August 2002

A Discussion with Leslie Lamport
by Dejan Milojicic

Dejan Milojicic: You have worked on several visionary problems that have come to practical fruition—sometimes even decades later. How much does timeliness affect your choice of topics?

Leslie Lamport: Some problems have come my way because engineers were building a system and needed an algorithm. The fast mutual exclusion algorithm and disk Paxos are examples. Some, such as cache coherence, I just thought up myself. I've never thought about timeliness in choosing problems.

Which of your contributions do you think had the most impact on the advance of computer science and industry?

My most often cited paper is 'Time, Clocks, and the Ordering of Events in a Distributed System.' (http://research.microsoft.com/users/lamport/pubs/pubs.html#time-clocks) I don't know if that's the same as impact, since I can't point to a lot of work that directly depends on anything in that paper, but perhaps it affected the way people think about distributed systems. I don't think I've had much impact in industry yet, though I suspect that Paxos and the state-machine approach is about to start making a significant impact on the design of distributed systems. It's certainly being noticed inside Microsoft.

Which contributions of yours do you consider most important, even if they have not made an impact?

I think my most important contribution is my work on writing structured proofs, which is perhaps more relevant to mathematicians than computer scientists. In the unlikely event that I'm remembered 100 years from now, I expect it will be for that.

What is the importance of specifications and verifications to industry today and how do you think they will evolve over time? For example, do you believe that more automated tools will help improve the state of the art?

There is a race between the increasing complexity of the systems we build and our ability to develop intellectual tools for understanding that complexity. If the race is won by our tools, then systems will eventually become easier to use and more reliable. If not, they will continue to become harder to use and less reliable for all but a relatively small set of common tasks. Given how hard thinking is, if those intellectual tools are to succeed, they will have to substitute calculation for thought.

Are you hinting at advances in the deployment of artificial intelligence techniques?

Having to resort to traditional methods of artificial intelligence will be a sign that we've lost the race.

What are the important remaining problems in computer science? How should the community tackle them?

For the past 20 or so years, I've been asked, 'What are the important problems?' My answer has remained the same: I have no idea. If I did, I'd be working on them.

What motivates you?

I don't think there's a pattern. As I mentioned earlier, some problems come from engineers and some I invent myself. Sometimes the problem comes after the solution. I recently discovered how to implement a nice variant of Paxos, and only afterwards realized that it provided a new solution to the distributed transaction commit problem. The Paxos algorithm itself came out of a failed attempt to prove that no such algorithm existed. Often, I come across an idea that I then try to turn into a problem and solution. It's like finding an arrow sticking in a wall, drawing a bull's-eye around it, and telling people what a great marksman I am. The 'Time, Clocks...' paper came about this way, where the arrow was shot by Paul Johnson and Bob Thomas, but it missed their target.

Have you ever been more interested in hard problems as opposed to relevant problems?

I've never thought about relevance. I do try to make sure that I work on real problems, as opposed to problems that are just artifacts of some language or formalism. For example, implementing an atomic register with a weaker class of register is a real problem; I can describe it in physical terms without reference to any model. An example of an artificial problem, which absorbed some people 20 years ago, is how can you distinguish a distributed system from a non distributed one? A real problem at least has a chance of turning out to be relevant. Even though the atomic register algorithms seem irrelevant today, engineers might some day want to implement atomic registers with non atomic ones.

What problems are irrelevant and underrated in your opinion?

Irrelevant problems are ones invented by theorists that are neither inherently interesting nor likely to have any practical benefit. Such problems are the ones that arise in studying a formalism and cannot be expressed outside that formalism. It's easy to come up with relevant problems, like building better device drivers. System management is perhaps the major relevant problem facing the computer industry. However, it's never clear in advance whether sitting down trying to solve a problem like that will lead to any interesting research. I prefer problems that can be solved by proving a theorem, rather than by writing a million lines of code. I suspect that system management is not my kind of problem.

You spent a lot of effort making your research interesting and memorable for the audience by drawing parallels with Byzantine generals, Paxos Island's parliament, and so on. How much do you consider research to be art as opposed to science? Should any research have elements of art?

Long ago, I observed that Edsger Dijkstra's dining philosopher's problem (www.cs.utexas.edu/users/EWD/ewd03xx/EWD310.PDF) received much more attention than I thought it deserved. For example, it isn't nearly as important as the self-stabilization problem (www.cs.utexas.edu/users/EWD/ewd03xx/EWD391.PDF), also introduced by Dijkstra, which received almost no notice for a decade. I realized that it was the story that went with it that made the problem so popular. I felt that the Byzantine generals problem (which was discovered by my colleagues at SRI International; http://research.microsoft.com/users/lamport/pubs/pubs.html#byz) was really important, and I wanted it to be noticed. So, I formulated it in terms of Byzantine generals, taking my inspiration from a related problem that I had heard described years earlier by Jim Gray as the Chinese generals problem.

The Paxos paper (http://research.microsoft.com/users/lamport/pubs/pubs.html#lamport-paxos) was written mostly for fun, though I did make use of the story to eliminate simple, uninteresting details by saying that those details had been lost. However, the paper was a disaster because no one could get past the story and understand the algorithm (Butler Lampson was the one exception I knew about.) When I submitted the paper to ACM Transactions on Computer Systems, the referees said that it was mildly interesting, though not very important, and it would be publishable if I got rid of all the Paxos stuff. I was so annoyed by what I felt to be computer scientists' lack of humor that I let the paper drop. Years later, Ken Birman [editor of TOCS at the time] was willing to publish it in its original form.

Along similar lines, to what extent do you consider research fun versus hard work?

Hard work is hauling bales of hay or cleaning sewers. Scientists and engineers should be grateful that society is willing to pay us to have fun.

What have been your best and worst experiences with the publishing process? Where do you think the publishing industry is heading?

The worst part of the process has been proofreading. After working hard to remove errors in the manuscript, you then have to remove the ones inserted by the printer. Springer is now typesetting directly from the LaTeX source, which is wonderful. I hope other publishers will do the same, but I'm not optimistic.

Do you see the Web as an enabler for traditional scientific publishing?

I would expect that paper copies of refereed journals will disappear and they will all be online, but I'm no better at predicting the future than anyone else.

Personally, I am fascinated with the LaTeX tool you gave to the community. What features would you like to add to it?

I would like to see a system that combines the good features of Word, such as ease of use for doing simple things, with the good features of LaTeX, such as the relative ease of separating logical content from formatting. I have ideas on how to design such a system, but it would take several people years to do, so it probably won't happen.

Ease of use versus rich or complex features is one dimension (for example, Word versus FrameMaker); interactive versus batch approach is another dimension. How much of LaTeX's strengths do you believe come from the batch approach?

Tex does a good job typesetting because it was designed as a batch system. In the early 80's, computers were too slow to be interactive and produce decent output. Donald Knuth chose quality rather than speed (though he did a remarkable job of optimizing his algorithms). Now, you can run a 20-or 30- page paper through Tex or LaTeX in a fraction of a second. That's as interactive as you need.

I think LaTeX was successful because, in return for a modest investment in learning how to use it, a scientist or engineer could write papers without worrying about what they were going to look like. LaTeX lacks many features of other systems. I don't think most of those features help an author of a technical work in any significant way to communicate his or her ideas to the reader. (The major exception is its lack of good facilities for drawing pictures.)

Online documents make possible new ways of interaction that can improve author and reader communication. Some day, we'll figure out how to take advantage of those possibilities. But before that happens, readers will probably have to put up with a lot of distracting screen-flash.

You addressed many system characteristics, such as consistency, failure tolerance, and real-time. Is it possible to reason in the same way about other characteristics of the systems, for example, mobility (disconnection, locating, and discovery), scalability, and so on?

No one has figured out how to do it yet. This suggests that, if it's possible, it will require a new way of thinking about the problem, and someone really smart to do that thinking. But not all problems are solvable.

What about the scalability of distributed systems?

All my distributed algorithms have a parameter N that represents the number of processes. These algorithms have costs, in time and messages that are functions of N. In most applications I've considered, the number of processes has been small enough that simplicity was more important than asymptotic costs. As people build systems with more and more processes, asymptotic costs become more important and people will devise new algorithms that reduce them.

There are various classifications and interpretations of systems evolution, such as distributed, mobile, and pervasive systems. Once the hype is removed, do you believe that there are inherently new disruptive characteristics in mobility, pervasiveness, and peer-to-peer, or do you consider them different variations of distributed computing?

These are the same old distributed systems, except with significantly different parameters. Sometimes, changing the parameters leads to a new set of interesting problems. For example, going from a set of processors inside a computer to a network of processors just changes the ratio of communication to processing delay. Sometimes these changes don't lead to anything new. For example, going from point-to-point communication to Ethernet didn't lead to any interesting new theory. I haven't found any good new theory coming from these new technologies; maybe someone else will.

What are the most important lessons you've learned in your career?

My career has been idiosyncratic. Any meaningful lessons I could draw from it would, at best, apply to people starting their careers 30 years ago.

What are you most proud of in your career?

The bakery algorithm (http://research.microsoft.com/users/lamport/pubs/pubs.html#bakery). It's the one algorithm that I feel that I discovered rather than invented.

What are your most spectacular failures and what have you learned from them?

Just as no one has benefited very much from my successes, no one has been harmed by my failures. Neither my successes nor my failures have been spectacular. I've done a few things that are worth remembering; and I've done a few things that are best forgotten. In general, the things that are worth remembering were simple little ideas, not grand visions.

What would you like to convey to current and future researchers and practitioners?

Don't pay attention to pedantic old farts like me telling you what to do.

For more information on Leslie Lamport, please visit his homepage at http://lamport.org/.