Session Summaries from the 17th Symposium on Operating Systems Principles (SOSP'99)

Editor: Jay Lepreau, University of Utah

Assistant editor: Eric Eide, University of Utah

Contributors:
Rajeev Balasubramonian, Gretta Bartels, Neal Cardwell, Chris Diaz, Bill Dieter, Steve Gribble, Jon Howell, Jay Lorch, Jeanna Neefe Matthews, John Regehr, Neil Stratford, Dave Sullivan, Sai Susarla, Kip Walker

The last SOSP of the second millenium—true no matter how one counts—seemed to be a great success in all respects. Attendance was excellent: 389 total, of which 175 were students, of which 60 had scholarships—out of 90 who applied. Moral to students and their advisors: apply!

The core of the conference remained the strong refereed paper presentations and the rich hallway interaction with a "who's who" in operating systems research. However, three unusual special events enhanced SOSP this year, two of which are summarized in the following pages. Mark Weiser, who recently suffered an untimely death, was honored in a moving special session. His friends and colleagues shared a wide range of reminiscences of this visionary researcher and very special person. In another session, Jerry Saltzer and Butler Lampson contributed two insightful, humorous, and reflective invited talks. In a late-breaking talk that is not further summarized herein, Ed Felten gave an entertaining after-dinner talk on the Microsoft–Dept. of Justice lawsuit. He focused on his role as an expert witness—what the experience was like—and included an "Antitrust 001" tutorial. He used an analogy accessible even to CS researchers: a company with a monopoly on flour is legally constrained in its actions in the sugar market.

This particular SOSP seemed to offer especially good schmoozing, at least to me, and I have heard from a number of people that they found it an especially good conference. In a novel perk, students who served as volunteers were offered "lunch with the big-shot of their choice." The receptions, supported by generous industrial sponsors, were frequent, convenient, and liquid. They were held right amongst the posters during two nights, so for once the 17 posters (summarized below) apparently received their due audience.

SOSP has always aimed for seclusion, to encourage interaction among attendees. The carefully-groomed Kiawah Island venue certainly was secluded—the guards and checkpoints on the road reminded me of my youth in Haiti. But the beach was close, wide, and breezy, occasionally serving as a place to clear the head, play Ultimate, or watch the meteor showers at 1:00 a.m.

The bulk of this report, however, focuses on the papers, the most important part of the conference. John Wilkes, program chair, deserves a huge amount of credit for organizing a successful reviewing process, doing more than his share of reviewing 60 or so papers, and helping to line up the special events on the program. 89 papers were submitted of which we provisionally accepted 19, all of which improved through the now-routine editorial review process called "shepherding." The OS field appears to be unique in the

quantity and quality of the reviews generated for its top conferences, in this case SOSP and OSDI. In this SOSP, second-round papers typically had 14 reviews, often lengthy. The program committee and external referees generated over 800 reviews, totalling over 2MB of text and 280,000 words. Based on this mass of data, the program committee selected four papers to forward to TOCS, recommending them for expedited editorial review. Their brief titles are "Porcupine," "Cellular Disco," "Click," and "Soft Timers." My congratulations to their authors.

We should all be grateful, as I am, to the student scribes who provided the following session summaries. The students were a pleasure to work with and very capable. They provided text that, typically, we edited only very lightly—as can be noted from the non-uniform format and level of detail, for which I bear responsibility. The scribes also gathered each speaker's slides, for regular papers, work-in-progress, and invited talks. The slides should be linked off the SOSP web page by the time you read this in OSR. Other material may also be there, such as pointers to the relevant home page or software. Gretta Bartels' eloquent summaries of the invited talks may have longer versions created; if so they will also be linked to the SOSP page: `http://www.diku.dk/sosp99/` and `http://www.cs.dartmouth.edu/SOSP99/`.

I thank Darrel Anderson of Duke, whose name does not appear on a summary but helped by taking detailed notes that served us in other ways. I am very grateful to Eric Eide of Utah for his great assistance editing and formatting this report. We thank the scholarship and general SOSP sponsors: SIGOPS, Compaq, HP Labs, IBM, Lucent Bell Labs, Mercury Computer Systems, Microsoft Research, and Xerox PARC. Finally, we should all thank David Kotz for his great job as general chair, and all the other organizers who mostly worked behind the scenes. The next SOSP will likely be at Lake Louise in Banff or Sun River Resort in Oregon. See you there!

Contents:

Tribute to Mark Weiser

Summarized by Dave Sullivan, Harvard

As people sat at their tables following dinner on Monday, a special session was held honoring Mark Weiser, a beloved member of the systems research community and program chair of the 1995 SOSP, who passed away from cancer earlier this year.

The session was organized and introduced by Doug Terry, who worked with Mark for many years at Xerox PARC, where Mark served as manager of the Computer Science Lab and later as chief technologist. Doug still remembers the first time that he met Mark. It was at the 1984 USENIX Conference, and Mark was raving about window systems. "He loved to rave," Doug recalled. "He had a passion for technology that I've rarely seen in someone." Doug also mentioned Mark's vision for ubiquitous computing, a term that he coined. Mark passionately pursued that vision, and it had a tremendous impact on many of the papers in recent SOSPs. Doug then brought out "the appropriate attire" for the event, a pair of red suspenders like the ones that were Mark's trademark. In an impromptu development, each of the speakers "wore" the suspenders while sharing stories of Mark and how he profoundly influenced systems research and the members of our community.

Marvin Theimer, who worked with Mark on ubiquitous computing at PARC, described what it was like to work with him. He mentioned Mark's "incredibly infectious enthusiasm and irreverence," and how he always went for the high-risk, high-payoff option. With ubiquitous computing, for example, they did almost everything from scratch, trying not to compromise on anything. While the risks did catch up with them at times, Mark's enthusiasm and risk-taking made it incredibly exciting to work with him; it felt like "working on the future."

M. Satyanarayanan spoke next, recalling the reaction to Mark's 1992 distinguished lecture at CMU on ubiquitous computing; most of the audience was "dumbfounded at the audacity of his vision." To help us appreciate the reaction, Satya reminded us that the Web did not exist at that point, laptops were slow and heavy, and there were no PDAs or wireless networks. Indeed, we still don't have the technology needed to realize Mark's vision. Given this fact, why was Mark so bold and tireless in enunciating his vision? Satya attributed it to Mark's realization that reaching for "a truly tantalizing vision" allows us to make progress that we could not make otherwise. "In this sense," Satya said, "I truly believe that Mark Weiser is in the company of people like one of the founders of our field, Charles Babbage, who enunciated a vision that took a century to realize." He spoke of Mark's courage in putting his reputation on the line in articulating his vision. Mark's legacy to all of us, Satya said, is "to never be afraid to ask the big questions, and to follow your guiding light wherever it takes you."

Maria Ebling spoke of Mark as an energetic researcher who believed in his vision of what the world would be and who loved his work. She recalled his willingness to listen to her ideas and to encourage her. Maria also remembered how Mark "bounced" with excitement after being appointed as program chair of the 1995 SOSP, asking for suggestions about how to make it the best conference it could be, as well as his excitement about the band he played with in his free time. She said that Mark's legacy still influences her work on pervasive computing at IBM.

Mary Baker recalled Mark's willingness to listen to a crazy idea, to find a salvageable part of it, and to give interesting ideas about how to pursue it. He was excellent in discussing both technical ideas and how they affected the people that used them. Mary noted Mark's role as a mentor to students throughout the country. He frequently attended research group retreats at Berkeley, and he would follow up with students by e-mail about their ideas. She shared a story about a memorable bus ride with him to one of the retreats, in which Mark and others put together a call for papers for "ICON '92, the Second International

Conference on Nothing." She shared parts of the hilarious CFP, giving us "a flavor of some of his insanity."

Frans Kaashoek spoke about Mark's work as program chair of the '95 SOSP, and his clear goal of broadening systems research as much as possible. He recalled Mark's personal touch: the members of the program committee got personalized mugs that reflected their roles on the committee. Frans also noted Mark's introduction of a CD version of the proceedings and his commitment to having as much of the relevant source code as possible on the CD so that people could attempt to reproduce the results. Frans concluded by mentioning the impact of Mark's vision on his work and those of others at MIT, including the new Oxygen project that has recently been started there.

Doug Terry then opened the microphone to anyone else who wanted to speak. Eric Brewer mentioned Mark's advice to students at the last Berkeley retreat to always have an "elevator pitch" about your work. David Cheriton recalled speaking with Mark when they would pick up their daughters at youth symphony rehearsals. He mentioned Mark's tremendous pride in his children, and the importance of family in Mark's life. Stefan Savage shared several memories of Mark, including his openness to a crazy (but since almost realized) idea of obtaining energy for mobile computers from people's shoes as they walk, his willingness to put Stefan up at his home during Stefan's internship at Xerox PARC, and Mark's tremendous enjoyment of his band.

Jay Lepreau reiterated Mark's commitment to the reproducibility of results—"that we should act like a science and not just be named one." He recalled a prescient "energy-scheduling" paper that Mark contributed to the first OSDI that relied on a type of hardware that didn't then exist, addressing the important issue of reducing power consumption. Jay also mentioned Mark's firm belief that "anybody could do anything," and how this belief extended to the way that he "looked for the best in people and got it."

Andrew Black recalled the courage of Mark's opinions. Mark not only argued for the reproducibility of results, he brought enough CDs to the 1993 Asheville SOSP for everyone to have a copy of the source code for his paper. Amin Vahdat spoke of Mark's feedback to Brad Chen, a student speaker at the same SOSP, "you have set the standard for this and all future SOSPs," which inspired Amin and his classmates to work to produce papers worthy of such praise. Karin Peterson mentioned his personal impact in recruiting people for PARC, his ability to keep open a wide range of possibilities when discussing a research project, and the personal dimension that Mark brought to the workplace, including his participation at Friday night social outings and his willingness to speak about things that didn't directly involve work. Drew Dean spoke of the way that Mark could enliven the lunchtime conversations at PARC.

Carla Ellis concluded this moving tribute to Mark by announcing that SIGOPS plans to establish the SIGOPS Mark Weiser Award, an annual award for "creativity, innovation and vision in operating systems research" to be given to a researcher less than twenty years into his or her research career. Every year, members of the systems research community will have an opportunity to nominate someone for this award with "real vision and creativity, to capture some of Mark's spirit."

Invited Talks

Summarized by Gretta Bartels, Univ. of Washington

Coping With Complexity

Jerry Saltzer (Massachusetts Institute of Technology)

In the first invited talk, Jerry Saltzer presented his engaging view of how software engineers and system designers should work to cope with complexity in large computer systems. First, Saltzer analyzed the sources of complexity. He explained that systems become unwieldy because their specifications call

for a tremendous laundry list of objectives, but system designers lack principles for achieving so many goals simultaneously.

As a series of object lessons, Saltzer presented a series of engineering failures in increasing cost, from failed upgrades to the New York City traffic light system to the flopped British Stock Exchange project to a dragging project to modernize the U.S. Government's tax collection system. He discussed the causes of failure in each of these systems, pointing to the second system effect, designers trying to tackle too many goals at once, the mythical man-month phenomenon, oversight committee meddling, and more. Saltzer also discussed what he termed the bad news-good news diode: the tendency in large bureaucratic organizations to pass only good news up the management chain, so that the highest levels of the organization are the most deluded.

In the second part of the talk, Saltzer presented a series of principles for coping with the rampant complexity problems plaguing systems development today. He suggested that system designers plan for iteration in the design process. Building iteration in allows bad ideas to fail sooner in the development cycle, making them less costly and easier to weed out. He also pointed out that novelty is sometimes included just for novelty's sake, and that managers must be strong enough to pick and choose their complexity, to say no to strong-willed designers who want their own good ideas included. Finally, Saltzer implored the audience not to allow their own systems to become case studies in future versions of his talk.

In the lively question and answer session that followed the talk, Ken Birman asked why it is that some complex systems do succeed. Saltzer replied that these systems were usually built up in small steps, with each intermediate system stable and functional. Dave Tennenhouse pointed out that virtually all of the examples in the talk were from the commercial, industrial, and governmental sectors. How can academic researchers benefit from the advice in this presentation? Tennenhouse pointed out that the Alto project at PARC and an MIT project whose manager's name he couldn't quite remember (to much laughter) were good examples of complex academic research projects. Saltzer's answer was that academic projects often have different goals, such as the spread of ideas, so a completely stable system may not be necessary, but that otherwise his principles transfer well. Finally, John Wilkes pointed out to the amusement of the audience that since we learn so much more from failures than successes, all graduate students should be forced to work on a large failing project.

Thumbnail slides from this presentation are at:
`http://web.mit.edu/Saltzer/www/publications/Saltzerthumbnails.pdf`

Computer Systems Research: Past and Future

Butler Lampson (Microsoft Research)

In Butler Lampson's invited talk, he treated the SOSP community to his view of computer systems research over the last twenty years, and then presented a charge for the next ten. Lampson started by setting a bit of context for the rest of his comments. He pointed out that the most important context for computer systems research is Moore's Law, and that people often lose sight of the enormity of the law's implications.

Next, Lampson presented his views of what successes our community has enjoyed. He presented a slide with a "Yes" column and a "No" column, and to demonstrate his even-handedness, he marked with a red dot the areas in which he'd worked. The audience laughed appreciatively at the prevalence of the red dots speckling the "failure" column of the slide. Among the successes he listed virtual memory, transactions, GUIs, the web, and algorithms; the failures included capabilities, fancy type systems, formal methods, software engineering, and distributed computing. He then spent several minutes explaining his

list of failures before too many members of the audience could become too incensed, and went on to share some "Maybes" which may yet pan out.

After completing the Yes, No, and Mabye lists, Lampson went on to say that because our community didn't invent the world wide web, we should be very embarrassed and take a long look at ourselves to figure out why we missed that opportunity. He postulated that we didn't invent it because it's too simple.

For the future, Lampson presented three issues: simulation, communication, and control. We've used simulation throughout the history of computing, but there are still a lot of things we don't know how to simulate: for example, clothing, cities, and complex organisms. We are doing pretty well with communication these days, but we're not yet doing a good job of getting all the information in the network where it's needed. Finally, for control, the canonical application used to be robots, but these days we have more interesting applications like MEMS and smart paint. There are also a number of programming challenges, such as dealing with uncertainty sensibly and harnessing the concurrency implicit in tomorrow's multiprocessor chips. Systems challenges include writing and meeting specifications in a measurable way and creating credible system simulations and analysis.

In the question and answer session, Jeff Mogul pointed out that one way for systems engineers to begin meeting Lampson's challenge would be for them to make it easier to find out what is going on within the system. Lampson replied that some engineers take that idea too far: some Microsoft products allow too many ways to look at what's gone wrong. Lampson did manage to flush out at least one person who would defend our community's role in the development of the web. Noah Mendelsohn argued that we did all the hard work such as connectivity and streams, while Tim Berners-Lee only caught the social aspects. Lampson quipped that in a way, that makes our failure even worse.

Slides from this presentation are at `http://www.research.microsoft.com/lampson/Slides/ComputerSystemsResearchAbstract.htm`

Distributed Systems (I)

Summarized by Steve Gribble, U.C. Berkeley

Manageability, Availability and Performance in Porcupine: A Highly Scalable, Cluster-Based Mail Service

Yasushi Saito, Brian N. Bershad, and Henry M. Levy (Univ. of Washington)

Yasushi, a UW student, presented this award paper. He described Porcupine, a cluster-based mail service that scales to handle very large client populations and that achieves high availability, high manageability, and graceful degradation in the face of failures. Porcupine makes extensive use of "functional homogeneity": any node in the cluster can perform any task, although tasks have affinity to some nodes that make use of soft-state indexes of the hard-state distributed across the cluster. Operationally, DNS round-robin is used to distributed SMTP requests across front ends; the front ends consult a soft-state "user map" (replicated on each node) to determine which node manages the user. The front end contacts that node, which looks up the user in a soft-state "mail map" to determine which nodes in the cluster currently store fragments of that user's mail. If a change in the cluster is detected, a cluster-wide membership protocol is run; a side-effect of this protocol is the generation of a new user map. A (parallel) scan of all disks in the cluster is then used to generate a new mail map; this scan only has to touch a small portion of the on-disk state, roughly proportional to the number of nodes redistributed in the user map after a cluster change. Porcupine also uses optimistic, eventually consistent replication - there is a small window of inconsistency after failures, but the nature of the inconsistency is extra copies of mail will be stored,

rather than mail will be lost. (However, the exact details of this inconsistency and a rigorous analysis of failure modes were not in either the presentation or the paper.) Porcupine uses dynamic load balancing to decide where in the cluster new mail fragments will be stored; a "spread" constrains the number of nodes on which a single user's messages will be stored, but within this (soft) constraint, queue lengths of pending RPCs on destination nodes are used to decide where to send message operations.

Measurements show that Porcupine's message throughput scales linearly with cluster size (the measurements show up to a 30 node cluster). With no replication, Porcupine handles about 800 messages/second on a 30 node cluster. With replication (and NVRAM to absorb log writes), Porcupine handles 400 messages/second. By comparison, sendmail+popd handles about 250 messages/sec on the same cluster. Measurements also show that Porcupine handles heterogeneity in the cluster and in the workload very well: with spread of 4 storage nodes per user, if 10% of nodes are made faster, Porcupine gets a 25% increase in throughput.

Questions

Greg Nelson, Compaq SRC: Q: Your performance graph showing message throughput during recovery suggests that your reconfiguration algorithm works well for 30 nodes, but how does that scale to larger cluster sizes? A: We only had 30 nodes, so we couldn't explicit measure higher sizes. The most expensive step in reconfiguration is the disk scan—takes 30 millisec over all nodes, and that should be independent of cluster size. So, it should be just as fast for larger clusters.

Jeanna Neefe Matthews, UC Berkeley: Q: You talked about spread, and also about not wanting voodoo constants, but it seems like spread is a voodoo constant. Is there a formula for calculating spread based on some parameters? A: The optimal spread depends on file system algorithms, buffering strategies, etc., but the basic rule is spread size of 2 is pretty much optimal heuristically, given a heterogeneous cluster. Q: Why is that fundamental, and not a voodoo parameter? A: Well, you only have a small number to choose from: not really voodoo; not too many choices make sense.

Gaurav Banga, Network Appliance: Q: A detail question: on the graph that simulated replication, you got performance to increase by 100 messages/sec by using NVRAM. Did you simulate the fact that accessing NVRAM is slower than memory? Would that bring replicated performance back down? A: No, we didn't take that into account.

Andy Tanenbaum, Vrije Universiteit: Q: You're barking up the wrong tree. One giant mail server for all the world is fundamentally wrong. The proper way is really distributed: sender sends to destination user's machine, giving distribution of mail all over world—the correct way to make it scale worldwide is to send mail to the owners machine. A: Firstly, AOL has a giant mail cluster, and stores user's messages in the AOL infrastructure, so they need a scalable mail server. Secondly, you can benefit from centralization, such as by enabling mail access all over world. Plus, you now have a homogeneous service architecture.

Jay Lepreau, University of Utah: Q: Users increasingly *do* want central mail systems, but several aspects of your architecture don't work well for that. Your workload is POP-like (i.e., retrieve all mail all of the time, deleting it from the server). Consider the Yahoo workload instead—all mail is always kept on servers. In this case, your spread will increase. You will need to migrate fragments together. Also, what about mailing lists? You don't want to replicate mail for all users in list. A: You're right. That's all future work.

Mohit Aron, Rice University: Q: Let's talk about load balancing in your system. Each node keeps track of all load in the system. What about oscillations and such? What if everybody gangs up on the least loaded node? A: Add randomization. You can compute many possibilities, the distribution over them,

spreads over the most least loaded, probabilistically.

Mike Rogers, University of Kentucky: Q: You balance load at the granularity of a message. Are there advantages to making this granularity smaller (i.e., if you divide up messages)? How much effort would that be? A: We haven't thought about it, but I can't see a specific advantage. Our assumption is that mail comes from all over in a highly concurrent/parallel fashion, so spreading individual messages doesn't seem like it would help. Maybe it would if you get a gigantic message.

On the Scale and Performance of Cooperative Web Proxy Caching

Alec Wolman, Geoffrey M. Voelker, Nitin Sharma, Neal Cardwell, Anna Karlin, and Henry M. Levy (Univ. of Washington)

In this presentation, Alec, a UW student, politely explained why large scale cooperative caching is a dead-end research area. The authors of the paper gathered simultaneous traces of the entire University of Washington campus (UW) and of Microsoft Corporation's Redmond site (MS), and used these traces to present a detailed analysis of cache hit rates under various population sizes, numbers of organizations, and cachability assumptions. The UW trace contained 7 days of traffic and 82.8 million requests, generated from 200 identifiable organizations within the campus (e.g., the music department). The average ideal hit rate (ignoring HTTP caching rules) local to an organization was 43%; adding a "perfect" cooperative cache across all organizations would increase the ideal hit rate to 69%. Including the HTTP caching rules decreases these numbers to 21% per organization, and 41% for a perfect cooperative cache. However, this perfect cache servers only 22984 clients, and could easily be run on a single machine serving the entire campus. Extrapolating on the client population size, hit rates increase only logarithmically with client population, up to the maximum cachability "ceilings" of 100% for an ideal cache and 61% for an HTTP-obedient cache. A city-wide cooperative cache (i.e., one serving around 1,000,000 clients) would hit these cachability ceilings, meaning that there is no point in having cooperative caches at larger scales than this.

In a second part of their study, the authors explored the benefits of cooperative caching across two large organizations by combining the UW and MS traces at a hypothetical shared cooperative cache. The MS traces (with 60,223 clients) increase the UW population by a factor of 3.6, but only increase the UW hit rates by 5.1% (for an ideal cache) and 4.2% (for an HTTP obeying cache). The authors also show that there is a negligible effect on user-perceived document delivery latency. From these results, they conclude that without significant workload changes, scalable cooperative caching schemes are essentially irrelevant. The speaker alluded to a similar result obtained from a detailed analytical model; these results are in the paper.

Questions

Jochen Liedtke, Universität Karlsruhe: Q: Assume you partitioned your traces not by department, but in some different way. Would you get the same results? Is there any difference when changing populations of organizations, not just the population of all clients? A: We got our organizations from considering departments. However, we also looked at our organizations vs. random organizations: hit rates were 5-10% higher with UW organizations vs. random organizations. So, it makes a slight difference, but not a huge difference.

Karin Peterson, Xerox PARC: Q: Have you run these traces at significant times apart? Is the amount of personalization that is creeping up making caching even worse? A: We've been tracing for about a year. Most of the high-level characteristics have been constant over the year. The most relevant characteristic is

the rate of overall uncachability. That rate hasn't changed in the past year, even though request rates have about doubled.

Amin Vahdat, Duke University: Q: Where do your misses come from? Is this a fair summary: about 1/2 of the requests from each person at UW aren't shared with anyone else at UW? A: The distribution of client requests to servers is very skewed: 50% of all requests go to the 800 most popular servers, even though there are over 200,000 servers in the trace. Q: Are the interests of people across your organizations so disparate that it's tough to get overlap? A: Some sites are very popular, but there is a very long tail.

David Cheriton, Stanford University: Q: I hate to defend cooperative caching, but: I thought that people wanted to use this for organizations on the scale of Saudi Arabia and Western Europe, where the cost of a cache miss is 100x the cost of getting the page out of a neighbor (due to poor connectivity to the USA). Doesn't this situation make it more beneficial to do cooperative caching? A: Proxy caching is of course beneficial. However, cooperative caching is also expensive in such environments. Q: I assume you're well-connected in Saudi Arabia, but leaving is expensive. This is exactly what you want for cooperative caching. A: We've shown your improvement in hit rate with cooperative caching. You should do your own cost/benefit analysis. If you think it's free to do cooperative caching, then fine, but if it's not free, it's not a good idea.

Preston Crow, EMC Corporation: Q: Clearly you showed some improved hit rate, and got concrete from the user's perspective with latency, but who cares about the users? ISPs care about bandwidth savings. A: True! In the paper, we do have graphs that show corresponding savings for bandwidth utilization. Q: Images are more cachable, and images are larger on average. Can we conclude therefore that larger things are more cachable? A: I don't remember the details, but I don't believe we saw a correlation between size and cachability.

Ram Rajamony, IBM Austin Research Lab: Q: What would happen if you compared UW to another university? In other words, what if you compared across similar organizations? A: Microsoft is less diverse than UW. For example, the hit rate for Microsoft is a bit higher than the hit rate for UW. Q: Would you get better cooperative caching with Rice and Houston than with UW and Microsoft? A: Once you're on the flat part of the hit rate curve, it just doesn't matter!

Client Systems

Summarized by Jay Lorch, U.C. Berkeley

The Interactive Performance of SLIM: A Stateless, Thin-Client Architecture

Brian K. Schmidt, Monica S. Lam, and J. Duane Northcutt (Stanford and Sun Microsystems)

The first talk of the session was Brian Schmidt's about SLIM, the Stateless Low-level Interface Machine. The central point of this work is that the fast, switched networks we have today are finally powerful enough to allow a transition back to the era of dumb terminals, paving the way for end devices with no need for administration and thus lower total cost of ownership. SLIM uses a simple encoding system to efficiently transfer frame buffer contents to these consoles. The evidence that this can work, and even be competitive with higher-level protocols like X, is well shown by user studies, benchmarks, and actual deployment. Brian also talked briefly at the end about his ongoing work on using persistent process sets, private namespaces, and a stateless operating system to ensure that you can recover from server crashes in this system as easily as from console crashes.

Peter Chen from the University of Michigan asked the first question. He wanted to know why Brian decided to design a new display encoding instead of using a standard one like MPEG. After Brian pointed

out that it wasn't really *his* decision, he gave two motivations for his colleagues' decision. First, the encoding settled on was to be generic, not targeted to any particular imaging mechanism, so that they could leverage this single technology for lots of different types of installations and keep the system useful for years to come. Second, it was simple and cheap to implement, unlike Peter's example of MPEG, which is "tricky."

The next question was from Michael Scott, from the University of Rochester. He pointed out that since we have already shown a tendency to ping-pong between a preference for dumb terminals and one with lots of smarts on the desktop, even if this is the proper solution for today, it's not very convincing that it'll be the solution for much longer. For instance, if immersive computing or other 3-D interfaces become the wave of the future, we could find ourselves throwing away all this cool console hardware that was supposed to last us through many years of computing. And, even today there are plenty of applications that require more smarts at the terminal than SLIM provides. Brian responded to the latter point saying that this system was targeted at installations where they run business applications, and will for the foreseeable future. He agreed that if immersive computing or something like that took over, SLIM couldn't accommodate that.

The third question, from Jonathan Shapiro of IBM, harkened back to the good old days of 1982, when the BLIT tried to do this same thing over 300-baud connections. The lesson learned at the time was that intelligence at the terminal was an inescapable necessity. So, Jonathan asked Brian, what made him think this work changes this invariant? The answer was that fast, switched networks are what changes the game now. And he conceded again that there were cases when you did need something like an MPEG decoder or 3-D pipeline on the terminal, but that there are plenty of users who just plain don't need that and know they don't.

Sitaram Ayer, from Rice University, pointed out the pitiful performance of the recent and similar VNC and essentially asked why SunRay could be so much better. Brian answered that it comes down to client-driven versus server-driven updates. While VNC has the client decide when to request an update, SunRay puts that decision in the hands of the server, which is better prepared to know when updates are needed and how much updating is needed.

Energy-Aware Adaptation for Mobile Applications

Jason Flinn and M. Satyanarayanan, Carnegie Mellon University

Jason Flinn, a CMU student, began the second talk by speaking passionately about the usefulness of improved energy management to solve one of the great important frontiers of system optimization: improving battery lifetime. He then got down to the details of his and Professor Satyanarayanan's work on energy-aware adaptation, as follows. Many multimedia applications can save a lot of energy in software by switching to lower levels of fidelity. With the authors' adaptation techniques, the user can specify a target battery lifetime, and the system will automatically and dynamically notify applications of the adjustments they should make to their fidelity levels in order to achieve the desired battery lifetime with amazing accuracy. An incidental observation from this work is that background power is a major impediment to getting massive battery lifetime increases, so *zoned backlighting*, the ability to selectively power down parts of the backlight, might help with this if we could get the hardware folk to implement it.

Margo Seltzer, from Harvard University, began the questioning. She complimented the author on his work, and agreed that this seemed like a good way to do adaptation of multimedia applications. However, she pointed out that she and most people she could think of used laptops very differently, for business and personal information management applications, which don't do any multimedia except perhaps to put an animated paper clip on the screen when you make the mistake of asking for help. She asked what can

be done for these applications. Jason responded that there wasn't much that could be done in the way of adaptation for these types of applications, and you had to rely on hardware energy management for the most part. He viewed his work as paying off in the imminent future, when multimedia applications were sure to become much more prevalent, even on people's laptops.

Ken Birman, from Cornell University, pointed out that the graphs in the paper of fidelity versus time were generally jumpy. He asked if the right thing to do instead was to degrade an application and then just leave it degraded, even if you later decided there was a small benefit from reversing the degradation. In essence, he said, there is a substantial cost to the user for any adaptation, and this wasn't reflected in Jason's scheme. Jason responded that indeed an important aspect of future work needed was to identify the magnitude of those costs relative to the costs of lower battery life and lower fidelity, so that the system could find the proper middle ground. Perhaps from his own experience watching the experiments run at lowest fidelity, he pointed out that there is something to be gained from running at higher fidelity even if the cost for this is an obvious transition.

Andy Tanenbaum, from Vrije Universiteit, jokingly answered Margo's question by suggesting that business applications could be adapted: a spreadsheet could just make guesses instead of doing actual calculations when battery life was low. He then asked if the real answer to business applications was to slow the clock frequency. Jason said that indeed these kinds of applications required hardware-level solutions, not the high-level software adaptations that his work focused on. And as for making guesses instead of actual calculations, there is a real-world example where you can reduce the amount of calculation you do in order to get satisfactory but not ideal answers: speech, which one of his group members is looking at.

Brian Noble, from the University of Michigan, remarked that PowerScope was cool, but that his gut reaction to it as a computer science person was that it was at base a multimeter, and thus icky hardware. He asked what could be done to give application writers reasonable information from PowerScope in a context they would understand and be comfortable with. Jason said that application writers should understand that you consume less energy if you consume less hardware, so energy profiling can show you the application performance in those terms. Energy profiling can also be useful in determining when a tradeoff between different pieces of hardware, like choosing a network-intensive algorithm instead of a disk-intensive one, pays off.

Preston Crow, from EMC, remarked on the similarity between this work and the work of the realtime community. The main difference is that the realtime community wants applications to scale their usage to meet deadlines while energy-aware adaptation wants applications to do it to meet energy targets. He asked how results from the realtime community could be leveraged in work like this. Jason agreed that there was a lot of similarity, but pointed out one important difference: that in the energy realm, there is only one goal that matters, namely making the battery last longer. This introduces a lot of extra flexibility in obtaining solutions.

Erik Cota-Robles, from Intel, pointed out that stochastic predictions of energy demands were all well and good in small benchmarks in controlled situations, but that the real world has annoying statistical properties such as the long tails that Vogels talked about in his paper on NT file systems. Jason said that naturally user studies were necessary, but his intuition was that his methods would do fine in the real world. Erik then cautioned that he'd had the same views in his work on realtime issues: things would look good analytically, and things would start out great in practice, but that then "something would happen." Jason again reiterated that his techniques work well in experimental cases but that validation would be necessary.

Networking (I)

Summarized by Neil Stratford, Cambridge University

Active Network Vision and Reality: Lessons From a Capsule-Based System

David Wetherall, Univ. of Washington

The talk focused on the experiences gained from the development of the ANTS Active Network Toolkit, a user level implementation of some 10,000 lines of code. Insights were generally positive. The ANTS architecture is believed to enable the deployment of useful services, with the exception of policy enforcement (firewalls) and services that override resource allocation (e.g., guaranteed rate). The performance was only a few orders of magnitude off ideal, but the results show it would be usable up to T1 rates using the current implementation. The levels of protection provided were equivalent to those in todays internet — services are unable to corrupt each other, but resource management remains an unsolved issue and is the subject of future research in the AN community. It was specifically pointed out that the use of TTLs are a weak solution, ignoring the possibility of attacks that cause work to be carried out on other nodes. The current architecture falls back on trust mechanisms.

Someone from INRIA kicked off the questions by asking if advantages could be found by using domain specific languages, such as PLAN, through for example proofs that the active program cannot loop. The reply was that it was possible for some programs, but unlikely in general. It was noted that analysing such programs to prove that code will not loop is good and such techniques will help in the future.

Jeff Mogul from Compaq WRL observed that there appears to be a contradiction in the design of Dave's system. The performance is best at the edges of the network, where there a lot of cycles per packet, but the applications described are better suited for the center of the network. Dave answered that there's no contradiction: the applications also work at the edges. Some things are also possible in the core, such as Random Early Discard and Explicit Congestion Notification. It's a continuum.

Ken Birman from Cornell stated that it's time to call the question of whether active networks are ever going to come to anything or not. He has yet to see one good reason for using an active network. Dave's response: you don't need novel applications; the old problems like multicast still aren't solved: that should be reason enough. Ken asked: but do we really need active networks for those old problems? The answer was that for fast deployment, we absolutely do need them. Dave believes that we're going to succeed, but in the end it won't be called "active networks" any more.

David Cheriton from Stanford followed up by declaring his distaste at agreeing with Ken, but that multicast *is* in all routers and it is just an issue of turning it on. People don't want to risk reliability, seeing extra activity as a threat. He also pointed out the trend of hardware forwarding in ASICs. The forwarding path will not be active but will have to push active packets to outside. Once outside the processing may as well be carried out anywhere on the network. Why does processing have to be carried out that close? Following some confusion and general laughter Wetherall replied that he didn't believe multicast was a solved problem; even if people want to use it, it still is not usable. It would be great if you could deploy your own version. Cheriton replied that it was the ISPs that don't trust multicast.

Timothy Roscoe from Sprint was only allowed to ask another question on the condition he *not* agree with Ken. Roscoe replied that he only slightly agreed and hence proceeded to declare that multicast was indeed in every router but not turned on, and that ISPs offer it only under special circumstances. It is generally disabled for reliability and resource management reasons. Playing the bad Telco guy, he added that billing was the real key which had been overlooked. How do we charge for chewing up resources? Wetherall replied that he had not considered billing, but that he will look at it once the other problems have

been solved! Roscoe captured the mood by asserting that once billing was a solved problem, there were no more problems. When the laughter had eventually subsided, Wetherall pointed out that everything can be tied back to bandwidth and packet rates.

Building Reliable, High-Performance Communication Systems from Components

Xiaoming Liu, Christoph Kreitz, *Robbert van Renesse*, Jason Hickey, Mark Hayden (Compaq SRC), Kenneth Birman, and Robert Constable (Cornell)

Robbert van Renesse presented recent research by the Ensemble communication group and the Nuprl automated formal reasoning group at Cornell, who are collaborating on a system for developing robust communication software from components. Component-based programming offers many advantages including software reusability and run-time system extensibility. However, typical component-based software may have poor performance due to excessive inter-component communication or redundant computation. Further, it is often difficult to verify that particular component compositions are correct (i.e., implement the properties required of the overall system). The goal of the Ensemble and Nuprl groups' collaboration is to tackle these problems in the context of communication protocols.

Van Renesse then described their "logical programming environment" for developing protocol software. The first piece of the system is Ensemble: an architecture in which programmers create protocol stacks by combining individual *micro-protocol* components. Ensemble provides a library of over sixty micro-protocol components that implement behaviors such as sliding windows, fragmentation and reassembly, flow control, group membership, message ordering, and other functions. The second piece of the system is Nuprl, a formal methods tool that operates on Ensemble components. Nuprl can read the (OCaml) source code for a protocol component and reason about it. More important, Nuprl is able to reason about *component compositions* in order to synthesize optimized protocol code.

Questions

Jay Lepreau from the University of Utah kicked off the questions by announcing that he was sympathetic to the approach, but had reservations about the practicality today. He made the comment that the specification needs to be written by hand, and anecdotal evidence suggests this means it will be difficult to write. He quoted the paper, which says that the proof of layers was "daunting," and even though it took less than a week to generate, it could take months for 4 layers. Van Renesse replied that they had only proved one layer of the ten, which took 40 pages and about a month. He added that formal tools will help to reduce many of the boring details. Lepreau expressed his understanding that a proof was required of each layer, but Van Renesse responded that they didn't have proofs for the other layers. It is possible to prove the correctness of a layer, or just assume it and prove things about the composition of layers. He agreed that it was hard and that the verification part had gone further than he had originally expected.

Brian Noble from the University of Michigan noted than when you have proofs of each layer you can compose them, but asked when did optimization continue to respect the conditions of the original. Van Renesse replied that Nuprl doesn't only output the code, but also a proof of correctness, which is fully automated.

Karin Petersen from Xerox PARC concluded by asking how they chose which properties to specify, as the proofs could only be as strong as the specification. Van Renesse responded that they could list properties, but it wouldn't make sense. They started by trying to enumerate the properties but ended up with long and complicated lists that were difficult to make internally consistent, so they used abstract

behavioral specifications instead.

File Systems

Summarized by Jeanna Neefe Matthews, U.C. Berkeley

File System Usage in Windows NT 4.0

Werner Vogels, Cornell

Werner Vogels started the session with an entertaining talk describing his study of file system usage in Windows NT 4.0. He had data on 237 million trace records from 45 NT workstations—plus cartoons, a top-ten list, and cookies!

Vogels began by explaining the goals of his study. He wanted to provide a new data point in the tradition of the 1985 and 1991 Sprite and BSD file system tracing studies on which much of the file system research in the last decade has been based. He pointed out that many of the measurements presented in these earlier studies are not statistically significant. Therefore, a major motivation for his work was to provide rigorous statistical analysis of file system trace data. Vogels also wanted to study the interactions of components within the Windows NT File I/O system, including the use of the important—but virtually undocumented—"FastIO path." He mentioned that unlike earlier file system tracing studies, he began his analysis with nothing in particular he was trying to prove.

Vogels then administered a little quiz, giving out cookies for the best answers (or at least for the best answers from people close enough to throw cookies to). He stumped the audience with questions like, "What is the most active directory in an NT file system?" and, "What is the cache read-ahead size set by NTFS?" He concluded that everyone really needed to read the paper since he heard so many wrong answers.

The majority of the talk was structured as a list of top ten observations of NT file system usage (a la David Letterman). Some interesting facts included: i) The file size distribution on various NT file systems looks remarkably similar because executables, DLLs, and fonts are huge and dominate the distribution. ii) 95% of changes to the file system are in the WWW cache and those occur mainly in the user profile. iii) 90% of files are open for less than 10 seconds for data transfer and less than 20 milliseconds for control operations. iv) 74% of opens are for control operations versus only 26% for data access. v) 80% of newly created files are deleted within 4 seconds.

Throughout his talk, Vogels returned to two major points: the presence of high variation and extremely heavy-tailed distributions of almost every measurement taken and the high variability between the systems traced. As a result, he concluded that it is quite misleading to combine all the trace data and treat it as characteristic of a typical NT workload. He hypothesized that one cause of this high variability may be that, unlike for UNIX, many people of varying skill levels develop Windows applications. He gave several examples of file access patterns which can only be described as extreme inefficiencies in the application.

Werner mentioned his intention to make his traces publicly available, but said that he was not yet ready to do so.

Questions

Bill Bolosky, Microsoft Research, recalled Vogel's data point that three-quarters of the files are over-written within 0.7 milliseconds after close and pointed out that this was an order of magnitude faster than a disk can seek. Bolosky asked Vogels to comment on what is causing this strange effect (and hopefully to

advise how to stop it). Vogels hypothesized that it might be intermediate files from a compiler or other data written for consistency and then removed. Bolosky emphasized that it could not be for recovery purposes since it is obviously not making it to disk.

Margo Seltzer, Harvard University, challenged Vogels' claim that the earlier tracing studies at Berkeley had specific goals they wanted to prove. She then asked Vogels what his data suggest about the right way to build next generation file systems. Vogels said that in his observations, the NT file system actually performed quite well (e.g., cache was never full so there was no need for new caching policies). He mentioned the possibility of predicting what a given process will do next or categorizing how specific file types are typically used to optimize their handling. He said that in general, however, he had no specific advice and no particular agenda at this point.

Deciding When to Forget in the Elephant File System

Douglas S. Santry, *Michael J. Feeley*, Norman C. Hutchinson, Alistair C. Veitch (HP Labs), Ross W. Carton, and Jacob Ofir (University of British Columbia)

Mike Feeley gave the presentation on the Elephant File System. He explained that modern file systems focus on protecting against data loss due to system or media failure. However, they stop short of protecting users from themselves! Accidental deletes or overwrites due to user or application errors always cause immediate data loss. Feeley argued that this limitation is simply an artifact of limited storage capacity and that current technology trends indicate an opportunity to remedy the situation. Specifically, disk capacity is rising at approximately 60% per year; 25–30 GB disks are already available and at the same rate 250–350 GB disks will be available in 5 years. Given these trends, Feeley concluded that disks are now large enough to store some old versions, but are still not large enough to maintain everything indefinitely. Therefore, the challenge for the Elephant File System was to decide when data can be safely deleted.

The key idea in Elephant is for the system, rather than the user, to decide when storage can be reclaimed. Old versions which are sufficiently important to retain must be distinguished from old versions which can be regenerated or which have become virtually indistinguishable from other retained versions.

Elephant has two distinct strategies for versioning. First, to provide near-term reversibility, the system retains all versions for a limited time. Second, the system identifies landmark versions to retain indefinitely. Elephant uses heuristics to automatically identify the landmark versions of user-managed files, based on the observations that (1) files tend to be edited in a bursty fashion and (2) a user's ability to differentiate between two versions degrades over time. To exploit these tendencies, Elephant observes the time line of edits to a file, identifies "barrages" of edits, and retains the version before the beginning of the barrage as a landmark version. Then, over time, the system increases the minimum time between distinct barrages.

Feeley reported the authors' experience with a prototype version implemented as a VFS in FreeBSD 2.2.8. This prototype stored 15 GB of data and was used for development and document preparation by 12 active researchers at HP Labs. Data on file type distribution and write traffic distribution was collected from the prototype in this environment. They classified files into five categories: source, documents, derived, archive, and temporary. For the derived and temporary files, no older versions were retained (keep-one). For the archive files, all versions were retained until they reached a certain age (keep-safe). For the source files, document files and all unclassified files, the system retained landmark versions as identified by the heuristics described above (keep-landmark). They found that the files designated keep-landmark accounted for 62.4% of the files but only 15.2% of the bytes in the file system. Files designated keep-safe represented 3.9% of the files and 28.5% of the bytes. Perhaps more interestingly, they found that almost all of the write traffic (98.7%) was destined for files designated keep-one. So only 1.3% of the write traffic needed

to participate in any version control. This clearly demonstrates the importance of per-file version control. In another metric, Feeley said that a 30 day history computed on the file system granularity would require 3.4 GB while a 30 day history in Elephant required only 0.042 GB.

Questions

Greg Nelson, Compaq Research, said that he liked this paper a lot. However, he said that he would like to be able to access the old versions in more complicated ways. For example, a user might remember that 5 years ago he or she had processed a certain environment variable switch but not know the right file name or the right date. Feeley replied that in his study questions like that didn't come up because the system was only used for a few weeks. However, he agreed that Elephant cold be viewed as a historical database and that it would be good to support queries by attributes other than just filename and time.

Mohit Aron, Rice, asked why they didn't implement Elephant as a front end to a version control system at user level. Feeley answered that they wanted it to have the same interface as a file system. In addition, the performance would not have been as good. For example, they couldn't have done copy-on-write at user level.

Separating Key Management From File System Security

David Mazières, Michael Kaminsky, M. Frans Kaashoek, Emmett Witchel (MIT)

This summary by Jon Howell, Dartmouth College.

Mazières presented a secure remote file system called SFS, built around the concept of self-certifying pathnames. The goal of the project was to allow users to access and share files globally while placing a low load on system administrators. To that end, the group took cues from the success of the Web by ensuring that anyone can create an SFS server, any user can access any server from any client, and any server can reference any other server.

Mazières framed the main contribution of SFS as separating key management from the file system. In SFS, public keys are embedded in every remote file path ("self-certifying pathname"); file system clients verify that the contacted server indeed holds the corresponding private key before trusting the server. Key management in the context of SFS, then, simply involves arranging for users to access the correct self-certifying pathnames. Key management may be manual, perhaps by installing on the client a human-readable symbolic link pointing to a self-certifying pathname. Alternately, key management may be more sophisticated, using shared secrets or certification authorities to retrieve self-certifying pathnames.

Questions

Jochen Liedtke of Universität Karlsruhe pointed out that since SFS is independent of key management, it is also independent of revocation, and thus can hold only a cache of public keys. If a public key changes (due to revocation), how can one find it again? Mazières suggested that the key management agent might install a symlink to bind the old Host ID (the root of the self-certifying path that securely identifies a server) to the new one. Liedtke asked that, since the Host ID depends on a secure hash, what do you do if the hash is proven weak? Mazières' response: "You'd better upgrade your software."

Karen Petersen from Xerox PARC asked about how to distribute a new key. If keys are contained in links, you would have to modify the link to point to a path containing the new key. Mazières replied that this was true; perhaps there would be multiple links pointing to the same server, and users could find a link that had been updated and follow that one.

Cliff Neuman of the University of Southern California asked about location transparency. As data moves, its name changes, and so does its key. Mazières replied that we already have many names for one file: symlinks. Most users will use symlinks; when the data moves, change those. Neuman asked about pathnames embedded in files, and Mazières replied, "Yeah, change those files."

Neuman also asked if it was possible to have a policy for name remapping that depends on where you are accessing the file from (for locality). Mazières replied that all clients are configured identically, but that the user agent has all user-specific state, and is forwarded site-to-site to follow the user.

OS Kernels

Summarized by Kip Walker, CMU

Integrating Segmentation and Paging Protection for Safe, Efficient and Transparent Software Extensions

Tzi-cker Chiueh, Ganesh Venkitachalam, and Prashant Pradhan (State University of New York at Stony Brook)

Tzi-cker Chiueh presented their paper, in which the segmentation and paging hardware in the x86 family of processors was used to provide memory protection from kernel- or user-level extensions. Most of the talk was spent on an explanation of the underlying hardware mechanism, and how it had been applied to memory protection for extensions. The evaluation showed that protected calls into extensions were almost as fast as unprotected calls.

A lengthy Q&A session began with Jochen Liedtke commenting that details of the lret/lcall instructions could be found in Intel's OS-writer's guide. The first question came from Jeff Chase, who inquired if work had been done to address protection issues unrelated to memory, such as preventing infinite looping in an extension. Chiueh responded that the focus of the work was on memory protection; however, a basic timer mechanism had been implemented to prevent an extension from stealing the CPU forever.

An unidentified audience member asked if extensions could be protected from each other without expensive operations. Chiueh's answer was that the same efficient mechanism could probably not be used for memory protection between extensions. The next question, from Stefan Savage, pursued the idea that memory protection is just a part of the solution in supporting safe extensions. He observed that in his experience, the majority of problems were subtler, such as invariants being broken as a a side-effect of crossing interface boundaries. Again, Chiueh responded that memory protection was the focus of the work, and that in his group's limited experience with building extensions, such a situation had not occurred.

In an effort to get things stirred up, Geoff Kuenning asked what the speaker had learned about "real computers" from this project. Chiueh nimbly deflected this by pointing out that x86 processors are rather common. The last question came from Jonathan Shapiro, who inquired why such a mechanism is useful when Liedtke and others have demonstrated such fast context switch times. Shapiro let the question go unanswered with the comment, "Put up or shut up is a fine answer."

For more information, see `http://www.ecsl.cs.sunysb.edu/palladium.html`

Cellular Disco: Resource Management Using Virtual Clusters on Shared-Memory Multiprocessors

Kinshuk Govil, Dan Teodosiu (HP Labs), Yongqiang Huang, and Mendel Rosenblum (Stanford)

Kinshuk Govil, a Stanford student, presented this award paper. Their work builds on Disco, a virtual machine monitor designed to allow commodity OSs to run on large SMPs. Cellular Disco brings scalable resource management and fault containment to Disco. Virtual machines run on a subset of "cells" in the SMP, and hardware faults disrupt only the VMs using resources in affected cells. Scalable resource management allows the full resources of the SMP to be used even if the operating systems running on the machine are not scalable.

Questions started with Ken Birman asking how faults are detected by the virtual machines, and how failures appear to applications. Govil responded that faults must be detected and contained by the hardware; the virtual machine monitor then shuts down all virtual machines that depend on any resources from cells affected by the faults. While it might be possible to do something interesting with alerting the operating systems to hardware failures, the designers of Cellular Disco wanted to modify the operating systems as little as possible. While software faults will take down a particular VM, neighboring virtual machines will run undisturbed due to the isolation provided by the underlying virtualization of the machine.

An unidentified questioner requested Govil's opinion of the following alternative design: if the operating system source were available, one could consider offering a different API to the OS rather than simply virtualizing the hardware's interface. Again, the response was that changing the operating system was not considered to be an option, so nobody had investigated such a possibility. Mike Swift inquired if the kernel text segments were shared across processors, and if the number of virtual CPUs known to one of the operating systems could be changed mid-execution. Govil answered that the kernel code was indeed shared, and that changing the number of VCPUs was not possible without restarting the virtual machine. The final question followed up on the idea of operating system modifications. Jonathan Appavoo asked how much knowledge of the OS had been needed to design Cellular Disco, and what changes to the OS had been necessary. Govil responded that the main changes consisted of instrumenting the idle loop and memory allocation routines. He further noted that there are ways to achieve the same functionality in the virtual machine monitor, without touching the OS.

For more information, see `http://www-flash.stanford.edu/~kinshuk/`

EROS: A Fast Capability System

Jonathan S. Shapiro, Jonathan M. Smith, and David J. Farber (Univ. of Pennsylvania)

After warming up the crowd with a "letter home from Camp SOSP" slide, Jonathan Shapiro, now at IBM T.J. Watson Research, presented this paper. Shapiro began the talk by arguing that in addition to performance, system designs must pay attention to security, integrity, high availability, fault tolerance, and evolvability. EROS achieves some of these goals through its use of capabilities as its fundamental building block. All resource access is accomplished through invocation of capabilities. The system keeps overheads low by using well-chosen abstract objects and caching techniques.

Jochen Liedtke brought up the observation that with a persistent system, even faults may end up being preserved. Shapiro responded that in his experience, faults had only come from active development on the system and that he had never seen bad state make it to the disk. A consistency checker in the kernel was designed to catch problems before they get checkpointed. Drew Dean commented that the work seemed to focus solely on discretionary security, and wondered if that implied that EROS was not interested in

mandatory, multi-level security. Shapiro answered that mandatory security was certainly of concern, but EROS was intended for exploring how more discretionary control could be given to users.

Satya closed the session with a well-phrased inquiry about what Shapiro saw as applications needing EROS (as well as capabilities in general). With the persistence architecture and very fast transaction facility, EROS could be applied to advanced databases. Shapiro also described a "stock negotiator" as possible application, where many buyers are trying to coordinate a stock price. Regarding the benefits of capabilities, Shapiro argued that object systems aren't "active"—that when persistence and controlled communication between objects is desired, capabilities are a good solution.

For more information, see `http://www.eros-os.org/`

Distributed Systems (II)

Summarized by Jon Howell, Dartmouth College

The Design and Implementation of an Intentional Naming System

William Adjie-Winoto, Elliot Schwartz, *Hari Balakrishnan*, and Jeremy Lilley (MIT)

Balakrishnan described the Intentional Naming System (INS), the goal of which is resource discovery in dynamic and mobile networks. The design goals included expressiveness, responsiveness, robustness, and easy configuration. They achieve expressiveness with a language that lets applications express what they want, not where it is. Responsiveness is accomplished by routing individual messages by name (late binding). INS is robust due to its serverless, decentralized design, and easy to configure because its resolver nodes (that form the overlay network that implements the service) self-configure. Applications are offered two services: intentional anycast, which delivers a message to a single service with a matching name, and intentional multicast, which delivers a message to every reachable service with a matching name.

Petros Maniatis from Stanford asked why INS binds the name service to the delivery service, citing the overhead of a long name in each packet. Balakrishnan's replied that it allows the system to track mobility. "I could be arbitrarily mobile. I unplug from the Ethernet, use my wireless connection, and everything keeps working." He said that names were on the order of 100 bytes, and that he didn't see that as an overhead. All mobility systems depend on indirection, he said, and how you do indirection is important. We should avoid building unneeded infrastructure.

Satya, CMU, asked why the group built their own tree structure, rather than using an embedded relational database, which would have been much more general. Balakrishnan answered that they wanted something simple that would run on a device as simple as a Palm Pilot. Satyanarayanan predicted that the tree structure would become a restriction in the near future. Balakrishnan replied that they were adding more expressive operators, and argued that constraining the programmer was not always a bad thing. The developers asked themselves, what is the minimum set of operators needed to do something useful?

Dan Wallach from Rice asked about authentication in the presence of dynamism. He pointed out that we can be (somewhat) assured that the "yahoo" site is unique thanks to "the great big DNS root in the sky." The answer was that security was not a goal; Balakrishnan suggested that perhaps self-certifying names (the SFS paper from an earlier session) would be a solution.

Mike Swift from the Univ. of Washington asked why the messages are routed all the way through the overlay network when the first resolver knows the network address of the final destination. Balakrishnan said that this is exactly what happens in the anycast case; forwarding through the resolvers is used for multicast delivery.

For more information, see `http://wind.lcs.mit.edu/`

Design and Implementation of a Distributed Virtual Machine for Networked Computers

Emin Gün Sirer, Robert Grimm, Arthur J. Gregory, and Brian N. Bershad (Univ. of Washington)

Sirer presented a new implementation architecture for the Java Virtual Machine (JVM) that allows JVM services to be factored out and distributed across the network. This puts fewer demands on the client architecture (enabling "JVM on a lightbulb"), provides physical separation of the services, and makes client systems more manageable by locating some services on a common server.

Services amenable to factoring include bytecode verification, security policy enforcement, and bandwidth optimization. Code verification at a server saves substantial resources on the client. Maintaining security policies on a central server is attractive because it makes site security more manageable. Bandwidth optimization involves profiling runs of the application, then sending future clients only the commonly-used code, with stubs that load the rest of the code on demand.

The factoring design involves two phases: the first is to inspect class files a priori, and the second to dynamically rewrite class files to inject code snippets into the application. The injected code has access to data- and context-dependent values on the client.

Ed Felten from Princeton asked how the group could improve security by moving the verifier: "you have added stuff to the trusted computing base: the network channel, the server, the server operating system. . . ." Sirer agreed, but said that moving the verifier addresses operational problems. He compared site-wide policy enforcement to firewalls, in that it is easier to secure a single, well-placed host. Felten argued that site-wide configuration depended on the clients having this factored JVM; and that if you could ensure that the client has the right software to begin with, the problem would already be solved. Sirer pointed out that with central management, an administrator has only to do O(1) work to update the verification procedure for all of the clients.

Fred Schneider from Cornell mentioned that binary rewriting dates from the 1970's. He said that he had found binary rewriting as easy for the x86 as for the JVM, and asked to what extent a JVM was necessary to achieve the claimed benefits. "Why isn't the title, 'Distributed Services,' period?" Sirer agreed, saying that they limited their scope to their experience.

Sitaram Iyer from Rice asked whether the injected verification bloats code, and Sirer responded that the effect was quite modest.

Drew Dean from Xerox PARC asked about interactions among multiple class loaders. Sirer said that the subject was deep, but that all checks that require knowledge of the class loader are done in the injected dynamic code. Dean also asked how the system prevents attackers from creating a remote shadow class intended to replace a server-created class, and Sirer replied that they simply reject any remote class in the package "edu.washington."

Jon Tidswell of IBM Research asked whether the approach would scale to tens of thousands of clients at the University of Washington. Would memory on the server be a bottleneck? Sirer answered that VM would be the limiting factor, but that their benchmarks represented a worst-case scenario. He indicated that a beefy server at the firewall should handle plenty of clients. Tidswell recalled the 60-70% ideal caching figure for web documents from a prior talk, and asked how cacheable Java classes are. Sirer said he had no numbers. However, your scribe suspects that they would be very cacheable, since any dynamic content viewed in a Java applet is likely to appear as variability in a data file, not as a dynamically generated Java class file.

For more information, see `http://kimera.cs.washington.edu/`

Networking (II)

Summarized by Bill Dieter, Univ. of Kentucky

The Click Modular Router

Robert Morris, *Eddie Kohler*, John Jannotti, and M. Frans Kaashoek (MIT)

Eddie, an MIT student, presented this award paper. Click allows system administrators to easily build and extend a software router running in Linux by specifying how packets flow between provided router elements. Each router element performs one simple function, such as packet classification, scheduling, queuing, or traffic shaping. Click modules are connected with "push" or "pull" connections, which describe the packet flow through the router. An upstream module transfers packets to a downstream module over a push connection, pushing the data downstream. With a pull connection the downstream module initiates the transfer, pulling the data out of the upstream module.

A simple Click IP router routing between two 100 Mbps interfaces ran 90% as fast as a Linux machine configured as a router. Click was able to route up to 73,000 64-byte packets per second vs. around 81,000 64-byte packets per second for Linux. Adding the random early detection (RED) dropping policy to the Click router reduced its performance to about 93% of the simple Click IP router.

Paraphrased Questions and Answers

Q: Jay Lepreau, Utah: Handling interrupts takes 70–80% of your overall time. If you get rid of interrupts by polling you will not look so good. Can you comment on this? A: That depends on what you mean by look bad. We measured a 33 microsecond per packet latency for Click and a 28 microsecond latency for Linux. Much of the overhead is coming from virtual function calls. We may try dynamic code generation to get around the virtual function call overhead.

Q: Rimon Barr, Cornell: Can you learn from database optimization techniques in the router space? A: Thanks for the pointer to databases. They have done quite a bit of optimization work. We could generate one element that does the work of many and automatically substitute it for a group of elements. We have manually rewritten a sequence of elements into one large element and seen over 20% improvement in some cases. We also plan to use dynamic code generation to try to reduce overhead.

Q: Unknown: Not all modules are input/output driven; some are timer driver. How do you model that and how do you guarantee schedulability? A: We implement timers with timers. Click has timer elements. Scheduling is harder. Scheduling will be important in the future. We will look more at how Scout and others do scheduling. Routers have lots of queues, which are natural areas for scheduling. We may look at that too and try to figure something out.

Q: Unknown: You described how to get queue length. How do you get information about other kinds of elements? A: Click can look for any property an element might have. Elements can be designed to have any property you might want. The queue in the talk is just one example of how to find out about a property of an element.

Q: Timothy Roscoe, Sprint: Many routers use routing protocols that alter the routing table quite often. How long does it take to reconfigure the routing tables if you are doing a multicast join or something complex like that? A: We don't have figures for that because we don't have a normal routing table. Writing a new configuration takes less than 15 milliseconds. We can write to the /proc filesystem to change what an element does. To do routing we use a user level process like "gated" to poke the routing information into /proc filesystem.

Q: Ken Birman, Cornell: This question comes from naivete. How does Click compare with other work? How is the configurability problem classically addressed within routers such as Cisco? A: We don't know a lot about Cisco because their routers are proprietary. Cisco has their own software infrastructure. From talking to them, their routers are not as modular for out of the box solutions. Linux and BSD router code is mixed in with the rest of the kernel.

For more information, see `http://www.pdos.lcs.mit.edu/click/` Slides from this presentation are at `http://www.lcdf.org/~eddietwo/click-talk.ps`

Soft Timers: Efficient Microsecond Software Timer Support for Network Processing

Mohit Aron and Peter Druschel (Rice University)

Mohit, a Rice student, presented this award paper. Soft timers schedule timed events by checking for timer expiration in frequently executed parts of the kernel. Checking for timer expiration in the kernel reduces context switch and cache overhead compared to traditional hardware-based timers which interrupt when the event occurs. Techniques like rate-based clocking of TCP connections and network polling that suffer from the high overheads of hardware based timers on high speed networks become practical with soft timers.

Paraphrased Questions and Answers

Q: Unknown, Bell Labs: I like this paper, it looks good. However, you assume incorrectly that hard timers interrupt on every clock tick. That is incorrect. You can have the hardware interrupt go off every N cycles if nothing is scheduled. A: In a high bandwidth network the operating system needs to schedule lots of interrupts close together.

Q: Jonathan Shapiro, IBM: You measured performance and showed results for applications where things happen all the time. What used to happen at every tick now happens more often. What are the implications of this for real-time applications? A: Soft timers provide best effort triggers. They are not guaranteed to happen at precisely the time they are scheduled, so some events may miss their deadlines. If an application cannot tolerate that it will have to use hardware timers.

Q: Eric Cota-Robles, Intel: I agree there are problems with hardware timers. As a user of interrupts, interrupts have not changed since the 60's. What would be a better hardware interface? If the hardware interface could be redesigned, how would you do it? A: It would have as low overhead as soft timers. [audience laughs] Q:You don't care how to do it? A: No.

Q: Unknown: I agree on the usefulness of this facility. How will it change as hardware changes? Is the assumption true that interrupts will always have high overhead? A: That is hard to answer. On receiving an interrupt the CPU still needs to save some state, and interrupts will still cause cache locality losses.

Q: Greg Minshall, Siara Systems: Rate-based pacing is interesting, but it is not done because of the timer overhead, so this work addresses that problem well. There are a number of things that will speed up flow of TCP connections. What are the implications for the rest of the net of using rate-based pacing? You need to prevent a bursty load to the network compared with slow start. It is unfair to say rate-based pacing is 10 times better than TCP. It is better to say the load is spread over more time. Please withdraw that claim. A: We mentioned that slow start has burstiness because it must wait for ACKs from the receiver. There is no reason for the sender to wait if the bandwidth of the network is known.

Q: Unknown, Univ. of Washington: Knowing the bandwidth is a big assumption. UW has done rate-based pacing and found that it didn't work well for many parameters because of properties with lots of connections. Feedback protocols are sensitive to changes. If an application uses the network less it could affect the net performance. Will applications resort to voodoo, like calling getpid(), to improve performance? A: We can use techniques to approximate bandwidth when opening a connection, such as assuming it is the same as the bandwidth for the last connection to the same host. Packet dynamics need to be studied more with rate-based pacing.

For more information, see `http://www.cs.rice.edu/CS/Systems/Soft-timers/`. Slides from this presentation are at `http://www.cs.rice.edu/CS/Systems/Soft-timers/soft-timers-talk.ps.gz`

Real Time

Summarized by John Regehr, Univ. of Virginia

Progress-Based Regulation of Low-Importance Processes

John R. Douceur and William J. Bolosky (Microsoft Research)

John presented this paper describing "MS Manners," which is designed to prevent a low-importance task (like a disk defragmenter or content indexer) from interfering with the performance of other applications. The idea is to establish a baseline for the rate of progress of the low-importance process and then to watch for its performance to degrade. Degradation is assumed to be caused by resource contention; when it occurs, the low-importance process is suspended. It awakens after some timeout—if contention is still present, it suspends for a longer time using exponential backoff. MS Manners assumes that the rate of progress of low-importance processes can be measured, and that the impact of resource contention is symmetric; that is, it slows down both contending processes equally. Advantages of MS Manners are that it is resource independent, it involves no kernel modifications, and it requires no human intervention or other knowledge about processes' use of resources.

David Steere, OGI, mentioned that there has been a lot of feedback work recently (for example, the Quasar Project at OGI `http://www.cse.ogi.edu/DISC/projects/quasar/` and Feedback Control Real-Time Scheduling at UVA `http://www.cs.virginia.edu/~cl7v/fcs.html`). He asked about support for applications with timing requirements, and if John and Bill had thought about methods of degrading the progress of the low-importance task other than suspending it? John answered that they had not considered time-dependent applications or other methods of degrading progress, but that the amount that the low-importance process is allowed to interfere with the progress of other applications is tunable.

The second questioner asked if there were interference or instability effects resulting from the exponential backoff, and if the authors had considered adding a random jitter to the backoff times. John said that instability has not been a problem—since MS Manners does not allow concurrency between low-importance process, there is no way for backoffs to collide or otherwise interfere with each other.

Jason Nieh, Columbia, asked why John and Bill didn't also measure the progress of the high-importance process in order to relax the assumption of symmetric degradation. John responded that their assumption is that they don't know what the high-importance processes are. This makes sense in an open system like Windows.

Ed Bugnion, VMware, said that he found the project interesting, but that he is more interested in retaining interactive performance under NT while running the compiler. John said that MS Manners would be suitable for this application if the compiler were considered to be the low-importance application.

Scribe's note: the Windows NT scheduler is not tuned to provide good interactive response when there are compute-bound tasks at the same or higher priority. An easy workaround is to reduce the priority of the compute-bound task.

Geoffrey Kuenning, UCLA observed that OS resource meters often stall, jump forward, and are otherwise unreliable, and asked if this is a problem for MS Manners. John responded that these meters often aggregate a lot of information that makes them misleading, but that this had not been a problem for MS Manners since it uses an application-specific progress metric.

Borrowed-Virtual-Time (BVT) Scheduling: Supporting Latency-Sensitive Threads in a General-Purpose Scheduler

Kenneth J. Duda and David R. Cheriton (Stanford)

Ken presented this paper, which argues that the BVT algorithm can be used to effectively schedule a wide variety of real-time and non-real-time applications. BVT is a proportional-share algorithm similar to start-time fair queuing (`http://www.usenix.org/publications/library/proceedings/osdi96/vin.html`), with the additional feature that threads have a "warp" value that prevents them from losing their share while they are not running; a high warp value increases the likelihood that a thread will be scheduled soon after it awakens, but does not increase its share. Ken next addressed the question of how to choose the share and warp values for tasks, and then discussed using BVT for real-time tasks. He proposed a two-level scheduling hierarchy: a top-level BVT scheduler with admission control that schedules real-time tasks and a lower-level BVT scheduler that admits any task. This allows the CPU needs of real-time tasks to be isolated from other tasks. Implementing BVT involved adding less than 500 lines of code to the Linux kernel. Ken next compared BVT to reservation-based scheduling; BVT was shown to out-perform reservations when the error in the estimate of the CPU time needed by the reservation exceeded about 15%.

Jochen Liedtke, Universität Karlsruhe, said that he likes this scheduler, and asked the first question: had they proven any real-time properties about BVT? Ken said they had not.

Mike Jones, Microsoft Research, thought that BVT was the most practical of the weighted fair-queuing schedulers, and agreed that estimating the amount of CPU time needed for a reservation is difficult. However, based on his experience with the Rialto system, he thought that the scheduler needs to know about the period of real-time applications. Ken said that tasks with tight requirements on their period should be put into the root scheduler and assigned a warp value high enough that they are likely to run whenever they wake up. Mike responded that this makes warp seem analogous to priority, and that this scheme would be vulnerable to the priority-inflation phenomenon that happens when different people assign priorities to the applications that they write: everyone assumes that their application is the most important. Ken responded that he didn't see this as a problem. Scribe's note: the central question here is who assigns warp values. If warp is computed by an admission controller, or if there is coordination among application writers, then good warp values can be chosen. Mike's point was that in absence of global coordination, it is better if application writers specify the resource needs of their applications directly, rather than specifying their relative importances.

Stefan Savage, UW, had experience with one of the early reservation-based schedulers, and said that he now prefers the BVT style of scheduling. However, he was concerned with the difficulty of manipulating dimensionless parameters like share and warp, as compared to estimate and period in a reservation system which correspond to actual physical quantities. He wondered if there might be a way to specify the parameters more intuitively, but retain the benefits of BVT. Ken said that this would be great, if we can figure out how to do it.

Mohit Aron, Rice, asked if they might consider using short quanta instead of warp, since this would be a different way to reduce scheduling latency? Ken answered that by default BVT has very long quanta (200ms) while still offering low scheduling latency for applications that need it. This is the right thing to do, given the detrimental effects of context switches on cache locality.

Greg Minshall, Siara, asked Ken to compare BVT with start-time fair queuing (SFQ). Ken said that although SFQ has bounded worst-case response-time, the bound is long since it's proportional to the number of tasks in the system. Scribe's note: as noted above, the same criticism might be applied to BVT if application writers are each allowed to choose the warp value for their application—low latency is achievable only if a person or admission control system with knowledge of all tasks assigns warp values.

David Steere, OGI, asked if there might be some relation between weight and warp in BVT to period and amount in a reservation based system? Ken thought there might be, and that this mapping would be implemented in the admission controller.

Erik Cota-Robles, Intel, wondered if BVT would do well when scheduling multiple applications with similar scheduling needs? Ken answered that multiple MPEG players worked fine, as long as there is enough processor time to run them all. Erik commented that some kinds of signal processing are highly predictable (arguing for a reservation-based scheduler).

Jason Nieh, Columbia, liked the comparison of BVT with reservations, and asked if it has been compared to fair-queuing without warp? Ken responded that taking away warp would be bad for response time, and interactive applications would not do well.

EMERALDS: A Small-Memory Real-Time Microkernel

Khawar M. Zuberi, *Padmanabhan Pillai*, and Kang G. Shin (Univ. of Michigan)

The last talk in the last session of the conference was given by Padmanabhan. The EMERALDS system is based on a re-engineering of OS services for small-memory systems. For example, dynamic overhead can be avoided since the location of resources is statically known, and many object instantiations are static also. To reduce task scheduling overhead, EMERALDS uses a new approach called combined static/dynamic scheduling. This avoids some of the dynamic overhead of earliest-deadline first scheduling without incurring the high schedulability analysis and lower utilization bound of rate-monotonic scheduling. EMERALDS also contains a number of optimizations that provide extra information to blocking calls in order to avoid context switches, to optimize priority inheritance to take constant time in the common case, and to speed up state message and mailbox-based communication.

Jason Nieh, Columbia, asked the first question; he wondered if system overhead was really an issue, given Moore's law. Padmanabhan responded that speed is an issue since things change more slowly for embedded chips, most of which are still eight or sixteen bits.

Werner Vogels, Cornell, said that the state messages in EMERALDS are quite similar to the ones in the MARS system. Padmanabhan said that while MARS is related work, EMERALDS has improved upon its state messages in several ways.

Mohit Aron, Rice, ask for clarification: does EMERALDS implement multiple address spaces? Padmanabhan said that there are versions with and without multiple address spaces. Mohit asked why we don't trust applications to remain in their address spaces if we trust them to provide the correct timing constraints. Padmanabhan answered that protecting against bugs in code is the issue, rather than trusting the author.

Jack Stankovic, Univ. of Virginia, said that for the small number of tasks in a typical embedded system there's not much difference between the overheads of rate-monotonic and earliest-deadline scheduling, and

that the combined algorithm will probably take up more memory. Padmanabhan answered that in principle the system generation tool could include only the scheduler that was needed, avoiding the memory cost when more general scheduling is not required.

Work-in-Progress Reports

Summarized by Chris Diaz (Univ. of Kentucky) and Neal Cardwell (Univ. of Washington)

Barbara Liskov, Jeff Mogul, and Fred Schneider selected the WIP talks from hardcopy abstracts submitted on-site. They ran a typically fast-paced but smoother-than-usual session, carefully pipelining actual speaking with the mechanics of visual aid prep. Thus, no speaker was "gonged" for exceeding their big six minutes.

Lessons Learned from a Wide Area Sharing Platform

Marc Shapiro (INRIA Rocquencourt)

Shapiro discussed the PerDiS system, which supports sharing across a wide area network. The system supports many applications, though the focus is on applications of concurrent engineering such as that used in building and construction. Such applications require shared mutable, or writable, data for users to cooperate and exchange information. In PerDiS, a user may obtain and cache a copy of shared data, which will not be changed by the write of another user. The server accepts a client's changes if no other write has occurred to the data in the meantime. Otherwise the server rejects the changes, and the application can recover. Such rejections occur rarely in practice, as engineers often agree upon such actions.

The PerDiS platform automatically manages both distribution and persistence, which is a big win for applications. A significant lesson is that the coarse granularity is essential to making distributed sharing manageable: failures can only occur at transaction and file boundaries. Performance is better than fine-grain systems thanks to much fewer messages. Also, the DSM abstraction makes the system very easy to understand and to use. The negative lessons are that much effort is needed to build a significant large-scale system, and that language integration is important for supporting legacy applications.

PerDiS `http://www.perdis.esprit.ec.org/` is available as open source.

Mutually-Distrusting Cooperative File Systems

William J. Bolosky, John R. Doucher, Marvin Theimer (Microsoft Research)

A symbiotic distributed file system consists of multiple independent file systems that work together without trust. For example, the components in such an environment may consist of stock traders. While each depends on the others for generating prices and trading shares, each distrusts the other and does not disclose information that reveals one's strategy. The goal of this work is to design and build a serverless, distributed file system whose nodes do not trust the other nodes.

The authors make three observations. First, system vulnerability is inversely proportional to the number of nodes that store data. That is, a completely centralized system is more vulnerable to attack than a decentralized one, because an attacker only needs to compromise one component to bring down a centralized system. Second, client resources are numerous, thus leading to decentralized data storage. Third, "bigger is better": it is easier to "hide" replicas of data from a malicious attacker in a large system with many nodes than in a small system with only a few nodes. Stated somewhat differently, it is easier to hide a needle in a large haystack than in a small one.

The authors conclude that the world is becoming more centralized, for example, with large systems for e-mail, auctioning, buying, and trading. While such approaches simplify administration, they are susceptible to the types of attack that the authors want to avoid.

For more information, see `http://research.microsoft.com/`

A Highly Configurable Emulation Facility for Distributed Systems and Networks

Jay Lepreau, Chris Alfeld, David Andersen (MIT), Kristin Wright (University of Utah)

Lepreau motivated the project by giving two slides of quotes from papers, reviews, and job talks. His implied message was that current methods of evaluating experimental research in distributed systems and networks are either poor or very hard to use. To remedy this, Lepreau's "Flux" research group at Utah is constructing a "configurable Internet in a room" that will be remotely available to all researchers. The system is large: on the order of 220 nodes and 1000 links, connected by a very large switch. The topology is configurable, the link bandwidth/latency/errors/drops are configurable, and every bit of software on the nodes is replaceable by the users.

Such a system raises many research challenges. These include providing a reasonable user interface; calibration, evaluation, and scaling; artifact detection and control; and the NP-hard problem of mapping the virtual network to the available physical resources. Solving the mapping problem is cruical in avoiding bottlenecks in the physical network that inadvertently affect the virtual network.

The system should be limping "soon," and Utah is looking for early users and sponsors. For more information, see `http://www.cs.utah.edu/flux/`

Incorporating MEMS-Based Storage into Computer Systems

Greg Ganger, Steve Schlosser, John Griffin, and David Nagle (Carnegie Mellon University)

MEMS-based storage is an exciting new technology that could provide significant performance gains over current disk drive technology and at costs much lower than EEPROM technology. Based on MEMS (MicroElectroMechanical Systems), this non-volatile storage technology merges magnetic recording material and thousands of recording heads to provide storage capacity of 1–10 GB of data in under 1 cm^2 area with access times of 1–3 ms and streaming bandwidths of over 50 Mbytes per second. Because MEMS-based storage is built using photolithographic IC processes similar to standard CMOS, MEMS-based storage has per-byte costs significantly lower than DRAM and access times an order of magnitude faster than conventional disks. MEMS-based storage can incorporate both storage and processing into the same chip, i.e., a single computing "brick" that contains processing and both nonvolatile and volatile storage. Developing these concepts is the central focus of a new research center at CMU called CHIPS.

Although MEMS-based storage devices may still be several years away from commercialization, their potential impact in reducing the memory gap makes them an important technology for systems designers' consideration. Therefore, the authors have begun the exploration process, seeking an understanding of how MEMS-based storage can improve application performance and how different MEMS device characteristics can fundamentally change the behavior and design of storage systems. Early results indicate that MEMS-based storage can reduce application I/O stall times by over 80-99% for a set of five file system and database workloads. The resulting speedups for these applications range from 10% to 20X, depending mainly on the ratio of computation to I/O. Ongoing work refines the device models, explores how they should change system architectures and memory hierarchies, and investigates newly enabled applications.

For more information, see `http://lcs.web.cmu.edu/research/MEMS/`

System Infrastructure for Ubiquitous Presence

Umakishore Ramachandran (Georgia Tech)

A hardware continuum of Computation, Communication and Interaction (CCI) devices exists. Currently, the CCI devices are explicit with use, for example, with laptops or cell phones. Ramachandran expects that future CCI devices will be embedded in many other places, such as lamp posts and ceiling panels. The hardware continuum will then range from single chip CCI (say, in a wristwatch) to distributed CCI devices (say, in a collection of ceiling panels). The challenge is to seamlessly integrate a distributed hardware spectrum. The goal of this work is to develop and build a software infrastructure for the ubiquitous capture of and access to data that is generated and stored by a collection of devices.

Some of the challenges presented in this work include the data intensive computing of emerging applications. Ramachandran expects applications of the future to primarily consist of stream-oriented processing such as interactive video and audio. Another challenge consists of the computational requirements to analyze such data streams and extract content, thus implying various uses of parallelism. Last but not least, the distributed environment opens the need to dynamically reconfigure and reallocate resources.

The Stampede project `http://www.cc.gatech.edu/~rama/stampede/stampede.html`, started at Compaq CRL, addresses these challenges. The current work on ubiquitous presence builds upon Stampede. The URL for the author's webpage is `http://www.cc.gatech.edu/~rama/homepage.html`.

Hierarchical Schedulers, Performance Guarantee, and Resource Management

John Regehr and John A. Stankovic (University of Virginia)

Regehr observed that users execute many types of applications, such as audio, video, process control and defense systems on general operating systems. He also observed that CPU schedulers on general operating systems cannot appropriately schedule all types of applications.

To address the problem, Regehr and Stankovic want applications to have the capability to obtain appropriate schedulers and incorporate them into the operating system when needed. Their approach involves hierarchical schedulers, where a top-level scheduler arbitrates among lower-level schedulers. This approach presents another problem of how new schedulers are added and work among lower-level schedulers. To address the second problem, the authors note that schedulers require and provide performance guarantees. As examples, one scheduler may provide a certain percentage of the total CPU to a process or thread. Another scheduler may guarantee a proportional amount of available CPU to a process or thread. A scheduler obtains its guarantees from higher-level schedulers and provides guarantees to the processes or threads it schedules. The authors note that some scheduler hierarchies (e.g., a real-time scheduler under a time-sharing scheduler) do not work.

Regehr and Stankovic use a *resource manager* to bridge resource requests with the scheduler hierarchy. The resource manager enforces policies attached to schedulers, users, and applications. As an example, a user may assign priorities to guarantee that activities such as answering a phone line are given a higher priority than a game related activity.

Implementation of the work on Windows 2000 has already begun. For more information, see `http://www.cs.virginia.edu/~jdr8d/`

CpU: Practical Components for Systems Software

Matthew Flatt, Alastair Reid, and Jay Lepreau (University of Utah)

Component-based systems can be difficult to configure properly, and often suffer from poor performance. Flatt presented "C plus Units" (CpU) from the Flux Research Group at Utah, which aims to make the linking of C-based components easier for programmers, and to eliminate much of the overhead of modularity.

CpU components are based on a theoretical model of *units* [PLDI98], which have well-defined import and export interfaces. Such interfaces help a programmer to understand the requirements and provisions of a component. The interfaces also give CpU more information for matching components during linking and for reporting mismatches. *Compound units* group related units into a single linking entity, which enables hierarchical composition.

Preliminary results from applying CpU to the OSKit [SOSP97] are promising. Work continues in exploring the optimizations made possible by CpU linking specifications. For more information, see `http://www.cs.utah.edu/flux/oskit/`

Interweave: Object Caching Meets Software Distributed Shared Memory

Michael L. Scott, Sandhya Dwarkadas, Srinivasan Parthasarathy, Rajeev Balasubramonian, DeQing Chen, Grigorios Magklis, Athanasios Papathanasiou, Eduardo Pinheiro, Umit Rencuzogullari, Chunqiang Tang (University of Rochester)

Scott presented Interweave, a distributed system that allows a compute-intensive parallel application executing on a multiprocessor or cluster to interact with other — possibly distant — "satellite" machines. Such applications include data mining, scientific visualizations, and distributed intelligent environments.

Interweave's shared memory API allows easier application implementation than application-specific message or RPC protocols. Specifically, Interweave allows a programmer to map shared segments into program components. Interweave uses a single-writer/multiple-reader protocol that associates each segment with a version number. A program-specific predicate allows the system to determine whether a cached segment copy is "recent enough" to be used by a component. Consistency is maintained with a novel hashing mechanism that stores the history of a segment in a bounded space to aid invalidation of inconsistent copies. Twin and diff techniques, like those used in many software DSMs, are used to track changes and update outdated copies.

Interweave employs a type system based on CORBA IDL for interactions between heterogeneous machines. When a segment is shared between components, Interweave automatically converts the segment to a standard wire format, remaps pointer data, and even allows programs to reorganize dynamically allocated data within a segment for spatial locality.

Interweave merges and builds upon the Cashmere and InterAct projects. It combines hardware coherence within multiprocessors, Cashmere-style lazy release consistency within tightly coupled clusters, and version-based consistency for distributed shared segments. A preliminary implementation is currently running on an AlphaServer cluster. For more information, see `http://www.cs.rochester.edu/u/scott/interweave/`

Feedback Control Real-Time Scheduling: Support for Performance Guarantees in Unpredictable Environments

Chenyang Lu, John A. Stankovic, Tarek Abdelzaher, Sang H. Son, and Gang Tao (University of Virginia)

Chenyang Lu presented this work which focuses on soft real-time systems in unpredictable environments, such as Web servers that would like to provide response time guarantees for applications such as business transactions. Lu noted that their approach was grounded in control theory and described an algorithm that uses feedback to keep deadline miss ratios acceptably low.

For more information, see `http://www.cs.virginia.edu/~cl7v/fcs.html`

Puppeteer: Component-Based Adaptation for Mobile Computing

Eyal de Lara, Dan Wallach, and Willy Zwaenepoel (Rice University)

Dan Wallach described a vision of the future in which users will be running desktop productivity applications to edit large files but will have low-bandwidth connectivity. Users will want to save edits to a safe, stable "home" file system over these slow links. Pupeteer provides a proxy-based system that uses existing hooks to adapt applications to these needs at the component level. Wallach described an analysis of PowerPoint files showing larger files are largely composed of images, which could be compressed for a significant space and time savings.

For more information, see `http://www.cs.rice.edu/~delara/talks/`

TACT: Tunable Availability and Consistency Tradeoffs for Replicated Internet Services

Haifeng Yu (Duke University)

Haifeng Yu began his work-in-progress presentation by arguing that replication is important for the Web, but replication of dynamic content is difficult because it is hard to achieve both consistency and availability. He described TACT as a toolkit that defines a consistency metric and provides a knob to allow applications to choose arbitrary points in the availability/consistency spectrum.

For more information, see `http://www.cs.duke.edu/ari/issg/TACT/`

Developing Correct and Efficient Multithreaded Programs with Thread-Specific Data and a Partial Evaluator

Yasushi Shinjo (University of Tsukuba) and Calton Pu (Georgia Institute of Technology)

Yasushi Shinjo noted that, although thread-specific data (TSD) is appealing because it requires no synchronization, is easy to use, and is fast, TSD is expensive in the POSIX model because it requires function calls. He described an approach where programmers focus on developing a correct program and the system employs partial evaluation and runtime specialization in order to convert TSD function calls into a few instructions.

For more information, see `http://www.hlla.is.tsukuba.ac.jp/~yas/papers/`

Representation and Evaluation of Security Policies

Tatyana Ryutov and Clifford Neuman (University of Southern California Information Sciences Institute)

Tatyana Ryutov said that the motivation for this work-in-progress was to be able to integrate security services at the application layer, providing conditional and extensible policies, EACLS, allowing users to integrate many security mechanisms. She described their Generic Authorization and Access-control API (GAA API) that allows applications to use this security infrastructure to implement security policies. For more information, see `http://www.isi.edu/gost/info/gaa_api.html`

Network Level Framing: Speeding Up a Multimedia Storage Server

Pål Halvorsen (University of Oslo)

Pål Halvorsen described the design of a multimedia storage server that aimed to reduce communication protocol overhead using techniques such as storing protocol headers on disk along with the data.

TRIAD

David Cheriton (Stanford University)

In a hilarious but fairly compelling introduction to a mile-a-minute talk, David Cheriton explained why "IPV6 is dead." He then presented his nomination for a replacement, "TRIAD": a new Internet addressing and routing architecture. The design philosophy of TRIAD is to build on systems that work well and are widely deployed, namely IPv4 and DNS. In TRIAD, all identification is based on DNS names. Names are temporarily translated into IPv4 addresses for local use in forwarding, but DNS names are the only visible, end-to-end, persistent identifiers. Cheriton described this scheme as having several advantages: there is no state in relays (allowing arbitrary scaling), end-to-end semantics are preserved, and backbone core routers do not need to be changed.

Towards Benchmarks for Maintainability, Availability and Growth/Evolution (MAGE)

Aaron Brown (University of California, Berkeley)

Aaron Brown started out by noting that benchmarks shape a field. Whereas most benchmarks today measure performance, many of today's and tomorrow's real-world problems are not strictly problems of performance, but rather problems of maintainability, availability, and ease of growing and evolving a system. He described a methodology for benchmarking availability that leverages existing performance benchmarks, combining them with fault injection, reporting results both graphically and numerically. He described future plans to add maintainability and growth/evolution benchmarks, noting that these would be more difficult because they involve more human factors issues.

For more information, see `http://iram.cs.berkeley.edu/istore/`

Poster Session

Summarized by Rajeev Balasubramonian (Univ. of Rochester) and Sai Susarla (Univ. of Utah)

Automatic Generation of I/O Prefetching Hints Through Speculative Execution

Fay Chang and Garth Gibson (Carnegie Mellon University)

The work tries to target the I/O latency bottleneck by speculatively issuing a prefetch request. Usually, processors remain idle while waiting for an I/O request to return. The idea here is to use this idle time to speculatively continue to execute code to detect future I/O accesses. This detection helps prefetch data so that the effective latency is reduced. The speculative execution is effected by using a binary modification tool.

An earlier naive approach (published in OSDI '99), that always attempted speculation, yielded significant performance benefits in all but one of the benchmarks used. But this approach fell well short of a manual smart modification of the code. Ongoing work has looked at optimizations to make this speculation more self-aware, i.e., use history information to do speculation only when benefits are likely.

For more information, see `http://www.pdl.cs.cmu.edu/TIP/spechint.html`

Distributed Shared Memory for Large Computing Clusters Based on Memory-Mapped Networks

Emmanuel Cecchet (INPG/INRIA — SIRAC Laboratory)

The poster described initial observations in designing a large-scale high performance cluster computing server. The cluster is comprised of a number of PCs interconnected by a Scalable Coherent Interface (SCI) network and uses a distributed shared memory (DSM) system called SciFS. This DSM is built atop the SciOS operating system and exploits the low-latency memory-mapped features of SCI to provide high performance. The DSM system does various optimizations like remote memory swaps as opposed to disk swaps.

A SCI network is organized as a ring. Initial experiments showed that the latency numbers are quite poor for rings of size more than 8. Hence, the network was configured as clusters of small rings with a switched interconnection. This has much better scalability and allows for DSM optimizations that attempt locality within a cluster.

For more information, see `http://sci-serv.inrialpes.fr/~`,`~http://sirac.inrialpes.fr/`

PASIS: Perpetually Available and Secure Information Systems

Han Kiliccote (Carnegie Mellon University)

PASIS is an innovative framework for demonstrating perpetually available information systems that guarantee the survivability of information under malicious attacks or system component failures. PASIS is based on a novel architecture which breaks all information into "chunks" and distributes these "information chunks" in novel ways by using information replication and dispersal methods. This enables PASIS to not have any single point of failure (i.e., it is not possible to destroy the information in PASIS or to degrade the performance, by eliminating or capturing few selected components or information chunks within the system) and thereby achieve a very high degree of security and resiliency against failures and attacks.

The INSTANCE Project: Operating System Enhancements to Support Multimedia Servers

Pål Halvorsen, Thomas Plagemann, and Vera Goebel (University of Oslo)

The project targets multimedia-on-demand servers. Retrieval of data from multimedia storage servers is a major bottleneck and the INSTANCE project attempts to identify and eliminate these bottlenecks. The poster described three such optimizations — network level framing, integrated error management and zero-copy-one-copy memory architecture — to alleviate the problem.

In network level framing, the server stores the multimedia packets in frame format and directly sends these when the clients demand it. This removes the overhead of encoding and decoding the data into frames and lends the abstraction of a network router to the server. Error encoding is also removed by saving the parity information. This is passed along to the client and the error-checking is now done on the client side.

For more information, see `http://confman.unik.no/~paalh/instance/`

Automatic Function Placement in Distributed Storage Systems

Khalil Amiri, David Petrou, Greg Ganger, and Garth Gibson (Carnegie Mellon University)

Traditional distributed systems statically partition their functions across clients and servers. Dynamic changes in the load render this partition suboptimal. The poster described a prototype targeting data-intensive applications, that dynamically migrated functions across the cluster.

The system, ABACUS, is comprised of a programming model and a run-time system. The programming model makes the programmer split the application into independent objects. Methods are provided to checkpoint and restore objects during migration. The runtime system takes care of resource monitoring and the migration of objects. Preliminary results have shown a great improvement in response times. For more information, see `http://www.cs.cmu.edu/~amiri/abacus.html`

Differentiated QoS Through Quality Aware Transformation of Web Content

Surendar Chandra, Carla Schlatter Ellis, and Amin Vahdat (Duke University)

The main aim here is to manage bandwidth for web servers. It does this by using quality-aware image transcoding for multimedia objects, i.e., it provides different QoS for different requests. Transcoding is a transformation used to convert a multimedia object from one form to another.

For transcoding to be useful, the underlying tradeoffs have to be well understood—information quality loss, computational overhead and space savings. Thus, different variations of the same multimedia object are provided to clients based on their dynamic access patterns in an effort to manage available bandwidth. For more information, see `http://www.cs.duke.edu/~surendar/research/`

Integrating Virtual Memory With User-Level Network Communication

Joon Suan Ong, Yvonne Coady, and Michael J. Feeley (University of British Columbia)

The project studies the interaction between virtual memory and high performance network communication. The goal is to provide zero-copy user-level messaging without page pinning at either sender or receiver.

Current systems use explicit page pinning and address translation for the network interface to access user-pages. The project attempts to allow for user-level transfers between unpinned virtual memories by maintaining shadow page tables on the NI. The NI synchronizes with the host during DMA transfers.

Since the NI has translation information, it reduces memory utilization and expensive system calls. For more information, see http://www.cs.ubc.ca/spider/feeley/DSG%20Web/dsg_p_netvm.html

RENS: A Framework for Rapidly Evolvable Network Services

Anurag Acharya, Maximilian Ibel, Matthias Koelsch, and Michael Schmitt (University of California, Santa Barbara)

Given that the Internet is transforming to an infrastructure that provides a range of services, there will soon be a need for lightweight network services that can be rapidly developed, easily extended, incrementally deployed and automatically operated. The poster described RENS, which is a prototype of such a service.

RENS services consist of distributed components connected by named communication channels. The communication structure is specified explicitly and the component programmer only has to worry about local state. The RENS runtime environment provides a range of services that are common to all clients—automatic instantiation, fault detection and recovery, scaling, etc. Thus, service designers need only focus on service-specific code.

For more information, see http://rens.cs.ucsb.edu/

Tuplink: A Communication System for PDAs and Micro-Devices

Yasushi Negishi (IBM Research, Tokyo Laboratory)

The main requirements of communication systems for PDAs include: supporting disconnected operations to minimize the use of costly links, working with limited resources, using low-quality links. Tuplink is a communication system that supports one-to-one communication and is based on Linda. There are two pools at the ends of the communication path and Linda operations are used for handling them. Synchronization operations are used to make the two pools consistent.

The Sombrero Distributed Single Address Space Operating System

Donald Miller and Alan Skousen (Arizona State University)

The key idea here is to provide a flat networkwide peer level architecture for user and system programs. The operating system provides a large distributed single address space, allowing access to the same addresses to multiple users across the network, while simultaneously enforcing protection.

The prototype uses Alpha 21164 based NT systems. The provision of object-grained and inter-thread protection requires hardware support for which modifications to the processor architecture have been proposed. For more information, see http://www.eas.asu.edu/~sasos/

Persistent Distributed Data Structures to Simplify Cluster-Based Internet Services

Steven D. Gribble, Eric A. Brewer, David Culler, and Joseph M. Hellerstein (University of California, Berkeley)

This project explores the use of a library of a few well-known distributed data structures (e.g., hash table, tree, log) as the base on which to build cluster-based internet applications. Apps can thus isolate themselves from the complex issues involved in developing these data structures, while reaping all the availability, fault-tolerance and performance benefits of a cluster-based implementation. Internally, these data structures (hash table) are carefully coded from primitive single-machine data structures called

"bricks" (e.g., by partitioning a distributed hash table among nodes in the cluster, replicating them across nodes in the cluster etc.). They use Java as the implementation language for these data structures, and employ an event-driven runtime model instead of a thread-per-task model.

Question: How do you partition the data structures among nodes, as a simple-minded page-based partitioning won't work? Answer: We partition them into multiple sub-hash tables, not pages.

For more information, see `http://ninja.cs.berkeley.edu/`

FiST: A Language for Stackable File Systems

Erez Zadok and Jason Nieh (Columbia University)

They propose a new language, FiST, to describe stackable file systems. FiST uses operations common to file system interfaces (VFS). From a single description, FiST's compiler produces file system modules for multiple platforms. The generated code handles many kernel details, freeing developers to concentrate on the main issues of their file systems. FiST uses Yacc-like grammar to describe file system extensions. He says, using FiST, it's possible to develop an Elephant File System-like file system very quickly with almost the same robustness and performance as the underlying ext2fs file system, instead of reimplementing it from scratch like the Elephant people did. FiST abstracts out the common aspects of vnodes and other FS-related data structures across a wide-variety of OSes and enables them to be manipulated in a platform-independent manner, while automatically generating code to handle platform-dependencies as much as possible. For example, FiST provides pre-call and post-call processing support for all VFS-calls.

FiST also lets you apply some operations to sets of file system operations such as all of those that change state (e.g., unlink or write), or all of those that do not change state (e.g., readlink or read). In addition, you can also refer to sets of functions that apply only to file names or to file data. This offers a convenient and concise method of affecting change in many functions at once. For more information, see `http://www.cs.columbia.edu/~ezk/research/fist-lang/index.html`

High-Performance Cluster-Based Internet Servers

Eric Jul, Povl Koch, Jørgen S. Hansen, Michael Svendsen, Kim Henriksen, Kenn Nielsen, and Mads Dydensborg (University of Copenhagen)

The goal of this project is to combine the theory of DSM systems with database technology to build a distributed database solution that can be used by search engines and E-commerce systems where access latency and availability are important and where updates to data should be reflected in the system as soon as possible. This project focus is on centralised cluster-based Internet servers with very large data sets — normally terabytes of data. The system uses the SCI technology as communication backbone, a technology that allows nodes to map memory regions of other nodes into their local address space. The SCI network handles consistency of the memory regions on the nodes.

Restricted Delegation: Seamlessly Spanning Administrative Boundaries

Jon Howell and David Kotz (Dartmouth College)

Restricted delegation enables flexible administrative boundaries. Conventional systems assume a hierarchy of administrative control, and thus cannot express non-hierarchical trust relationships. Restricted delegation, on the other hand, can model both hierarchy as well as arbitrary trust graphs. They have operators for conjunction (e.g., of capabilities of both A and B), super-imposing restriction (e.g., apply restriction R to further restrict capability A). They also have the ability to defer access control decisions

to the ultimate resource server. For more information, see http://www.cs.dartmouth.edu/~jonh/research/delegation/

Distributed, Flexible Memory Management in an Operating System Supporting Quality of Service

Ian McDonald (University of Glasgow)

The virtual memory management policies provided by traditional OSes is too rigid and doesn't support app-specific quality-of-service requirements well. However, writing complete user-level virtual memory management systems is very hard, even if the OS provides support for app-specific policies.

A flexible, extensible, distributed memory management hierarchy has been built that provides application developers with the ability to specify sub-components that would best meet the application's needs. This new hierarchy utilises compressed caching, pre-fetching from the local disk, and memory on remote hosts. Each component of the hierarchy has a strongly typed interface allowing developers to replace individual components with their own implementation without the need for recompilation.

This is Implemented in the Nemesis operating system, which provides user-level VMM support. Currently working on extending the capabilities of this new hierarchy to provide a high-level QoS interface that allows the app-developer to specify the level of page fault handling performance, leaving the construction of the relative individual sub-components to the VM system. For more information, see http://www.dcs.gla.ac.uk/people/personal/ian/research/

A Unix-Like Personality Supporting Quality-of-Service

Rolf Neugebauer (University of Glasgow)

They developed a Unix-like run-time environment for the Nemesis operating system. Challenges addressed: i) providing Unix-style linkage in a Single Address-Space Operating System (SASOS), and ii) designing the personality so that no unwanted interactions between different processes occur.

Nemesis is a library-based OS with support for QoS for all shared resources in the system. Hence the Unix personality is implemented entirely as a set of shared library-based components (like Exokernel) avoiding any shared state. Unix-style linkage is simulated using small veneer libraries, which reedirect standard Unix function calls to a closure based implementation. This approach allows them to tailor the run-time environments of processes to their needs—an important feature for embedded systems.

For more information, see http://www.dcs.gla.ac.uk/~neugebar/

Suez: High-Performance Real-Time IP Router

Prashant Pradhan and Anindya Neogi (State University of New York at Stony Brook)

"Suez" is a high-performance IP router that is built from off-the-shelf Intel hardware and Gbit/sec system-area network technology from Myrinet. Configuration: 8-node Pentium-2/300MHz interconnected by a Myrinet switch, exposing 16 100-Mbps Fast Ethernet ports.

Highlights include: i) Separate processing pipeline for realtime and non-realtime flows; ii) maps network addresses to virtual address; therefore can use CPU cache hierarchy for fast routing table lookup. iii) Suez can route packets on a flow-by-flow basis by reusing routing table lookup effort for long connections.

For more information, see http://www.ecsl.cs.sunysb.edu/suez.html