A Conversation with Sam Leffler

PHOTOGRAPH BY TOM UPTON

The seeds of Unix and open source were sown in the 1970s, and Sam Leffler was right in there doing some of the heaviest cultivating. He has been actively working with Unix since 1976 when he first encountered it at Case Western Reserve University, and he has been involved with what people now think of as open source, as he says, “long before it was even termed *open source*.”

While working for the Computer Systems Research Group (CSRG) at the University of California at Berkeley, he helped with the 4.1BSD (Berkeley Software Distribution) release and was responsible for the release of 4.2BSD. He has contributed to almost every aspect of BSD systems, most recently working (again) on the network-

A Unix and BSD

PIONEER DISCUSSES THE
OPEN SOURCE MOVEMENT.

ing subsystem.

After his time at Berkeley, he ended up at Lucasfilm and Pixar, where he mostly did computer

graphics. He then joined Silicon Graphics, where he worked on everything from operating systems to UI tool-kits. He created one of the first host-based laser-printer systems, which eventually became a Silicon Graphics product. Later Leffler moved over to VMware, before striking out on his own as an independent consultant, focusing on system design.

Discussing his view of the open source world with

Leffler is James Russell, an open source veteran at IBM. Russell was involved in some of the first open source projects at IBM, where he was “a part of IBM’s own coming to terms with, understanding, and, ultimately, embracing of open source,” says Russell. He is now in the Lotus Division of IBM, building development tools for the various Lotus platforms.

JAMES RUSSELL You’ve been involved with Unix and open source from its inception, and you worked on BSD in its early days. I’m interested in hearing your thoughts on the influence of the Unix community and how BSD evolved. It certainly had, I think, a pretty big impact on the way networking and ultimately the Internet evolved.

SAM LEFFLER There were probably two periods in the lifetime of BSD. There was the period of time where I was at UC Berkeley with Bill Joy [who went on to found Sun Microsystems] and we were funded by DARPA [Defense Advanced Research Projects Agency]. The source code was licensed and available only to licensees. The reach of the community was smaller and the focus was very different.

After I left, after 4.3, and they started working on 4.4BSD, the focus changed quite a bit toward trying to eliminate the source code licensing and make it open.

We used to hear the arguments that the software that came from Berkeley wasn’t production quality. That always sort of rankled us a little bit, because in the end, I think that people found that the software that came out of the Berkeley group was potentially higher quality in terms of a production environment than what they got elsewhere.

We tended to have very strong peer review within our group. We didn’t just open up the source to let people go and change things, which is a very different method of operation from today’s methods.

In fact, it took me quite a long time to get used to that when I started getting reinvolved with the open source community, because at Berkeley people sent us code and if they had useful stuff, we integrated it into the tree. So there was a chokepoint, if you will.

JR Isn’t Linux more or less managed that way today? There are only a few people who are committing it or integrating it into the tree?

SL To some extent I’m sure that’s true, but I see that at least in the BSD open source world, it’s more of a free-for-all, and it varies from group to group. Some groups do require some sort of peer review, but still you find these little nuggets in the tree—in the software somewhere, where someone has committed something that if some-

one else had looked at it, boy, it never would have gotten in there.

That sort of thing rarely happened at Berkeley in the early BSD time. We were pretty careful about that.

JR It’s always interesting to compare an open source development model with a closed- or proprietary-source development model. Do you think that the model that exercises more control or keeps things more closed is going to yield higher-quality software?

SL I really don’t think it’s the model per se that decides the quality. As in everything, it comes down to the people. The quality of the product is a reflection of the people. But you still need someone or some group of people that coordinates and architects, that makes decisions that steer projects in the right direction, that ultimately decides or defines how good a result you get. At Berkeley, we were a very small community. All of the people who contributed were on the same wavelength. They knew the direction and the goals. I think that was the ultimate reason why the quality was so good.

I’m not trying to say that the old days were better. But today, things work differently, and you see a lot more of an anarchic environment. There are good aspects to it, as well as negative. One negative aspect is that you get more burnout from people who get frustrated watching others do things that you wish they hadn’t done.

JR Do you think that this applies in both the open source and closed source worlds?

SL Actually, I’ve seen it in both. When I was at Silicon Graphics, there were quite a few people working on the operating system. At a company where lots of people are involved, you tend to have senior people and junior people, and you get some frustrations here and there. In a commercial environment, there is more of a review process and more of a responsibility, so when there’s a bug in the system, people know who did it.

In the open source community, if you do something and people would like to talk to you about it, you can vanish. It’s kind of hard to disappear when your cubicle is sitting down the hall.

JR That’s true. “We know how to find you.”

Are there particular open source projects over the years that you’ve looked at and said, “Gee, that’s been pretty successful”?

SL Wherever you find a successful project, you’ll almost always find one or two key people who are really sharp and really have a handle on what’s going on. They also tend to have a strong personality that either draws people

to them or lends an air of trust that brings people in.

The most valuable attribute of an open source project is you get potentially a very large community in which to toss your software out and get people to try it. The feedback that you get—in terms of bug reports or new ideas or just pushback on your ideas—is really what drives things.

The hardest part about an open source project—a truly open source project—these days seems to be attracting productive developers, because the number of available projects, the accessible projects, are getting smaller and smaller. A lot of things are getting done, so there's not as much available to work on these days. The projects tend to be more complex, and trying to find a little corner where you can do your work is harder and harder.

As I look back, for example, in Linux, I see one of its greatest successes was that there were so many pieces of it for people to work on, and everyone could join in and feel that they were being productive and contributing.

At the same time, some of the BSD systems weren't as accessible to these same people, so the opportunities there weren't as great and the entry was a little more difficult.

JR Some well-known companies, such as Red Hat and IBM, employ and pay programmers to work on open source projects. Do you think that that is a good thing from the open source point of view? Do you think that's one way around the lack of available resources to keep some of these open source efforts going?

SL Absolutely. I think it's critical. There are people who are really capable of doing very high-quality work, but if they don't have a source of funding, then they cannot contribute.

There's a good match in many ways between companies and these programmers, so that both the company and the open source community benefit.

■ **JR** There's a lot of the core Internet infrastructure that seems to run on open source software. Do you think that's a reflection of the time in which it was evolving?

SL What open source software did was accelerate the adoption of the Internet and the spread of the Internet by lowering the cost of entry. The perception was that it was easier or cheaper to get open source software than to go

out and buy some product from DEC, for example.

It also probably accelerated things in that the software came out quicker than it would have if it had gone through a commercial development process. This mostly happened after my time at Berkeley, but has become increasingly more important as more people grow to depend on open source software.

The other thing that was really key was that because almost everyone was running the same software, you were guaranteed interoperability. So you had uniformity. Just about everyone was running some version of the BSD TCP/IP stack, for example.

JR As the Internet continues to evolve and grow, now that everybody has accepted the value and will probably be willing to bear a higher cost to enter, do you think that there is an increasing role for commercial software? Do you see some of the infrastructure necessarily switching off the open source base onto a commercial base?

SL I don't see the user base switching, basically because people are lazy. If it works, they are not going to change. There's been an ongoing push for IPv6 [Internet Protocol version 6] and so on, but the open source community has been leading that charge for a long time.

In terms of commercial versions coming in and displacing open source software, I don't really see it happening, except in cases where there are proprietary platforms

like Apple's OS X. Even in that case, a lot of the software itself—the infrastructure aspects of the software—is available in open source form through Darwin.

There are areas, though, where you are seeing commercial software dominate. For example, in the wireless networking environment, you don't really see open source software being compared head to head with commercial offerings, because they are just not comparable.

JR Why is that? Is it simply because no project or no community of people has ganged up to build one?

SL The key thing that has been retarding growth has been the availability of information about radios. In the project I'm working on now, I have an agreement with a company that gives me full access to all the information, and I'm trying to build very high-quality wireless networking support that will allow the open source community to make full use of contemporary radio hardware.

JR Certainly, wireless is in many ways as revolutionary as perhaps the Internet itself—at least in terms of enabling people to connect in different locations. Do you see a similar kind of growth in wireless or wireless usage if what you're trying to do takes off, if you enable the open source community? Or do you think that the commercial software is going to do just fine in that regard?

SL In the wireless environment the commercial support is needed and is fine for certain applications. The marketplace is driving the cost down so low that there's really no opportunity for open source software to have any real

impact there.

What you're going to see, however—and the reason I got involved in this—is that there are people who want to leverage wireless networking to do other kinds of things that aren't necessarily marketable at this time. For example, wireless ISPs, or people who want to build community networks based on mesh topologies—you won't find too many commercial outfits trying to address those users.

That's where having open source support for the wireless networking hardware that's available today is important. With good software support, these new applications will grow. Then you may see some useful things that result in commercial applications that you wouldn't have seen otherwise.

JR In your current efforts, are you actually creating open source software or are you enabling open source software to be created by opening up the APIs?

SL Actually, it's both. For many years, the only real accessible wireless product was based on a part from Intersil. So that's what you find in open source software these days.

About a year ago, I got hooked up with Atheros, which makes one of the best-quality next-generation wireless products, and I arranged a deal with them where I have full access to the documentation for their hardware. My promise to them was that I would provide open source

software that supported their hardware, and I would protect the interfaces to their hardware, which they needed to control because of regulatory requirements. They couldn't release documentation or software that programmed their radios because people could then train them to bands that were illegal or raise the transmit power and so forth.

By having me act as an intermediary and promise to protect those interfaces, I'm able to provide the open source community with 90 percent of what they need in source-code form. There's a small component, which I can only give out in a binary form.

JR That's a pretty interesting intersection between commercial and open source.

SL You get a certain segment of the open source community that says this isn't open source, because they don't get all the source code; and you get the people who don't really care because all they want is something that works. But then you've also got other sources saying that, for example, when there's a problem, I'm the bottleneck.

The original motivation was that I wanted to build mesh networks and I didn't want to work on old technology. So, I went this route. A lot of the open source projects that you find these days are based on people like me who basically want to accomplish something for themselves, and that's what drives them. The result is useful for other folks, so it gets picked up.

There are certainly companies that can benefit greatly

from the open source development model. Companies that develop hardware, for example, like the wireless company I'm involved with, benefit dramatically by having open source software that leverages their hardware. They can just point people at it, and instead of having a royalty-based commercial product, they can use Linux and my software or something like that.

JR Do you see a relation between open source and the official standards organizations?

SL There's a strong bind there. The open source community tends to be quick to produce results that don't necessarily define, but implement, standards, and in so doing, allow you to get some feedback on whether the standard is a good one or not. Open source development provides a reality check on standards.

JR How about the other way around? By building something and putting it out there, isn't the open source community setting a standard, independent of, or in advance of, the more formal standards process, which can take quite some time?

SL That happens, but I think that also happens in the commercial world. The big difference is the speed with which it reaches users, which then drives the standard faster.

JR I want to go back to what you were saying about your current experience, where you are in the position of supporting an open source community but not able to give away all of the source of what you're writing. One of the comparisons that we see in the open source world is a difference in philosophy—the way the current BSD or Apache opens up its source or licenses its source, versus the way the GPL [GNU Public License] does it.

SL I've been giving away software for quite a long time. Probably the first package I gave away was a library for reading and writing an image-file format called TIFF. I was at Pixar at the time, and we wanted to be able to get images out to people, and we needed a format that everyone could read back. I just basically said I'm going to write my own library and I'm going to make it free. And I meant free. I didn't care what you did with it. You could turn it into a product. The result was that it became a de facto standard.

The point is that when I think of free software I think *free*—no ties at all. The reason I don't agree with the GPL is because it attaches strings to giving away software.

People who work on open source projects are moti-

vated by doing work that they want to do, and they want to share. The sharing is what's really the key. The licensing is—for better or worse—sometimes an encumbrance in that it makes it harder to do this. My attitude all along has been you give it away; you don't look back.

Open source software needs a licensing model that reflects that attitude. I don't agree with licenses that attach requirements. I think it's fair to ask people to acknowledge your work when they take it, but you can only ask them to. I don't think you can require it. In the same sense, I think it's fair that people will recognize that proprietary improvements or changes in code or fixes should come back to the originator. But I don't think it's right to require it.

JR We've talked about various business models that are synergistic with open source. You're involved in one right now. The hardware manufacturers that you're working with have got a much greater market or community for their hardware as a result.

We see this in the IBM world, too, where our investment in making Linux better or in making our Eclipse IDE [integrated development environment] open source pays us dividends in the form of greater usage, more standardization, less need to worry about compatibility—all the things that we've talked about today. It's worth it for the commercial company to invest—either in you or Red Hat or whomever.

SL It's a very hard thing for a company to get its head around, because the developers are often remote and they need to develop a relationship with a company they're working with, which tends to have a higher cost to entry. It's probably one of the main things that retards companies investing in open source development.

JR This touches on the question of the cost of open source software.

SL This whole notion that open source software is free is a big fallacy. First, when you start using it as an end user, and something goes wrong, then you have to pay for support or you have to figure out how to fix it. Also, probably the most productive open source developers are those who need a paycheck. They are the ones who need to find a mechanism for funding their habit. I don't know what the answer is other than to find your sugar daddy. ☹

LOVE IT, HATE IT? LET US KNOW

feedback@acmqueue.com or www.acmqueue.com/forums

© 2004 ACM 1542-7730/05/0500 \$5.00