Interview with Andrew Tanenbaum

RIK FARROW

Andrew S. Tanenbaum was born in New York City and raised in White Plains, NY. He has a BS from MIT and a PhD from the University of California at Berkeley. He is currently a Professor of Computer Science at the Vrije Universiteit in Amsterdam. Professor Tanenbaum is the principal designer of three operating systems: TSS-11, Amoeba, and MINIX. In addition, he is the author or coauthor of five books, which together have been translated into more than 20 languages. In 2004 Tanenbaum became an Academy Professor, which carried with it a five-year grant totaling one million euro to do research on reliable operating systems. His university matched this amount. In 2008 he received a prestigious European Research Council grant of 2.5 million euro to continue this research. Tanenbaum is a Fellow of the ACM, a Fellow of the IEEE, and a member of the Netherlands Royal Academy of Arts and Sciences. In 1994 he was the recipient of the ACM Karl V. Karlstrom Outstanding Educator Award. In 1997 he won the ACM SIGCSE Award for Outstanding Contribution to Computer Science Education. In 2007 he won the IEEE James H. Mulligan, Jr. Education Medal. ast@cs.vu.nl

Rik is the editor of *;login:*. rik@usenix.org

Although I had certainly encountered Andrew Tanenbaum at other conferences, the first time I remember talking to him was in a "terminal" room at USENIX Annual Tech in 2004. By that time, a terminal room was a place where you could have a hardwired connection to the Internet, and Andy showed me www.electoral-vote.com, a political Web site which he was working on that analyzes polling data. For someone who focuses on working with PhD students, and writing books and operating systems, building such a useful political site seemed a bit far afield to me. But the more you know about Andy, the more you learn just how broad his interests are.

After that meeting, we would usually spend some time talking at the systems conferences that we both attended. While we mostly discussed MINIX 3 (www.minix3.org), we also talked about other things, such as his current position within Vrije Universiteit in The Netherlands. If you are wondering how Andy wound up there, you should read his FAQ [1]. But given the brief meetings, there were some things I didn't get to ask him, and with his past experience in distributed systems, I thought that now would be a good time to interview him.

Rik: While at the OSDI conference (2014), I heard someone mention that people have forgotten about all the work that was done on building distributed or parallel systems in the 1980s and early '90s. Could you explain why there was such strong interest in systems like Amoeba [2, 3], and Sprite?

Andy: There wasn't a lot of commercial interest in parallel or distributed systems in the 1980s, but there was some in academia from people who try to stay ahead of the curve. Already then, cheap workstations and PCs existed, and it occurred to some people that you could harness them together and get a bigger bang for the buck than buying a supercomputer. At the University of Wisconsin, for example, there was work on harvesting the power of idle workstations to form an ad hoc supercomputer.

My work consisted of putting sixteen Motorola 68000s in a rack and letting people start jobs there from their desktop machines without having to worry too much about the details. We called the rack the "processor pool" and built an operating system (Amoeba) to control it. We published some papers about it, but it didn't get much attention in commercial circles. Nowadays it is called "cloud computing" and gets a lot of attention.

Rik: I think people often forget just how slow processors were in the '80s, right into the early '90s. My first UNIX system, a 68010, had a blistering clock rate of 10 MHz (1983). A 1987 Sun-4/260 ran at 16.67 MHz, and was noticeably faster than the 68030s it replaced. Having a rack of systems, where a user could run programs on the least busy one, surely must have seemed like a great idea.

Andy: What the Amoeba processor pool did was create a shared resource, which is more efficient than dedicated ones. If all the money available for computing resources was spent to give each user the most powerful computer you could buy for that amount, you would often have the situation that one user needed a lot of computing power for a short time, for example, to run "make" to compile a big program, while all the other computers were idle but unavailable to the one who needed the power. By having a processor pool available to everyone, if one user needed the whole thing and nobody else needed any computing, the one user could get all of it. If two users needed a lot of computing, they would each get half. So the model was to put most of the power in the processor pool and just give users simple terminals. This whole model foreshadowed cloud computing, which also centralizes the computing power and lets you take as much as you need for a short period and nothing when you are sitting around scratching your head deciding what to do next.

Rik: Did the work on Amoeba have anything to do with the development of MINIX, the operating system you wrote to help students learn about operating systems?

Andy: Amoeba didn't influence the development of MINIX much as Amoeba was intended as a research vehicle and not as a UNIX clone. For example, the Amoeba file system, the bullet server, wrote files onto the disk as consecutive sectors so they could be read back at very high speed (basically one disk command to read a whole file). Files were also immutable. The system was based on cryptographically secure capabilities managed directly by user programs. It was an attempt to push the envelope on research and was completely different from MINIX, which was initially intended for teaching students how a UNIX-like system worked inside.

Rik: MINIX started out as a microkernel, moving away from the generally accepted design of monolithic kernels, which are still dominant today. What were the advantages of using a microkernel for MINIX?

Andy: Since my initial goal in writing MINIX was to have students learn from it, I thought that breaking it into a number of smaller chunks that interacted in very well defined ways would make it easier to understand. Generally speaking, for example, six programs of 2000 lines are easier to understand than one program of 12,000 lines.

But also from the beginning, I was aware that putting most of the operating system in "user mode" as multiple processes would make it more reliable and more secure against attempts to hack it. Now the 8088 didn't have kernel and user modes, but I assumed that some future version of the 8088 would have them, and that is what happened, of course.

Rik: So why don't we see more microkernels used today?

Andy: Because they are mostly used in embedded systems, where reliability matters. In mission-critical embedded systems, microkernels like QNX are widely used but they are invisible to the user. Also, L4 is used in the radio chip inside over a billion smartphones. I think monolithic kernels are mostly used due to inertia, whereas for each new embedded system the designers look around and see what is best right now without worrying too much about legacy. Performance used to be a problem with microkernels, but L4 showed this is not inherent. In many other areas legacy systems dominate, even though they are inferior to other ones.

For example, I have never heard an argument why the furlong-stone-fortnight system used in the UK and US is better than the metric system other than "We've always done it that way." Consider Fahrenheit vs. Celsius. Try arguing that the NTSC (Never Twice the Same Color) television system is better than the alternatives. What about point-and-shoot cameras that have an aspect ratio of 4:3, like 1950s TV sets? C is still widely used even though it is not type safe, and C programs are subject to buffer overflow attacks and more. COBOL is horrible but lasted for decades. In general, once some technology gets established, it is very hard to dislodge.

I think the research community is too fixated on Linux, and any monoculture is bad. Even a stable, mature, open-source system like FreeBSD hardly gets any attention.

Rik: I've written many times that running microkernels on current CPU architectures cannot work well, as microkernels and monolithic kernels rely on very different designs for system communication. Monolithic kernels keep all modules in one privileged address space, which is convenient, fast, as well as considerably less secure. Microkernels minimize the amount of code running within the privileged address space, but at the cost of having to make context changes when communicating with or between system modules. Also, unprivileged modules need privileged access for many of the tasks they perform.

Do system architecture changes like the IOMMU [4], as well as others I either don't know about or haven't imagined yet, help microkernels run as fast or faster than monolithic ones, but with a much higher level of security?

Andy: Better hardware certainly helps but I don't think IOMMUs are necessarily the answer. One thing that may help is multicore architectures. One of my PhD students has been doing research on a prototype system in which the major server components each run on their own core. This way when it is needed, there is no context switching, the cache is warm, and the server is ready to run with no overhead. As we move toward a world in which all chips

have cores to spare, this could make microkernels more competitive since people won't worry about "wasting" cores, just as no one worries about "wasting" RAM on bloated software now.

Rik: Now that you've been retired [5] from Vrije Universiteit, what do you plan on doing? You've always stayed busy, much more so than most people.

Andy: For one thing, I will continue teaching one course I give in our masters program (on how to write a grant proposal). I also still have five PhD students to supervise.

For another, I want to continue publicizing MINIX 3. It is more popular than many people realize. According to the statistics from the log, visible at minix3.org/stats, we had over 60,000 downloads of the .iso file in 2014 and over 600,000 since 2007. The minix3.org site has had over 3 million visits since I put the counter on there about five years ago.

Still, I would like to build a more active community. One thing I will probably do in that respect is sign up for the ACM Distinguished Speakers Program and give lectures about MINIX at universities. I need to maintain my Platinum Medallion status on Delta Airlines somehow :-)

Furthermore, I have five books that are current and in constant need of new editions. Fortunately, I have excellent coauthors to help me out.

In addition, I had about 50,000 of my negatives and slides scanned in, and I want to organize, label, and clean them up with Photoshop. I also have a couple hundred hours of video that need work. I recently bought a Mac Pro (garbage can model) to handle the video processing.

So I don't think I'll be bored, for a few months, anyway.

References

[1] Andrew S. Tanenbaum's FAQ: http://www.cs.vu.nl/~ast/home/faq.html.

[2] A. Tanenbaum et al, "The Amoeba Distributed Operating System": http://www.cs.vu.nl/pub/amoeba/Intro.pdf.

[3] This page has links to papers, as well as a good description of Amoeba: http://en.wikipedia.org/wiki/Amoeba_%28operating_system%29.

[4] Simon Peter, Jialin Li, Irene Zhang, Dan R. K. Ports, Doug Woos, Arvind Krishnamurthy, Thomas Anderson, and Timothy Roscoe, "Arrakis: The Operating System Is the Control Plane," OSDI '14.

[5] Retirement: http://www.few.vu.nl/~ast/afscheid/.

Buy the Box Set!

Whether you had to miss a conference or just didn't make it to all of the sessions, here's your chance to watch (and re-watch) the videos from your favorite USENIX events. Purchase the "Box Set," a USB drive containing the high-resolution videos from the technical sessions. This is perfect for folks on the go or those without consistent Internet access.

Box Sets are available for:

- **SREcon15**
- **FAST '15:** 13th USENIX Conference on File and Storage Technologies
- **LISA14:** 27th Large Installation System Administration Conference
- **OSDI '14:** 11th USENIX Symposium on Operating Systems Design and Implementation
- **TRIOS '14:** 2014 Conference on Timely Results in Operating Systems
- **USENIX Security '14:** 23rd USENIX Security Symposium
- **3GSE '14:** 2014 USENIX Summit on Gaming, Games, and Gamification in Security Education
- **FOCI '14:** 4th USENIX Workshop on Free and Open Communications on the Internet
- **HealthTech '14:** 2014 USENIX Summit on Health Information Technologies
- **WOOT '14:** 8th USENIX Workshop on Offensive Technologies
- **URES '14:** 2014 USENIX Release Engineering Summit
- **USENIX ATC '14:** 2014 USENIX Annual Technical Conference
- **UCMS '14:** 2014 USENIX Configuration Management Summit
- **HotStorage '14:** 6th USENIX Workshop on Hot Topics in Storage and File Systems
- **HotCloud '14:** 6th USENIX Workshop on Hot Topics in Cloud Computing
- **NSDI '14:** 11th USENIX Symposium on Networked Systems Design and Implementation
- **FAST '14:** 12th USENIX Conference on File and Storage Technologies
- **LISA '13:** 27th Large Installation System Administration Conference
- **USENIX Security '13:** 22nd USENIX Security Symposium
- **HealthTech '13:** 2013 USENIX Workshop on Health Information Technologies
- **WOOT '13:** 7th USENIX Workshop on Offensive Technologies
- **UCMS '13:** 2013 USENIX Configuration Management Summit
- **HotStorage '13:** 5th USENIX Workshop on Hot Topics in Storage and File Systems
- **HotCloud '13:** 5th USENIX Workshop on Hot Topics in Cloud Computing
- **WiAC '13:** 2013 USENIX Women in Advanced Computing Summit
- **NSDI '13:** 10th USENIX Symposium on Networked Systems Design and Implementation
- **FAST '13:** 11th USENIX Conference on File and Storage Technologies
- **LISA '12:** 26th Large Installation System Administration Conference

Learn more at: www.usenix.org/boxsets