echo. Yet the climber may hear her own voice being reflected by the mountain and surrounding reflection objects even after she has stopped shouting.

Electronic echo reflection points found in telephony systems are generally related to analog-to-digital conversation, 2-4-wire hybrid, and impedance mismatch. In a telephony system, when a caller experiences echo, it's likely that the reflection point is at the remote end of the conversation. In fact, the remote party of the conversation may find the communication sounding perfectly fine, simply because he or she is beyond or behind the reflection point, much like the mountaineer yelling from the bottom of a mountain to her friend at the top. This is commonly known as the far-end problem.

Echo in VoIP Systems

When sound is applied to the handset transmitter of a VoIP device, it is digitized, processed, encoded, and transmitted via the network interface as IP packets. The near-end telephone intentionally sends some of the electrical signal from the transmitter to the receiver. These signal components, known as side-tone, are used to simulate the natural expectation of a user to hear his or her own voice while speaking. Since side-tone originates locally, it carries only a small amount of delay, too short to be perceived by a user.

In a pure VoIP system (i.e., an IP-IP call), there is no chance of echo being introduced until the packets arrive at the far-end telephone. Electrical crosstalk in the far-end telephone can occur and couple some of the received signal in its transmit path, leading to leakage of the speaker's voice in the return path. The handset on the far-end telephone can generate an acoustic output that is coupled back to the transmitter as well. The combined electrical and acoustic signals get digitized, processed, encoded, and transmitted to the near-end phone, where they appear at the receiver. Since these packets carry much longer delay than those generated locally on the near-end devices, they become a more noticeable side-tone with a much longer delay—in other words, echo.

The perceived quality of the connection can often be impacted by the amount of delay these signals carry as well as by their amplitude. Typically, when the signal delay exceeds 15–20 ms, the user perceives the delay, and the experience becomes more unpleasant if the delay has a high amplitude. Fortunately, analog telephone systems have low enough latency that signals are generally not perceived as echo.

VoIP packets are known to have low resilience to delay and latency; the effect is even more undesirable when it comes to echo. Besides typical transmission delay of IP networks, VoIP packetization intervals (framing duration of received audio for transmission) and jitter buffers can contribute further delay.

VoIP and POTS

In traditional analog telephone systems, whenever a 2–4-wire hybrid is used, some receive signals will leak back into the transit path, owing to the imperfection of coupling. Similar coupling issues can be seen if the user's telephone impedance mismatches that of the phone service provider. Because of the short delay characteristics of local calls on the PSTN (Public Switched Telephone Network), signals created by such 2–4-wire hybrid reflection generally are not perceived as echo. For long distance and international calls, telephone service providers feature echo cancellers to take care of the added transmission delay. An echo canceller is a software algorithm that tries to remove the portion of the signal caused by the transmitter picking up receive output of the telephone; it does so by attenuating delay samples in uncompressed form (e.g., 12-bit PCM). Generally, echo cancellers are disabled for local calling, as they are simply redundant.

When VoIP systems are communicating with POTS phones, echo can originate from a number of places:

- VoIP to local numbers: The PSTN provider has its echo canceller disabled, as it assumes that local calls have short latency; it has not factored in the possible added delay inherited from an IP network. (A VoIP user may experience echo, mainly electrical.)
- Poor echo canceller on a gateway: An inexpensive media gateway usually will not feature a hardware echo canceller; echo cancellation performed in software may not converge fast enough to cope with the ever-changing characteristics of a call, especially on a busy system. (A VoIP user may experience echo, mainly electrical.)
- Underpowered echo canceller: With dedicated hardware, if an echo canceller has a relatively short echo cancellation capacity, it will not be able to remove echo effectively. (A VoIP user may experience echo, mainly electrical.) A typical T1/PRI echo canceller is capable of removing undesired signal up to 128 ms across all channels.
- Impedance mismatch on a gateway: The FXO (Foreign Exchange Office) interface on a media gateway may not couple with the PSTN, so

the voice of the POTS user gets reflected into his or her return path. (The POTS user may experience echo, mainly electrical.)

- Poorly designed hybrid on a gateway: The quality of each 2–4-wire hybrid that comprises a telephone call tends to correlate with the amount of echo a user perceives. Good circuit design can ensure better coupling, minimizing reflection points. This can be compensated for by an echo canceller, if applied correctly. (POTS and VoIP users may experience echo, mainly electrical.)
- Hands-free telephones: These tend to have higher output amplitude on the speaker and higher sensitivity, because a user is usually sitting further away from the phone than with a handset telephone. As a result, the acoustic coupling of the speaker to the microphone is high and can result in echo. With the added latency of VoIP, echo from hands-free telephones can become quite strong. (POTS and VoIP users may experience echo, mainly acoustic.)
- PC to PC: During a pure IP-IP call, when packets arrive, electrical and acoustic echo may be introduced. If a PC-to-PC call involves a speaker and a microphone that have the sensitivity characteristics just described, echo becomes inevitable. The same applies to PC-to-PSTN calls. (VoIP and POTS users may experience echo, mainly acoustic.)

Both acoustic and electrical echo can be compensated for by proper application of echo cancellation. If an echo canceller is applied too far from the source, the undesired signal may outrun the capability of the canceller (e.g., a canceller with 128 ms of capacity will not remove a delayed copy of someone's voice of 400 ms). In the case of a call between a VoIP device (either a PC or an IP handset) and the PSTN, if the call is identified by the PSTN as a local call, the echo canceller may be disabled, making the call vulnerable to the sorts of echo problems just described.

VoIP applications are often challenged by echo problems, impacting call quality. It is important to remember that echo is usually a far-end problem. If echo cannot be avoided, placing an echo canceller closest to the source can ensure more effective compensation, giving faster convergence and less DSP-intensive operations.

Although echo is often viewed as a far-end problem, this may not always be the case. Consider a telephone call where one end is on a mobile phone. The conversation starts out fine but deteriorates as soon as a generic headset is used on the mobile phone (in which case only the mobile user experiences echo). As soon as the generic headset is swapped with one that is specifically designed for the phone, the echo problem diminishes. In this case, no changes are made to the system except that a near-end device has been replaced. It is possible that there is impedance mismatch between the generic headset and the mobile phone, which suggests a near-end problem.

It is difficult to convince the user of a poorly designed telephone that his phone is coupling the signal it receives and sending the delayed signal back to the other party on a well-designed phone. In effect, the user of the well-designed telephone will hear a delayed copy of her own voice, while the conversation sounds just fine to the user of the poorly designed telephone. The situation can be misleading. The user of the poorly designed telephone believes that his telephone is working well, whereas the user of the well-designed telephone may wonder whether her telephone set is the source of echo, when just the opposite is true.

REFERENCE

http://microtronix.ca/echo_problems.htm.

ROBERT G. FERRELL

/dev/random



Robert is a semiretired hacker with literary and musical pretensions who lives on a small ranch in the Texas Hill Country with his wife, five high-maintenance cats, and a studio full of drums and guitars.

rgferrell@greatambience.com

WE ALL GET BEES IN OUR BONNETS

from time to time, and one little critter that's been buzzing around in mine for quite a spell now is the gradual disappearance of the concept of personal responsibility. Growing up in West Texas, when you made a mistake you stared down at the ground and scraped your toe in the sand in embarrassment for a few seconds, wishing that teleportation weren't just a cheesy sci-fi effect, then straightened your back, squared your shoulders, and took the consequences. Sure, we occasionally produced stink beetles who tried to blame missing that easy pop fly to center field on the sun being in their eyes or getting distracted by one of those balsawood glider-sized dragonflies endemic to the area, but by and large folks in my neck of the cactus plantation had a pretty firm grasp on societal cause and effect. OK, I think I've got all the bugs out of my system now. Did I mention how much I hate big red wasps?

The fundamental idea that *you*, not some corporation/neighbor/government agency/supernatural influence/alignment of fantastically distant celestial objects, are responsible for what happens to you has apparently joined the Ivory-billed Woodpecker and the woefully misnamed "common sense" on the critically endangered list. I suppose such quaint and outdated mores simply don't resonate with today's society, working as they do in opposition to the generation of new SUVs and beach properties for the legal profession.

Some examples of total abdication of personal responsibility are obvious, such as the recent spate of well-publicized lawsuits over fumble-fingers who scald themselves with beverages, consumers who take that term a bit too literally and wolf down anything and everything put in front of them, blaming the food vendors when this indiscriminate consumption results in arteries with the structural characteristics of uncooked pasta, and parents who think the rest of the world should be held accountable for their own inability to control what their children read and watch on a daily basis. Others are more insidious: every computer sitting at the terminus of an always-on broadband Internet link, for example.

Imagine that you buy a new car and leave it, unlocked and running, in your driveway 24 hours a day. Imagine further that some lowlife slithers up, steals said car from said driveway, and employs it in the commission of a crime. Do you share any of the blame for this event? You didn't actively participate in or condone the action itself, so why would you? Is providing easy access to a potent tool acting as a sort of unwitting accessory to the crime? The legal arguments surrounding this scenario are more complex than they may first appear, and let me state now for the record that I Am Not A Lawyer (I prefer to sleep at night), but fundamentally this boils down to a question of personal responsibility, however oblique that may seem to the case as stated.

Items deemed by expert consensus to be potentially dangerous to health or safety are ideally accessible only by those who bear the brunt of responsibility for their use. This includes weapons, prescription drugs, vehicles, theatrical adaptations of classic comic books, and . . . computers. That's right: computers. A potential for economic and sociological mayhem, if not actual public safety concerns, resides right there in that oversized shoebox sitting on edge under your desk. Whether or not you admit it, if you don't take some fairly simple steps to keep out the riffraff it is almost a dead certainty that your computer will be recruited into a multicellular evil entity whose metabolic functions are anything but beneficial to the society on which they ravenously feed. I liken botnets, as these infernal networks are called, to digital cancers. They spread one cell at a time throughout a logical system until most or all of the CPU cycles of that system are devoted not to their original benign purpose (I'm being charitable here, I realize) but to the propagation of the infection and the bidding of their demonic overlord(s). Or mayhap your box will be hijacked to serve as an anonymizing launch pad for malicious activity such as identity theft or reading celebrities' text messages to their dog groomers. Either way, that PC you rely upon for delivering your spam and checking your online sports memorabilia auctions will become just one more hapless cog in a larger, sinister enterprise—I mean, in addition to already being part of the Internet.

So, you may well ask, what does that have to do with personal responsibility? The answer, my friend, is simple in the end: everything. Ultimately, you are responsible for what gets done using your hardware. It's *your* computer, in *your* house, on a network *you* pay to access. If you don't understand how to keep it secure because you're *nontechnical*, fine: Outsource that job to someone you trust. Simply throwing up your hands and claiming exemption because you don't understand the issues or technology involved does not wash. Do you understand the myriad mechanical, chemical, and physical processes that take place when you drive your car? Probably not. Does that make you any less liable when you plow through a crowd drinking coffee at a sidewalk café? No. It is implied by your assumption of the role of pilot of a motor vehicle that you have received basic instruction in the safe operation thereof.

So it should be with the increasing potential hazard that is the Internet. Millions of people all around the globe depend on you to do your small part to minimize the spread of malware. It doesn't take much—a few mouse clicks in most instances and a basic policy of not opening attachments unless you're absolutely certain of their origin and contents—to prevent 99% of all commonly encountered exploits. As with automobiles, the benefits of learning to operate the machine safely exceed the effort required by several orders of magnitude. The ROI, in other words, be phat. The penalties for irresponsible computing so far have been diffuse and

applied too far up the chain to do much good. The effort has to come from end users like us to be successful. Put another way:

Take it to the house y'all, lock it or lose it.
Don't leave it in the open 'cause the thugs'll abuse it.
Close it off, shut 'em down, make 'em find another kill'n.
The botz take you down unless you get physical.

Here, then, is my Contract with the Digerati: If I don't see a marked decrease in DDoS and other botnet-related activity in the near future, I'll pen more of these, uh, lyrics. I don't think you want that, now, do you? I knew you didn't. I could see it in your eyes.

But, you protest, I'm an inherently superior UNIX user and savvy enough to secure my own systems. Cleverly, I'm counting on that. It leaves you free to devote yourself to helping those less technically blessed to secure theirs. Now get to work, slackers. Don't make me come over there. Dword?



Copyright 2006 R. Moon

book reviews



ELIZABETH ZWICKY,
ERIC SORENSON, AND
SAM STOVER

THE .NET DEVELOPER'S GUIDE TO WINDOWS SECURITY

Keith Brown

Addison-Wesley, 2004, 392 pages.
ISBN 0-321-22835-9.

OK, so this is not an obvious review choice for me. I'm not a developer, I don't do .NET, and I don't do much Windows—which leaves "Security" as the only relevant word in the title. To be honest, I didn't request the book, for no very obvious reason the publisher sent it along with books I did request, and I picked it up mostly because I thought "Hey, it's short and irrelevant; I can glance at it, determine that I don't care, and move on rapidly."

So I picked it up and read a random page, which to my surprise was lucid and interesting. So I read the whole thing; and then I read bits of it aloud to my colleagues, who said things like, "Hey! Can I borrow that?" even though they aren't .NET developers either.

If you happen to know any .NET developers, you should run out, buy this book, and force them to read it. Then force them to reread the bits on running as a normal user and developing software that system administrators don't hate. If you are a system administrator, you should definitely read those bits yourself,

just so you can have the warm fuzzy feeling that out there somewhere is a developer who understands.

And if you need to know how security-relevant parts of Windows actually work, buy this book even if you have no interest at all in .NET, because it's full of clear explanations and practical tips. I was awe-struck by the explanation of why group nesting works the way it does. I mean, I suppose I had always assumed there was some reason, but nobody had ever mentioned one, let alone diagrammed it out. And now I have been enlightened.

LEARNING WINDOWS SERVER 2003

Jonathan Hassell

O'Reilly, 2006. 723 pages.
ISBN 0-596-10123-6.

THE ULTIMATE WINDOWS SERVER 2003 SYSTEM ADMINISTRATOR'S GUIDE

Robert Williams and Mark Walla

Addison-Wesley, 2003. 956 pages.
ISBN 0-201-79106-4.

As I said above, I don't do Windows much. In fact, the Windows 2003 machines I administer are entirely virtual (users and all), and if something goes wrong, I cheerily destroy them and start another nice clean one that a real Windows administrator built an image for. Thus, I'm reviewing these books from the point of view of a UNIX administrator with a need to understand things periodically, not of somebody who lives and breathes Windows system administration.

However, I have had reason to consult these books recently, and there's a clear pattern. *Learning Windows Server 2003* has the information I need, without a lot of other stuff, and with handy information about command-line tools. *The Ultimate Windows Server 2003 System Administra-*

tor's Guide has more deployment information, but it's less clearly written and often obfuscates rather than clarifying important details. For instance, it tells you that users can read files even when they don't have permission to read the folder the files are in, but it doesn't mention that this is in fact a user right that can be revoked. It has lots of illustrations of the dialogs used to set up groups, and of particular group configurations (many of which it tells you not to use), but it doesn't have a table that says what kind of users and groups can be members of each kind of group. *Learning Windows Server 2003* has such a table. The .NET book also has such a table, plus an explanation of why the groups work the way they do and what the pros and cons of each kind of group are.

Here's how each book introduces groups:

The .NET Developers Guide to Windows Security: "Most developers have a basic idea of what a security group in Windows is all about. It's a way to simplify administration by grouping users together. In a large system, a clearly defined group can allow an administrator to assign permissions for hundreds of files, registry keys, directory objects, and so on, without having to think about each individual user that will be in the group to which he or she is granting permission."

Learning Windows Server 2003: "The point of groups is to make assigning attributes to larger sets of users easier on administrators. Picture a directory with 2,500 users. You create a new file share and need to give certain employees permission to that file share (e.g., all accounting users)."

The Ultimate Windows Server 2003 System Administrator's Guide: "A group is a collection of

users, computers, and other entities. It can be a Windows Server 2003 built-in group or one created by the system administrator to conform to required attributes. Windows Server uses standard groups to reflect common attributes and tasks.”

As you can see, the books have very different styles. I recommend *Learning Windows Server 2003* as a general reference, and *The .NET Developers Guide to Windows Security* for understanding underpinnings.

THE BEST SOFTWARE WRITING I

Joel Spolsky, Editor

Apress, 2005. 305 pages.
ISBN 1-59059-500-9.

This is a collection of essays, some of them brilliant, some of them thought-provoking, some of them laugh-out-loud funny, and (from my point of view) one clunker (no, I won't tell you which one). Most of them are available on the Web already, although if you've found them all you read way too many blogs.

The canonical reader here is a senior programmer in a startup, but anybody who likes to think about programming, programming languages, or the computer industry will find something to chew on and something to laugh at. Or maybe weep at—I'm honestly not sure whether the appropriate response to “Powerpoint Remix” is to laugh or to cry.

If you're looking for a great airplane book, this is quick but meaningful stuff. I enjoyed it vastly.

EXTRUSION DETECTION: SECURITY MONITORING FOR INTERNAL INTRUSIONS

Richard Bejtlich

Addison-Wesley Professional, 2005.
ISBN 0-321-35996-2.

REVIEWED BY ERIC SORENSON

Extrusion Detection is, in sum, a refocusing of the methodology

Bejtlich detailed in *The Tao of Network Security Monitoring*. Here he shifts the emphasis from threats that generate traffic out on the Internet to ones that are already inside an enterprise's perimeter and must be detected by inspecting outbound traffic. Since *Extrusion Detection* ought to be useful by itself without requiring *The Tao of NSM*, Bejtlich necessarily repeats some material but fortunately provides enough new information to make for some worthwhile reading, even for experienced NSM practitioners.

He begins by introducing general network security and intrusion detection principles along with the NSM credo: “Prevention eventually fails.” The goal of network security monitoring is to provide a framework for answering the dreaded question, “We've been compromised—What now?” As such, the bulk of Part I of *Extrusion Detection* describes specific measures an administrator can put in place to produce good answers: blocking and proxying outbound traffic, placing packet capture sensors at choke-points, and using router features such as null routes and reverse path forwarding. With these in place, the administrator will run a “defensible network,” that is, one that gives a reasonable chance of dealing with an intruder.

In Parts II and III, Bejtlich walks through what “dealing with an intruder” might entail, first as a procedural framework (Part II) and then with a specific extrusion in the form of an unauthorized botnet (Part III). Unfortunately, here the book's greatest strength is offset by its biggest weakness. The sections that detail specific steps and principles to observe during incident response and when gathering network forensics data could be powerfully useful references for

one of the primary target audiences of the book—savvy enterprise network administrators who have not had much opportunity to do formal incident response. But instead of presenting the information in capsule, checklist form and then providing detail, there is lengthiness—in some cases multiple-page—output from commands and sample data between steps, which makes it difficult to get a good overview of the entire process. In an emergency, the book would be frustrating if not outright dangerous to thumb through as you're responding. (The author's three-page description of his difficulty reading a mangled pcap file in the Network Forensics chapter is a prime example of this problem.)

However, this should not detract from the overall thumbs-up I give this book; a prepared admin team will have customized the book's suggested incident response procedure for their environment and summarized it (on paper!) before anything actionable happens—right? There is plenty of useful information in *Extrusion Detection*: the comparisons of SPAN monitor ports versus network taps, the routing and filtering tricks, and the walkthrough that shows the discovery of a botnet are worth the price of admission. The exploits that lead to a high-profile Web server's defacement get most of the press, but Richard Bejtlich's *Extrusion Detection* describes a methodology for dealing with threats that are potentially far more damaging, because they move from the inside out.

BUFFER OVERFLOW ATTACKS

James C. Foster, Vitaly Osipov,
Nish Bhalla, and Niels Heinen

Syngress, 2005. ISBN 1-932266-67-4.

REVIEWED BY SAM STOVER

When I first started amassing my “Foster Library,” I was pretty excited. I couldn’t wait to find the time to really sink my teeth into the guts of buffer overflows and exploit code. I still have that desire to learn, but as I start plowing through the books, I’m becoming more and more disappointed with the library. The book entitled *Buffer Overflow Attacks* (BOA) was written in 2004 and provided the foundation for a more recent book I’ve reviewed previously called *Writing Exploits and Security Tools* (WSTaE). Well, it’s not just the foundation, but the house, garage, yard, trees in the yard, birds in the trees, etc., etc.

I think saying that the overlap between these two books borders on the criminal is not an overstatement. I think saying that WSTaE is an updated version of BOA is misleading. True, some of the grammar and wording has been changed, such as the shift from “commonest” to “the most common.” Some of the chapter

titles have changed: “Buffer Overflows: The Essentials” has become “Writing Exploits and Security Tools,” and “Stack Overflows” is now “Exploits: Stack.”

Other than that, not much has changed. WSTaE actually does have more content with chapters devoted to Metasploit, Nessus, and Ethereal, but the core of the book is so “cut and paste” that over half of WSTaE is completely redundant. It would have made more sense to omit the overlap and release an update to BOA—but at a drastically reduced price.

There are a couple of gems in BOA that aren’t in WSTaE, but I’m not convinced that they are worth the \$35. There are three sections dedicated to case studies, which walk through 11 exploits as well as providing an introduction to Inline Egg (which is discussed in much greater detail in WSTaE).

This commonality puts me in a rather awkward position—there isn’t much to review that I haven’t already discussed in a previous review. The crux of the matter, then, is to help you the reader and potential purchaser to make a decision as to which

book fits your needs. If you’ve already purchased BOA, then you’re bound to be disappointed when you have to throw down another \$50 for WSTaE, which does have a lot more information, notably the chapters on Metasploit. However, if you already own the *Penetration Tester’s Open Source Toolkit* (Syngress, 2005), then you have two of the three chapters on Metasploit. Sheesh. If you already own WSTaE, you’ll definitely want to think carefully before ordering BOA—at the very least cruise over to your local bookstore and see if the case studies are really worth your hard-earned dollars. If you don’t own any of these books yet, and are looking for a first purchase, don’t waste your time with BOA, go directly to WSTaE.

In conclusion, I have to say that *Buffer Overflow Attacks* was a pretty big disappointment. The incestuous relationship between Mr. Foster’s books leaves me with a sour taste in my mouth. Writing exploits is a hot item right now, and ripping content from one book seems a direct attempt to exploit the unwary novice (pun intended).

NICHOLAS M.
STOUGHTON
AND ANDREW JOSEY

USENIX standards activities



Nick is the USENIX Standards Liaison and represents the Association in the POSIX, ISO C, and LSB working groups. He is the ISO organizational representative to the Austin Group, a member of INCITS committees J11 and CT22, and the Specification Authority subgroup leader for the LSB.

nick@usenix.org



Andrew Josey is the director of Certification within The Open Group and chairs the Austin Group, the working group responsible for development of the joint revision to POSIX and the Single UNIX Specification.

a.josey@opengroup.org

An Update on Standards: Diary of a Standard Geek

NICK STOUGHTON

Have you ever wondered just what happens at a standards committee, what it is like to attend an international meeting to determine the future of a standard that affects millions of your fellow workers? Here are some notes from a trip to Berlin in April 2006 for the ISO-C committee (officially known as ISO/IEC JTC 1 SC 22/WG 14, it's charged with c99, a.k.a. ISO-C, the language accepted by most modern compilers).

Saturday: I get on a plane to go to Germany for an ISO-C meeting (a three-leg flight from Oakland, through Dallas and Zurich), arriving in Berlin sometime tomorrow. I have 14 hours to reread all of the documents submitted for the meeting (11 papers and 2 draft documents). Well, that took up one hour . . . just another 13 to go.

Monday: The meeting starts at 9:30 at the DIN headquarters. Is this the place they invented DIN plugs? Well, yes. But standards shouldn't be about invention (though in the case of electrical connectors, they often are). We start gathering at about 8:45—there's plenty of coffee and pastries on offer in the meeting room (thanks to SAP). There are 26 of us, representing five national bodies. All participation in ISO meetings is by national body (otherwise known as a “country”). In the case of ISO-C, it is also jointly developed by ANSI (the U.S. national body), and so the U.S. contingent is 20 of the 26. (The astute mathematicians will figure out that the delegations from the other countries are modest in comparison.)

Every working group has its own methods for achieving consensus; in ISO-C we try as hard as possi-

ble to avoid formal votes. We regularly will stop proceedings to hold straw votes. These are taken to get a sense of the room on an issue. They are nonbinding, but they often stop us from going down ratholes where it is clear that there's only one or two people who believe in a given direction.

However, meetings are formal. They have a definite pattern, with an agenda to work through and, often, time limits. Low-level working groups, such as ISO-C or the Austin Group (where POSIX gets written and maintained) are driven by technical matters. We will spend an hour arguing over whether or not an optimizer is allowed to reorder certain memory accesses for performance, especially if the program happens to be multi-threaded (and this particular question won't go away any time soon). But this meeting, as are most all standards meetings, is more for direction setting. The “real” work of such a working group happens in the papers that are developed between meetings.

Monday morning is spent working through administrivia, liaison reports (including two from me, one from POSIX, and one from the FSG on LSB status), and potential defect reports. One of the other style issues that differs from working group to working group is how defect reports are handled. In the ISO-C case, defect reports can be raised either through a national body (e.g., the UK can submit a defect report directly) or by individuals who submit them to the chair, and they then get considered during this “potential defect” agenda slot. At this point, we simply have to agree whether or not they are defects, and if so, they get added to the list of defects we will work on later in the week. This time, I get to submit a small handful

on behalf of the Austin Group, where a conflict between C and POSIX appears to be present.

Monday lunch has me off to a local hotel for lunch provided by SAP. This is a rare treat . . . we usually don't get lunch provided for us!

Monday afternoon: Two papers are to be considered, one on "Managed Strings" and one (coming from the C++ committee) to add, to the floating point handling, macros to handle a maximum number of significant decimal digits. Although the concepts in the "Managed Strings" paper were interesting, they were all invention, with little existing practice, and overlapped with a paper I'm developing on I/O functions that use dynamic memory. The decimal digits work could be passed off to a subgroup who are writing a Technical Report on Decimal Floating Point (to align with the new revision of IEEE 754).

After coffee, it is time to talk about the "Security" TR (Technical Report). You may recall I have written about this in previous columns. At least it's now called "the bounds checking TR." It is looking like it may be ready for its next ballot at this point, and it appears to me now to be mostly harmless! However, members of the Austin Group had had serious concerns over it, and I presented a paper from them on these concerns. The author agreed to write a response to this paper. This may need a few discussions in the bar tonight!

Tuesday: It's time to talk about the decimal floating point document. This meeting is pleasant, because there is little political controversy, and for the majority of the meeting I can simply concentrate on the technical aspects of what we are doing. Now, if this had been SC 22, the parent

committee in ISO, the tone would be very different, and the discussions in the bar would have a very different feel to them!

Floating point is not one of my favorite subjects, so I spend much of the time while this is being discussed preparing for the Defect Report work we will be starting this afternoon.

In fact, it seems that most of us in the room feel like this . . . there are only three people talking, and everyone else is frantically clicking the keyboards on their laptops. Maybe they are reading their email. But you can't stop listening to the subject matter. Maybe there will be something that matters to POSIX, or to the LSB. No . . . that was too much to hope for!

Defect Reports take up much of the week. We start on Tuesday and will finish sometime Thursday. We work through the log of defects (which has just gotten longer because of the potential defects we turned into actual defects yesterday), usually starting from the top and going to the end. Those old, old defects at the top of the list have been looked at many times; we just can't find the right answer. Sometimes we will assign a small group to go off and consider a response (if there is anyone prepared to serve on such a breakout group). Or we'll give homework to one individual to write a response. If you want to see what a defect report looks like, look at <http://www.open-std.org/jtc1/sc22/wg14/www/docs/summary.htm>.

Wednesday: More defect reports are discussed. You'd think a language as well established as C wouldn't have a lot of defects in the specification. But it is hard to write a specification, and one that is heavily used will always have many issues in it. Not all of them are bugs in the standard.

Often it is a misunderstanding on the part of the submitter, or a question that is outside the scope of the standard, or any of a host of other issues.

"If an incomplete array type has elements of unknown size, should the incomplete array type be a VLA type?"

"Must bit fields of type char nevertheless have the same signedness as ordinary objects of type char?"

"What if `asctime()` is called with a `tm` structure whose `tm_year` field results in a year > 9999?"

"The first sentence of 6.7.5.2p2 seems to suggest that any ordinary identifier can have both block scope and function prototype scope and no linkage has a variably modified type. This is clearly wrong."

And so on . . .

Thursday: Wow! It looks like we might actually finish the agenda early for the week! We are done with defects. We just have the final reports from the defect review and the closing business to get through! Where's the next meeting? Portland, Oregon. What about the one after that? We have an invitation from the U.K. Then there's the action item review: who has to do what? by when? I have to help write the response to the Austin Group's concerns on the bounds checking TR (or at least ensure that the response is delivered). And I have to write the dynamic memory I/O functions report. My work is cut out for the next meeting. I can finally enjoy a couple of hours in Berlin before my flight home this evening.

Why Get Involved in POSIX?

ANDREW JOSEY

One of the key factors leading to success for Linux and the UNIX system has been the adoption of popular, open standards such as the X Window System, TCP/IP, and the POSIX standards. Today we see a rapid evolution of IT systems and applications brought about by the adoption of the Internet and the changes that has brought to the way we work. But are the standards evolving fast enough to keep pace with the changes? This article gives a high-level look at the current POSIX standardization activity, how it works, and how you can contribute to helping it keep pace.

What is POSIX? POSIX, a registered trademark of the IEEE, is an acronym for “Portable Operating System Interfaces.” The name POSIX was suggested by Richard Stallman in 1986. The most well known POSIX standard is IEEE Std. 1003.1 (or ISO Std. 9945, which is the same document), known for short as “POSIX.1.” It specifies application programming interfaces (APIs) at the source level and is about source code portability. It is neither a code implementation nor an operating system, but a standard definition of a programming interface that those systems supporting the specification guarantee to provide to the application programmer. Both Operating System Vendors (OSVs) and Independent Software Vendors (ISVs) have implemented software that conforms to this standard.

The major sections of POSIX.1 are definitions, utilities (such as `awk`, `grep`, `ps`, and `vi`), headers (such as `unistd.h`, `sys/select.h`, and other C headers), threads, networking, real time, interna-

tionalization, and math functions. In total, the standard describes over 1,350 interfaces.

If POSIX.1 is mentioned as a requirement for your software project, that does not tell you much. POSIX.1 is large (3,600 pages) and no project needs everything (even OSVs rarely implement every optional interface). The POSIX.1 standard is structured into modules known as option groups. A minimal set of interfaces and functionality is required for all POSIX systems. The majority of all functionality is optional. For a good description of options and their status in Linux and glibc, see <http://people.redhat.com/~drepper/posix-option-groups.html>.

There are several common misconceptions about POSIX. Since its development started in the mid 1980s, one common misconception is that it has not changed for some time; it is outdated and irrelevant. The latest version of the POSIX.1 standard was produced in 2001 and updated in 2004: it is known as IEEE Std. 1003.1, 2004 Edition. Work is now underway to revise the standard to produce a new revision in 2008. Although the new versions of the standard are in general upwardly compatible with the original, many hundreds of interfaces that have been added since then. Your participation is needed to help keep it up to date and to keep pace with the developments in the industry.

Another common misconception is that you need to be an IEEE member to participate. The latest edition was developed by the Austin Group, an open working group found at <http://www.opengroup.org/austin/>. Participation is free and open to all interested parties (you just need to join the mailing list). Decisions are made by consensus;

sometimes consensus is reached easily, and sometimes only after heated discussion! The more people involved in such discussions, the more likely it is that when consensus is reached, the right decision has been made. That's why your participation is so important. Readers should note, however, that the mailing lists are not a technical support forum. All the major UNIX players and open source distributions are represented in the Austin Group.

Today the approach to the POSIX standard development is one of “write once, adopt everywhere,” with a single open technical working group and the resulting documents being adopted by IEEE as the POSIX standard, adopted by The Open Group as the Base Specifications of the Single UNIX Specification, and by the International Standards Organization as an international standard (which in turn means that it may be a national standard in your country; for example, the British Standards Institute has adopted ISO 9945 as a BS).

So does this mean that the POSIX.1 standard is complete and perfect? No. Like any large product, it has bugs, and there is an ongoing bug reporting and fixing process to refine the document as implementation experience grows. Although the standard cannot change overnight, there is a mechanism both to make regular technical corrections and also to collect items for future directions. To report a bug or suggest a change, please use the defect report form at <http://www.opengroup.org/austin/defectform.html>.

Is POSIX still relevant? Yes. Standard interfaces mean easier porting of applications. The POSIX interfaces are widely implemented and referenced in other standardization efforts, includ-

ing the Single UNIX Specification and the Linux Standard Base.

Why should you get involved? Feeding back issues with the standard based on implementation experience allows the standard to be improved and extended with new functionality, which in turn can “raise the bar

of commonality” among systems. There is often much more to be gained by having key functionality share a common interface and/or behave in exactly the same way, than for it to be different.

More information on POSIX.1 and the Austin Group, including how to join and participate, is

available from its Web site at <http://www.opengroup.org/austin/>.

The html version of the standard is freely available from The Open Group’s Single UNIX Specification Web site at <http://www.unix.org/version3/>.

Renew Your USENIX and SAGE Memberships Online Today!

Renewing your USENIX and SAGE memberships has never been easier. Visit <http://www.usenix.org/membership> and click on the appropriate links.

In addition to your subscription to *login*, your USENIX benefits include:

- Online access to all Conference Proceedings from 1993 to the present
- Substantial discounts on technical sessions registration fees for all USENIX-sponsored events
- Jobs Board and Resume posting
- And more . . .

Are you a SAGE member too? SAGE membership benefits include:

- Short Topics booklets in both online and paper versions
- Discount on registration for LISA, the annual Large Installation System Administration conference
- Jobs Board and Resume posting
- Access to sage-members mailing list and the archives
- And more . . .

Don’t miss out on the benefits that help you keep up with the latest technologies, network with your peers, and find out about new job opportunities.

Renew both memberships today! <http://www.usenix.org/membership>

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.

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

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

USENIX BOARD OF DIRECTORS

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

PRESIDENT

Michael B. Jones,
mike@usenix.org

VICE PRESIDENT

Clem Cole,
clem@usenix.org

SECRETARY

Alva Couch,
alva@usenix.org

TREASURER

Theodore Ts'o,
ted@usenix.org

DIRECTORS

Matt Blaze,
matt@usenix.org

Rémy Evard,
remy@usenix.org

Niels Provos,
niels@usenix.org

Margo Seltzer,
margo@usenix.org

EXECUTIVE DIRECTOR

Ellie Young,
ellie@usenix.org

USACO NEWS

ROB KOLSTAD

USACO Head Coach

The USA Computing Olympiad continues to lead the world in training programs. As of this writing, 52,178 students have registered for training, including 8,900 U.S. students. The students solve training tasks and vie for top dog status in our monthly contests, which generally attract 900–1,000 participants from as many as 80 countries.

These fierce competitors battled through six contests, spread over most of the school year, to earn the coveted invitation to Colorado Springs, where they spent eight days on the campus of Colorado College. The 2006 USA Invitational Computing Olympiad (USAICO) garnered competitors from Russia, Poland, Romania (2), China, Canada (4), and, of course, the good old U.S. of A. (12).

Along the way, wunderkind Matt McCutchen, a senior from Rockville, MD, was the top U.S. performer in three monthly contests and the U.S. Open. He was crowned USACO National Champion. Zeyuan Zhu from China earned the USACO World Champion plaque for his outstanding sustained performance as an international competitor.

The USAICO comprises five traditional contests (3–4 challenging algorithmic problems over 3–5 hours) and the exciting Speed Round, where scores for the 15 serially presented tasks are updated in real time and contestants can unseat winners with each new programming task.

In the end, the U.S.A.'s John Pardon and China's Zeyuan Zhu fought to a tie for the gold medal. Silver medals went to Matt McCutchen and Canada's Richard Peng. Bulgaria's Rostislav Rumenov won the bronze medal.

The top four U.S. students were invited to participate in the IOI (International Olympiad on Informatics, the holy grail of pre-college programming

competition) in Mexico: Juniors Matt McCutchen, John Pardon, and Bohua Zhan, and senior George Boxer.



*U.S. IOI Contenders (left to right):
Matt McCutchen, John Pardon, George Boxer,
Bohua Zhan*

The IOI was held August 10–20, 2006, in Mérida, Mexico, the heart of the Yucatán peninsula. A city of about one million people, Mérida is surrounded by territory of the former Mayan empire and is only 1.5 hours from Chich'en Itza, the renowned pre-Columbian archaeological site.

Mérida's weather in August is hot, of course: close to 100°F, with very high humidity. The contestants spent most of their time in air-conditioned spaces, but we poor organizers worked in the hot convention center, where the roof actually collected more heat than we'd have experienced sitting outside.

The first day's tasks were quite traditional, similar to the USACO contests and algorithmic in nature. Matt McCutchen led the way with a perfect 300 points; all four U.S. students were in contention for gold medals (awarded to 1/12 of the contestants).

The second day's tasks were nontraditional; scores were dramatically lower across the board. John Pardon again pulled a gold medal out of an extremely stressful situation; the other three earned silver medals.

All in all, the USACO continues to serve the world market, with many students at the IOI saying that USACO training enabled them to earn a berth on their country's team and a medal at the IOI.

Congratulations to all the great performers. Thanks to USENIX, we can

continue this fine program that encourages and recognizes pre-college programmers.

SAGE UPDATE

STRATA CHALUP

SAGE Programs Manager

Greetings from USENIX. We've been working on many SAGE fronts: print, Web, and the LISA conference. Here's the latest.

Booklets News

Another manuscript in the SAGE Short Topics in System Administration series, *The Internet Postmaster*, by Nick Christenson and Brad Knowles, has now gone through a stringent technical review and is in production. Serving as postmaster for a site involves knowing a couple of decades' worth of oral tradition, plus a solid grounding in technical and RFC issues. There really hasn't been a single place where this kind of information has been gathered, until now. As someone who has done a fair bit of postmastering in the past, I have to say that I'm completely pleased with this booklet. It's a real first!

Another booklet that "pulls it all together" is in the pipeline, on the topic of superuser privileges—how to manage them, how to implement them, what constitutes ethical behavior, and more. A first draft of *Being Root*, by William Charles and Xev Gittler, has been reviewed by our editors, and we expect that by the time you read this, the final manuscript will be in technical review. I'm really looking forward to this one, too.

Current SAGE members can access the whole Short Topics series online at <http://www.sage.org/pubs/>. Drop by and enjoy some great info.

LISA '06 Registration

It's going to be an outstanding LISA again this year! We've got a bunch of great tracks and training sessions, and some surprises. I'm really looking forward to Cory Doctorow's keynote, "Hollywood's Secret War on Your

NOC." Elizabeth Zwicky will be talking about her experiences Teaching Problem Solving, our wildly popular Hit the Ground Running sessions return, and the refereed paper track has some really hefty stuff ranging from Customer-Friendly Kernel Analysis, through NetFlow, to Managing Large Networks of Virtual Machines. We mentioned surprises, but first . . .

Back again by popular demand, training will be going on throughout the week. Starting on Sunday, we are offering some of our most popular training during the entire week of the conference. There's so much going on during the main portion of the conference that combining tech sessions with training is still the best way to get the most out of your LISA experience. Plus, you get the biggest discount! Find out more at <http://www.usenix.org/lisa06/>.

Now, about those surprises:

#1: An entire Friday track on security, covering topics such as Black Ops pattern recognition, fighting remote takeover attacks such as zombie or bot-net kits, and special issues for corporate security. Way cool.

#2: Bob Apthorpe and Dan Klein will present a new conference closer, "Improv for SysAdmins."

Registration opened at the beginning of September. Register by November 10 to save up to \$300!

New SAGE Web Site

We've been unveiling the new look in bits and pieces, putting our efforts primarily into delivering our programs and conference activities. But we're scheduled to have the site live in early October, maybe sooner. Expect a crisp, clean look, lots of new info and new services, updates of long-neglected pages, and better navigation.

Learn more about SAGE at <http://www.sage.org/>.

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 hard-

ware or software that you helped install and configure, as long as you are not affiliated with or paid by the company you are writing about.

- Personal attacks

FORMAT

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

DEADLINES

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

COPYRIGHT

You own the copyright to your work and grant USENIX permission to publish it in ;login: and on the Web. USENIX owns the copyright on the collection that is each issue of ;login:.

You have control over who may reprint your text; financial negotiations are a private matter between you and any reprinter.

FOCUS ISSUES

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

conference reports

THANKS TO THE SUMMARIZERS

2006 USENIX ANNUAL TECH

Marc Chiarini
Rik Farrow
Wei Huang
John Jernigan
Scott Michael Koch
Kiran-Kumar Muniswamy-Reddy
Partho Nath
Aameek Singh
Chris Small
Yizhan Sun

SRUTI '06

John Bethencourt
Balanchander Krishnamurthy
Anirudh Ramachandran

EVT '06

Aaron Burstein
Sarah P. Everett
Dan Sandler
Ka-Ping Yee

SUMMARIES

2006 USENIX ANNUAL TECHNICAL CONFERENCE	85-104
2ND WORKSHOP ON STEPS TO REDUCING UNWANTED TRAFFIC ON THE INTERNET (SRUTI '06)	104-106
2006 USENIX/ACCURATE ELECTRONIC VOTING TECHNOLOGY WORKSHOP (EVT '06)	107-113

2006 USENIX Annual Technical Conference

Boston, MA
May 30-June 3, 2006

KEYNOTE: PLANETLAB: EVOLUTION VS. INTELLIGENT DESIGN IN PLANETARY-SCALE INFRASTRUCTURE

Larry Peterson, Professor and Chair, Department of Computer Science, Princeton University; Director, PlanetLab Consortium

Summarized by Yizhan Sun

PlanetLab is a global platform for evaluating and deploying network services. It currently includes 670 nodes, spanning 325 sites and 35 countries, and has more than 3 million users. PlanetLab hosts many kinds of services, including file transfer, routing, DNS, multicast, Internet measurement, and email.

Larry Peterson summarized design requirements for the PlanetLab architecture:

1. It must provide a global platform that supports both short-term experiments and long-running service.
2. It must be available now, even though no one knows for sure what "it" is. In other words, we must deploy the existing system and software.
3. We must convince sites to host nodes running code written by unknown researchers from other organizations. This requirement is satisfied by building a relationship between users and service providers through trusted PLC (PlanetLab Consortium).
4. Sustaining growth depends on support for site autonomy and decentralized control.

5. It must scale to support many users with minimum resources available.

Peterson explained that they favor evolution over a clean slate, and design principles over a fixed architecture. Design principles include:

- leverage existing software and interfaces
- keep VM monitor and control plane orthogonal
- exploit virtualization
- give no one root (no more privilege than necessary)
- support federation

VIRTUALIZATION

Summarized by Marc Chiarini

■ Antfarm: Tracking Processes in a Virtual Machine Environment

Stephen T. Jones, Andrea C. Arpaci-Dusseau, and Remzi H. Arpaci-Dusseau, University of Wisconsin, Madison

Stephen Jones presented an approach that allows virtual memory managers (VMMs) to track the existence and activities of guest OS processes. Process-aware VMMs are better able to implement traditional OS services such as I/O scheduling, which leads to improved performance over host and guest OS implementations. The main advantages of the authors' approach are threefold: (1) the VMM does not require detailed knowledge of the guest's internal architecture or implementation; (2) no changes to the guest OS are necessary (a big win in the case of legacy or closed-source components); and (3) accurate inferral of process events incurs a very low overhead (2.5% in their worst-case scenario). The team implemented and evaluated their techniques on both x86 and SPARC architectures with the Xen VMM hosting Linux and the Simics full-system simulator hosting Windows.

Jones described the mechanism by which a VMM can detect process creation, destruction, and context switches in the guest. On x86, Antfarm tracks the contents of the privileged CR3 register, which points to the page directory for the process currently running in the guest. When the CR3 changes to any of a particular range of values, it can be inferred that a context switch has occurred. If the CR3 is loaded with a previously unseen value, it can further be inferred (and the VMM can track) that a new process has been created. The VMM makes two more observations to determine whether a process has been destroyed: Windows and Linux systematically clear nonprivileged portions of page table pages before reusing them; the TLB must also be flushed once an address space has been deallocated. If the VMM determines that the number of assigned pages in a process's address space has gone to zero and that the TLB has been flushed (by loading CR3 with a special value), the VMM can rightly infer a process exit. Similar techniques are available for SPARC architectures. Jones concluded with a case study of Antfarm's performance improvements for an anticipatory disk scheduler: By understanding which disk I/O requests come from which guest processes, a scheduler can try to optimize requests across all processes in all guests.

■ *Optimizing Network Virtualization in Xen*

Aravind Menon, EPFL; Alan L. Cox, Rice University; Willy Zwaenepoel, EPFL

Awarded Best Paper!

Aravind Menon presented three modifications to the Xen architecture that significantly improve its network performance. First, the approach implements high-

level network offload features for guest domain interfaces, including scatter/gather I/O, TCP/IP checksum, and TCP segmentation offload; second, the performance of the I/O channel between the guest and network driver domains is enhanced; last, the VMM is modified to allow guest OSes to use efficient virtual memory primitives, including superpage and global page mappings.

After a brief overview of the Xen Network Virtualization Architecture, Menon discussed the team's optimizations in detail. He noted that 60–70% of the processing time for transmit/receive operations is spent in the I/O channel and bridging within the driver domain. To help combat this bottleneck, an offload driver is inserted just before the NIC driver on the path to the physical NIC. This driver implements in software whichever offload features are not already implemented on the NIC. A 4x, 2.1x, and 1.9x reduction in execution cost was achieved in the guest domain, driver domain, and Xen VMM, respectively.

Menon and his team also attacked the mechanisms used to transfer packets over the I/O channel between the guest and driver domains. They found that the current technique of page remapping for each network packet is not necessary in many cases. Using simple methods, such as data copying, packet header investigation, and MTU-sized socket buffers, the team achieves a 15.7% and 17% improvement in transmission and reception, respectively, across the I/O channel.

Overall, the optimizations explored in the research improved the transmit throughput in guest domains by a factor of 4.4 and the receive throughput in the driver domain by 35%. The team needs to do further research to

determine effective techniques for improving receive performance in the guest domain. In the Q&A, Mike Swift asked about other common network optimizations and how they may be applicable to this work. Menon responded that more offload features may be useful but their benefit has not yet been studied.

■ *High Performance VMM-Bypass I/O in Virtual Machines*

Jiuxing Liu, IBM T.J. Watson Research Center; Wei Huang, The Ohio State University; Bulent Abali, IBM T.J. Watson Research Center; Dhableswar K. Panda, The Ohio State University

Jiuxing Liu presented a new device virtualization model, VMM-bypass I/O, that allows guest OSes to perform time-critical I/O operations without diverting through the VMM or other specialized I/O VMs. The problems with these techniques are manifest when one considers that every I/O operation involves the VMM, making it a potential bottleneck. Additionally, the second technique results in expensive context switches. The key is to use a “guest module” device driver installed in guest VMs that handle setup and management operations of direct I/O. These modules communicate with “backend modules” within either the VMM or a privileged device driver VM. Co-located with backend modules are the original privileged modules that know how to make requests to intelligent I/O devices.

Liu went on to describe the InfiniBand architecture and the design and implementation of Xen-IB, their InfiniBand virtualization driver for Xen. Infiniband is a high-speed interconnect to various devices that supports OS-bypass, allowing processes in a host OS to communicate (semi-)directly with the hardware. The prototype built by the

research team supports all privileged InfiniBand operations, including initialization, resource management, memory registration, and event handling. Liu gave a rundown of the InfiniBand cluster used as a testbed, consisting of Xen 3.0 running RedHat AS4 on Intel Xeon machines. Comparisons were presented between native InfiniBand and Xen-IB for latency and bandwidth (negligible differences), event/interrupt handling (10- to 25- μ s overhead introduced), memory registration (25–35% overhead), IP over InfiniBand (<10% throughput degradation for >16KB-size messages), and MPI bandwidth and latency benchmarks (negligible).

Finally, Liu discussed some remaining challenges. Providing a complete and efficient bypass environment requires addressing some remaining important issues, such as safe device access, QoS among competing VMs, and VM check-pointing and migration. The team is eyeing some directions they think will be fruitful. The prototype can be downloaded at <http://xenbits.xen-source.com/ext/xen-smartio.hg>.

INVITED TALK

■ *Deploying a Sensor Network on an Active Volcano*

Matt Welsh, Harvard University

Summarized by John Jernigan

Matt Welsh related his experiences during two deployments of sensor arrays on volcanoes in Ecuador. He explained that the arrays could potentially provide civil authorities with warnings of volcanic activity and help mitigate hazards. The research team spread “motes” (small and inexpensive wireless sensors) in a swath around the volcano and measured seismic and acoustic activity in real time. By compari-

son, traditional methods of seismic data logging involve manual collection of data from the field site, which can be very remote and difficult to reach. In addition, wireless sensors can cover a larger amount of terrain because of lower costs.

The basic system involved the motes, synchronized by GPS timestamps and sophisticated algorithms, propagating data over an ad hoc mesh network to a radio modem that communicated with a base station many kilometers away. Deploying wireless sensors presents significant technological challenges, however, as high sampling rates generate huge amounts of data, and maintaining accurate timing of captured events is absolutely critical for use by seismologists.

He indicated that node reliability was one of the largest concerns. Ironically, it was not the sensor network that failed often in the deployment, but the base station at the observatory, where a laptop would experience sporadic electrical outages. Logistical issues and bad luck seemed to overshadow the technological acclaim of the sensors as system uptime sank. Seismologists working with the research team lost confidence in the data set as a result of reliability issues. Nevertheless, the data set could be cleaned up and analyzed using external validation techniques, including data from third-party data-logging stations.

Some of the lessons learned from the deployments were as follows: Accurate timing of captured events must be the first priority. The goals of the computer scientists and the seismologists were sometimes disparate, and this affected the usefulness of the gathered data. Nodes should have been collocated with existing data-logging stations for later verification of data. Finally, never take for granted that you can

simply find an electrical outlet when you need one!

The next steps for the technology involve nailing down timing issues so that earthquakes can be localized in real time, and utilizing 3D mapping techniques to map the inside of volcanoes.

STORAGE

Summarized by Wei Huang

■ *Provenance-Aware Storage Systems*

Kiran-Kumar Muniswamy-Reddy, David A. Holland, Uri Braun, and Margo Seltzer, Harvard University

Kiran-Kumar Muniswamy-Reddy presented a provenance-aware storage system. In the context of his work, provenance refers to the information that describes data in sufficient detail to facilitate reproduction and enable validation of results. Kiran-Kumar started his talk with several usage cases of provenance-aware storage, such as applications in homeland security, archiving, and business compliance, where accessing the history of files may be critical to end users. However, as Kiran-Kumar pointed out, support for provenance is very limited in file systems. Most of the current solutions are domain-specific, which may cause the data and the provenance to be out of sync. And in many cases the solutions are simply lacking.

Kiran-Kumar argued for the importance of PASS, which keeps the data and the provenance tightly bound and provides transparent management. He then introduced their design of PASS. In their design, the collector records the provenance data or events and passes the records to the file system. The storage layer, which is a stackable file system called PASTA, uses an in-kernel database engine to store the metadata. And the query tool makes the

provenance accessible to users. Kiran-Kumar showed that their implementation had reasonable overhead on applications, both spatially and temporally.

Kiran-Kumar concluded his talk with several research challenges they are experiencing through their prototype study, such as searching suitable security models, pruning of provenance, and addressing the network attached storage. In the Q&A session, Kiran-Kumar was asked whether there are any micro-benchmark evaluations for PASS. He indicated that small file operations micro-benchmarks entail up to 100–200% overhead time. However, since most applications do not access the storage system that often, the overhead is usually acceptable for applications.

■ *Thresher: An Efficient Storage Manager for Copy-on-write Snapshots*

Liuba Shrira and Hao Xu, Brandeis University

Thresher targets BITE (Back-In-Time Execution) applications that take snapshots of the past state, inspect the snapshots with BITE, and retain snapshots deemed as interesting for an unlimited time for future analysis. Liuba started her talk with a discussion of why today's snapshot systems are inadequate for BITE applications. She pointed out that it is critical to provide applications with the ability to discriminate among snapshots, so that valuable snapshots can be retained while the less valuable ones can be discarded or moved offline, because although disk space is cheap, administration of storage becomes costly.

In the second part of the talk, Liuba introduced Thresher, a snapshot storage manager based on new copy-on-write snapshot techniques. Thresher is the first to provide applications with the ability to discriminate among

snapshots efficiently. Liuba focused on two important concepts in Thresher: discrimination and segregation. Applications discriminate among snapshots by ranking them according to their importance. The storage manager segregates differently ranked snapshots efficiently, so that higher-ranked snapshots can be accessed faster and lower-ranked snapshots can eventually be discarded without affecting the accessibility of higher-ranked ones and without disk fragmentation.

Lazy segregation technique allow the rank of snapshots to be specified after the snapshots are taken, enabling BITE-based ranking. Liuba focused on the diff-based segregation technique and the optimizations for low-cost reclamation and faster access to snapshots. Liuba concluded her talk with performance evaluation of Thresher. She showed that lazy segregation and faster snapshots can be implemented with very low performance overhead, allowing a huge reduction in storage requirements for snapshots.

■ *Design Tradeoffs in Applying Content Addressable Storage to Enterprise-scale Systems Based on Virtual Machines*

Partho Nath, Penn State University; Michael A. Kozuch, Intel Research Pittsburgh; David R. O'Hallaron, Jan Harkes, M. Satyanarayanan, Niraj Tolia, and Matt Toups, Carnegie Mellon University

Partho Nath presented their experience on applying Content Addressable Storage (CAS) to enterprise-scale systems based on virtual machines. Partho first described the Internet suspend/resume (ISR) client-management system, which is the execution environment at which their work is targeted. ISR is a virtual-machine-based client manage-

ment system. It stores the user execution environments as parcels, which are the complete VM images, including memory and disk snapshot. Different versions of parcels are stored in a lossless manner.

Partho then asked two questions, both of which are answered by their evaluations in this paper: Can content-aware storage reduce the (1) storage and (2) network requirements in ISR systems? And, if so, by how much? Their evaluation consisted of three dimensions: the policies, the chunk size, and gzip compression. They evaluated three policies for managing the parcels: the non-CAS baseline policy ("delta"), which stores different versions of parcels for each user as the diff of the previous version; the intra-parcel policy, where each parcel is represented by a separated pool of unique chunks shared by all versions from the same user; and the ALL policy, where all parcels for all users are represented by a single pool of chunks. Gzip can be used to further compress the data.

Partho showed their evaluation results. He pointed out that adopting CAS into the storage system significantly reduces storage requirements, especially when using relaxed policy (ALL policy). And within CAS policies, using smaller chunks works best in spite of metadata overheads. Another important observation is that CAS policies alone can consume less storage than a non-CAS policy with gzip compression, which avoids the expensive compression operations. In response to a question on the performance overhead of hash calculation for CAS policies. Partho indicated that they had not experienced any noticeable slowdown for hash calculations.

INVITED TALK

■ Panel: Open Source Software Business Models

Mike Olsen, Oracle, Sleepycat; Brian Aker, MySQL; Miguel de Icaza, Novell, Ximian

Moderator: Stephen Walli, Optaros, Inc.

Summarized by Scott Michael Koch

The discussion began with each panelist sharing his opinions and experiences with OSS. Although the panel agreed that OSS businesses can be very successful, Miguel felt that giving away your company's product for free was a risk, and he does not recommend starting a business of this type. The panel seemed to agree that there are only certain circumstances in which a OSS business can have success. Mike reminded us that it is hard to start any sort of business, and Brian added that selling any sort of software, whether proprietary or open source, today is like "setting up a tip jar" in that you just hope that enough people are willing to pay for your software. Brian felt that, if you want to make money, a service and support model or an ASP model makes the most successful long-term option, instead of trying to sell a binary. It was pointed out that people are becoming very comfortable with the subscription model. Miguel felt that the model of building a proprietary server with free clients was the way to go. Everyone agreed that for the traditional model of selling OSS to be successful, it was key to find a niche in the market where your product was something that everyone needed. Mike summarized this well by saying that using open source software can be successful as a tactic if it supports your overall strategy as a business.

The panelists then went on to talk about the interactions and relations with the communities

that surrounded their respective businesses. Mike explained that the community surrounding Bdb consisted mostly of users of the Bdb library, and although they benefited from the many eyeballs examining their code and enforcing high quality, there are no outside contributors. For MySQL, Brian said that ideas for features and quality bug reports are the most important contributions they receive from their community. Miguel explained that his current project, Mono, receives many external code contributions, and he believes that the amount and type of contributions strongly depend on the maturity of the code base. Mike then said that the most important contribution from the community is the adoption of their software, which increases the visibility and popularity of the software in the community. Inspired by a question in the audience, the panel discussed some lessons they had learned from their past experiences with OSS businesses. The only common problem they mentioned was that it can be frustrating trying to deal with the slashdot-type community, and anyone starting an OSS business should be aware of the energy and effort required to constantly nurture that community. Learn to communicate with your audiences appropriately. Marketing to the typical OSS user is best done through attending conferences, setting up blogs, and communicating with them one-on-one.

SHORT PAPERS SESSION I

Summarized by Kiran-Kumar Muniswamy-Reddy

■ Compare-by-Hash: A Reasoned Analysis

J. Black, University of Colorado, Boulder

John Black presented this paper, a rebuttal to Val Henson's HotOS

2003 paper that criticized the use of hash functions to compare two files to tell whether they are the same. John presented various arguments to make the point that although hash functions may not be strong enough for scenarios where there is an adversary, they are more than sufficient for usage scenarios where there is no adversary. John concluded the talk by stating that the computation power needed to find collisions in a 128-bit hash function in 24 days would cost around \$80,000, and for a SHA1 it would take \$80,000,000 and 2 years. So other approaches such as social engineering might be more successful. In the Q&A session, John agreed that the current hash functions may not be secure after 20 years.

■ An Evaluation of Network Stack Parallelization Strategies in Modern Operating Systems

Paul Willmann, Scott Rixner, and Alan L. Cox, Rice University

The paper was presented by Paul Willmann. The paper evaluates three different strategies for parallelizing network stacks: (i) message-based (MsgP), (ii) connection-based using threads for synchronization (ConnP-T), and (iii) connection-based using locks for synchronization (ConnP-L). MsgP is the slowest of the three, as it has a significant amount of locking overhead. ConnP-T has lower locking overhead but experiences significant scheduling overhead. ConnP-L has the best performance, as it mitigates both locking and scheduling overheads. Paul concluded the talk by stating that current programs themselves haven't been written to take advantage of parallelism.

■ **Disk Drive Level Workload Characterization**

Alma Riska and Erik Riedel, Seagate Research

The paper, presented by Eric Riedel, characterizes workloads in various kinds of devices, including PCs, laptops, and home devices. The authors collected traces by inserting SCSI or IDE analyzers into the I/O bus and intercepting the signals. Some of their findings are as follows. The read/write ratio, the access pattern, and write traffic vary by application. The request size is around 4 kB. I/O bus and disks are underutilized. In the enterprise/desktop environment, requests are spread all over the disk. Videos are highly sequential. Access characteristics depend on the environment: cache management, arrival, and service processes at the disk drive. Characteristics common in environments include idleness and burstiness.

Two key questions were addressed in the Q&A session. (1) How are your results different from the Hewlett-Packard paper? Eric replied that they don't compare, because of the large difference between the devices and environment presented in this paper and those in that paper. (2) Can we get the traces? Eric replied that they may be able to give out the traces.

■ **Towards a Resilient Operating System for Wireless Sensor Networks**

Hyoseung Kim and Hojung Cha, Yonsei University

Currently, the only way to recover from crashes in sensors is to reset the sensors. Hyoseung presented RETOS, a resilient, expandable, threaded operating system. RETOS achieves this by introducing dual mode operation and static/dynamic code checking. Dual mode separates out the

application and kernel code. RETOS then performs checks on the machine instructions to ensure that applications do not write to or jump to an address outside their logical portion. Some of the instructions can be verified statically at compile time and others need to be verified at run time; for the latter, verification code is injected while compiling the code.

■ **Transparent Contribution of Memory**

James Cipar, Mark D. Corner, and Emery D. Berger, University of Massachusetts, Amherst

The talk was given by James Cipar. Contributory applications such as condor, SETI@home, and Farsite utilize wasted CPU cycles, idle memory, and free disk space on participating user machines. They can, however, disrupt user activity by forcing user pages to disk. Normal approaches such as scheduling do not help with memory and disk usage. James presented the Transparent Memory Manager (TMM), which controls memory usage by contributory applications, thereby ensuring that it does not impair normal system functionality. TMM works by detecting the imprint of user applications and then limits the memory footprint of contributory applications accordingly. TMM detects the memory imprint by keeping an LRU histogram of memory accesses. When pages need to be allocated but there are no free pages and both normal and contributory apps have exceeded their limit, normal apps are favored. Otherwise, the page is evicted from the class that has exceeded its limit.

INVITED TALK

■ **Success, Failure, and Alternative Solutions for Network Security**

Peiter Zatko, BBN Technologies

Summarized by John Jernigan

Peiter Zatko, a.k.a. "Mudge," spoke of the current state of affairs in network security, offering his musings and concerns. He began with a summary of his background; he is a former member of l0pht, has worked with the National Security Council, and started his own security company, Intrusic. He is now working for BBN Technologies.

Peiter first addressed some of the pertinent questions in network security today, asking how much progress we have really made, where we have messed up, and where we are spinning our wheels. He points out that the Internet has far outpaced our understanding of security. As the Internet grew and added nodes and users, the threat model increased, but software was still not being designed with any notion of security. Eventually, a distinction between internal and external environments evolved, much like a military compound with a fence and a gateway to swap credentials, but internal resources were not themselves secure. Presently, many networks are watched by intrusion detection systems, which only let in and out certain traffic and flag dubious behavior. However, 0-days still penetrate these defenses and will always be one step ahead of patches by definition. Of even greater concern is that the defenses do little to prevent unauthorized activity within the network itself. Peiter emphasized that our threat model has changed, but our defenses have not grown with the environment.

On the topic of buffer overflows, he suggests that, even if they all went away, we would be left with plenty of threats, such as root-kits, sniffing, and trojaned applications. In addition, overflows of many different types, such as heap-based and pointer overflows, abound. In other areas, it has become too easy for naive developers to create enterprise applications, such as with PHP, and vulnerable software is live and rampant.

Peiter suggested that firewalls, intrusion detection systems, and intrusion prevention systems are not really the answer to our security woes. We should really be looking at what goes on inside a network as well. We should not see drastic changes in the behavior of nodes or out-of-order packets on internal systems with few routers. We need to adhere to RFCs and also to detect when behavior does not match real-world trends on the network.

The thought we are left with is that security is still a cat-and-mouse game, and more intelligent methods of security are strongly needed to keep pace with developing technologies and Internet expansion.

SERVER IMPLEMENTATION

Summarized by Scott Michael Koch

■ *Implementation and Evaluation of Moderate Parallelism in the BIND9 DNS Server*

Tatuya Jinmei, Toshiba Corporation; Paul Vixie, Internet Systems Consortium

Tatuya Jinmei presented a paper about improving the performance of ISC's BIND9, a widely used DNS server. The authors found that it had poor performance with threads and did not benefit from having multiple CPUs. Some of the key bottlenecks they found were memory

management, operations on reference counters, and thread synchronization. While looking for these bottlenecks they found that 43% of total run time was spent waiting to acquire locks. They were able to eliminate the bottleneck in memory management by enabling an internal memory allocator to provide each thread with a separate pool of memory, and they also separated the workspaces of the threads since the temporary data used by a single thread did not need to be shared. They eliminated the bottleneck on reference counters by using atomic operations without locks instead of using pthread locks. Although this solution is less portable, since it depends on specific hardware architectures, all the same platforms are supported, as before, through an abstract API. They also implemented more efficient reader-writer locks by basing the design on Mellor-Crummey's Algorithm.

By identifying and eliminating the thread synchronization overhead and these other bottlenecks, they significantly improved BIND9 performance with multiple threads. They confirmed their improvements by testing them on a four-way machine. Their improvements should be available in BIND9 as of version 9.4.0a5. Although they focused on BIND9, they feel the techniques and improvements that they used are applicable to other thread-based applications.

■ *Flux: A Language for Programming High-Performance Servers*

Brendan Burns, Kevin Grimaldi, Alexander Kostadinov, Emery D. Berger, and Mark D. Corner, University of Massachusetts, Amherst

Brendan Burns talked about a new programming language with the goal of simplifying the process of building high-perfor-

mance servers. The authors felt that, when building these types of servers, having to deal with the thread programming makes the code much harder to reuse and adds the possibility of deadlocks in the code. Having to worry about threading is an unnecessary burden on the programmer and can significantly complicate debugging. Flux aims to separate the programming process so that all the concurrency control is taken care of with its simple language, and the logical programming of the server is done in C, C++, or Java. They found that programming with this separated method allowed the programmer to better understand the overall functionality of the different parts of the server without having to worry about the underlying implementation of each of the parts. Using Flux they were able to put together a Web server, image rendering, a BitTorrent peer, and a game server that performed as fast as or faster than their counterparts written entirely in C. More information and a working example of both the HTTP and BitTorrent Server can be found at <http://flux.cs.umass.edu/>.

■ *Understanding and Addressing Blocking-Induced Network Server Latency*

Yaoping Ruan, IBM T.J. Watson Research Center; Vivek Pai, Princeton University

The last paper in this session was way over my head. Even after going through the presentation and attempting to read the paper, any attempt at writing a summary just turned into trying to reword the abstract of the paper. You can find out more about this paper at http://www.cs.princeton.edu/nsg/papers/latency_usenix_06/.

INVITED TALK

■ Is University Systems Teaching and Research Relevant to Industry?

Moderator: Gernot Heiser,
NICTA/UNSW

Panelists: Stephen Geary, HP, head of Linux strategy in R&D; Orran Krieger, IBM, K42, Xen strategy; Margo Seltzer, Harvard, Oracle, Sleepycat; Tim Roscoe, Intel Research Berkeley, OS, distributed systems; Jim Waldo, Sun Labs, Jini, Harvard; Andy Tannenbaum, Vrije University, Minix, 16 textbooks

Summarized by Chris Small

Gernot started by stating the claims of industry: that universities are not producing the kinds of systems people need and are producing irrelevant research. Industry does research, but because it doesn't get published, industry gets no respect from academia. He then asked each of the six panelists to respond to his opening statement.

Andy Tannenbaum: What are universities for? To serve students? Industry? Government? Faculty? The average student's career lasts 40 years; I want to focus on stuff that will be useful for 20 years, emphasizing principles, not facts. Teaching how MS-DOS works might have been very interesting 20 years ago but is less interesting now. I want to teach how to keep the design simple, good software engineering practice, and to expect paradigm shifts. Think in terms of systems. Ignore hype; don't forget the past; ideas get recycled. I think sometimes I'm supposed to teach "bloat-ology."

Jim Waldo: I'm the industry guy and mostly agree with Andy. But he's going to concentrate not on 20 years from now but now. Students never have to maintain a system longer than it takes to write the paper. "I don't fix bugs; I have a Ph.D.—I write new things." When you get your

Ph.D. you're not done: You're ready to start. But how can you teach system building and maintenance in a university setting? You can't. Industrial research used to be "short-term"; academics did "longterm" research. But it's no longer true—academics are doing short-term one-off things to get the next grant; industry looks at the longer term. So people coming out of universities are not ready for the adult world, but Jim doesn't expect them to be.

Tim Roscoe: I don't like the division between industry and academic research. Intel sets up lablets of 10–20 researchers, closely attached to the university. They do not pursue patents on joint work. Everything is supposed to be published, open source, etc. Intel can do this because Intel is a manufacturing company, not a software company. It probably doesn't make sense for Microsoft to do this. Intel wanted to strategically influence the way universities work. Can we get universities to do work that's of more value to Intel? Intel provides industrial relevance and resources. Planet-Lab is an example—an attempt to change the research culture in distributed systems in academia. There is less emulation and less "we ran this on 17 machines, so clearly it scales up to 100,000 nodes" thinking. Students take their distributed systems textbook and try to implement the ideas on PlanetLab—and it doesn't work. They learn a lot about what really matters by doing it. How do you teach systems principles? It's very hard, unless you're getting experience building real systems.

Margo Seltzer: There are two different topics here that this nice academic-oriented panel are trying to hide from you. Undergrad education and grad

education are two very different things. This conference is for graduate types, but industry mostly hires undergrads. Andy is fundamentally wrong: Universities and colleges are there to serve society, not students or faculty. We need to help students figure out what path they want to follow and how to follow it, not to build raw fodder for industry, nor clones of ourselves. They want to develop thinkers and people who can make good decisions, even if they end up being lawyers. Tension exists between giving them tools and giving them a specific skill set. In the long term, the tools are more important.

Orran Krieger agrees with Jim that there has been a longterm/short-term inversion. K42 was developed even though many people in the company thought it was a waste of time, but important skills and knowledge were brought into the company—for example, Linux and pervasive virtualization. What we want from Ph.D.s are people who will come up with radical ideas to change things. Hammond said, "Don't read all the relevant literature—think about the fundamentals and the problem for a month, then go read the literature." Researchers should work on big, irrelevant systems and work in teams. We used to have five-, six-, and seven-year Ph.D.s, and that gave them time to thrash and come up with their own ideas.

Stephen Geary: Hey, I'm a mechanical engineer. I have product responsibilities, making sure that Linux and open-source technologies work on Itanium-based systems. A chunk of code or a piece of research by itself is not interesting, or not as interesting as long-lived supported systems that do things for customers. You get them for four years; I get

them for 40 years. You have to teach people about budgets and schedules.

Andy: The job of the university is to serve society, but they're turning out lawyers.

Jim: What I'm really looking for when I'm hiring is people who know "how," not who know "that"—people who know how to think, not people who know facts (e.g., how to build a particular kind of hash table).

Q. Is academia doing anything right?

Margo: We need to adjust expectations. Andy's students understand how to think about systems, but they don't understand every line of Windows.

Gernot: How is it that academia can churn out mechanical and electrical engineers but not computer systems folks? Why are there so few real systems departments?

Orran: Linux progress is much slower than it should be because they ignore the literature. They did a brilliant job of cloning UNIX. But that's not going to revolutionize the field. The success of Linux has stifled the ability to do the kind of research that will move the field forward. Ten years ago there were more ideas moving things forward.

Andy: It's not the job of universities to produce open source code. But many of the people producing open source code are university graduates.

Q. People build their own tools. They should come out of university with the start of their own personal toolkit.

Tim Roscoe: That's insightful. One thing I've noticed about textbooks, particularly in systems, is that almost all of them are useless at teaching how to think about operating systems, planning to build a system, or

how to deal with a large body of code.

Margo: Open source is a fraud—there are a handful of people who commit to the Linux source tree, not tens of thousands.

Orran: We should have a tax on corporations—where their top people come from, money goes.

Margo: Computer science headcount is plummeting. Students think "computer science means programming, and programming will be outsourced."

Orran: One of the best things that is happening to academia is dropping enrollment. People used to get into academia because they were excited; for a while these were people who thought it was a good career move. Now a higher percentage of people are passionate about it.

Margo: It's not that we're only getting the passionate people. In 1992 (at Harvard) we had 30 concentrators; this year we have 12. People who are passionate about technology think, "Oh, I know how to write programs; I don't need to study computer science." Or people in other intellectual disciplines (e.g., physics), who used to have to learn how to program to get a summer job, got seduced, but now they learn these things in high school and ignore computer science in university.

Q. Of 16 graduates, 11 were double majors, and these were mostly in economics.

Gernot: To wrap up, what can we do? Or should we just give up?

Stephen: Gelato Consortium is a good example; it was founded by HP university relations to advance Linux, Itanium, and supercomputing.

Clem Cole: We need to teach people how to collaborate.

SECURITY

Summarized by Yizhan Sun

■ *Reval: A Tool for Real-time Evaluation of DDoS Mitigation Strategies*

Rangarajan Vasudevan and Z. Morley Mao, University of Michigan; Oliver Spatscheck and Jacobus van der Merwe, AT&T Labs—Research

An ISP network today faces many DDoS attacks. The defense decision for DDoS attack is often manual and complex. Many defense/mitigation strategies are available, and it is difficult for a network operator to choose the appropriate one in real time. The approach presented here is the Reval simulator framework.

Reval takes network state, attack info, and mitigation policy as input and goes through initialization, mitigation setup, traffic setup, and evaluation steps. The output of Reval is the optimal solution for a DDoS attack.

A case study on the Abilene network was illustrated in the talk. Two mitigation mechanisms can be applied in this case: blackholing and scrubbing. The result of using Reval to determine the right mitigation strategy in real time was explained and evaluated.

■ *LADS: Large-scale Automated DDoS Detection System*

Vyas Sekar, Carnegie Mellon University; Nick Duffield, Oliver Spatscheck, and Jacobus van der Merwe, AT&T Labs—Research; Hui Zhang, Carnegie Mellon University

Several strategies and their drawbacks for DDoS attacks were introduced:

- Wait for customer to complain—not effective at all
- Buy a per-egress detection device—expensive and not scalable
- Install devices at select locations—gives incomplete coverage and inaccurate limits on sensitivity

- Use existing data feeds (e.g., SNMP and Netflow)
- Use SNMP—entails low overhead and yields few false negatives, but has low diagnostic ability
- Use Netflow—has good diagnostics, yields few false positives, but has higher overhead and does not scale

LADS is a better approach. The mechanism behind LADS is to use time-series anomaly-detection triggers collection of Netflow and do fine-grained analysis afterward. Benefits of LADS include detection of high-impact attacks, efficient data collection and reduced computational cost, and flexibility.

■ ***Bump in the Ether: A Framework for Securing Sensitive User Input***

Jonathan M. McCune, Adrian Perrig, and Michael K. Reiter, Carnegie Mellon University

Jonathan McCune first introduced how a user's input (user name and password) can be stolen by a malicious application installed on Windows systems. Then he introduced a threat model and some assumptions of BitE, including a priori knowledge of which software is good. Then he proceeded to explain BitE architecture, setup, and operation.

BitE system architecture is based upon a partially trusted host platform with a BitE Kernel module installed and executed. The BitE kernel module and mobile client participate in key setup and bypass the traditional input path to avoid information being stolen by malicious applications.

BitE can be set up through device association and application registration and operates through several steps, including application request, verification of attestation, user interaction,

and establishment of session keys.

INVITED TALK

■ ***Architectures and Algorithms for Biomolecular Simulation***

Cliff Young, D.E. Shaw Research, LLC

Summarized by Partho Nath

This talk by Cliff Young was on the need for developing more powerful hardware to get closer to answering challenging questions in modern biology, chemistry, and medicine. A typical means of understanding phenomena in these fields is via molecular dynamics (MD)—simulation of biologically significant molecules at the atomic level. If performing such experiments were accurate and infinitely fast it would be easy to perform arbitrary computational experiments such as determining structures by watching them form, transforming measurements into data for mining later, etc. However, for a goal of, say, simulating about 64,000 atoms at a millisecond scale, with explicit water molecules, one would need a 10,000-fold increase in computational power if a single state-of-the-art processor were used, or a 1,000-fold speedup if a modern parallel cluster were used. The talk considered the pros and cons of several different available architectural options: (a) clusters of commodity processors, (b) general-purpose supercomputers (e.g., Blue-Gene), and (c) special-purpose supercomputing architectures.

The speaker was of the opinion that new specialized, enormously parallel architectures with special-purpose ASICs specially tailored for MD simulations are the answer. Optimizations could include arithmetic specialization, hardware tailored for speed

(e.g., hardware tables with parameters) but not too programmable, data flow to exactly where the data is needed, and design for almost never touching off-chip memory. Given that the class of algorithms to be run on these machines is well known, such a machine could be an order of magnitude faster than general-purpose supercomputers. The speaker commented that production of such a machine was already underway and could be expected in 2008. This machine is designed to have 16 segments at the physical level, each consisting of 512 nodes (ASICs) in a 8-cube toroidal mesh (to reflect the physical space being simulated). The speaker detailed the performance of this machine for the NT algorithm (a parallel algorithm for range-limited pairwise interactions of atoms). He noted that this architecture showed asymptotically less inter-processor communication, which translates to better scaling.

Most of the questions to the speaker addressed the machine under production. Regarding soft errors (given that the machine has thousands of nodes), the speaker commented that off-chip memory has ECC, whereas on-chip memory is supposed to be free from such errors. Additionally, the runtime does a checkpoint and reload of the simulation once every hour. Another question was whether writing code for such specialized hardware was going to be a significant bottleneck. The speaker agreed that this might be a significant issue, especially given that programmers were writing code in assembly for a specialized hardware. No compiler was being developed because the development cycle for a compiler would be longer than that of developing the code for the corresponding algorithms in

assembly itself. Given that the architecture is simplified by the absence of both speculation and out-of-order execution, writing efficient code for such an architecture may not be too bad. Answering a query on power demands, the speaker said that housing the machine would be another nontrivial task, both in terms of the physical space required and the cooling costs. On a question on the numeric precision of the machine, the author remarked that no floating-point arithmetic is used. Computations use a fixed-point subset of double-precision, i.e., 32-bit single-precision fixed-point arithmetic. The advantages gained here were that the simulation runs would be more deterministic and that the pipeline design would be simpler. Another question was on whether such a machine is viable at all: Given that the world market may absorb only about five such machines, would it not be cheaper to just build commodity clusters instead of such a specialized cluster? The speaker commented that with commodity clusters a 1,000-fold speedup would not be possible in a five-year timeframe. The speaker conceded that the size of the market justified by such an investment is still an open question.

MANAGEMENT AND ADMINISTRATION

Summarized by Kiran-Kumar Muniswamy-Reddy

■ *Sharing Networked Resources with Brokered Leases*

David Irwin, Jeff Chase, Laura Grit, Aydan Yumerefendi, and David Becker, Duke University; Kenneth G. Yocum, University of California, San Diego

David Irwin presented Shirako, a system to coordinate resource allocation between providers and consumers. Shirako introduces brokers, a software entity that

maintains inventories of resources offered by providers and matches requests with available resources. Leases are used to bind a set of resource units to a consumer for a lease term. Brokers issue tickets to consumers that are redeemed for leases at the providers. Shirako's design makes resource allocation independent of the application. During the Q&A, Vivek Pai asked how Shirako knows what data the application needs (i.e., what do you do when the applications need disk space but not CPU?) David replied that they tried to allocate better bandwidth and to allocate systems closer to the consumers. Vivek then asked how they dealt with applications that checkpoint their state and restart on a different system. David replied that other groups have been looking at this and they plan to build on that work.

■ *Understanding and Validating Database System Administration*

Fábio Oliveira, Kiran Nagaraja, Rekha Bachwani, Ricardo Bianchini, Richard P. Martin, and Thu D. Nguyen, Rutgers University

The talk was given by Fábio Oliveira. The goal of this work is to reduce database downtime. Most of database downtime is caused by mistakes made by database administrators. To this end, the authors conducted a survey of experienced administrators at SAGE to better characterize the source of these errors. They found that one common source of errors is that the deployment environment is different from the test environment. They also found that DBAs of all experience levels are prone to make mistakes.

They presented three forms of validation to reduce operator errors: trace-based, replica-based, and model-based. In the trace-based approach, they log the requests to and replies from

the live system, play the traces onto the system/components to be tested, and compare the two results to detect any errors. Some operations, such as change in schema, cannot be validated by the first two methods, so they propose a model-based approach. In this approach, the operator can specify the expected behavior using their model. The dynamic behavior of the system is then validated with that of the predicted model. In the Q&A, Atul Adya asked whether they closed the loop, that is, did they go back to the DBAs with their results? Fábio replied that they did not. In response to another question by Atul, Fábio replied that they did not deal with triggers.

■ *SMART: An Integrated Multi-Action Advisor for Storage Systems*

Li Yin, University of California, Berkeley; Sandeep Uttamchandani, Madhukar Korupolu, and Kaladhar Voruganti, IBM Almaden Research Center; Randy Katz, University of California, Berkeley

The talk was given by Li Yin. The common approach to meeting the service level objective (SLO) for storage systems involves the observe, analyze, and act loop. This approach involves manual interaction and is slow. There are existing tools that help automate the task, but these are again restrictive, as they can correct only one action, such as work throttling, data migration, or addition of new resources. Li presented SMART, a framework that considers multiple corrective actions.

SMART aims to maximize the system utility for a give optimization window. SMART contains four key components: (1) INPUT modules (containing sensors monitoring system state, SLOs, component modules, workload request rate, etc.), (2) a utility evaluator (which calcu-

lates the overall utility delivered by the system), (3) single action tools (to automate invocation of a single action), and (4) an action advisor that, based on the other three components, generates a schedule for actions to be invoked to improve system utility. The action advisor operates in two different decision modes: normal and unexpected. In normal mode, it proactively generates decisions to forecasted workloads by optimizing local actions to achieve global optima. In unexpected mode, it makes defensive decisions in response to unexpected variations in workloads. The reason for it being defensive is that the unexpected workload may be transient. There were no questions after the talk.

INVITED TALK

■ *Permissive Action Links, Nuclear Weapons, and the History of Public Key Cryptography*

Steven M. Bellovin, Columbia University

Summarized by Partho Nath

This talk traced the history of PALs (Permissive Action Links), detailing the motivation for their invention and those responsible for their creation. The speaker ran through a timeline of their use and evolution, highlighting the possible design choices made at those junctures, along with cryptography and key management for the different designs. The talk concluded with possible designs for modern-day PALs and what we might learn from them in designing secure systems. The slides for the talk can be found at <http://www.cs.columbia.edu/~smb/talks/pal.pdf>. The content for the talk can be found at <http://www.cs.columbia.edu/~smb/nsam-160/pal.html>.

SHORT PAPERS SESSION II

Summarized by Wei Huang

■ *sMonitor: A Non-Intrusive Client-Perceived End-to-End Performance Monitor of Secured Internet Services*

Jianbin Wei and Cheng-Zhong Xu, Wayne State University

Jianbin Wei first described the inadequacies of existing approaches for monitoring end-to-end user-perceived performance of Internet services, especially with increasing deployment of HTTPS services. Jianbin indicated that there is a strong need to deploy a performance monitor, which is nonintrusive, easy to deploy at the server side, and can handle HTTPS services.

Jianbin presented sMonitor, their solution to these goals. sMonitor consists of a package capture to collect live network packets, a packet analyzer to reconstruct the pages of HTTP/HTTPS transactions, and a performance analyzer to derive client-perceived response time of the monitored services. Jianbin focused on their solutions to several key design challenges, such as identifying encrypted HTTP requests from packet size analysis, handling pipelined requests, and parallel downloading.

Jianbin concluded with their evaluation of the accuracy of sMonitor in measuring HTTPS and HTTP services. He showed that errors between the client measurements and the reported performance of sMonitor, which is deployed at the server, are less than 8%.

■ *Privacy Analysis for Data Sharing in *nix Systems*

Aameek Singh, Ling Liu, and Mustaque Ahamad, Georgia Institute of Technology

The *nix access control model, as Aameek Singh pointed out, must provide good support for both data selectivity (e.g., which

data to share with other users) and user selectivity (e.g., with whom to share the data).

However, Aameek's study revealed that, in current *nix systems, the lack of convenience in data-sharing mechanisms often leads to users compromising their security requirements to conveniently fit the specifications of the underlying access-control model. Aameek talked about their studies on two multi-user *nix installations. Simply by scanning readable user directories and guessing executable-only directories, along with email and browser statistics, they were able to "attack" massive amounts of privacy data, which, they believed, were not exposed on purpose. Since the technical sophistication of the attacks is low and there is no quick fix to such vulnerabilities of private data, Aameek raised a major concern about the inadequate protection of privacy in *nix systems.

Aameek concluded the talk with some possible solutions to enhance privacy protection, such as using privacy auditing tools to monitor potential privacy data exposures or virtualizing the file system hierarchy differently for different users. But until that happened, Aameek said, users should pay more attention to monitoring the privacy of their own data.

■ *Securing Web Service by Automatic Robot Detection*

KyoungSoo Park and Vivek S. Pai, Princeton University; Kang-Won Lee and Seraphin Calo, IBM T.J. Watson Research Center

KyoungSoo Park presented a automatic robot detection framework to support a secure Web service. KyoungSoo first talked about the widespread existence of malicious bots, including those for password cracking and DDoS attacks. The increasing

abuse of robots motivated an accurate robot detection system.

KyoungSoo described their techniques to separate human browser activities from robot-generated Web traffic. They include browser detection and human activity detection. Browser detection is based on the observation that most robots are not standard browsers; it catches robots if the behavior deviates from that of normal browsers. Human activity detection directly detects humans by observing human activities such as mouse movement or keyboard events behind the browsers. Hardware events are being tracked in dynamically embedded Javascript and the activity is indirectly reported to the server via a fake image request. This technique is based on the fact that current robots are not generating hardware events.

KyoungSoo showed that most human activities can be distinguished within tens of HTTP requests. And the maximum false positive rate is low (2.4%). KyoungSoo also mentioned that with their system deployed on a CoDeeN content distribution network, complaints on robot-related abuse have dropped by a factor of 10. KyoungSoo admitted that serious hackers can still break their detection system and suggested using machine-learning techniques as a remedy.

■ ***Cutting through the Confusion: A Measurement Study of Homograph Attacks***

Tobias Holgers, David E. Watson, and Steven D. Gribble, University of Washington

David Watson introduced a measurement study of homograph attacks. A homograph is a character or string that is visually confusable with a different character or string. A homograph attack tries to fool a user into

visiting a nonauthoritative Web site.

David presented a study on the nature and quantity of homograph attacks. Using a nine-day trace of Web traffic from the Computer Science Department of the University of Washington, they probed the DNS to find registered names that are confusable with (i.e., a homograph to) the names of visited sites. The results of the study were fourfold: (1) No user visited a nonauthoritative site during the trace; (2) popular Web sites are more likely to have registered confusable names than unpopular sites; (3) registered confusable names tend to consist of substitutions of two or fewer confusable Latin characters, though some IDN (International Domain Name) substitutions were found; and, (4) the intent behind most registered confusable names is benign—predominantly advertisements. David concluded that homograph attacks currently are rare and not severe in nature. However, given the recent increase in phishing incidents, homograph attacks seem like an attractive future method for attackers to lure users to spoofed sites.

■ ***Stealth Probing: Efficient Data-Plane Security for IP Routing***

Ioannis Avramopoulos and Jennifer Rexford, Princeton University

Ioannis Avramopoulos started his talk by introducing the challenges in secure IP routing. He argued that data-plane monitoring must be part of any complete solution. However, existing proposals for secure forwarding with link-level fault localization capability are heavyweight, requiring cryptographic operations at each hop in a path. Ioannis presented a lightweight data-plane mechanism that monitors the availability of paths in a secure fashion. In intradomain routing, this mechanism also

enables the management plane to home in on the location of adversaries by combining the results of probes from different vantage points (called Byzantine tomography). Ioannis discussed advantages of stealth probing, including its incremental deployability, backward compatibility, and incentive compatibility.

Ioannis presented two deployment scenarios for stealth probing. He described how an ISP can deploy stealth probing to secure its own infrastructure. He also discussed how a pair of edge networks can deploy stealth probing to secure the path through untrusted ASes on the Internet.

INVITED TALK

■ ***Gold and Fool's Gold: Successes, Failures, and Futures in Computer Systems Research***

Butler Lampson, Microsoft Research

Summarized by Kiran-Kumar Muniswamy-Reddy

Butler Lampson started off by discussing trends in computer use. He then briefly enumerated things in the history of computer science that worked, things that didn't work and why they didn't work, and a list of things that "maybe" worked. He claimed that the future of computer science lay in applications that dealt with avoiding catastrophes and uncertainties.

In the context of Moore's law, improvement in hardware simplifies software. Better hardware enables new applications with the complexity going into software. Accordingly, the fields in which computers have been used has been growing. In the 1950s, computers were used for simulation. In the 1980s, they were used for communication and storage (e.g., email, airline tickets, and search engines). By

2010, computers will be embodied in the physical world, that is, interacting nontrivially with the physical world, embedded in factories, cars, robots, etc.

He then gave a list of things that worked: virtual memory, address space, packet nets, objects/subtypes, transactions, RDB and SQL, bitmaps and GUIs, the Web, and algorithms. The list of things that did not work includes capabilities, fancy type systems, formal methods, software engineering (all they did was have interfaces and count the number of lines), RPC (which failed because the idea was to try to mask the fact that the call was remote distributed computing), persistent objects (in which you end up storing a bunch of rubble, because of program bugs), and security (getting worse because there is a lot more software now; also, people don't like security, because security says no but people want to say yes), RISC (Intel retrofit the good ideas of RISC into their chips).

Things that may have worked include parallelism (which now we actually need because we have multi-core systems, but many programmers don't know how to apply the theory, so we probably can't make it work), garbage collection (which was not designed to be used by systems), interface and specifications (with substantial overhead in breaking down the system and specifying the interfaces, they are slightly successful in hardware but not in software), and reusable components (which [1] are expensive to develop, [2] are specific to how resources are allocated and have unique failure models, [3] have been successful in filters and big things [e.g., OSes, DBs, browsers]). Reusable components have not worked for Ole/COM/Web services; how-

ever, they have worked for companies such as Amazon, who can afford to have 20% of the things displayed wrong.

Systems research has failed at times, the classic case being that we didn't invent the Web. This is mainly because of the way we think. For example, we felt that the design and the idea of the Web are too simple. The idea of the Web had been around for some time but was never tried. Computer scientists would have tried too hard to come up with an optimal design. Another reason for the failure is that computer scientists tend to deny that things might work. For example, in the case of the Web, they would have just argued that it would never scale.

The future of systems research involves building systems that deal with uncertainty and that avoid catastrophe (e.g., reducing highway traffic deaths to zero). The problem involves computer vision; building world models for roads and vehicles; dealing with uncertainty about sensor inputs, vehicle performance, and a changing environment; and, finally, dependability. Butler defines a dependable system as one that avoids catastrophes. This ensures that the focus is on the really important and provides a way to reduce aspirations for a system. Catastrophe prevention has not always worked; for example, air traffic control specifications state that the downtime should be 3 seconds/year/workstation. But this is not true. The architecture of the system should have a normal mode and a catastrophe mode. The catastrophe mode should have clear, limited goals, implying limited functionality, have <50K lines, and have high assurance. Another issue is dealing with uncertainty. Any "natural" user interface should make assump-

tions. For example, a speech-understanding program will get some unknown or uncertain input that the computer has to approximate. So one way to deal with this may be to build paradigms where distribution is a standard data type and can be parameterized over a domain (like lists).

Peter Honeyman asked the first question. Is it right to attribute the World Wide Web to physicists? Wasn't Mosaic developed by computer scientists? Butler: Could be I oversimplified. Question: Why is distributed computing a failure? Don't we have the Web? Butler: We don't do distributed computing. We do client-server, where only two machines are talking to each other. Grid? I don't understand it. Margo Seltzer: IBAL is a language that supports probability as a fundamental datatype. I encourage everyone to try it. Margo Seltzer: It looks like catastrophe code is similar to recovery code as it is never run. Butler: Catastrophe code should be a subset of normal code and shouldn't be used only in catastrophes. Marc Chiarini: Is AI a success or a failure? Butler: Yes, it is successful. When it is successful, it's spun off, for example, computer vision. AI continues to be a success and continues to be a mess. Question: Are not large-scale bank computer crashes computer-only catastrophes? Butler: Not true; although it will inconvenience a lot of people, there is enormous redundancy that will get things back to normal. Question: RISC is a success, since most game systems run on it. Butler: There has not been a successful RISC system since then.

PLENARY SESSION

■ *Why Mr. Incredible and Buzz Lightyear Need Better Tools: Pixar and Software Development*

Greg Brandeau, Vice President of Technology, Pixar Animation Studios

Summarized by Scott Michael Koch

The talk began by explaining the process involved in creating a movie at Pixar. Using examples from their latest movie, *Cars*, and past movies such as *Monsters Inc.*, he explained the key steps involved in turning an idea for a movie portrayed in storyboard drawings into a detailed, computer-rendered movie. Although all of Pixar's movies are essentially cartoons, the company feels it is important for its movies to contain lifelike effects. Special attention is paid to detail when creating the environments in which their stories take place, by taking into account effects such as weather and fire. When appropriate, Pixar tries to avoid characters appearing to have a plastic texture by giving them fur or other more detailed textures. The next part of the talk began with the showing of a trailer for *Cars* and an explanation of how it compared to some of the movies Pixar had done in the past. Their movies typically take three to five years to complete, and although *Cars* took about the same amount of time to complete as their earlier *Toy Story*, it required 300 times the computing power. He explained the basics of various lighting effects, such as irradiance, ambient occlusion, and reflection, that were used to improve the realistic characteristics of the cars. He also showed a demo of an in-house tool called the Cars Driving System that simulated the movement and interaction of the cars with their environment, so that the animators did not have to worry about underlying car-

toon physics such as the car's suspension, steering, and turning.

He talked about some of the challenges they encountered in creating *Cars*, as well as some of Pixar's other movies. Besides the basic process mentioned here, each movie is custom-made. Each has a different director, environment, characters, and technology. Using a technique called Reyes Rendering, the memory space of a 32-bit architecture machine was not enough, so the company were forced to switch to 64-bit machines when rendering *Cars*. In fact, a single car required more than 2 GB of memory. Overall, it required 2.4 CPU millennium to render the movie.

Along with using commercial applications, third-party libraries, and other in-house applications, the majority of Pixar's work is done with a more than 2-million-line in-house application that has been developed over the past 20 years. The application is written in a number of different languages including C++, C, Python, Perl, and sh. The application is constantly being customized to meet the ever-changing needs of the current movie. To take advantage of the best tools at any given time, Pixar feels it is important to keep their software as cross-platform as possible.

A perceived major problem is that Linux/OSS development has not kept up with the innovation of hardware. Having had mixed results with gdb and purify, they felt there needed to be a better debugging utility geared toward larger applications. Using current debugger solutions, a process that usually takes several hours turns into a weekend-long process when run under a debugging environment. They would like OSS developers to

design software with large production applications, long run times, OpenGL, and 64-bit technology in mind. They also would like to see a Visual Studio-type IDE for Linux. They also mentioned wanting vendors to provide a sitewide license for software, to make management of licenses for a large number of machines less complicated.

The talk got a mixed response as far as audience questions were concerned. Several attendees from other large companies attested to the fact that the problems and challenges mentioned were not exclusive to Pixar. Others questioned Pixar's contributions back to the open source community. Although they are active in submitting bug reports and patches to projects they use, some thought that they need to be the ones taking the initiative to start solving these problems in the community, and others will join them if they see the project to be worthwhile. There were also suggestions about making all or portions of Pixar code open source in various ways, but the company does not feel that would be appropriate for their type of software. There were also a few suggestion about using Solaris's dtrace, which is something they are considering.

WIDE AREA DISTRIBUTED SYSTEMS

Summarized by Wei Huang

■ *Service Placement in a Shared Wide-Area Platform*

David Oppenheimer, University of California, San Diego; Brent Chun, Arched Rock Corporation; David Patterson, University of California, Berkeley; Alex C. Snoeren and Amin Vahdat, University of California, San Diego

David Oppenheimer's talk tried to answer one question: Can intelligent service placement be

useful on a shared wide-area platform such as PlanetLab? At the beginning of his talk, David laid out five perspectives, from which they will analyze the resource characteristics of shared wide-area platforms and try to answer this question: the variability in resource competitions across nodes; the variability in resource demands across slivers (allocated resources on a single node for an application); how random placement behaves; how the quality of initial resource mappings decay over time; and whether resource competition can be predicted.

David presented their studies on these five aspects from a six-month trace of node-, network-, and application-level measurements of PlanetLab. They found out that CPU and network resource usages are highly variable across the nodes. And the resource demands across instances of applications also varied widely. These trends suggested that an intelligent service placement will benefit applications. This was demonstrated by David's simulation results on running OpenDHT, Coral, and CoDeeN, which showed that there were more slivers satisfying application resource requirements by using intelligent service placement. David also pointed out that node placement decisions can be ill-suited after about 30 minutes, which suggested that migration may help applications if the cost is acceptable. David also indicated that a node's CPU and bandwidth usage can be predicted by its utilization of that resource recently, which implies that a migration service need not require high measurement update rate. However, David said that they found no daily or weekly periodicity on resource utilization.

■ *Replay Debugging for Distributed Applications*

Dennis Geels, Gautam Altekar, Scott Shenker, and Ion Stoica, University of California, Berkeley

Awarded Best Paper!

Dennis Geels presented liblog, a debugging tool for distributed applications. Dennis started his talk with challenges of the debugging process in distributed applications. Many errors are due to race conditions and usually are impossible to reproduce locally. Because of this "limited visibility," testing or simulation is usually not sufficient to reproduce and catch the errors. The current state-of-the-art technique for debugging is still to use the print statement. However, once the software is deployed, this technique requires that the developer choose to expose the affected internal state before the fault manifests.

To address the difficulties of debugging distributed applications, Dennis proposed liblog, which provides lightweight logging and deterministic replay, is transparent to applications, and requires no patch to kernels. It intercepts all libc calls and logs all sending/incoming messages. Each message is associated with a lamport clock so that it can be used later for deterministic replay. Dennis discussed several key challenges and design choices of liblog. He talked about how to deal with concurrent threads, where deterministic replay was harder owing to the lack of kernel support. He also mentioned how to do user-level annotation for TCP traffic and using liblog in a mixed environment of logging and nonlogging processes.

In the Q&A session, when asked whether there are any success/failure cases,, Dennis briefly mentioned their experience using liblog on I3/Chord and

OCALA proxy. He said that liblog helped to find errors caused by broken assumptions about network or coding errors.

■ *Loose Synchronization for Large-Scale Networked Systems*

Jeannie Albrecht, Christopher Tuttle, Alex C. Snoeren, and Amin Vahdat, University of California, San Diego

Jeannie Albrecht started by addressing the inadequacy of current barrier semantics in large-scale distributed heterogeneous computing environments. She argued that the current barrier semantics is too strict to be effective for emerging applications. For example, network links may be unreliable and machines may become unresponsive. A traditional barrier may lead to the situation where progress is limited by the slowest participant or where one must wait for an indefinite time for failed hosts.

Jeannie proposed several possible relaxations of strict barrier synchronization (or partial barrier), which are designed to enhance liveness in loosely coupled networked systems. She proposed two partial barrier semantics: early entry, which allows nodes to pass through without waiting for certain slow participants, to prevent a few nodes from slowing down the whole process; and throttle release, which releases the barrier participants within a certain interval, to avoid resource overload by preventing all processes from simultaneously coming into the critical section. Jeannie also talked about several heuristics to dynamically choose the parameters used in partial barriers, such as detecting the knee of the curve (at which point the arrivals are considered to be slow) and finding the optimal capacity of the critical section.

Jeannie presented their experience in adapting wide-area services by using partial barriers for synchronization. She showed promising results. For instance, using a semaphore barrier (a special variation of a throttle barrier) to perform admission control for parallel software installation in Plush enabled an overall completion rate close to the optimal value achievable by manual tuning.

Jeannie was asked about the flexibility of their partial barrier schemes. She answered that the schemes should be very flexible, since applications receive callbacks when the events happen, (i.e., when some nodes are detected to be slow). The applications still have control of the progress and thus have the flexibility to make the best decisions.

INVITED TALK

■ *Routing Without Tears, Bridging Without Danger*

Radia Perlman, Sun Microsystems Laboratories

Summarized by Chris Small

Bridges, being at some level simpler than routers (at least, requiring less configuration), are often thought to have come first. Actually, routers came first, bridges came later. And it's a myth that bridges are simpler:

Layer 1 relay = repeater
Layer 2 relay = bridge
Layer 3 relay = router

Wait, this doesn't make sense: layer 2 is defined as neighbor-to-neighbor.

We'll see why this makes sense later. Ethernet is a misnomer: It's not a network, it's a multi-access link. Layer 2 is flat, no topology. If we need to connect two networks of machines connected using a layer 2 protocol, we have no topology information. (If they

were connected with layer 3, we could use a router.) So we use bridges. (Switches came from a different direction, but they ended up being the same thing as bridges through parallel evolution.)

The basic idea is that bridges listen promiscuously, learn who is on each side of the bridge, and only forward packets between the two networks as appropriate. But loops (cycles) are a disaster, since layer 2 has neither hop counts nor topology, so you get exponential proliferation of packets. (See Radia's book for the detailed story.)

Let's compute a spanning tree (i.e., subset the graph to make a tree and do not include cycles), only transmit along the links that are in the spanning tree, and save the other links for backups.

On bridges, the spanning tree algorithm turns links on when it thinks the primary links are dead—so if you drop packets, the spanning tree gets turned back into a general graph. On routers, if you drop packets, the link gets shut down. Now that everyone has converged on IP (i.e., everybody is using the same layer 3 protocol), why use bridges at all? Why not routers? Well, bridges are simpler to configure—self-configuring, even.

With link state routing, you discover who you are connected to and broadcast this to your neighbors. Everybody collects this information and forwards it on. Eventually everybody has full information about the entire network.

There is a solution to the bridge/spanning tree problem, called Rbridges. They can replace bridges and are safer. Basically they are bridges that gather global link state information. Each RBridge builds its own

spanning tree, which is more robust.

Then each RBridge encapsulates packets to tunnel across the network to other Rbridges. Add a layer 2 header at each RBridge, with destination address set to the last RBridge.

To fit this into MPLS, we needed to map a 6-byte MAC addr into 19 bits. The trick is to use a nickname (mapping 6-byte MAC addrs to 19-bit nicknames).

Q. What about overflowing maximum packet size?

A. This turns out not to be a problem in practice. The original max packet size was set because the first Ethernet had a very limited amount of RAM, so the max packet size was set where it is. Everybody can handle larger packets these days.

NETWORK AND OPERATING SYSTEM SUPPORT

Summarized by Aameek Singh

■ *System- and Application-level Support for Runtime Hardware Reconfiguration on SoC Platforms*

D. Syrivelis and S. Lalis, University of Thessaly, Hellas

Dimitris Syrivelis presented this paper describing an approach to enable programs running on a reconfigurable System-on-Chip (SoC) to modify the underlying Field-Programmable Gate Array (FPGA) behavior at runtime. Accomplishing this requires support from both the underlying system and the application running on it. The reconfiguration is achieved using a quick suspend-resume mechanism, in which the FPGA bitstream corresponding to the new hardware layout is stored in external memory; the system saves its current runtime state and initiates its FPGA reprogramming (i.e., the entire FPGA is reprogrammed from

scratch, as opposed to dynamic partial reconfiguration). After the reconfiguration, the system restarts and manages all effects and potential side effects of the operation. Such reconfigurable systems can offer significant advantages over systems that have soft-core CPUs, drivers, or controllers. Changing the runtime characteristics of underlying units can help applications adapt to different requirements and boost overall performance, though a number of issues such as device addressing need to be resolved. The applications interact with the reconfigurable system through a library that issues device addition/removal requests. The paper also discusses two sample applications of a Mandelbrot calculation and an audio signal monitor.

■ **Resilient Connections for SSH and TLS**

Teemu Koponen, Helsinki Institute for Information Technology; Pasi Eronen, Nokia Research Center; Mikko Särelä, Helsinki University of Technology

Teemu Koponen presented this paper, which addresses a common concern of SSH/TLS connections dropping owing to a network outage or travel. The SSH and TLS protocols are extended to provide more resilient connections that can withstand changes in IP addresses and long disconnections. The authors argue that such mobility issues are best handled at a higher session layer as opposed to the data link or network layers, the traditional approaches employed in wireless handover or mobile IP mechanisms. This is especially true in the presence of long disconnection periods and absence of any network infrastructure such as mobile IP home agents. The proposed protocol extensions are made while ensuring that they do not require any network changes or middleware

boxes, can be deployed incrementally, and minimize end-point changes. The extensions use in-band signaling (thus easing deployability), negotiations (for an end-point to agree to use the extension—a support that is already present in SSH and TLS), and authentication of reconnection (to prevent hijacking). One unanswered question is the security analysis of these extensions. The authors believe that the extensions do not introduce any new vulnerabilities, but a formal evaluation has yet to be made.

■ **Structured and Unstructured Overlays under the Microscope: A Measurement-based View of Two P2P Systems That People Use**

Y. Qiao and F. Bustamante, Northwestern University

Fabián E. Bustamante presented a measurement-based study of two file-sharing peer-to-peer systems based on unstructured (Gnutella-based) and structured (Distributed Hash Table [DHT]-based Overnet system) topologies. The unstructured systems do not dictate the topology of the network, and thus are thought to be more resilient to peer churn (peers joining/leaving the network). In contrast, the structured systems offer guaranteed and scalable $O(\log N)$ lookup performance (where N is the number of peers).

Based on observations, the authors conclude that both systems are efficient in handling churn; even the Overnet DHT-based system was surprisingly efficient. Both systems had good performance for exact-match (precisely matching an object) queries of popular objects, but Overnet had almost twice the success rate for querying shared objects. Keyword searching was fast in both systems, and load balancing was better handled by Overnet.

INVITED TALK

■ **An Introduction to Software Radio**

Eric Blossom, Blossom Research; Coordinator and Maintainer of the GNU Radio Project

Summarized by Rik Farrow

Software radio means using code to modulate/demodulate radio signals by using as little hardware as possible. Instead of soldering parts, you change the code that is controlling the software radio, providing extreme flexibility, on-the-fly reconfiguration, the ability to act as multiple radios simultaneously, and a much quicker development cycle. Software radio is currently used by the military, SIGINT, research, and cellular companies. Another potential use would be public safety, where interoperability of radios has been a problem (recall the Katrina disaster relief fiasco).

Blossom introduced some basic concepts required for building any radio transceiver, with the focus on doing this as much in software as possible. Radio waves range from kilohertz into the gigahertz frequencies. To properly digitize any signal, you must sample it according to Nyquist's rule, at least twice the bandwidth. That sampling is done in hardware using analog to digital converters (ADC), sampling rates as high as 6 GHz, and sample sizes ranging from 8 to 24 bits. Think about that for a moment. If you sample at 6 GHz and 16 bits, we are talking about 12 GB/s. You won't be doing this on your desktop system soon. But researchers have recorded HDTV signals and stored them to disk, requiring a disk storage capacity of 40 MB/s.

Not all radio requires such high sampling rates (for example, FM radio), and there are projects at GNU Radio (www.gnu.org/software/gnuradio) for an FM

receiver and a 1 Mb/s data transceiver. Blossom said that you can buy hardware today that includes four ADC pairs, an FPLA (for onboard computations, programmed using GNU radio), up to four daughter cards that include analog parts for filtering signals from antennas, and a connection via USB to your desktop system. Using this setup and a 2x2 phased array antenna 1.5 m on a side, Blossom has created software that can track aircraft using the signal from an FM radio station antenna on a mountaintop near his home in Nevada. There are regulatory issues when building your own radio transmitters, but these can be dealt with by using certain frequencies and signal strengths. Blossom called the FCC a bunch of politicians, lawyers, economists, and engineers who regulate bandwidth as if radio were stuck in the 1920s. A single VHF TV channel wastes 6 MHz of bandwidth, for example, and compulsory channels use more than all the bandwidth used by cellular channels. Creative use of software radios would make much better use of bandwidth without causing interference with other forms of radio communication.

GNU Radio uses data flow abstractions, event-based overlay, message queues, and messages, all written in a hybrid of C++ and Python. The software is free and the hardware now costs under \$1,000 (www.ettus.com). You can learn more at <http://comsec.com/wiki>.

CLOSING SESSION

■ Real Operating Systems for Real-time Motion Control

Trevor Blackwell, CTO, Anybots

Summarized by Marc Chiarini

Trevor Blackwell gave an interesting and entertaining presenta-

tion about his experiences building robots and other devices that are controlled by humans in real time. Some useful areas for these include performing dangerous tasks (bomb squad and underwater salvage), avoiding long-term travel (Mars rover), and, perhaps somewhat controversially, supplying efficient on-demand manual labor (e.g., someone half a world away does your household chores, much to the delight of homebodies and eight-year-olds!).

The bulk of the talk comprised four parts: fundamentals of robotics and control, software platforms and components, levels of abstraction for controlling robotic devices, and a discussion of his construction of self-balancing motorized vehicles. For the first part, Blackwell quickly took the audience through a primer on a spectrum of computing components and sensors, from large to small, that serve different purposes and are placed on different sections of robots. The joints on humanoid robots are primarily pneumatic and are actuated by software-controlled proportional valves. Human control of the robots he builds, such as those for experimenting with bipedal motion, utilize common sensors in gloves and cameras that provide constant feedback on hand and/or arm position. Several videos comically demonstrated the difficulty of real-time robot control, particularly when lag was involved (even human sensory lag).

Blackwell moved on to show a breakdown of his component-based heterogeneous infrastructure for robotic experimentation. He starts with a rack of BSD servers responsible for performing complex motion vector and other computations (mostly programmed in Python). These are connected via wired (or wireless) TCP/IP to embedded CPUs run-

ning UNIX and mounted at various points on the robots. The CPUs communicate with a set of microcontrollers that drive the actual hardware, valves, etc. By tinkering with several parameters in the embedded FreeBSD kernel, it is possible to achieve millisecond response times when coordinating controllers. Timing is very important to this task, because even a slight lag in actuation can result in, for example, a robot losing balance, running into a wall, or crushing an object. This led naturally into a discussion about the levels of abstraction for motion control: actuator, position control, position control with feedback, high-level, and fully autonomous. A graph gave the audience a good idea of what could be effectively controlled by a human (given human tolerances) or software at each level: Using just actuators and position control, a device at the level of a Roomba vacuum cleaner is achievable; with feedback, unmanned aircraft or an arm on wheels can be controlled; high-level computations might permit reasonable bipedal motion, but a fully humanoid robot would require a high degree of autonomous control. There are, of course, cracks in this picture and not every device falls neatly into a single category.

The last part of the presentation focused on Blackwell's originating hobby of building self-balancing vehicles such as his Eunicycle and Segway-like scooter. Most of it turns out not to be rocket science, but it still requires a reasonable knowledge of mechanical engineering and classical physics. There were several poignant questions asked during the Q&A: Is building these things an affordable endeavor for hobbyists? Scooters and such are definitely reasonable. Humanoid robotics, especially smaller projects, are quick-

ly becoming an option. Why didn't Blackwell incorporate force-feedback in his projects? Latency is a significant stumbling block, especially for fine control. Why did Blackwell ignore other biologically inspired nonhuman robot designs? The response was that he was most interested in robots that could do tasks designed for people in environments designed for humanoids. See <http://anybots.com> and <http://tlb.org/scooter.html> for further details.

2nd Workshop on Steps to Reducing Unwanted Traffic on the Internet (SRUTI '06)

July 7, San Jose, CA

Thanks to sponsorship by AT&T, Google, and Microsoft Research

Summarized by Anirudh Ramachandran and John Bethencourt and edited by Balachander Krishnamurthy

Rob Thomas of Team Cymru began the workshop with a scintillating keynote address on the underground economy. Although much of the research community working on unwanted traffic issues has focused on technical aspects of various subproblems, Rob brought his direct experience with ongoing study of the underground economy dominated by the criminal elements trading in credit cards, passwords, and the like. He painted a grim picture of the underground economy and stressed the need for a closer examination of activities common in that world that are largely unknown to the research community.

The talk was laden with various anecdotes and examples chosen from actual IRC sessions, but the thrust was to convey a sense of the breadth of the activities across the financial underpin-

nings of the worldwide economy, which is increasingly dependent on the Internet. Periodically, he delved into details of some of the attacks, motivations of the criminals behind them, and their varying technical expertise. Participants often peddle compromised hosts in more valuable domains (such as .mil and .gov), all the way up to gigabit backbone routers. The information gathered from these hosts, such as bank or credit card details and entire identities (Social Security cards, birth certificates, and visa information), are traded.

The offers and exchanges are performed on robust IRC servers. The servers are often, but not always, hosted in countries with lax cybercrime laws. Honor among thieves is missing but attempts to swindle are met with retributions in the form of attacks or, more often, documentation of the fraud and banishment from the trading community. Team Cymru works closely with many partners to reduce the overall threat level. However, there is significant pessimism, given the extent to which criminal elements with significant profit motives are able to stay ahead in the technical arena, where an overwhelming majority of users are less knowledgeable and thus susceptible to social engineering.

■ *The Rising Tide: DDoS from Defective Designs and Defaults*

Richard Clayton, University of Cambridge, Computer Laboratory

The first technical paper session focused on flooding and Distributed Denial of Service (DDoS) attacks and its mitigation strategies. Richard Clayton discussed flooding arising from defective software and firmware designs. Defective design in software results in unwanted traffic, such as malformed DNS traffic to root nameservers, Network Time Pro-

tol traffic from hard-coded domain names, etc. Flawed designs of components assumed to be secure, such as wireless routers, cause unwanted attack traffic to hosts on the Internet. The author warns of other possible sources of flooding (i.e., not from compromised hosts) resulting from design flaws. He suggests distributing services, out-of-band authorization, education, and economic disincentives as some of the ways to mitigate flaws in design.

■ *Efficient and Secure Source Authentication with Packet Passports*

Xin Liu and Xiaowei Yang, University of California, Irvine; David Wetherall and Thomas Anderson, University of Washington

This paper discusses the design of a packet "passport" system to securely authenticate the source of a packet. The goal is to prevent source address spoofing and help filter unwanted hosts in DDoS attacks. The technique involves generating a Message Authentication Code for each packet over the spoofable fields of a packet header, using pre-shared keys to perform encryption. The paper also addresses complications with such a scheme, including key distribution, preventing replay attacks (using bloom filters), and secure bootstrapping using shim headers piggybacked on ordinary BGP route announcements.

■ *Cookies Along Trust-Boundaries: Accurate and Deployable Flood Protection*

Martin Casado, Stanford University; Aditya Akella, University of Wisconsin, Madison; Pei Cao, Stanford University; Niels Provos, Google; Scott Shenker, University of California, Berkeley, and ICSI

In this paper, the authors propose "flow cookies" for limiting DDoS attacks, building on previous work on capabilities and fil-

tering. The idea is a “hack” to TCP by using the currently unused timestamp field in the header coupled with syn-cookies. It uses cookie middleboxes stationed ahead of susceptible servers to complete TCP handshakes with clients and issue temporary capabilities. The proposal includes the notion of a “trust region”—ISP A trusts ISP B if A is a customer of B—to facilitate broader, incremental deployment of flow cookies along trust boundaries, thus providing benefit for the autonomous systems that choose to deploy it.

In sum, the papers presented in this session addressed both the cause and effect sides of the DDoS problem and proposed novel, incrementally deployable network-based solutions for DDoS mitigation.

The second paper session considered several abuses of resources, along with the difficulties in making accurate measurements of unwanted traffic.

■ **Separating Wheat from the Chaff: A Deployable Approach to Counter Spam**

Youngsang Shin, Minaxi Gupta, and Rob Henderson, Indiana University; Aaron Emigh, Radix Labs

The authors proposed two new techniques for spam filtering aimed at reducing Mail Transfer Agent (MTA) processing load. The first technique—token-based authentication—involves adding a cookie-like token as a new header in outgoing messages. Any replies to such messages will naturally include the token header and may be delivered immediately without further spam filtering efforts. The second technique, history-based prioritization, utilizes various characteristics of an incoming SMTP connection from another MTA that may be considered

before the message content is processed. Upon receiving a new connection, a receiver that maintains a history of past connections can examine whether that server has contacted it before, how long ago, whether it sent spam earlier, etc. Based on this information, some messages may be deemed unlikely to be spam and are delivered immediately. Other messages may be queued up for processor-intensive filtering. An experimental evaluation of seven months of email data in a university environment revealed that this technique provides 90% accuracy in spam identification without processing the message body. Reducing the number of messages that must undergo expensive spam detection algorithms (which may take as long as 10 seconds per message) reduces delivery latency.

■ **Strider Typo-Patrol: Discovery and Analysis of Systematic Typo-Squatting**

Yi-Min Wang and Doug Beck, Microsoft Research, Redmond; Jeffrey Wang, PC-Rethinking.com; Chad Verbowski and Brad Daniels, Microsoft Research, Redmond

The authors examined the practice of registering domains similar (in the sense of string edit distance) to existing, popular domains. Such domains are used in many ways: They may host a page of ads (known as domain parking) or, more maliciously, be used for phishing or distributing malware. A systematic investigation of the problem of typo-squatting as it currently exists on the Internet has been carried out via several automated tools. For each of the most popular 10,000 domains (according to Alexa traffic rankings), many typo-like variations of the domain according to several simple rules are generated. For each typo that resolved in DNS, an HTTP request was sent, and the response was recorded along with

any redirection URLs. For one sort of typo (a missing dot after www), 30% of the generated typo domains were found to be registered. The vast majority of typo-squatting was found to be due to a relatively small number of large-scale domain-parking companies, including Applied Semantics (formerly Oingo), which is currently owned by Google.

■ **Tracking the Role of Adversaries in Measuring Unwanted Traffic**

Mark Allman, ICSI; Paul Barford, University of Wisconsin; Balachander Krishnamurthy and Jia Wang, AT&T Labs—Research

The authors evaluated a number of techniques used in malicious traffic to foil network measurement systems. They pointed out the importance of network monitoring systems such as intrusion detection systems, firewalls, honeypots, and application-level filters in maintaining awareness of malicious traffic. So far, malicious traffic constructed in an intelligent attempt to bypass or disable such monitoring systems has been given little consideration; this paper attempts to fill that gap. A variety of attacks on monitoring systems were discussed, ranging from direct attacks that attempt to compromise or overload the monitoring system itself to methods for avoiding monitoring systems while compromising other hosts. To better understand these attacks and take a principled approach to countermeasures, two metrics for classifying their effects on measurement systems were proposed: consistency and isolation. These metrics were shown to provide a taxonomy dividing the polluting effects of malicious traffic on measurement systems into four groups. The talks of the last paper session concerned detection of botnets used for various purposes

and possibilities for pushback mechanisms that disable or block bot software on the infected host itself.

■ *An Algorithm for Anomaly-Based Botnet Detection*

James R. Binkley and Suresh Singh, Portland State University

The authors reviewed some practical botnet detection techniques at a university. The university's dormitory networks are frequently home to botnets, because of the proliferation of poorly secured Windows PCs. These hosts launch DDoS attacks. Since many botnet programs use IRC for coordination, the authors have found that inspecting layer 7 for anomalous IRC traffic is useful in discovering the botnets. The authors define a "TCP work weight" metric for each host on the network as the ratio of TCP control packets sent to total TCP packets sent. The metric is helpful in revealing whether a particular host is participating in DDoS attacks.

■ *Revealing Botnet Membership Using DNSBL Counter-Intelligence*

Anirudh Ramachandran, Nick Feamster, and David Dagon, Georgia Institute of Technology

In this paper, the authors propose a different technique for detecting botnets that have yet to be used in malicious activities. DNS-based blackhole lists (DNSBL) are currently used to keep track of hosts that relay large amounts of spam and allow MTAs to easily query this information. Botnets that are not yet listed in DNSBL fetch a much larger price in underground markets such as those discussed in the keynote address. Purchasers of such botnets may first check the IPs in DNSBL to verify that the botnet is still useful for sending spam. Two heuristics were

developed to detect such anomalous DNSBL query activity. Running the heuristics against actual DNSBL logs revealed that these query patterns were in fact present and that the hosts queried were later used as spam relays. This discovery provides a method for passive botnet detection before the botnet is used for spam relaying.

■ *Leveraging Good Intentions to Reduce Malicious Network Traffic*

Marianne Shaw, University of Washington

Shaw discussed speculative thoughts on a new strategy for combating the compromised machines in botnets. A large majority of such systems are owned by technically ignorant but well-intentioned users ("grandma"). Some of these users may be willing to allow some sort of backdoor system on their host that will allow external systems to block or disable malware on their machine should it begin producing malicious traffic. Since malware on the compromised system will naturally attempt to disable any such mechanism, it would have to be located outside the control of the host operating system. Proposed were several locations, including the firmware of cable modems, NIC firmware, and inside a VM wrapped around the host operating system. This last possibility has been implemented as a research prototype under the Denali VMM. The system developed provides a mechanism for a host under DDoS attack to return a blocking request to the enforcement mechanism in the VMM. Experiments demonstrated an acceptable, yet significant, impact on the host's network performance.

■ *Panel: Real World Experiences*

Richard Clayton, Sean Donelan, Mark Seiden, Rob Thomas

The workshop ended with a one-hour panel discussion on real-world problems and the social issues involved in efforts to reduce malicious Internet traffic. The panel opened with the question "If you had a magic wand, what one thing would you change?" Responses focused on increased law enforcement efforts directed at the sorts of communities discussed in the keynote. It was acknowledged that it would be difficult to track down and prosecute all (or even most) perpetrators of online fraud, but any efforts that increased the costs and risks of online fraud would help serve as a deterrent. Further discussion centered on the role of user education in improving the security of end hosts and thereby reducing the numbers involved in malicious activities. The general consensus was that user education in security is at worst a hopeless task and at best of limited utility in certain environments. The lack of faith in end-user education led to examining whether more responsibility for compromised hosts could be placed on the service providers. One problem in this approach is the lack of an incentive for service providers to combat problems affecting other networks by cutting off the access of their own paying customers who have compromised hosts. The panel discussion revealed the inherent difficulty of eliminating malicious Internet traffic without reducing the ability of technically naive users to access network services—the same network services we are ultimately trying to protect.

2006 USENIX/ACCURATE Electronic Voting Technology Workshop (EVT '06)

Vancouver, B.C., Canada
August 1, 2006

KEYNOTE ADDRESS

■ Who Won? Statistical Election Fraud Detection and Its Limits

Walter R. Mebane, Jr., Cornell University

Summarized by Sarah P. Everett

Professor Walter Mebane began his work in quantitative political analysis in 2000, after the election fiasco in Florida. He developed methods to detect statistical anomalies in voting like those that occurred owing to the use of the butterfly ballot. After the problems in the 2004 election in Ohio, Mebane expanded his work to use statistical anomalies to detect fraud in elections. He looked at patterns in various states, including Ohio and Florida, where he examined the problems caused by the new electronic voting machines.

Mebane's project required him to develop new statistical methods. To use the previous methods, much information would have been needed about the election. However, using a new tool based on Benford's Law, all one needs to examine anomalies are the actual votes. Mebane has applied this method to the recent Mexico election data.

Many political scientists became interested in the usability of voting ballots and equipment after the Florida butterfly-ballot recount. To examine how the butterfly ballot affected that election, Mebane fit a statistical model to votes in many counties. For example, the method allows a researcher to look at the probability that someone voted for Buchanan. This model takes into

account voter registration by political party and demographics to predict how many votes Buchanan should have gotten in the county. When the model does not approximate the reported votes, it suggests that something abnormal happened in the county. For the presidential race, box plots of studentized residuals can be made and outliers identified. In Mebane's analysis of the 2000 election data, Palm Beach County was approximately 35 standard deviations away from the rest of the data. This means that the county's count was not produced in the same way as those of the rest of the counties in Florida. In another comparison, absentee ballots (which were not butterfly ballots) were compared to election day ballots to see the percentages who voted for Buchanan. Again, Palm Beach County pops out of the data.

In Mebane's publication "The Wrong Man Is President," he looks at votes and overvotes in the 2000 Florida election. Some overvotes reflect confusion on the part of the voter. This confusion can be caused by, for example, ballots that instruct voters to vote on every page, although a race may be split between pages. In this election, two-mark and multiple-mark overvotes were enough to make up the difference between the totals for Bush and Gore. In Florida, many types of punchcards and optical scan ballots were used. One can look at the ratio of allocated ballots to certified vote counts to see the rate of overvoting in different counties and on different technologies. Mebane explored how many of the two-mark and multiple-mark overvotes were errors. He used a method that looked at Senate votes and produced true votes. Without the overvote errors, he found, Gore would have gained over 46,000 votes

and Bush would have gained approximately 11,000 votes. This led to his belief that the wrong man became president, since Bush won by fewer than 600 votes in Florida.

The butterfly ballot is only one example of how voting methods can cause confusion. In California, an arrow paper ballot displays two languages, English and either Chinese or Spanish. In Ohio in the 2004 election, Cuyahoga County used punchcard ballots in which the ballot order of names was rotated. Many precincts voted in the same place, and some voters were given a book that did not line up with their ballot or their ballot was processed through the wrong counting machines. This led to between 1,000 and 2,000 votes for Kerry being lost. Yet another example of problems was the 2004 Broward County ballot. It was clearer than the 2000 Florida butterfly ballot, but too much space was left between the candidate's name and where the voter marked his choice.

For the 2006 election in Mexico, Mebane is using Benford's Law, which examines the frequency of digits, to look at the second digit of vote counts. Results indicate that certain states, such as Mexico and Distrito Federal, show irregularities. The irregularities could be due to votes being swapped or to votes being thrown out as invalid. Mebane used the residual outlier analysis again and found that that one district, Distrito Federal, stood out for its many outliers.

When the residual vote rates by machine type in the Ohio 2004 election data were studied, much higher rates were found with punchcards than with other machine types. The rate of DREs (direct recording electronic voting machines) fell between those of punchcards and optical scan

machines. Overall, the residual vote rates were not high enough to change the election. A comparison of the 2002 gubernatorial votes to the 2004 presidential votes showed that Kerry had higher turnout in areas where he was strong, and the same was true for Bush. The pattern obtained by this analysis suggested no tampering or switching of votes in Ohio.

Mebane also applied the Benford's Law test of second digits to the Florida 2004 election data. He found that in Miami-Dade County, electronic voting machines do not seem to have been problematic.

Mebane has also studied the problem of auditability. It is true that not all electronic machines get equal numbers or kinds of voters. This can be due to reasons such as crowding. From machine logs, you can tell when each vote was cast. Most machines are used throughout the day, but some are used for only a few hours. This means that all machines are not used randomly, as may have been assumed. This is why precincts satisfy Benford's Law but machines do not. Of course, these records depend on the accuracy of machine logs, that is, to know where the machines were, you need a map of the machines' locations on election day.

When Mebane studied the machine allocation problem in Ohio in 2004, he looked at the correlation with ballots cast per voting machine. The number of ballots cast per machine is lower in areas with higher proportions of African-Americans. Although the ballots were longer in those areas, the difference does not explain the discrepancy in number of registered voters per machine, since in areas where

there are higher numbers of African-American voters, polls close later. Mebane's analysis indicates discrimination in the allocation of voting machines.

Mebane uses an assortment of statistical tools that can help assess how voting machines affect voting accuracy, transparency, fraud, etc. Researchers can look at machines in connection with administrative practice and decisions, how they are used in polling places, and how people (both voters and poll workers) respond to machines. Any of these analyses rely on substantial non-quantitative knowledge in addition to the statistics.

Q: You didn't mention exit polls. Do you view them negatively or positively in this?

A: Unless people steal 90% of the votes, exit polls are mostly useless. There was a bias in the 2004 exit polls, a demographic bias and a large sampling error.

Q: About your conclusions: you examined paper and machine ballots?

A: No, I didn't say we should move to paper. I'd recommend optical scan ballots, where the voter gets feedback, the error is reduced, and you have the ballots for recount.

Q: Could you talk more about using Benford's test?

A: It's best to use the second digit, because the test is almost never satisfied by the first digit. If you look at the error rates of when someone means to vote yes and actually votes no and vice versa, then you get this second-digit pattern. If you simulate clumping of votes, you get the second-digit pattern.

USABILITY

Summarized by Aaron Burstein

■ Making Ballot Language Understandable to Voters

*Sharon J. Laskowski, NIST;
Janice (Ginny) Redish, Redish &
Associates, Inc.*

The authors examined more than 100 ballots from all 50 states and the District of Columbia, as well as four DRE voting systems, to determine whether their instructions and, in the case of the DREs, system messages conformed with best practices for writing instructions. These best practices were drawn from disciplines such as cognitive psychology, linguistics, and the study of human-computer interaction. Laskowski reported that most, if not all, ballots she and Redish examined violated some best practices. Laskowski highlighted instances of instructions appearing after voting selections; opaque, legalistic language (e.g., "Choose such candidate as you desire"); and instructions whose meaning was obscured by the use of double negatives ("If that oval is not marked, your vote cannot be counted for the write-in candidate"). In addition, some DREs generated system messages that are unlikely to help voters or poll workers understand what problem the DRE system has detected or how to correct it. In addition to recommending that ballot instructions be phrased in clear, direct language and that voters be warned of the consequences of an action before they have an opportunity to take that action, Laskowski outlined several directions for further research. This research should include gaining a better understanding of how voters read a ballot and determining whether voters understand commonly

used ballot terms, such as “cast a ballot,” “partisan,” “contest,” and “race.”

A workshop participant asked whether voters actually read ballot instructions; Laskowski replied that she did not know, but that if voters do read instructions, they should be as clear as possible. Another participant asked whether election officials should consider using pictures and images, rather than prose, to convey ballot instructions. Laskowski pointed out that the interpretation of images varies widely with cultural background but that some research into this area might be warranted. Finally, Laskowski stated that some of the guidelines developed—excluding ballot layout guidelines—in this work will be incorporated in Version 2 of the Voluntary Voting System Guidelines (VVSG).

■ *A Comparison of Usability Between Voting Methods*

Kristen K. Greene, Michael D. Byrne, and Sarah P. Everett, Rice University

Kristen Greene reported the results of the authors' usability studies, measuring efficiency (ballot completion time), accuracy (error rates), and satisfaction (a subjective response), on three traditional voting methods: the open response ballot, the bubble ballot, and the mechanical lever machine. (An open response ballot provides a pair of parentheses within which a voter marks his or her selection but does not indicate what kind of mark the voter should use, whereas a bubble ballot provides an oval that must be darkened to select a candidate.) This study provides a baseline of traditional voting system usability against which electronic voting systems can be compared. A total of 36 subjects participated: 21 female, 15 male; ages 18–56; 23 Rice University undergraduates and

13 subjects from the general population of Houston, Texas. The ballot consisted of 27 races between fictional candidates. Subjects in the study used each ballot type and voted using varying levels of information about the candidates. Greene reported that the three ballot types were generally equally efficient. The ballot types also did not generate statistically different error rates, but the error rate was rather high: 17% of all ballots contained at least one error. Finally, subjects preferred the bubble ballot to the open ballot or lever machines.

Workshop participants asked several questions about the study's design and the composition of its subjects. Greene stated that subjects from outside Rice were recruited through ads on Craigslist and in the *Houston Chronicle* classified section; the latter subjects displayed an error rate that was significantly higher than the average. The study contained a control for prior voting experience, because elections in the Houston area have previously used punchcard ballots. Finally, in response to questions about differences in voters' incentives in an experimental study versus a real election, Greene acknowledged that voters might take additional care to vote accurately in a real election but noted that neither voters nor test subjects received any tangible incentive to vote accurately. Finally, Greene believes that it is unlikely that governments will devote additional resources to voter conditioning in order to reduce error rates.

■ *The Importance of Usability Testing of Voting Systems*

Paul S. Herrnson, University of Maryland; Richard G. Niemi, University of Rochester; Michael J. Hanmer, Georgetown University; Benjamin B. Bederson, University of Maryland; Frederick G. Conrad and Michael Traugott, University of Michigan

Paul Herrnson reported the results of usability tests he and his co-workers performed on several electronic voting systems: ES&S Model 100 (paper optical scan ballot), Diebold AccuVote-TS (touchscreen machine with smartcard activation), Avante Vote Trakker (touchscreen with a readable paper printout for verification), Zoomable (a touchscreen prototype developed specifically for this study), Hart Intercivic eSlate (electronic display with a mechanical dial and buttons for navigation and selection), and Nedap LibertyVote (full ballot electronic display with membrane buttons to select candidates). This study was restricted to assessing usability and accuracy in order to develop recommendations for those aspects of electronic voting systems. Herrnson devoted most of his presentation to a field election that involved approximately 1,500 voters but noted that his group's study also included an evaluation of the six voting systems by human-computer interaction experts, a laboratory experiment, and field experiments in Florida and Michigan. Subjects in the field studies were recruited from such diverse locations as inner cities, shopping malls, universities, and business offices. They were asked to complete a ballot with 18 races (more than one selection was allowed in some races), 4 ballot questions, and a write-in option. Study participants indicated a fairly high level of satisfaction with all machines,

with some preference for the Diebold system and significant dissatisfaction with the Hart system. Regarding accuracy, Herrnson reported that study participants successfully cast their ballot for the candidate they wanted 97–98% of the time. Study participants reported that the designs of the ES&S system and the Avante system made it difficult to change their selections. Herrnson stated that few voter characteristics influenced satisfaction, while more education and computer experience, lower age, and greater proficiency with English correlated with fewer help requests and greater accuracy. Finally, Herrnson reported that most field study participants ignored the paper verification features of the ES&S and Avante systems and actually reported a lower level of confidence in those systems than in the Diebold and Zoomable systems.

In response to a question from a workshop participant about accessibility testing, Herrnson said that he and his co-workers had intended to study this aspect of voting systems but lost the part of their budget that was allocated for doing so. Herrnson also noted that his team did not have access to scanners for optical ballots; the researchers had to tally those ballots by hand.

TECHNOLOGIES

Summarized by Dan Sandler

■ *Secure Data Export and Auditing Using Data Diodes*

Douglas W. Jones and Tom C. Bowersox, The University of Iowa

In order to be communicated to the public, election results must be moved from secure tabulation facilities to public networks. Current best practices involve convoluted chains of dissimilar and obscure computer networks,

or physically transported USB storage devices. However complex this chain of networks or disk swaps, each link is bidirectional, so unauthorized communication from the public into the secure inner network is possible.

To create a truly secure transmission system, the authors have devised a data diode, a one-way optical communications medium. What distinguishes the data diode from previous similar approaches is its extreme simplicity: it uses no black boxes or even transistors, so its circuits can be understood and directly inspected. Comprehensive documentation describes the purpose of each component and each trace in the system; the authors call upon all designers of ostensibly verifiable components to do likewise.

A question was asked about timing channels; clearly the diode does not hinder these, and our best tool remains scrupulous analysis of source code on the transmitting side (including deep examination of the serial hardware). Any access to real-time clocks is a red flag. Other measures such as a Faraday cage around the entire tabulation room were proposed by the audience. A pointed question called the big picture “hopeless” even if the diode is a localized success. Jones stressed that the focus of this work is specifically to eliminate the air gap in data transmission, a place where jurisdictions currently make very bad mistakes. By solving this problem we force attackers to resort to other, more challenging attacks.

■ *Simple Verifiable Elections*

Josh Benaloh, Microsoft Research

True voter verifiability: My vote and *all* other votes are cast as intended and counted as cast. VVPAT (Voter Verified Paper Audit Trail) in practice comes nowhere near this goal, but mis-

leadingly implies that it does. We can achieve the goal with complex crypto, but can we achieve it in a way that is understandable and usable by typical voters? Obviously, a completely transparent election—for example, votes posted on a public Web site—achieves this goal, but at the cost of secrecy.

A cryptographic voting system that is trustable and secret should be conceptually simple and require no more of voters than current DREs do. Such a system allows voters to cast encrypted ballots and then verify that those encrypted ballots were tallied correctly (e.g., using re-encryption mix nets). When encrypting ballots with potentially untrusted devices, we might use “unstructured auditing,” that is, in advance of the election we might allow some voters to create an arbitrary number of encrypted ballots with a device that might be vulnerable. The voter can then choose either to cast each ballot or take it home to check its encryption. A tiny fraction of voters choosing to undertake this audit should detect even a 1% rate of defective encrypted ballots.

Question: With this system I can verify that my own vote was cast and counted correctly, but not others? Answer: You do not know how others voted, but you can still verify that all others were counted correctly.

■ *Prerendered User Interfaces for Higher-Assurance Electronic Voting*

Ka-Ping Yee, David Wagner, and Marti Hearst, University of California, Berkeley; Steven M. Bellovin, Columbia University

Ping Yee offered a voting machine design in which almost all of the user interface is prerendered long before election day. This design helps jump a number of hurdles facing voting

machine vendors wishing to develop secure systems. The first is accessibility vs. security: making an accessible voting system requires a lot of potentially faulty user-interface code. By prerendering entire ballots we can remove a lot of this UI code from the trusted voting machine, decoupling UI design from security. Anyone could download a prerendered ballot and try it at home, for education or practice or to verify its correctness.

The second issue is that of proprietary code. Vendors would prefer not to disclose code. By reducing the size of the security kernel, vendors can get away with disclosing less. Third, the size of the code base directly affects verification time and complexity; a smaller security kernel is clearly a win here. Finally, vendors worry about the constantly changing requirements for voting machines and the impact on the code base, which must be reverified for each change. The authors argue that a great many of such changes occur in the ballot-design phase of preparing an election, which in their design is removed from the trusted security kernel. The goal is to reduce by an order of magnitude the voting-specific trusted software, with similar or better usability than current systems. The authors' solution consisted of 293 lines of Python and a few libraries.

A member of the audience expressed concern that usability testing isn't being substantially improved by rendering ballots earlier. Ping replied that official usability testing is still essential, but is no longer the last word on the matter, since any constituent is able to download and examine the ballot ahead of time. While it doesn't reproduce the experience of using the voting machine,

publishing ballot pictures does allow anyone to vet the interaction. In response to another question, Ping explained that they don't currently plan to apply his techniques to paper (e.g., opscan) ballots. Another participant suggested that the authors investigate usability studies of QWERTY (used in the prototype for write-ins) with other free text-input mechanisms. Finally, Ping reassured a questioner that candidate rotation, i.e., shuffling, is possible with their system by prerendering all the permutations and including them with the final ballot.

■ *Ballot Casting Assurance*

Ben Adida, MIT; C. Andrew Neff, VoteHere

Ben Adida began by saying that voters will, or should, always have concerns about the correctness of voting machines until we offer them end-to-end, voter-centric verification. Voters should have a reasonable assurance that their votes are cast as intended, counted as cast, and not susceptible to coercion or purchase. This talk addresses the cast-as-intended problem, in which we attempt to safeguard the voter's intent until it reaches the ballot box. VVPAT systems address a portion of the chain-of-custody problem—they allow us to ignore the correctness and correct deployment of the voting machine code—but they do not guarantee that results cannot be modified or that they are stored and transported safely.

In VVPAT terms, Ballot Casting Assurance (BCA) means that ballots are cast as intended and the chain of custody is perfect. Such a system might force the voter to revote until the ballot is verified to be acceptable and then give her an authentic receipt that could later be used as evidence in a challenge of count accuracy.

Invalid receipts would signal the presence of faulty or malicious voting equipment. The MarkPledge and Punchscan systems follow this model. Finally, it is not enough merely to detect errors; we must also supply solid policies for error recovery. The voter's hand-off of the ballot must not be our last opportunity to deal with errors. As David Dill has said, "The difference between using computers for voting and for flying airplanes is that you know when the airplane crashes."

Audience questions prompted discussion of the usability of secure election receipts, especially for large contests. Many options are available to address this particular problem, but only usability testing will tell us for sure which work best for voters. The threat model of the system was clarified: the described systems are intended to detect any malicious software in the voting stack.

POLICY & PRACTICE

Summarized by Ka-Ping Yee

■ *Transparency and Access to Source Code in Electronic Voting*

Joseph Lorenzo Hall, University of California, Berkeley

Transparency and the election process are the foundations of a representative democracy. Hall's definition of "transparency" has four parts: accountability, public oversight, comprehension, and access to the entire process. "Open source" can refer to the open source license or to the development model. Source code can also be disclosed even if the disclosure doesn't include all the components of the official Open Source Definition. Though computer scientists often say that all voting code should simply be made open source, the issue is

more complex than that: it has both positive and negative effects on security and on the market.

Source availability offers several benefits: more people can examine the code; you can build the code yourself and debug it; you can use automated tools to evaluate it. However, software alone is not enough. For a full evaluation you need access to the complete system in its running environment. Some states are starting to require code escrow and disclosure. Open source also brings risks. It exposes vulnerabilities to the public, and it would require a process for handling flaws discovered just before an election.

Barriers to disclosing source code for voting technology include: (1) regulations require system recertification whenever code changes; (2) certification and contractual performance bonds are expensive; and (3) to field a product, you need more than just code development. It remains an open question how we can level the playing field for open source or disclosed source. As an incentive, the government might offer a prize in a Grand Challenge to develop an open source voting system, subject to some requirements. It may be very difficult for vendors to move to a disclosed source regime, because their code wasn't designed to be exposed; it may contain patented work or work improperly copied from other sources, for example.

Question: If you designed your system not thinking about it being opened, what will you do when it finally leaks? Even when there are strong controls on source code access, it seems often to be leaked or reverse-engineered. Answer: Maybe we need to put vendors on notice that you need to design your code as if you have nothing to

hide. Maybe it's time to start now. An audience member commented that with regard to foundations for transparency in a representative democracy, we might look at Arrow's impossibility theorem: in order to verify the conditions of the theorem, such as that the decision is not imposed or that the decision responds positively to changes in individual preference, you would need transparency. Another participant commented that Arrow's theorem is about the process of vote tallying, but you could disclose the tallying software without disclosing the vote selection software.

Question: What do you think would happen if federal legislation immediately mandated software disclosure? Answer: Because vendors compete on razor-thin margins, you may see an exodus. But some vendors are more confident about the quality of their code than others. I'm not really sure what would happen.

■ *A Critical Analysis of the Council of Europe Recommendations on E-Voting*
Margaret McGaley and J. Paul Gibson,
NUI Maynooth, Ireland

McGaley explained that the Council of Europe, CoE, is an organization of 46 member states and is not directly connected with the EU. In 2003 the CoE created a committee to develop legal, operational, and technical standards for electronic voting. E-voting was first deployed in Europe in 1982 (in the Netherlands) and then in 1991 (in Belgium) and has since been tested in the U.K., Italy, Spain, and Ireland. The U.K. and Ireland are pulling back from their more ambitious plans for various reasons, including some detected fraud in postal voting.

The U.S. standards effort is older. The first FEC (Federal Election Commission) standards were

produced in 1990, whereas the CoE document is only two years old. The U.S. standards are nominally voluntary but in many states are legally required. In Europe, only Belgium appears to be using the CoE standards, which are shorter and less detailed than the FEC standards.

The authors evaluated the CoE standards from a software engineering perspective: they examined consistency, completeness, scope, over/underspecification, redundancy, maintainability, and extensibility. Many problems were uncovered: Some of the standards are vague, ill-defined, or nonsensical, although it is conceivable that better systems might fail to meet these standards while worse systems might pass.

The authors propose a restructuring of the standards, categorizing them according to the five basic rights identified in the original standard: that they ensure universal, equal, free, secret, and direct suffrage. Organizing the standards in this fashion prevents inconsistency and redundancy, maximizes coverage, and makes them easier to understand and use. In their proposed restructuring, some of the standards are merged, some are revised or omitted, and some additional standards have been added.

Question: You mentioned bug-tracking software in your proposed standard. Were you thinking about soliciting comments during the use of a voting system and incorporating the changes during a further development process? Answer: What we had in mind is that each bug would have an identifier and would be traceable as to how it was resolved or not resolved. The system purchased by the Irish government didn't have any sort of bug-tracking system, so after a

problem was reported, it was hard to trace. Question: Is anybody at the CoE listening to your recommendations? Answer: They are: one of the members read our paper and was very interested in it. Question: Does the CoE ever solicit input from nonmember nations or international organizations? Answer: Yes. In fact, Canada is a regular participant!

■ *An Examination of Vote Verification Technologies: Findings and Experiences from the Maryland Study*

Alan T. Sherman, Aryya Gangopadhyay, Stephen H. Holden, George Karabatis, A. Gunes Koru, Chris M. Law, Donald F. Norris, John Pinkston, Andrew Sears, and Dongsong Zhang, University of Maryland, Baltimore County

Sherman explained what his group found when they evaluated four vote verification products: a Diebold VVPAT, an MIT audio system developed by Ted Selker, a software system called Scytl Pnyx.DRE, and the VoteHere system based on cryptographic receipts. By 2007 Maryland will have spent \$96 million on Diebold systems. The authors believe that governments should

spend some fraction—even if only 2 percent—of that money on voting system research. Their study looked only at how vote verification products worked with the Diebold voting system, not at whether the voting system as a whole is secure. Adding verification to the system would be challenging, since it would add complexity and would require that Diebold revise their software.

The authors evaluated each of the verification products in terms of reliability, functional completeness, accessibility, data management, election integrity, implementation difficulty, and impact on voters and procedures. Each product could probably improve the situation somewhat, but none is a fully ready product. For example, the Diebold VVPAT can't be used by blind voters, and the MIT-Selker audio system can't be used by deaf voters. Also, integration with the DRE machine can be complicated; indeed, the Scytl Pnyx.DRE system can cause the DRE to fail. The VoteHere cryptographic system provides strong election integrity and is imple-

mented in high-quality open source software. However, it may be more difficult for the user. Parallel testing, a powerful technique, was found it to be in some ways better than these vote verification products.

Question: I don't share your confidence in parallel testing. It doesn't seem particularly difficult for malware to beat parallel testing, even if it's conducted fairly carefully. Answer: The easiest way to subvert parallel testing is to load the wrong software onto all the machines and then signal the machines that are being tested to operate correctly. I don't mean to imply that parallel testing is perfect, but I do believe it meaningfully raises the bar by addressing the thread of systemic failure. Question: I'm surprised you rated Scytl higher in terms of election integrity than VVPAT. Could you elaborate on why? Answer: Scytl uses cryptography to protect the information in more places, as compared to the chain of custody issues of a VVPAT.



2007 USENIX Annual Technical Conference Refereed Papers Track

Sponsored by USENIX, The Advanced Computing Systems Association

<http://www.usenix.org/usenix07>

June 17–22, 2007

Santa Clara, California, USA

Important Dates

Submissions due: *Tuesday, January 9, 2007, 11:59 p.m.*

PST (hard deadline)

Notification to authors: *Monday, March 19, 2007*

Final papers due: *Tuesday, April 24, 2007*

Program Committee

Program Co-Chairs

Jeff Chase, *Duke University*

Srinivasan Seshan, *Carnegie Mellon University*

Program Committee

Atul Adya, *Microsoft Research*

Matt Blaze, *University of Pennsylvania*

George Candea, *EPFL*

Miguel Castro, *Microsoft Research, Cambridge*

Fay Chang, *Google*

Nick Feamster, *Georgia Institute of Technology*

Marc Fiuczynski, *Princeton University/PlanetLab*

Terence Kelly, *Hewlett-Packard Labs*

Eddie Kohler, *University of California, Los Angeles,
and Mazu Networks*

Z. Morley Mao, *University of Michigan*

Erich Nahum, *IBM T.J. Watson Research Center*

Jason Nieh, *Columbia University*

Brian Noble, *University of Michigan*

Timothy Roscoe, *Intel Research, Berkeley*

Emin Gün Sirer, *Cornell University*

Mike Swift, *University of Wisconsin, Madison*

Renu Tewari, *IBM Almaden Research Center*

Win Treese, *SiCortex, Inc.*

Andrew Warfield, *Cambridge University and
XenSource*

Matt Welsh, *Harvard University*

Yuanyuan Zhou, *University of Illinois at Urbana-
Champaign*

Overview

Authors are invited to submit original and innovative papers to the Refereed Papers Track of the 2007 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.

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
- ◆ System and network management
- ◆ 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 the USENIX Compendium of Best Papers (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. PST, Tuesday, January 9, 2007. *This is a hard deadline; no extensions will be given.*

All submissions for USENIX Annual Tech '07 will be electronic, in PDF format, through the conference Web site. Annual Tech '07 will accept two types of papers:

- ◆ **Regular Full 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 single-blind. Papers longer than 14 pages will not be reviewed.

- ◆ **Short Papers:** Authors may submit short papers that publicize early ideas, convey results that do not require a full-length paper, or advocate new positions. The same formatting guidelines apply, except that short papers are at most 6 pages long. Accepted short papers will be published in the Proceedings and included in the Poster Session, and time will be provided in Short Papers Sessions for brief presentations of these papers.

In addition, the program committee may accept some regular submissions as 6-page short papers if they judge that the submission is interesting but do not accept it as a full-length paper. Please indicate explicitly if you do not wish your regular paper to be considered for acceptance as a short paper.

Specific questions about submissions may be sent to usenix07chairs@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. USENIX, like other scientific and technical conferences and journals, prohibits these practices and may, on the recommendation of a program chair, take action against authors who have committed them. In some cases, program committees may share information about submitted papers with other conference chairs and journal editors to ensure the integrity of papers under consideration. If a violation of these principles is found, sanctions may include, but are not limited to, barring the authors from submitting to or participating in USENIX conferences for a set period, contacting the authors' institutions, and publicizing the details of the case.

Note that the above does not preclude the submission of a regular full paper that overlaps with a previous short paper or workshop paper. However, any submission that derives from an earlier workshop paper must provide a significant new contribution, for example, by providing a more complete evaluation.

Authors uncertain whether their submission meets USENIX's guidelines should contact the program chairs, usenix07chairs@usenix.org, or the USENIX office, submissionspolicy@usenix.org.

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, March 19, 2007. Accepted papers may 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 accepted paper will receive a registration discount of \$200. USENIX will offer a complimentary registration upon request.

Poster Session

The poster session, held in conjunction with a reception, will allow researchers to present recent and ongoing projects. The poster session is an excellent 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.

Birds-of-a-Feather Sessions (BoFs)

Birds-of-a-Feather sessions (BoFs) are informal gatherings organized by attendees interested in a particular topic. BoFs will be held in the evening. BoFs may be scheduled in advance by emailing bofs@usenix.org. BoFs may also be scheduled at the conference.

Training Program

USENIX's highly respected training program offers intensive, immediately applicable tutorials on topics essential to the use, development, and administration of advanced computing systems. Skilled instructors, hands-on experts in their topic areas, present both introductory and advanced tutorials.

To provide the best possible tutorial slate, USENIX continually solicits proposals for new tutorials. If you are interested in presenting a tutorial, contact Dan Klein, Training Program Coordinator, tutorials@usenix.org.

Program and Registration Information

Complete program and registration information will be available in March 2007 on the Annual Tech '07 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/mailling.html>.



Announcement and Call for Papers **USENIX**

HotOS XI

11th Workshop on Hot Topics in Operating Systems

Sponsored by USENIX, The Advanced Computing Systems Association, in cooperation with the IEEE Technical Committee on Operating Systems (TCOS)

<http://www.usenix.org/hotos07>

May 7–9, 2007

Catamaran Resort Hotel, San Diego, CA

Important Dates

Paper submissions due (hard deadline): *January 4, 2007*

Notification to authors: *March 6, 2007*

Final position papers due: *April 10, 2007*

Publication of papers for participants: *April 17, 2007*

Conference Organizers

Program Chair

Galen Hunt, *Microsoft Research*

Program Committee

George Candea, *Ecole Polytechnique Fédérale de Lausanne*

Landon Cox, *Duke University*

Armando Fox, *University of California, Berkeley*

Rebecca Isaacs, *Microsoft Research Cambridge*

Rodrigo Rodrigues, *Instituto Superior Técnico and INESC-ID*

Margo Seltzer, *Harvard University*

Michael Swift, *University of Wisconsin, Madison*

Amin Vahdat, *University of California, San Diego*

David Wetherall, *Intel Research and University of Washington*

John Wilkes, *Hewlett-Packard Labs*

Emmett Witchel, *University of Texas at Austin*

Yuanyuan Zhou, *University of Illinois at Urbana-Champaign*

Overview

The field of Operating Systems research is more relevant today than ever. Massively multi-core architectures, field programmable hardware, virtual machine monitors, safe languages, security, and dependability requirements are all in significant flux while processor speeds have stagnated. Decades-old assumptions about computer architecture and the computing environment are being challenged or changing.

In the context of these dramatic changes, the 11th Workshop on Hot Topics in Operating Systems will bring together people conducting innovative work in the systems area for three days of interaction, with all attendees being active participants and contributors throughout the workshop. Continuing the HotOS tradition, this workshop will be a place to present and discuss new ideas about computer systems and how technological advances and new applications are shaping our computational infrastructure.

We request submissions of position papers that propose new directions of research, advocate nontraditional approaches to old (or new) ideas, or generate insightful discussion. HotOS takes a broad view of what the systems area encompasses and seeks contributions from all fields of systems practice, including operating systems, data storage, networking, security, ubiquitous computing, Web-based systems, tools, and systems management. As a venue for exploring new ideas, HotOS encourages contributions influenced by other fields such as hardware design, networking, economics, social organizations, biological systems, and the impact of compiler developments on systems and vice versa. We particularly look for position papers containing highly original ideas.

To ensure a productive workshop environment, attendance is limited to about 60 participants who are active in the field. We urge practitioners as well as researchers to contribute submissions. Each potential participant should submit a *position paper* of five or fewer pages that exposes a new problem, advocates a new approach to an old idea, or reports on actual experience. Participants will be invited based on the submission's originality, technical merit, topical relevance, and likelihood of leading to insightful technical discussions at the workshop. Multiple authors may share a position paper, but at most two authors per paper will be invited to participate in the workshop.

Preliminary online proceedings will be made available via the Web by April 17, 2007, for workshop participants. Printed proceedings, including a summary of the interactions at the workshop, will be published and mailed to participants after the workshop.

Submitting a Paper

Position papers must be received by 11:59 p.m. Pacific Standard Time, on Thursday, January 4, 2007. **This is a hard deadline—no extensions will be given.**

Submissions must be no longer than 5 pages including figures, tables, and references. Text should be formatted in two columns on 8.5-inch by 11-inch paper using 10 point fonts on 12 point (single-spaced) leading, and 1-inch margins. Author names and affiliations should appear on the title page (reviewing is not blind). Pages should be numbered, and figures and tables should be legible in black and white without requiring magnification. Papers not meeting these criteria will be rejected without review, and no deadline extensions will be granted for reformatting.

Papers must be submitted in PDF format via the Web submission form, which will be available at <http://www.usenix.org/events/hotos07/cfp/>.

All submissions will be acknowledged by January 9, 2007. If your submission is not acknowledged by this date, please contact the program chair promptly at hotos07chair@usenix.org.

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. The proceedings will be published to the Web no earlier than April 10, 2007. Papers accompanied by nondisclosure agreement forms are not acceptable and will be returned to the author(s) unread.

Simultaneous submission of the same work to multiple venues, submission of previously published work, and plagiarism constitute dishonesty or fraud. USENIX, like other scientific and technical conferences and journals, prohibits these practices and may, on the recommendation of a program chair, take action against authors who have committed them. In some cases, program committees may share information about submitted papers with other conference chairs and journal editors to ensure the integrity of papers under consideration. If a violation of these principles is found, sanctions may include, but are not limited to, barring the authors from submitting to or participating in USENIX conferences for a set period, contacting the authors' institutions, and publicizing the details of the case.

Authors uncertain whether their submission meets USENIX's guidelines should contact the program chair, hotos07chair@usenix.org, or the USENIX office, submissionspolicy@usenix.org.

LISA'06

20TH LARGE INSTALLATION
SYSTEM ADMINISTRATION CONFERENCE

A **Blueprint** for Real World **System Administration**

DECEMBER 3-8, 2006 | WASHINGTON, D.C.

6 days of training by experts in their fields

3-day technical program:

- Keynote Address by Cory Doctorow, science fiction writer, co-editor of Boing Boing, and Senior Fellow, USC Annenberg Center for Communication
- Invited Talks by industry leaders
- Refereed Papers, Guru Is In Sessions, and WiPs

Vendor Exhibition

And more!



Registration is open and the full program is available at www.usenix.org/lisa06

Sponsored by

USENIX & [sage]



Early Birds! Register by November 10 and save! www.usenix.org/lisa06



;login

USENIX Association
2560 Ninth Street, Suite 215
Berkeley, CA 94710

POSTMASTER
Send Address Changes to ;login:
2560 Ninth Street, Suite 215
Berkeley, CA 94710

PERIODICALS POSTAGE
PAID
AT BERKELEY, CALIFORNIA
AND ADDITIONAL OFFICES
RIDE ALONG ENCLOSED