conference reports

This issue's reports focus on The USENIX Annual Technical Conference and on the 15th Annual FIRST Conference.

OUR THANKS TO THE SUMMARIZERS:

FOR USENIX ATC '03

William Acosta
Raya Budrevich
Francis Manoj David
Rik Farrow
Shashi Guruprasad
Hai Huang
James Nugent
Manish Prasad
Peter Salus
Benjamin A. Schmit
Wenguang Wang

FOR FAST 2003

Anne Bennett

USENIX Annual Technical Conference
June 9–14, 2003
San Antonio, Texas

AWARDS

The USENIX Lifetime Achievement Award (the Flame) was given to Rick Adams for implementing Serial Line IP (SLIP) and founding UUNET, thereby making the Internet widely accessible. In 1982 Rick ran the first international UUCP email link at the machine seismo (owned by the Center for Seismic Studies in Northern Virginia), which evolved into the first (UUCP-based) UUNET. He maintained "B" News (at one time the most popular Usenet News transport), wrote the first implementation of SLIP (Serial Line IP), and defined the first protocol for running TCP/IP over ordinary serial ports (in particular, dial-up modems). The SLIP protocol was superseded, years later, by PPP, which is still in use. Rick founded a nonprofit telecommunications company, UUNET, to reduce the cost of uucp mail and netnews, particularly for rural sites in America. (UUNET was founded with a $50,000 loan from the USENIX Association, which was subsequently repaid.) UUNET became an official gateway between UUCP mail and Internet email, as well as between North America and Europe. It hosted many related services, such as Internet FTP access for its UUCP clients and the comp.sources.unix archives. Rick spun out a for-profit company, UUNET Technologies, which was the second ISP in the United States. The for-profit company bought the assets of the nonprofit, repaying it with a share of the profits over the years. The nonprofit has spent that money for many UNIX-related charitable causes over the years, such as supporting the Internet Software Consortium. The for-profit ISP became a multi-billion-dollar company and was merged with MFS (Metro Fiber Systems, a wide-area optical-networking company), MCI, and then Worldcom, rising to challenge the largest telecommunications companies in America. He is co-author of *!%@:: A Directory of Electronic Mail Addressing & Networks*, published by O'Reilly Books. He is also co-author of RFC-850, the Standard for Interchange of USENET Messages, which was updated to become RFC 1036 in 1987.

The Software Tools User Group (STUG) award recognizes significant contributions to the community that reflect the spirit and character demonstrated by those who came together in the STUG. Recipients of the annual STUG award conspicuously exhibit a contribution to the reusable code-base available to all and/or the provision of a significant, enabling technology directly to users in a widely available form.

The 2003 award was given to CVS (the Concurrent Versioning System) and its four main authors, Dick Grune, Brian Berliner, Jeff Polk, and Jim Klingmon. Without CVS, it wouldn't be possible for any number of people to work on the same code without interfering with each other. It can be argued that without remote-collaboration tools such as CVS, most of the larger free and open source software that is available today could not have existed. While individuals can produce significant software, collaborative methods are often needed for complex and wide-ranging projects.

Dick Grune: The original author of the CVS shell script, written in July 1986, Dick is also credited with many of the CVS conflict resolution algorithms. He developed the script at the Free University of Amsterdam (Vrije Universiteit), where he teaches principles of programming languages and compiler construction. He was involved in constructing Algol 68 compilers in the 1970s and participated in the Amsterdam Compiler

Kit in the 1980s. He is co-author of three books: *Programming Language Essentials*, *Modern Compiler Design*, and *Parsing Techniques: A Practical Guide*.

Brian Berliner: Coder and designer of the first translation of the CVS scripts to the C language, in April 1989, Brian based his design on the original work done by Dick Grune.

Jeff Polk: Jeff rewrote most of the code of CVS 1.2. He made just about everything dynamic (by using malloc), added a generic hashed list manager, rewrote the modules' database parsing in a compatible but extended way, generalized directory hierarchy recursion for virtually all the commands, generalized the loginfo file to be used for pre-commit checks and commit templates, wrote a new and flexible RCS parser, fixed an uncountable number of bugs, and helped in the design of future CVS features.

Jim Kingdon: While at Cygnus, in 1993 Jim made the first remote CVS, which ran over TCP or rsh or kerberos'd rsh, and eventually over TCP/IP. The remote-CVS protocol enabled real use of CVS by the open source community; before remote CVS, everyone had to log in to a central server, copy their patches there, etc. Some years later, Jim formed Cyclic, a company which offered CVS support and development.

KEYNOTE ADDRESS

Neal Stephenson

Summarized by Peter H. Salus

Neal Stephenson – sci-fi author extraordinaire of *The Big U* (part funny, part silly, but worth it), *Snow Crash*, and *Cryptonomicon*, among others – began his keynote by remarking that Bertrand Russell and Stephen Jay Gould didn't rewrite: they had "no delete key."

He then discussed at length Antonio Damasio's *Descartes' Error* (1994), which concerns "how the brain works." In case you don't recall, Descartes separated mind and body; Damasio – and, by extension, Stephenson – think this wrong. The separation actually dates back to Plato, and Stephenson sees it as the difference between Spock and libido (Kirk or Bones).

Stephenson went on to talk of his production of *The Big U* (1984), his first opus, his creation of "steaming mountains of crap," resulting in a process of cutting and distillation. He analogized this with coding and with "distilling whiskey from beer."

When we execute there is a "foreground process" and a "background process" which goes on "all the time." That background process needs space to do what it does – "which is a mystery." What we have is "a faculty of maintaining the entire stream all the time, while accessing it only bit-by-bit at a time."

Stephenson feels that we should "kick down the Platonic model." He is forcefully against PowerPoint.

"I use a fountain pen," he remarked, and went on to say that he does not use "smileys." He thinks Larry Wall came up with something in Perl because there's "more than one way to do something."

Stephenson returned to Damasio and cited Einstein's "clear images," which are "visual and muscular," though he admitted that he wasn't certain what Einstein meant by "muscular." (I'd guess "vigorous" or "forceful" would be the appropriate gloss: the German is "kraeftig.")

Mathematical entities can be combined in an infinite number of useless forms, Stephenson said, but they are capable of leading us to mathematical truth. "Invention is choice."

Down with Plato or Descartes: "We possess an emotional marking system."

"I'm going to make no thunderous pronouncements," Stephenson pronounced. "Novelists, or coders, or philosophers, or paleontologists don't go on churning out masses of filterable stuff, but succeed by doing it right the first time.

"What we do is a much more physical kind of work and emotional kind of work than is believed. Pressure to just churn out lines of code will not lead to good things in the end."

Yep.

INVITED TALKS

ENGINEERING REUSABLE SOFTWARE LIBRARIES

Kiem-Phong Vo, AT&T Labs – Research

Summarized by Wenguang Wang

Phong Vo has 20 years of experience building general purpose software libraries, which cover a wide range of computing areas such as I/O, memory allocation, container data types, and data transforming. In this talk, he showed how to build efficient, flexible, and portable software libraries using the discipline and method architecture.

Vo first pointed out that traditional standard software libraries such as malloc, stdio, curses, and libc have many problems. For example, malloc fragments memory; many container data types have multiple incompatible interfaces; inconsistent and inadequate interfaces are common; programmers often resort to hacks around problem areas, which cause problems in maintenance, porting, and performance.

Vo then discussed the characteristics of ideal standard libraries. These libraries should have good standard interfaces; address unsatisfied needs; be easy to configure, use, and upgrade; and, most importantly, have decent performance. These libraries should enable applications to tailor algorithms for specific needs and simplify library composition to optimize resource usage. Vo argued that with these ideal libraries in hand, programmers could focus on writing libraries instead of writing programs, since a program consists of libraries plus the application specifics. The productivity of programming should increase due to the maximal library reuse and minimal special code.

USENIX '03 Reception

In traditional software libraries, the handle plus operations model is often used, where the handle represents resources and the operations define resource management functions. Although this model can address immediate needs and is easy to use, it causes the problems discussed above. To address these limitations, Vo presented the discipline and method architecture (D&M), which was used when he developed the Vmalloc, Sfio, Cdt, and Vcodex libraries with other researchers.

Vo used the Vmalloc library as an example to demonstrate the D&M architecture. Vmalloc is a popular memory allocation library used to replace the standard malloc interface. In Vmalloc, a discipline defines the type of memory and the event handling of memory allocation. Typical disciplines include heap and shared memory. Programmers can define their own disciplines to extend the type of memory. A method in Vmalloc defines the memory-allocation policy. Typical methods include Vmpool, which allocates objects of the same size; Vmbest, which allocates objects based on a best-fit policy; and Vmlast, which allocates memory for a complex structure and releases all objects together in constant time.

These methods are predefined in the library and cannot be extended by programmers. They can be selected dynamically by environment variables. For example, after the VMDEBUG environment variable is set to 1, Vmalloc uses the debugging-enabled methods instead of the default-efficient methods for memory allocation, which allows various memory allocation problems to be detected automatically. This feature makes the debugging and the profiling of applications very easy whenever needed.

Vo then used the Vcodex library to show how to design a D&M library. The Vcodex library can transform data using compression/decompression, data differencing, encryption/decryption, etc. Designing a D&M library requires determining resource types and general operations, characterizing resources in a general discipline interface, and capturing algorithms/resource management in methods. In Vcodex, bytestream is identified as a resource type (i.e., discipline). All data-transforming operations (compression/decompression, encryption/decryption, etc.) are different methods since they all change some set of bytes to produce other bytes.

All the D&M libraries discussed in this talk (Sfio, Cdt, Vmalloc, Vcodex) can be downloaded from http://www.research.att.com/sw/tools. The AST OpenSource software collection built on top of some of these libraries is available at http://www.research.att.com/sw/download.

THE CONVERGENCE OF UBIQUITY: THE FUTURE OF WIRELESS NETWORK SECURITY

William A. Arbaugh, University of Maryland at College Park

Summarized by Rik Farrow

Bill Arbaugh, who often works in the wireless arena, started off his talk with a joke (I wish I would remember to do that). He showed slides of Free Software, Free Willy, Free Kevin, Free Martha, Free Wireless, and even Free Beer. But what Arbaugh really wanted to talk about was the past, present, and future of wireless security.

Hotspots are a part of the present of wireless networks. You can find maps of wireless networks, collected by war driving, for most large American cities, and some of these networks are open, requiring no authorization or encryption key. Arbaugh postulated that your next generation cell phone will work over wireless networks when possible, and fall back on slower, but more ubiquitous, cell phone technology when necessary. As you walk out of a hotspot talking on your cell phone, it will switch transparently from the Internet to the cellular network. Arbaugh quipped that the poor sound quality provided by cell phones has made short breaks in communication and occasional garbling expected in phone calls, and that will make Internet use more acceptable.

The ghost of wireless security past is WEP, or Wired Equivalent Privacy. Arbaugh described the author of WEP as being in hiding, having created a cryptographic protocol that failed in every imaginable way.

The present of wireless security is Wi-Fi Protected Access (WPA). WPA bolts on over existing hardware and solves some of the problems inherent in WEP. For example, keys will change with each packet sent, and packet integrity will actually work, due to a different set of algorithms (TKIP). Stronger access control (unlike the MAC addresses used by WEP) is included. WPA2 will follow, require hardware changes, and support AES.

Alas, WPA does not solve the current denial-of-service issues. Arbaugh demonstrated this by sending management packets (which are never encrypted or authenticated), which knocked people off the room's wireless channels.

Arbaugh's future includes smaller devices, with the transparent transitioning between hotspots and cellular mentioned earlier, IPv6 addresses and routing, and always-on connections. All devices will need personal firewalls in addition to anti-virus – not that that will protect users from the management back doors that already exist in many cell phones today.

Obviously, we are in for a lot of surprises, and interesting work, in the future of wireless.

Intellectual Property in an Age of Commerce: Core Issues in the SCO / Linux IP Suit

Chris DiBona, Damage Studios

Summarized by James Nugent

Chris DiBona gave a highly informal discussion-format talk on the SCO v. IBM suit and what it may mean. Jon Hall and Don Marti (*Linux Journal*) also participated in a semi-panel format.

First, a general discussion of the issue is in order. On March 6, 2003, SCO filed a suit alleging that IBM had stolen some code from them and placed it in the Linux kernel; they asked for $1 billion [now $3 billion – ed.] in damages. This was supposedly done as part of Project Monterey , a joint effort involving SCO, IBM, and other companies to produce an enterprise UNIX OS for the IA-64. SCO has indicated that they may go after large-scale users of Linux and mailed letters to that effect to 1500 companies. Even if IBM is exonerated, it is not clear that this would prevent SCO from bringing suits against other companies. This is a problem for small companies, which lack the money for a well organized legal defense.

Several legal issues were brought up. One of them, an obscure legal ruling from the late 1800s involving the sale of a pregnant cow when neither the seller nor the buyer knew this fact, may be used to allow SCO to continue, despite their releasing Linux code under the GPL.

SCO has not yet divulged in June where the proprietary code is. A panel of people from the open source community is forming to look at the code under a very restrictive NDA, thus making the panel of questionable value.

Jon Hall brought up the point that a comparison of the SCO and Linux code is insufficient, since code could have been inherited from a multitude of other places, such as BSD.

The impact of this suit on the future is unclear, although it could affect the willingness of companies to share code.

How to Build an Insecure System Out of Perfectly Good Cryptography

Radia Perlman, Sun Microsystems Laboratories

Summarized by Rik Farrow

Radia Perlman always seems like she is having a great time, and this talk was no exception. She selected cryptography bloopers from a much larger collection than she could present during one talk. She began with an email system based on a one-time pad. One-time pads are the strongest, most secure way of encrypting data, provided the pads themselves are kept secure and are used correctly. In the example she described, the email system used the pads with the XOR operation two times too many, making it possible to extract the pads from each message (and decrypt the messages as well).

One of Perlman's personal pet peeves is screensavers. When her screen goes blank, she immediately assumes that her screensaver has kicked in, and types her password. Of course, while this might be the case, a power-saving feature might have just blanked the screen, so if you see a weird set of characters in an email from her, it just might have been a password that she just had to change.

Perlman lambasted systems such as one where whenever you change your password, it gets emailed back to you. Or an SSL-protected online site that emails you back a receipt that includes your credit card information.

Perlman also contrasted secret and public keys. One big reason for Kerberos, quipped Perlman, was the cost of public key licenses. Kerberos requires a directory that must always be available and securely hold everyone's secrets. Public keys no longer have the onerous license fee but require Public Key Infrastructures instead.

She described the four types of PKIs: monopoly, oligarchy, anarchy, and bottom-up. In monopoly, everyone pays Verisign to play. In oligarchy, 80 or more vendors all have self-signed keys in all browsers, with no mechanism for revocation. In anarchy, we use the PGP model, which scales poorly. Bottom-up appeared the most reasonable, in which each organization has its own certification authority (CA), and organizations sign the certificates for the CAs they need to work with.

Perlman went on to rant about the craziness in the SSL certificate scheme (use of X.509), the "300 content-free pages of ISAKMP framework" for IPSec, and her top-ten favorite list of people and things that get in the way of better security. During the question and answer period, a manager begged her to lobby for better security, but Perlman stated that the world just doesn't care enough.

Alternative Top-Level Domains (aka The Game of the Name)

Steve Hotz, New.net

Summarized by Shashi Guruprasad

Steve Hotz, the CTO of New.net, gave one of the most controversial talks of this year's USENIX ATC. This was evident from the audience's hostility and bitter reactions, including a spate of profanities hurled at New.net and a few even directed at the speaker himself. New.net is a domain name registrar offering consumers a large number of new and alternative top-level domains (TLDs) – e.g., .family, .kids, .shop – that are not ratified by the Internet Corporation for Assigned Names and Numbers, or ICANN, a technical coordination body composed of a broad coalition of the Internet's business, technical, academic, and user communities. Among other things, ICANN is responsible for assignment of domain names.

New.net's position is that additional TLDs provide more choices and richer naming schemes and that a market for them exists. Currently, there is no real Internet directory and nothing to fill the gap between whois and search engines. An artificial scarcity exists and has not been mitigated by ICANN. The speaker repeatedly stressed that this is not the most important problem that needs solving in the Internet but just a place where a market exists. Some of the issues in new TLDs were: how many, Internet stability, trademarks, who will control them, and, finally, where does the money ($1 billion per year) go? New.net's perspective is that no single organization should dictate policy but that these issues should be decided by the market.

The approach that New.net has taken does not involve configuration pains for ISPs or individuals and is backward compatible with the existing system. New.net domains can be enabled either (1) via recursives that rely on US government root servers but augment with New.net domains, or (2) through user machines that have the New.net client plug-in (which still relies on the US government root servers). New.net currently offers 88 new domains. They currently have 150 million customers in total. They have tied up with five out of the top seven US-based ISPs and with many worldwide. The speaker also mentioned that their efforts to talk to ICANN and other organizations have not succeeded. They also avoid conflicting TLDs with ICANN or country-code TLDs. However, it remains to be seen if ICANN will break the Internet by launching conflicting TLDs. New.net also plans to partner with other alternative TLD providers that may bid on TLDs with ICANN, such as .kids or .golf. Some of the big companies such as Microsoft, AOL, AT&T, Verisign, and Nokia are potential new TLD providers. However, New.net is not really making an effort in this direction. The reality of "breaking the root" will occur only if multiple non-cooperating businesses promote competing namespaces. The last point was that New.net has been operational for the past two years, yet the Internet is not broken.

During the Q&A, one person expressed the wish that New.net would fail. Another mentioned that New.net has broken the fundamental principle of a single consistent naming system wherever you go. Someone else said he believed that things would break only if there was significant adoption. Another person asked how it matters whether there is a .usenix or usenix.org. The answer was that it is not a necessity but a matter of choice or desire that some people would like to have. An important counterpoint that came up was how to handle Uniform Domain-Name Dispute-Resolution Policy (UDRP) when numerous TLDs exist. For example, how would they deal with someone wanting to open up a .boycott or .sucks domain with organization names in it for people to criticize companies. It would also be harder to enforce anti-cybersquatting laws. To the speaker's position that we need a consortium rather than one organization, one person responded that ICANN was the consortium. Lastly, it was pointed out that New.net names will be invalid if DNS security extensions are implemented.

Modeling the Internet

Harry DeLano and Peter H. Salus, Matrix NetSystems

Summarized by William Acosta

As the Internet continues to grow, it becomes harder to represent the network graphically using a geographical model. Such representations of the network lack the ability to visually express such details as the location of the "big pipes," peering relations, and routing information. To illustrate the increasing complexity of the visual models of the Internet, the speakers presented a history of "maps" of the Internet, starting with the earliest known maps of ARPANET from 1969 and continuing with the geographical maps through the 1970s and 1980s that included the first satellite links to both Hawaii and Europe and the integration of multiple networks like Usenet to form what is now know as the Internet.

One of the fundamental questions that modeling attempts to answer is, how does the Internet behave? This question can be broken down into several components: What are the nodes and their connections? How can the Internet be visualized? How can its health be monitored? How can we detect and respond to events? Some of the techniques used for measurement analysis include the traceroute program, BGP information, and DNS data. Similarly, tools such as ping and the Internet Weather Report [now discontinued - ed.] are used for monitoring the "health" of the Internet. However, these tools are not perfect. As an example, the speakers noted that traceroute does not accurately reflect the existence of multiple parallel paths to some destination.

The speakers discussed several projects (from Lumeta, CAIDA, MIDS) which analyze and track certain sets of Internet information. In particular, the speakers showed the Internet averages of latency, packet loss, and reachability observed during events like the September 11, 2001, terrorist attacks and the April Fools' Virus, to demonstrate how these tools tracked the effect these events had on the Internet. Additionally, the speakers discussed tools, such as Graphviz from AT&T Labs and Walrus from CAIDA, that help visualize the topology of the Internet.

Currently, data collection is subjective and performed in an ad hoc manner. Additionally, the data sets obtained lack context and may be incomplete. As an example, it was noted that paths from traceroute and BGP data contain inconsistencies. The speakers suggested that data collection needs to be more objective and that the interpretation of the results deserves more scrutiny. To this end, the speakers suggested the formation of an international body patterned after the Centers for Disease Control that would both monitor the health of the Internet and be able to respond to events such as disasters and virus outbreaks.

During the Q&A session, John Quarterman asked what are the fixed points (analogous to cities on traditional road maps) of Internet maps? Suggestions from attendees included using Internet exchanges and the root nameservers. Someone pointed out that maps of the physical world are backed by a physical reality that is less mutable than the Internet. This prompted the question of whether we need better mapping science. Other issues raised during the Q&A included a discussion on the scalability of current measurement tools and the limitation on probing large numbers of hosts frequently.

URLs

CAIDA: *http://www.caida.org/*
Graphviz: *http://www.research.att.com/sw/tools/graphviz/*
Lumeta: *http://www.lumeta.com/*
MIDS: *http://average.matrix.net/*
Rocketfuel: *http://www.cs.washington.edu/research/networking/rocketfuel/*

Nanotechnology: As Hardware Becomes Software

J. Storrs Hall, Institute for Molecular Manufacturing

Summarized by Francis Manoj David

What is nanotechnology? It is the construction of self-replicating machines with atomic precision. Molecular manufacturing is actually a better term to describe this process. The manufacturing process is not straightforward. Because of their size it is not possible to just pick atoms and place them together. A molecule can be used as a handle to move the atom to the desired site. A bond-forming chemical reaction is used to fuse the atom to the rest of the structure.

Self-replication is a key requirement for nanomachines. This results in a sophisticated product with exponential growth in capital because of the 100% reinvestment. A parts assembly robot is used to build other robots. Just like a car factory, pipelining and convergent assembly lines are used to speed up the process.

Josh went on to illustrate applications of this technology. Nanomachines can be built to simulate molecules. One such design is "utility fog." These machines can "hold hands" with each other to form larger objects. By controlling the direction of the "hand-holding," one can create solids, liquids, or gases. Control of such machines requires nanocomputers. Nanocomputers, however, cannot be programmed using the usual techniques. The energy required to erase a bit, though small, is very significant at such small scales. Thus, these nanocomputers need to be programmed using reversible programming techniques that avoid erasing bits.

Flying cars are now possible. Airflow can be controlled over the skin of the car. Negative drag gives all the required thrust. Thus, the car can be sleek and elegant. Instant houses? At bacterial replication speeds, nanorobots can turn a pile of dirt into a house. However, there exists the problem of runaway nanorobots. There are several means to control these nanorobots. Removal of fuel can stop replication. Also, a fixed supply of replication unit kernels can control the replication.

Other interesting applications include thin skins that can insulate and protect humans. These skins can be so thin that they wrap around a single strand of hair, providing air management and sweat management! Respirocytes are yet another design for small nanomachines that function like red blood cells. They can store and release oxygen in the bloodstream as and when necessary. Space travel can also be made cheaper by the construction of an extremely tall platform using artificial diamonds. Thus nanotechnology has tremendous potential for the future of humankind.

URLs: *http://www.imm.org/Parts/*
http://discuss.foresight.org/~josh/
http://www.losthighways.org/radebaugh.html
http://www.moller.com/

TECHNICAL SESSIONS – GENERAL TRACK

ADMINISTRATION MAGIC

Summarized by Francis Manoj David

UNDO FOR OPERATORS: BUILDING AN UNDOABLE EMAIL STORE

Aaron B. Brown and David A. Patterson, University of California at Berkeley

This won the General Track Best Paper award. Human error is the primary barrier to building dependable systems. A small mistake such as misconfiguring an email virus scanner can result in lots of lost email. This damage could be prevented if operators were able to undo their mistakes. This paper presents an implementation of undo capabilities for an email distribution system.

Three Rs – rewind, repair, and replay – are necessary to achieve such capabilities. Rewind brings back a system to a time in the past, repair fixes the mistake, and replay ensures that new information is not lost. Paradoxes can arise during a replay, though. This is because the system might make changes to message bodies or restore missing email to a mailbox. To explain this to the user, explanatory emails are sent.

The architecture for the undoable email store consists of rewindable storage and a proxy that intercepts user events. All user events are encoded as "verbs" which are logged. An undo manager is used to control all undo operations. Studies with a Java implementation show that the time and space overheads are not significant. Responding to a question at the end of the presentation, Aaron also clarified that the explanatory messages themselves are not treated as user events and hence no verbs are generated for them.

ROLE CLASSIFICATION OF HOSTS WITHIN ENTERPRISE NETWORKS BASED ON CONNECTION PATTERNS

Godfrey Tan, John Guttag, and Frans Kaashoek, MIT; Massimiliano Poletto, Mazu Networks

This paper presents methods to automatically group hosts in a network based on their communication patterns. Grouping of hosts can simplify network management and can also detect anomalous conditions. For example, a developer's machine behaving like a manager's machine would be interesting information to an administrator.

The proposed scheme uses probe hardware to sniff all network traffic. A central aggregator then collects the information from all the probes. It is responsible for maintaining a "neighbor relationship" between hosts that communicate regularly. Groups are then identified based on host similarities. Similarity is defined as the number of common neighbors. Group formation is treated as a graph theory problem. Each node in the graph is a host and each edge with weight *e* represents the fact that there are *e* common neighbors between the hosts. For a given value of similarity, say *k*, a *k*-neighborhood graph is constructed with edges of weight *k*. Bi-connected components in this *k*-neighborhood graph are considered separate groups. Special cases are used when the graph is a tree. Groups can also be merged into a larger group based on group similarities. For more details on this, please refer to the paper.

The effectiveness of this scheme has been studied by comparing the groups generated automatically with groups of hosts determined by system administrators. Results of this study show that the groups extracted automatically closely match the desired groups. This scheme shows that automatic role classification of hosts based on their communication patterns is feasible. Mazu Network's PowerSecure software now incorporates refined versions of most of the algorithms described in this paper.

A COOPERATIVE INTERNET BACKUP SCHEME

Mark Lillibridge, Hewlett-Packard Labs; Sameh Elnikety, Rice University; Andrew Birrell, Mike Burrows, and Michael Isard, Microsoft Research

Backup service providers on the Internet are costly. This paper describes a scheme by which individual computers on the Internet can cooperate to back up each other's data. Each computer uses a set of partners to store its backup data. In return, it holds part of each partner's backup data. By making redundant copies of the backup on multiple computers, a high level of reliability is obtained. Thus, inexpensive backups can be obtained.

The backup scheme depends upon cooperation between hosts. There are always bound to be hosts that refuse to cooperate. In its basic form, this scheme has several pitfalls. Hosts may promise to hold data and not keep their word. This is discouraged by several mechanisms, including periodic challenges to ensure that partners are cooperating and a novel method called "disk-space wasting" designed to make cheating unprofitable. Attacks aimed at disrupting the service are also possible in the basic scheme. The paper outlines some solutions to prevent these attacks as well.

Results from an initial prototype show that these techniques are feasible as far as performance is concerned. The costs involved are also quite low compared to other Internet backup options.

POWER

Summarized by Hai Huang

Currentcy: A Unifying Abstraction for Expressing Energy Management Policies

Heng Zeng, Carla S. Ellis, Alvin R. Lebeck, and Amin Vahdat, Duke University

A system-level energy management technique is discussed. Energy is classified as yet another resource that can be managed by the operating system. An energy quota is allocated to each process periodically, and when each process accesses an energy-consuming hardware device (e.g., disk, CPU, network interface), it is charged the amount of energy that was spent accessing the device. When all its energy quota is spent, it cannot execute further until the operating system replenishes its quota. The authors were able to show that the system is able to achieve desired runtime, if it is possible, with a high success rate. By managing the energy quota of each application, it is easy to suppress the execution of the less important tasks, while giving more energy to the more important ones to keep the system running longer when the energy supply is low.

Design and Implementation of Power-Aware Virtual Memory

Hai Huang, Padmanabhan Pillai, and Kang G. Shin, University of Michigan

A novel technique to reduce the power dissipation in the DRAM was presented. As workloads become more data-centric, more energy is spent to sustain the ever-growing DRAM in systems. The intuitive basis of current power-management technology is that processes execute one at a time, and when each is executing, it only uses a small fraction of the total system memory. By figuring out the memory nodes each process uses, energy can be saved by putting unused memory nodes into a low energy state. Proactively allocating physical pages to minimize the number of memory nodes each process uses creates opportunities to power off a large percentage of nodes in the system, thereby saving a significant amount of energy. The authors introduced other techniques – library aggregation and page migration – to achieve energy savings. These techniques are compatible with various DRAM architectures, including SDR, DDR, and RDRAM.

GET VIRTUAL

Summarized by Hai Huang

Operating System Support for Virtual Machines

Samuel T. King, George W. Dunlap, and Peter M. Chen, University of Michigan

The authors describe several techniques to reduce the virtualization overhead associated with type II virtual-machine monitors (VMM) so they can achieve performance similar to type I VMMs. In their experiments, they used UMLinux as the guest operating system. One of the techniques they use to improve performance is to move the VMM from the user-space to the kernel, so they can avoid high-overhead ptrace calls and reduce the number of context switches when guest system calls/signals are invoked. Other techniques include the use of segmentation registers to reduce overhead of guest user-to-system and guest system-to-user boundary crossing, and the use of multiple address spaces for the virtual machine to reduce the guest user-to-user switching overhead. The performance of their system was comparable to that achieved by VMware Workstation.

A Multi-User Virtual Machine

Grzegorz Czajkowski and Laurent Daynès, Sun Microsystems; Ben Titzer, Purdue University

This paper describes the implementation of adding multi-user capability in a multi-tasking virtual machine (MVM). A MVM allows a user to execute multiple processes in the same Java Virtual Machine (JVM), and by sharing resources among these processes, it reduces resource consumption and improves overall performance. Multi-user MVM takes this a step further and allows multiple users to safely execute different processes on the same MVM. As if executing in a traditional OS, processes, files, and other resources belonging to the same user will not be illegally accessed by another user. MVM-2 extends MVM by having a separate instance of Jlogin per user session to manage user identity, environment, and other access control information. It was shown that for a typical workload, MVM-2 only incurs a small performance overhead over MVM, while providing a safe multi-user environment.

NEEDLES AND HAYSTACKS

Summarized by Wenguang Wang

A Logic File System

Yoann Padioleau and Olivier Ridoux, IRISA / University of Rennes

Organizing information using the hierarchical paradigm, as is done by traditional file systems, can provide navigation functions but is rigid in that an object can only be reached via one path. The Boolean query paradigm used by search engines provides flexible keyword-based search capability but lacks a navigation mechanism. In this talk, Yoann Padioleau presented a logic file system (LISFS) based on the Boolean query paradigm but extended to provide navigation. Each file in LISFS is associated with a conjunction of properties of interest. The directories are used to represent properties. One can navigate in LISFS by giving the desired or undesired properties as directory names.

A prototype of LISFS has been implemented by the authors. The data structure of the current implementation is optimized for searching and navigation. It is somewhat slow for file and property creation. The performance experiments of the prototype showed efficient navigation execution but 4 to 34 times longer creation time than ext2. The disk space overhead is 2–5KB per file, given 50 properties. A prototype of LISFS and more information on this project can be downloaded at *http://www.irisa.fr/LIS*.

Application-Specific Delta-Encoding via Resemblance Detection

Fred Douglis and Arun Iyengar, IBM T.J. Watson Research Center

Delta-encoding is a data-compressing method that represents an object using its difference relative to a similar object. In this talk, Fred Douglis showed how to use delta-encoding to compress a set of files without specific knowledge of these files in advance. Unlike previous approaches, the resemblance between files is detected dynamically.

Ted Hayelka, Jim Larson, and Bart Massey enjoying the USENIX Reception

Douglis first explained how to detect resemblance between objects. The Rabin fingerprints are used to compute hashes of overlapping sequences of bytes in a file. A subset of these fingerprints (i.e., features) are used to represent the file. Files sharing many features would, hopefully, have similar contents.

After two objects are detected as similar, delta-encoding is applied to compress the objects. The authors evaluated a number of parameters of this approach. They found that delta-encoding using Rabin fingerprints can improve on simple compression by up to a factor of two, depending on workload. A small fraction of objects can potentially account for a large portion of the space and bandwidth savings. More importantly, when multiple files match the same number of features, any file can be used as a good base for computing delta.

Opportunistic Use of Content Addressable Storage for Distributed File Systems

Niraj Tolia, Mahadev Satyanarayanan, Carnegie Mellon University and Intel Research Pittsburgh; Michael Kozuch, Brad Karp, Intel Research Pittsburgh; Thomas Bressoud, Denison University and Intel Research Pittsburgh; Adrian Perrig, Carnegie Mellon University

A distributed file system on WAN is often slow. Niraj Tolia presented their work on building a distributed file system, called CASPER, which exploits the readily available Content Addressable Storage (CAS) to cache the objects in a remote server so that the read traffic through WAN can be reduced.

File recipes, which are the file content hashes that describe the data blocks composing the file, are computed for each file and stored in the server. When a client wants to read a large file, it first fetches the recipes of this file from the server. The client then uses these recipes as keys to search the file blocks in CAS. If a block is found, the client retrieves it from CAS; otherwise, the client reads it from the server. The client finally reconstitutes the full file content using these blocks. The experimental results of CASPER showed that when client-server bandwidth is low, CASPER achieves a significant reduction of runtime when reading large files from the server, given that a significant portion of the file blocks can be found in the CAS.

CHANGE IS CONSTANT

Summarized by Shashi Guruprasad

System Support for Online Reconfiguration

Craig A.N. Soules, Gregory R. Ganger, Carnegie Mellon University; Jonathan Appavoo, Michael Stumm, and Kevin Hui, University of Toronto; Robert W. Wisniewski, Dilma Da Silva, Orran Krieger, Marc Auslander, Michal Ostrowski, Bryan Rosenburg, and Jimi Xenidis, IBM T.J. Watson Research Center

Craig Soules spoke about their work to extend and replace active OS components to perform an online reconfiguration. OSes are complex pieces of software that are required to work under a variety of hardware resources and usage patterns. To overcome this problem of one size fits all, OSes are required to be tuned with the right set of components to work well under many demanding conditions. The need to bring down the system in order to reconfigure it is costly in terms of availability, human time, and lost system state. The goal is to perform such online reconfiguration with minimal overhead. Implementing this in a traditional OS is difficult because of unstructured component boundaries, and thus it is hard to backfit new code. An object-oriented OS such as IBM's K42 is a suitable fit and is used for this purpose.

To support online reconfiguration, component boundaries must be clearly defined, external references to the component must be updated, state transfer mechanisms must be defined between the components being replaced, and components should be quiesced during reconfiguration so as to avoid state corruption. An object translation (indirection) table approach is used to take care of external references. Components handle their own state transfers, since the states of different components vary. An interposition phase interposes a mediator around an existing object that is being reconfigured. The mediator performs the hot-swapping of the object in three phases: forward, block, and transfer. It uses a thread-generation mechanism to detect active calls, and eventually blocks new calls and thus detects when a quiescent state is reached. Once new calls are blocked, the new object is hot-swapped and a state transfer is initiated. After this process, the mediator is detached, the old component is destroyed, and all new calls directly occur on the new component.

The performance overhead was around 500 CPU cycles for the interposition phase and 4000 cycles for the hot-swapping phase on an IBM RS/6000 24 600Mhz Power-PC CPU-based server. Two benchmarks, Postmark and SDET, were used for the evaluation. Several well-known adaptive algorithms performed the online reconfiguration to demonstrate the utility. Some of the open issues include coordinated swapping and recovery from broken components that replace working ones.

Checkpoints of GUI-Based Applications

Victor C. Zandy and Barton P. Miller, University of Wisconsin

Victor Zandy talked about transparently migrating the GUI (X Windows) belonging to unmodified applications between hosts without premeditation, a system that he calls "guievict." GUI replication as well as process migration is also possible using this system. Some of the related work such as VNC or xmove needs premeditation and uses proxies to achieve GUI migration.

The interesting part of the system is the use of program-editing techniques to introduce new code into a running application to achieve the goal. A tool with an architecture-independent API known as Dyninst is used for "hijacking" an application. It's easier to hijack a dynamically linked library than a statically linked one, which requires introducing the dynamic linker code into the application's process space. In order to achieve a high level of transparency, the system tries to automatically determine the original X-server host so as to redirect communication to the new X-server. This is achieved by enumerating the process's socket descriptors that are communicating with a peer port that falls in the well-known X-server port range. This could fail, however, if tunneling is used, and is remedied by trying to determine the right socket by a brute-force approach which involves connecting with every port that the application is already connected to. A user could also explicitly provide this information. Another step in the process is to find a suitable point in the X message stream so as not to corrupt or miss any message. This is performed by walking through the process stack in search of X library stub functions and, if any exist, repeating the operation after a timeout. A limitation of this approach currently is that it does not work with stripped statically linked binaries, for which alternate mechanisms are being worked out. The last step is to retrieve the list of GUI resources that exist on the source X-server and re-create them on the target X-server. X-fonts present a problem because of lack of enough state on the application side to determine the font names. The current solution is to retrieve all fonts and match the font geometry with the ones being used. This hideously slow approach will likely be solved with client-side fonts in the future. Currently, this overhead is mitigated in subsequent requests by caching font data.

URL: *http://www.cs.wisc.edu/~zandy/guievict/*

CUP: Controlled Update Propagation in Peer-to-Peer Networks

Mema Roussopoulos and Mary Baker, Stanford University

Mema Roussopoulos presented work addressing the problem of reducing search query latency in peer-to-peer (P2P) systems. Search queries are a performance bottleneck in both structured (Chord, CAN, Pastry, Tapestry) and unstructured (Gnutella, Freenet) P2P systems. Recent work has used caching of metadata that is returned in response to these queries at intermediate nodes along the path and is referred to as Path Caching with Expiration (PCX). PCX mechanisms are only a partial solution, as there is no maintenance of the caches along the path.

CUP addresses this problem by having asynchronous propagation of metadata updates with independent local node policies rather than global ones. It is independent of the underlying search algorithm and uses incentive-based policies. A node propagates an update only if it has an incentive to do so. For example, a node would be willing to receive updates as long as there are queries for a particular piece of metadata, since the node would have to propagate the query further if it did not maintain the cache locally. Two policies are defined, probabilistic and history-based. Separate logical query and update channels are maintained and queries are coalesced. The number of overlay round-trip hops for queries to come back with answers starting from the originator is used as a metric for the cost of a query. Miss cost is defined as the total number of hops incurred by all misses. As long as the difference in miss costs between PCX and CUP is larger than the overhead of update propagation, CUP recovers its cost.

The Stanford Narses P2P flow-simulator is used to evaluate the algorithm with a variety of cut-off policies, network scales and topologies, outgoing update capacity, and query arrival distributions. Based on their model, CUP recovers its cost by a factor of 2 to 300 under a variety of workloads described in detail in the paper. The various cut-off policies have comparable performance when the query rates are high, while second-chance history-based policy is better with low query rates.

SECURITY MECHANISMS

Summarized by Rik Farrow

The Design of the OpenBSD Cryptographic Framework

Angelos D. Keromytis, Columbia University; Jason L. Wright and Theo de Raadt, OpenBSD Project

Angelos Keromytis described the rationale and mechanisms behind the API created within OpenBSD, and adopted by FreeBSD and NetBSD, for integrating hardware cryptographic devices. The basic concept was to provide an abstraction layer so that programs and kernel routines could ignore the differences between devices, or even the lack of a device. Previous implementations could stall the CPU waiting for a device to respond.

Within the kernel, three functions handle creating, dispatching, and freeing a crypto_session. The dispatch call includes a callback parameter, so that the crypto device can progress asynchronously. Out in user-land, /dev/crypto provides a uniform entry point, controlled via sysctl() calls. The current version of OpenBSD Cryptographic Framework (OCF) is synchronous at the user level.

Another big goal of OCF beyond transparency is performance. The new API does add overhead, but in systems with hardware cryptographic devices it improves performance because the hardware devices can operate in parallel with the CPU. Future work includes smarter load balancing, algorithm chaining within the kernel thread, using the second processor as a cryptographic device in dual-processor systems, and minimizing copying overhead.

NCryptfs: A Secure and Convenient Cryptographic File System

Charles P. Wright, Michael C. Martino, and Erez Zadok, Stony Brook University

Charles Wright presented this paper by explaining motivation (providing protection to data) and describing prior work. He mentioned that Matt Blaze's CFS suffered from network and data copy overhead, and I watched Blaze, just a few rows ahead of me, sit up a bit straighter at that statement. Wright also mentioned TCFS, Microsoft's EFS (which does not work over a network), Cryptfs, a proof of concept by the same authors, BestCrypt (commercial), and StegFS.

Design goals include strong encryption; convenience for users, system administrators, and programmers; and performance. NCryptfs has three sets of players: system administrators, who set up NCryptfs through mounts but do not hold keys; owners, who hold encryption keys; and others, called readers and writers, to whom access is delegated by owners. All keys are stored using a long-term key, and each owner uses a passphrase (currently) to authenticate and access his or her own keys.

Borrowing from Blaze's CFS, NCryptfs also uses an attach to set up encrypted directories for owners. An attach is like a lightweight user-mode mount that, unlike a regular mount, cannot hide files or directories under a mount point. Only data gets encrypted, not the metadata, so any underlying store can be used (local disk, NFS, or CIFS). In an interesting addition to this scheme, owners may delegate their access to individuals and to arbitrary groups, simply by adding authorizations for several individuals. And this delegation can even override permissions found in the underlying file system (when mounted with VFS bypass permission).

NCryptfs outperformed CFS, TCFS, but not BestCrypt in the Am-utils benchmark. In an I/O-intensive benchmark, NCryptfs was fastest. Future work includes better key management, lock-box mode, centralized key servers, and threshold secret serving.

Matt Blaze was on his feet even before Wright finished. Blaze asked if Wright was sure he didn't have the CFS and TCFS performance reversed? Wright said that they fixed some problems with TCFS that had a performance impact. All three crypto file systems were using Blowfish, but had different overhead outside of encryption.

Marc Stavely asked, Why not use underlying file system checks? Wright said that there are no ACLs on lower-level file systems, and that their ACLs provide the key needed to read/write files.

A Binary Rewriting Defense Against Stack-Based Buffer Overflow Attacks

Manish Prasad and Tzi-cker Chiueh, Stony Brook University

Manish Prasad explained that the goal was to instrument a program to defend against stack-based buffer overflow attacks without having access to the source. There are tools available, such as Etch, that can perform binary code translations, but the authors are not aware of any that perform return address defense (RAD) based on binary rewriting.

The authors use both linear instruction disassembly and control flow analysis. Because Intel processors use variable-length instructions, and 248 out of 256 bytes are legitimate starting points for instructions, linear analysis has real problems distinguishing code from data. The authors then follow with a second pass, starting with the code's entry point (as found in the program header), and following all function calls and branches. Even here there are problems, as GUI-based applications tend to use callbacks instead of direct function calls.

RAD replaces the prologue and epilogue of functions with its own instructions, storing the return address during the call and checking it when the function returns. They instrumented several Microsoft applications, and estimate that they missed less than 1% of functions. When used against open source Windows applications, such as gzip and Wget, they obtained similar results, but not with Apache, which includes functions without any absolute addresses (called from tables).

Overhead for an instrumented binary varied 1–4%, and a test of an exploit against an RAD-protected version of winhlp32.exe worked. Prahad ended his last slide with a note: I'm looking for work! One person asked if safe languages would help. Prahad answered that that is the best solution. A person from CERT asked about how RAD handles detected changes. Prahad said that the program exits.

FAST SERVERS

Summarized by Manish Prasad

Kernel Support for Faster Web Proxies

Marcel-Catalin Rosu and Daniela Rosu, IBM T.J. Watson Research Center

The paper addresses the challenges in Web proxy design that arise out of a large number of TCP connections. Handling a large number of network connections causes the proxy server to incur huge overheads due to context switches and user/kernel data copies. The paper presents two mechanisms to ameliorate this overhead, namely, user-level connection tracking and data-stream splicing.

User-level connection tracking allows an application to coordinate its non-blocking I/O operations with significantly fewer system calls. It is implemented by exposing certain elements of a connection's state to user-space over a piece of memory shared between kernel and user-land. The amount of shared memory required is just a small fraction of a typical Web server main memory (1MB, for application with 65K concurrent connections, 16 bytes per connection). The authors implemented a user-level `select()` wrapper called `uselect()`, which reads from the shared memory area whenever possible, and calls `select()` only when the required information is not available in user-space.

The other mechanism proposed is data-stream splicing, which allows forwarding of data between server and client streams from within the kernel, thus reducing a lot of data-copy and context-switching overhead. The two connections (to client and to server) are spliced at the socket level. The novel features in this mechanism are support for request pipelining and persistent connections, content caching decoupled from client aborts, and efficient splicing even for short transfers.

The proposed mechanisms were evaluated on a testbed comprising commodity hardware. Experiments were done using the Squid proxy server and benchmarked using PolyGraph. The speaker presented CPU utilization results for different request rates for a benchmark biased toward small files, which showed best overhead reduction with a combination of uselect and splicing. Splicing achieved very good results for large objects. Also, the proxy hit response times showed substantial reductions using a combination of uselect and splice (10–30%). However, the miss response times reduced only by about 0.5 to 1.3% using the same combination.

Multiprocessor Support for Event-Driven Programs

Nickolai Zeldovich, Stanford University; Alexander Yip, Frank Dabek, Frans Kaashoek, and Robert T. Morris, MIT; David Mazières, New York University

The contribution of this paper is libasync-smp, a library that allows event-driven applications to take advantage of multiprocessors by running code for event handlers in parallel. Typically, event-driven programs are structured as a collection of callback functions which a main loop calls as I/O events occur. However, to execute callbacks concurrently on a multiprocessor requires running multiple copies of the application or fine-grained synchronization.

This paper maps this problem to graph-coloring by allowing the programmer to choose a color for each callback, so that callbacks with the same color are never executed concurrently. Thus, event-driven servers can get more juice out of a multiprocessor machine with minor modifications in code. As proof of concept, they modified the SFS file server (about 90 lines of changed code out of a total of about 12,000), and observed that the modified version on a four-CPU server ran 2.5 times as fast as an unmodified SFS server running on one CPU. Further improvements were observed in task-processing rates with thread-affinity optimizations.

BIG DATA

Summarized by Manish Prasad

Seneca: Remote Mirroring Done Write

Minwen Ji, Alistair Veitch, and John Wilkes, Hewlett-Packard Labs

Minwen Ji presented Seneca, a prototype remote-mirroring storage system. A primary contribution of the paper is a comprehensive taxonomy of design choices for remote mirroring over various axes: fault-coverage (component-failure tolerance), degree of synchronization (divergence), update propagation and acknowledgment, and location of data duplication. They also classify related work in the area, based on their taxonomy.

The second contribution of the paper is the design of a robust asynchronous remote-mirroring protocol that supports write coalescing, asynchronous propagation, and in-order delivery and that provides resilience to many kinds and sequences of failures, low network bandwidth demands, and low (and tunable) data loss. The principal goal of Seneca's design is to make the data available and keep each copy consistent despite disk array failures, Seneca box failures, network failures, and both temporary and permanent site outages. Its levels of fallback divergence modes are time-bounded, log-space-bounded, and unbounded. Updates are propagated either atomically in order, asynchronous batched, or out of order. In Seneca, the data duplication is done in the duplexed SAN appliance.

Seneca's correctness is verified using a simulator which "approximates" the I/O automaton model. One of the key goals of performance evaluation experiments is to answer the question as to how much network bandwidth can be saved by delaying I/Os (asynchrony) and coalescing writes. Results show that substantial reductions in WAN traffic (5–40%) were observed for a batch size of 30 seconds.

EVICTION-BASED CACHE PLACEMENT FOR STORAGE CACHES

Zhifeng Chen and Yuanyuan Zhou, University of Illinois at Urbana-Champaign; Kai Li, Princeton University

The work addresses the problem of cache placement (not replacement) in the context of buffer cache management for lower-level caches (resident on the back-end storage server) in a multi-level storage hierarchy (client-file server-disk array). Cache placement typically follows an access-based policy, which places a block into a cache when this block is accessed, so that the block that resides in the upper-level cache is also contained in the lower-level cache. While this might be essential when upper-level caches are significantly smaller than the lower-level caches, it's not quite so important in a storage cache hierarchy where storage-server and file-server (storage client) caches are comparable in size. An eviction-based strategy places a block in the cache only when it is evicted from an upper level.

The paper uses idle distance as a metric for evaluating the effectiveness of a cache-placement policy. Idle distance is the period of time the block resides in the cache without being accessed. A better cache-placement policy will have lower average idle distances. The paper presents real-world storage-access traces, which reveal that eviction-based strategies show lower average idle distances than access-based policies. The speaker presented simulation results comparing the two placement policies as applied in combination with various cache-replacement strategies (LRU, frequency based, 2Q, MQ), which showed that the eviction-based strategy always performed better.

The work proposes a mechanism to transparently obtain eviction information from client buffer caches, which is nontrivial because a client buffer cache always silently evicts a clean page and only writes out dirty pages to the back-end storage system. The main idea is to maintain a mapping between buffer addresses and the disk block that it houses, so that by intercepting read/write I/O operations, any changes detected in the mapping can be attributed to the block being evicted from the cache. Evaluation on real systems showed a 22% improvement in cache hit ratios and a 20% improvement in the transaction rate.

FAST, SCALABLE DISK IMAGING WITH FRISBEE

Mike Hibler, Leigh Stoller, Jay Lepreau, Robert Ricci, and Chad Barb, University of Utah

Robert Ricci talked about Frisbee, a prototype disk-imaging system from Utah. The paper motivates the use of raw disk imaging over differential update which operates above the file system. Advantages offered by disk imaging include generality across file systems, robustness to file-system corruption, and speed. Disk imaging, however, is low on bandwidth efficiency. Frisbee (derived from "flying disks") incorporates file-system-aware data compression, data segmentation, and a custom application-level reliable multicast protocol.

The talk proceeded with a description of the Emulab environment, where the prototype was built and tested, which comprises a cluster of 168 commodity PCs. Being a time-shared setup (in the true sense of the term), every user has root access to all machines in the cluster. This necessitates returning the cluster nodes to a known state with change of users. Thus a system like Frisbee turns out to be a compelling requirement in an environment like Emulab.

Frisbee was evaluated for scalability with an increasing number of cluster nodes (from 1 to 80). Experiments were also conducted to evaluate scalability in the presence of various percentages of packet loss. Frisbee has been in production use for over 18 months. Speed was reported as the single most attractive feature of Frisbee, being able to write 50GB of data to 80 disks in 34 seconds. On this note Ricci quoted colleague Mike Hibler: "Earlier I could go for lunch while the disk reloads, but now I can't even make it to the bathroom!"

FREENIX TRACK

NETWORK SERVICES

Implementation of a Modern Web Search Engine Cluster

Maxim Lifantsev and Tzi-cker Chiueh, Stony Brook University

Summarized by William Acosta

The authors presented the design and analysis of Yuntis, a prototype for a scalable and extensible Web search engine. The design of Yuntis attempts to provide performance by using an event-driven model with one main thread of execution and "select"-like functionality to avoid the context-switching overhead of a multi-threaded design. Similarly, Yuntis employs custom disk data storage and intra-cluster communication mechanisms.

The overall process of creating the search database involves crawling the network to retrieve documents, which are compressed and stored upon successful retrieval. A page quality score is iteratively computed, keywords are extracted, and an index is created of the extracted phrases. The prototype implementation consists of 12 cluster nodes and a data set of 4 million crawled pages, with statistics and information stored in 121 separate data tables. The content is partitioned over the entire cluster. Future work includes workload balancing of the cluster, scalability improvements, and fault-tolerance mechanisms.

URL: *http://www.ecsl.cs.sunysb.edu/yuntis/*

MAIL

Summarized by Raya Budrevich

ASK: Active Spam Killer

Marco Paganini

Currently, there are three main prevention approaches to the problem of unwanted commercial email: real-time black hole lists, keyword identification, and distributed anti-spam networks. Each of these methods has many drawbacks, however, including high percentages of false negatives. ASK proposes a new solution to the problem by using a challenge-authentication model. It applies authentication to the email without interpreting the content and focuses on validating the senders rather than the message; this approach is highly effective, since spammers use fake addresses to hide their identity. The system consists of three lists: white, ignore, and black. When a message is received into a waiting queue, a confirmation is sent to the sender, and once a confirmation is received the message is moved into the in-box and the email address of the sender is added to the white list; thereafter, a confirmation from that sender will not be required.

There are a few specific issues that the software deals with. In order to catch spammers who impersonate the owner, a "mailkey" can be added to one's own mail. To avoid mail loops, the software keeps the last few emails (the number is variable) in a FIFO queue and checks whether a message occurs more than once. It is also possible to send commands to configure the queue remotely. A limitation of the software is dealing with bounces: The software cannot distinguish between real and spam bounces without accessing the MTA. Another issue that needs to be addressed is that simple challenges can be defeated with a smart auto-responder.

Learning Spam: Simple Techniques for Freely Available Software

Bart Massey, Raya Budrevich, Scott Long, Mick Thomure, Portland State University

Today there are many approaches to the problem of spam, including black/white lists, laws, and automated filtering, which includes feature recognition and classification; these methods can all work together. Feature detection involves extracting information from the header and body of the message. For example, SpamAssassin uses hand-coded features with linear weights combined with threshold values. More sophisticated techniques include information gain and clustering. The simplest format of features is binary, works on all algorithms, and is robust. There are a wide variety of machine-learning techniques. The authors chose techniques that were simple, popular, and educational. The general description of machine learning involves gathering a large number of training instances, measuring statistical properties with respect to a feature set, classifying target instances, and, finally, measuring the accuracy of the classification.

Bart Massey described five different techniques for mail classification, focusing on only a few. The most simple is the minimal Hamming Distance Voting. Here a message is compared against the messages in the corpus to determine its classification; the classification time for this method is huge and it is necessary to have a large variety in the data. The simplest neural net is a single neuron call perceptron, but a more complex neural net works even better. Basically, features are given as input and the network's output provides the classification. There is a need for a representative data set. Tests were run on different corpora, personal messages, and synthetic data. There were enough different characteristics to make a difference. The complex neural network seemed to have the best results, with <1% false positives and 1–3% false negatives. Even humans cannot achieve 0% misclassification rates. Machine learning is a key aspect of the filtering.

NETWORK PROTOCOLS

Summarized by Benjamin A. Schmit

Network Programming for the Rest of Us

Glyph Lefkowitz, Twisted Matrix Labs; Itamar Shtull-Trauring, Zoteca

Itamar Shtull-Trauring started the first talk by comparing network programming to driving a car: An automatic transmission frees the driver from having to make low-level choices, at the possible cost of some performance. Similarly, programmers who want to do networking without caring for peak performance should use a networking toolkit.

The main design goal for the Twisted networking framework was providing a cross-platform solution that still allows access to platform-specific features. It is written in Python, a high-level language that helps beginners avoid common programming errors by providing, for example, automatic boundary checking. The framework is built around an event loop and allows programmers to easily add even complex services such as a Web server to their programs.

In order to show how the Twisted framework can help a programmer speed up development, Conch, an SSH application, was implemented. The implementation contains 5000 lines of code written by a single person, as compared to 64,000 lines by 84 people in the case of OpenSSH. The framework is available for download from *http://www.twistedmatrix.com/* and was released under the LGPL.

In-Place Rsync: File Synchronization for Mobile and Wireless Devices

David Rasch and Randal Burns, Johns Hopkins University

David Rasch identified the space requirement of rsync as a major problem for mobile and wireless devices with little storage capacity. Without this restriction, rsync would be a good tool for synchronization with mobile devices. With the approach presented, the update of a file can be done in place, without the creation of a temporary work copy.

A problem that needed to be solved, however, was that data necessary for later rsync commands should not be overwritten by earlier ones (as he explained, rsync does its work by trying to find blocks from the old file somewhere in the new file). The solution here is the creation of a dependency graph; though it might contain cycles, these are broken up by retransmitting data overwritten by earlier commands, for which two algorithms are implemented.

The benchmarks, which update files typically used on handheld computers, show that there is very little difference in performance as compared to the original rsync implementation. The new algorithm, however, needs more main memory for calculating the dependency graph. This problem can be solved by introducing windowing, which is not yet implemented.

NFS Tricks and Benchmarking Traps

Daniel Ellard and Margo Seltzer, Harvard University

The initial research goal of this paper was to improve the performance of an NFS server by optimizing its read-ahead caching strategy. Daniel Ellard pointed out that 5–10% of the NFS requests arrive at the server in the wrong order, which could hinder it from doing read-ahead properly.

The benchmarks designed to measure performance improvements showed an unusually large amount of variance. This was caused mainly by the fact that modern hard disks have different data transmission rates at different physical areas. New benchmarks that took this into account still showed problems, this time located within the FreeBSD operating system, where a hashtable had been designed too small for today's requirements.

After these problems were solved, the authors improved the NFS server performance by introducing cursors, which made the server able to recognize several concurrent file reads. On a heavily loaded NFS server, these optimizations speed up file requests by a factor of up to 2.4.

BIOS AND VIRTUAL DEVICES

Summarized by Shashi Guruprasad

Flexibility in ROM: A Stackable Open Source BIOS

Adam Agnew, Adam Sulmicki, William Arbaugh, University of Maryland at College Park; Ronald Minnich, Los Alamos National Labs

This won the Best Student Paper award. Adam Agnew presented the first open source PC BIOS that could boot any modern OS. They leveraged earlier work in Linux BIOS, which could only boot Linux. Unlike Linux, some of the OSes rely on services such as video, hard drive, memory sizing, and a PCI table provided by legacy BIOSes. Therefore Linux BIOS could not directly boot these OSes. Legacy BIOSes do a poor job of setting up a PC, requiring modern OSes such as Linux to redo some of the tasks, increasing bootstrap latency. The advantages of using Linux BIOS are numerous: (1) dramatic increase in boot speed – for example, they have achieved a record of a three-second boot – the main bottleneck is the time required to bring the IDE hard drive spin to normal operational speed; (2) small size – an image size of 36KB uncompressed offers scope for more powerful tools to be part of BIOS; (3) open source – it's easily debuggable and royalty free and saves motherboard manufacturers $3–$5 per PC in royalties; (4) remote administration is possible via console support.

The proposed solution employs a combination of Linux BIOS and Bochs x86 emulator, using a custom wrapper functionality known as Adhesive Loader (ADLO), whixh doesn't require the Bochs emulator to be modified. The Bochs emulator has excellent PC BIOS interrupt support, and using an existing mechanism avoided maintaining more software. Together, these provide the missing legacy BIOS support except for the Video BIOS, which is extracted from the Video ROM and packaged along with the above to complete the BIOS. One of the advantages of such a solution is that different components can be stacked, leading to different configurations, of bootloaders, for example, or more sophisticated tools in the BIOS, such as console support.

Possible future work was discussed, which included TCPA, console over other devices such as USB, authenticated booting, virtual machine/virtual machine monitor support in BIOS, transparent encrypted storage, and transparent backup and intrusion detection that cannot be turned off. There was also a demonstration of a quick-booting Windows 2000 OS. During the Q&A, someone asked about menu-based configuration support. The answer was that currently no configuration support is present and it is necessary to re-flash.

URLs:
http://www.missl.cs.umd.edu/
http://Linuxbios.org/
http://bochs.sourceforge.net/

Console over Ethernet

Mike Kistler, Eric van Hensbergen, and Freeman Rawson, IBM Austin Research Laboratory

Eric van Hensbergen talked about console support over an Ethernet device. Traditional forms of console support are through the serial devices that are aggregated through KVM switches in clusters. The main problem with serial console is in supporting denser clusters that have more machines per rack where it is necessary to support only the most essential of the I/O interfaces for reducing both space requirements and the heat dissipation of each machine. An example of such a cluster that researchers in the IBM Austin lab built is known as a superdense server with 200–300 x86 servers in a 42U rack. These servers already support an Ethernet interface and have no room for serial ports on the board.

Two different solutions for console over Ethernet were developed. The first solution, which is a TCP/IP-based Linux console named "etherconsole," uses a kernel interface to the socket library to send and receive console messages instead of performing low-level device operations on the serial device. A major downside of this approach is that console support is possible only after TCP/IP initialization. Problems that occur before this phase will not be reported on the console and, therefore, debugging such problems is difficult. A second solution involves the use of link layer networking, that is, the sending of console messages using raw Ethernet frames with a special Ethernet type and a broadcast Ethernet address. At the receiving end, a program captures these frames and provides the console output. A combined approach was then used for console support: the link-layer approach until DHCP is performed, and the TCP/IP approach afterwards.

Security issues were discussed. The possibility of break-ins and denial-of-service is increased when the console is available over the network. The suggested solutions included a separate private network for console and switch VLAN support. The latter does not address denial-of-service. This was integrated with Linux BIOS into custom firmware at IBM and allowed full system administration and debugging. Some future areas of research in this area were better BIOS integration, serial port emulation, and frame-buffer emulation for OSes that do not support character-based consoles.

Implementing Clonable Network Stacks in the FreeBSD Kernel

Marko Zec, University of Zagreb

Virtual hosting, network simulation, and advanced VPN provisioning require support for multiple protocol stacks on the same physical host. This paper focuses on the design, implementation, and performance of an experimental clonable network stack in the FreeBSD kernel. By separate stacks, the author means independent states – routing tables, interfaces, and firewall rules – rather than independent protocol code. In other words, this work does not support the creation of a separate protocol stack that is not already supported by the OS. Virtualizing a network stack for the above applications is a more lightweight alternative to a traditional virtual machine.

The key design goals were: API/ABI (Application Binary Interface) compatibility and low or negligible performance overhead. A naïve approach to implement clonable stacks is to add an array dimension to the kernel networking data structures. Such an approach, however, increases the amount of kernel code modification. Instead, the approach used was to group the kernel data structures that maintain the state of the network stack together and provide access through an additional level of indirection. This grouping is termed a "virtual image" and has an instance of network stack associated with it. Virtual images are organized in a hierarchy similar to the UNIX process hierarchy that helps in binding unmodified programs to different network stacks.

There was also a discussion on the implementation of CPU load and usage accounting/limiting per virtual image, which helps in resource management and avoids receive livelock. The performance degradation is hardly noticeable, around 3.5% lower than an unmodified kernel where the test system had one active network stack among 128 stack instances. In some tests, the performance improved slightly (5.7%) due to better locality of kernel network stack variables which led to higher CPU cache hits.

URL: *http://www.tel.fer.hr/zec/BSD/vimage/index.html*

FILE SYSTEMS

Summarized by Manish Prasad

StarFish: Highly Available Block Storage

Eran Gabber, Jeff Fellin, Michael Flaster, Fengrui Gu, Bruce Hillyer, Wee Teck Ng, Banu Özden, and Elizabeth Shriver, Lucent Technologies, Bell Labs

Michael Flaster presented this FREENIX award paper. The principal contribution was the dissemination of a replicated block storage system to the open source community, built from commodity servers running FreeBSD, connected by standard high-speed IP networking gear. StarFish is not a distributed file system. It assumes a single-owner scenario and thus doesn't deal with issues resulting from multiple writers.

StarFish architecture comprises a commodity server with an appropriate SCSI or FC controller, called host element (HE), which helps the client host access data from a set of storage elements (SEs) that talk to the HE using TCP/IP. An SE forms a single unit of replication. The SEs could be connected to the HE via either a dedicated link or the Internet. Writes to StarFish are considered to be complete on receiving ACKs from all the SEs that form the required quorum.

The presenter projected steadfast reliability as the unique selling point of StarFish and presented in detail how it behaves when things go wrong, like restarting an out-of-date SE or the failure of an HE (manual failover), and also how the HE ensures that a quorum of SEs are in sync when one SE happens to be slower than the other.

Finally, the author presented some interesting experimental results for read and write availability against various quorum sizes and arrived at a quorum size of two and a set of three replicating SEs as a recommended configuration to achieve good read and write availability. He concluded that SEs not in quorum could be connected over the Internet, thus eliminating the need for a dedicated high-bandwidth low-latency link.

Secure and Flexible Global File Sharing

Stefan Miltchev, Jonathan M. Smith, Sotiris Ioannidis, University of Pennsylvania; Vassilis Prevelakis, Drexel University; John Ioannidis, AT&T Labs – Research; Angelos D. Keromytis, Columbia University

Stefan Miltchev presented this work on authentication in network file systems. The paper presents Distributed Credential FileSystem (DisCFS), which uses trust management credentials to "directly authorize actions rather than divide the authorization task into authentication and access control" and "to identify files being stored; users; and conditions under which their file access is allowed." DisCFS is intended to address the weaknesses of existing file access control mechanisms, which are either too coarse-grained (e.g., Web, FTP) or are unsuitable for use across administrative domains (e.g., NFS).

DisCFS uses a direct binding between a public key and a set of authorizations. A user can delegate trust to another, which results in creation of a chain of trust, similar to systems like SPKI. Only a subset of privileges may be delegated to another user, making privilege escalation impossible. DisCFS can be implemented over any existing mechanism of data exchange (NFS, HTTP, FTP) and can leverage existing IPSec infrastructure for secure client-file server communication.

The experimental platform comprised a set of machines with modest hardware running OpenBSD. Measurements using Bonnie micro-benchmark showed that write throughput was comparable to that of NFSv2, although read throughput was observed to be lower than NFS. The Postmark benchmark, representing a heavy workload of small files, was used for versions of DisCFS without any security and with and without credential caching.

The talk concluded with a discussion of future work, the most noteworthy concerning implementation of access control beyond permission bits and avoiding the use of inode numbers as file handles because of security holes created by inode number reuse.

The CryptoGraphic Disk Driver

Roland C. Dowdeswell, NetBSD Project; John Ioannidis, AT&T Labs – Research

Dowdeswell presented the CryptoGraphic Disk Driver (CGD), "a pseudo-device driver that sits below the buffer cache and provides an encrypted view of an underlying raw partition." CGD targets the problems arising from laptops and other portables "growing legs" and vanishing from public places! Here, "protection of data from other concurrent users is not essential, but protection against loss or theft is important."

Due to its positioning in the I/O stack (just above the disk driver), it is completely transparent to file systems. Some of the goals of the work are to maintain good performance while providing adequate security, ease of use, and seamless integration into the OS release. The in-kernel driver supports an ioctl interface to attach CGD to an underlying disk device or partition and configure the required parameters such as encryption algorithm, key length, IV method, etc. They define a modular framework for adding cryptographic algorithms.

The driver was evaluated on DEC Personal Workstation 500a, Pentium 4–based PC, and an IBM Thinkpad 600E. Results were presented for read-and-write throughput for various block sizes, comparing various encryption algorithms – Blowfish, AES, and 3DES – against unencrypted I/O. As expected, encrypted I/O throughput edges closer to raw I/O with the increase in block size. The talk concluded with a discussion of future work, which included addressing the problem of key revocation and leveraging hardware cryptographic accelerators to achieve better performance.

X WINDOW SYSTEM

Summarized by James Nugent

XSTROKE: FULL-SCREEN GESTURE RECOGNITION FOR X

Carl D. Worth, University of Southern California

The goal of Xstroke is to provide full-screen gesture recognition for X Windows. A gesture is recognized and is passed to the system as a keystroke or a series of keystrokes. The focus is on handheld devices, so it is important that the entire screen can be used for recognition, unlike some systems that have a special "gesture area." The size and location of the gesture are also important. Finally, the gesture "draws" on the screen with a transparent color, thus allowing the gesture to be seen as it is made, but not to obscure data beneath.

One difficulty is toggling gesture recognition on and off. Several approaches were discussed, but none was ideal. The Recognizer is a 3x3 grid based on the bounding box of the stroke. A grid has the problem that similar strokes may look different near corners; thus regular expressions were used to recognize the strings of grid numbers that defined a stroke. An additional problem is that if the device is held tilted, strokes will also be tilted. The solution to this is that the common horizontal strokes (backspace and space) were used to reorient the grid when one was recognized. One nice feature is that tilting the device enough to cause incorrect recognition produces garbage characters, alerting the user to correct the problem with backspace. More information and software are available at *http://www.xstroke.org*.

MATCHBOX: WINDOW MANAGEMENT NOT FOR THE DESKTOP

Matthew Allum, OpenedHand Ltd.

Matchbox is an X Window manager designed for small devices that have low memory, no keyboard, and limited CPU power. Matchbox is designed to be small, fast, flexible, and configurable. It has some restrictions specific to its environment: Only a single full-screen app is supported at a time, and applications cannot resize windows or make windows larger than the screen. A toolbar/virtual keyboard is always visible.

Matchbox's focus is on providing window management that is also compliant with standards, most notably the ICCCM standards. Matchbox is themeable, has runtime and compile-time configuration options, and is extensible via plugins. It also has a PDA-style app launcher.

X WINDOW SYSTEM NETWORK PERFORMANCE

Keith Packard and James Gettys, Cambridge Research Laboratory, HP Labs

Several X clients were set up to talk to an X server attached to a passive packet-monitoring tool and then to low-bandwidth X (LBX) and SSH proxies with a NISTNet router. (NISTNet is capable of producing a variety of delay/bandwidth-limited network conditions.) The xplot performance visualization tool was used to understand the raw packet data. The usefulness of this tool is difficult to overemphasize; it allows the network trace to be examined in detail and problem areas to be picked out easily.

The LBX proxy uses application-specific compression to reduce the size of requests. In general, LBX was found to be of very little utility in a modern setting: some of the requests it can compress are obsolete, plus bandwidth for X requests is now a problem only for large image files. SSH's gzip compression, in fact, was found to be superior in almost all cases.

In general, latency dominates bandwidth, and most of the latency comes from synchronous requests. Many of these can be eliminated by batching them, as with internAtom requests (already done by some toolkits). Finally, restructuring applications can reduce the effects of latency. Image files were found to be the dominant reason for bandwidth usage. It was suggested that using original image formats, which are usually compressed, would be helpful.

Client-side fonts were also studied; they had the effect of significantly reducing the round trips and, hence, application startup time. Bandwidth usage is about the same, although compressing the glyphs for transport would reduce this.

EXPERIENCES

Summarized by Raya Budrevich

BUILDING A WIRELESS COMMUNITY NETWORK IN THE NETHERLANDS

Rudi van Drunen, Dirk-Willem van Gulik, Jasper Koolhaas, Huub Schuurmans, and Marten Vijn, Wireless Leiden Foundation

The purpose is to provide free wireless access in historical cities in the Netherlands. The technology uses 802.11b with three available concurrent channels without interference. The network is on ISM band, shared with ham radio, microwave, and vehicle ID. It is an IPv4 network with a private address space, routing by OSPF, with ISC-DHCP and ISC-DNS; the network is transparent to the user and provides no user-level security.

Currently, there is a 20km^2 outdoor coverage for the city of Leiden, 20+ nodes, 300 private users, and three proxies to connect to the Internet; many local schools and libraries are also connected. There is a need to find free locations for the antennas, since there is no funding to pay rental fees. At every site the strength of the other nodes and the interference are measured. Network simulation software is used to determine the construction of all sites.

A node consists of multiple antennas with donated PCs or embedded systems. The systems run FreeBSD because it is stable, single distribution, and tagged release. Industry-standard wireless cards are used.

The wireless network is available freely to all inhabitants of Leiden. The Wireless Leiden Foundation is a nonprofit organization that tries to create strong connections with schools and universities. The network has many uses, such as P2P, gaming, and VPN. On any given day, about 100 users are connected.

Community acceptance, project management, and interference were the main problems facing the project.

OpenCM: Early Experiences and Lessons Learned

Jonathan S. Shapiro, John Vanderburgh, and Jack Lloyd, Johns Hopkins University

OpenCM is a new configuration-management system that uses cryptographic authentication and high integrity. The architecture uses cryptographic naming, and the content of a configuration file is a DAG; every revision is a pointer to the root of the DAG. If we use crypto hashes for the pointers and sign the revision records, we get end-to-end integrity and auditability. SSL is used as the authentication technology. Because of the permissions setup of the system, anonymous access is easy and convenient.

Currently authentication and integrity work well. During the last two years there have been only three breaches. The hazards include OpenSSL bugs yielding an unreliable transport. Boehm-GC flaws yield leaked memory and regular server hangs.

Several decisions, during implementation, that seemed plausible led to errors later on:

1. Using a texty format for debugging – This cost 30% more space and didn't compress, and file systems didn't respect cases. Therefore one should plan ahead for binary format.

2. Server-side integrity checks – Files might be damaged when they reach the server. The server needs to serialize/deserialize, which leads to the server knowing the object schema.

3. Build a simple file-based store first – One file per object was stored, using gzip to compress, but in switching to binary format, it was discovered that 60% of the files were less than 500 bytes.

4. Build an event-driven server – Used non-blocking I/O and an event loop, which simplified the code and the server but caused problems with SSL.

5. Use GC and exception handling – This was well engineered for memory management but has many platform dependencies, leaks memory in large objects, and doesn't work on OpenSSL.

To fix GC and exceptions a new management, GC_SCOPEs, is introduced. Currently, schema issues described in the paper are all fixed in the development branch. There's still a need to modify storage format and replace Boehm-GC with a portable application-specific collection strategy.

Free Software and High-Power Rocketry: The Portland State Aerospace Society

James Perkins, Andrew Greenberg, Jamey Sharp, David Cassard, and Bart Massey, Portland State University

The research group consists of undergraduate and graduate students in science and engineering at Portland State University, people in local industry, and local aerospace enthusiasts. The current model reached 3.6km; the next-generation model will leave the atmosphere. The goal is to put nanosatellites in orbit.

The active guidance computer on the rocket determines the rocket's current position, heading, and course; the computer then steers the rocket to keep it on course. In order to determine this information the computer uses several sensors: GPS, inertial measurement unit (IMU), magnetometers, and pressure and optical sensors.

The launch tower is used to launch the rocket, view rocket status, and send emergency commands to the rocket. The on-board system includes the flight software avionics firmware.

It was difficult to decide which embedded operating system to use. There were several candidates but RedHat's eCos was selected. It has all of the needed criteria: RT, POSIX, free for use, small.

Commercial OEM GPS boards don't work because of their software limitations in determining acceleration velocity and altitude. GPL-licensed firmware for GPS receivers uses eCos. Differential GPS base station and receiver provide precision timing and altitude determinations.

Future work includes creating an enhanced inertial measurement unit, developing next-generation navigation algorithms, investigating loose coupling of IMU and GPS, GPS aiding of the IMU unit, deep coupling of IMU/GPS and other sensors, and adding a steerable hybrid motor.

PRIVILEGE MANAGEMENT

Summarized by Benjamin A. Schmit

POSIX Access Control Lists on Linux

Andreas Gruenbacher, SuSE Linux AG

The POSIX.1 access permission model is not always sufficient, especially when interacting with Windows clients via SAMBA. The abandoned standard POSIX.1e (the last draft, 17, is available to the public) is taken as the basis of most ACL implementations.

In the Linux implementation, which has become an official part of the 2.5 kernel version, access permissions (read, write, and execute) can be granted to the owner, named users, the owning group, named groups, and others. A mask is used to retain backward compatibility for non-ACL-aware applications. Default permissions make it possible to inherit permissions that were set at directory level.

The Linux ACL implementation can be used for ACL-aware network protocols, the most important ones being NFS (only version 4) and CIFS (formerly SMB). Since these two protocols implement ACLs in a different way, a mapping needs to be done here. The Linux implementation does not support granting somebody other than the owner the ability to change permissions.

Privman: A Library for Partitioning Applications

Douglas Kilpatrick, Network Associates Laboratories

In this talk, Douglas Kilpatrick described the Privman privilege management library. Some UNIX applications make use of privileges that normal users can't acquire and, therefore, run with root privileges because UNIX does not yet support relinquishing access privileges. This is no problem as long as the applications do not contain any bugs (which could lead to root privileges for any local user).

Privman is implemented as a user-space library which mirrors some calls to the libc as well as some system calls. A program linked to the library starts by forking into a privileged server and a client that gives up all its access privileges. When the client needs to make a privileged call, it contacts the server on a pipe. The server than decides, based on a policy file, whether the access should be granted.

The main advantage to Privman is that few changes need to be added to the source code. The overhead induced by the wrapper calls can be huge for a single system call, but because these calls occur rarely, the performance of a typical application decreases by only about 5%. Privman has been used to adapt OpenSSH and wu-ftpd for privilege management.

The TrustedBSD MAC Framework: Extensible Kernel Access Control for FreeBSD 5.0

Robert Watson, Wayne Morrison, and Chris Vance, Network Associates Laboratories; Brian Feldman, FreeBSD Project

In his dense talk, Robert Watson explained the implementation of the Mandatory Access Control system within FreeBSD 5.x. The MAC framework is implemented partly within the kernel, partly in user-space. Common applications require more than just the standard UNIX security mechanisms. Operating system support for access control is important; without it, applications would have to cope with several different security mechanisms, the result being mostly an intersection of the individual permissions.

For MAC, the FreeBSD kernel has been extended by additional events. MAC-aware applications register their interest in some events, then get these events, and possibly de-register afterward. Access policies are divided into adaptations of traditional access policies, traditional MAC policies, and new ones like a port of the SELinux FLASK policy to FreeBSD.

The main benefit of MAC is that the source code does not have to be modified to use it. The implementation still lacks support for multi-threaded applications and multi-threaded kernel threads. Also, object labeling within the kernel currently decreases performance, which could be solved by introducing a cache.

KERNEL

Summarized by Francis Manoj David

Using Read-Copy-Update Techniques for System V IPC in the Linux 2.5 Kernel

Andrea Arcangeli, SuSE; Mingming Cao, Paul McKenney, and Dipankar Sarma, IBM

Locking and synchronization are extremely important tools in multi-threaded environments. Atomic increment, the key to locking, is possible on current CPUs. However, newer processors (like the Pentium 4) consume significantly more cycles than older processors during a single atomic increment. This is because the whole pipeline needs to be flushed, and this is an expensive operation. For data that is read-only, paying this penalty is unreasonable.

Read-Copy-Update (RCU) is a technique that "allows lock-free read-only access to data structures that are concurrently modified on SMP systems." To illustrate RCU concepts, let us look at an example. In a linked list, deletion of an element and traversal of the list cannot happen simultaneously. This is because the traversal code might reference a pointer that is freed by the deletion code. The RCU-based solution is to defer the freeing of the pointer until it is certain that no other code is traversing the list. The paper references a couple of more efficient solutions to this problem.

The authors have successfully applied RCU techniques for semaphore data structures in the Linux 2.5 kernel. There is a dynamic array called ipc_ids that needs to be traversed for all semaphore operations. This array is protected by a global lock. This design prevents semaphore operations from proceeding in parallel. Using RCU, the global lock was eliminated and deferred deletion of semaphores was implemented. The total number of new lines of code added to the semaphore implementation was only 151. Micro-benchmarks show very good results. In the future, the authors plan to use RCU to optimize more code in the kernel. The current code is available under the GPL license.

URLs:
http://www.rdrop.com/users/paulmck
http://sourceforge.net/projects/lse

An Implementation of User-Level Restartable Atomic Sequences on the NetBSD Operating System

Gregory McGarry

A Restartable Atomic Sequence (RAS) is a mechanism used to efficiently implement atomic operations on uniprocessor systems. It was designed to provide atomic operations on processors that don't provide them. On processors that do provide them, RAS increases processor performance.

System calls are one option when it comes to performing atomic operations for user threads. This emulates memory-interlocked instructions. However, this comes with a large overhead. RAS is a better solution to this problem. An RAS is basically some code that provides some primitive like test and set or increment a variable (e.g., `count++` in C). In order to guarantee that this code is called atomically, a user thread should not be preempted in the middle of this code. If it does get preempted, then the code needs to be restarted. This restarting would ensure that the operation is performed atomically.

RAS has been implemented for the NetBSD OS. The user thread registers the entry and exit points of the RAS with the kernel. Whenever there is a context switch, the kernel checks to see whether the execution was in the middle of the RAS, and if it was, the RAS is restarted when the thread returns. Thus, atomic operations are provided. Also, one cannot write arbitrary code for RAS and expect things to work correctly. An RAS should have the following properties. It should have a single entry and exit point, should not modify shared data, and should not invoke any functions. This ensures that, on restarting the sequence, the thread is restored to a correct state. Thus, the responsibility for getting things to work correctly is shared between the kernel and the user thread.

The user interface to the system is similar to that provided by mmap. RASes are registered with the kernel. The kernel checks for such sequences in the `cpu_switch` function and restarts sequences if necessary. Performance studies show that this approach is better than the syscall approach. RAS is currently transparently used in the NetBSD pthread library for uniprocessor machines.

Providing a Linux API on the Scalable K42 Kernel

Jonathan Appavoo, University of Toronto; Marc Auslander, Dilma Da Silva, David Edelsohn, Orran Krieger, Michal Ostrowski, Bryan Rosenburg, Robert W. Wisniewski, and Jimi Xenidis, IBM T.J. Watson Research Center

K42 is a new research OS kernel designed to be highly scalable and extensible and written in object-oriented style. It also supports online reconfiguration of kernel services. K42 pushes a lot of usual kernel functionality into user-space, enabling efficient IPC. Also, global data and locks are avoided in order to improve performance.

The motivation for this work was to run all standard Linux programs on K42, thus increasing the number of applications available to those working with K42. This paper describes the challenges that the authors faced in providing the Linux API on K42. Linux processes were supported by mapping them to K42 processes. A K42 process called the `ProcessLinuxServer` maintains these relationships. Fork was implemented by placing most of the code in user-space. The file descriptor table was lazily replicated for performance reasons. For signals, all state was kept in the client. To provide a Linux environment, a new version of glibc, targeted toward K42, was compiled. `Exec` was also implemented in user-space. This involves an unload operation followed by a reload operation. Namespace resolution for files was also implemented in user-space. The entire Linux emulation layer was provided by a modified Linux kernel, with K42 as the target architecture. Currently, the project has a fully functional implementation for 64-bit architectures and is available for download under the LGPL license.

URL: *http://www.research.ibm.com/K42/*