# Digest of Proceedings
# Seventh IEEE Workshop on Hot Topics in Operating Systems
# March 29-30 1999, Rio Rico, AZ

M. Satyanarayanan
Program Chair

## Introduction

The Seventh IEEE Workshop on Hot Topics in Operating Systems was held on March 29-30 1999 at the Rio Rico Resort & Country Club, south of Tucson, Arizona. The General Chair, Peter Druschel, and the Local Arrangements Chair, John Hartman, had gone to considerable effort to make the operation of the workshop smooth and pleasant for the participants. The secluded desert locale, the effect of brilliant sunshine and blue skies on winter-jaded northerners, and the enthusiasm and energy of the participants all combined to offer an ideal setting for two days of technical exchange and out-of-the-box thinking. A variety of outdoor activities awaited participants who arrived early, and a kickoff reception on Sunday evening enabled participants to get to know each other before the technical meetings began.

What follows is a summary of the discussions that took place during the workshop. It is meant to be a supplement, not a substitute, for the papers in the proceedings. Rather than producing a verbatim transcript, I've tried to focus on those interactions that seemed most insightful, controversial or evoked most response from the audience. Any such report must, by its very nature, be subjective. I've tried hard to be objective, but I'm sure that are some places where my personal biases show through. My apologies in advance if you attended the workshop and your favorite comment, question or discussion isn't mentioned here. This material is based on detailed notes taken by four student volunteers (Gretta Bartels, Buck Krasic, Rajeev Balasubramonian, and Ben Fleis). They did an excellent job of taking detailed and complete notes: any errors or omissions in this digest are my responsibility, not theirs.

## Session 1: File Systems

After opening remarks by the General and Program chairs, the IEEE TCOS award for best student contribution was given to Douglas Santry of the University of British Columbia. He was the lead author of the paper entitled "Elephant: The File System that Never Forgets".

Santry et al's paper was presented as the first of three in the session on file systems chaired by Mary Baker. The key insight in this paper is the recognition that the growth of disk capacity has been exceeding the growth rate of read-write user data for some time now, and this trend shows no signs of abating. The paper describes a way of translating this spare disk capacity into improved system usability by separating file operations from file retention policy. The result is a file system that automatically saves all versions of all files for the short term, provides user control over long-term version retention, and uses heuristics to guide retention in the absence of explicit user advice.

In the discussions that followed, Margo Seltzer asked whether directories were grow-only, and whether repeated recompilations would result in unwieldy object-code directories. Santry replied that this is indeed the case in the current implementation, although a refinement to move dead names into a separate directory segment is under consideration. The cost/benefit tradeoff of this refinement will not become clear until adequate usage experience is available. In response to a question from David Mosberger, Santry confirmed that Elephant could obviate the need for RCS and ad hoc tools for checkpointing. Satya asked whether Elephant provided facilities for applications, rather than end users, to guide retention policy. Santry replied that there were currently no specific provisions for this purpose, but agreed that application assistance could prove valuable. Responding to a question from Dickon Reed, Santry explained that renaming did not complicate the tracking of file versions: a rename operation was represented as an unlink in one directory and a hard link in the other.

Peter Chen asked a series of questions probing the performance impact of Elephant's retention strategy. Santry responded that the implementation made heavy use of log-structured file system techniques, that buffer cache usage would indeed increase because of the presence of multiple versions, and that each version of a file corresponded to a separate vnode in the system. Mitchell Tsai asked how user behavior was affected by a system like Elephant. Santry replied that it should make users less paranoid about the safety of their data, but that only a user-study could confirm this. Tsai then asked how Elephant handled cases where an application like Emacs created a new file rather than overwriting the original. Santry agreed that this was a problem for the current system, and that support for managing related groups of files has to be an important future capability. In response to a final question from Mary Baker, Santry explained that a user study was being planned and that it was going to involve both qualitative and quantitative aspects.

The second paper of the session, presented by Karin Petersen, explored the caching issues induced by the design of the Placeless Document System being built at Xerox PARC. This system enables per-user, dynamic customization of documents, allows sharing of customizations between users, and cleanly separates the bit-storage and customization layers. Maintaining the validity of cached copies of a customization is done through pieces of code called Verifiers and Notifiers. These are unique to each customization and serve, respectively, to re-validate a cached copy and to trigger a callback.

Armando Fox opened the question period by asking how naming was linked to customizations; for example, one doesn't wish to remember five totally different filenames corresponding to five formats of a document. Petersen replied that the example could be handled by creating an active property that created the five formats, thus requiring only a single name for that property. Mike Jones asked what kind of file system support was required for maintaining cache consistency; Petersen's response was that systems like AFS that provide callback-based consistency are particularly helpful, and that systems like NFS force reliance on heuristics such as file modification times.

Jon Howell asked how systems like emacs which changed file names were handled. Petersen replied that many dirty tricks have to be played in such situations to keep related items together, and that programs like Microsoft Word are even more complex to handle. To Satya's question regarding support for sharing, Petersen replied that the document space is partitioned into two levels, personal and universal; users can create links (similar to symbolic links) in their personal space to existing documents at either level. Jeff Chase asked how one installed active properties; Petersen replied that this is currently done using a modified browser, but that approaches requiring less human intervention will be needed in the future. In response to Mary Baker's question about how collections were identified, Petersen replied that a collection is a query over a document space plus explicit inclusions and exclusions.

Tom Kroeger presented the final paper in this session, "The Case for Adaptive File Access Modeling". Using long-term file reference traces from the Coda project, the paper compared the relative merits of a variety of approaches to predicting file accesses. The results show that Finite Multi-Order Context Models have the best predictive power, and that Partitioned Context Models come close, while incurring much less space cost.

Margo Seltzer opened the question period by complimenting the speaker on the completeness of the work. She then asked if there were any similarities between file access prediction and branch prediction in computer architecture; Kroeger replied that he hadn't done a deep enough study of the two problem domains to comment on that. Seltzer then asked whether improved predictive ability translated into substantially improved performance. In response, Kroeger pointed out that only a real implementation could provide this information, and that such an implementation was indeed being planned. He added that he did expect partitioned context models to offer a substantial performance improvement over last successor models, since they are able to eliminate nearly a third of the misses sustained by the latter.

## Session 2: Great Expectations

The second session, chaired by Jeff Chase, consisted of three papers reporting on the early stages of projects with ambitious goals — hence the session title "Great Expectations". The first paper, presented by Aaron Brown, described the goals of a new storage system called ISTORE being built at UC Berkeley. Driven by the belief that maintainability, availability and scalability will dominate raw performance in importance, this project seeks to build a system that is extensible and self-tuning. The scope of the project is very broad, spanning hardware, language support, operating system and application components.

Opening the question session, Margo Seltzer expressed skepticism that a designer could anticipate the full range of storage management policies likely to be needed by diverse user communities. Brown replied that the policies in ISTORE were management policies not performance policies, and that these spanned a narrower range because system administrators typically have good intuition of what they need in this regard. Rick Schlichting then asked two related questions regarding changes to policies: whether the adaptive policies could themselves be adapted, and if so, how they were distributed and deployed in a consistent manner through the system. Brown responded that they were still grappling with these issues, and that language-level support in form of triggers, views and mobile code were likely to prove useful for this purpose. Milan Milenkovic observed that the architecture of the system suggested the possibility of moving functionality, such as file system support, directly on to intelligent disk controllers. Brown replied that this was indeed the intent.

Following up on the response to Margo Seltzer's earlier question, Karin Petersen disagreed that characterizing failures was easier than characterizing performance, particularly in a wide-area distributed system like the Web. In response, Brown said the range of failures was relatively narrow since ISTORE was focused on systems that fit within a single rack rather than those distributed over a wide area network. Armando Fox asked who would watch the watchers; in other words, how could self-adjusting code itself be monitored? Brown's reply was that a small, well-trusted diagnostic core would be responsible, but that this was an area still being conceptualized. The final question, from Hari Balakrishnan, observed that the Internet could be viewed as a highly scalable self-adjusting system, and asked whether any thought had been given to exchanging ideas from that domain. Brown replied that they planned to look at wide-area algorithms in the future, but had not done so yet.

The second paper in this session, entitled "OS Support for General Purpose Routers", was presented by Larry Peterson. He described the work as being in the "incubation state", with the goal of designing router software that allows easy customization. The work differs from the active networks effort in that it also aims to customize edge routers at the trust and control boundary between one's local network and the rest of the Internet. In contrast, active networks imply customization of all routers in an end-to-end path. Current commercial routers levy such a high performance penalty on any packets which require special handling that customization is not attractive. This work strives to preserve high performance on the fast path while reducing the performance penalty for customization.

David Patterson asked what components of the proposed routers could be obtained off the shelf. Peterson replied that he expected all the components to be available off the shelf in about 3 years. Patterson then asked why a special-purpose router was necessary, and whether a general-purpose workstation would suffice. Peterson's reply was that bus bandwidth and expansion limitations were the primary reason for needing special-purpose hardware. Hari Balakrishnan did a quick back-of-the-envelope calculation to derive an aggregate Internet demand of about 10-20 Gb/s, and asked how this demand was going to be met. Peterson's reply was that local area traffic demand from applications such as high-end graphics was likely to be even higher, at about 200 Gb/s. The only way he saw of meeting this high demand was by careful and well-planned engineering. Dickon Reed requested clarification regarding which routers in the Internet were likely to use the proposed system. In reply, Peterson said that this system was meant only for use in edge routers; interior nodes in the Internet were likely to remain unchanged. Jeff Chase asked how many cycles per packet could be spared for customization. Peterson replied that discovering this quantity through experimentation was one of the goals of the project.

The final paper in the session was presented by Mark Yarvis. This paper describes an architecture for adaptive middleware called "Conductor", whose goal is allow easy customization of adaptation strategies without requiring modifications to applications. The idea is analogous to stackable file systems, which allow customization of file system internals while preserving application transparency. A given adaptation typically, but not necessarily, consists of a pair of adaptation modules, such as compression/decompression and encryption/decryption. Conductor allows such adaptations to be composed, with individual adaptation modules distributed into the network at Conductor-enabled nodes.

The first question, by David Kidston, was how Conductor ensured that data streams passed through the proxies representing adaptation modules. Yarvis replied that they used split TCP connections. Kidston then asked if this didn't violate end-to-end TCP semantics. Yarvis replied that Conductor provides an end-to-end reliability model on top, which allows adaptation modules to preserve end-to-end semantics if they choose to. Doug Terry followed with two questions: who performed the planning for adaptation, and whether adaptors could themselves adapt to changing conditions. Yarvis responded that installing an adaptor is currently a heavyweight operation, and therefore it makes sense to build adaptors that have the ability to respond to changes in the environment. Hari Balakrishnan observed that this approach was similar in some ways to application-level framing, and asked whether adaptors could perform retransmissions at the level of segments meaningful to applications. Yarvis replied that this was difficult to do because adaptors are transparent to applications.

## Session 3: Networking

Rich Draves chaired the third session, focused on networking. The first paper in this session, presented by Hariharan Rahul, described an approach to congestion management that centralizes currently-fragmented knowledge of networking conditions on various flows. The component responsible for maintaining this knowledge, called the Congestion Manager (CM), exports an API that enables applications to make queries and place callbacks triggered by changes in network conditions.

Buck Krasic asked the first question: does involving applications in congestion control increase CPU scheduling delays and thereby reduce agility of adaptation? Rahul replied that the CM was in the kernel, just like the rest of TCP, and so should incur little additional scheduling overhead. However, he did agree that the results in the paper were based on simulations and that only an actual implementation could reveal the incidental overheads of this approach. Peter Druschel asked whether the CM did policing to detect uncooperative or malicious applications. The reply clarified that the CM API assumed cooperative applications, and that policing would require the assistance of edge routers. Rich Draves expressed concern that the greater number of system calls due to increased application-kernel interactions would degrade performance. Rahul replied that buffering could reduce some of this overhead, and that the improved programming model offered to applications would more than compensate for the increased overhead. He also noted that if reducing context switching overhead became critical, as in a Web server, the CM could be implemented in the kernel along with the transport protocol.

The second paper, entitled "The Case for Informed Transport Protocols", was presented by Stefan Savage. In a hilarious introduction that won applause from the audience, Savage characterized early use of the Internet as two graduate students downloading the BSD Unix source tree; he then contrasted this with the current state of the Internet, characterized by every graduate student frantically performing short e-trades. The main point of his talk was that TCP's congestion control mechanisms were designed with a certain set of assumptions that no longer hold. His proposed solution is to allow different machines to share information about network resources they use in common. This approach allows TCP sessions with short lifetimes to make informed decisions about the correct rate for sending traffic.

Robert Haas began the question period by observing that Savage seemed to be making the case for application-specific transport protocols, not informed transport protocols. Savage's reply was that there was a spectrum of possibilities ranging from a purely end-to-end approach to an ATM-like approach where the network has full awareness; informed protocols support a broad range of possibilities in this spectrum.

This was followed by an extended interaction between Hari Balakrishnan and Savage. Balakrishnan pointed out that one of the scariest problems is figuring out how to age information. He disagreed with the proposed aging value of 10 seconds, and suggested that this number should be tightly coupled to the round trip time (RTT) since the state of congestion changes at that scale. Savage countered that Balakrishnan's own work showed that unless you examine fine-scale behavior, traffic behavior does not change that quickly. Therefore, variations below 10 seconds are not something you can control for. Further, TCP's existing congestion control mechanism already adapts at the time scale of RTT. Balakrishnan then pointed out that the factor of 2 by which TCP shrinks its window on congestion is not arbitrary, but is because you know during slow start that half the current window worked the last time. Savage agreed, but noted that the choice of 2 as the growth factor during slow start was an arbitrary choice proposed by Van Jacobson in his seminal TCP congestion control paper.

Hariharan Rahul asked how much overhead would be imposed by the sharing of control information in informed protocols. Savage's response was that this would depend on the implementation. If the information is piggybacked, then it would impose minimal overhead; but if additional packets are generated for pacing by the sender or receiver, that might pose some overhead. Rahul followed by asking if it was being assumed that there was adequate bandwidth in the network; Savage confirmed that this was the case, but added that it wasn't a big deal in modern networks.

Ian Pratt pointed out that the paper showed how an informed protocol could be helpful to the sender, but did not show how it was useful to the receiver. In practice, however, it is usually a server talking to many sites that is most in need of help. Savage agreed that servers suffered most from bottlenecks, but felt that they could also benefit from informed protocols through indirect control of traffic rates. David Mosberger closed the question session by requesting clarification on an apparent paradox: on the one hand, the talk claimed that a single flow is not long enough to get useful congestion information; on the other hand, the probability of having two or three flows close together within 30 seconds in a bottlenecked situation is low. Savage agreed in principle, but insisted that using older estimates under those circumstances was better than picking a random estimate for a new flow.

David Kidston presented the final paper in the session, entitled "Transparent Communication Management in Wireless Networks". The goal of this work was to improve application performance in wireless environments in a transparent manner, using proxies to intercept and modify network flows. An environment manager, running on each node, collects and shares information on network conditions with its counterparts on other nodes. The transformations of a proxy are guided by information from its environment manager.

Stefan Savage began the question period by noting that his work avoided proxies for two reasons: they introduce single points of failure, and they modify end-to-end RTTs in ways that mislead TCP sources. Kidston rebutted the first point by observing that wireless base stations already represent single points of failure. He addressed the second point by indicating that deliberate delays could be introduced by proxies to preserve end-to-end RTTs. In response to Savage's followup question as to whether the source code for this work was ftpable, Kidston replied that he wasn't sure but would check into it.

Srini Seshan asked whether stream compression by a proxy would interfere with packet retransmissions. After some debate on this issue by Seshan and Kidston, Hari Balakrishnan offered the suggestion that retransmitting a superset of missing packets would allow stream compression without affecting end-to-end packet boundaries. Balakrishnan then followed by asking why a simple split-TCP approach was not being used. Kidston's response was that the proxy approach allowed modification of both packet headers and contents, while conventional split stream only allowed modification of contents. The session ended by Armando Fox asking about the kinds of applications this work was directed at, and Kidston replying that MPEG video was a typical target application.

## Session 4a: Lies, Damn Lies and Benchmarks

The fourth session consisted of two mini-panels. The first of these, chaired by Mary Baker, focused on the issue of benchmarking. Each of the three panelists, John Regehr, Jeff Mogul, and Margo Seltzer, gave a brief opening statement. After these statements, the floor was opened for questions from the audience as well as panelists.

John Regehr described the perverse effects that benchmarks can have on a system. In a case study involving a soft real-time scheduler for Windows NT, certain performance bottlenecks were discovered in device drivers. These had been introduced by the device manufacturer in an attempt to improve performance on well-publicized benchmarks, but had unfortunate side effects. Once one device manufacturer went down this path, competitors had little choice but to follow suit so that they did not suffer in benchmarks.

Jeff Mogul explored the purpose of benchmarks, and distinguished between reproducibility and predictive power. He deplored the tendency of academic researchers to focus on reproducibility, saying that it gave the illusion of scientific precision, but failed to be useful. He argued for a shift in focus to more realistic benchmarks, whose results could better predict real-world performance.

Margo Seltzer was the perfect follow-on to Mogul, since she described a methodology for benchmarking that addressed exactly the shortcomings exposed by him. In this methodology, overall system performance is obtained as the product of a system vector that captures workload intensity and mix, and an application vector, that characterizes the performance of each application in isolation. The elements of the application vector are obtained through microbenchmarks. The system vector can be set to represent any existing or anticipated workload.

The floor was then opened for questions. Dave Patterson directed the first question to Seltzer, asking whether the micro-benchmarks in her methodology had proved useful in capacity planning. She replied that this was indeed the case: they had obtained stunningly good results for Web servers by recording their behavior for 3 months and then using this data to predict 12 months into the future. She acknowledged, however, that she did not yet have such good results for OS benchmarks. Patterson then made the observation that the OS community should start working on other problems besides performance — this needs leadership. Mogul concurred, but observed that there was often a price premium for performance, and that vendors often had to demonstrate improved performance to counter the propaganda of competitors. Patterson repeated his plea for the OS community to broaden its vision, and pointed to system reliability as an area crying out for quantitative study and characterization.

A different line of questioning was initiated by Godmar Back, who asked whether Seltzer's methodology applied to rapidly evolving systems. He observed that the academic research cycle is typically 2-3 years long, and that benchmarking done on an early version of the system rarely reflects the performance of a later version. This triggered an extended debate involving Back, Seltzer and Karin Petersen. Seltzer's position was that program committees of conferences should be encouraged to accept papers that revisit performance measurements reported earlier. Petersen observed that this alone would not be adequate: authors should be required to clearly identify the causes for performance changes in a way that would give her confidence that their tweaks would work in her system. Seltzer countered that the use of a system vector in her methodology served just that goal: by comparing system vectors, one can determine whether the workload reported in a publication matches another system's workload closely enough to expect portabilty of results.

Kim Keeton then shifted attention to industry standard benchmarks. She observed that the TPC benchmarks for transaction processing were hard to use. They are complex, require a large amount of hardware, and involve numerous configuration and scaling parameters. In addition their use on specific systems is typically conditional upon limited disclosure of results — a restriction known as the "DeWitt clause". Mogul concurred with Keeton, and suggested that the 189 parameters in the TPC benchmark indicated a strong need for self-tuning benchmarks. In defense of TPC, however, he observed that comparisons based on it were a reasonable reflection of real-world performance.

The discussion then shifted to evolution of benchmarks. Satya observed that once a benchmark becomes widely-used, benchmark-resistant strains of systems evolve. This is very much like the dialectic between viruses and immune systems in biology. Perhaps we should borrow a page from that domain and randomly mutate our benchmarks from time to time. Seltzer responded that her application-specific benchmarking methodology was well-suited to this — you just had to plug in a different system vector to reflect a changed benchmark. Satya expressed concern that the methodology seemed brittle in a different way: if an application changes, the old application vector may no longer hold. To this, Seltzer replied that you receive an application vector with each new release of that application. Regehr remarked that benchmark mutation requires an ongoing investment of time to stay ahead, to which Satya replied that one should accept this as part of the normal cost of developing a benchmark. This requires us to be proactive, and think like a virus, not the immune system. Mogul expressed skepticism about keeping benchmark mutations secret, observing that security through obscurity has never worked. Satya disagreed, drawing an analogy with examinations where last year's version is available ahead of time but this year's is a secret until it is given. Seltzer closed this line of discussion by observing that mutating the benchmark would not, in any event, have helped in the specific example that Regehr had described.

The panel closed with a question from Dickon Reed about the size of a typical benchmark in Seltzer's methodology. Her reply was that it was more important to focus on identifying a system vector that was representative and finite than to worry about its size.

## Session 4b: The Thin Red Line

The second mini-panel, focusing on the protection boundary between user and kernel space (the "red line"), was chaired by Doug Terry. It began with brief presentations from the panelists. Prashant Pradhan described how the segmentation hardware on Intel platforms could be used to isolate software modules. Godmar Back explained why the language-level security provided by Java did not remove the need for kernel-user isolation, and described how such isolation could be provided. Jon Howell presented his views on how partial evaluation (PE) in a programming language can be extended to subsume operating system services.

The discussion opened with an extended interaction between the panelists. Pradhan asked Howell whether a kernel had to be rewritten to use the PE approach. Howell replied that the PE approach was currently a pie-in-the-sky vision, but that he hoped to be able to run existing code with minor changes. Back expressed doubts about the pragmatism of the PE approach, though he agreed that it was a great vision. He had doubts in his mind about the cost of specialization, about modularity, and about concurrency. With reference to Pradhan's presentation, he mentioned that he would have like to have seen this work put in the context of single address space OSs, and better justification of the assumption that there is little cross-boundary communication. Howell agreed that PE does not eliminate the need for secure APIs. He observed that APIs do not go away in his approach, but just become specification tools.

Mike Jones expressed skepticism about the value of the PE approach in practice. He noted that at the last OSDI conference, there had been a panel on the subject of OS support for languages. In that panel, Rob Pike had confessed that a fundamental mistake in Inferno had been to assume that the OS-user boundary could be eliminated; this had seriously hurt maintability because one was faced with a sea of pointers when trying to debug. On a different point, he remarked that code did not have to be modified when moved across a kernel-user boundary; systems such as Chorus had shown how this could be done. Back agreed with Pike's observation that eliminating the kernel-user boundary resulted in a big soup of data and pointers. He also agreed with Jones' observation that code could be portable across the kernel-user boundary. Howell defended the PE approach by noting that it was important to distinguish between use of the line for specification and using it at runtime. Eliminating the line could be made automatic, using techniques such as incremental recompilation.

Robert Haas posed two questions for the panelists. Of Pradhan, he asked how his work differed from the Vino approach. Of Howell, he asked how the kernel could defend itself against buggy user code, especially since many of the obvious safety tests were undecidable. Pradhan's response was that his work was similar to sandboxing but differed in that it relied on hardware rather than binary rewriting. Howell clarified that the red line need not have to coincide with the protection boundary. He agreed that PE involved self-modification of the kernel and that this posed a robustness hazard.

Drawing upon his experience with extensibility in SPIN, Stefan Savage said that he was disturbed by the characterization of the red line implicit in the panelists' presentations. The real problem, he observed, was how wide the interface became in the presence of extensibility. The Unix interface is relatively narrow, typically 172 calls that are precisely specified. Interfaces involving 1000 or more calls are very hard to manage because they involve lots of semantic properties. There is danger in thinking that memory protection solves the problem of extensibility. Howell responded by saying no one on the panel would disagree with this point. Savage followed up with the observation that none of the panelists had addressed any of the really difficult problems that SPIN had tried to tackle. In particular, none of them had suggested how to draw the many new red lines that would be implied by extensibility. Howell replied that he didn't mean to imply that PE solved those problems exposed by SPIN. His approach was meant to fit into software engineering approaches like SPIN. Savage completed his comments by agreeing with

Dave Patterson's earlier observation about the OS community having the wrong priorities.

Directing his question to Howell, Armando Fox asked whether globbing applications with runtime services leaving only a thin kernel layer didn't effectively amount to an exo-kernel approach. Howell clarified that by "globbing" he only meant to imply that PE did not require a fixed interface in relation to software engineering boundaries. An exo-kernel, in contrast, lowered this interface. Fox followed up by asking how a good compiler differs from what PE offers. Howell's response was that PE might recognize boundaries that current compilers might not. In any case, he felt that it was premature to comment on this question since this approach had not been implemented yet.

Dickon Reed asked what prevented 10,000 JVMs from running on a machine, a figure that Godmar Back had mentioned in his talk. Back replied that the technology wasn't there yet and that it took too much resources. He also added that the 10,000 figure wasn't made up but came from an Oracle example. Reed pressed this point further, saying that C took a while before large systems could be built with it and that it should only be a matter of time before Java was capable too. Back's response was that Java certainly wasn't there yet, and that the smallest unit of sharing being a page might lead to fragmentation problems that limited scalability.

The final question in the session was by Milan Milenkovic. He mentioned that there were two industry efforts to standardize Java runtime environments for CE and digital TV environments, ATSC DASE (American Television Standards Committee, Digital-TV Application Software Environment) in USA and DVB MHP (Digital Video Broadcast, Multimedia Home Platform) in Europe. Since these address embedded systems, there is no secondary storage and hence no support for virtual memory. To complicate matters, target platforms are real-time OSs with non-Java threads coexisting with Java threads. He wanted to know if Back's Java proposal offered hope for this type of situation. Back replied that he was only looking at all-Java environments at the moment, but hybrid solutions might be possible in the future.

## Outrageous Opinions

Following a longstanding HotOS tradition, the participants got together after dinner for a session of uninhibited discussions and expressions of opinion. Since it was a pleasant moonlit evening, people gathered on the lawn outside the HotOS hospitality suite, where an ample supply of libations was close at hand to liven the proceedings. Jeff Chase moderated the discussion.

Dave Patterson began by observing that he had come to HotOS to learn what was hot in operating systems, but was disappointed to see the community still fixated on performance even though it is no longer a problem. Ignoring Peter Chen's observation that there were papers on other issues like power management at the workshop, he observed that academic communities tend to get too stable in many fields, and just keep publishing and praising. From his perspective, the OS community seems in danger of settling into this comfortable middle age.

Not willing to let these remarks go unchallenged, Jeff Chase suggested that performance is the only thing we know how to measure and that is why the OS community focuses on it. He reminded Patterson of his own quote, "For better or worse, benchmarks shape a field." Patterson admitted to having made the statement, and said that in this case it was for the worse — at which the audience let out a collective groan. Jon Howell pointed out that performance benchmarks make a system that reboots quickly appear as good as a reliable system; Stefan Savage ended this avenue of discussion by saying it was too embarrassing for the community to have to admit to frequent reboots.

Continuing his role as rabble-rouser, Patterson observed that research labs are focusing on engineering these days, and rely on academics to do research. Unfortunately, the latter are often focused on near-term development, sometimes on topics that are over 20 years old. He urged the OS community to shift its focus to problems that are crying out for attention such as maintainability, usability, scalability and availability. Just because these are hard problems, they shouldn't be ignored. He pointed to fields like psychology, where researchers work on the human brain — that's hard, but they do it anyway. The OS community should stop complaining, end the focus on easy stuff like performance, and get on with doing the hard but important research.

By now, the crowd was provoked. Robert Haas pointed out that performance continues to be important because new features are only good if they are feasible. Stefan Savage added that we are not the OS community, but the performance community. As evidence that the community addressed issues other than performance, Dickon Reed mentioned that the Nemesis operating system addressed QoS. But Patterson dismissed QoS as merely an extension of performance. Responding to this cavalier dismissal, Savage pointed out that reliability could also be reduced to a performance metric — it is merely the time before you have hit the reset button. Armando Fox and Jeff Mogul, in different ways, responded to Patterson's claim that academics were doing engineering rather than science. Fox's position was that as long as academics are working on problems that matter to real-world people, who cares what label you attach to the work. Mogul's position was that we are engineers, and claiming that we are scientists is bogus.

Peter Chen tried to steer the debate along a less controversial path by suggesting that the metric for relevance should be the number of people who care about what one is doing. By that metric, he observed that performance does indeed rank low. Savage refused to be persuaded by this argument: he observed that the main complaint about the Web was that its services were too slow. Mitchell Tsai also disagreed, but for a different reason: he found that users were concerned about ease of setup, time for reboot, time for installation, and so on. These are performance metrics that determine how much of a user's time the system wastes, and people care very much about performance in this broader sense. Mogul voiced agreement, suggesting that a good metric would be to have a thousand grandmothers install Windows98 and see how many of them succeed by the next day. Patterson, to much cheering from the crowd, suggested that a better metric would be to use a thousand Microsoft executives rather than grandmothers.

A more fundamental rebuttal of Chen's viewpoint was offered by Robert Haas. He refused to accept the premise that a sign of good research is how many people on one's block care about the results. He reminded the audience that we were discussing research, not development. Chen countered this by saying that good research had to matter eventually, even if not immediately. Patterson asked what research done 20 years ago mattered to people today; can we explain it to them and will they care. Haas persisted that utility should not be required of research — one should be free to try out wild new ideas. Patterson retorted that the OS community is not full of wild ideas — just performance. Jeff Chase asked the audience what we were doing 10-15 years ago, to which there were various responses: file systems, performance, kernel organization, GUIs. Chase then pointed out the paradox that we all think we are researching OS's, yet 90++ percent of us are running Windows98, 99% or more of us have never seen its source, and most of us don't work for Microsoft!

Mitchell Tsai then posed this rhetorical question: if the OS community is so good, why didn't it predict the Web? Haas responded that we are not God, and Karin Petersen followed by observing that the Web did not arise out of the blue. Patterson asked what the OS community contributed to the Web. Tsai suggested TCP, but Patterson dismissed that saying that it was hard to think of Vint Cerf as an OS guy; Vint thought he was doing networking, and TCP is more about availability than performance. Tom Kroeger suggested that security and reliability were two areas through which the OS community could be relevant to the Web.

Patterson then turned to Satya and observed, amidst much laughter from the crowd, that he had been around a long time and asked what he saw new in the OS community. Satya returned the compliment by noting that although his hair was a lot grayer now, he hadn't lost as much of it as Patterson had over the last 20 years. On a more serious note, he pointed out that the relatively new field of mobile computing is at the forefront of OS research. The hostility of a typical mobile computing environment generates many new problems besides performance. Patterson asked if he was referring to the networking aspect, but Satya replied that it was more fundamental. Early attempts at coping with the challenges of networking have yielded ideas such as distillation of data, and degraded computation for applications like speech recognition. These are very different concepts from what one is used to on a desktop. Armando Fox suggested that many mobile computing folks would agree that good infrastructure support is the key, resulting in the same kinds of robustness and scalability problems that Patterson had alluded to earlier. Satya disagreed with such a narrow characterization of mobile computing. In his view, mobile computing demands a different way of looking at computation — an acceptable answer now is preferable to a precise answer later. This shift in viewpoint is a fertile source of interesting problems.

Rich Draves then turned the discussion to the issue of software reliability. He pointed out that software engineering issues like reliability and maintainability are ignored in the toy systems that researchers build. As a result such systems do not scale. He wondered how we could avoid this syndrome. Patterson offered a ray of hope by observing that the OS community has some of the smartest people, and that not all communities have these people. But he agreed with the seriousness of the problem, quoting John Ousterhout as saying that software was a big catastrophe, and that no one could have predicted things could possibly be this bad. Draves concurred, saying that software seems to be the Achilles heel of civilization. Patterson agreed with this characterization and added that it was perhaps we are so smart. Chase suggested the alternative possibility that perhaps this was really a hard problem. Patterson was skeptical of this, saying that after 30-40 years and all those PhDs, one would expect better. He suggested that perhaps they only picked the low-hanging fruit, and that's the problem. This suggests the need for more improvements in areas that are harder but offer bigger rewards.

Dickon Reed offered the opinion that reliability isn't really a problem: Linux works and hardly crashes even though applications may crash. Jeff Chase added that his NT box doesn't crash either, if no applications are run on it. Satya offered the suggestion that perhaps software is in such a mess because it seems so easy. Patterson followed this line of reasoning by observing that hardware culture is to test exhaustively. The "compile" step is too expensive, so repeated and thorough testing is the practice. Perhaps software would work better if it cost as much to develop as hardware. Chase characterized this point of view as saying that the compiler people had done too good a job, and Rich Draves groaned in dismay at this answer. Patterson challenged Chase to prove him wrong by identifying the thing that should be improved to make software quality routine.

Stimulated by this challenge, a chorus of voices offered suggestions. Stefan Savage suggested automated checks for memory accesses. Satya suggested restricting programmers to one compile a day so that they would think harder about the code they write. Richard Draves offered the idea of continuous sanity checks of key data structures, as used in some telephone switching software. Armando Fox suggested better support for handling out-of-band events. Patterson himself suggested better programming languages. But most of these suggestions were soon shot down. Savage reminded Patterson that at a recent OSDI keynote speech he had claimed programming language research isn't real because it is not quantifiable. Srini Seshan disagreed with Satya, saying that limiting compiles isn't the answer; instead, what is needed is the equivalent of simulation in software, which is debugging.

Sensing an opportune moment to convey a key message, Dave Patterson urged the audience to become more actively involved in ensuring support for computer science research in Washington, D.C. He pointed out that both DARPA and NSF are in critical need of program managers. Further, there is a once in 20-year opportunity to substantially increase research funding, but this requires much more active participation by the research community. Senators and Congressmen need to hear from their constituents that support for information technology is important. He urged the audience to be active in this arena, and to call or write to their government representatives.

As the discussion languished, Jeff Chase observed that the proceedings weren't outrageous enough. David Ingram stepped in to offer the outrageous opinion that all software should be required to have open source code, and to publish all interfaces. But the crowd failed to be roused by this baiting. Stefan Savage then tried a new tack. He reminded everyone that at HotOS-4 in 1993, Brian Bershad had bet Jeff Mogul a bottle of Cabernet Sauvignon that in four years time Sun would no longer be in the OS business. At HotOS-6 in 1997, he had graciously admitted defeat. But Bershad and Mogul had made a second bet at that workshop, that by Dec 31 2001, DEC would no longer be in business. Savage accused Mogul of trying to weasel out of the bet by claiming that DEC still existed because it was a wholly-owned subsidiary of Compaq. He appealed to the audience to judge whether this was fair. Mogul, however, insisted that the exact wording of the bet be examined. Fortunately, someone had a copy of the scribe record of the wording of the bet, and Dave Patterson was charged with verifying it. There were numerous jeers, boos and hisses as Mogul defended his position that DEC was still alive, mentioning for example that he still received DEC's contributions to his pension plan. Savage, on the other hand, dismissed these arguments as merely appealing to the letter of law, and urged the audience to vote their consciences.

In the midst of these appeals for votes came the shocking news that top-rated Duke had lost the NCAA basketball championship to UConn. After a brief break for drink refills, the audience gathered on the lawn again to hear closing arguments on the Bershad-Mogul bet and to vote. A show of hands supported Mogul by a modest margin, and Savage congratulated him on his victory. But Mogul graciously suggested that the person who deserved the prize was Savage himself for putting up such a good fight on behalf of his advisor, who hadn't even shown up!

The rest of the session was anti-climactic. There were numerous attempts to rouse the energy of the crowd, but to no avail. In the waning moments of the evening, a number of wild ideas were tossed out, but it was clear that the evening was over. Jeff Chase declared the Outrageous Opinions session officially closed, and people headed back to their rooms.

## Session 5: Potpourri

Many sleepy eyes and some yawns greeted Doug Terry, as he chaired the first session on Tuesday morning. However, these soon gave way to interest and attention, as three exciting papers on unusual topics were presented.

The first paper, entitled "The Case for Higher-Level Power Management" by Carla Ellis, described why operating systems should treat energy as a first-class resource. An appealing aspect of this presentation was that its motivation was based on first-hand experience. In trying to build an application called a Hiker's Buddy using a PalmPilot, the author had come to grips for the first time with trying to manage energy in a real-life situation. One of the slides in the talk, showing a stunning photograph of the author on a hike in the Olympic Peninsula with her Hiker's Buddy, won sustained applause from the audience.

Mitchell Tsai began the question session by first observing that Alexey Rudenko in his group was working on power management, but faced multiple problems. First, he needed a method for devices to wake up the computer under certain conditions but this wasn't possible on today's hardware. Second, battery life depends on how you drain it, and this is not a simple linear function. Ellis agreed that new architectural features were needed to support higher-level energy management. Tsai followed up with the observation that if total energy consumption could be kept below that obtainable from solar energy, the whole problem would go away. Ellis responded, to much laughter from the audience, that she was in the Pacific Northwest where solar energy was not a viable option.

Margo Seltzer complimented the speaker on the work, and asked why the GPS needed to talk to the Pilot for the entire three minutes, while it was trying to stabilize. Ellis replied that the duration was not always three minutes, and that the requirement was driven by the dumb design of the GPS unit. Milan Milenkovic also commended the speaker for the fine work, and observed that we need better APIs for controlling OS power consumption. He cited the Intel/Microsoft ACPI specs, and wondered how much of this information was available to applications. Ellis replied that she would have to check on this.

Peter Chen presented the second paper, entitled "Reliability Hierarchies". This paper argues that system designers should treat reliability as a property that induces a hierarchy of levels in a system, much as access time induces a memory hierarchy. This viewpoint gives the designer a conceptual tool for balancing performance and reliability. Chen elaborated on these ideas using the Rio system as a case study.

The first question, by Dave Patterson, asked whether the server at Michigan from which statistics were reported in the paper used RAID. Chen replied that it did not, and the disks were just a bunch of ordinary disks. In response to Patterson's followup question on how RAID would have affected the reliability numbers presented, Chen replied that it have improved the media failure figures but these only account for a small part of all failures. RAID would have done nothing for software crashes.

Mary Baker then triggered an extended dialog with the speaker by observing that the use of a special sync call in Rio, to get data out to disk just before a crash, is a weakness in the system. In her experience, such special features are difficult to test and debug, and hence hard to trust. Chen countered that the mechanism had been tested in thousands of crashes. Unconvinced, Baker pressed the point by observing that these crashes were deliberately induced and that real-life crashes may be messier. Chen held his ground, only conceding that more extensive testing is always useful.

Satya asked whether the server data presented in the paper was from live use of the system, or from stress tests. In particular, he wanted to know if the speaker's email was stored on the server. Chen replied that the data was not from a system in live use.

Referring to the methodology for inducing crashes, Karin Petersen asked how one could reliably capture control at the start of a crash, and whether the artificial crashes used in Rio accurately reflect how real crashes happen. For example, her NT box freezes once a day, requiring a hard reset. She couldn't see how a system like Rio could capture control in such cases. Chen replied that the current version of Rio does not exhibit freezes, though earlier versions did. He agreed, however, that nothing is perfect and that there were undoubtedly some kinds of crashes which the system would not be able to handle. In response to a followup question, Chen said that 3% of the crashes lost data stored on disk, but only 2% of them lost data stored in Rio. A related question from Jeff Chase asked how often the special crash-inducing key sequence had to be used; Chen replied that he did not have this information since he had not personally conducted the crash testing.

Persisting in Petersen's line of questioning, Ian Pratt observed that many failures on their PCs caused the hardware to lock so solidly that even the power switch was ineffective since it was only an interrupt to the BIOS. Chen replied that the Rio approach assumed that interrupt masking was not happening at the hardware level, but at the software level. If this assumption isn't true, Rio can't handle the resulting crashes. Jeff Chase added that such hard failures are typically due to buggy hardware.

Peter Chubb observed that a common failure mode of NFS was to fill a buffer with nulls rather than data; flushing that buffer to disk is the worst Rio could do. Chen observed that this was a file system failure, and that Rio could hardly be held responsible. Chubb followed up by observing that Chen was effectively describing a checkpointing approach. Chen agreed with this characterization, and added that deciding on the commit point is the hard part. Rio treats the start of a crash as the commit point; particular applications may have other notions of where the commit point should be.

A shift in perspective was offered by Bill Tetzlaff, who noted that 50 years of tape data were unreadable and hence lost, because they were recorded using obsolete hardware or software. Chen responded that in his methodology, loss rate metrics treated all data as equally important. In practice, of course, that is not right — very old data might be unimportant. One could come up with a whole family of metrics to address this limitation.

The final presentation in the session was by Mitchell Tsai, whose paper "Command Management System for Next-Generation User Input" described his experiences in using speech recognition for application control. The central lesson reported by him was the importance of striking the right balance between centralizing speech functionality as an OS component and exploiting application-specific knowledge in handling ambiguity and other language-related problems. A successful design does not have a clean layering, but is rather messy. Tsai also identified a global undo capability as an important OS feature.

Tom Kroeger began the question session by asking how one could avoid recursion in undo. Tsai replied that this was indeed a problem, but that it could be alleviated if the system recognized that it was already in an undo context. To Kroeger's follow up question on whether he had used speech recognition systems other than Microsoft's, Tsai replied that he had Dragon Dictate and many others, but depended on Microsoft for the PowerPoint application and the speech system development kit.

Mike Jones suggested that the reason for needing pervasive undo was to avoid requiring total accuracy in recognition. Tsai responded that an alternative approach would be to structure applications so that all major actions were structured as a two-phase commit, where the user is given an indication of how the application has interpreted his spoken commands before execution became irreversible. Jones followed up by asking if there was such a thing as an undo barrier, and if so, when one would use it. Tsai's reply was that this would depend on the application, and that it wasn't particularly useful with PowerPoint.

Milan Milenkovic asked for clarification on a point made during the talk, about accuracy improvement through dynamic restriction of command choices. Tsai replied that one could do that in a command and control system through a restricted grammar.

At the close of the session, Satya asked whether applications like Web browsing had been considered, and whether such read-only applications were more tolerant of speech recognition errors. Tsai replied that he had not tried that, though he was aware of other researchers who had. He observed that Microsoft's Internet Explorer already offers much better hooks for speech command and control than other browsers and Microsoft Office.

## Session 6: Is Resource Management a Solved Problem?

The next session was organized as a panel of five, with Bill Tetzlaff as session chair. The many faces of resource management represented the underlying theme of the papers represented in the panel. Each panelist gave a short presentation, followed by a brief question session. After all the panelists had made their presentations, the floor was opened for further questions and discussion.

David Ingram began by describing modifications to Linux that enable a subset of processes to be offered soft real-time guarantees. The resulting system, called Linux-SRT, is upward compatible with existing application binaries. Doug Terry asked why accounting was an important feature in a real-time system. Ingram's response was that you needed fine-grained accounting to know when to preempt things; he also observed that this was not difficult, but just involved examining clocks.

The next panelist, Dickon Reed, described a system called Xenoserver that could execute untrusted user code and provide accurate accounting of all resources used during execution of that code. Such a system could form the basis for a remote execution service in wide-area networks. In the brief question period following the talk, Satya observed that the whole notion of security and accounting was designed from the Xenoserver's point of view. He wondered what guarantees, if any, the programs executing on these servers were offered — if a program migrated to a Xenoserver in response to an advertisement that it was lightly loaded, other programs might do the same and flood the Xenoserver. Reed agreed that a distributed contract negotiation problem lay hidden in this model, but deferred mechanisms for addressing it to the next talk in the panel.

Picking up on this lead, Neil Stratford described an architecture for managing resources in systems that support QoS. The essence of the architecture is the use of frequently renegotiated timed resource contracts. Godmar Back pointed out that resources are not completely independent entities, but often depend on each other. For example, not obtaining sufficient physical memory can increase page faults to the point that the CPU, a different resource, is saturated handling them. Stratford replied that such relationships could be measured through the use of feedback, and the information used to modify resource requests. Hari Balakrishnan asked whether congestion management would be done at the same time scale as TCP, or longer; the answer was that it would be done every few seconds. Balakrishnan then offered the caution that strange interactions could occur between independent congestion control schemes, as in the case of TCP over ATM networks. Mitchell Tsai asked if the system provided support for futures; the reply was that it did not, but this was being considered.

Next, Dave Sullivan described two extensions to lottery scheduling that provide greater flexibility in resource allocation while preserving isolation. One of these extensions allows applications to modify their resource rights by exchanging resource-specific tickets with each other, yet these exchanges do not affect the resource rights of processes uninvolved in the exchange. In response to a question from Satya regarding the duration of ticket exchanges, Sullivan explained that the exchanges made by a given client are revoked when the client exits the system. Following up, Satya observed that if ticket exchanges could extend beyond the period in which a client is using the system, an entire barter system could be built up around this notion; for example, a laptop that had been disconnected for six months might reconnect and ask for all the bandwidth it could get, reminding donors of the many generosities it had shown them in the past. Srini Seshan asked how conversion rates between tickets for different resources were determined, and how gaming of the system was avoided. The reply was that this was a good question that had been thought about, but was not resolved yet. In response to Peter Druschel's question about what happened to unused resources, Sullivan replied that a client's tickets are deactivated when it is not actively competing for a given resource, and the corresponding resources are redistributed among the active clients.

The final panelist, Volkmar Uhlig, showed how applications could directly control pinning of pages in their virtual address spaces. Although this capability is usually reserved for the operating system, Uhlig explained why applications could benefit from direct control, and how this could be implemented without compromising protection boundaries. Robert Haas asked what prevented a process from artificially inflating its working set. Uhlig's reply was that this could be done on today's operating systems, and that his system prevented monopolization of physical memory. Margo Seltzer had two questions: what kind of application could benefit from this work, and how this work was related to compensation using LRU with placeholders. Uhlig's answer to the first question was that any system that performed a network file transfer could benefit since it could pin the pages being transferred; since he was not familiar with the work mentioned in the second question, he could not comment on how similar it was.

At this point, the floor was opened for broader questions to any of the panelists. The first question, by Naomaru Itoi to Dave Sullivan, asked how an application like a Web server could tell what resources it would need. Sullivan's response was that a different currency could be used for each site hosted by the server, and that a resource negotiator for a given site could use feedback to learn about its resource usage. To a followup question, he added that a resource negotiator could benefit from an initial estimate of the site's resource needs.

Robert Haas directed the next question to Dickon Reed. He wanted to know why anyone would use a Xenoserver when bandwidth was free. Reed replied that bandwidth was not going to be free forever, and that it was likely to become more expensive. At that point, Xenoservers would be attractive. Bill Tetzlaff joined the discussion at this point, pointing out that a client typically paid a fixed price for a network connection, and so the marginal cost per packet was zero. Reed disagreed, saying that the shift to per-byte charges was already happening.

Prashant Pradhan asked Dave Sullivan how his approach differed from min-max fairness, and why negotiations were necessary. Sullivan replied that although the default policy was to give all clients an equal number of tickets for each resource, processes might benefit from giving up some of their rights to one resource in return for additional rights to another resource. Pradhan then asked if this didn't imply the existence of a single entity that was aware of all allocations in the system. Robert Haas, one of Sullivan's coauthors, disagreed, saying that global knowledge was something that ticket exchanges were trying to avoid. Sensing that progress wasn't being made, he suggested that the discussion be continued offline.

Addressing Dickon Reed and pushing further on Bill Tetzlaff's earlier comment, Margo Seltzer suggested that Xenoservers might be solving the wrong problem. Even if a transatlantic link was of T3 quality, her link to her ISP was much lower, perhaps 56Kb/s. She did not understand how a Xenoserver could help her; further she wondered whether the economic incentives were being structured in the wrong way. Reed's response was that there were applications where the trunk bandwidth was the limiting factor, and Xenoservers were useful in that case. Transcoding could cope with last-mile issues, but this would have to be done somewhere else in the system. He agreed that the social issues were tricky and would have to be addressed in getting this system to work.

Mitchell Tsai observed that Akamai Technologies, a private spin-off by some MIT professors was working with many of the world's largest Web sites (Yahoo, CNN, Go Network) to offer FreeFlow, a service similar to Xenoservers. The idea is to offer content distribution services like Web caches and proxies that migrate data closer to users.

## Sessions 7 & 8: Break-Out Discussions

The last part of the workshop was an opportunity for participants to brainstorm about the future of operating systems research in small groups. The topic for discussion was the question "What are the 5 most important problems that must be solved by OS researchers in the next decade?" The groups were encouraged to treat this question broadly, factoring in both technical and non-technical issues. Each group was to report its conclusions in a 10-minute summary, giving a prioritized list of problems and corresponding rationale.

In honor of the workshop's desert location, the groups were named after species of cacti: *Opuntia*, *Pereskia*, *Rebutia*, *Rhipsalis*, and *Saguaro*. Each group was between 9 and 12 people in size, and had a designated leader. The leaders were: Karin Petersen (Opuntia), Srini Seshan (Pereskia), Margo Seltzer (Rebutia), Mike Feeley (Rhipsalis), and Mike Jones (Saguaro). All the groups chose to work outdoors, some by the pool (with some of their members in the pool!), others on the patio outside, and so on. Rounding up everyone for the final session indoors was not easy, but the group leaders did accomplish their mission.

### Opuntia

Speaking for the Opuntia group, Robert Haas listed the following problems:

1. Getting good systems into the real world

2. Systems management

3. Reliability

4. Wide-area Networking

5. Systems that scale or adapt to different environments.

Haas explained that the first problem would require researchers to work on real problems, to participate actively in technology transfer and deployment, to work with the relatively small number of technology providers, and to strive for simplicity. The issue of system management spanned many sub-issues, including monitoring of large systems, simplifying operation from an end-user's standpoint, and reducing total cost of ownership. Meaning to reinforce this point, Haas said that most users were not very smart, and that a good goal would be to make computers that mothers could use. Mary Baker objected to this characterization vehemently, noting that some mothers like herself were quite computer-savvy. After acknowledging his faux pas, the speaker went to the third problem, reliability. He observed that the main issues here were system crashes, fault isolation, and interoperability.

Turning to wide-area networking, he noted that this was one of the few remaining problem areas for performance in modern systems. Better OS support, and rethinking many of the classic distributed systems primitives like RPC in a Web world were some of the other issues under this topic. The emerging field of pervasive computing, and the need for a "lingua franca" for interoperability were related challenges. Finally, Haas noted that there were many contexts in which a system could operate: high vs. low end; many vs. few vs. single node; and high vs. low security. Ideally a single system should be able to configure itself to all the environments.

## Saguaro

David Oppenheimer presented the second summary, on behalf of the Saguaro group. He noted that this group had a contentious discussion, and could only agree on one slide with a couple of high level points. The first point was entitled "No-Futz Computing", a term coined by Mary Baker to refer to the absence of a broad range of common headaches in computer systems. It implies avoidance of fuss in computing, and includes the ability of people to move anywhere, minimal wastage of people's time in system administration and related activities, and reliable infrastructure and universal interoperability. The second point was security, again "no-futz" in flavor.

By way of substantiation, Oppenheimer observed that a huge fraction of every user's time is squandered in "futzing" with the system today. This is biggest bottleneck in the performance of most systems. Since machines are cheap and people are expensive, striving to reduce the amount of "futzing" is the most important goal for researchers.

The reaction from the audience to the concept of "no-futz computing" was highly positive. The phrase seemed to capture what was latent in every group's thinking that afternoon. To much laughter, it was suggested that the "FutzMark(TM)" be henceforth recognized as the official benchmark of "no-futz computing".

## Rhipsalis

The presenter for the third group, Rhipsalis, was by Stefan Savage. He began with the disclaimer that predicting way into the future is highly inaccurate. For instance, he noted that no one at HotOS-II would have guessed that the most significant problem of 1999 would be Y2K! His list of five problems were:

1. Tools for building OSs

2. Adaptive systems

3. Science

4. Device drivers

5. Big groups of interconnected dumb devices

Savage observed that people doing OS design are not good at checking, and that this is a harder problem in OS design than in application design. This is why tools emerge as a leading requirement for OS research. Next, he explained that the need for adaptive systems arises because systems need to be robust across time, even in the face of changing specifications. He emphasized that the robustness had to be not just with respect to performance but also error handling and configuration. On the third problem, Savage characterized the OS community as being rather poor scientists. He advocated the goal of practicing the scientific method by testing hypotheses, rather than building systems and showing that they worked. This would require OS researchers to be better educated about statistics. Turning to the fourth problem, Savage noted that device drivers are not glamorous and no one publishes papers about them, but they are really important. Writing device drivers is difficult, but we don't fully understands why this is the case. Finally, he justified the last problem by observing the we are entering an era where the Internet and active "stuff" enters the home, and where each home has its own network. Many of the devices on such a network will not be smart enough to run real OSs, but there will be many problems in configuring them.

## Rebutia

Armando Fox, representing the Rebutia group, began by noting that performance was a subliminal message in many of the group's arguments. He then proceeded to list the following problems:

1. Managing complexity
2. Security
3. Resource management
4. Flexible, adaptable OSs
5. New computing models for the post-PC era

Fox observed that there was an implicit sixth problem, that of metrics. Indeed, none of the other problems could be meaningful without metrics and so it should really be viewed as a theme that cuts across all of them. He then presented the rationale for the ranking of problems. While new computing models are exciting, we don't need it right away. Similarly, flexibility and resource management are problems that can be made to go away by throwing enough money at a system. Security, however, cannot be solved in that manner. Complexity ranks even higher than security because a system that is too complex for users to understand will be a security risk.

## Pereskia

The final group, Pereskia, had Jeff Chase as its spokesperson. Noting that almost all his points were trademarked statements of others, Chase listed the following problems:

1. Performance is paramount.
2. Nothing has changed: everything is broken
3. The Internet is the Computer
4. The Whole is Greater than the Sum of the Parts
5. Naming

Proceeding to explain these sound-bites, Chase noted that performance continues to dominate attention, but needs to be defined more broadly. All other problems would be trivial if we didn't care about it. Other problems, characterized as "*ility" for "reliability", "manageability", "usability" and so on, can be addressed by the budget surplus of raw performance that systems now have. On the second problem, Chase said that this was not necessarily a research problem but we did seem to be getting dragged into the same rat holes time after time — making systems work smoothly is critical. Turning to the third problem, Chase noted that scaling local systems up to a worldwide scale remained a challenge. On the fourth problem, he said that parts are cheap but integrating them into a whole is expensive. To be able to deploy power wherever we need it, incremental scalability and backwards/forwards compatibility are needed. Finally, Chase reminded the audience that naming still remains a poorly solved issue, especially at the intergalactic scale.

## Discussion and Conclusion

After these presentations, there was a brief period of open discussion where the participants tried to sort out their thoughts and decide whether a clear winning theme had emerged. There was substantial sentiment that the concept of "No-Futz Computing" was a winner. The discussion then turned to how we could address this challenge. One school of thought, represented by Armando Fox and Margo Seltzer, felt that there was insufficient market penalty for low FutzMark ratings and that the immaturity of the field was the source of this problem. A second school of thought, represented by Jeff Mogul, felt that system management was primarily about policy, not mechanism. Mogul remarked that it is all of the things you have to do when the computer can't adapt itself to you. As such, it is not easily automatable and is really a social, not technical, problem. This led to a heated and confusing debate, involving Margo Seltzer, Mary Baker, Armando Fox, Stefan Savage and a number of others, on how complexity could be reduced in systems. No real answer emerged, though there were many clever repartees.

In the waning moments of the workshop, the participants were asked for a show of hands on whether "No-Futz Computing" emerged as the dominant problem from the breakout session. A substantial majority of hands were raised in response, and so this was declared to be the clear winner. In response to a request from Randi Thomas, the participants were also asked for a show of hands on whether "Post-PC Computing" emerged as an important problem. Many hands were raised, but distinctly fewer than for "No-Futz Computing".

In closing the workshop, Satya requested a final round of applause for the excellent leadership of the General Chair, Peter Druschel. He also noted that the true measure of this workshop's success would become apparent only a few SOSP's or OSDI's hence, when the ideas that were debated here emerge at the core of high-quality research publications on innovative systems.