;login:

THE MAGAZINE OF USENIX & SAGE
June 2003 • volume 28 • number 3

Well, there's egg and bacon; egg sausage and bacon; egg and spam; egg bacon and spam; egg bacon sausage and spam; spam bacon sausage and spam; spam egg spam spam bacon and spam; spam sausage spam spam bacon spam tomato and spam; spam spam spam egg and spam; spam spam spam spam spam spam baked beans spam spam spam . . .

inside:

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

USENIX Upcoming Events

2003 Linux Kernel Developers Summit
JULY 21–22, 2003
Ottawa, Ontario, Canada
http://www.usenix.org/events/kernel03/

10th Annual Tcl/Tk Conference
Sponsored by Noumena Corporation, in cooperation with ActiveState, USENIX, and IEEE SE Michigan Computer Chapter
JULY 29–AUGUST 2
Ann Arbor, Michigan, USA
http://www.tcl.tk/community/tcl2003/

12th Usenix Security Symposium
AUGUST 4–8, 2003
Washington, DC, USA
http://www.usenix.org/events/sec03/

BSDCon 2003
SEPTEMBER 8–12, 2003
San Mateo, California, USA
http://www.usenix.org/events/bsdcon03/
Camera-ready final papers due: July 8, 2003

5th IEEE Workshop on Mobile Computing Systems & Applications
Sponsored by the IEEE Computer Society, in cooperation with ACM SIGMOBILE and USENIX
OCTOBER 9–10, 2003, Santa Cruz, CA, USA
http://wmcsa2003.stanford.edu

17th Systems Administration Conference (LISA '03)
Sponsored by USENIX and SAGE
OCTOBER 26–31, 2003
San Diego, CA, USA
http://www.usenix.org/events/lisa03
Final Papers due: August 4, 2003

Internet Measurement Conference 2003
Sponsored by ACM SIGCOMM and co-sponsored by USENIX
OCTOBER 27–29, 2003, Miami, FL, USA
http://www.icir.org/vern/imc-2003/

2004 USENIX/ACM Symposium on Networked Systems Design and Implementation (NSDI '04)
Co-sponsored by USENIX, ACM SIGCOMM, and ACM SIGOPS
MARCH 29–31, 2004, SAN FRANCISCO, CALIFORNIA
http://www.usenix.org/events/nsdi04

Third USENIX Conference on File and Storage Technologies (FAST '04)
MARCH 31–APRIL 2, 2004, SAN FRANCISCO, CALIFORNIA
http://www.usenix.org/events/fast04

Third Virtual Machine Research and Technology Symposium (VM '04)
MAY 6–7, 2004, San Jose, CA, USA
http://www.usenix.org/events/vm04
Paper submissions due: October 13, 2003

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

contents

;login: **Vol. 28, #3, June 2003**

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

;login: (ISSN 1044-6397) is published bi-monthly 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 $110 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.

©2003 USENIX Association. USENIX is a registered trademark of the USENIX Association. Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. USENIX acknowledges all trademarks herein. Where those designations appear in this publication and USENIX is aware of a trademark claim, the designations have been printed in caps or initial caps.

Cover: An average email day . . .

motd

by Rob Kolstad

Rob Kolstad is currently Executive Director of SAGE, the System Administrators Guild. Rob has edited *;login:* for over ten years.

kolstad@sage.org

SAGE: Approaching the Crossroads

I've now been in the SAGE directorship position for a year. Here are a few words on some of the interesting challenges I've encountered.

A summary of the job is in order, of course. SAGE's goal is "To advance the profession of system administration." To that end, the professionalization of system administration is often cited as a corollary. Paul Evans started the professionalization movement many years ago when he pointed out that professions have common properties:

- A book of knowledge
- A university degree program
- A certification program of some sort
- Conferences/gatherings
- A code of ethics
- Professional journals/publications
- Recognition programs for outstanding contributors

SAGE is moving down the road to creating all these various properties, which is good. Best of all, public recognition for the career of system administration is high and growing. It appears that high school students all know of such things.

I think there are about 750,000 syadmins in the USA (and an additional substantial fraction of that many throughout the rest of the world). This means that the odds of having one as a neighbor are fairly high. The profession's recognition is making progress.

Some barriers to organizational success, though, have come as a surprise to me.

One of the most jarring recent events was the realization that many of our prospective members have very committed ideas about what a "system administrator" is (and thus what the audience for SAGE is) – and SAGE isn't on their radar.

Often, these people are network administrators, security administrators, or some other sort of administrator who looks upon "system administrators" as "someone else." Sometimes, they view sysadmins as a sort of inferior species, which is also surprising to me. Apparently, specialization has some sort of value that I don't understand very well (and, since I don't work in a large company, I'm going to have to work extra hard to learn more about this particular phenomenon).

At any rate, none of SAGE's messages gets through to these other administrators since they're not tuned to "system administration" information. One email conversation I had was very illuminating in that my correspondent simply could not hear "generic administrator" when filling out a form. If a question did not specifically address "network administrator," then he/she simply could not answer it. This puzzled me greatly.

I subsequently urged many in our community to create an umbrella term that could be used to refer to the collective of all sorts of system administrators. Many great suggestions have emerged, but none of them seems perfect, yet. Often, the problem I'm talking about is misunderstood or denied. This is very strange to me.

Another of the other biggest challenges I'm now facing is that joining SAGE isn't very simple. To that end, I'll be lobbying strongly at the USENIX Board of Directors meeting in San Antonio to create a simple and extremely affordable fee structure for joining SAGE. Wish me luck. My goal: a click on a website and a small amount of money gets you a shiny membership card and all the rights and privileges of SAGE membership.

Our industry's economic slowdown has resulted in reduced SAGE staffing so my personal response time for many issues continues to worsen. I have never been so far behind on email or on my to-do list as I am today.

To that end, I am soliciting volunteers to assist with some specific SAGE projects. Here's the first three:

- Completing the paradigm for creating and maintaining SAGE-affiliated organizations both in the USA and around the world.
- Submitting white papers for the SAGE Web site
- Repairing and implementing various functions on the SAGE site.

These positions are neither "advisory" nor "personnel management". They are roll-your-sleeves-up and make results happen positions. I'll endeavor to insure that obstacles are removed – but the organization really needs a bit more manpower right now. If you're interested, please email me and let me know of your interest.

I think that the field of system administration (and all the sub-fields that, in my mind, it contains: network admin, security admin, database admin, LAN admin, etc.) is not only technically fascinating but one of the highest-leverage fields of endeavor. I'd really like our organization to be the premier technical association for the hundreds of thousands of administrators (of all sorts of systems – SAGE is not parochial about this). If you have suggestions or ideas, please do forward them to me. Finances and timing combine to make 2003 the year that we must all succeed on this task. Any assistance is appreciated!

First Haiku Contest

Let's write some Haiku. You've probably seen Haiku, little three line poems with five syllables in the first line, seven in the second line, and five more in the last line. Syllabic stresses (accents) are not important. Here's a quick one:

Surfing excitement
No response to any site
Sadness rules the night

The brief Haiku captures a thought, a moment, a scene, or a vision. The best Haiku amplify insight or even cause an "Aha!" or epiphany.

Many Haiku traditionally describe the seasons of the year. Being a technical society, this doesn't seem quite so germane to our mutual interests.

This first Haiku contest solicits your entries describing some state of the Internet. Many entries will be posted on the web-site and potentially printed in these pages. The winning entry will be highlighted and rewarded: its author will be win a handsome polo shirt commemorating his/her vision and creative writing ability.

Details: Please submit entries to *haiku@usenix.org* with the subject line "*Haiku — internet*".
Entry deadline: July 1, 2003. No limit on entries (other than pragmatism). Anonymous entries win no physical prizes. Void where prohibited by law.

;login:

EDITORIAL STAFF

EDITOR:
Rob Kolstad *kolstad@usenix.org*

CONTRIBUTING EDITOR:
Tina Darmohray *tmd@usenix.org*

MANAGING EDITOR:
Alain Hénon *ah@usenix.org*

COPY EDITOR:
Steve Gilmartin

TYPESETTER:
Festina Lente

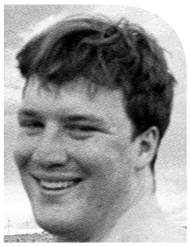
MEMBERSHIP, PUBLICATIONS, AND CONFERENCES
USENIX Association
2560 Ninth Street, Suite 215
Berkeley, CA 94710
Phone: 510 528 8649
FAX: 510 548 5738
Email: *office@usenix.org*
login@usenix.org
conference@usenix.org
WWW: *http://www.usenix.org*
http://www.sage.org

Microsoft: what's next?

by Kragen Sitaker

Kragen Sitaker is a multilingual hacker who's used UNIX since 1992, presently consulting on server-side Web software development in San Francisco. See *http://pobox.com/~kragen/* for more.

kragen@pobox.com



Microsoft just won their years-long antitrust lawsuit: they flouted the law, they perjured themselves with impunity, and they got off with a slap on the wrist.

The time has come for those of us in the free-software community to think about what this means, because now Microsoft considers us Microsoft Enemy #1. What should we expect in the next few years?

I don't think my writing this will help Microsoft much – they've probably already thought this stuff through quite thoroughly – but perhaps it will help the rest of the world.

Well, we can probably kiss Microsoft Office on Linux goodbye. It works now – at least, up to Office 97 – but Microsoft will do everything in their power to ensure that future versions of Office don't run on Linux, or, for that matter, on old versions of Microsoft Windows. In the past, they've licensed some products only for Microsoft operating systems. Antitrust law forbids this, but they might not care – they just laughed in the face of an antitrust case from the world's most powerful government and won.

In any case, they can certainly legally use technical means to make them difficult to run.

For example, they can integrate big chunks of application code into the operating system; running the applications on another operating system would then require that the other operating system include re-implementations of all of this application functionality. Taken to the logical extreme, this would mean including all Microsoft applications with every copy of the OS, only encrypted or disabled in some other way; the application CDs would merely contain activation keys. This would make it harder to upgrade the applications independently of the operating system, but it seems likely that Microsoft can use their "critical update notification tool" to distribute the necessary updates ahead of the application releases.

Strategic GPL applications on Microsoft Windows could become technically very difficult to run, especially when Microsoft can upgrade everybody's operating system to break them on a daily basis. Microsoft, of course, has no legal obligation to verify that their software updates don't break third-party applications.

Along similar lines, Microsoft Windows licensing might forbid linking GPL applications to system libraries, on the grounds that it might imperil Microsoft's intellectual property.

The Windows XP license forbids providing remote access to your desktop and, if I recall correctly, uses various technical means to make this difficult. These technical means won't work when the Microsoft Windows OS runs inside a virtual machine like VMware. So Microsoft could "legitimately" break VMware compatibility, and probably will. (Microsoft can break VMware compatibility easily, especially when they can update their software on a monthly basis.)

Microsoft has filed for a number of strategic patents on the .NET virtual machine. If they get them – which they probably will – they will legally control all .NET deployments, including those built on free software. Microsoft couldn't control GPL implementations of .NET this way – a patent holder can prevent them from being distributed, but cannot impose conditions on such distribution.

Ximian has built a free .NET virtual machine clone called Mono, which they originally licensed under the GPL; they relicensed it under a license that allows patent control and proprietary derivatives. Most likely, Microsoft offered them some quid pro quo for so doing, perhaps licenses to use the patents. If this comes to pass, Ximian and Microsoft will sell licenses for Mono and solutions built on it, but anybody else who does this will have to comply with arbitrary restrictions imposed by Microsoft. For example, Microsoft could require specific virtual machine features to change in a way incompatible with existing GPL application code, or require a per-seat license.

Microsoft Internet Explorer has essentially a complete monopoly on the Web browser market. Microsoft can use this monopoly to encourage use of Microsoft products on the server side in two straightforward ways. First, they can break existing functionality when interoperating with non-Microsoft server software – perhaps an occasional header misparsing, some extra processing delays, or some inefficient code that only runs when talking to non-Microsoft servers. Second, they can add new functionality that only works with Microsoft server software.

Microsoft Internet Explorer has essentially a complete monopoly on the Web browser market.

Palladium offers Microsoft a doomsday weapon against competition. Palladium-enabled software applications can store their files in formats that other software cannot decode and that software running on another operating system also cannot decode, and they can reliably refuse to run on other operating systems themselves.

In the past, Microsoft has exerted great pressure on hardware vendors not to support Microsoft's competition and not to differentiate their products. In the near future, this practice could work to great advantage against competing operating systems. For example, Microsoft can continue to forbid hardware vendors from offering machines configured with multiple boot options; they can give better financial terms to vendors who offer no competing operating systems; they can encourage hardware vendors to use hardware without good Linux support; and they can forbid hardware vendors to offer Linux drivers or give help to developers of Linux drivers.

Finally, many people perceive Linux as more secure than Microsoft Windows; if Microsoft can destroy that perception, they can prevent these people from leaving Microsoft Windows for Linux. For example, they could hire programmers to write better viruses and worms for Linux, and they could talk up Linux worm incidents.

getting the problem statement right

by Dan Geer

Dan Geer is a USENIX Past President and is Chief Technology Officer at @Stake, Inc.

geer@world.std.com

All of us who even occasionally get to be engineers have at one time or another discovered that we were solving the wrong problem. Make that solved the wrong problem. To put it a little brusquely, it's about then we ask ourselves, "What good is the right answer to the wrong question?"

In science, sometimes the right answer to the wrong question means opportune discovery. In commercial engineering, the right answer to the wrong question just means inopportune overtime, opportunity cost, and/or negative rates of return. Science may be about invention or discovery, but the rest of the world is about execution.

Of course, there are a lot of situations where the problem statement is dictated by someone with a vantage point that bears no resemblance to the vantage point of those of us who have to execute. In my own line of work (security), "We need a firewall" almost always means something else, even if you can't discuss what the real problem statement ought to be until after the damned firewall goes live. Surely there are a raft of parallel examples.

Let's take a look at some problem statements that are not so obvious.

Among the techno-geek community, one sees pervasive antipathy to digital rights management (DRM) technology, while one sees just as pervasive affection for privacy enhancing technologies (PET). Emotionally, DRM equals the record industry equals profit enhancement equals badness, while PET equals cryptoanarchy equals self-actualization equals goodness. Yet the problem statement for both DRM and PET is one and the same, viz.: *controlled release of information you own at a distance in space and time*.

One can perhaps argue endlessly over whether you "own" a tune or whether you "own" your bank account number – "endless" is probably what any genuine argument is and ought to be – but if you accept the problem statement, then you have to conclude, at least with respect to the technology itself, that DRM = PET or, in outcome terms, with respect to privacy and digital rights we get both or we get neither. Arguing in favor of one and against the other is to argue for a solution space that is the null set. If you find this distasteful to your worldview, then dispute the problem statement or at least apply yourself to adaptive re-use of the available means to the ends you favor.

Let's try another one.

The intrusion detection paradigm says that you want to know when an attempt is made to cross a network perimeter from the outside of the company to the inside. Is that what you care about? An intrusion is definitionally the illegitimate *acquisition* of legitimate authority. Is that where the risk is? No, the risk is the illegitimate *use* of legitimate authority, which is precisely why just about everyone knowledgeable in security acknowledges that the biggest threats are on the inside. The problem statement for intrusion detection is: *keep dishonest people honest*.

In implementation terms, an intrusion system is about manning a guard station at the network perimeter of the firm. So, by analogy, which do you think is bigger: (A) the sum of US border protection manpower or (B) the sum of all the police departments in-country? Obviously, the correct answer is B, so the real problem statement is: *keep honest people honest*.

That is a high enough goal, trust me. Interdiction of the illegitimate acquisition of legitimate authority, i.e., the effort already put into intrusion systems, ought to grow no more than it already has. Intrusion detection's sunk costs are just that, sunk costs, but that doesn't mean you have to keep sinking more. It is time to put more effort toward the behavioral analysis of surveillance data collected when operation is normal rather than signature analysis of intercepted data when operation is exceptional. Surveillance is consistent with keeping honest people honest even if the surveillance logs are only used forensically.

On the other hand, if you have all those intrusion systems creating masses of logs (that you never read), what can you do with them? The usual intrusion detection problem statement has an explicit subtext: *Never let anyone know that attackers tried to get in, did get in, how it was that they did get in, or what they did while they were in.*

This is just another variant on the most venerably stupid problem statement in all of security: *security through obscurity*.

The correct problem statement is: *threat identification and mitigation*.

It is not: *threat identification and hiding*.

In other words, share your logs with other firms like yours. If you fear debate in the boardroom over sharing this sort of data with selected peers, then here's how to win that debate in a single sentence: Unless you share your intrusion logs with like firms you will not and you cannot ever know whether you are a target of choice or a target of chance, and you will therefore waste needless cash or incur needless risk.

Unless you share your intrusion logs with like firms you will not and you cannot ever know whether you are a target of choice or a target of chance, and you will therefore waste needless cash or incur needless risk.

Let's try yet another one.

This one is harder and it doesn't have a solution until you stir in your own situation data. Many of the readers of *;login:* have run systems at one time or another. Some do nothing but run systems. Let's take just the client side: Nothing is so easy to manage as systems that are, by design, identical and dataless. "Identical and dataless" is the right answer if you can get away with a problem statement (goal) like: *low cost to manage with minimum time to repair (local) failure*.

Of course, identicality means that a local failure can escalate to cascade failure rather easily; think Slammer with its 8.5-second doubling time. Unless you must simulate dumb customers or something like that, a better problem statement would be: *maximize net cumulative productivity*.

That's a problem statement which would naturally lead to operational strategies like: *All applications must be platform independent*; and *No OS can control a majority of platforms*.

You get the idea.

If all this sounds obvious, then great! It is obvious – just as obvious as the Emperor's absence of clothing. It is harder to do day-to-day. It is way too easy to say, "I have a hammer" and to conclude, "Let's find some nails to pound." What's hard is to think big enough to find a problem statement that is at once doable, elegantly simple, and which can be communicated to others who could care less what the hell you are talking about, but who "know" that they want a firewall, who "know" that they want to copy music but don't want you to copy their documents, who "know" that all the bad guys are outside and that that's a secret, and who "know" that if everything were exactly the same on every desktop the company would be better off because that's precisely how you get the best upfront purchase price. A little thought is a dangerous thing . . . especially in problem statements.

.profile

by Travis Howard

Travis Howard is the founder of Asymptotic Systems, a security consulting organization. He has 17 years of experience in computer security and specializes in applying research advances to harden systems.

auto92089@hushmail.com

Of all the dot files, *.profile* presents the greatest opportunity for customization. This file is read by *sh* derivatives as part of the login process, and usually sets environment variables that influence the behavior of many programs invoked as part of that login session. This article describes how to customize *.profile*, but first we should examine how *.profile* is used and what is appropriate to place in it.

Specifically, *sh* derivatives consider a shell a login shell when argv[0] begins with a dash ("-"; ASCII value 45). Note that this is a violation of the convention that the first element of argv[] contain the last component of the executed program's path (for more information see *execve(2)*). This is the only way to flag *sh* as a login shell. However, *ksh* accepts -l and *bash* accepts -login as alternate ways of flagging a shell as a login shell.

All *sh* derivative login shells first process the systemwide /etc/profile if it exists. The next file processed depends on the shell; *sh* and *ksh* process $HOME/.profile, while *bash* processes the first it finds of $HOME/.bash_profile, $HOME/.bash_login, or $HOME/.profile. Note that *bash* has a flag -noprofile, which inhibits processing any of these files.

You may wonder where *sh* derivative login shells get their notion of $HOME. The HOME environment variable is set by *login(1)*, as are SHELL, PATH, TERM, LOGNAME, USER (if BSD), MAIL (if not BSD), and TZ (if Solaris). For portability's sake, we should rely only on the common subset of environment variables set by *login(1)* (if we rely on any of them at all).

Note that *login(1)* will print a variety of messages, unless $HOME/.nologin exists. By the same rationale, we should feel free to output information from *.profile* under the same conditions. The nologin support is primarily for UUCP and other automated logins, so we need not heed this rule too carefully, unless you intend to use your *.profile* for your automated users as well.

Since we have the full power of the shell at our disposal when processing *.profile*, we may easily use its branching constructs to process different parts of this file under different conditions. This allows us to potentially use the same *.profile* in many locations – unlike some other dot-files – simplifying the customization of multiple environments into a single file distribution problem.

Furthermore, we will wish to do some similar and repetitive tasks in our *.profile*, so it will be advisable to use its native function definitions as a macro facility. If we were generating different *.profile* files using some kind of macro language such as m4 or cpp, we could avoid depending on the availability of functions in our shell. However, the environments I am interested in all support functions within /bin/sh, so that is an implicit assumption in my *.profile*. This also allows runtime recursion, which preprocessors could not provide (not that we will need it).

It's extremely difficult to come up with guiding principles and structure for a *.profile*. For one thing, several dependencies must be taken into account when creating a linear order for your statements. There are site-specific, OS-specific, release-specific, and architecture-specific components, and components that are specific to two or more of these categories.

Often it is not clear which category certain statements fall into. For example, BSD UNIX uses BLOCKSIZE to affect the reporting units in df, du, and some other programs. It is not clear whether this is a variable which should be set globally, or perhaps in a BSD-specific portion of the *.profile*.

There are design tradeoffs, such as the desire for an uncluttered environment and the desire to keep the *.profile* simple. Another tradeoff occurs when two OSes have similarities. For example, SunOS and BSD both have /sbin directories for the superuser. Do you add /sbin to the path in both cases, causing code duplication, or do you make a case statement that puts them together, and add /sbin to the path there? Thus, this *.profile* is most definitely a compromise among several competing desires.

Since *.profile* is only processed once per login session, it's tempting to do a little more work in order to make the *.profile* simpler. For example, you might want to invoke Perl and have it tell you where its man pages are located, rather than guessing and testing several possibilities. Similarly, rather than putting /sbin handling into all BSD-like OS-specific sections, it might be cleaner to test for the presence of /sbin and handle, adding it to the PATH no matter which OS you are running. Also note that I usually don't bother to preserve environment variables that are already set via */etc/profile*, although you may wish to do so.

```
# $Id: .profile,v 1.108 2002/11/12 08:12:47 username Exp $
# Hey Emacs this is -*- sh -*-

# This is sourced at login-time by sh, ash, ksh, and bash.
# Assumptions:    login(1) sets HOME SHELL PATH TERM LOGNAME
#                 XDM(1) sets DISPLAY PATH SHELL XAUTHORITY

# Login script order:
# sh, ash, ksh    /etc/profile ~/.profile
# bash            /etc/profile (~/.bash_profile ~/.bash_login ~/.profile)
```

I keep my *.profile* under CVS, so I have an $Id:$ keyword in the first line. I also tell Emacs that this is a shell script in the following line (I believe that more recent versions of Emacs allow mode:sh). This is followed by documentation of assumptions and a little reminder of when this file was evaluated.

Next I give the user some indication that the *.profile* is being run (and send it to the console, not to an output file):

```
## Show login stuff:
# Echo to fd 2 (stderr).
e2 () { echo "$@" >&2; }
e2 'Running .profile'
```

The first thing the *.profile* should do is give an indication that it has been invoked. This will definitely help debug login problems as well as give a visual indicator of how heavy the system load is (if it takes a long time to show this, the load is extremely heavy). This code snippet defines a function (e2), which echoes all of its arguments to the standard error file descriptor (read that operator as "standard output gets tied to fd 2"). There are occasions where standard output is buffered, but standard error is usually line-buffered at most, so that is why I use it. Note the use of double-quoted $@; this incantation preserves whitespace and argument boundaries, even if the arguments include whitespace. It is a good idea to get into the habit of using this instead of the boundary-destroying $*. The code snippet then invokes e2 to display a simple message. Should you wish, you could easily test for the presence of a ~/.nologin file to suppress printing any output.

Next I inform the OS that I wish to make all my file ownerships private:

```
## Set a paranoid umask.
umask 077
```

I want to make sure that this *.profile* can find the programs I want to invoke, so I then must attend to setting the PATH environment variable, which controls the search path for said programs. Clearly, I will want some functions to help me manipulate the colon-separated list of directories:

```
## Set colon-separated search path elements:

# Test a directory (sanity check).
# Returns true (0) only if it is a directory and searchable.
test_directory () {
      test "$#" -eq 0 && e2 "Usage: test_directory dirname" && return 2
      test -d "$1" && test -x "$1"
}

# Check to see if a directory is already in a search path.
in_search_path () {
      test "$#" -lt 2 && e2 "Usage: in_search_path path dirname" && return 2
      local n="$1"
      local d="$2"
      # Save input field separators.
      local OLDIFS="$IFS"
      # Break up on colon boundaries
      IFS=":$IFS"
      # Now set the positional args to elements of the named variable.
      eval set -- $n
      # Restore input field separator.
      IFS="$OLDIFS"
```

```
        # Loop through each colon-separated element.
        while test "$#" -gt 0; do
            # Compare to dirname.
            # TODO: how portable is relying on -ef?
            test "$1" -ef "$d" && return 0
            shift
        done
        return 1
}

# Sanity-check then append a directory to a search path.
dirapp () {
        test "$#" -lt 2 && e2 "Usage: dirapp varname dirname" && return 2
        local n="$1"
        local d="$2"
        test_directory "$d" || return 1
        eval in_search_path \"\$$n\" $d && return 1
        if eval test -n \"\$$n\"; then
            eval $n=\"\$$n:$d\"
        else
            eval $n=\"$d\"
        fi
}

# Sanity-check then prepend a directory to a search path.
# TODO: Allow caller to "move" directory to front with this funcall.
dirpre () {
        test "$#" -lt 2 && e2 "Usage: dirpre varname dirname" && return 2
        local n="$1"
        local d="$2"
        test_directory "$d" || return 2
        # eval in_search_path \"\$$n\" $d && return 1
        if eval test -n \"\$$n\"; then
            eval $n=\"$d:\$$n\"
        else
            eval $n=\"$d\"
        fi
}

# Call dirapp for a list of directories.
dirapplist () {
        test "$#" -lt 2 && e2 "Usage: dirapplist varname d1 d2 ..." && return 2
        local n="$1"
        shift
        while test "$#" -gt 0; do
            dirapp "$n" "$1"
            shift
        done
}

# Call dirpre for a list of directories.
# NOTE: Directories will appear in reverse order in varname.
dirprelist () {
        test "$#" -lt 2 && e2 "Usage: dirapplist varname d1 d2 ..." && return 2
        local n="$1"
        shift
        while test "$#" -gt 0; do
            dirpre "$n" "$1"
            shift
        done
}
```

Note that I will try to use the term *path* to refer to a list of directories, starting with the root and ending with a file or directory, whereas I use the term *search path* to refer to a colon-separated list of paths.

I return 2 to distinguish from the exit code of the last command, which could be 0 or 1.

Next I set up PATH elements which should always be present. I think all reasonable operating systems start PATH with these elements, but it doesn't hurt to be sure:

```
# These should be present on any target system.
# In fact, they should already be in the search path.
dirapplist PATH /bin /usr/bin
# I like to be able to run e.g., ifconfig, sendmail.
dirapplist PATH /sbin /usr/sbin /usr/games /usr/libexec /usr/ccs/bin
dirpre PATH /usr/ucb
export PATH

# Set the search path for manual pages.
dirapplist MANPATH /usr/share/man /usr/share/man/old /usr/contrib/man
export MANPATH
```

Note that I mark these variables as exportable. In general, my policy is to do so the first time I manipulate the variable.

```
## Do shell-specific handling:

# This code distinguishes between various shell versions.
if test "$(echo ~)" != "$HOME"
then
      # This is the standard Bourne shell.
      :
else
      # This is not the Bourne shell, so it supports aliases.
      test -r "$HOME/.kshrc" && test -z "$ENV" && export ENV="$HOME/.kshrc"
      # TODO: Figure out a deterministic way to distinguish shells.
      # Should I just check $SHELL instead?
      if test "${RANDOM:-0}" -eq "${RANDOM:-0}"
      then
          # This is ash, which does not have a type built-in.
          # TODO: Recent versions of ash do indeed have type, so I should test for its presence somehow.
          test -r "$HOME/.ashtype" && . "$HOME/.ashtype"
          # Tell ash where to find our shell functions.
          test_directory "$HOME/functions" && dirapp PATH "$HOME/functions%func"
      fi
fi
```

This code tries to figure out which *sh* variant we are using. It then takes care of the shell-dependent initialization commands. Note that since it uses dirapp, it has to occur after dirapp has been defined, else I would have placed it earlier in the file.

```
## Run this stuff on logout.
if test -r "$HOME/.shlogout"
```

```
then
      trap '. $HOME/.shlogout' 0
else
      # Make a reasonable attempt to clear the screen.
      trap 'clear' 0
fi
```

This code takes care of cleaning up when we log out. If $HOME/.shlogout exists and is readable, we evaluate that and exit with the last command's exit code; otherwise we just clear the screen. That is, this code runs when a login shell exits. If this varied from site to site, it might be more appropriate to put it in the site-specific section.

Clearly, the universal *.profile* will need facilities for examining its environment to determine which parts should be processed in a particular situation. Now that we've set a reasonable PATH (enough to reach the system binaries, anyway), we can try executing some commands to ascertain facts about the platform:

```
## Set the environment variables:

# Try to set the envar called by name in arg1 to the output of the commands that follow, one argument per command.
setvarcmd () {
      test "$#" -lt 2 \
          && e2 "Usage: setvarcmd varname \"cmd1 args\" \"cmd2 args\" ..." \
          && return 2
      local n="$1"
      shift
      while test "$#" -ge 1 && eval test -z \"\$$n\"; do
          eval "$n=\"$($1 2>/dev/null)\""
          shift
      done
      # TODO: what if no commands generate output?
      export $n
}
```

We realize that we will be calling a lot of commands in sequence until we find one that gives us the information we need, which we will then export. Therefore, we enshrine this process in a shell function, setvarcmd. Note that I use the newer $() syntax rather than ''; this allows levels of command expansion to be nested (although we do not do that here). Errors, such as invalid options or nonexistent commands, are directed to /dev/null when executing the commands, because we will be trying several commands in order and don't care if some of them fail.

Next we use setvarcmd to try yo set four critical variables:

```
setvarcmd OS_NAME "uname -s" "uname"
e2 "Operating system: $OS_NAME"

setvarcmd OS_RELEASE "uname -r"
e2 "Release: $OS_RELEASE"

setvarcmd HW_NAME "arch" "uname -m"
e2 "Hardware/Architecture name: $HW_NAME"

# TODO: This frequently does not include the domain name.
setvarcmd HOST_NAME "hostname -f" "uname -n" "hostname"
e2 "Host Name: $HOST_NAME"
```

These variables will be used to decide which portions of the *.profile* to evaluate. There is no error checking here, because if these commands fail to ascertain the desired information, it is not clear what else we could do. However, the user logging in would see that some of the variables were not set, so we can hand-wave here and say the user should investigate and remedy. It may well be that a system which does not readily yield its details may be an unsuitable platform for a universal *.profile*.

Now that OS_NAME is set, I can run some OS-dependent commands to manipulate the environment:

```
# Find some other binary directories, but only for the right architecture.
# Set MAIL to point to mailbox so shell can tell us when we have mail.
# TODO: fix for mailbox in $HOME/mbox.
case "$OS_NAME" in
      AIX)
          dirapp PATH /public/ibm/bin
          ;;
      SunOS*)
          dirapp PATH /public/sun4/bin
          # Find my mailbox and have this shell check it periodically.
          # NOTE: Do not export or subshells will check mail.
          test_directory /usr/spool/mail && MAIL=/usr/spool/mail/$LOGNAME
          ;;
      *BSD)
          test_directory /var/mail && MAIL=/var/mail/$LOGNAME
          ;;
esac
```

The additions to PATH are from one of the sites that I used to work at and are nonstandard. Therefore, should they exist, it's fairly certain it's because this *.profile* is being invoked at that site. Should these directories exist at some other site, just by chance, then maybe I'll want them in my path there, too. If not, I'll have to change my *.profile* accordingly.

The MAIL environment variable tells the shell to alert me between commands if new mail has arrived. If I exported it, subshells, such as the ones created by system(3), or those created by xterms under X, would also alert me, and I don't want to be told more than once.

Next I want to try to set an environment variable to point to the first directory in a list of potential directories:

```
# Set a specified variable to equal the first valid directory in a list.
# TODO: Should I check to see if it is set already?
setvardir() {
      test "$#" -lt 2 && e2 "Usage: setvardir varname dir1 dir2 ..." && return 2
      local n="$1"
      shift
      while test "$#" -gt 0; do
          test_directory "$1" && eval "$n=\"$1\"" && export $n && return 0
          shift
      done
      return 1
}
```

First I use setvardir to find Openwin:

```
# I had to use Openwin on some SunOS machines.
if setvardir OPENWINHOME /usr/openwin; then
      dirapp PATH "$OPENWINHOME/bin"
      dirapp MANPATH "$OPENWINHOME/share/man"
      dirapp LD_LIBRARY_PATH "$OPENWINHOME/lib"
fi
```

Next I try to locate the X Windowing System binaries:

```
# XWINHOME is used by some startx(1), XF86Setup(1), and apparently xman(1), XF86_S3(1), etc.
# Technically, I should only accept /usr/X386 if we are on an x86,
# but what would it be doing there on another architecture anyway?
if setvardir XWINHOME /usr/X11R6 /usr/X386; then
```

```
        # put X executables in search path
        dirapp PATH "$XWINHOME/bin"
        # put X man pages in search path
        dirapp MANPATH "$XWINHOME/man"
fi
```

I then define a short function that I use to find the Perl man pages, given a "root" like /usr/local:

```
# TODO: There has got to be a good way to find the Perl manual pages by querying Perl.
findperlmanpages() {
        local r="$1"
        dirapplist MANPATH    "$r/lib/perl5/man" \
                              "$r/lib/perl/man" \
                              "$r/share/perl5/man" \
                              "$r/share/perl/man"
}
```

I then want to set a variable LOCALIZED to point to where the locally installed programs reside:

```
# Find locally installed programs.
if setvardir LOCALIZED /usr/local /local /lusr /opt; then
        # Prepend locally installed program dir so it can override system binaries.
        dirpre PATH "$LOCALIZED/bin"
        # Search here for manual pages.
        dirpre MANPATH "$LOCALIZED/share/man"
        # BSD systems might have this, others probably will not.
        dirapp PATH "$LOCALIZED/sbin"
        # Some sites insist on per-package bin directories, sigh.
        # NOTE: This could go later in this file, as a site-dependent section,
        # but these directories probably will not exist on most systems.
        for i in tex gnu tk tcl elm expect ghostscript lotus netmake newsprint tk mh; do
            dirapp PATH "$LOCALIZED/$i/bin"
            dirapp MANPATH "$LOCALIZED/$i/man"
        done
        findperlmanpages $LOCALIZED
        dirapp MANPATH "$LOCALIZED/teTeX/man"
        # This is the info path for GNU info hypertext command, and Emacs
        dirapplist INFOPATH   "$LOCALIZED/info" \
                              "$LOCALIZED/share/info" \
                              "$LOCALIZED/teTeX/info"
        # This was required to run Lotus Notes at one site.
        dirapp LD_LIBRARY_PATH "$LOCALIZED/lotus/common/lel/r100/sunspa53"
        # Set the cool Concurrent Version System repository directory.
        setvardir CVSROOT "$LOCALIZED/share/cvsroot"
fi
```

Next I want to set up any user-specific binaries so that I can compile my own copy of a program and have that program be called in place of the system-installed binary of the same name. In other words, this gives the user of this *.profile* the ability to mask system binaries with his/her own copies. We're implicitly creating search paths that start with the user's binaries, followed by the system's (local) binaries and, finally, the OS binaries. One disadvantage of this is that sometimes upgrading the OS will cause the OS binaries to be more up-to-date than the ones closer to the front of the search path. Note that this is done only if the HOME directory is not the root, since /bin is already in the search path. Note also that we allow a user to have binaries that are OS-specific or even release-specific. This is useful in a heterogeneous environment with shared user home directories.

```
# Find my installed programs.
# NOTE: If we boot up in single-user mode, home directory is root.
```

```
if test "$HOME" != "/"; then
      # I have even more control over these so they get prepended.
      dirprelist PATH   "$HOME/bin" "$HOME/bin/$OS_NAME" "$HOME/bin/$OS_NAME/$OS_RELEASE"
      dirpre LD_LIBRARY_PATH "$HOME/lib"
      dirapp MANPATH "$HOME/share/man"
      findperlmanpages $HOME
      dirapp INFOPATH "$HOME/share/info"
      # Set the Pretty Good Privacy filepath (where it finds its files).
      setvardir PGPPATH "$HOME/pgp"
      # Set the cool Concurrent Version System repository directory.
      setvardir CVSROOT "$HOME/share/cvsroot"
fi
```

Note that CVSROOT is being set here again and can thus override the earlier value based on the LOCALIZED dir.

Next I do some variable exporting which wouldn't fit in easily to the above clauses:

```
# There is no convenient place to do this above so do it here.
export INFOPATH
export LD_LIBRARY_PATH
```

Now that we have a complete PATH (among other things), I would like to set certain environment variables that want complete path information. To this end, I create a shell function to find executables:

```
# Echo the full file name of the executable in the path to stdout.
# NOTE: All args are call-by-value.
findinpath () {
      test "$#" -lt 2 && e2 "Usage: findinpath exe_basename path" && return 2
      local f="$1"
      local IFS=":$IFS"
      set -- $2
      while test "$#" -gt 0
      do
          test -x "$1/$f" && echo "$1/$f" && return 0
          shift
      done
      return 1
}
```

This function searches a given path in much the same way that the shell and execlp(3) and execvp(3) do. It echoes the full pathname to standard out. Now that we've got that ability, we set some variables with the pathnames of desired binaries (and a group of standard variables, too).

```
# This is my preferred editor.
EDITOR=$(findinpath vi $PATH)
export EDITOR

# This is the visual, or full-screen editor of choice.
VISUAL="$EDITOR"
export VISUAL

# This is the editor for the fc built-in (for ksh).
FCEDIT="$EDITOR"
export FCEDIT

# EX init file or commands (used in vi(1))
EXINIT="set tabstop=4 showmode"
export EXINIT
```

```
# TODO: is this test sufficient and correct?
test "$OS_NAME" = "NetBSD" && EXINIT="$EXINIT verbose"

# This is my personal CVS working area.
setvardir CVSHOME "$HOME/dev/cvs"

# TEMP is a temporary directory for many programs: cc gcc mailq merge newaliases sendmail rcs (and friends)
# ghostscript i386-mach3-gcc perlbug perldoc
setvardir TEMP /var/tmp

# TMPDIR is a temporary directory for these programs:
# sort
# NOTE: gcc tries TMPDIR, then TMP, then TEMP
setvardir TMPDIR /tmp

# Use the large tmp dir for metamail.
setvardir METAMAIL_TMPDIR /var/tmp

# BLOCKSIZE is the size of the block units used by several commands: df, du, ls
# For more information see NetBSD environ(7).
BLOCKSIZE="1k"
export BLOCKSIZE

# CVS_RSH lets us use ssh instead of rsh for client/server
CVS_RSH=$(findinpath ssh $PATH)
export CVS_RSH
```

The next step is to set the environment variable PAGER to either less or, failing that, more:

```
# Set the pagination program for man, mailers, etc.
if PAGER=$(findinpath less $PATH); then
      # We found less, so set less options:
      #   -M = more verbose than "more"
      #   -f = force special files to be opened
      LESS="-Mf"
      export LESS

      # latin1 selects the ISO 8859/1 character set. latin-1 is the same as ASCII, except characters between 161
      #   and 255 are treated as normal characters.
      LESSCHARSET="latin1"
      export LESSCHARSET
else
      # Every system should have this.
      PAGER=$(findinpath more $PATH)
fi
export PAGER
```

Note that the conditional part of the if statement is true if and only if findinpath returns a true value.

Next comes the site-dependent clauses:

```
# This is the site-dependent section.
case "$HOST_NAME" in
      hostname*|*company.com)
          NNTPSERVER=”news.company.com”
          http_proxy=http://proxy.company.com:8000/
          export http_proxy
          ;;
      otherhostname*|*othercompany.com)
          NNTPSERVER="nntp-server.othercompany.com"
```

```
            ;;
    esac
    export NNTPSERVER
```

I then print a fortune:

```
    ## Show fortune for fun:
    type fortune > /dev/null 2>&1 && fortune -a
```

Then I do some last-minute fixes for certain OSes:

```
    ## OS-dependent fixes:

    case "$OS_NAME" in
        NetBSD*)
            case "$OS_RELEASE" in
                0*|1.0*|1.1)
                    # MANPATH does not work in early releases  of NetBSD
                    unset MANPATH
                    ;;
            esac
            ;;
    esac
```

Next I have the terminal-handling routines:

```
    ## Set up terminal and start X if appropriate.

    # Determine if we started under XDM.
    if test -z "$DISPLAY"; then

        # If tset exists, use it to set up the terminal.
        if type tset > /dev/null 2>&1
        then
            # Set TERM and TERMCAP variables
            if test "$TERM" = "pcvt25h" && test "$OS_NAME" = "NetBSD" && test "$OS_RELEASE" = "1.1A"
            then
                e2 "Skipping tset due to $OS_NAME termcap buffer overflow bug"
            else
                eval $(tset -s -m 'network>9600:?xter' -m 'unknown:?vt100'  -m 'dialup:?vt100')
            fi
        fi

        export LINES
        export COLUMNS

        # This is OS-dependent terminal setup.
        case "$OS_NAME" in
        *BSD*)
            if ispcvt 2> /dev/null
            then
            # 28 lines on screen, HP function keys, 80 columns
            # NOTE: due to bug in scon, it only sets the row/col
            # of ttys if it is done in two commands like so:
            scon -s 28 && scon -H && scon -8
            LINES=25
            COLUMNS=80
            else
            # TODO: Set up wscons.
            :
            fi

                # Start X if we are on PCVT or WSCONS
                case $(tty) in
                    /dev/ttyv*|/dev/ttyE*)
                        sx
                        ;;
                esac
                ;;
            *)
                # Be conservative about screen if not known.
                LINES=24
                COLUMNS=80
                ;;
        esac
fi
```

This rather large block has several functions. First, note that it is only processed if we are *not* starting under XDM (which would set DISPLAY to something). Next, it tests for the presence of tset using the type built-in. If tset exists, it first makes sure it is okay to use tset on this OS. Assuming it is, it tries to figure out what kind of terminal we're on. Note that tset -s actually produces shell commands that must be evaluated. For more information see the tset man page.

After we have set up the terminal, it marks LINES and COLUMNS as exported to child processes. Then we do some OS-specific checks to see what kind of terminal we're on, and if we are on the console, we start X using a shell function sx. Note that we put all the commands related to starting X in a shell function, so we can invoke it from the command line as easily as from *.profile*.

Finally, I end with a true value:

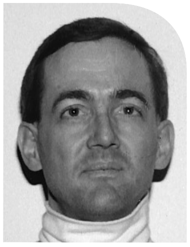
```
# Exit with true value for "make test".
:
```

C# types

by Glen McCluskey

Glen McCluskey is a consultant with 20 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



In our examination thus far of the C# language, we've looked at the overall architecture and also discussed some basics of compiling and executing programs. In this column we'll start to consider the various types that C# offers.

C# is an object-oriented language, and its type system thus centers around system-provided and user-defined classes, classes that represent an abstraction of some sort (such as a calendar date or a geometric X,Y point). All the details of how classes work cannot be covered in a single column, and we'll touch on them only briefly in this initial presentation.

Built-in Types

C# offers a standard set of built-in data types, such as short, long, and double. These are similar to the corresponding types in C/C++. They have a standard size; for example, long is 64 bits. Values of the char type hold a single Unicode character (16 bits); an 8-bit unsigned byte type exists as well.

One substantial difference from C is the provision of a decimal type, in addition to the usual floating-point types float and double. Floating-point arithmetic does poorly at handling decimal calculations (e.g., payroll deductions), and the decimal type offers an alternative. Here's an example of where it matters:

```
using System;

public class Dec {
      public static void Main() {

          // add 0.1 to itself twice and compare to 0.3,
          // using the double type

          double dbl = 0.1;
          if (dbl + dbl + dbl == 0.3)
              Console.WriteLine("double is equal");
          else
              Console.WriteLine("double is unequal");

          // the same, but using the decimal type

          decimal dec = 0.1m;
          if (dec + dec + dec == 0.3m)
              Console.WriteLine("decimal is equal");
          else
              Console.WriteLine("decimal is unequal");
      }
}
```

Using a double type, 0.1 added to itself three times does not result in 0.3, due to floating-point representation problems: 0.1 is the sum of an infinite series of negative powers of two (0.00110011001...) and therefore is not exactly representable in floating-point format. The decimal type solves this problem and is useful in areas such as financial calculations. Of course, it's a bit slower in execution.

Another difference from C is the string type. Here's an example:

```
using System;

public class String {
      public static void Main() {
          string s = "testing";
          Console.WriteLine(s);
      }
}
```

A rich set of functions for manipulating strings is also available.

Type-Checking and Conversions

C# applies tighter type-checking rules to the use of built-in types than C and C++. For example, in this code:

```
using System;

public class Conv {
      public static void Main() {
          ulong a = 0xffffffff;
          uint b;

          //b = a;
          b = (uint)a;
      }
}
```

an explicit cast is required to convert the unsigned long value to unsigned int. Such conversion often represents a programming mistake, and the programmer is required to specify explicitly that the conversion is desired (and, presumably, to consider its implications).

A related example is the use of Boolean expressions:

```
using System;

public class Bool {
      public static void Main() {
          int x = 100;
```

```
            //if (x) {}
            if (x != 0) {}
        }
    }
```

C# requires that the controlling expression in an if statement be of Boolean type, not simply a numeric or pointer type that can be checked against zero.

Boxing and Unboxing

What is the relation of built-in types like int to class types? In some languages, there is no connection – the two kinds of types have nothing in common.

C# approaches things a little differently. An automatic conversion called "boxing" is supplied by the compiler in order to convert a value of a built-in type to a class type. Let's look at an example:

```
using System;
using System.Collections;

public class Box {
    public static void Main() {

        // box an integer value

        int n = 100;
        object obj = n;

        // unbox it

        //n = obj;
        n = (int)obj;

        // add some integer values to a collection

        ArrayList list = new ArrayList();
        list.Add(1);
        list.Add(2);
        list.Add(3);
        for (int i = 0; i < list.Count; i++)
            Console.WriteLine(list[i]);
    }
}
```

In the first part of the code, an object of the root class System.Object is initialized with an integer value (100). The conversion involves creating a wrapper object of class type System.Int32 for the integer value. In other words, an object of the class System.Int32 is created and initialized with the value 100. Because this data type is part of the class hierarchy with System.Object as its root, the assignment is valid.

Here's some intermediate language output that illustrates what is going on in the first part of the example:

```
IL_0000:  ldc.i4.s      100
IL_0002:  stloc.0
IL_0003:  ldloc.0
IL_0004:  box           [mscorlib]System.Int32
IL_0009:  stloc.1
IL_000a:  ret
```

The boxing conversion is not without cost, but at the same time eases programming. In the second part of the example, some values of a built-in type are added to an ArrayList collection. The collection requires values of "object" type, so the integers are boxed automatically before being added to the collection.

The automatic boxing conversion, along with the existence of wrapper classes such as System.Int32, implies that the distinction between built-in and class types is blurred to some extent.

Enumerated Types

C# also supports enumerated types, similar to those offered by C. Here's some code that defines an enum to represent the colors red, green, and blue:

```
using System;

enum Color : byte {RED = 1, GREEN = 2, BLUE = 3}

public class Enum {
    public static void Main() {
        Color c = Color.GREEN;

        //int i = c;
        int i = (int)c;

        Console.WriteLine(i);
    }
}
```

The enum base type is byte, an unsigned value 0–255. Enum types are not interchangeable with integral types; conversion requires an explicit cast.

Class and Struct Types

The C# concept of a class is similar to that found in the C++ and Java languages. A class defines some data (which objects of the class will contain) and operations on that data, expressed through functions ("methods") of the class. If you're a C programmer, a class is a grouping of some data defined in a struct coupled with some functions that operate on instances of the struct. Data in objects is generally hidden or private, and accessible only via the methods of the object's class.

Let's look at an example, one that uses a class to represent X,Y points:

```
using System;

public class PointClass {
      private int x, y;

      public PointClass(int x, int y) {
          this.x = x;
          this.y = y;
      }
      public int getX() {return x;}
      public int getY() {return y;}
}

public struct PointStruct {
      private int x, y;

      public PointStruct(int x, int y) {
          this.x = x;
          this.y = y;
      }

      public int getX() {return x;}
      public int getY() {return y;}
}

public class Struct {
      public static void Main() {
          PointClass pc = new PointClass(10, 20);
          PointStruct ps = new PointStruct(30, 40);
      }
}
```

The PointClass class defines private data members x and y, along with a constructor to create new objects (a constructor is denoted as a method with the same name as the class). There are also getX and getY accessor methods, to access the x and y values from a PointClass object.

In the Main method, an object of PointClass is created using the new operator, and the constructor is called after space is allocated from the heap. C# uses garbage collection, so you don't need to worry about explicitly freeing object space when you're done with it; garbage collection takes care of this for you automatically.

PointStruct is a class that seems identical to PointClass, except that it's defined with a struct keyword instead of a class. What's the difference? A struct is a lighter-weight type than a class, with some restrictions. For example, a struct cannot inherit from another class or struct.

A really key difference is that a struct is a value type, whereas a class is a reference type. In the example above, when an object of PointStruct is allocated via the new operator, it's allocated from the stack, not the heap. Here's some intermediate language output that shows the difference between the two new calls:

```
IL_0000:  ldc.i4.s     10
IL_0002:  ldc.i4.s     20
IL_0004:  newobj       instance void PointClass::.ctor
IL_0009:  stloc.0
IL_000a:  ldloca.s     V_1
IL_000c:  ldc.i4.s     30
IL_000e:  ldc.i4.s     40
IL_0010:  call         instance void PointStruct::.ctor
IL_0015:  ret
```

When struct objects are passed to methods, they are passed by value, with a copy made of the object. For example, in this code:

```
using System;

public class A {            // class
      public int x;
}

public struct B {           // struct
      public int x;
}

public class Struct2 {
      public static void f(A aref, B bref) {
          aref.x = 30;
          bref.x = 40;
      }

      public static void Main() {
          A aref = new A();
          aref.x = 10;

          B bref = new B();
          bref.x = 20;

          f(aref, bref);

          Console.WriteLine("{0} {1}", aref.x, bref.x);
      }
}
```

the values "30 20" are printed. The object of class A is passed by reference, whereas the B object is passed by value. Method f can modify the value of the A object in a way that's visible outside of f, because it has a pointer to the object. But f has only a copy of the B object.

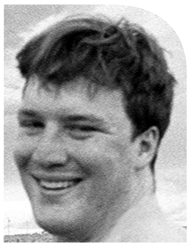
A struct is most useful for types that are very simple. If you're defining an X,Y point type, for example, it might be worth using a struct instead of a class.

In future discussions, we'll get into more detail of how classes and structs work and look at related concepts such as interfaces and abstract classes.

non-proprietary flash solutions

by Kragen Sitaker

Kragen Sitaker is a multilingual hacker who's used UNIX since 1992, presently consulting on server-side Web software development in San Francisco. See *http://pobox.com/~kragen/* for more.



kragen@pobox.com

Flash, or SWF, is an open-file format developed by Macromedia. Flash files can contain executable code, raster and vector images, and sounds in a variety of formats. Many people use Flash for animated banner ads, online games, and more advanced applications. Macromedia frequently refers to Flash files as "movies," but they can do much more than, say, AVI files.

Because Macromedia distributes a browser plug-in, running ("playing") a Flash file is as simple (for most people) as viewing a Web page.

I think the executable code represents a tokenized and possibly compiled version of an ugly scripting language Macromedia calls "Lingo." (Recent versions of Flash support scripting in JavaScript, known as "ActionScript," as well.) Supposedly the code is confined so that it can't communicate with arbitrary network addresses from the executing machine, or with arbitrary code running on your machine, although it can communicate with JavaScript in a Web page that embeds it.

On my laptop, I can't run Flash files, because although Macromedia has released details on the file format, its "playing" software is proprietary, although available gratis, and I don't install proprietary software on my laptop.

I also write a lot of cute little hacks and post them to kragen-hacks, but few people run them, because it typically takes several steps to install them. Many of them could be converted to Flash without much loss, and then people could run them very easily. But since Macromedia's Flash-authoring tools are also proprietary, I don't install them.

So I took an inventory of the world's Flash reading and writing software (except for the proprietary stuff). The results follow.

Free SWF Tools for Producing and Viewing Flash (2003-03-15)

DrawSWF, *http://drawswf.sf.net* – Java 1.4 drawing program that draws in SVG and exports to SWF. The SWF library is licensed under a BSD license. Not sure if it's useful for arbitrary animation creation, and it surely isn't useful for editing existing Flash files.

Tubesock, *http://tubesock.sf.net* – GTK/GNOME shockwave file player. Looks dead as of mid-2002. Probably in C. They planned a Mozilla plug-in eventually.

SWF Tools, GPL, *http://www.quiss.org/swftools/* – a merging tool (swfcombine); an extracting tool (swfextract); conversion from PDF, JPEG, PNG, AVI, and WAV to SWF; a text parsing tool called swfstrings; an SWF parser called swfdump; and rfxswflib, a library for reading and writing SWFs. Some pretty cool Flash files here, made with the SWF Tools, including a CGI script that generates Flash files dynamically!

swfdec, *http://swfdec.sf.net* – a library for rendering Flash animations, including a GTK+ player (swf_play) and a Mozilla plug-in. Might be why Tubesock died. Looks quite active. Swfdec 0.2.0 is out. LGPL.

gplflash, *http://www.swift-tools.com/Flash/* – a GPL library for rendering Flash animations, also including a stand-alone player, a plug-in, and a KDE screensaver. I have a feeling it's out-of-date. Don't know if it really is.

svg2swf, *http://www.eskimo.com/~robla/svg2swf/* – a Python script that parses an SVG file using SAX and writes an SWF file with the Ming library.

Ming, LGPL (according to Freshmeat), *http://ming.sourceforge.net* (formerly *http://www.opaque.net/ming/*) – an SWF output library in C with bindings for C++, PHP, Perl, Python, and Ruby; as though development is active again.

Ming-Sharp, *http://ming-sharp.sourceforge.net* – an LGPL .NET binding for Ming? Works with Mono. Says it supports "almost all of Flash 4's features, including shapes, gradients, bitmaps (PNGs and JPEGs), morphs ('shape tweens'), text, buttons, actions, sprites ('movie clips'), streaming MP3, and color transforms – the only thing that's missing is sound events." Presumably that means Ming supports all these too.

gAnim8, W3C free software license, *http://ganim8.sf.net* – a "suite of tools" for viewing and editing movies, including SWF. GTK. Looks actively developed. Apparently it uses ffmpeg to write SWF files and can't read them. In Python?

the tclsh spot

by Clif Flynt

Clif Flynt is president of Noumena Corp., which offers training and consulting services for Tcl/Tk and Internet applications. He is the author of *Tcl/Tk for Real Programmers* and the *TclTutor* instruction package. He has been programming computers since 1970 and a Tcl advocate since 1994.

clif@cflynt.com

Capturing and Analyzing IP Packets on the AX-4000 Using AxTcl

Previous Tclsh Spot articles described building a program for generating Ethernet packets and doing simple validation tests on them.

This article explains how to use the Spirent/AdTech AX-4000 to capture and analyze packets, report good/bad packet statistics, and generate human-readable descriptions of the packets.

The AX-4000 (*http://www.adtech-inc.com/*) is a configurable piece of hardware that can generate and analyze data packets on four different transmission technologies (IP, ATM, Ethernet, and Frame Relay) simultaneously at speeds up to 10 Gbps.

The AX-4000 can be configured with either a GUI or the AdTech Tcl extension. The GUI is suitable for many purposes, but the Tcl interface is ideal for situations where you want to run the same test multiple times while modifying the test environment.

The basic activity flow for an AdTech Tcl script is:

1. Load the AxTcl extension.
2. Initialize the connection to the AX-4000 controller.
3. Reserve an interface card set.
4. Create a generic interface object attached to the card set.
5. Create an analyzer or generator attached to the interface.
6. Configure the analyzer or generator.
7. Run the test.
8. Analyze the results.

As described in a previous Tclsh Spot article (*;login:*, Vol. 28 #1, February 2003), this script will initialize the AX-4000 by loading the extension, describe the location of the hardware definition (BIOS) files, lock an interface card set, and initialize it:

```
# Define the base directory for the Spirent software.
set base "[file join [file dirname [info script]] ../..]"

# Add directories to search for packages.
lappend auto_path . $base/bios/tcllib/xml $base/tclclib

# Load the extension.
if {[catch {package require ax4kpkg}]} {
     if {[catch {load $base/tclwin/libax4k.dll ax4kpkg}]} {
         catch {load $base/tclclib/libax4k.so ax4kpkg}
     }
}

# Define the hardware bios directory.
ax hwdir $base/bios

# Initialize the AX-4000.
ax init -remote $ipAddress -user clif -conout 1

# Lock a device.
enet lock $deviceID $ipAddress $logicalID

# Create and configure the interface to this card set.
interface create int1 $logicalID -ifmode IPoETHER

int1 set -mode normal -dataRate MBS10
int1 run
```

This code implements steps 1–4 of the basic activity flow and can be copied into each application with little modification.

The AX-4000 can be used as either a packet generator or a packet analyzer, or it can both analyze and generate.

The goal of this software is to validate the previously described packet generator, so we must create an analyzer, but do not need a generator object.

Syntax: analyzer create *Name Device*

analyzer create	Create a new analyzer object.
Name	The name to assign to the new analyzer.
Device	The device to attach this analyzer name to. This is the logical device that was locked in a previous enet lock command.

The analyzer create command follows the paradigm used by Tk to create image objects. It initializes a new analyzer data object, and creates a new command to use to interact with that object.

The analyzer command supports many subcommands, including:

analyzerName set *option value*
  Sets one or more configuration options for this analyzer. Configuration options vary for different analyzer cards.

analyzerName display
Returns a list of the current settings.
analyzerName run
Starts the analyzer running.
analyzerName reset
Stops the analyzer and clears all the statistics the analyzer can gather.
analyzerName stop
Stops the analyzer but does not clear any values.
analyzerName destroy
Destroys the analyzer, freeing it for other use.
analyzerName stats
Returns a set of keyword/value pairs as a list. The exact return depends on the analyzer being used.
analyzerName capture *?subcommand*
Configures the analyzer to capture packets.

The *analyzerName* stats command will return statistics about the packets that have been seen (how many good, how many with certain failures, average size, etc.). By collecting the starting and ending statistics, we can show the statistics of line traffic over a period of time.

The test application must control both the AX-4000 and the script that generates packets. The packet-generating code could be merged into the AX-4000 control script for testing, but it makes more sense to keep the generating code as a separate program, as it will be used in the final application.

There are two ways to execute another program from within a Tcl script. You can open a pipe to the second program with the open command, or, if your script just needs the program to run and is not monitoring the output from the program, you can use the exec command.

The exec command will execute an external program, and whatever output that program would send to stdin or stdout will be returned to the script as the exec return.

Syntax: exec *?-options? arg1 ?arg2...argn?*

exec	Execute arguments in a subprocess.	
-options	The exec command supports two options:	
	-keepnewline	Normally a trailing newline character is deleted from the program output returned by the exec command. If this argument is set, the trailing newline is retained.
	- -	Denotes the last option. All subsequent arguments will be treated as subprocess program names or arguments.
arg	These arguments can be either a program name, a program argument, or a pipeline descriptor.	

This line of code will run the testGen.tcl application at the proper time.

```
exec tclsh /home/clif/IP/PacketMaker/testGen.tcl
```

The packet generator should be started as soon as the AX-4000 has been configured. The AxTcl commands that configure the AX-4000 might take several milliseconds to complete, as the new configuration details are copied from the host machine to the AX-4000.

Since the AxTcl configuration commands return immediately, instead of waiting for all processing to be complete, the Tcl script needs to pause after sending the setup commands before continuing processing.

The Tcl after command provides access to a timer with millisecond resolution. Like other Tcl event-driven programming support, this command allows the script to pause or to schedule a future script.

Syntax: after *milliseconds ?script?*

after	Pause processing of the current script, or schedule a script to be processed in the future.
milliseconds	The number of milliseconds to pause the current processing, or the number of seconds in the future to evaluate another script.
script	If this argument is defined, this script will be evaluated after *milliseconds* have elapsed.

To ensure that the AX-4000 is ready, we must schedule a delay between configuring the AX-4000 and generating packets and between starting and stopping packet collection.

This can be done by simply pausing the application with a command like after 1000, but while the application is paused, it does absolutely nothing: it doesn't check for events, won't respond to keyboard input (except control-c), etc.

For code that might be embedded in an interactive application (I actually put this code inside a GUI test harness), a better paradigm is to schedule events to happen in the future, and use the vwait command to synchronize events.

The vwait command causes the interpreter to stop linear processing of a script until a variable is assigned a new value. While the interpreter is waiting for the variable to change, it continues processing events.

Syntax: vwait *varName*

varName	The variable name to watch. The script following the vwait command will be evaluated after the variable's value is modified.

The next example initializes an analyzer, starts the packet generator running, and collects five seconds' worth of statistics.

```
# Get the starting stats.
set anaStats1 [ana1 stats]

# Start the analyzer.
ana1 run

# Schedule the packet generator to start in 1 second.
# Schedule a break to happen after 5 seconds of operation.

after 1000 {exec tclsh /home/clif/IP/PacketMaker/testGen.tcl}
after 6000 {set stop 1}

set stop 0
vwait stop

# Stop the analyzer, get stats, and destroy it.
set anaStats2 [$analyzer stats]
$analyzer stop
$analyzer destroy

# And unlock the device for the next user.
enet unlock $logicalID

genReport $anaStats1 $anaStats2
```

The *analyzerName* stats command will return the statistics as a list of key/value pairs. The foreach command can be used to step through multiple lists, to collate results. The genReport procedure steps through the list of statistics saved before the application was run, compares it to the results after, and only displays the values that changed.

```
proc genReport {stats1 stats2} {
        foreach {key1 val1} $stats1 {key2 val2} $stats2 {
            if {$val1 != $val2} {
                append report [format "%-30s %12s %12s\n" $key1 $val1 $val2]
            }
        }
    puts $report
}
```

The output from the code above resembles this:

```
ANALYZER STATS
-elapsedTime                  3              5603
-totalPackets                 0               800
-totalPacketBytes             0             67400
-goodPackets                  0               500
-goodPacketBytes              0             38600
-goodDatagramBytes            0             26800
-totalPacketRate              0               212
-goodPacketRate               0               133
-goodPacketBitRate            0                82
```

```
-goodDatagramBitRate      0.0          56.8
-lineRatePerc             0.00          1.93
-tcpPackets               0           300
-tcpRatio                 0.00          0.38
-tcpChecksumErrors        0           200
-udpPackets               0           300
-udpRatio                 0.00          0.38
-udpChecksumErrors        0           100
-icmpPackets              0           200
-icmpRatio                0.00          0.25
-ipPackets                0.00        800.00
-avgDatagramLength        0            54
-minDatagramLength        0            32
-maxDatagramLength        0            88
-avgPacketLength          0            77
-minPacketLength          0            64
-maxPacketLength          0           106
-substreamCount           0             4
-substreamErrorCount      0             2
-filterCount              1             5
```

These results show that 800 packets were generated, of which 300 were UDP packets, 200 were ICMP packets, and 300 were TCP packets; 100 of the UDP packets and 200 of the TCP packets had checksum errors.

The generator program was configured to generate the packets described, so this result was expected. While good statistics are a good start, to validate the generator, we should confirm that the checksum errors are all of the expected errors.

To analyze individual packets, we must first capture them. The analyzer is configured to capture packets with the *analyzerName* capture setup command.

Syntax: *analyzerName* capture setup *-option value*

capture setup — Configure the capture rules.

-option value — Option value pairs to configure the capture. Options include:

-segmentsize *Num*
: Specifies the size in blocks of the captured segment. For OC-12c, a block is 2 kilobytes. For OC-48x, a block is 4 kilobytes.

-qualifyEquation *equation*
: Specifies the equation used to qualify the capture. The default is 1 (always true).

-triggerEquation *equation*
: Specifies the equation used to trigger (start) the capture. Default is 1 (start capture immediately).

-triggerPosition *Position*
: Defines when the capture will commence in relation to the trigger event. This will control whether the packet that triggers capturing to start is also captured. May be one of: AFTER_START, BEFORE_CENTER, AFTER_CENTER, or BEFORE_END.

There are many more options that can be used to define the conditions under which you want the AX-4000 to start capturing packets. These are enough for this fairly simple test.

The trigger and qualifying equations support logical operations, with multiple levels of parentheses allowing a script to define arbitrarily complex rules for packet capture. For example, you can define how many bad packets must be seen before capture starts, and which types of packets will be captured after the capturing is triggered.

To test the packet generator, we want to capture all packets for as long as the test is running. Thus, the trigger and qualify equations are trivial, just a TRUE value (1).

After the analyzer is running, the capture can be armed with the *analyzerName* capture arm command. This will cause the capture subsystem to look at (but not necessarily capture) packets. Once the trigger condition is met, packets that match the qualifying rules will be captured.

Syntax: *analyzerName* capture arm *boolean*

capture arm	Turn on (or off) the capture subsystem.
boolean	A boolean value. A True (1) will start the capture subsystem examining packets, and a False (0) will make the subsystem stop examining packets.

This code will initialize the capture subsystem and start capturing packets:

```
# Set up the capture buffer size.
# The qualify and capture equations are both true.
# Stat capturing as soon as the "arm" is set to 1.

ana1 capture setup -segmentSize 1024 \
     -qualifyequation 1 \
     -triggerPosition AFTER_START -triggerequation 1

# Start capturing packets.
ana1 capture arm 1
```

Once a set of packets have been captured, the capture can be turned off by disarming the capture circuit with an *analyzerName* capture arm 0 command.

The next step is to analyze the captured data.

The *analyzerName* capture getdata command retrieves the captured packets from the AX-4000.

Syntax: *analyzerName* capture getdata ?option?

capture getdata	Returns the captured data, or a requested subset of data, either as a list or in an array.
?-array *arrayName*	Stores captured data in an array, instead of returning as a list. Elements are indexed numerically in the order they were received.
start#-end#	The *start* and *end* values are integers representing the first and last packet in the sequence to return.

The data is stored in the array as a keyword/value list, with keywords that include:

-timestamp
  The absolute time in nanoseconds that the packet was received.
-indexNum
  Numeric location of this packet in the packet stream (after trigger).
-eventFlags
  A list of errors in the packet.
-payload
  The packet data (as hex bytes).

With this return, we can check whether or not packets had errors and whether the data conforms to the expected values.

Retrieving the hex values of the packets allows us to do a bit-by-bit comparison between the packets we intended to generate and the data that hit the wire. Doing this by hand is tedious and error-prone. The axdecode command will decode a data packet into human-readable output.

Syntax: axdecode *protocol ?-option value?*

-payload {<Tcl_list>} | -message "<message_string>"} | -dgram <datagram_object> | -hexstring "<hex_string>" | -file <filename>

protocol	The protocol of the message to be decoded. May be one of: PPP, ARP, IP, IS-IS, NHRP, RARP, RTCP, BISUP, B-ICI, or Q.2763.
-option value	Options include:

-payload
: The packet is a numeric list.

-message
: The packet is a string of ASCII characters.

-datagram
: The packet is a datagram object.

-hexstring
: The packet is a string of hexadecimal digits.

-file
: The packet is contained in the specified file.

In order to use the axdecode command, your script must know the type of packet being decoded and convert that value to an appropriate string.

The 12th and 13th bytes of an Ethernet II packet, or 20th and 21st bytes of an 802.3 packet, contain the 16-bit packet type. Converting these bytes to an ASCII string is a simple lookup operation, easily performed with an associative array.

The raw data consists of two hex bytes separated by a space. We normally use a single word for an associative array index. The two hexadecimal bytes could be merged into a single 16-bit value with regsub or the newer string map commands. However, while it is common practice to use a single word for an associative array index, this is not required. Any characters between the parentheses are accepted as an index, so we can easily use two strings as an associative array index.

These two lines of code will define a lookup table and will set the variable type to the appropriate string for an Ethernet II data packet and then display a decoded version of the packet.

```
# Define a lookup table.
array set etherTypes { {08 00} IP {08 06} ARP {80 35} RARP }

# Download captured data to associative array "array1".
ana1 capture getdata -array array1

# Extract first packet from array of captured packets.
array set packet $array1(0)

# Convert bytes 12 and 13 to a string.
set type $etherTypes ([lrange $packet(-payload) 12 13])

# Extract the IP packet from the Ethernet II packet.
set payload [lrange $packet(-payload) 14 end]

# Display the decoded output.
puts [axdecode $type -hexstring $payload]
```

This code would generate output like this for an ICMP Address Mask Request packet with a bad checksum field:

```
-summary {IPv4-<ICMP-<address mask request}
-decode {
<0000, 0032, 01> | ---- packet: IP ----
<0000, 0020, 02>   | ---- packet: IP PDU ----
<0000, 0014, 03>      | ---- packet: IP header ----
<0000, 0001, 03>      | {0100 ....}= 04h [004d] version: IP Internet Protocol
<0000, 0001, 03>      | {.... 0101}= 05h [005d] header length: in 32 bit units, must be 5 or more
<0001, 0001, 04>        | ---- packet: flags ----
<0001, 0001, 04>        | {000. ....}= 00h [000d] precedence field: Routine
<0001, 0001, 04>        | {...0 ....}= 00h [000d] minimize delay: normal delay
<0001, 0001, 04>        | {.... 0...}= 00h [000d] maximize throughput: normal throughput
<0001, 0001, 04>        | {.... .0..}= 00h [000d] maximize reliability: normal reliability
<0001, 0001, 04>        | {.... ..0.}= 00h [000d] minimize monetary cost: normal monetary cost
<0001, 0001, 04>        | {.... ...0}= 00h [000d] unused: must be 0
<0001, 0001, 04>        | ---- end of packet: flags ----
<0002, 0002, 03>      | 00-20 = 20h [032d] total length: in octets include header length, must not
                                              be less than 20
<0004, 0002, 03>      | 00-02 = 02h [002d] identification
<0006, 0001, 03>      | {0... ....}= 00h [000d] reserved: valid
<0006, 0001, 03>      | {.0.. ....}= 00h [000d] Don't Fragment: May Fragment
<0006, 0001, 03>      | {..0. ....}= 00h [000d] More Fragments: Last Fragment
<0006, 0002, 03>      | fragment offset:
<0006, 0001, 03>      | {...0 0000}= 00h [000d] bits 12-8
<0007, 0001, 03>      | {0000 0000}= 00h [000d] bits 7-0
<0006, 0002, 03>      | fragment offset = 00h [000d]: in 64 bits units
<0008, 0001, 03>      | {0010 0000}= 20h [032d] time to live: seconds
<0009, 0001, 03>      | {0000 0001}= 01h [001d] protocol: ICMP (Internet Control Message)
<000A, 0002, 03>      | 07-78 = 778h [1912d] header checksum: checksum is correct
<000C, 0004, 03>      | 192.168.9.2 source ip address
<0010, 0004, 03>      | 192.168.9.17 destination ip address
<0000, 0014, 03>      | ---- end of packet: IP header ----
<0014, 000C, 03>      | ---- packet: IP datagram ----
<0014, 000C, 04>        | ---- packet: ICMP ----
<0014, 0001, 04>        | {0001 0001}= 11h [017d] type: address mask request
<0015, 000B, 05>          | ---- packet: address mask request ----
<0015, 0001, 05>          | {0000 0000}= 00h [000d] code:
<0016, 0002, 05>          | EE-99 = EE99h [61081d] checksum: checksum is incorrect, expected
                                                    EEFCh [61180d]
<0018, 0002, 05>          | 00-01 = 01h [001d] identifier
<001A, 0002, 05>          | 00-02 = 02h [002d] sequence number
<001C, 0004, 05>          | 0.0.0.0 mask
<0015, 000B, 05>          | ---- end of packet: address mask request ----
<0014, 000C, 04>        | ---- end of packet: ICMP ----
<0014, 000C, 03>      | ---- end of packet: IP datagram ----
<0000, 0020, 02>    | ---- end of packet: IP PDU ----
<0020, 0012, 01> | packet pad:
<0020, 0010, 01> | 0000: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 28 79 | ........ ......(y
<0030, 0002, 01> | 0010: E3 C2 | ..
<0000, 0032, 01> | ---- end of packet: IP ----
}
```

Note that the decoded packet shows an ICMP packet with a checksum error in the payload, but the stats command reported no ICMP errors.

The output from tcpdump also shows an error:

```
01:08:11.848609 192.168.9.2 < 192.168.9.17: icmp: address mask request
    (wrong icmp csum) (ttl 32, id 2, len 32)
         4500 0020 0002 0000 2001 0778 c0a8 0902
         c0a8 0911 1100 ee99 0001 0002 0000 0000
```

The code that generated this packet is shown below. It will generate a packet with a checksum error in the payload and a correct checksum in the IP header. The packet-generating code was described in the previous Tclsh Spot article. Initializing the checksum to a non-zero value will generate an invalid checksum.

```
# Generate a bad checksum icmp Address Mask Request
set icmp2 [packet::make ICMP type ICMP_ADDRESS code 0 checksum 99 identifier 1 \
    sequence 2 subnet 00]

set ip2 [packet::make IP version 4 hdrlen 5 tos 0 length 32 id 2 flag 0 \
    offset 0 ttl 32 protocol ICMP checksum 0 source \
    192.168.9.2 dest 192.168.9.17 options {} payload [$icmp2 getField packet]]

set ep2 [packet::make ETHER dest 00:E0:4C:00:14:4D src 00:A0:CC:D1:B6:00 \
    type IP payload [$ip2 getField packet]]
```

The AX-4000 packet scans don't examine payloads. The *analyzerName* stats command returns no IP errors because there are no packets with bad IP checksums. The axdecode and tcpdump -s 15000 -l -x -n -v -i eth1 commands examine the payload and report the internal errors. (A simpler tcpdump -li eth1 command misses the error condition.)

While validating and debugging the packet generator, it's useful to get not only the error report (which tcpdump provides) but also the expected value (as reported by axdecode).

The ability to get a human-readable output becomes more important when we get to confirming more complex packets like the ICMP Destination Unreachable message, which includes the IP header and part of the payload of the packet that failed to reach a destination.

This code will generate a TIMESTAMP request, package it into an IP packet, and then use that IP packet as the body for an ICMP ICMP_DEST_UNREACH message.

```
# Generate a good icmp TIMESTAMP request
set icmp1 [packet::make ICMP type ICMP_TIMESTAMP code 0 \
    checksum 0 identifier 1 sequence 2 \
    originate 0x98765430 receive 0x98765431 transmit 0x98765432]

set ip1 [packet::make IP version 4 hdrlen 5 tos 0 length 32 id 2 flag 0 \
    offset 0 ttl 32 protocol ICMP checksum 0 source \
    192.168.9.2 dest 192.168.9.17 options {} \
    payload [$icmp1 getField packet]]

set pld1 [$ip1 getField packet]

# Make a good DESTINATION UNREACHABLE icmp message, using the
#    previous IP Packet for the body.

set icmp2 [packet::make ICMP type ICMP_DEST_UNREACH \
    code ICMP_DEST_UNREACH.ICMP_NET_UNREACH \
    checksum 0 zero 0 ipHeader [lrange $pld1 0 19] \
    payloadHeader [lrange $pld1 20 27] ]

set ip2 [packet::make IP version 4 hdrlen 5 tos 0 \
    length [expr 20 + [llength $pld]] id 2 flag 0 \
    offset 0 ttl 32 protocol ICMP checksum 0 source \
    192.168.9.2 dest 192.168.9.17 options {} \
    payload [$icmp2 getField packet]]
```

This generates a lengthy description of the packet, which is much more readable than the tcpdump hex display of the packet contents.

```
<0000, 003C, 01> | ---- packet: IP ----
<0000, 0038, 02< | ---- packet: IP PDU ----
<0000, 0014, 03>     | ---- packet: IP header ----
<0000, 0001, 03>     | {0100 ....}= 04h [004d] version: IP Internet Protocol
<0000, 0001, 03>     | {.... 0101}= 05h [005d] header length: in 32-bit units, must be 5 or more
<0001, 0001, 04>       | ---- packet: flags ----
<0001, 0001, 04>       } {000. ....}= 00h [000d] precedence field: Routine
<0001, 0001, 04>       | {...0 ....}= 00h [000d] minimize delay: normal delay
<0001, 0001, 04>       | {.... 0...}= 00h [000d] maximize throughput: normal throughput
<0001, 0001, 04>       | {.... .0..}= 00h [000d] maximize reliability: normal reliability
<0001, 0001, 04>       | {.... ..0.}= 00h [000d] minimize monetary cost: normal monetary cost
<0001, 0001, 04>       | {.... ...0}= 00h [000d] unused: must be 0
<0001, 0001, 04>       | ---- end of packet: flags ----
<0002, 0002, 03<     | 00-38 = 38h [056d] total length: in octets include header length, must not
                                                  be less than 20
<0004, 0002, 03<     | 00-02 = 02h [002d] identification
<0006, 0001, 03<     | {0... ....}= 00h [000d] reserved: valid
<0006, 0001, 03<     | {.0.. ....}= 00h [000d] Don't Fragment: May Fragment
<0006, 0001, 03<     | {..0. ....}= 00h [000d] More Fragments: Last Fragment
<0006, 0002, 03<     | fragment offset:
<0006, 0001, 03<     | {...0 0000}= 00h [000d] bits 12-8
<0007, 0001, 03<     | {0000 0000}= 00h [000d] bits 7-0
<0006, 0002, 03<     | fragment offset = 00h [000d]: in 64-bit units
<0008, 0001, 03<     | {0010 0000}= 20h [032d] time to live: seconds
<0009, 0001, 03<     | {0000 0001}= 01h [001d] protocol: ICMP (Internet Control Message)
<000A, 0002, 03<     | 07-60 = 760h [1888d] header checksum: checksum is correct
<000C, 0004, 03>     | 192.168.9.2 source IP address
<0010, 0004, 03>     | 192.168.9.17 destination IP address
<0000, 0014, 03>     | ---- end of packet: IP header ----
<0014, 0024, 03>     | ---- packet: IP datagram ----
<0014, 0024, 04>       | ---- packet: ICMP ----
<0014, 0001, 04>       | {0000 0011}= 03h [003d] type: destination unreachable
<0015, 0023, 05>         | ---- packet: destination unreachable ----
<0015, 0001, 05>         | {0000 0000}= 00h [000d] code: network unreachable
<0016, 0002, 05>         | C2-F7 = C2F7h [49911d] checksum: checksum is correct
<0018, 0004, 05>         | 00-00-00-00 = 00h [000d] unused
<001C, 001C, 06>           | ---- packet: IP packet caused error ----
<001C, 001C, 07>             | ---- packet: IPv4 ----
<001C, 001C, 08>               | ---- packet: IP PDU ----
<001C, 0014, 09>                 | ---- packet: IP header ----
<001C, 0001, 09>                 | {0100 ....}= 04h [004d] version: IP Internet Protocol
<001C, 0001, 09>                 | {.... 0101}= 05h [005d] header length: in 32-bit units, must
                                                        be 5 or more
<001D, 0001, 0A>                   | ---- packet: flags ----
<001D, 0001, 0A>                   | {000. ....}= 00h [000d] precedence field: Routine
<001D, 0001, 0A>                   | {...0 ....}= 00h [000d] minimize delay: normal delay
<001D, 0001, 0A>                   | {.... 0...}= 00h [000d] maximize throughput: normal
                                                           throughput
<001D, 0001, 0A>                   | {.... .0..}= 00h [000d] |maximize reliability: normal
                                                            reliability
<001D, 0001, 0A>                   | {.... ..0.}= 00h [000d] minimize monetary cost: normal
                                                           monetary cost
```

```
<001D, 0001, 0A>                   | {.... ...0}= 00h [000d] unused: must be 0
<001D, 0001, 0A>                   | ---- end of packet: flags ----
<001E, 0002, 09>                 | 00-20 = 20h[032d] total length: in octets include header
                                                     length, must not be less than 20
<0020, 0002, 09>                 | 00-02 = 02h [002d] identification
<0022, 0001, 09>                 | {0... ....}= 00h [000d] reserved: valid
<0022, 0001, 09>                 | {.0.. ....}= 00h [000d] Don't Fragment: May Fragment
<0022, 0001, 09>                 | {..0. ....}= 00h [000d] More Fragments: Last Fragment
<0022, 0002, 09>                 | fragment offset:
<0022, 0001, 09>                 | {...0 0000}= 00h [000d] bits 12-8
<0023, 0001, 09>                 | {0000 0000}= 00h [000d] bits 7-0
<0022, 0002, 09>                 | fragment offset = 00h [000d]: in 64 bits units
<0024, 0001, 09>                 | {0010 0000}= 20h [032d] time to live: seconds
<0025, 0001, 09>                 | {0000 0001}= 01h [001d] protocol: ICMP (Internet Control
                                                           Message)
<0026, 0002, 09>                 | 07-78 = 778h [1912d] header checksum: checksum is correct
<0028, 0004, 09>                 | 192.168.9.2 source IP address
<002C, 0004, 09>                 | 192.168.9.17 destination IP address
<001C, 0014, 09>                 | ---- end of packet: IP header ----
<0030, 0008, 09>                 | ---- packet: IP datagram ----
<0030, 0008, 0A>                   | ---- packet: ICMP ----
<0030, 0001, 0A>                   | {0000 1101}= 0Dh [013d] type: timestamp request
<0031, 0007, 0B>                     | ---- packet: timestamp request ----
<0031, 0001, 0B>                     | {0000 0000}= 00h [000d] code:
<0032, 0002, 0B>                     | 2D-05 = 2D05h [11525d] checksum: checksum is incorrect,
                                                              expected 5284h [21124d]
<0034, 0004, 0B>                     | 00-01-00-02 = 10002h [65538d] Originate Timestamp
<0031, 0007, 0B>                     | ---- end of packet: timestamp request ----
<0030, 0008, 0A>                   | ---- end of packet: ICMP ----
<0030, 0008, 09>                 | ---- end of packet: IP datagram ----
<001C, 001C, 08>               | ---- end of packet: IP PDU ----
<001C, 001C, 07>             | ---- end of packet: IPv4 ----
<001C, 001C, 06>           | ---- end of packet: IP packet caused error ----
<0015, 0023, 05>         | ---- end of packet: destination unreachable ----
<0014, 0024, 04>       | ---- end of packet: ICMP ----
<0014, 0024, 03>     | ---- end of packet: IP datagram ----
<0000, 0038, 02>   | ---- end of packet: IP PDU ----
<0038, 0004, 01> | 0007: 00 BE 9F BA | .... = BE9FBAh [12492730d] packet pad
<0000, 003C, 01> | ---- end of packet: IP ----
```

Note that the innermost ICMP checksum is reported as being incorrect. This is because the packet analysis program is recalculating the checksum on the incomplete ICMP message.

In terms of using an AX-4000, this is a trivial application. The equipment is capable of monitoring and collecting data on a much more complex set of rules. However, it's a useful tool for testing and validating this application, which is the point of this exercise.

As usual, the code described in this article is available at *http://www.noucorp.com*.

practical perl: keeping it simple

by Adam Turoff

Adam is a consultant who specializes in using Perl to manage big data. He is a long-time Perl Monger, a technical editor for *The Perl Review*, and a frequent presenter at Perl conferences.

ziggy@panix.com

CPAN modules are a great way to construct a program by reusing existing components. However, gluing CPAN modules together is certainly not the only way to write a Perl program. In some cases, it is both simpler and easier to write a program *without* using CPAN modules. After all, with or without CPAN, Perl is still a great language for text hacking, automation, and application glue.

Building a program using CPAN modules is a great way to start writing a new program, and a great way to reduce development time. However, using CPAN modules adds dependencies that can complicate deployment. Remember that a program will not run unless all of its required modules are installed. In some extreme cases your program may run, but it will exhibit buggy behavior because it uses an older version of a module dependency. Although these issues are not insurmountable, they are worthy of consideration, especially in situations where a program will be installed both widely and often.

Dave Cross' NMS project (*http://nms-cgi.sourceforge.net*) is a perfect example. Dave wanted to write replacement programs for the ancient, buggy, insecure yet popular scripts found on Matt's Script Archive. Dave's replacement programs are targeted at unsophisticated users who want to add a stock feature on a Web site. Many of these users are not Perl programmers, nor do they wish to learn Perl just to install a silly guest-book script. The NMS programs use standard Perl features and core modules that are distributed with Perl, and have no dependencies on any modules found on CPAN. This makes it easy for users to just drop a file in their cgi-bin directory and get something that "just works."

I came across a similar situation recently. I maintain an online journal (Web log) called "use Perl," a community Web site for Perl programmers (my journal can be found at *http://use.perl.org/~ziggy/journal/*). I also receive email notifications when my friends post entries in their journals. I don't always have time to read these journals when they are posted, and often a few dozen will accumulate while I am busy working on a project. So I wrote a quick little program to download the email notifications so that I can catch up on my backlog as quickly as possible.

I use procmail at my ISP to store all email notifications of "use Perl" journal postings in a separate mailbox. The easiest way for me to automate viewing a few dozen journal entries at a time is to download the mailbox file, extract the entry URL in each message, and load it in a Web browser. This certainly isn't the most important program I have ever written, but it does exhibit two virtues of being a programmer: laziness and impatience. After all, I do not want to spend hours at a time scanning through old journal entries. I'd rather read these messages as quickly as possible and move on to more interesting tasks.

First Attempt: Use CPAN Modules

I wrote my first view-useperl script about two years ago. I think it took all of 10 minutes to write. The process it automates is quite simple: Download a mailbox from my ISP, get the first URL in the body of each message in the mailbox, and display each URL in turn. I started by looking around CPAN, and I found Mark Overmeer's Mail-Box distribution. This distribution contains the Mail::Box and Mail::Box::Manager modules, which handle the key task of parsing a mailbox into a series of messages. I've used it before, and it suited my needs for this quick hack.

The first version of my view-useperl script looked something like this:

```
#!/usr/bin/perl -w

use strict;
use Mail::Box::Manager;

## (1) Download the mailbox from the ISP.
my $mbox = "/tmp/useperl.$$";
system("scp $ENV{USEPERL_MAILBOX} $mbox")
     and die "Error downloading mailbox.\n";

## (2) Open the temporary mailbox.
my $manager = new Mail::Box::Manager;
my $folder = $manager->open(folder => $mbox);

## (3) Convert the mailbox from a list of messages
## to a list of URLs. (Find the first URL in each message
## body.)
my $i = 0;
my @urls;

while(1) {
     my $msg = $folder->message($i++);
     last unless $msg;

     $msg->body() =~ m/(http:.*?)$/sm;
     push(@urls, $1);
}
```

```
## (4) Close and delete the temporary mailbox.
$folder->close();
unlink($mbox);

## (5) Show the URLs, one at a time.
foreach my $url (@urls) {
      print $url;
      system "open '$url'";
      <>;
}
```

The main body of this script starts in part (1) by copying the mailbox from my ISP to a local file. The remote location of the mailbox containing my use.perl.org notifications is stored in an environment variable, USEPERL_MAILBOX. I use this technique to avoid hard-coding sensitive information in my source code. Because the location is stored in an environment variable, I can publish the source code without requiring other users to edit the program text in order to customize it.

The system call may seem counterintuitive at first. That's because system is unlike normal Perl primitives, like open, that return true on success and false on failure. Instead, system returns a nonzero failure code (true), and a zero value (false) to indicate success. This is consistent with the behavior of the system call in the standard C library. Although it made sense for Perl to adopt the C-style behavior many years ago when most Perl programmers had some background in C, the situation is quite the opposite today. Perl6 will reverse this legacy behavior so that the standard "open or die" idiom can be used with system as well, and do away with the current, counterintuitive "system and die" usage.

The code in part (2) creates a Mail::Box object to scan through the messages in the mailbox I just downloaded. The Mail::Box interface requires that I first create a manager object, and use a factory method on that object to create a folder object, $folder.

The while loop grabs each email message in the folder, one at a time, and terminates when there are no more messages to retrieve. It then locates the first URL in each message body, and appends that to a list of URLs.

Part (4) is some basic housekeeping code to close the folder object and delete the local copy of the mail folder.

The real value of this program is found in part (5). Finally, I have a list of URLs I want to load in my browser. Because MacOS X is my platform of choice, I use the standard open command to open each URL. This command will intelligently open a filename using the appropriate application. In this case, when I pass a URL to the open command, it will load that Web page in my preferred Web browser. On another system, I could write a small program called open that is just smart enough to take a URL and load it in a Web browser. Alternatively, I could replace open with a mozilla -remote or similar command.

I usually run this program when a few dozen emails have accumulated in my mailbox. Opening a few dozen browser windows all at once is a great way to consume a lot of memory, saturate my network connection, and generally make my computer quite sluggish. Instead, I ask for a line of input after each Web page is loaded. This allows me to load URLs quickly, yet only open a few at a time. The input is irrelevant, so a simple return will allow the program to continue and load the next URL.

Problems with Dependencies

As I mentioned before, I wrote this script about two years ago. Since then, my main computer died, I purchased a replacement, reinstalled an OS on my laptop a few times, and have at least two Perl installations on each machine I regularly use (both version 5.6.x and 5.8.0). From a systems management perspective, it's been an eventful few months.

My view-useperl script is always one of the first things I install in my home directory on a new machine. With all of the shuffling, I've come to regret using Mail::Box for this little script. After all, if it only took 10 minutes to write, why should I need to spend another few minutes on each new machine to install a dependency when I use my program on a new machine? Mail::Box certainly is *not* a bad module, but this is the only program I use that requires it. Eliminating the dependency will make it easier to copy my script around as I switch machines (and Perl installations).

Once I came to this realization, I saw it was time to rewrite my program to not use Mail::Box anymore, and just process the mailbox files directly.

Second Attempt: No CPAN Modules

Mail messages in a mailbox file start with a line that contains the word From, a space, an email address, and a date. (This is why a line in the body of an email message that begins with "From" usually has a ">" character preceding it.) Mail messages are also separated by a blank line preceding the From line.

With this information at hand, I decided to rewrite view-useperl. I can view a mailbox file as a series of email records, and then use standard Perl operators to convert a mailbox file into a sequence of relevant URLs.

The email messages I am processing are consistently formatted. The first URL in each message is the location of the journal entry I want to view. Also, there are no URLs found in the message headers, so I don't even need to bother splitting the message header from the message body. The first URL I encounter in each message will be the URL I want to view. Subsequent URLs will appear in each message, and they should be ignored.

Here is the updated view-useperl script that takes advantage of these observations:

```
#!/usr/bin/perl -w

use strict;

sub read_mailbox {
      my $mbox = shift;

      ## Read a sequence of messages,
      ## delimited by a blank line and 'From'.
      local $/ = "\n\nFrom ";

      open(my $fh, $mbox);
      my @urls = map {m/(http:.*?)$/sm; $1} <$fh>;
      close($fh);
      unlink $mbox;

      return @urls;
}

## (1) Download the mailbox from the ISP.
my $mbox = "/tmp/useperl.$$";
system("scp $ENV{USEPERL_MAILBOX} $mbox")
 and die "Error downloading mailbox.\n";

## (2) Read the URLs found in the mailbox.
my @urls = read_mailbox($mbox);

## (3) View each URL in turn.
foreach my $url (@urls) {
      print $url;
      system "open '$url'";
      <>;
}
```

Not only does this program avoid CPAN modules, but it is slightly shorter and a little easier to read. Because it does not have any external dependencies, I can expect it to work wherever I copy it (so long as I define my USEPERL_MAILBOX environment variable).

The overall structure of this is unchanged. First, fetch the mailbox. Next, convert a set of email messages to a list of URLs. Finally, display the URLs, one at a time. And this sequence of steps is clearly stated in the main program text. The more complicated process of converting a mailbox into a list of URLs is now handled by an appropriately named sub, read_mailbox. This sub eliminates the need to create Mail::Box objects, yet it performs the same task: converting a mailbox into a set of email messages, extracting the first URL from each message, and returning a list of URLs.

The first thing read_mailbox does is modify the input record separator, stored in the $/ special variable. This variable contains a new-line character by default, and that is why file input operations generally occur one line at a time. Because a mailbox is nothing more than a concatenation of email messages, I can specify a sequence of characters found between email messages as a record separator. When I read the mailbox file in list context, I receive a meaningful list of email messages, not a meaningless list of lines in a file.

The $/ variable is used globally within a Perl program. Changing its value is bad style, unless changes are localized to a specific scope. Here, the statement local $/ = "\n\nFrom " modifies the value of the input record separator within the read_mailbox sub and any subs that it calls. When read_mailbox exits, the previous value of $/ is restored. This is important, because the console input operation at the end of the program expects the record separator to be a new line, not "\n\nFrom ".

Now that $/ has been modified appropriately, reading messages is a breeze. First, I open the local copy of the mailbox. Next, I read the mailbox in as a list of messages (my @urls = ... &lt;$fh&gt;;). But the messages themselves aren't very meaningful in this particular program. In fact, each message is just a stream of meaningless text, followed by a URL, followed by more meaningless text. The map operation transforms the list of email messages read in from &lt;$fh&gt;; and replaces each chunk of text with the first URL in that chunk. These URLs are then gathered together into the list @urls.

In effect, a few lines of module initialization and method calls in the original program are replaced with one line to reset the value of $/ and one line to read and convert a mailbox into a list of URLs. This is the power of Perl.

Observations

CPAN is certainly the best thing that has ever happened to Perl. However, Perl without CPAN is not too shabby. Sometimes, the easiest or the best solution to a problem is to *avoid* CPAN. In the example of my view-useperl program, the before and after versions are equally good. However, the modified version has a very attractive property – it does not require me to install Mail::Box on each new computer I use.

The reason why I chose to avoid Mail::Box has nothing to do with the quality of that module, but just reflects the desire to remove an external dependency for view-useperl. If my little program were doing something more complicated, like deleting some messages in a mailbox file, or selecting messages based on header characteristics, then I certainly would have kept using it. Instead, I chose to rewrite this program as a common needle-in-a-haystack type of solution. The fact that the data file was a mailbox is largely irrelevant here.

There are other circumstances where avoiding CPAN modules is simply unwise. When using a relational database, there's no good reason to avoid the DBI family of modules. Similarly, there's no reason to start with raw socket programming to fetch Web pages when the LWP library has been doing a great job for many years. And the list goes on and on.

Conclusion

Whether you are writing a quick hack or a program of significant size, there are many very good reasons to start with CPAN modules to make your job easier. However, there are also some circumstances where there are benefits to avoiding CPAN modules. Remember that both options are available to you. Choose wisely.

USENIX and SAGE Need You

People often ask how they can contribute. Here is a list of tasks for which we hope to find volunteers.

The SAGEwire and SAGEweb staff are seeking:

- Interview candidates
- Short article contributors (see *http://sagewire.sage.org*)
- White paper contributors for topics like these:

Back-ups	Emerging technology	Privacy
Career development	User education/training	Product round-ups
Certification	Ethics	SAGEwire
Consulting	Great new products	Scaling
Culture	Group tools	Scripting
Databases	Networking	Security implementation
Displays	New challenges	Standards
Email	Performance analysis	Storage
Education	Politics and the sysadmin	Tools, system

- Local user groups: If you have a local user group affiliated (or wishing to affiliate) with SAGE, please email the particulars to *kolstad@sage.org* so they can be posted on the Web site.

;login: always needs conference summarizers for USENIX conferences. Contact Alain Hénon, *ah@usenix.org*, if you'd like to help.

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

Summertime, but is the living easy? As I write, the world is in turmoil, reeling from the effects of the past two US national elections. A very un-Republican interest in world affairs, coupled with increased repression in the "homeland" the neo-conservatives claim to be defending. Even the term "homeland" conjures up connections with another country which used the term "motherland."

For an example of repression, consider the Pennsylvania law that permits the attorney general to force worldwide Internet service providers to block access to a list of IP addresses. The intent of this law is lofty – to deny access to sites containing child pornography. But the side-effects include blocking access to any site co-hosted at the blocked IP address. The attorney general of Pennsylvania has refused to release this list of addresses, claiming this would be tantamount to providing access to child pornography. I wonder how this can be true if those addresses have been blocked?

Other non-American news sites have been blocked, whether from vigilante activity or perhaps some unannounced official interference. Democracy relies on freedom of speech, and blocking access to news sites that contain information that does not agree with a particular perspective is un-American.

And the economy? Let's not go there.

Instead, I'd like to write about Sendmail. Twice in March 2003, buffer overflows were revealed in Sendmail. It is hard to imagine that anyone reading this column wouldn't be aware of this, as most sysadmins (and those running their own UNIX/Linux systems) scrambled to get patches installed on all systems running Sendmail. With Sendmail running as root, and accessible through firewalls (or via internal relays, which would work just as well), there were lots of vulnerable systems around.

Connecting to Sendmail from the network can provide useful information:

```
220 spirit.com ESMTP Sendmail 8.12.8p1/8.12.10; [date removed]
```

Installing the patches provided by the Sendmail Consortium does update the version number and patch level received when you connect to port 25/tcp. The second part of the version information comes from whatever you have entered in the sendmail.cf file (or the macros that are used to construct it) to define the Z macro. I decided that my version of Sendmail would be a bit more advanced than most...

The first buffer overflow appeared in the crackaddr() function, which Sendmail uses to canonicalize addresses. The non-patched version was quite complex, and that was where the trouble lay. Each time a left angle-bracket was seen, a flag was set, and the number of bytes should have been decremented by one – but wasn't. When the right angle-bracket was encountered, the counter got incremented by one, making the overflow possible.

The Polish hacking group Last Stage of Delirium came up with an example exploit that would only work on Slackware 8. To be honest, a lot of us spent time trying to crash our versions of Sendmail just to see if they were vulnerable. Upgrading to Sendmail 8.12.8 meant (for many sites) upgrading to a new version of the config file as well, version 10. So the easiest path, replacing the Sendmail binary with the newest version, was not easy.

The LSD exploit hadn't been posted to their Web site (*http://lsd-pl.net*) but was posted to Bugtraq and archived there when I wrote this: (*http://archives.neohapsis.com/archives/bugtraq/2003-03/0054.html*). The exploit involves sending 138 pairs of <>s, followed by a set of 0xf8s surrounded by parentheses (a pathological comment), and followed by the address to be overwritten. The exploit is not a simple one, as crackaddr() uses a statically defined buffer that gets stored on the heap instead of the stack. The consequences of this are twofold: first, that mechanisms that defend against stack-based exploits fail; and second, that any exploit writers have a much more difficult time.

The LSD programmers had an additional "concern" in that most Sendmail servers will be protected by firewalls that will only permit incoming connections to the server on port 25/tcp. Though this is a thoughtful idea, I doubt it is true. At any rate, their exploit could not be written to include a standard backdoor shell, by listening at a port and exec'ing a shell. Instead, their exploit connects back to the exploit program and executes uname -a. Most firewalls allow outgoing connections, and the LSD exploit actually assumes that a Sendmail server can reach port 25/tcp at arbitrary IP addresses (very reasonable).

The example exploit, or Proof of Concept (PoC) as such exploits are now euphemistically called, failed even to crash a RedHat 7.3 version of Sendmail that reportedly was vulnerable. Part of the problem is that heap exploits rely on the layout of memory allocated on the heap and finding the right set of bytes so that freeing a block of memory passes control of the program counter to the shell code. The LSD exploit passed their shell code as a very long line (about 2k) that gets sent right after the "Subject:" line, but with no intervening blank line (part of the normal message format defined in RFC 822).

The second buffer overflow, found by Michael Zalewski, involved a similar problem but in a different function, prescan(). Prescan() tokenizes addresses by looking for delimiters, and during this process a special value, 0xff, gets skipped, decrementing a counter. By providing specially formatted addresses, one could overflow the buffer used, pvpbuf[], which gets allocated on the stack. This buffer has a default size of MAXNAME plus MAXATOM, or 1256.

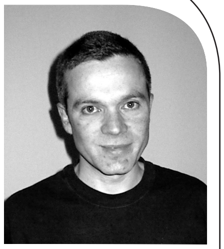
What these two exploits have in common is that they violate limits suggested in RFC 821 (*http://www.faqs.org/rfcs/rfc821.html*). If you read section 4.5.3, the suggested maximum size of an address is 256 characters, and of any line, 1000 characters. The LSD exploit sends an address that is about 300 characters long, along with a very long line containing the shell code. The vulnerability found by Zalewski appears to require an address in excess of 1256 characters, again outside the suggested limits. Note that the RFC does state that these limits are suggestions only, and that MTAs could be written that handle larger addresses and lines.

Still, sites employing application gateway (AG)-based firewalls, with the SMTP AG actually enabled, would have blocked both of these attacks without modification or updating. I actually contacted several firewall vendors (SecureComputing, Symantec, and Watchguard) and asked if their firewalls could block the LSD exploit. All three claimed that their AG firewalls (each vendor has multiple products) could block this attack if the AG were used. That's some good news, at least. One vendor, Symantec, also blocked by default access to WebDAV, another vulnerability (in IIS 5) announced in March, with the potential to be the next base for Code Red version 8.

I wonder how the patching wars have gone between the time I wrote this column and the time you will be reading it. Patching is never easy, but it is definitely easier if you have built an infrastructure for safely and reliably distributing and installing patches. I certainly wish all of you good luck, whatever the fortunes of war may bring.

organized pruning of file sets

by Diomidis D. Spinellis



Diomidis Spinellis is an Assistant Professor in the Department of Management Science and Technology at the Athens University of Economics and Business; he is the author of *Code Reading: The Open Source Perspective* (Addison Wesley, 2003).

dds@aueb.gr

In a number of backup scenarios, backup files simply accrue in a directory that should be periodically cleaned up. Typical examples include backup data from databases, PDAs, network clients, and routers. The data contents of the above are often not directly backed up onto tapes, but are copied to disk-based files by an appropriately scheduled cron(8) job.

The continuous increase in disk capacities enables us, in many cases, to keep multiple sets of these backup files in a given directory as a substitute for a regular tape backup. These files are typically kept in a manageable size by periodically purging all old files. Well-organized tape-based backups, however, offer an additional advantage: Through a carefully staged tape retention schedule, users can often retrieve files much older than the number of retained tapes would suggest.

A tape backup schedule might, for example, involve daily incremental backups, weekly full backups retained for two months, and monthly tapes retained for two years. If I discover today that sometime in the previous six months I deleted a file I had created a year ago, I can go to the retained monthly backups that followed the file's creation and retrieve it from there.

Backups to files are not often organized in this manner. Two approaches I have seen for managing their size involve either naming each file with a periodically repeated date element, such as the day of the week or month, so that newer files will overwrite older ones, or tagging each file with a unique identifier, such as the complete date, and having a separate script remove files older than a given date. Both approaches, however, lack the property of selectively retaining a subset of older files. More elaborate schemes can, of course, be constructed by carefully synchronizing and staging separate cron(8) jobs, but I have never seen them applied in practice. The problem of selectively retaining old files gets especially difficult when the backups are created at irregular intervals – for example, each time I synchronize my PDA or remember to back up my cellular phone directory.

On the other hand, a file-based backup scheme offers the additional possibility of automatically examining all the retained files and selectively pruning those we decide are not worth keeping. The key concept for deciding which files to keep is a *retention schedule*. In tape-based schemes, this simply revolves around weeks, months, and years. If we have a tool for managing the file pruning, we can be more creative in selecting a retention schedule and, hopefully, use one that will, violating Murphy's Law, offer us an increased probability of recovering that old file we discovered missing.

Retention Schedules

When I first decided to work on the file pruning problem, I considered using an *exponential* retention schedule. I would like to keep yesterday's backup, a backup from two days ago, then backups aged 4, 8, 16, 32, and 64 days. With 10 files I could cover a period lasting more than a year. This schedule uses two as the schedule's base; one could select any smaller number to increase the number of retained files or a larger number to decrease them. The idea behind this schedule is that recent backups are more valuable than older ones.

Creeping featurism made me think of different possible schedules. One other possibility is a Fibonacci schedule. Here the retention sequence starts with 1, 1, and each subsequent term is the sum of the two previous ones: 2, 3, 5, 8, 13, 21 34, 55. At that point, I had to wonder which of the two schedules was better for securing my valuable data.

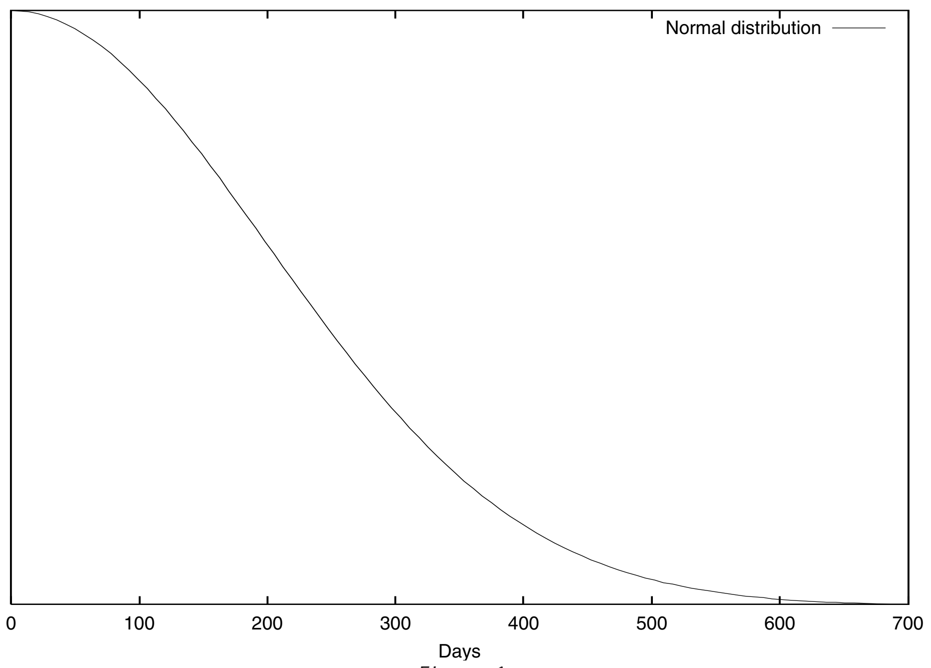


Figure 1

It turns out that neither is. If we were to sample data recovery requests in a large data center, we would probably find that the age of the requested files would follow the ubiquitous bell-shaped *normal*, or *Gaussian*, distribution. The exact shape of the bell is determined by the *standard deviation* of the requested file ages; this expresses the variation (in the same unit as we measure the file ages) between the ages of different requested files. Since recovery requests from all users (apart from pointy-haired managers) always refer to the past, the shape is actually one-half of a bell curve. You can see the normal curve for a standard deviation of 200 in Figure 1.

The formula defining the normal curve is actually quite complex:

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{-x^2}{2\sigma^2}}$$

but once it is coded in a program, its application can be a breeze. The curve represents the probability that a file of a given age will be requested.

You can see that, following intuitive expectation, as files age they are less likely to be needed. In order to distribute our archive files in a way that reflects this diminishing probability distribution, we need to define our retention interval schedule so that the interval's length is proportional to the probability of requiring a file within that interval. This is represented by the area under the curve for the given interval; for the mathematically inclined, the area for an interval from `a` to `b` is given by the integral

$$\int_a^b f(x)dx$$

We therefore need to divide the whole area under the curve into a number of equally sized parts, as many as the files we can afford to retain, and then calculate the respective intervals.

Unfortunately, there is no mathematical formula with a finite number of terms that can give us the numbers we are looking for. Initially, I wrote code to numerically integrate the normal function, adjusting the interval while moving back into time. A few days later, my colleague Stavros Grigorakakis, reading a draft of these notes, pointed me to an excellent analysis of the Gaussian function available online at *http://mathworld.wolfram.com/GaussianDistribution.html*. There I found that the cumulative distribution function (the integral I was painstakingly calculating) can be determined by means of the so-called *error function*, which – surprise, surprise – is part of the UNIX C math library. You can see in Figure 2 how we would spread 30 files in a period of around 2000 days using an exponential distribution with a base of 1.3 and a normal distribution with a standard deviation of 1000. For comparison purposes, I have also included how a Fibonacci distribution and an exponential distribution with a base of 2 would appear in the above scheme; only 18 files would fit in the Fibonacci distribution and 12 in the base-2 exponential.

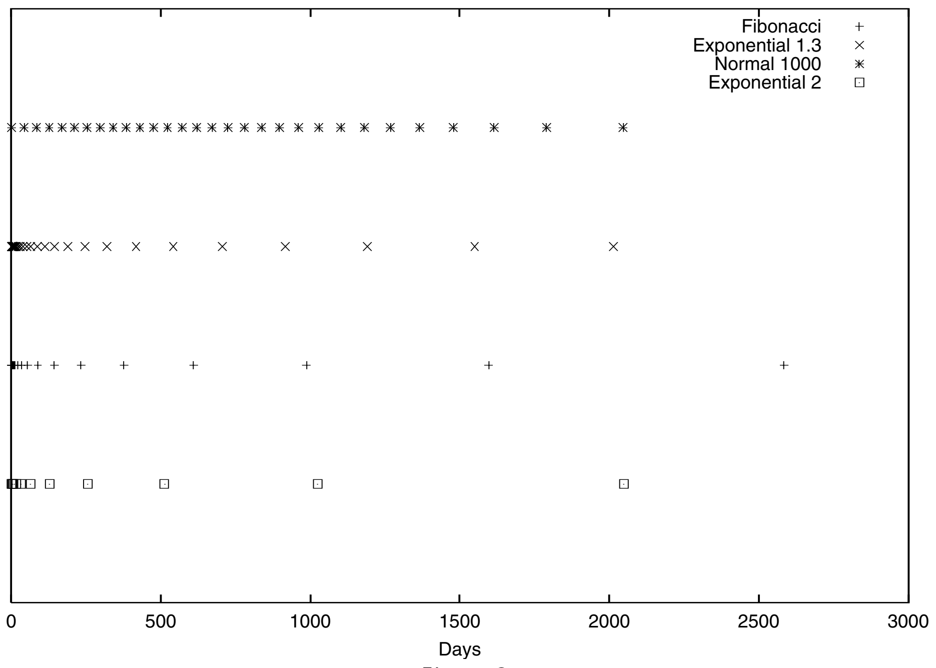


Figure 2

The Prune Tool

Putting code where my mouth is, I wrote a C program to implement the file-pruning strategies described above. It is available for download in source form through a BSD-style license from *http://www.spinellis.gr/sw/unix/prune*. *Prune* will delete files from the specified set, targeting a given distribution of the files within a certain time, while also supporting size, number, and age constraints. Its main purpose is to keep a set of daily-created backup files in manageable size while still providing reasonable access to older versions. Specifying a size, file number, or age constraint will simply remove files starting from the oldest, until the constraint is met. The distribution specification (exponential, Gaussian, or Fibonacci) provides finer control of the files to delete, allowing the retention of recent copies and the increasingly aggressive pruning of the older files. The retention schedule specifies the age intervals for which files will be retained. As an example, an exponential retention schedule for 10 files with a base of 2 will be:

1 2 4 8 16 32 64 128 256 512 1024

This schedule specifies that for the interval of 65 to 128 days there should be (at least) one retained file (unless constraints or other options override this setting). Retention schedules are always calculated and evaluated in integer days. By default *prune* will keep the oldest file within each day interval, allowing files to gradually migrate from one interval to the next as time goes by. It may also keep additional files, if the complete file set satisfies the specified constraint. The algorithm used for pruning does not assume that the files are uniformly distributed; *prune* will successfully prune files stored at irregular intervals.

Prune is invoked through the following syntax:

prune [**-n**|**-N**|**-p**] [**-c** *count*|**-s** *size*[**k**|**m**|**g**|**t**]|**-a** *age*[**w**|**m**|**y**]]
[**-e** *base*|**-g** *standard deviation*|**-f**] [**-t** **a**|**m**|**c**] [**-FK**] *file ...*

The numerous options reflect the tool's flexibility. You can specify the distribution to use (exponential, Gaussian, or Fibonacci) using the -e, -g, and -f options as well as the constraints for the number (*count*), *size*, or *age* of the files to retain using the -c, -s, and -a options. By default the constraints are used to specify the upper limit of the size or number of files that will be retained. If more files can be accommodated (because, e.g., An important aspect of a disk-based backup system is the employed retention schedule.

some intervals are empty), or the specified size limit has not been reached, *prune* will retain additional files, deleting old files until the constraint is satisfied. The -F flag can be used to override this behavior. On the other hand, if a constraint is violated, *prune* may not retain any files in a given interval; the -K flag can be used to always keep at least one file in each interval. Finally, the -t flag allows you to specify whether *prune* will use the creation, access, or modification time of the specified files for determining their age.

The following examples illustrate some possible uses for *prune*:

```
ssh remotehost tar cf - /datafiles \ >backup/`date +'%Y%m%d'`
prune -e 2 backup/*
```

backs up remotehost, storing the result in a file named with today's timestamp (e.g., 20021219). Subsequently, prunes the files in the backup directory so that each retained file's age will be double that of its immediately younger neighbor.

```
prune -g 365 -c 30 *
```

keeps at most 30 files. The ages of these files will follow a Gaussian (normal) distribution, with a standard deviation of one year.

```
prune -e 2 -s 5G *
```

prunes the specified files following an exponential schedule so that no more than 5GB are occupied. More than one file may be left in an interval if the size constraint is met. Alternatively, some old intervals may be emptied in order to satisfy the size constraint.

```
prune -F -e 2 -s 5G *
```

acts as above, but leaves no more than one file in each scheduled interval.

```
prune -K -e 2 -s 5G *
```

acts as in the first example of the 5GB-constrained series, but leaves exactly one file in each interval, even if this will violate the size constraint.

```
prune -a 1m -f
```

deletes all files older than one month; it uses a Fibonacci distribution for pruning the remaining ones.

Conclusion

Increasing disk capacities and network bandwidth allow us to implement disk-based backup mechanisms. An important aspect of a disk-based backup system is the employed retention schedule. The prune tool allows you to rationally specify and automatically manage the retention schedule to suit your needs. An exponential schedule with an integer base or a Fibonacci-based schedule can be easily understood by unsophisticated users, while a schedule with a normal distribution and an appropriately set standard deviation is more likely to reflect your true file-retention requirements.

ISPadmin

Bayesian Spam-Filtering Techniques

by Robert Haskins

Robert D. Haskins is an independent consultant specializing in the Internet Service Provider (ISP) industry.

rhaskins@usenix.org

and Rob Kolstad

Rob Kolstad is currently Executive Director of SAGE, the System Administrators Guild. Rob has edited *;login:* for over ten years.

kolstad@sage.org

Introduction

In this installment, Rob Kolstad and I look at the arrival of what has become known as the "Bayesian" spam-filtering technique. The reason Bayesian filtering is so intriguing is its high accuracy rate (for Haskins, 97.6% as tracked by POPFile; for Kolstad, approaching 99.8%) and low false-positive rate (estimated by Haskins at 1% or less, again with POPFile; Kolstad is getting 0.1%), making it one of the most useful anti-spam tools available to date. Before getting into how specific Bayesian implementations work, it is useful to have a good understanding of how Bayesian theory (which has been around since the 18th century) works.

While the vast majority of the discussion revolves around spam, the Bayesian technique can easily be used to augment (or replace) the "usual" filters found in mail client programs. Also, it is likely this sort of filtering functionality will be integrated into the email client program, eliminating the need for proxying except under nonstandard circumstances.

Bayesian Theory

Regrettably, "Bayesian theory" is not precisely the heart of the statistical spam-filtering methodology under discussion, but the name is used universally for this technology. Briefly, let's enter the world of probability theory for a quick refresher. If you are not a math or statistics aficionado, please skip to the next section.

Recall that probabilities assign a numerical value to the belief (or observation) that an event might or might not happen. The probability of a coin flip ending up "heads" is 0.50 – half the time one flips a coin, one should get "heads." (Certain coins are weighted so that this doesn't happen, but they are less interesting to use for this example.) The notation for simple probability is also simple:

```
P('heads') = 0.5
```

This is read "The probability of 'heads' is 0.5."

Probabilities are interesting when they help one deduce facts or understand information better. Knowing that using pharmaceutical P1 cures disease D1 99% of the time is a good piece of data to have when one has disease D1. Knowing that pharmaceutical P2 cures D1 only 1% of the time might very well influence a decision to use P1 instead of P2.

Conditional probability is just a little more complex. Conditional probability addresses the probability of an event's occurrence **given that some other event has occurred.** Consider the probability that one has cancer given that one is a regular smoker of cigarettes. This is written:

```
P (cancer | smoker) = [some number]
```

This is read "The probability of 'cancer' given 'smoker' is . . . ."

Some interesting math about conditional probabilities. Consider the two probabilities:

```
P (cancer | smoker)
```

P (cancer | nonsmoker)

For this discussion, these are all the possibilities that were “observed” or “measured.” Summing these with the probabilities of smoking tells us the total probability of cancer:

P(cancer) = P (cancer | smoker) x P(smoker) + P (cancer | nonsmoker) x P(nonsmoker)

Rev. Thomas Bayes published an article in 1763 with a more advanced formula. It concerns computing probabilities for a hypothesis given a potentially new set of observations. Consider a new observation “O” and its impact on a hypothesis “H”.

$$P(H \mid O) = \frac{P(H) \times P(O \mid H)}{[P(H) \times P(O \mid H) + P(\text{not-}H) \times P(O \mid \text{not-}H)]}$$

Which is read: “The probability of our hypothesis ‘H’ given the new observation ‘O’ is . . .” and then the formula. The various components:

- P(H) is the probability we had assessed before the new observation showed up.
- P(O | H) is the probability of the observation given that the hypothesis H is true.
- P(not-H) is just 1 - P(H), the probability that H is not true.
- P(O | not-H) is the probability of the observation in the world where the hypothesis is false.

Let’s use an example to demonstrate the formula. Suppose a disease appears in 1% of the population:

P(disease) = 0.01
P(not-disease) = 0.99

and let’s presume that a test for the disease is positive 80% of the time when the disease is present and 10% of the time when the disease is not present:

P(positive test | disease) = 0.80
P(positive test | NOT disease) = 0.10

Then we can answer the question, What is the probability of the disease if the test on someone is positive?

Here’s the formula:

$$P(H \mid O) = \frac{P(H) \times P(O \mid H)}{[P(H) \times P(O \mid H) + P(\text{not-}H) \times P(O \mid \text{not-}H)]}$$

Filling in the values from above (H is “has the disease”; O is “positive test”):

$$\begin{aligned} P(\text{disease} \mid \text{positive test}) &= \frac{0.01 \times 0.80}{[\ 0.01 \times 0.80 + 0.99 \times 0.10]} \\ &= 0.074766\ldots \\ &= \sim 0.075 \end{aligned}$$

Thus, we can conclude using Bayes’ theorem that a positive test for the disease gives one only a 7.5% chance of actually having the disease – surprisingly less than one might expect!

Why is it so low? Because 99 out of 100 people don’t have the disease, yet 20% of those 99 people will test positive. This huge number dwarfs the small percentage who actually have the disease.

Bayesian Spam Filtering

Spam prevention methods use Bayes' theorem in its chain form, with ever more observations entering into the formula until a final probability is assessed. Of course, in the spam-prevention case, the hypothesis is "This note is spam." It is the observations that make the result interesting.

The key paradigm in Bayesian spam prevention is the observation of sets of words or combinations of words. Consider observations of the form:

- A single word (e.g., "Viagra" or "enlargement")
- Pairs of words (e.g., "money fast" or "bank account")
- Triples, quads, or longer sequences of words (e.g., "make money fast," *friend@public.com*)
- Header information tokenized in a clever way. Here's a header:
  `Message-ID: <20010214170157.C323@ubiqx.mn.org>`

One of Paul Graham's experiments tokenizes pairs like this:

```
Message-ID 20010214170157
Message-ID C323
Message-ID ubiqx
Message-ID mn
Message-ID org
```

The imagination is the limit on these!

Bayesian methods read (usually after de-MIMEing) email (in whole or in part) and try to deduce if the mail is spam or not. They tokenize the mail (often removing punctuation; sometimes – but, these days, more often not – removing capitalization), create a zillion observations (e.g., every possible run of 1, 2, 3, 4, or 5 words and every properly ordered subset of that list), and then look up these observations in a huge database. The probabilities are all combined (when they exist), and a final result is determined. This is compared to a threshold to determine whether the program believes the message to be spam.

Note that while this sounds simple, the numbers involved theoretically range from 1.0 down to unbelievably small numbers for combinations that only show up once. The software must be careful that probabilities smaller than 1.0e-350 or so don't end up rounding to 0.

Initial probabilities are calculated by running a set of training mail (both spam and non-spam) through a program that tokenizes and calculates probabilities for the various phrases that appear. This is interesting because some industries use the term "offer" a lot, a word commonly used in spam. By training with longer phrases, a database of probabilities can be built with relatively small amounts of very representative good and bad email.

By way of example, Kolstad's training base now has these statistics:

Spam email: 317 messages, 130,770 lines of text (3.3MB)
Non-spam email: 63 messages, 55,133 lines of text (0.75MB)

These training bases have been augmented over time by adding to them each time the filter makes a mistake.

The key paradigm in Bayesian spam prevention is the observation of sets of words or combinations of words.

The beauty of a statistical approach to identifying spam is that it "learns" the methods spammers use to get past more static filters.

History of Bayesian Theory as It Relates to Spam

Interestingly, Microsoft was granted a patent for a method of filtering spam using the Bayesian methodology (USPTO patent # 6161130). However, this work was preceded by at least two other published works. One is "ifile," a filter for a number of command-line-based mail programs including MH, Pine, and procmail-compatible clients. It is unclear what, if anything, Microsoft did with this work and associated patent.

The problem in the past with the Bayesian method was that the accuracy of various software implementations wasn't nearly close enough to 100%. Users complained about both false negatives (spam delivered as legitimate mail) and false positives (real email that was incorrectly identified as spam). Few users will tolerate more than a minuscule false positive rate (0.5% may actually be too high). Of course, the more spam there is, the more such tolerance might go up. Now that everyone is inundated, those sending important mail know that it can get lost in the flotsam and jetsam.

What makes newer algorithms (implementations, anyway) different? According to Paul Graham's follow-up paper, "Better Bayesian Filtering," things are improving because of:

- The size of the body of training mail
- Using mail headers as tokens or not
- The message tokenization methodology
- Improved probability calculations
- More intelligent bias against false positives

The beauty of a statistical approach to identifying spam is that it "learns" the methods spammers use to get past more static filters. As a result, spammers are forced to make their messages look just like your regular email, thereby reducing the effectiveness of spam. This could cause spammers to stop spamming!

Likewise, as Paul Graham points out, eliminating 99% of delivered spam would presumably eliminate 99% of the responses a spammer seeks (i.e., to sell a product or service). Few businesses can continue with a 100 times decrease in revenue. Maybe that would cause them to stop spamming. We can only hope.

PC Email Proxy Client (POPFile)

POPFile is a Perl-based POP3 proxy that runs on your PC, between your email client program (such as MS Outlook, Eudora, etc.) and the POP3 server (or anti-virus software). It takes a little bit to set up and train, but once it has a "corpus" of spam, it works very well.

The concept of POPFile is this. First, you set up your email client to work with the program. Consult the POPFile Web site for detailed instructions on how to configure POPFile for various email clients, including MS Outlook, Outlook Express, and Eudora. Also, consult the HOWTO discussion group for other setup notes, including working with virus scanners, setting up Netscape Mail, etc.

Once your email client has been configured, you begin categorizing email per the "buckets" (classifications) you set up. Haskins created two buckets, one called "spam" and one called "notspam." There is no reason why you cannot create other buckets for other classifications of email to enhance your email client's filtering capability.

When your mail client checks mail, POPFile receives the message, runs it through its filters, and either tags the subject line with the bucket name or adds a header to the message "X-Text-Classification: <*bucket name*>" if your email client can filter based on arbitrary mail headers.

Unlike some other filtering programs, it specifically does not come with a preloaded corpus of spam. Instead, the program builds a very specific corpus for your incoming email, making it very accurate.

POPFile's weaknesses include:

- No support for IMAP
- Issues with clients who leave their mail on the server
- No mechanism for inputting a preexisting corpus of spam and non-spam to "jump start" the filtering

POPFile is certainly an excellent implementation. If you have a POP3 mailbox, and don't store your mail on the server, we heartily recommend it!

It is very likely that Bayesian-style filtering will be implemented in other email client software. The Mozilla folks are hard at work implementing Bayesian filtering in their Mail application. Check out Mozilla version 1.3 alpha if you are interested. While initial reports indicate the functionality doesn't work perfectly, it is a step in the right direction. The SpamAssassin folks have also started using Bayesian filtering.

Command-Line Email Client Support

Bayesian solutions exist for the server and command-line email clients such as Pine. Most of these are hooked in via procmail, so if you host your mail on your own server, then you are set. If not, ask your provider if it can activate procmail for you.

Some Bayesian command-line email client filters include ifile, SpamProbe and bogofilter. More information on these filters is available on their respective Web sites.

Server Side

There are several server implementations of Bayesian filtering. Most are easiest to implement via SpamAssassin or procmail.

Probably the best known is CRM114, which, incidentally, did not start out as a Bayesian filter but as a rather large regular expression matcher. However, Bayesian filtering was added to the mix as of November 2002.

Your best bet is probably to implement the native SpamAssassin Bayesian filters available in 2.50, or to attempt the CRM114 hooks to provide Bayesian filtering for an entire server. Be aware that this is bleeding-edge software, and plan accordingly.

Experience with CRM114

Kolstad's mail shows up the old-fashioned way: It is delivered to a mailbox on his desktop. He configured his `.forward` file like this:

```
"| /home/kolstad/bin/nospam"
```

and then created a simple Perl program to run the mail against CRM114:

- Read in the message.
- Run the message through CRM114 with the spamfilter configuration file.
- Examine the exit code to deduce if the message is spam.
- Non-spam messages are appended to the mailbox, with locking; biff is called with a seven-line networking sequence.

- Spam messages are either tagged (for testing!) or diverted to another file for later review.

He has been running this system unmodified for about two months. Two additional scripts augment the two training databases ("this is spam"; "this is not spam") with mail that was misclassified.

He gets 250 emails per day, unless something special is happening (e.g., broken salary survey, some other SAGE member interaction, etc.). Right now, the spam filter is missing about one spam per day (false negative) and tagging one or two emails per week as spam. Happily, those emails really do resemble spam and would not have been missed.

The nospam program itself requires an average of 55 ms per message; CRM114 averages 270 ms. Incoming mail suffers only the smallest of delays.

Note well that the CRM114 distribution out-of-the-box did not perform as hoped (though it compiled with no problems). Some of the scripts have been rewritten (one had a huge regular expression converted to a Perl program that ran 100–1000 times faster) and converted for spam handling. Likewise, the new scripts dramatically ease the user interface.

The absolute best part of CRM114 (and Bayesian spam detection in general) is that it does not require the relatively frequent updating that SpamAssassin really needs. Half-a-dozen "retrainings" (a single keystroke on clever mail readers) per week keep it in good shape.

If you're interested in packaging this solution for simple and more widespread distribution, please contact Kolstad.

Conclusion

The Naive Bayesian method as outlined by Paul Graham is a very effective way to filter spam. The algorithm has a very high accuracy rate and a decreasingly small false positive rate, making it the best currently available method to identify spam. There are PC client implementations (including POPFile) as well as UNIX-based packages (CRM114, bogofilter, SpamAssassin plug-in) available.

REFERENCES

POPFile home page: *http://popfile.sourceforge.net/*

Paul Graham's "A Plan for Spam": *http://www.paulgraham.com/spam.html*

International Society for Bayesian Analysis (ISBA): *http://www.bayesian.org/*

Microsoft Bayesian spam-filtering patent: *http://patft.uspto.gov/netacgi/nph-Parser?patentnumber=6161130*

Old ifile README: *http://www.ai.mit.edu/~jrennie/ifile/old/README-0.1A*

New ifile site: *http://www.nongnu.org/ifile/*

The RAND MH Message Handling System: *http://www.ics.uci.edu/~mh/*

Pine: *http://www.washington.edu/pine/*

Procmail: *http://www.procmail.org/*

Learning to Filter Spam E-Mail: A Comparison of a Naive Bayesian and a Memory-Based Approach: *http://citeseer.nj.nec.com/androutsopoulos00learning.html*

POPFile HOWTO forum on SourceForge: *http://sourceforge.net/forum/forum.php?forum_id=234504*

SpamProbe: *http://spamprobe.sourceforge.net*

Bogofilter: *http://bogofilter.sourceforge.net/*

Mozilla page regarding anti-spam features in 1.3a: *http://www.mozilla.org/mailnews/spam.html*

SpamAssassin: *http://spamassassin.org/*

CRM114: *http://crm114.sourceforge.net*

Better Bayesian Filtering: *http://www.paulgraham.com/better.html*

private, HMO, and group support

Which Model Works for You?

by Steven M. Tylock

Steve Tylock has been managing infrastructures for the past 15 years in the Western New York area, and helped organize GVSAGE as a local SAGE for the Genesee Valley region.

Stylock@gvsage.com

If the system administration field is like the medical field, are your users stuck with a group HMO plan when they want a personal physician? This article continues an earlier comparison of the fields of computer and network care and that of medical care by looking at the insurance vs. support model. Consider the similarities to computer care in this review of the state of health care over the past couple of decades:

- In the old days, a premium was paid for an insurance policy that basically said, When you use medical services, present this information and the insurance company will pay for those services on your behalf.
- As the cost and quantity of medical care grew, the health insurance industry decided to tack on an extra service fee (a "co-pay"), in order to have those who use the services bear a larger portion of the cost.
- The HMO, or Health Maintenance Organization, came into existence with the feature that in order to cut costs all around, services would be restricted more than in the past, and only certain service providers could be used to obtain those services.
- Group care went one step further: Instead of maintaining a one-to-one relationship between doctor and patient, a "group" of doctors would be substituted, easing scheduling concerns and maintaining profitability by increasing throughput (theoretically).

A significant detraction of the increasing restrictions on the end user (the patient) was the loss of personal contact with a highly skilled professional. And so, not only were patients paying more for the service (increased premiums, co-pays, deductibles), they were left with a feeling that they didn't really matter anymore.

Centralized vs. Decentralized Care

IT organizations have gone through periods of tremendous growth where support methodologies have been tried, modified, and discarded in favor of "new" approaches.

- The dedicated system administration function has provided the most personal and detailed attention to a functional group's computers and networks. The highest level of costs accompanies this methodology.
- When management wants to equalize the cost of the above solution, they adopt a cost-allocation strategy that may or may not appropriately penalize/reward the desired behaviors. Service will not necessarily change, but it will get talked about a lot more.
- Approved lists of acceptable/unacceptable support expenses provide the next step to control costs. Requests within the bounds of the cost-savings measures are honored, while requests outside of those bounds subject the requestor to an endless stream of approvals (most often requiring the requestor to pay the full freight for the requested activities).
- The last step in the internal process, centralizing services, cuts any remaining personal contact between the requesting organization and the service organization. While the official count of support costs will go down, the indirect costs of self-administration and end-user "workarounds" is usually not accounted for.

The end user may be receiving the products and services but is *not* the customer.

- "Outsourcing" remains the one available option for senior management. This measure ensures that end users realize that they didn't know how good they had it when they had their own sysadmin. In order to get what they want done, they may task one technical resource within their department as the local sysadmin, at least part time.

Yes, this author acknowledges that the above cycle is not cast in stone and that certainly at least one company has made centralized or outsourced system administration work. Unfortunately, it appears that, more times than not, the system breaks down and fuels the growth of an underground system administration function that hides personnel with not-so-accurate titles.

Why?

It makes sense if you examine the money trail; who is the customer? When you or I visit the doctor, who is the customer? We are, or so we think. If you have signed up for an employer-sponsored health plan, often the employer pays something like 75% of the cost of that plan. But no matter what share of the cost is assumed by the employer, guess who is negotiating that plan with the health insurance company – the employer.

And so it follows: Which of time, quality, or cost is the employer most interested in optimizing? Certainly time and quality are not left out of the equation, but they are not the driving factor that cost is. On the positive side, many employers know that some percentage of their workforce considers factors other than cost and is willing to pay more for a plan that enhances health care "time and quality." These companies offer additional selections (for additional cost).

The cost of system support runs the same way. You might think decentralized system administration provides good end-user support, but even then, the functional department or unit, not the users, is picking up the support costs. The first attempt to saddle "users" with the cost of their own support does not reduce the need for that support, but makes the cost very visible. Restrictions on support work as well as restrictions on health care – the large majority of situations can be taken care of by a small part of the process, but the unique situations cannot be ignored (even if the business unit decides ahead of time that it is acceptable to have a two-day recovery period for all end-user system crashes). In order to get a network specialist to look at your problem, you must first get a referral from the group of primary sysadmins.

Complete centralization of system administration work is successful at wringing out the last dollar of savings on direct administration costs because the "company" is paying the bill for the support. The company is telling the support organization what the acceptable level of service is – that which can be achieved with *x* dollars of support. The end user may be receiving the products and services but *is not* the customer.

Where To?

I'm a firm believer in moderation in all things. The goal of infrastructure support is to provide a high level of service to a dynamic population in the most cost-effective way. If end-user needs are paramount, then the system needs to absorb extra costs to manage those needs. If cost constraints are paramount, end-user expectations must be managed to absorb the decrease in service. The middle ground provides an opportunity to manage the environment to the peak of service and cost.

Having spent several years as the system administrator slave, I can attest that the "one dedicated system administrator to N systems" approach fails. This decentralized sysadmin is given the most local control of systems, but is placed in the most critical role; when the sysadmin is absent for any reason, work will wait until he or she returns. While management is unhappy at the exposure to these events, it is quite happy to be able to direct the activities of the sysadmin to local projects and requests.

Having spent several years subject to the "help desk," I can say that the approach known as "global services" also fails. The first person to answer the phone at the central help desk is usually trained in answering the phone and filling in the help ticket. Sysadmins lose the local connection and history of the environments, while end users lose their sanity and find ways around the system.

Why can't we centralize to a properly sized team environment? Consider a team composed of seven to nine members with a mix of senior, journeyman, novice, system, network, and PC administrators. This team could be tasked with optimizing the management of a set geographical or logistical environment. While a trouble ticket system is required, the distribution of those tickets can be managed at the team level.

It is more cost-effective to centralize than not to, but over-centralization brings about monetary gains through loss of quality, flexibility, and speed.

Proposed Model

The *Systems Squad Model* of system administration optimization is based on:

- Continuity of superior service
- Economies of scale
- Appropriate cost sharing
- Team-based sharing of skills, talents, and training
- Team empowerment and flow

Continuity of Superior Service

The goal is to provide superior service with the "right level" of customer contact, skills, history, and sharing of responsibilities.

Economies of Scale

It is more cost-effective to centralize than not to, but over-centralization brings about monetary gains through loss of quality, flexibility, and speed.

Appropriate Cost Sharing

Finance has often tried to justify a system of cost allocation merely because it is the system currently in place. The Systems Squad Model for cost allocation would:

- Place costs at the lowest level at which they can be divided – individual, project, department, organization, or the company as a whole.
- Individual costs provide some idea of the cost structure at that level.
- Group costs justify "bigger ticket" items.
- Provide a fixed cost each month, with variable costs based on events and requests.
- Provide an element of cost control through approvals.
- Include flexibility for special situations.

Team-Based Sharing of Skills, Talents, and Training

The Systems Squad Model provides an environment where all experience levels both train and teach. More experienced members have explicit obligations to the junior members, but they will also learn through teaching and through working with the more experienced members of peer teams.

Team Empowerment and Flow
With a nod to politics (corporate, educational, governmental, or otherwise), the squad reports to the highest organization that it supports. All of the squads report to a central organization, providing consistency between groups and setting support rates through a corporate liaison.

Team members flow in and out, with hiring and transfer decisions based heavily on team decision-making. Though teams maintain integrity, members flow between teams throughout the company, building the entire organization.

Barriers
This model faces objections from many fronts. Some of the most likely come from:

- The decentralized environment: Sysadmins in this environment like the local control over systems, would love to have help in a shared fashion, and hate the concept of centralization. Managers of local sysadmins like the control (over the sysadmin), are tempted by the potential for better service through consolidation, and are scared by the thought of centralization.
- The centralized environment: Sysadmins and managers in this environment are probably not so keen about it, but accept it as the way things are. The political structure above the sysadmins, however, would likely be threatened by it, and the financial controllers might be worried about a system with less visible controls.
- They've all become comfortable with the way things are, and change is something to be feared (or so the common belief is).

Conclusion
The Systems Squad Model looks to optimize value, beyond just an optimization of the money spent on infrastructure. It hopes to optimize the cost, quality, and speed of the organization to deliver more to the end user, project, department, organization, and company than any centralized or decentralized model has done in the past.

And I think that's what we want in health care, too.

trends in the outsourcing industry

by John Nicholson

John Nicholson is an attorney in the Technology Group of the firm of Shaw Pittman in Washington, D.C. He focuses on technology outsourcing, application development and system implementation, and other technology issues.

John.Nicholson@ShawPittman.com

In my last column,[1] I talked about some of the common reasons why outsourcing deals go badly and ways to avoid those problems. The purpose of this column is to discuss some of the current, macro-level trends in outsourcing with regard to (1) the vendors providing the services, (2) the services themselves, and (3) the customers receiving the services.

The Vendors

Trend 1 – Vendors Becoming Public Companies

The market for global outsourcing services is a somewhat unsettled area. Based on size and resources, the "Tier 1" providers are IBM, CSC, EDS and, according to some, ACS. Behind them are the "Tier 2" providers: Perot Systems, CGI, Accenture, Unisys, Lockheed Martin, and Siemens. In addition, hardware manufacturers like Hewlett-Packard and Dell are trying to move into the more lucrative IT services business.2 Also, some new, offshore players like Tata Consulting Services ("TCS") are trying to expand into the US market.

Right now, IBM is the dominant player in the global outsourcing market. This is not because IBM has distinguished itself in terms of capability, service quality, or price (no global services vendor has). It is simply a reflection of some challenging leadership and financial circumstances faced by the other Tier 1 players[3] (a point happily emphasized by the IBM salespeople) and the old hardware adage that "No one ever got fired for buying IBM." IBM recognizes and exploits these advantages whenever possible.

What used to be the consulting arms of the "Big Five" have all morphed in some way and, with the exception of Braxton, are all now publicly held companies:

Yesterday	Today
Andersen Consulting (created as part of Arthur Andersen, then "separated" into an arm of Andersen Worldwide)	Accenture
PricewaterhouseCoopers	IBM
KPMG	Bearing Point
Deloitte Consulting	Braxton
Ernst & Young	Cap Gemini Ernst & Young

The public nature of these companies has two important consequences. First, public companies are subject to the vagaries of the market. Hardware and software companies are already driven by quarterly and annual results, and these now-public service providers will be subject to the same forces. Securities industry analysts are beginning to have concerns about continued growth in the IT services and outsourcing industry because of reduced backlogs, longer lead times to conclude transactions, and a general downturn in the industry. This could be due to general issues with the economy, maturation in the industry or some customer dissatisfaction with the outsourcing model, performance, price, or something else entirely, but, regardless of the cause, it affects the stability of public outsourcing vendors. Second, these companies use their stock as compensation, and as the value of their stock decreases, their ability to attract and retain talent decreases as well.

You're shipping your company's crown jewels halfway around the world, where they are subject to all kinds of possible threats.

The Services

There are two major types of outsourcing deals being structured in the market today: IT outsourcings ("ITO") and business process outsourcings ("BPO"). The market for ITO deals is much more mature, and the deals are fairly well understood.

Trend 2 – Offshore Outsourcing

The hot area in the US for ITOs is offshore outsourcing, either on its own or as part of a larger transaction.[4] Offshore outsourcing is attractive because of its price (currently a factor of 5 to 10 times less expensive than using domestic providers) and the quality of the services provided.(Most Indian vendors have been certified by the Software Enterprise Institute as achieving "Level 5" of the Capability Maturity Model.)[5]

Currently, the main location for offshore ITOs is India, but the demand for technology workers in India is beginning to outstrip the supply, which is driving up the cost of services in India. While services in India are still substantially less expensive than those available in the US or Europe, services provided in China, Eastern Europe, Malaysia, the Philippines, and Russia are becoming more attractive.[6] The major players in offshore outsourcing are divided between the larger firms listed in the previous section, each of which either has its own offshore operations or a close tie to an offshore provider, and the native, "pure-play" offshore companies such as TCS, HCL Technologies, Infosys, Wipro, and Cognizant.

In addition to increasing competition from other countries, the challenges for offshore providers are:

1. The ability to scale their operations while maintaining quality – The fundamental premise of outsourcing is that an outsourcing provider whose core business is providing technology services can provide those services to multiple customers at a lower cost. As more companies move IT operations offshore, the Indian vendors may not be able to maintain the level of quality for which they have become known.
2. Business continuity – The distance between the US and India, combined with the level of infrastructure in India, also raises some questions regarding disaster recovery and business continuity. On one hand, having facilities on the other side of the globe provides some redundancy, but, the very real issue is that your outsourced operations in India are being performed in a country with nuclear weapons that is on unfriendly terms with its next-door neighbor, which also has nuclear weapons. For offshore ITOs that go through one of the larger global providers that has operations or a close alliance with an offshore provider, this can be mitigated by the global provider.
3. Customer concerns related to confidentiality and non-disclosure – You're shipping your company's crown jewels halfway around the world, where they are subject to all kinds of possible threats. What if something gets disclosed? What if an employee takes a copy of your data to a competitor? How will you be able to protect yourself? What remedies will you have against the outsourcing vendor? These are probably the areas that concern potential offshore outsourcing customers the most. In the case of India, the local outsourcing vendors understand the concerns of US companies and are generally willing to agree to contractual terms that are satisfactory to the customer – including robust audit and inspection rights. (Note, however, that you must actually *exercise* those rights by performing the audits and inspections.)

The Indian companies understand the economic value of providing outsourcing services, and they recognize the risk to their economy if foreign firms do not feel safe with regard to confidential information. This is also an issue that can be addressed by contracting with one of the larger global providers. However, this may become more of a real issue as outsourcing moves to countries like China, where there has been a history of intellectual property concerns.

TREND 3 – BUSINESS PROCESS OUTSOURCING ("BPO")

While the ITO market, including the offshore services market, is relatively mature, with large, multi-disciplinary deals a relatively regular occurrence, the BPO market is far from mature (contrary to what most vendors will try to tell you). The earliest form of limited BPO has been going on for decades – payroll processing. Other limited-scope BPOs include HR, finance/accounting services, call centers, and claims processing, but until a large, multi-disciplinary, "shared services" outsourcing agreement is successfully completed and implemented, the BPO market cannot be considered mature. This isn't likely to happen until one of the Tier 1 providers purchases some large company's shared services functionality and resells those services to other customers.

Some people believe that because a BPO provides a "higher" level of functionality (i.e., an ITO provides the services that enable business processes to be performed, a BPO provides the services that touch the end user/customer), it should be easier to identify, structure, and complete a BPO transaction. The opposite is true. Because fewer models for BPOs exist, and because vendors have less experience in structuring BPOs, it is important for both customers and vendors to focus on the correct statement of work with appropriate service levels and pricing. Although the functions included in a BPO are supported by IT services, they are significantly more complicated and involve more unique activities performed by personnel. Because of this, BPO services may be less fungible among the customers of a given vendor, making the outsourcing economic model less viable.

Because of this uncertainty and immaturity in the BPO marketplace, it is important for BPO customers to spend the necessary time and energy to get these deals structured correctly. One of the most important things for an outsourcing customer to do is to understand *in detail* exactly how the vendor proposes to provide the services. Vendors will resist providing this information. They know that once they are in the door and performing the services, you are unlikely to kick them out, due to the pain of bringing in a new vendor. However, acquiring a detailed understanding and documentation of exactly what the vendor will do and what the vendor will charge for it is the only way for a customer to evaluate whether the savings promised by the vendor are actually achievable and whether the pricing structure makes sense.

The ability of a customer to renegotiate is strengthened by strong, clear, termination-for-convenience provisions in your agreements.

The Customers

TREND 4 – RENEGOTIATION OF EXISTING AGREEMENTS

Another issue that outsourcing vendors are facing is that their customers have also been hit by the economic downturn. These customers, who initially signed outsourcing deals on the promise of significant savings, are now demanding further concessions and renegotiating deals to reflect lower volumes. Frequently, the pricing structures designed when the agreements were originally signed are not appropriate to address the new situation, and in the new negotiations customers are even more focused on price.

The ability of a customer to renegotiate is strengthened by strong, clear, termination-for-convenience provisions in your agreements. When negotiating with vendors at the beginning of a transaction, make sure that you have the ability to terminate for convenience. Frequently, vendors will agree to allow you to terminate a contract early provided that you give them notice and pay a termination-for-convenience fee.[7] If you terminate the contract early, it isn't unreasonable for a vendor to recover costs that the vendor expected to recover during the term. The vendor shouldn't suffer harm because you elected to terminate a contract early. However, once you terminate the contract, the vendor won't be providing services or incurring costs, so there is no reason for the vendor to receive the total revenues that the vendor would have received for providing the services during the remainder of the term.

Nor is there much ground for the vendor to argue that it should receive the profits that it would have received during the remainder of the term. Vendors will try to use the argument that they have a right to lost profits because they shouldn't suffer harm because you decided to terminate the contract early. If the contract had never happened, the vendor would not have received the profits that they are trying to claim, so allowing the vendor to include lost profits in a termination-for-convenience charge puts the vendor in a better position than if the deal had never happened. The vendor should be allowed to recover costs, but not lost profits.

By understanding the economic forces driving outsourcing vendors, you can renegotiate with them for a better deal. Just as you can negotiate with hardware and software vendors who need to make their end-of-quarter or end-of-year numbers, you may be able to negotiate with outsourcing vendors in the same way.

Trend 5 – Piecemeal Deals

Given the fact that negotiating outsourcing deals is a lengthy process, companies are trying to expedite their deals by breaking up the scope of services and negotiating multiple deals rather than one large deal. This is the wrong approach for several reasons: first, the larger the scope of the deal, the more negotiating leverage you have with the vendor; second, negotiating outsourcing deals is a very resource-intensive process for the customer, and doing more, smaller transactions is much less efficient; third, outsourcing can lead to morale problems until the deal is finished, and doing multiple deals just extends the uncertainty for the customer's workforce and creates the potential for extended morale issues; fourth, vendors will use the multiple negotiations as opportunities to re-negotiate provisions that came out in the customer's favor in earlier deals. For BPOs, in particular, the preceding sentence is still true, but the customer needs to recognize that BPO services are less mature, and, therefore, the customer will need to pay much more attention to the scope, service levels, and pricing of the transaction.

Piecemeal deals significantly increase the cost and risk to the customer.

Conclusion

Economics and the maturing nature of the industry are driving many behaviors in the outsourcing industry, and by understanding them you can help your company deal with vendors. Because vendors and customers will increasingly be focused on the short-term bottom line, it is increasingly important that each party spend the time and energy to first get a deal right and then keep it on track. The relationships between the vendor account team and the customer's program management will determine whether outsourcing relationships succeed.

NOTES

1. "Common Problems with Outsourcing Deals and How to Avoid Them," *;login:*, vol. 28, no. 1, February 2003, pp. 6–10.

2. See Kennedy, Siobhan, "HP Eyes IBM with Multibillion-Dollar Deals," Reuters, April 11, 2003, as posted at *http://biz.yahoo.com/rb/030411/tech_hewlettpackard_5.html*.

3. See, for example, Bonasia, J., "EDS Investors Watch New CEO," *Investor's Business Daily*, April 14, 2003, as posted at *http://biz.yahoo.com/ibd/030414/tech01_1.html*; and "Accounting Fears Hit Computer Sciences," Reuters, April 2, 2003, as posted at *http://biz.yahoo.com/rb/030402/tech_computersciences_stocks_1.html*.

4. The potential for offshore outsourcing is limited in Europe by EU data privacy regulations. Since these rules regulate the transfer of data outside the EU, using an offshore vendor may not be possible for many European companies.

5. For more information, see *http://www.sei.cmu.edu/cmm/cmm.html*.

6. See Chai, Winston, "Outsourcing haven India running out of techies," *http://www.silicon.com*, February 18, 2003. See, also, "Gaining Ground," *Information Week*, March 31, 2003, *http://www.informationweek.com/story/IWK20030328S0003*.

7. Note that there is a difference between "termination for convenience" and "termination for cause." Termination for cause is when the vendor has breached the contract in some way and you are firing the vendor. Termination for convenience means that you have simply decided that you don't want the services anymore.

put your data where your mouth is

Public-Private Partnership Begins with Reporting Cybercrimes

by Erin Kenneally

Erin Kenneally is a Forensic Analyst with the Pacific Institute for Computer Security (PICS), San Diego Supercomputer Center. She is a licensed attorney who holds Juris Doctorate and Master of Forensic Sciences degrees.

erin@sdsc.edu

Society has developed a love-hate relationship with the concept of "sharing" security data. On the one hand, sharing has been a rallying cry for combating cyberterrorism, but it is also bemoaned by corporate financiers to justify a protectionist mentality that sees "sharing" as Big Brother wrapped in sheep's clothing. This article illustrates the ideas of "sharing security data" and "public-private partnership" in hopes of motivating others to move beyond the current holding pattern of cyber-infrastructure needs assessment and into a strategy for securely grounding our structures in response to the dangers of cyberspace.

The San Diego Chapter of the High Technology Crime Investigation Association (HTCIA), a grassroots organization founded to share information relating to investigations and security, has developed "Working with Law Enforcement to Abate Cybercrime." These guidelines are a proactive attempt by law enforcement to communicate elements of policies, incident response plans, and evidence handling procedures that are vital to the effective identification, prosecution, and prevention of cybercrime, as well as infrastructure protection.

The HTCIA Guidelines are in print and available from the author at *erin@sdsc.edu*. They can also be downloaded from *http://www.catchteam.org* or *http://security.sdsc.edu*

Motivation for Sharing Data Between the Public and Private Sector

The HTCIA Guidelines resulted from a desire to enable the private sector cybercrime victims to communicate clearly with law enforcement. All too often, law enforcement is called onto a cybercrime scene – whether it be a hacker intruding on a company's network, denial of service attack, theft of intellectual property, discovery of child pornography, or insider abuse of privileges – only to find inadequate or nonexistent policies and procedures for handling cybercrime incidents, which prevents investigators from tracking and punishing the digital perpetrator. Corporate victims often wanted to know how to ready the cybercrime scene for optimal law enforcement.

National Strategy to Secure Cyberspace

The recently released "National Strategy to Secure Cyberspace" has been the most popularly recognized call for a public-private partnership and sharing of threat data. However, it has been criticized as not doing enough to hold responsible parties' "feet to the fire" in attempting to secure cyberspace. The Strategy is a framework of suggestions rather than a regulation, so it has few real teeth.

Computer Crime Laws

The need to share data is underscored further by the fact that there are laws in existence that criminalize both unauthorized access to computer systems and exceeding access privileges. There is a misperception that just because most laws targeting the security of computer data congregate around specific business practices within select industries (i.e., HIPPA for the health-care industry and GLBA for banking and finance institutions), there is a dearth of criminal remedies that the private sector can utilize to address cybercrime. On the contrary, the federal Computer Fraud and Abuse Act, as well as its state counterparts, can be a remedy for industry, with one caveat: In order to

The possibility of bad publicity has been used to justify sweeping cybercrime incidents under the rug.

invoke it, victims of cybercrime must first report the incident to law enforcement. We have been conditioned not to think twice about hailing the cops when someone breaks into our safes or assaults one of our employees at the work site, yet the same knee-jerk reaction to an insider breach of access control or external trespassing on our networks does not occur frequently enough.

Countering Myths About Sharing Cybercrime Data

Corporate Reputation

The possibility of bad publicity has been used to justify sweeping cybercrime incidents under the rug. Companies can, however, communicate concerns about confidentiality and the desire to minimize publicizing the incident. This will not ensure that this information will never become public if prosecution occurs, but it can lengthen the amount of time until this information is subject to public accessibility and give your organization time to craft a strategy that minimizes potential negative publicity. Additionally, the government recently announced FOIA exemptions for companies wishing to share related cybercrime data.

If public embarrassment and the assumed detrimental effects on business profitability are concerns, consider the bad public relations ramifications of failing to report incidents in the current culture of increasing litigation and the call for disclosure spurred by Enron and security broker scandals. Often, the hint of impropriety or cover-up is enough to send stock plummeting or scare away potential clients and partners.

Some states may choose to follow California's lead in enacting laws that require the disclosure of security breaches that expose customers' personal data. So, whether it is through market or regulatory mechanisms, the costs of not reporting may ultimately be more ominous than what appears on the surface. Furthermore, publicity can sometimes work to your advantage, as companies can distinguish themselves as taking a leadership role by reporting and setting a standard for others to follow.

Notification of cybercrimes to authorities does not obligate victims to participate in the investigation. However, lack of participation may affect successful legal remedies.

Cost of Not Reporting Cybercrime

Another excuse is the notion that reporting cybercrime and pursuing prosecution is cost prohibitive. In other words, it is cheaper to eat the loss and not disrupt the business process. It is prudent to assess the cost of not reporting, too. The mechanisms advocated here should be consistent with the mechanisms that should already be part of your organization's strategy to deal with cybercrime in-house. The Guidelines address the relevant security risks and obligations that you must know to effectively meet your responsibility to your investors, partners, clients, and customers.

Additional costs for evidence-handling and the like should be minimal, since most procedures are similar to what companies should have implemented already. The costs associated with the potential liability (negligence, shareholder suits, regulatory noncompliance) make sharing less ominous. Furthermore, the costs of complying with potential future regulation created by the corporation's lack of reporting will likely equal or exceed the costs of implementing the current recommendations.

High-Tech Crime Fighters

Another misperception used to justify failure to report is that law enforcement is technically ill-equipped to effectively resolve cybercrime. State and federal agencies have teamed up all across the country to establish high-technology task forces whose sole mission is to investigate and prosecute cybercrime. California is a leading example, where the state has six regional teams made up of investigators from local, state, and federal agencies that are specifically trained to handle everything from kiddie porn traders to identity thieves to high-level hackers and corporate espionage. Unless victims call upon their services, funding to support and expand this cadre of skilled investigators will dry up. With a more concerted effort to report cybercrime, we increase the likelihood of laws that place fewer restrictions on cybercrime investigations under the rubric of national security. So, sharing cybercrime incidents may actually facilitate the protection of civil liberties and privacy.

Without reporting, we cannot quantify the incidence of cybercrime.

Cyberinsurance and Risk Management

Actuarial data is another motivation for sharing cybercrime incidents. Without reporting, we cannot quantify the incidence of cybercrime. Obtaining actuarial data related to the incidence of cybercrime is relevant to your organization if you are at all concerned with risk management. From a systemic level, the more accurate the data on cybercrime, the more accurate the assessment of risk. Cyberinsurance will almost certainly become as ubiquitous as automobile insurance and is another tool for management of information security.

Damages and risk factors are difficult to measure and are often exaggerated [CSIS 2000 – Center for Strategic and International Studies, *Cyber Threats and Information Security: Meeting the 21st Century Challenge*]. Actuarial data grounded in reported incidents of cybercrime will enhance the accuracy of probability-of-loss estimates and premium pricing.

HTCIA Public-Private Guidelines

Below is an abbreviated outline of the HTCIA Guidelines:

I. INFORMATION SECURITY POLICIES

A. DEFINITION PHASE

1. Roles and Responsibilities, Personnel Who Deal with Law Enforcement (LE)

    a. Establish two points of contact (POC) – one security/technical and one legal/senior administrative.
    b. Educate and integrate designated POCs into LE and prosecutorial agencies and high-technology associations to establish trust relationships (i.e., HTCIA, Infragard).
    c. Designate chain-of-command regarding authority to control investigation and report to LE.
    d. Define who a "user" is for application of the Acceptable Use Policies.

2. Privacy Expectations

    a. Define scope and coverage in order to inform users of the when, where, why, and what regarding expectations of privacy, enabling companies to deal with violations properly.
    b. Establish organization ownership of computer facilities (hardware, software, data, communication devices).
    c. Establish that use of computer facilities should be work-related and the scope of use should be duty-related.

3. Define Acceptable Use Policies

4. Define What Constitutes an "Incident" for Application of Policies

5. Define and Document Incident Response Plan

    a. Examples of reportable computer crimes
        i. Actual intrusion into system circumventing access controls
        ii. Exploiting vulnerable programs
        iii. Denial-of-service attacks
        iv. Theft of bandwidth
        v. Exceeding authorized access
        vi. Child pornography storage or transmission
        vii. Theft of intellectual property or trade secrets
    b. Examples of incidents better resolved internally or under civil law rather than by LE

6. Define Consequences of Non-Compliance (reserving the right to institute penalties, including criminal prosecution)

B. Documentation Phase

1. Document Policies

    a. Obtain acknowledgment via click-thru forms on the Web that evidence acknowledgment of policies.
    b. Obtain signature evidencing user understanding and pledge to abide.
    c. Include policy language stating that the company reserves the right to change the policy, and the user is obligated to regularly visit a clearly demarcated location where policies are accessible.

2. Document Audit Policy and Audit Logging

3. Document Incident Cost Model

    a. Document policies and procedures for collecting measurable loss data in response to computer security incidents
        i. Replacement of hardware, software, or other property that was damaged or stolen.
        ii. Lost productivity by users who were unable to use systems during relevant time period.
        iii. Time spent by all staff to clean up the damage to systems under your control (e.g., analyzing what has occurred, re-installing the operating system, restoring installed programs and data files, etc.).
        iv. Who worked on responding to or investigating the incident.
        v. Indirect costs: any revenue lost, cost incurred, or other consequential damages incurred because of interruption of service (must have methodology and reasonable justification for calculation).

C. Dissemination Phase

II. IMPLEMENTING AND ENFORCING POLICIES

A. Incident Response Checklist

1. Pre-Incident Planning

2. During Suspected Incident

3. Post-Incident: What Law Enforcement Needs to Investigate (reactive mode)

    a. Preserve all relevant logs on all systems (i.e., Web logs; Intrusion Detection System (IDS); firewall; mail logs).
    b. Obtain name list of all users, new hires, and terminated users within past six months.
    c. Identify all network access points (trusts granted to other networks): Internet gateways, VPNs, LAN/WAN connections.
    d. If dealing with an intrusion from outside the company, and if you have trained in-house security response capabilities, exhaust all methods of intraoffice security investigations in tracing back prior to contacting law enforcement.
    e. Collect copies of complaints sent to organization during timeframe of incident.
    f. Calculate time offset (including time zone) of all affected computers.
    g. Collect copies of badge/entry logs, security cameras, etc. for internal incident.
    h. Identify all correspondence with external organizations/individuals, especially foreign to the United States.
    i. Preserve forensic image(s) or actual drives from compromised system (law enforcement can supply media and manpower for image). (See D, below, Evidence Recovery and Handling).

4. Notify Law Enforcement

    a. Notification should generally be made immediate on the discovery of a suspected violation. However, you may choose to have skilled and trained staff conduct a forensically sound internal investigation prior to calling in LE. The advantage here is that you may be able to obtain certain evidence more efficiently, yet still within the bounds of the law, than when LE is involved.

NOTE: Notification does not obligate you to participate in the investigation. However, lack of participation may affect successful legal remedies.

5. Questions that LE Will Likely Ask When You Make the Complaint

    a. What evidence do you have that you were victimized?
    b. What is the chronology of the event?
    c. What is the impact to your network?
    d. Are your systems still running?
    e. When did the incident first occur?
    f. When was the incident discovered?
    g. Who discovered the incident?
    h. Is the activity ongoing?
    i. Who do you think is responsible for the incident and why do you suspect them?

j. What is the internal or external IP address for the attacker?
k. Can you provide a complete topology of your network?
l. Who in the organization has been notified?
m. Who outside the organization has been notified?
n. From this point forward, who does law enforcement contact and who can they speak to if they are contacted?
o. What are your estimated damages?

B. Evidence Recovery and Handling Guide

Once the nature of the incident has been defined, the next step is to identify where data relevant to the incident may reside. The steps taken to identify the incident, which may have a forensic impact, need to be completely documented.

Additionally, the decision whether to secure and maintain evidence should be factored into your organization's risk analysis. This is a cost-benefit analysis that should consider the short- and long-term economic consequences of keeping log data, including the effect on potential civil and criminal legal actions. Even if no legal action is taken, organizations may want to consider maintaining logs for a limited amount of time in accordance with document retention policies.

1. Essential Elements to Evidence Recovery and Handling
    a. Identification of relevant data
    b. Isolation of evidentiary data
        i. Considerations – once data is identified, relevant systems should be secured to avoid possible contamination of digital evidence.
        ii. Partial or complete system analysis (copy of relevant logs, logical image of media, physical image of media).
        iii. Real-time backup of data to remote location in accordance with data retention model and policies.
    c. Preservation of evidentiary data
    d. Resources that should be considered

Proper evidence recovery and handling requires specialized forensic tools and training. The resources available may be internal (IT, Security), external (private consultants), and law enforcement.

Conclusion

Our increasing dependence on technology has enhanced business functionality and productivity while simultaneously exposing our organizations to more frequent and severe threats. Securing organizations demands better cooperation with law enforcement, which provides critical and unique information services beyond the capabilities of any one organization.

The HTCIA Guidelines are intended to help victims of cybercrime more effectively interact with law enforcement so that the goals of both entities are better served and to advance public-private cooperation by identifying essential elements of an organization's policies, incident response plans, and electronic evidence-handling procedures that will meet the goals of both your organization and law enforcement. Without more victims working with law enforcement to track down cybercriminals, we cannot expect to abate the frequency and severity of cyber threats.

the bookworm

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@netpedant.com

Readers of this column probably know that I'm a devotee of history, which is great since the main items this month are two history books. Each has some shortcomings, but they are both excellent and help us understand just how we got here, with respect to both hardware and software.

Software

Martin Campbell-Kelly has been writing interesting historical work for quite a while. I recall his book on ICL occupied me while I was at a UKUUG about 15 years ago. His collaboration with Aspray, *Computer*, is an excellent survey. This new book on the software industry is really splendid. I can only fault it for what's not there.

Describing the history and development of software can make one feel like one of the blind men examining the elephant. Campbell-Kelly has chosen to look at the industry from the 1950s to 1995. What he chooses to represent in those 40 years is quite selective, too: FORTRAN and COBOL, but no ALGOL; C, but no C++; SHARE, but not DECUS or ADUS; SCO but not mtXinu. There is one line on Linux; nothing on any of the BSDs. Burroughs is there, but not Ferranti. The general "feel" is Anglocentric: SAP but no Chorus.

But that's the sort of thing I complain about. What I'd really like to see is a genuine history of software: one that includes languages and tools (like grep or make or sort or . . .); one that includes GNU and BSD and Linux; Gnome; Opera; even the Open Software Foundation's Motif. News – Usenet – isn't "commercial," I guess. But Usenet and UUNET were really important, as were rn, ANews, BNews, and CNews.

This is a book about the software industry – economics and business. It's not a history of software. Campbell-Kelly limns a portion of the story and does it very, very well. I wish someone would try the whole thing.

Hardware

The first edition of Ceruzzi appeared five years ago. The new, second edition is emended and enlarged. The enlargement consists of an additional chapter covering the period 1995–2001. Even though Ceruzzi's emphasis is on hardware, he does a fine job where the Microsoft antitrust suit and Linux are concerned. There is a very clear (though brief) discussion of open source software (though Stallman and GNU are never mentioned).

One gets the impression that Ceruzzi, in the National Air and Space Museum in Washington, DC, is more urbane than Campbell-Kelly at the University of Warwick. Ceruzzi mentions the Ferranti Atlas and the Cromemco. He recognizes Zuse and work at the ETH in Zurich. He recognizes the importance of Wilkes and of Dijkstra (neither of whom appears in Campbell-Kelly).

There are things one might want described differently, but on the whole Ceruzzi's book is one anyone interested in computing can read with profit.

Buy them both. They're really worthwhile.

TCP/IP, Once More

Mansfield has produced a very handy, very useful book which is applicable to "Linux and Windows." I know nothing

BOOKS REVIEWED IN THIS COLUMN

FROM AIRLINE RESERVATIONS TO SONIC THE HEDGEHOG: A HISTORY OF THE SOFTWARE INDUSTRY
MARTIN CAMPBELL-KELLY
Cambridge, MA: MIT Press, 2003. Pp. 372. ISBN 0-262-03303-8.

A HISTORY OF MODERN COMPUTING, 2D ED.
PAUL E. CERUZZI
Cambridge, MA: MIT Press, 2003. Pp. 445. ISBN 0-262-53203-4.

PRACTICAL TCP/IP
NIALL MANSFIELD
London, UK: Addison-Wesley Professional, 2003. Pp. 851. ISBN 0-201-75078-3.

PRACTICAL UNIX AND INTERNET SECURITY, 3D ED.
SIMSON GARFINKEL ET AL.
Sebastopol, CA: O'Reilly, 2003. Pp. 954. ISBN 0-596-00323-4.

UNDERSTANDING AND DEPLOYING LDAP DIRECTORY SERVICES, 2D ED.
TIMOTHY A. HOWES ET AL.
Boston: Addison-Wesley, 2003. Pp. 936. ISBN 0-672-32316-8.

about running TCP/IP on a Windows network, and a provident deity may render it unnecessary for me to acquire such knowledge. But I can reveal that the information concerning Linux is well-presented, and – as far as I can tell – correct. (This last is not as silly as it may seem; I try not to write about the books that just get it wrong.) This is not intended to replace the Comer or the Stevens sets, but I think it's more detailed than Craig Hunt's O'Reilly volume. The inside front covers contain a useful list of TCP and UDP port numbers; the rear covers have a table of decimal, binary, and hex numbers. As I'll be 1000001 by the time you read this, you may need it to keep score.

Third At-Bat

Garfinkel and Spafford hit a homer over a decade ago with *Practical Unix Security*. They hit another five years later with *Practical Unix and Internet Security* (1996). The cast now includes Alan Schwartz, and the third edition, actually marginally shorter than its predecessor, is another base-clearer. The book really covers Solaris, MacOS X, Linux, and FreeBSD, and it should sit on your shelf together with Bishop and Cheswick, Bellovin and Rubin. Fine job!

Another Return

Understanding and Deploying LDAP Directory Services was really useful when it appeared five years ago. The new edition has put on nearly a hundred pages, but it's still the LDAP book to have. These guys know their stuff; just look at RFCs 1558, 1778, 1823, 1959, 1960, 2251, 2254, 2255, 2559, 2587, 2596, 2696, 2798, and 2849.

Twenty Years Ago in USENIX

by Peter H. Salus

Toronto. A beautiful city; over 1200 attendees; an interesting meeting.

Tuesday evening, July 12, 1983

Neil Groundwater convened a meeting of the Software Tools User Group. The first speaker was Brian Kernighan, who delivered a personal view of the development of the tool concept and the tools from 1969 to the formation of STUG.

Wednesday, July 13

Mike Tilson thanked everyone for everything. Lou Katz announced that the next meeting would be in Washington, D.C. (to be dubbed "snowstorm #1") and the subsequent one in Salt Lake City (memorable for Stu Feldman's architecture keynote).

Mike Lesk then delivered the keynote: "Technology-Driven Software vs. Psychology of Users." Among the points made were:

- Less documentation is better.
- Terseness does not mean documentation need be cryptic.
- The UNIX manual used to be small; now manuals issue a master's degree in stty.

Once upon a time (before the 7th edition), you could carry the UNIX docs in your briefcase; today you can do it only because the docs are on CD.

Mike was followed by Larry Iseley of Western Electric (remember them?). It was divulged that UNIX licensing had been assigned to Western Electric; that the Technology Licensing Group had moved *in toto* to North Carolina; and that it was now headed by Otis Wilson. He gave the number of source licenses, but he refused to say how many binary licenses there were. (Note: the PC had appeared; IBM had introduced the PC-AT. The XT would be released in 1984.)

A Few Other Papers

There were a few other papers: Holt, Mendel, & Perlgut of the CSRG talked about TUNIS, which was a UNIX-compatible kernel written in Euclid; Michel Gien talked about the Sol Operating System, which was implemented in Pascal and ended up as Chorus.

Thursday, July 14

Bob Kridle and Kirk McKusick delivered a paper on "Performance Effects of Disk Subsystem Choices for VAX Systems Running 4.2BSD UNIX." John Chambers and John Quarterman spoke on "UNIX System V and 4.1C BSD," followed by Mike O'Dell on "Berkeley UNIX after 4.2BSD."

Rob Pike delivered "UNIX Style, or cat -v Considered Harmful," and Dave Korn introduced "KSH – A Shell Programming Language."

It's really amazing to look back 20 years and reflect on the importance of much of this. But there was yet more.

Friday, July 15

Mike O'Dell chaired a session on UNIX mail. He gave a brief talk, too. But the other participant was Jim McKie, on "Where Is Europe?" – he hadn't moved from Amsterdam to New Jersey, yet.

Laura Breeden and Mike O'Brien talked about the (brand new) CSNET; Joe Yao spoke on "Dynamic Configuration" and Dan Klein gave a paper on "MIRAGE – An Assembler Generator and Relocatable Linker."

A really fine conference.

conference reports

This issue's reports focus on the 2003 Spam Conference, USITS '03, and FAST '03

OUR THANKS TO THE SUMMARIZERS:

FOR SPAM '03:
Chris Devers

FOR USITS '03:
Ajay Gulati
Xuxian Jiang

FOR FAST '03:
Scott Banachowski
Nate Edel
Preethy Vaidyanathan

2003 MIT Spam Conference

CAMBRIDGE, MA

JANUARY 17, 2003

http://spamconference.org/

Summarized by Chris Devers

[Editor's Note: The summaries by Chris Devers were condensed for publication in ;login:.]

SPARSE BINARY POLYNOMIAL HASHING AND THE CRM114 DISCRIMINATOR

William S. Yerazunis, Mitsubishi Electric Research Laboratories

Yerazunis wrote the CRM114 filtering mini-language and then wrote MailFilter in CRM114 as an implementation that can be used with other spam-fighting programs. The basic idea is to decompose a message into a set of "features" composed of various runs of single words, consecutive words, words appearing within a certain distance of one another, etc.

He claimed that with this software he could get better than 99.9% accuracy in nailing spam, and a similar percentage in avoiding "ham" (the term everyone was using for false positives – legit mail that was falsely identified as spam). One of Yerazunis' observations is that the best way to defeat the spam problem is to disrupt the economics: if a 99.9% or better filter rate were to become the norm, then the cost of delivering spam could be pushed higher than the cost of traditional mail and the problem would naturally go away without requiring legislation.

THE SPAMMER'S COMPENDIUM

John Graham-Cumming, POPFile

Most of this very entertaining talk was about the ingenious tricks that spammers resort to to obfuscate spam against filters, including, most diabolically, one example that placed each column of monospace text in the message into an HTML column, so that the average HTML-capable mail client would render the message properly, but it would be absolute gibberish to most mail filters. The ultimate lesson was that any good filter has to focus not on "ASCII space" (the literal bytes as transmitted) but the "eye space" (the rendered text as seen by the user), which, by extension, may mean that any full-scale spam parser/filter could also have to include a full-scale HTML and JavaScript engine.

As for Graham-Cumming's software, it's a Perl application, available for all platforms (Windows, Mac, and, of course, Linux) that enables users to filter POP3 mail. Interesting stuff if you're a POP user: *http://popfile.sourceforge.net*.

SHOPIP

John T. Draper

Most of Draper's work seemed to be focused on profiling spammers, as opposed to profiling spam itself, by throwing out a series of honeypot addresses and using data collected to hunt down spammers. *http://spambayes.sourceforge.net*

SPAM RESEARCH: ESTABLISHING A FOUNDATION AND MOVING FORWARD

Paul Judge, CipherTrust

Judge's big argument, which no one really disagrees with, is that spam has become not just a nuisance but an actual information security issue. To that end, he is advocating much more collaborative effort to address the problem than we have seen to date: conferences like this, mailing list discussions, better tools, and public data repositories of known spam (and ham). To that last point, one of his observations (which others made as well) was that there are no universally agreed-on standards for what qualifies as spam, so repositories for spam will not be accurate for all users (e.g., spam for your programmers will be the bread-and-butter of your marketing department). Plus, there are obvious privacy issues in publishing your spam and ham for public scrutiny. And to add another wrinkle, one danger of public spam/ham databases is that spammers can poison them with false data, screwing things up for everyone. That said, he encouraged users to help out with building *http://spamarchive.org*.

BETTER BAYESIAN FILTERING

Paul Graham, Arc Project

Graham is the man who organized the conference and kicked off everything this week with his landmark paper from last fall, "A Plan for Spam." Graham's spam-filtering technique famously makes use of Bayesian statistics, a technique popular with nearly all of the speakers. The nice thing about a statistical approach, as opposed to heuristics, simple phrase matching, RBLs, etc., is that Bayesian statistics can be very robust and accurate; the down sides are that they have to be trained against a sufficiently large "corpus" of spam (most techniques have this property, though) and they have to be continually retrained over time (again, this is common). Graham was too modest to produce numbers, but subjectively his results seemed to be even better than what Yerazunis gets with MailFilter by an order of magnitude or more.

Like other speakers, he predicted that spammers are going to make their messages appear more and more like "normal" mail, so we're always going to have to be persistent about this; as one example, he showed us an email he received IN ALL CAPS from a non-English-speaker asking for programming help, and although it was legit, the filters insisted otherwise. "That message is the one that keeps me up at night."

Everyone interested in the spam issue should go read Graham's paper immediately.

INTERNET LEVEL SPAM DETECTION AND SPAMASSASSIN 2.50

Matt Sergeant, MessageLabs

SpamAssassin is a well-known Perl application for heuristically profiling messages as spam, adding headers to the message, saying, for example, "I am 72% sure this is spam because it has X Y Z," and passing off the message to procmail, or whatever, to be handled accordingly. SA can handle a message throughput great enough that it can be deployed at the network level (whereas some of the other applications, which might have somewhat better hit rates, are still too inefficient at this point). Deployed this way, the differences in effectiveness for single vs. multiple users becomes very apparent, as 99% effective rates fall down into the 95–80% range. This happens because, again, different users define different things as spam, so mapping one fingerprint to all users can never work quite right.

For an example of a tool that your company can deploy right now and get fast, decent results, SA looks like a good choice; but for the long run it looks like a Bayesian technique is going to get better performance, and SA is adding a statistical component to its toolkit. Good talk.

ANTI-SPAM TECHNIQUES AT PYTHON.ORG

Barry Warsaw, Pythonlabs at Zope Corporation

This was another example of the "monocultures are dangerous" philosophy, as Warsaw explained how he is helping to use a variety of anti-spam techniques – from clever Exim MTA configuration to good use of SpamAssassin and procmail to fine-tuning of the Mailman mailing list engine – to work together to manage the spam problem for all things Python (Python.org, Zope, many mailing lists, a few employees, etc.).

He pointed out that some very simple filters can be surprisingly effective: run a sanity check on the message's date, look for obviously forged headers, make sure the recipients are legit, scan for missing Message-ID headers, etc. In response to the person who originally posted the article, yes, he did mention blocking outgoing SMTP as an effective element of a many-tiered spam management approach.

Among other tricks for getting the different filtering tiers to play nice together, they make heavy use of the X-Warning header so that if an alarm goes off in one tier of their mail architecture, other components can respond appropriately. Cited projects included ElSpy and SpamBayes.

SPAM: THREAT OR MENACE? AN ISP'S VIEW

Barry Shein, The World

His core argument is that spam is "the rise of organized crime on the Internet," that filters are nice but that the mail architecture itself is fundamentally flawed, and that ISPs like his – in 1989, The World was the world's first dialup ISP – are being killed by the problem.

Shein was very annoyed that all these talented people are having to clean up a mess like this when they should be out working on more interesting stuff. His big hope seemed to be that legislation will someday come to the rescue, but he sounded very pessimistic. (Others in the room seemed to feel that this was a very interesting machine-learning problem and weren't really fazed by his pessimism – but, then, most of the people in the room don't run ISPs.)

He also suggested that we need to find a way to make spammers pay for the bandwidth they are consuming (rather than having users and ISPs shoulder the burden) but didn't seem to know how we might go about implementing this. At all.

SMARTLOOK: AN E-MAIL CLASSIFIER ASSISTANT FOR OUTLOOK

Jean-David Ruvini, e-lab Bouygues SA

This was an interesting product. Ruvini's company is developing an extension to

Outlook 2000 and XP that will watch the way users categorize messages into folders, come up with a profile for what kinds of messages end up in which folders, and then try to offer similar categorization on an automatic basis. Think of it as procmail for Outlook, without having to mess with (or even be aware of!) all the nasty recipes.

Obviously, if you have a spam folder, then spam will be one of the categories it looks for, but, more broadly, it will try to categorize all your mail as you would ordinarily categorize it. This makes SmartLook a broader tool than "just" a spam manager.

SmartLook is another statistical filter, though it uses non-Bayesian algorithms to get results. e-labs' tests suggest that the product is able to properly categorize messages about 96% of the time, with no false positives, and (for their tests, mind you) that it performed better than Bayesian filters over three months of usage.

One nice property of this tool was that it works well with different (human) languages – some strategies fall apart and/or need retraining when you switch from English to some other language. For certain markets (e-lab is in France) this is a crucial feature, and having a tool that works with one of the biggest mail clients out there (most people don't use Mutt or Pine, sadly enough) can be very valuable. Very clever – watch for the inevitable embrace and extend three years from now.

Lessons from Bogofilter

Eric Raymond, Open Source Initiative

He didn't say anything about guns, but he did try to correct one of the other speakers for misusing the term "hacker."

Like Graham, ESR is a Lisp fan, but he knows that the vast majority of people aren't, and he also knows that the vast majority of people need to be using something like Graham's spam software. So on a lark, he came up with a clean version in C, named it bogofilter, and put it on SourceForge, where a community sprang up to, well, embrace and extend it.

As good as Graham's Bayesian algorithm is, ESR felt – as did many of the other speakers – that the nature of your spam/ham corpus is much more significant than the relative difference among any handful of reasonably good algorithms. (Back to the often-repeated point about how corpus effectiveness falls apart when used for a group of users, as opposed to individuals.)

To that end, he strongly felt that the best way to deal with the spam problem is to get good tools into the hands of as many people as possible, and to make them as easy to use as possible.

As an example, one of the first things he did was to patch the Mutt mail agent so that it had two delete keys: one for general deletion and one to "get rid of this because it's spam." That second key, and interface touches like it, seem like the way to get average people to start using filters on a regular basis.

Spam Filtering: From the Lab to the Real World

Joshua Goodman, Microsoft Research

Unlike ESR, Goodman felt that algorithm selection does make a big difference, but, this being Microsoft, he refused to disclose what algorithms his team is working with – except to say that, when delivered, they will be more accessible for average users than SpamAssassin, procmail recipes, or Mutt.

Microsoft has been working on the spam problem since 1997, but because of how big they are, they've had unique problems in bringing solutions to market. As a case in point, they tried to introduce spam filters in a 1999 Outlook Express release, but were immediately sued by email greeting card company Blue Mountain because their messages were being inaccurately categorized as spam. With that in mind, they have been very reluctant to bring new anti-spam software out since then, because they would like to see legislation protecting "good faith spam prevention efforts."

As a very large player, Microsoft faced certain difficulties in developing useful filters: It may make sense for you as an individual to filter all mail from Korea, but this doesn't work so well if you are trying to attract customers from Korea. This has forced them to put a lot of work into thoroughly testing different strategies before offering them to the public.

In spite of what millions of Webmail users might have expected, Hotmail and MSN are currently being filtered by Brightmail's service, and plans are underway to re-introduce spam-management features to client-side software again. (Just imagine how bad it would be if they weren't paying someone to filter for them!)

An interesting barrier his group has had to grapple with was what he called the "Chinese menu" or "madlibs" spam generation strategy: that it's easy to come up with a template for spam – "[a very special offer] [to make your penis bigger] [and please your special lady friend all night!" vs. "[an exclusive deal] [for genital enlargement] [that will boost your sex life!]" etc. – and have a small handful of options for each "bucket" multiplying into a huge variety of individual messages that are easy for a human to group together but almost impossible for software to identify.

Integrating Heuristics with n-grams using Bayes and LMMSE

Michael Salib, extremely funny MIT student

Unlike nearly all other filter writers of the day, Salib's approach was heuristic: find a handful of reasonable spam discriminators, throw them all against his mail, and see how much he can identify that way. "It's sketchy, but this is a class project. I don't have to be realistic. These results may be completely wrong."

Much to his surprise, he's trapping a lot of spam. He pulls in a little bit of RBL data ("the first two or three links from Google, whatever"), looks for some patterns, and then churns it through LMMSE, an electrical engineering technique that, as far as he can tell, doesn't seem to be known in other fields. Basically, this involves running the messages through a series of scary-but-fast-to-calculate linear equations. It turns out that he can process this much faster than a Bayesian filter, to the point that customizing his approach for each user in a network would actually be feasible.

For a small spam corpus, he got results better than SpamAssassin did, though for a large corpus his results were worse; he couldn't really account for why this would be the case, or predict how things would scale as the corpus continued to grow.

Forty Years of Machine Learning for Text Classification

David D. Lewis, Independent Consultant

The core of Lewis's argument, as ESR said earlier in the day, is that for any machine-learning technique, the quality of the learning corpus is much more important than the algorithm used. Bayes is one such algorithm, but there are many other good ones in the literature. Lewis pointed out that all of this has been publicly discussed since the first machine-learning paper was published in 1961.

Observations: "Lots of task[-non-specific] stuff works badly, but task-specific stuff helps a lot." It is important to use different bodies of text for training and for general use, so that you don't train your machine to focus too much on certain types of input (this is a point that Microsoft's Goodman made as well).

As Graham did, Davis emphasized that spam is going to slowly start looking more like natural text, and we're going to have to deal with this as time goes on.
http://www.daviddlewis.com/events/

How Lawsuits Against Spammers Can Aid Spam-Filtering Technology: A Spam Litigator's View from the Front Lines

Jon Praed, Internet Law Group

To a burst of tremendous applause, this talk began with the sentence, "My name is Jon Praed, and I sue spammers."

He brought a legal take on the "not everything is spam to everybody" angle, emphasizing that we need a precise definition of what qualifies as Unsolicited Commercial Email (UCE). In particular, it has been difficult trying to pin down whether the mail was really unsolicited, as this is where the spammers have the most wiggle room. However, if you can track down the spammer, they have, to date, rarely been able to verify that the user asked for mail, and so Praed has been able to successfully prosecute several spammers using this angle. But he doesn't expect this to work forever.

According to Praed, "Laws against spam exist in every state, and more are pending," but he doubts that a legal solution will ever be completely effective as long as spam is lucrative. By analogy, he pointed out that people still rob banks, and that has never been legal.

Praed informed the audience that there are several ways to get back at spammers, including injunctions, bankruptcy, and contempt, and all of these can be very effective. He pointed out that, to be blunt, a lot of these people are desperate low-lifes, and spam has been their biggest success in life. After these legal responses, their lives all get much worse.

It hadn't occurred to me to see spammers as pitiful before, but I can now. Most importantly, Praed stressed that these legal remedies can be very effective, and he strongly warned against taking vigilante action. This is almost always worse than the spam itself, and it only serves to get you in even deeper trouble than the spammer.

Most spam comes from offshore spam houses, abuse of free mail accounts (Hotmail and Yahoo, free signups at ISPs, etc.) and bulk software (which may apparently soon become illegal in certain areas, provided that a law can be found to ban spam software while allowing tools like Mailman and Majordomo). Interestingly, he questioned the idea that IP spoofing is a big problem and claimed that in every case he has dealt with he has been able to track down the messages to a legit source sooner or later.

Suggestion: If you get a spam citing a trademarked product (e.g., Viagra), forward it to the trademark holder and they will almost always follow up on it. Suggestion: Be fast in trying to track down spammers, as some of them have gotten in the habit of leaving sites up long enough for mail recipients to visit, but taking them down before investigators get a chance to take a look. Legal observation: Spam is almost always fraud, and can be prosecuted accordingly.

Praed wrapped up his talk by citing the encouraging precedent that the famous Verizon Online v. Ralsky case set:
(1) that the court is interested in where the harm occurs, not where the person doing harm was when causing it, and
(2) it is assumed that you have to be familiar with a remote ISP's acceptable usage policies, and ignorance is no defense. (Just as you can't say, "I didn't know it was illegal to shoot someone," Ralsky couldn't say that he didn't know Verizon prohibits spam. He had to have known that the AUP wouldn't allow what he was doing, so he deliberately didn't read it.)

That precedent makes the idea of future prosecution of spammers much more encouraging. While, again, legal solutions may never eliminate the spam problem, a precedent like this can be an important supplement to filtering efforts.

Desperately Seeking: An Anti-Spam Consortium

David Berlind, ZDNet executive editor

His talk was primarily about how he receives a huge quantity of email from ZDNet readers, and he can't afford to use any spam-filtering solution strategy that would allow *any* false positives. As one of the speakers said, getting a 0% false positive rate is easy: just classify nothing as spam. Getting a 100% hit rate is also easy: just classify everything as spam. Any solution besides those two is always going to have some degree of error either way, and determining how much of what kind of error you want to accept is up to you.

Most users will tolerate a moderate false negative rate (some spam gets through) if it means that the false positive rate (legit mail is deleted) is very low. In Berlind's case, the false positive rate has to be vanishingly small, because reading all customer mail is, to him, a critical sign of respect for his readers.

Further, his business is also a legitimate mass emailer, sending out millions of free newsletters to users every day, and if Shein's proposal to bill bulk mailers were to catch on, even a very low rate would quickly put a company like Berlind's in the red. One obvious solution, which wasn't mentioned: start charging a subscription for these mailings, and make them profitable. I don't want to see this happen but if it did, then the economics would tilt back toward making things feasible again.

Though Berlind is appreciative of the anti-spam work that is being done, he is skeptical of how pragmatic most of what is being proposed can really be. He feels we need a massive effort to rework the way mail is handled and, to that end, hopes ZDNet can help promote a cooperative effort between the parties working on this. They don't want to be involved – they are journalists and publishers, not standards developers – but they are eager to get things going and want to cover the story as it progresses.

As Shein said, he feels it's a waste for all these talented people to be working on combating penis enlargement offers, and he hopes that we can find a way to get past this and work on real problems "like world peace."

Fighting Spam in Real Time

Ken Schneider, Brightmail

As mentioned earlier, Brightmail provides an ASP service for real-time filtering of both incoming and outgoing mail. As would perhaps be expected, bigger ISPs and networks attract larger amounts of spam: 50% of mail coming into big ISPs and 40% coming into big companies is now spam. Brightmail offers the Probe Network, a patented system of decoy honeypot addresses that gathers data for analysis at their logistics center, and then distributes spam-filtering rules to their clients where a plug-in for $MTA (using the open source or proprietary MTA of the client's choice) can act on the database.

An interesting property of their system is that they have a mechanism for aging out dormant rules as well as for reactivating retired ones, so that the currently active rule set can be kept as lean and efficient as possible. A big source of difficulty for them is legitimate commercial opt-in lists, because things have gotten more shady and blurry over time and it's now hard to distinguish this mail from much of the spam out there. Whitelists help here, but the problem remains difficult.

4th USENIX Symposium on Internet Technologies and Systems (USITS '03)

Seattle, Washington

March 26-28, 2003

[Only a few reports were received on this conference – Ed.]

SESSION: ROBUSTNESS

Summarized by Ajay Gulati

Why Do Internet Services Fail, and What Can Be Done About It?

David Oppenheimer, Archana Ganapathi, and David A. Patterson, University of California, Berkeley

David Oppenheimer and his group studied various causes of failures for Internet services and the effectiveness of various techniques used to mask service failures, as part of their recovery-oriented computing project currently going on at University of California, Berkeley, and Stanford. Today, Internet services have 24/7 expectancy from users, and in spite of a lot of techniques used by the designers for higher availability, they still fail, although such failures are not always directly visible to users. He talked about the difficulties users face in convincing service providers to allow access to their problem-tracking databases, saying "Nobody wants their failure information to be public."

They studied three large-scale services, which he classified as "Online" (a mature service/Internet portal), "Content" (content-hosting service), and "ReadMostly" (mature readmostly Internet service). They got access to problem-tracking databases on the first two and a log of user-visible failures on the third. Oppenheimer made a clear distinction between component failures and service failures: a service failure is one which is visible to end users; component failures are sometimes masked by redundancy and do not cause service failure.

According to Oppenheimer, operator errors were found to be the leading cause of failure in two of the three services. Most of these errors were due to misconfiguration and post-installation changes made by the operators. He showed that operator errors are the largest contributors both in terms of numbers and time to repair (TTR), contributing approximately 75% of all TTR on both online and content services. Oppenheimer also observed that the highest proportion of operator error eventually became visible to users as compared to any other type. In read-mostly services, network errors dominated operator errors and caused 76% of all the service failures. Oppenheimer attributed that to simple and more robust application software and less need for day-to-day maintenance on the part of operators.

Oppenheimer went on to explain various techniques commonly used to mitigate failures, such as online correctness testing, exposing/monitoring failures, redundancy, config checking, and online fault injection. He presented three techniques, namely, component isolation, proactive restart, and pre-deployment correctness testing, which are not currently in use but which could have prevented some failures from occurring. He stressed that most of the techniques work well to mask hardware, software, and network failures, but that we lack efficient techniques to mask/detect operator failures. Also, operator errors are difficult to diagnose and detect before they convert into failures.

Finally, Oppenheimer stated some of the difficulties in extracting data from problem-tracking databases, since data entered into them is sometimes incorrect and cannot be analyzed by writing just a few database queries. David said that a global repository of common errors, and what was done to handle them, might be of great help in such research.

Using Fault Injection and Modeling to Evaluate Performability of Cluster-Based Services

Kiran Nagaraja, Xiaoyan Li, Ricardo Bianchini, Richard P. Martin, and Thu D. Nguyen, Rutgers University

Thu Nguyen started off by saying that today unavailability costs are really high and even 99.9% of availability is not sufficient for Internet services in some cases. Most of the large Internet services use large clusters of commodity computers as their infrastructure. These services are often quite complex and have large design space. Measurement of availability is mostly based on a practitioner's experience and intuition rather than a quantitative methodology. Nguyen proposed a metric combining performance and availability. Before going to the two phases of the metric, he explained a seven-stage piecewise linear model showing various stages that a server goes through, from fault occurrence to recovery: (1) component fault occurs, (2) fault detected, (3) server stabilizes (with performance degradation), (4) component recovers, (5) server stabilizes (still not performing at peak), (6) operator resets, (7) normal operation.

In the first phase, they tried to measure the system's response to individual faults. For the second phase, evaluators need to use an analytical model to combine expected fault load for the server with the measurements taken in the first phase. This gives them a sort of dot product of faults and behavior vectors. To demonstrate the effectiveness of the methodology, the authors studied performability of four versions of PRESS (a highly optimized yet portable cluster-based locality-conscious Web server) against five classes of faults associated with network, disk, node, and application. In phase two of the case study, they compared performability of various versions of PRESS and showed how the model can be used to evaluate various design tradeoffs, such as adding RAID or increasing operator support. Finally, Nguyen discussed some of the lessons learned after applying their methodology to PRESS.

MAYDAY: DISTRIBUTED FILTERING FOR INTERNET SERVICES

David G. Anderson, MIT

Denial-of-service attacks are quite common these days; it doesn't require a high degree of sophistication to implement them. Many measures have been suggested to prevent and avoid DoS attacks, but for various reasons none of them has been globally deployed. Either they require lots of changes in routers, affect normal operation of the server, don't provide any guarantees about immediate relief to the deployer, or take too much time to recover once the attack is detected. Anderson proposed Mayday, an architecture that provides pro-active protection against DDoS attacks, imposing overhead on all transactions to actively prevent attacks from reaching the server.

He made it clear in the beginning that Mayday works only for flooding attacks and not for other smart attacks that could potentially crash a server with incorrect data. He also pointed out that an attacker who can watch all traffic in the network is too powerful to resist; one who targets a particular node or can watch only a part of network traffic, however, can be eliminated, basically because the server has time to detect and prevent the attack before the attacker gets hold of all the nodes. Mayday combines overlay networks with lightweight packet filtering that is efficiently deployed in routers around the server. This filter ring of routers actually provides Internet connectivity and a set of overlay nodes that can talk to the server via the filter ring. Clients communicate with overlay nodes using some application-defined client authenticator. Overlay nodes authenticate the client, perform protocol verification, and then send it to the server through the filter ring using a lightweight authenticator. According to Anderson, this leaves the designer with a lot of choices to trade off among security, performance, and ease of deployment. The performance and robustness of the resulting system depend heavily on overlay routing techniques and the authentication mechanism.

Anderson went on to discuss the pros and cons of various lightweight authentication techniques such as the use of the server's destination port/address, overlay node source address, or any other field in the header for authentication. Similarly, in the case of overlay routing, one can give access privileges to all overlay nodes or only to some of them. In the latter case, access can be by: (1) indirect routing – overlay nodes pass the message to a particular node which has access to the server; (2) random routing – the message is propagated randomly in the overlay until it reaches a node having access to the server; (3) mix routing – each node knows the next hop but not the final destination. Fake traffic can be generated between overlay nodes to confuse the eavesdropper. Once such a protection mechanism is in place, the server can switch between normal mode of operation and secure mode when an attack is detected.

The paper then covered some of the practical attacks that can be used against such filtering-based schemes. Probing or timing attacks, for example, can quickly determine a valid lightweight authenticator and use that to pass through the filter ring. Finally, some of the more sophisticated attacks that might get control over an overlay node and launch internal attacks in the overlay were discussed.

Q: What can Mayday do about attacks that are not well known?

A: It's always possible to come up with some attacks that are not taken care of. But my work mostly concentrated on flooding and DDoS attacks, and it is quite sufficient to handle them.

Q: How long does it take to change the configuration and other things in a router so as to avoid attacks?

A: It depends on how much of a window you want to allow for attackers to detect and cut through the security system. But automated tools these days take minutes, and sysadmins can do this in a matter of hours.

SESSION: RESOURCE MANAGEMENT AND SCHEDULING

Summarized by Ajay Gulati

ADAPTIVE OVERLOAD CONTROL FOR BUSY INTERNET SERVERS

Matt Welsh and David Culler, University of California, Berkeley, and Intel Research

Matt Welsh started off by saying that 9/11 has reminded us of the inability of most Internet services to scale and handle spikes in demand dynamically. Peak workload may be orders of magnitude higher than the average, and managing the performance of a server under such conditions becomes really difficult. Most of the common approaches apply strict limits on resources, such as bounding the number of open sockets or threads or limiting the maximum CPU utilization. He stressed the point that these limits should in some sense be representative of user response time and not just the characteristics of the servers.

Overload management techniques are based on SEDA (staged event-driven architecture), which is a model for scalable and robust Internet services. SEDA decomposes a service into a graph of stages, where each stage is an event-driven service component. Each stage uses a small, dynamically sized thread pool to handle some aspect of request processing. These stages are connected via explicit queues that act as a mechanism for control flow between the stages and a boundary between them. Each stage's incoming event queue is guarded by an admission controller that accepts or rejects new requests for the stage. Each stage can do dynamic resource control, to keep itself in its normal operating mode by tuning parameters for its operation, such as changing the number of threads based on the workload and performance of the stage. Welsh showed a code snippet to demonstrate that overload management is built into the application itself, as any stage can reject a queue request if it feels that accepting it might lead to performance loss. He discussed some of the alternatives to rejecting requests for load shedding. One might start working at degraded performance and tell the user, "This will take time, please wait," as most airlines do. Another way would be to send explicit rejections, such as "We are busy now, try again later." Some Web sites also do some social engineering by sending error messages, saying, for example, "Zipcode is wrong," just to confuse users and gain more time to handle the request. This scheme allows overload control to be performed in response to measured bottlenecks, which is better than having an external control based on general service capacity. Also, handling rejected requests can be done on a stage basis, since the application knows which stage was bottlenecked for a given request. Welsh presented three mechanisms of overload control in SEDA:

1. Performance metric: A 90th percentile response time is used, which is a realistic and intuitive measurement of client-perceived system performance. The response time value may be set by the system administrator and might depend on request type – for example, not kicking out a user with lots of stuff in a shopping cart.

2. Response-time controller design: The controller associated with each stage observes a history of response times and throughput of the stage, and adjusts the rate of acceptance for new requests to meet the goals of performance.

3. Class-based differentiation: This scheme can prioritize requests from certain users over others and handle service level agreements based on what different clients are paying for the service. The authors developed a Web-based email service, which was a clone of Yahoo mail, allowing users to access and manage their emails. During large load spikes, SEDA compared favorably to servers with fixed connection limits in meeting 90th percentile targets. Also, SEDA's rejection rates were lower than other approaches. Finally, multi-class service differentiation led to different rejection rates for various classes, with requests from one class meeting the 90th percentile response time more frequently than the other, less-preferred class.

Q: Is this architecture designed for a single machine or a cluster?

A: It will work very well on a cluster as well. In that case, different stages might be implemented on different machines.

Model-Based Resource Provisioning in a Web Server Utility

Ronald P. Doyle, IBM; Jeffrey S. Chase, Omer M. Asad, Wei Jin, and Amin M. Vahdat, Duke University

Summarized by Ajay Gulati

Jeff Chase started off by saying that provisioning of shared resources at large-scale network services is one of the biggest challenges for system research today. The authors focused on automation of on-demand resource provisioning for multiple services hosted by a shared server infrastructure competing for resources such as memory, CPU time, and throughput from storage units. A slice of the these resources is allocated to each service to meet service quality targets decided in SLAs. Chase presented a novel model-based approach, in which internal models of service behavior are used to predict initial resource allotment and are changed dynamically using a monitoring and feedback system. He introduced the notion of a "utility operating system" that handles resource management across the utility as a whole. The authors observed that network service loads have been studied extensively, have common properties, and can be represented by models fairly accurately. Chase went on to discuss some of the models derived from basic queuing theory and showed that behaviors predicted from these models were quite similar to actual observed behaviors. Resources such as memory, storage I/O rate, and response time were closely modeled by parameters like requests/s, average object size, average CPU demand/req., memory size for object cache, and peak storage throughput in IOPS.

Once the models were obtained, a resource provisioning algorithm was used to plan least-cost resource slices and get an "allotment vector" for each service, representing CPU, memory, and storage allotments. This algorithm consists of three main primitives:

1. Candidate – plans initial allotment vectors that are guaranteed to meet SLA response time targets for each service. It does not consider resource constraints, which is done by the other two primitives.

2. LocalAdjust – takes a candidate allotment vector and request arrival rate as input and outputs an adjusted vector adapted to local resource constraint or surplus exposed during initial assignment. It basically constructs an alternative vector that meets target within the resource constraints.

3. GroupAdjust – works on a set of candidate vectors to adapt to a resource or a surplus exposed during assignment to meet system-wide goals. For example, it can reprovision available memory to maximize the hit ratio across a group of hosted services.

Chase presented various graphs showing how these primitives allocated surplus memory to optimize global response time and flexibility of the model-based approach in adapting to changes in load or system behavior. Finally, he presented the evaluation methodology for the technique, for which they used a cluster of load-generating clients, a reconfigurable switch, DASH Web server, and network storage servers accessed using DAFS (direct access file system). The results showed that the predicted and observed resource utilizations were fairly close and that model-based provisioning is quite effective for resource management on cluster utilities.

CONFLICT-AWARE SCHEDULING FOR DYNAMIC CONTENT APPLICATIONS

Cristiana Amza, Alan L. Cox, Rice University; Willy Zwaenepoel, EPFL

This work focused on scaling a dynamic content site (e.g., Amazon.com) through a new technique called conflict-aware scheduling. Dynamic content sites consist of three tiers: Web server, application server, and database. The need for scaling the main site arises because there may be many clients accessing such sites. Replicating the front tiers is easy because they do not contain the dynamic content; the data that changes is stored in the database. Currently, state-of-the-art dynamic-content Web servers rely on a single very expensive database supercomputer to satisfy the volume of requests. The solution introduced was to scale the database tier by using replication on clusters. This allows a low-cost solution and, most importantly, incremental scaling and strong consistency at the same time. Amza justified the choice of replication based on characteristics of dynamic-content applications such as locality (hot-spots in the workload) and higher read-query complexity as compared to the write-query complexity. On the other hand, traditional replication has a known down side. It is well known that one cannot get both scaling and strong data scheme (for consistency), asynchronous writes (for scaling), and conflict avoidance (for improved scaling).

The TPC-W e-commerce benchmark with its three workload mixes – browsing, shopping, and ordering – was used to evaluate conflict-aware scheduling. The group used commodity hardware and software components in the evaluation: the Apache Web server and PHP module for the Web and app servers and the MySQL database engine. The only correct protocol that could previously be used to satisfy TPC-W's requirement for strong consistency was the eager protocol with synchronous writes. The scaling of the eager protocol is poor and gets worse with increasing writes in the mix and larger clusters. Amza's results showed that conflict-awareness compared favorably to both eager and conflict-oblivious lazy replication over a large range of cluster sizes and conflict rates. Scaling was close to ideal in the conflict-aware protocol for the browsing and shopping mixes of TPC-W up to large cluster sizes. The ordering mix was a pathological case for the protocol. It had a very high fraction of writes (50%) and consisted mostly of ordering transactions, which were very long and held locks on all useful tables. The scaling for this mix flattened at 16 databases, although conflict awareness still brought significant improvement over eager, which did not scale at all in this mix.

INVITED TALK

FAST, RELIABLE DATA TRANSPORT

Michael Luby, Digital Fountain, Inc.

Summarized by Xuxian Jiang

An interesting talk on the data transport issue.

Luby first examined the weakness existing in traditional data transport (UDP, TCP etc), especially the interconnection between rate control and reliability mechanism in TCP data transport. In traditional data transport, the rate control is highly constrained by the quantity of unacknowledged data allowed, and loss estimation is mainly based on acknowledgments received.

Secondly, he talked about the digital fountain data transport approach, which decouples the relationship between reliability and flow/congestion control. The reliability is provided without using feedback. Such feedbackless-based reliable data transport has many desirable features in data transport: (1) speed over large distances and loss networks; (2) predictable speedy control of data transport; (3) global transport with local performance; (4) massive scalability; (5) flexibility in choosing a flow/congestion control mechanism.

Such an approach can be widely adopted in many situations: (1) wireless and satellite communication; (2) enable receiving when receivers have intermittent connectivity; (3) enable data transport even in highly unreliable communication, which may experience unknown or variable loss.

The digital fountain approach is analogous to a water fountain: it doesn't matter what is received or lost, it only matters that enough is received. The sender sends encoded data at the rate decided by the selected flow control mechanism and the receiver receives some of the encoded data and is able to reconstruct the original data. It is not necessary for the receiver to provide feedback for the transport, what really matters is that enough information is received. The encoding and decoding technology does the essential core of the magic.

There are serveral erasure codes which can succeed in the encoding for this purpose, including Reed-Solomon codes (1960, Reed and Solomon), Tornado codes (1997, Luby et al.) , LT codes (1998, Luby), and Raptor codes (2001, Shokrollahi). The common properties of these digital fountain codes include: (1) encoding only as long as data flows; (2) recoverability of data from required encoding, (3) low complexity for encoding and decoding; (4) ability to encode very large data; (5) ability to produce an unlimited flow of encoding.

The pros and cons of example digital fountain codes were examined in terms of data length, encoding length, flexibility to receive from multiple sources, memory requirement, computational work, reception overhead, failure probability, etc. Interested readers are referred to the corresponding encoding papers, especially LT codes (1998, Luby) and Rapter codes (2001, Shokrollahi).

The talk concluded with some typical and interesting application scenarios and on-going projects, such as CINC deployment, broadcasting data to automobiles, and robust communication in challenged environments.

2nd USENIX Conference on File and Storage Technologies (FAST '03)

San Francisco, California

March 31–April 2, 2003

KEYNOTE ADDRESS

Data Services – From Data to Containers

John Wilkes, Hewlett-Packard Labs

Summarized by Scott Banachowski

After a few opening remarks from conference chair Jeff Chase, John Wilkes of HP Labs kicked off the conference with the keynote address. As he introduced the talk, Wilkes clued any bored listeners to look for themes in the photographs scattered throughout the presentation slides. Wilkes's keynote foreshadowed many of the themes that would appear in the following conference sessions: how to deal with the rising complexity and ability of storage systems and meet our expectations for new capabilities.

The solution to storage problems lies in using Quality of Service (QoS), because it encompasses everything we'd like to say about our storage problems. Therefore the path to a solution includes defining data QoS needs, using storage QoS abilities, and automating storage and data management.

The enterprise IT plan is a complex, large-scale system; Wilkes demonstrated this assertion with a convoluted enterprise IT architecture schematic. Some of the requirements of these systems resemble those of existing large-scale scientific applications, so we can look to these applications as predictors of future trends: they access huge quantities of data using specialized access patterns. As examples, Wilkes reviewed the storage systems required by the human genome sequencing engine and the CERN electron collider.

It is important for storage researchers to distinguish between data and storage, because we often confuse the data's attributes with those of the containers. Wilkes outlined many data metrics, pointing out that it is easy to measure amounts, rates of growth, or access patterns, but difficult to get a handle on resilience or security. It is important to consider that not all data are created equal (some have little value, some never change, and some can be regenerated) and that data have differing lifetime and security requirements. All of these attributes can be captured by QoS, which provides storage systems with both a set of objectives and a contract for service. In a brief overview of storage containers, Wilkes covered new technologies such as MEMS, MRAM, and smart disks (bricks). He proposes using the SNIA shared storage model to get a handle on how to set up large storage systems.

Recognizing that storage systems are just a part of larger computing systems, it is important to grapple with the management challenges. The main challenge is keeping administration costs low by automating tasks. We have already generated many techniques for managing systems, so Wilkes recommends that we learn to use existing techniques before creating new ones.

The key to develop future storage systems is to embrace complexity. In this era, we must use a data-centric viewpoint for putting everything together, and this must be driven by QoS. Wilkes summarized the problem as moving from "some of the parts to sum of the parts" as the required step to accessing data anywhere and anytime. During the Q&A, someone noted the agricultural theme running through Wilkes's slides, and asked what the equivalent to fertilizing and weeding is in the storage field. Wilkes chuckled but supplied no answer.

SESSION: INTERNET SCALE STORAGE

Summarized by Preethy Vaidyanathan

Pond: The OceanStore Prototype

Sean Rhea, Patrick Eaton, Dennis Geels, Hakim Weatherspoon, Ben Zhao, and John Kubiatowicz, University of California, Berkeley

Peter Honeyman from the University of Michigan chaired this first session of the conference. Sean Rhea presented OceanStore, a global-scale storage system, and the prototype Pond, a self-organizing, self-maintaining, secure Internet-scale file system among untrustworthy hosts. (This paper received the Best Student Paper award.)

The main challenges in a global-scale storage system are availability and manageability. All the resources in the system are virtual, and replication is used to provides fault tolerance and reliability. Tapestry, a decentralized scalable object location and routing system, is used in this prototype to identify the resources.

The prototype uses erasure codes and a modified Byzantine agreement protocol to provide fault tolerance and consistency among the replicas. Erasure codes are more durable than data mirroring for the same space. The Byzantine protocol was modified to decrease the number of messages passed to make replica copies consistent. Each object is assigned a primary replica by an inner ring server. The updates are in-place among the inner-ring servers.

The Pond prototype was tested using two experimental testbeds at Berkeley and PlanetLab (*http://www.planet-lab.org*). Andrew benchmark test results compared to NFS show improvements in read access and performance degradation on writes. Rhea explained the write cost and limitations as due to erasure code. This cost would be alleviated when servicing large writes. Pond has good performance in other benchmark experiments.

Pond source code is available at *http://oceanstore.cs.berkeley.edu.*

Data Staging on Untrusted Surrogates

Jason Flinn, Intel Research Pittsburgh and University of Michigan; Shafeeq Sinnamohideen, Niraj Tolia, and M. Satyanarayanan, Intel Research Pittsburgh and Carnegie Mellon University

Mobile computers are increasing in popularity, and Jason Flinn presented a mechanism by which surrogates close to the mobile device can be used as data staging stations, reducing transfer latency. This architecture provides less latency for the client when accessing data, as the data is now retrieved from the surrogate and not the file server.

The data staging architecture consists of a client proxy that observes file system traffic and initiates a surrogates help if need be. A data pump near the file server acts as an intermediate point between the client and the server. When a client accesses data, the pump authenticates the message, reads from the file system, encrypts it, and sends the cryptographic hash to the client and the data to the surrogate. All this is done through a secure channel. The client reads the data from the surrogate, decrypts it and checks validity using the hash.

The architecture design is independent of the underlying file system. For experimentation, this design is implemented on the Coda file system. The data staging design assumes client, file server, and data dump file system to be trustworthy and entrusts the surrogate and the network.

Flinn presented two experiments to test the data staging architecture. The first tested the performance of aggressive perfecting in this architecture. The results shows that surrogate data miss affects the performance more than the surrogate having data that are never accessed. The second set of experiments tested what factors affect the performance of the system. The results showed that latency hurt the performance more than bandwidth, which iterates the assumption for this work.

The source code of this project can be obtained at *http://info.pittsburgh.intel-research.net.*

Plutus: Scalable Secure File Sharing on Untrusted Storage

Mahesh Kallahalla, Hewlett-Packard Labs; Erik Riedel, Seagate Research; Ram Swaminathan, Hewlett-Packard Labs; Qian Wang, Pennsylvania State University; and Kevin Fu, Massachusetts Institute of Technology

Mahesh Kallahalla presented a cryptographic storage system for secure file sharing. Data security is different from network security, and Kallahalla described Plutus, a decentralized key management system where all keys are handled by the client with minimum trust on the server. Plutus architecture is scalable and secure, as there is no single point of failure, because the client does the encryption and decryption work and the server acts as a data store.

Blocks of the file are encrypted with symmetric key called file-block key. There is a file-lockbox key for the file. To minimize the number of keys generated, files with similar attributes are grouped into file groups. All the files in the file group share the same file-lockbox key. The architecture differentiates between readers and writers by using file-verify and file-sign public-private key pairs.

File updates might result in file group fragmentation. This is handled in the Plutus architecture by key rotation. Each update results in the creation of a key version, not a new key. The owner of the data can only generate the next version. The readers keep track of the newest version of the key, and previous versions can be rolled back from the current version. A prototype has been designed using OpenAFS. The Plutus system is tested using UNIX traces and synthetic benchmarks. Kallahalla concluded that in spite of the overhead of encryption and decryption, Plutus performance is favorable with key points such as file grouping, key rotation, and lazy re-encryption.

SESSION: FILE STORAGE

Summarized by Scott Banachowski

Metadata Efficiency in Versioning File Systems

Craig A. N. Soules, Garth R. Goodson, John D. Strunk, and Gregory R. Ganger, Carnegie Mellon University

The first talk of the Storage Session, chaired by Margo Seltzer of Harvard, came from Craig Soules of CMU. Soules presented the Comprehensive Versioning File System (CVFS), a log-based file system that keeps old versions of data using structures that reduce the storage overhead of metadata, essentially trading back-in-time performance for space.

A versioning file system keeps multiple versions of data for backing out mistakes, failure recovery, and history analysis. Current versioning systems write a new copy of the metadata for each version of a file, leading to high metadata overhead. The goal of CVFS is reduction of storage overhead, which is accomplished by combining journaling and b-trees.

The system maintains the most current metadata version and differences between previous versions, stored in a journal. The journal approach saves space because it only records incremental changes, but retrieving previous versions is not efficient because all previous versions must be unrolled in sequence to recreate the desired version. To reduce the roll-back time, CVFS occasionally store an entire version, i.e., a checkpoint. Multiversion b-trees provide an efficient structure for storing multiple versions of data using keys comprised of a name/time pair. B-trees are more efficient for single-lookup operations than journals, so CVFS uses b-trees to maintain directories, where lookup is the most common operation and modifications are infrequent.

The performance of CVFS was evaluated by playing back 1 month of NFS traces that contained 164 GB of data traffic from 30 users. Compared to conventional versioning system, CVFS saved 53% in file metadata and directory space. The performance relative to other systems diminishes when keeping less comprehensive data, for example storing only on-close versions or periodic snapshots. During the Q&A, someone questioned Soules about the usefulness of such comprehensive versioning, considering that many writes may never be seen by the file system due to caching, to which Soules replied that it depends on the application.

yFS: A Journaling File System Design for Handling Large Data Sets with Reduced Seeking

Zhihui Zhang and Kanad Ghose, State University of New York, Binghamton

Recognizing that most file system designs are based on file-size assumptions from years ago, Zhihui Zhang presented yFS, with the goal of handling large and small files with equal ease. yFS was implemented in FreeBSD.

The features of yFS include extent-based allocations, multiple inode formats, b*-tree structures for managing inode data, support for large directories, and lightweight logging. The file system divides the disk into allocation groups, each containing its own metadata. Space is allocated to files in both fragments and blocks, although, unlike other segment-based systems, there is no restriction concerning which segment a file begins, and fragments may have variable size. For large files, there are competing goals of contiguity and locality; yFS scatters large files across allocation groups.

yFS was compared with FFS enhanced with Soft Updates using four benchmarks: a kernel build, the extraction of an archive file, the PostMark benchmark, and a file system aging test. Without Soft Updates, the synchronous metadata updates of FFS lead to performance that is not comparable to yFS. However, even with Soft Updates, yFS outperformed FFS in all measurements except for one phase of the compilation benchmark.

Semantically-Smart Disk Systems

Muthian Sivathanu, Vijayan Prabhakaran, Florentina I. Popovici, Timothy E. Denehy, Andrea C. Arpaci-Dusseau, and Remzi H. Arpaci-Dusseau, University of Wisconsin, Madison

Remzi Arpaci-Dusseau explained the concept of "semantically-smart disk systems." Storage systems are currently layered into the file system and the disk or RAID system; the origin of this separation is the hardware/software boundary, but now each layer is becoming increasingly complex. Because layers are separated by a bus or protocol, the semantics of file system operations are lost in the disk layer, so semantically-smart disks aim to reacquire this information in the disk layer using both offline and online techniques. The approach allows RAID systems to exploit their processing and memory capability without changing their interface.

Semantically-smart disks understand file system operations, and discover the layout of on-disk structures and operations by reverse engineering the block stream. Static knowledge of file system layout is determined with the aid of a gray-box tool called Extraction of Filesystems (EOF). EOF creates a disk traffic pattern that, when observed by the disk, provides hints that allow the smart disk to determine the types of blocks and their layout. With this information, the smart disk may then take steps to improve performance, for example by automatically caching inodes and directory blocks in NVRAM.

The paper includes several case studies for using semantically-smart disks (SDS), but during the talk Arpaci-Dusseau focused on adding its secure deletion feature. By detecting when files are deleted, the disk can automatically remove its dead-blocks so that they cannot be reread using magnetic microscopy techniques. The file system cannot reliably do this, because it may absorb writes in cache or may leave stray blocks on the disk. Using SDS it is possible to detect deletes, which traditional RAID systems cannot do, and overwrite the file's data blocks with patterns multiple times, so they cannot be reread. In the Q&A session, someone asked if it is possible to also learn semantics through the interfaces of object-based storage. The approach may be similar in philosophy but it is still an open issue.

INVITED SESSION: PETABYTES AND BEYOND

Reagan Moore, San Diego Supercomputer Center; Thomas M. Ruwart, University of Minnesota Digital Technology Center, Intelligent Storage Consortium; and Clod Barrera, Director of System Strategy, IBM Systems Group

Summarized by Preethy Vaidyanathan

The "Petabytes and Beyond" panel was hosted by Jeff Chase of Duke University.

Reagan Moore started out by listing the challenges that will be important in the future for applications dealing with large amounts of data. He pointed out that, based on the current data growth trend, handling large amounts of data would require organizing data into collections. Some major challenges would be to tag these large amounts of data to generate information, organizing information into collections, querying collections for relationships (data mining) and organizing relationships in concept spaces.

Moore presented some of the projects dealing with petabytes of data: NASA Earth Observing Satellite, Large Hadron Collider, and NSF Teragrid. Teragrid is a project that handles large volumes of data in a distributed environment. It aims to provide a collective set of resources greater than any one site can provide. The participating groups in this project are the San Diego SuperComputer Center (SDSC), the National Center for Supercomputing Applications (NCSA), the Argonne National Laboratory (ANL), and CalTech. Some of the challenges include managing the resources over WAN, discovery of data, and naming conventions.

Other works include Digital Library, data grids like Grid Bricks that build cheap storage systems (bricks) using common disks, and IDE drives and tape archives. A data grid accesses distributed resources and should manage namespace, user authentication, and user access control across bricks.

Moore concluded by presenting specific directions for some of the challenges. Discovery of data in this environment is essential. This can be provided by an infrastructure-independent naming convention. Data discovery is implemented by cataloging a database that manages the logical name and abstracts the physical location. Because of WAN latency management is another challenge. The SDSC Storage Resource Broker focuses on this problem. One solution is to enable the application at the destinations to pull data from remote resources by sending an aggregate message that eliminates the large overhead of a number of individual messages.

Thomas Ruwart presented "Storage on the Lunatic Fringe." He introduced the DoE Accelerated Strategic Computing Initiative (ASCI), High Energy Physics (HEP), NASA Earth Observing System Data Information Systems (EOSDIS), DoD NSA and DoD Army High Performance Computing Centers, and the Naval Research Center as the "lunatics" who are dealing with petabytes of data. The problem that these projects are looking at is what the industry will face in 5–10 years, so a lunatic in this scenario corresponds to a visionary.

Ruwart gave a brief historic outline of large-scale computing resources, starting with supercomputer centers in the '90s to the current ASCI Q and the ASCI Red Storm, Purple, and NASA RDS.

He then went on to present some specific problems. In HEP, thousands of scientists look at large datasets with different access pattern considerations. The Data Grid project is a distributed architecture and the main issue is managing data for long periods of time. How to handle a trillion files is the challenge for the NSA project. He said that considering 256 bytes of metadata per file, for a trillion files this itself would result in 256 TB. If this number was not enough to overwhelm, Ruwart raised the question of backups in this scenario. Other problems raised included searches for content in these datasets, security, and availability.

With data on such a scale, legacy block-based file systems will not work. A vision for the future is more intelligent storage devices whose functionality would migrate from the operating system to the storage device. The storage device apart from storing data, should handle managing and administering data.

Some of the technologies addressed by the Lunatic Fringe include object based storage devices, intelligent storage, and data grids.

Clod Barrera presented "Size and Shape of Things to Come." He stated that the main concern in the future apart from performance and scalability, would be data management. He stressed that the ease and cost of management should be taken seriously.

He presented life science research as an audience who will need to deal with petabytes of storage requirements. At this requirement level, data access consistency, error recovery, high performance, and a scalable file system would be essential. One such project dealing with a scalable file system is the Storage Tank project. A storage network of petabyte scale might start with a fiber channel but could be other technology such as IP over SCSI. Holographic storage is an emerging technology that simplifies content searching. Barrera concluded by pointing out that an intelligent system with knowledge of the different storage technologies can map the application to the right storage technology for optimal performance.

SESSION: STORAGE SYSTEMS

Summarized by Preethy Vaidyanathan

USING MEMS-BASED STORAGE IN DISK ARRAYS

Mustafa Uysal and Arif Merchant, Hewlett-Packard Labs; Guillermo A. Alvarez, IBM Almaden Research Center

Mustafa Uysal presented the first talk (given Best Paper award). Two widely used storage technologies are NVRAMs and disks. NVRAMs are faster, more expensive, and less reliable, whereas disks are slower and cheaper devices. Current I/O performance is still bounded by disk access.

MEMS is a new technology being developed with access characteristics between NVRAM and disk. They provide persistent and nonvolatile storage like disks but have different characteristics. They have no rotational delay, with slow moving parts making acceleration easy and high density storage providing a short seek distance. Uysal expanded upon the various architecture alternatives for this new device, assuming it was cost-effective and useful.

Uysal presented five array architectures. MEMS replacing disk (MEMSdisk), MEMS replacing NVRAM (MEMScache), and three hybrid architectures: MEMSmirror, Logdisk, and DualStripe. In the hybrid architectures, MEMS does not totally replace any device. MEMSmirror has a MEMS device as mirror for the disk. No access or layout change is needed here: reads of data not in cache are handled by MEMS and writes are propagated from NVRAM.

Logdisk and DualStripe have data from MEMS mirrored in disk for redundancy.

In the Logdisk architecture, updates to MEMS are propagated to disk in a log-structured manner. Reads are handled by MEMS and sequential reads can be effectively handled by disk. DualStripe is a dynamic architecture. Reads if cache miss are handled by MEMS for a short queue length and handled by disk for queues greater than a threshold or if it's a large sequential read. Sequential access detection is implemented in firmware.

The different array architectures were tested for synthetic and trace workloads. The overall conclusion: having a MEMS device in your storage array architecture will be cost-effective and efficient. The hybrid architecture provide a good cost/performance benefit and Logdisk provides the most cost-effective architecture.

In the question period, Uysal clarified that this work studied the placement of MEMS in disk arrays so the scheduling policy was the same as disk. Another question led to the conclusion that a possible 3-level hierarchy of NVRAM, MEMS, and disk would be an extension to this work when the exact characteristics of the MEMS device is known.

OPTIMIZING PROBE-BASED STORAGE

Ivan Dramaliev and Tara Madhyastha, University of California, Santa Cruz

Probe-based storage or MEMS is a new technology with characteristics such as low power consumption, high density, high parallel tip movement producing high throughput, and no rotational movement. These devices can be modeled to different design points each resulting in different performance measures. This would answer the question of which workload would best suit probe-based architecture. Madhyastha outlined a parameterized analytical model to compute average request latency for MEMS devices.

The Probe-based storage is characterized by X and Y movements, with no rotational movement such as disks use. The data layout goal is to minimize movement for consecutive requests, similar to traditional disks. There is a mobile part which moves when servicing a read or write request. The repositioning time comprises seek time and time taken while moving with a constant velocity in Y direction to access data (transfer time).

Madhyastha illustrated how these two times can be calculated for this device. In the model, the seek time depends on bit per tip, distance moved in the X and Y directions, bit width, acceleration, and settle time. The transfer time is proportional to the data per tip. The service time model gives a formula into which values can be plugged to compute the service time of the request.

This model was tested by comparing it with simulation results for a wide range of parameters. Block level traces were used to test the performance of the model. The error computed was small (up to 15%) when compared to simulation.

In the Q/A section, Madhyastha agreed with the observation that the reliability of these devices should be considered in future research.

ARC: A SELF-TUNING, LOW OVERHEAD REPLACEMENT CACHE

Nimrod Megiddo and Dharmendra S. Modha, IBM Almaden Research Center

Dharmendra S. Modha discussed how to manage cache or what page to replace to maximize hit-ratio. The cache replacement strategy presented was with respect to demand paging.

Two popular techniques, Least Recently Used (LRU), and Least Frequently Used(LFU), are algorithms that have long been used for cache replacements. LRU captures locality of reference; LFU, the frequency of reference. Modha presented a new scheme, ARC, that captures both these characteristics by maintaining two self-consistent lists. The first lists the pages seen only once and the second lists pages seen at least twice.

In ARC a sliding window of the size of the cache is used to determine what page to replace. The sliding window overlaps the two lists and the percentage of overlap dynamically varies depending on the workload. This implementation has low computational overhead and is tested with a wide range of trace data. ARC consistently outperforms LRU and has similar performance to an offline replacement algorithm that is optimally tuned for the workload.

In the Q&A Modha clarified that the sliding window starts initially with the midpoints in the two lists and the sliding movement is sensitive to the request.

The source code is available at *http://almaden.ibm.com/cs/people/dmodha*.

SESSION: SHARING BLOCK STORAGE

Summarized by Nate Edel

FAÇADE: VIRTUAL STORAGE DEVICES WITH PERFORMANCE GUARANTEES

Christopher R. Lumb, Carnegie Mellon University; Arif Merchant, Hewlett-Packard Labs; and Guillermo A. Alvarez, IBM Almaden Research Center

Christopher Lumb described work at HP Labs on the Façade system, a storage system that provides service level guarantees. Unlike existing solutions, which don't differentiate between workloads, the Façade system is able to adapt to changing workloads to attempt to meet each separate workload's service level objective (SLO).

Façade works by intercepting storage requests and prioritizing them based on their SLOs. SLOs are latency bounds at a given rate of I/O operations; a workload may have separate SLOs for read and write operations, and multiple workloads/SLOs may share one RAID.

The system has three components: the I/O scheduler, the controller and monitor, and a target queue. The I/O scheduler receives all requests and then timestamps and queues them. The scheduler then watches the wait times and dispatches the IO requests based on earliest deadline to the target queues. The adaptive controller and monitor monitors the response times and workloads; if requests aren't meeting deadlines, it will makes changes to the target queue behavior.

The target queue maintains latency requirements by shrinking in depth when latency targets are not met to decrease throughput, and growing in depth when latency targets are met to increase throughput. The target queue growth rate is conservative, because shrinking queue depth is harder, as I/O operations have to first drain the queue to the desired depths, and further I/Os have to finish to allow new operations to enter the queue.

Lumb presented benchmark numbers for a sample set of workloads with three different SLOs, and showed that without Façade, the different workloads would have roughly equal latency and would not meet their targets. With Façade, by reducing the I/O rate for a continuous process, the other two burst workloads met their latency targets almost all the time. There were spikes during transition from high throughput to SLO compliance; he noted that these were optimizable but that the best way to do so was not clear.

Finally, Lumb compared two workloads on separate arrays against Façade on a higher resource single array; with Façade, the combined array had essentially the same latency for both processes and the same throughput for the burst workload. However, with Façade, the continuous workload could take advantage of bandwidth unused by the bursty workload when it was not active, allowing some degree of overprovisioning to be avoided.

In the question-and-answer period, someone asked if Façade works well when all workloads compete equally, or what happens when workloads compete for different resources? This was noted as not clear, but it would be an interesting experiment, which may be able to account for some aspects, and they could increase the complexity of the model by taking into account other aspects, but Lumb was not sure if it would make a difference. Someone else asked if the project had tested other metrics, such as bandwidth? The project did not, but it would be simple to use bandwidth rather than request rate if that was preferred. Another question was how Façade prevented starvation? It doesn't handle admission control; they assume the system could eventually service all requests.

Design and Implementation of Semi-preemptible IO

Zoran Dimitrijevic, Raju Rangaswami, and Edward Chang, University of California, Santa Barbara

Zoran Dmitrijevic described work done on developing a system for preemptible disk-access. The key benefit of preemptibility is decreasing the initial latency of high priority IO requests. The size of other, lower priority I/O requests will not have as great an impact on these requests; and premptibility may be able to improve other scheduling.

Overall, the time to execute a read request depends on several factors – wait for seek, rotational delay, and the maximum IO size and maximum disk IO size. These are typically selected to balance throughput and latency with all tasks being equal; without normal, non-preemptible IO, the total response time is the waiting time plus the service time. For an average command, the expected wait time is one-half of the service time for an average IO. Preemption allows elimination of waiting time, and its key metric is the reduction of expected waiting time.

Without preemption implemented on the disk itself, the proposed implemention of semi-premptible IO splits lower priority IO into several commands. This allows the controller or OS to interpose higher priority IO requests into the stream of smaller requests, a technique called chunking; because of disk read prefetching and buffering, the overhead of IO bus traffic and kernel CPU activity remains close to constant.

Along with chunking, Dmitrijevic discussed two other techniques. The first, just-in-time-seek, attempts to calculate pre-seek slack using the rotational delay to make that time preemptible – pre-seek slack can be used for "free" perfecting and seek-splitting. The second, seek-splitting, takes long seeks and splits them into multiple shorter seeks. This allows preemption of seeks and takes advantage of rotational slack. The down side is that multiple short seeks take slightly longer than a single direct seek.

There were several implementation issues addressed: disk block mappings needed to be implemented, the optiminal chunk size had to be determined, and rotational factors and seek curves were analyzed. For the optimal chunk size, Dmitrijevic noted there were both lower and upper bounds: a minimum size, below which throughput suffered, and a maximum size above which it seemed that prefetching might not continue to be a factor. SCSI and IDE drives showed similar curves, although SCSI was more efficient for a range of smaller transactions.

The experimental implementation was a user-mode driver running on the SCSI generic driver on Linux, and tested using traces from a specially instrumented Linux kernel. It was tested with a simulated workload of random IO using FIFO and elevator scheduling, as well as with TPC and multimedia streaming traces. Expected waiting time is much lower with some workloads, and only very slightly worse throughput with random accesses.

The talk closed by noting that the contributions were measuring the preemptibility of disk accesses and showing that preemptibility can cut down waiting time. Noted future directions were the use of semi-preemptible IO in scheduling algorithms, and a QOS disk scheduler for Linux.

John Wilkes asked if it would be possible to implement this on the on-disk controller. The response was that while it may be possible, existing drives would need more onboard computing power on the drive; while Dmitrijević was unsure how much more would be needed, he indicated that it should be possible.

BLOCK-LEVEL SECURITY FOR NETWORK-ATTACHED DISKS

Marcos K. Aguilera, Minwen Ji, Mark Lillibridge, John MacCormick, and Erwin Oertli, Hewlett-Packard Labs; Dave Andersen, Massachusetts Institute of Technology; Mike Burrows, Microsoft Research; Timothy Mann, VMware; and Chandramohan A. Thekkath, Microsoft Research

Marcos Aguilera described the Snapdragon file system prototype, which was created to test a security model for network attached disks (NAD) storage. This was contrasted with standard distributed file systems, with disks on a server and all accesses via that server; in that case, the server is the main performance and reliability bottleneck.

NAD is like a storage area network (SAN), but is simpler because there is no separate network for storage; in either case, a server is used only for metadata, and file access is direct to disks over the (shared or separate) network. Because the server is out of the data access path, this offers better scalability and better reliability on server failover. The problem is that the server no longer guarantees security; a bad client can overwrite data for other clients, or hackers or malicious/viral code can access whole disks.

Eliminating the security problem is non-trivial: Ddisks are dumb, low-level devices with no idea of permissions or even files. The solution is to make disks smarter. One proposed way of doing so is to use higher-level objects rather than block I/O, but block I/O has important advantages. It is simple and well understood, so people would like to keep it.

Achieving security with block I/O is possible. The naïve way is to store with each block the owner, group, and mode. However, that list can grow quite large and is tied to particular OSes. Capability-based security is the better alternative: the server provides a capability, and then the client passes the capability to the disk with a write request. The Snapdragon system uses capabilities that contain a block range, permissions (r or r/w), and a cryptographic signature. These are checkable by relatively simple disk hardware.

The cryptographic system used is Message authentication codes (MAC); these are like digital signatures but are short and easy to compute, and use a shared key rather than a public key. The detailed protocol was originally proposed for the NASD system in 1997. In order to allow capability revocation, a revocation list is kept on the disk, sent directly from the server. This will increase in size over time, and is bounded by garbage collection; in part, this is achieved by the expiration time of capabilities, but large numbers of revocations within a short time can fill it. It is further limited by capability groups – the server can invalidate whole groups – while new capabilities can be issued by the server if a valid client gets rejected.

The system is able to avoid replay attacks; because timestamps/counters have drawbacks, the combination of bloom filters and epoch numbers are used. Bloom filters check for duplicate messages; the epoch number is a per-drive counter. The down side is a single rejected request per client per epoch change; to avoid this, the drive keeps a window of one old epoch's filter. As long as clients stay in regular contact, no messages should be rejected.

The resulting system provides a low-level block device, secure NAD devices, and a very high degree of flexibility and portability. The Snapdragon prototype is a client and server kernel-space implementation on Linux. Each disk is implemented using a simple program. The system was benchmarked using low-end hardware (400Mhz Intel Celeron-based machines for the client, server, and disks) over gigabit Ethernet; the resulting system is approximately 16% slower in throughput and 5% slower in latency than the system without security.

The actual protocol overhead is small; capabilities are a 116-byte block, and only 128Kb are required on the disk. Similarly, the software complexity on the disk is small – it only has to be able to compute the MAC, verify the capability, and check the bloom filters for a duplicate; as a result, it is suspected, but not verified, that it will be implementable in firmware for existing disk hardware.

In the Q&A section, the system drew praise from Garth Gibson, who went on to ask what the effect of a difference between block model and object model would be. The response was that minimizing change on disk was the motivation, as it was for the bloom filter.

Someone noted that the bloom filters offered only a probabilistic guarantee, and would have some false positives, and asked how this was handled. The system adds a nonce to each request and has a false positive rate of about 0.1%; while this may allow a denial of service attack, this could also be accomplished by bombarding the drive with any sort of invalid request.

WORK-IN-PROGRESS REPORTS

Summarized by Nate Edel

PARALLEL EXTENSIONS TO THE DAFS PROTOCOL

Peter Corbett, Netapp

Peter Corbett discussed extending the DAFS (Direct Access File System) protocol to be used by a parallel file server. This provides excellent support for high performance computing with clustered and parallel clients, with support for fencing, shared key reservations, and locking, and for parallel IO semantics such as asynchronous I/O and completion groups. This extension to DAFS was implemented as a single tier file server with a clever client; the fastest implementation was in user space and opens doors for a DAFS client that understands parallel files.

World-wide Repository for I/O Trace Collection and Analysis Tools

Arnold Jones, Storage Networking Industry Association

Arnold Jones discussed the creation by SNIA of a repository of IO traces, workloads, collection, analysis tools and snapshots, for use by industry and academia, as well as a forum for the discussion of tool and trace problem solving, data collection, and similar issues. Participants will also be able to request traces on physical media. The design of the repository is in place, and they hope to be online by 7/1/2003.

http://www.snia.org/apps/IOTTA_Survey/register.php

The Zettabyte File System

Jeff Bonwick, Sun Microsystems

The Zettabyte file system (ZFS), developed at Sun, is to be released later this year. Bonwick noted that existing file systems have problems: no defense against data corruption, and lot of limits, such as on the number of files, maximum file sizes, and so forth. As a result, existing file systems are "excruciating to manage," between tools like fsck, many configuration files, and managing partitions, volumes, and the like. ZFS hopes to "end the suffering" by offering end to end data integrity, immense (128-bit) capacity, and very simple administration. All operations are copy-on-write transactions, with the on-disk state always valid. All data is checksummed to prevent silent data corruption, with support for self-healing in mirrored or RAID configurations. It also supports pooled storage models to eliminate partition management.

Running NFS over RDMA

Brent Callaghan, Sun Microsystems

Callaghan described NFS as implemented over Remote Direct Memory Access (RDMA). This is useful at 1gb/sec, and will probably be a critical requirement at 10gb/sec. It allows direct data placement (DDP), and has been implemented as a new transport on NFS, in parallel to UDP and TCP. He showed a method of doing DDP with RPC/NFS packets and benchmark numbers: 60Mb-sec peak with NFS/TCP, and 102MB-sec peak with NFS/RDMA. Further, CPU utilization is much lower with RDMA. There is a prototype running on Solaris, over gigabit Ethernet support for Infiniband is in progress.

Using SATF Scheduling in Real-Time Systems

Lars Reuther, Dresden University

Reuter began by pointing out that disk request scheduling is best handled at disk; on the other hand, the OS loses some control, which is undesirable for real-time systems, and onboard queue sizes on disk are small. His work examines request scheduling at driver level, especially using SATF, and asks whether it can be used for QoS guarantees. In doing so, this work measured the time between two requests – including the command overhead, seek and rotational delays, and the time to actually read a sector – with instrumentation on a SCSI device driver to determine the match between the model and measured values.

Benchmarks indicate that the external scheduler can match the performance of the disk internal scheduler on a slower disk – catch up with the internal scheduler; on a faster disk, total effective bandwidth is 12% lower but the faster queue benefits real-time guarantees. This shows that a system can do scheduling in software with QoS guarantees – allowing the tradeoff between queue size for throughput and QoS guarantees.

Using a Vector-Based Approach to Predict Performance of Distributed Storage Systems

Alexandra Federova, Harvard

Federova discussed using a vector-based approach to analyzing the performance of n-tier distributed systems. The problem domain is as follows: in an n-tier system – for example, Web servers to application servers to db servers – the impact of adding servers at a tier to improve a perceived bottleneck at that tier is difficult to test. At the same time, determining the impact in advance through simulation is tough to get right, and either approach takes time. Modelling this is difficult because of the complexity of the problem. Vector based modeling has been used for stand-alone systems, and Federova and her colleagues are looking into whether it can be used on distributed systems. The presentation briefly touched on how vector modeling is used for simpler systems, and some proposed rules for the composition of models for distributed systems.

DUMB STORAGE DEVICES SEEK SMART CLUSTER STORAGE SYSTEM SOFTWARE

Christian Saether, Clustor.com

Saether discussed a mechanism for improving access to shared metadata in a cluster, based on earlier work on VAX clusters. The motivation for this work was the availability of cheap and dense multi-system hardware. It uses WAN protocols for data "right next door," with a new access layer for transactional updating of shared storage. Data objects are mapped for fault tolerance and performance, and write-ahead logging is used when there is no contention for data. The system is implemented at the kernel level, above the disk driver, "like an LVM," below the buffer cache. Modifications are made via transactions with nested redo and undo operations, and using a distributed lock manager to maintain data coherency.

THE STORAGE TRANSPORT PROTOCOL

Pat Shuff, Texas A&M University

Shuff discussed a proposed solution to the problem of how to use excess disk on campus, without central administration, on a variety of OS platforms. Their group estimated that the campus had approximately 200 terabytes of excess storage "lying around" – as compared to 2 terabytes of online storage for students and faculty. Existing mechanisms do not offer the ability to take advantage of excess space on unrelated systems or to find existing redundancy to use as backups.

Existing partial solutions include network backup and rsync, file-system level sharing, and block level sharing, but these all have some combination of cost, management, portability, or scalability issues. As an alternative, Shuff's group is working on a new storage protocol, to be implemented under the vnode layer or under the Windows virtual disk interface, which will act as an intelligent manager to handle variable locations for data. They are working on a protocol for network access accounting for security and automation which should work with existing file systems automatically. *http://people.tamu.edu/~pshuff/*

TESTING FOR DISTRIBUTED FILESYSTEMS

Richard Hedges, Lawrence Livermore National Labs

Hedges started by discussing briefly the need for and scope of very high performance computer clusters at LLNL; one large cluster supports up to approximately 100 TFLOPS, with 100 gigabytes per second IO throughput to a single parallel app. The team at LLNL works with other projects, including the Lustre filesystem – a collaboration between the three big DOE labs and industry (see *http://www.lustre.org/*).

There were several testing methods, including both traditional serial testing with readily available tools such as fsx, iozone, bonnie++, and the posix verification suite, and cluster validation testing.

One set of this testing is done using the IOR code, which was recently rewritten, designed as peak-performance throughput test for supercomputing data patterns. It is used as heavy IO load testing for parallel file systems, and is good for modeling data patterns, acceptance testing, and development activities.

Another tool used is Simul, which is an MPI-coordinated suite of system calls and library function calls, accessing a single file up to thousands of times or thousands of different files. It tests only minimal data transfer but high instantaneous load, and is used as a race-condition finder and for testing massive serialization.

For for information, see *http://www.llnl.gov/icc/lc/siop/*

CLUSTERFILE: A PARALLEL FILE SYSTEM

Florin Isaila, University of Karlsruhe, Germany

Isaila discussed technical issues with different types of parallelism: logical parallelism consists of multiple compute nodes accessing a file system, and physical parallelism consists of striping of data across multiple disks. The main problem this leads to is a poor match between the two types of parallelism. The proposed solution is a shared data representation.

A model was found in the PARADIGM compiler, which has been extended to their system for data representation. This is implemented using the physical partition of files into subfiles, and logical partitioning into views. Directions include implementing collective IO, disk directed IO inside the file system. The system can detect matching physical and logical distributions.

A second area is cooperative caching, with the cooperation of IO nodes and compute nodes' buffer caches as a remote disk driver for Linux. Using this, a node can fetch remote blocks into the local buffer cache. The policy to do so is downloadable and highly flexible. Two possible policies have been implemented so far.

DECENTRALIZED RECOVERY FOR SURVIVABLE STORAGE SYSTEMS

Ted Wong, CMU

Ted Wong discussed research into the problem of putting data objects on a storage server, intending them to be retrievable 5, 10, or 20 years hence with confidence and privacy. The goals of this work are longterm availability and confidentiality; the method is to distribute data with (m,n) threshold sequence sharing techniques. These work by splitting the data into n devices, and the threshold sequence techniques mean that only m pieces must be available to recover the data: up to $n–m$ failures are OK, and to compromise the overall data object, at least m parts must be compromised. Over time servers will fail, and there is a need to be able to recover from failures or compromised servers. The proposed technique is verifiable secret redistribution for threshold shared data; the system would use a witness value to prove possible reconstruction. To do this, the original shares split into sub-shares, and there is a broadcast protocol for share and subshare witnesses. For more information see *http://www.cs.cmu.edu/~tmwong/research/*

Federation of Local File System Data into a Shared-Disk Cluster File System

Anjali Prakash, Johns Hopkins University

Prakash discussed a system for "hassle-free data management" in a cluster file system (IBM Storage Tank). The goal is that it be easy to set up seamlessly integrate existing data, and allow for incremental migration of existing data into the cluster. The specific requirement was to add online access to local file system data to the cluster file system. Whether to migrate that data or not is a management decision.

Federated DAFS: Scalable Cluster-based Direct Access File Servers

Murali Rangarajan, Rutgers

Rangarajan described the design and implementation of a portable user-level DAFS implementation, for use in a federation of DAFS servers using memory-to-memory communication. The DAFS client and server in user space share a virtual interface architecture for communications; DAFS calls are translated to RPC on the server, using Berkeley SEDA. The system is implemented on Linux, FreeBSD, and Solaris. Compared to Harvard kernel-based DAFS, overall performance is close, with slightly more slowdown at with higher file sizes.

Synthetic IO Workload Generation Based on RS Plots

Junkil Ryu, POSTECH Korea

Ryu discussed Synthetic IO workload generation based on RS plots, a mechanism intended to generate a synthetic workload statistically equal to real traces.

Transparent Page Cache Coherence Support for Linux-based Stackable File Systems

Manish Prasad, Stony Brook University

Prasad discussed issues with VFS stacking, giving the example of a user process accessing an encryption file system built over ext2. Because, in Linux, file read and write is purely through page cache, there is a high risk of inconsistency in a stacked VFS environment: for example, reads may occur from an upper level, writes to a lower one. The existing stackable layer was modified to be centralized-cache-manager aware, trying to support native filesystems non-intrusively. This was set up to figure out stack order, detect and intercept calls such as `write()` and `sync()`, and then resolve them by invalidating (rather than updating) stale caches. Some future directions include performance evaluation, extension to work with the dentry and inode caches, and integration with network file systems.

SSM: A Self-Tuning, Self-Protecting, Self-Healing Session State Management Layer

Benjamin Ling

Ling defined a session as a period of interaction between user and application, and state is the temporary data which has to persist during the course of the session. He gave the example of a customer signup for a brokerage account. In a typical installation, this would be a database application stored on a standard file system, such as Netapp filer or similar. To improve performance, in-memory replication could be used, but that adds state to the middle tier, and performance is then coupled with uneven distribution of load after a failure. The SSM exploits properties of session state to separate it from the application servers (stubs) and state servers (bricks, which store state in parallel in memory) using a "write to many, wait for few" technique. There is a windowing mechanism (similar to TCP) for stubs to track bricks, and self-healing to recover from errors. There is a prototype, written in Java, running on the UC Berkeley Millennium cluster.

Decoupled Storage: Free the Replicas!

Andy Huang, Stanford

Huang discussed ongoing work at Stanford intended to reduce the cost of persistent state in Internet services with the goal of making managing state very simple, much like stateless front ends and app servers. This is done using a separate state store for non-transactional data. It uses a hash table API, and uses quorums to simplify recovery and keep data available throughout. Updates are handled by broadcasting a message to the replicas and then waiting for majority reply. On a read, the system accepts the reply with the most recent timestamp, and then writes back the new data to all out-of-date replicates. An initial prototype has been implemented.

StoRM: Storage Resource Management

Sandeep Gopisetty, IBM Almaden

Gopisetty noted that the cost of storage management often exceeds the cost of physical hardware, and that a typical heterogenous environment will have various administrative tools that may or may not be interoperable. His group at IBM is developing an enterprise systems management product, distinct from the existing Tivoli products (SAN viewer and data viewer).

The product is intended to manage the complete storage life cycle in several phases: identifying data storage assets, evaluating data in terms of priorities and storage problems, controlling policy and automation, and predicting usage and growth trends. This uses what they call an "autonomic architecture" which supports self-configuration, optimization, correction, and healing. They are engaged in research on automated provisioning for optimal use of resources, modeling for dependability, reliability, and performance, and have stated a goal of developing a policy-based architecture

Scalability of NFSv4 Next Steps

Dean Hildebrandt, CITI at the University of Michigan

Hildebrandt discussed the scalability of the upcoming NFSv4 protocol, in work funded by ASCI. He noted that NFSv4 is stateful on the file server for open, lock, and delegation operations. Questions related to scalability include how to share an object among multiple NFSv4 servers, or, more generally, how to share state. Their proposal was to implement a division between a state server and a data server, and then to extend the NFSv4 redirection mechanism to handle relocating individual files. The process flow would be for a client to mount onto the state server; open requests would go there, and then read or write is relocated to the data server via load balancing, with state copied to data servers as needed.

Is Parity-Protected RAID Obsolete?

Eran Gabber, AT&T

Gabber noted that disk capacity increases at 80% per year, while access time improves much more slowly, at about 12% per year. With the bottleneck of IO rate, not capacity, parity protected RAID has undesirable performance characteristics – 4 I/Os for every write, 2 if everything is in cache, while mirroring always only uses 2 I/Os. With disks that are so large, why bother with RAID?

This doesn't apply in all cases: for read-mostly data, where there are few writes, there is little write penalty. And for cases where the absolute lowest latency is necessary, the need for high-end disk devices may mean that customers cannot afford to replicate; similarly, where the absolute lowest cost is an issue, customers may also not be able to afford to replicate. Another interesting question is, how does MEMS fit? "But for the common case, watch the trend."

INVITED SESSION: ENTERPRISE STORAGE: THE NEXT DECADE

David Black, EMC; Garth Gibson, Panasas; and Steve Kleiman, Network Appliance

Summarized by Scott Banachowski

David Black started the panel discussion by noting that in the future, people will be the scarce resource in storage systems, because any headway in management scaling is instantly consumed by increased capacity. We must address the problem by changing what must be managed. Black mentioned approaches such as content-addressed storage and fixed-content data. These approaches not only solve some of the performance problems, but also change management problems, by requiring no directories or hierarchical namespaces.

Black outlined the "mystery meat" analogy: identifying data that was stored "in the freezer" for a long time. Grid researchers have started working on the problem of tracking and locating data sets, focusing their attention onto the content of the data rather than where it's stored.

Black reviewed some emerging technologies such as iSCSI and storage bricks, noting that how they will be used is unpredictable, as the innovation of early adopter markets dictates their future use. He concluded his segment by noting that the interesting problem in enterprise storage is no longer performance, but robustness.

Garth Gibson noted that most of the great ideas generated in the '90s still show no value to customers. Now that we live in leaner times, cost-effectiveness is high priority. For the rest of his talk, Gibson described the ideas that he predicts will survive in the following decade.

A trend toward simpler administration will survive, as it leads to cost-effectiveness. The most cost-effective mainframe computer is a cluster, and Gibson believes this is true of storage systems as well, as storage clusters leverage commodity storage and connectivity products. Other notable survivors are NAS and SAN, which are converging as new systems mix and match these storage network infrastructures to improve performance and manageability. Separating the control paths from data and asymmetrically accessing data by exposing parallelism will lead to better performance due to increased data bandwidth, offloaded metadata services, and fewer bottlenecks. Gibson explained that he liked object devices because the device encapsulates metadata, and clients don't need to be trusted when servers do authentication.

Gibson described a direction for enterprise storage that includes changing policy and namespace management. Rule-based policy is the rage, but making it work requires a body of well-understood expertise on using policies correctly, otherwise administrators will be cleaning up the mess left by misbehaving AI algorithms. The direction for namespace is toward search-engine-like interfaces for data access. Gibson concluded that in this decade, enterprise storage must deliver the research of the previous decade in order to cope with increasing scale and bandwidth demands.

Steve Kleiman focused on the problems of the next few years: supporting access to petabytes of data in geographically dispersed locations with thousands of users and nodes, and running diverse applications. Further complicating the systems, data will have different availability and recovery requirements, as well as different access patterns.

Kleiman provided a review of the evolution of enterprise architectures from their beginning as proprietary networks to wide-area large-scale enterprise data infrastructures. Driving this evolution is reduced cost of fast storage links and high volume, reliable disk systems. The problem isn't the technology but the management, especially considering the amount of geographic dispersion between data centers.

In order to increase manageability, Kleiman suggests virtualization of devices, elimination or drastic simplification of existing paradigms, and unification of existing enterprise storage solutions. Only by changing the paradigms and allowing the new technologies to enable new strategies will we solve our main problem of data management.

Jeff Chase offered a quick viewpoint before opening the floor for discussion. He noted that because interfaces for storage systems exist at different levels, people have different views of the meaning of convergence. Chase warned that this is leading us down a road that repeats problems facing cluster computing research in the ’90s. During the discussion period, the speakers mostly reiterated points made in their talks. The liveliest part came when someone asserted that policy-based management is “evil” and “foolish.” The systems are much too complex, with many contradicting rules for different scenarios, so that automated management will lead to fiasco. The panel agreed that policies are difficult to define and will never replace administrators.

SESSION: MEASURING THE TECHNOLOGY

Summary by Nate Edel

Modeling Hard-Disk Power Consumption

John Zedlewski, Sumeet Sobti, Nitin Garg, and Fengzhou Zheng, Princeton University; Arvind Krishnamurthy, Yale University; and Randolph Wang, Princeton University

Sumeet Sobti discussed a disk simulator developed at Princeton which gives an estimate of how much energy the disk is likely to consume, given a disk IO trace and a description of the disk. The motivating application is to determine the impact of file system attributes on energy consumption. Locality, burstiness, and the type and number of requests are all key factors; as such, power consumption will be based on user workload and the file system parameters.

Data layout policies, asynchrony, data layout, and background reorganization are all possible parameters, and in total are a huge design space to explore. The original goal was comprehensiveness, which proved very time consuming – simulator speed was key.

A flaw in many existing models is that disks don’t consume power at a constant rate. For example, the IBM microdrive varies by 20–25% during active periods, and by a factor of 10 from idle to active. As such, a coarse-grained simulator is not adequate.

The team developed a fast and fine-grained disk power simulator, which worked well for the two disks it was tested with – and evaluated it against coarse-grained power models. The architecture of the simulator was in two parts. The simulator itself is based on DiskSim with an added Energy Simulator model. The second part is an automatic parameter extractor, which builds on the existing performance parameters extraction.

The energy simulator calculated total energy, which is in turn composed of of active energy states – seek/rotating/reading/writing – and idle states – low power modes and transitions. These are estimated via DiskSim to gather statistics about disk stages, with seek energy for example, determined by seek distance; rotate/read/write determined by constant use differing per activity. A table is used to approximate the behavior of available low-power modes and the transitions between them. Power values are extracted by the combination of hardware and software, tested on a 30ms basis. This is too coarse-grained to get individual operations, so they are instead spread across longer traces and then statistically determined.

Storage over IP: When Does Hardware Support Help?

Prasenjit Sarkar, Sandeep Uttamchandani, and Kaladhar Voruganti, IBM Almaden Research Center

Prasenjit Sarkar began by distinguishing Storage over IP from conventional SAN systems: although in both cases, storage is a service over the IP network, in conventional SANs servers are attached to storage over a specialized SAN network, while with IP SAN, a combined gigabit IP network is used by both servers and storage systems.

IP SAN implementations are flexible, with three common approaches: a software-only implementation with a generic network adapter (HBA), an adapter which implements a TCP Offload Engine (TOE) which supports the TCP/IP network stack on the network adapter, and an intelligent-HBA approach where both the IP storage protocol and TCP are implemented on the adapter.

The down sides to a software-only approach are the TCP copy overhead, multiple interrupts per data transfer, and high communications overhead; variants such as jumbo-frames and zero-copy TCP can ameliorate these somewhat.

A TOE adapter uses DMA to stream data to the network adapter, which implements TCP/IP; this reduces the overhead on the host and results in one interrupt per data transfer, reducing communications overhead, although the TCP copy overhead remains.

An intelligent HBA approach uses a single DMA to the HBA and removes both the storage protocol (iSCSI) and TCP overheads from the host, with one irq per data transfer, and has the lowest communications overhead.

More than an Interface – SCSI vs. ATA

Dave Anderson, Jim Dykes, and Erik Riedel, Seagate Research

Eric Riedel aimed to dispel myths and confusion about hard drives. He noted that while many consumers and businesses divided up the market by interface between SCSI and ATA, the market segmentation as it was seen by the disk drive industry was quite different. The two main segments he went on to discuss are drives for personal storage (PS), used in desktop systems and low-end servers, and drives for enterprise storage (ES), used in servers, high-end workstations, and drive arrays. He also noted that there are separate market segments for mobile drives, such as those used in notebook computers, and for drives for consumer devices/appliances, but that these would be a topic for future discussion. He also noted the persistent myth that drives are built along one assembly line, tested at the end, and the "bad ones" get ATA controller boards and the better ones get SCSI.

He compared two Seagate drives, a 10k RPM Cheetah vs. a 7200 RPM Barracuda, to show the differences between an enterprise drive and a lower-end model. The Cheetah had a smaller platter (84mm vs. 95mm), a much larger actuator assembly to reduce seek times, and a more rigid case structure for durability and vibration-proofing, which he noted were only three major factors, out of many. He also briefly noted differences from a 15k RPM Cheetah, which has a 65mm platter for still lower mass and shorter seeks, but at the expense of lower capacity.

Seek times are much more aggressive on enterprise drives, with the level of separation increasing; the rate of improvement is low on personal drives, and somewhat quicker on enterprise. Seek time is very sensitive to both the mechanics and signal processing, and thus costly. Sensitivity to external vibration is also a factor; the rotation of one drive can affect neighbors, and while enterprise drives are designed to block those vibrations, personal drives are not. This can have a negative effect on performance.

The talk closed with a comparison of the direct performance impact of certain design choices, comparing two IBM drives; area density and platter size were large on the personal drive. RPM was higher on the enterprise drive, overall resulting in a slight bandwidth advantage to the personal drive. However, an enterprise drive of comparable generation has a higher bandwidth (53MB/s vs 37MB/s), while the changes (more platters, higher RPM) result in a higher cost.

STOP MONITORING
YOUR MONITORS.

THE AGE OF AGENTLESS
IS HERE.

10 DAY FREE TRIAL. Chances are your agent-based monitoring system has become a monitored system — monitored by you. It's a relationship you just don't have time for. It's time to go agentless. SiteScope from Mercury Interactive constantly monitors your systems, alerting you the instant a problem pops up. And because it's agentless, maintenance aggravations are kept to a minimum. Download a full version of SiteScope for a free 10 day trial.

www.sitescope.com/trial

MERCURY INTERACTIVE

©2003 Mercury Interactive Corporation. Mercury Interactive and the Mercury Interactive logo are trademarks or registered trademarks of Mercury Interactive Corporation in the United States and/or select foreign countries.

CONNECT WITH USENIX

MEMBERSHIP AND PUBLICATIONS

USENIX Association
2560 Ninth Street, Suite 215
Berkeley, CA 94710
Phone: 1+ 510 528 8649
FAX: 1+ 510 548 5738
Email: office@usenix.org
login@usenix.org
conference@usenix.org

WEB SITES

http://www.usenix.org

http://www.sage.org

http://sagewire.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 its entirety or in part 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
RIDE ALONG ENCLOSED