Fred B. Schneider on Distributed Computing

Interview by Dejan Milojicic, dejan@spica.hpl.hp.com

The interviews for IEEE Distributed Systems Online, which investigate such areas as open problems, past achievements, future opportunities, and deployment aspects, represent a continuation of the IEEE Concurrency interview series.

Distributed systems encompass many areas of computer science, such as computer architecture, networking, operating systems, embedded devices, and security. Along with new technologies, such as wireless and wearable computers, distributed systems permeate everyday life—in homes and cars, and at work. Various aspects of distributed computing, such as performance, security, and reliability, can easily determine the success or failure of many businesses, cities, or even countries. Computers and networking provide what were thought to be, until recently, unthinkable opportunities, but they also increase our dependency on distributed systems. In order to move the technology further, we might need new ways of thinking about and developing these systems.

Fred Schneider was my first choice as an interviewee for the first issue of IEEE Distributed Systems Online. Having known Fred for a number of years, and the high standards to which he adheres, I was sure that interviewing him would be very interesting and insightful. Fred Schneider brings expertise from various distributed systems fields, such as security and high availability. He can draw the big picture and also discuss the details. The most rewarding part of conducting the interviews is being able to explore the greatest minds in today’s technology. This was certainly the case with this first interview.

—Dejan Milojicic

What are the critical unsolved problems in distributed systems today?

People build distributed systems that, by and large, would work in the best of all possible worlds. But little attention is paid to hostile attacks or to the realities of a world where people make mistakes, there are environmental disruptions, and so on. We need to figure out how to build distributed systems for the real world.

In the past, our approach has been to build systems involving mutually trusting and mutually cooperating subsystems. That is how the telephone system was initially structured and that’s how early versions of the Internet were built. The architecture was sensible then, because there were only a small number of providers. But distributed systems today involve too many different cooperating entities for each one to trust all the others. We need architectures that work when there are mutually distrustful components. We need architectures that support cooperation for achieving a common goal but that do not require subsystems to make strong assumptions about peers.

The need for new paradigms extends to algorithmic problems as well. For example, a fundamental problem in distributed systems has been that of reliably distributing information to a collection of processors. This is called the broadcast or the agreement problem. Past work has concerned enforcing strong guarantees (for example, Byzantine Agreement protocols) or enforcing virtually no
guarantees (for example, best-effort delivery). Yet, we might be able to implement sufficiently strong guarantees by using gossip or diffusion-protocols that are cheaper than Byzantine Agreement but better behaved than best-effort delivery.

We also don’t understand emergent behaviors of networks. We use and study the Internet, but nobody understands how or why the Internet behaves the way it does. People are surprised when they see statistics indicating how often routes change, or the extent to which the load at a switch or along a link fluctuates. So, while we might understand what individual pieces of the Internet do, we have a remarkably poor understanding of what behavior emerges when those pieces are connected, and how we might ensure Internet-wide behavior that will be desirable.

**Do you think that we can solve these problems simply by improving scalability and all other kinds of aspects, or do you think there is a need for some revolutionary change? You mentioned a change in the whole architecture.**

Revolutionary change is needed. We need to develop qualitatively different approaches. I would make an analogy with macro-scale physical phenomena, like fluid flows and heat. Understanding the behavior of individual atoms or molecules does not explain things such as turbulence or temperature. And we don’t derive those properties from first principles—instead, we have additional sets of laws for the different levels of abstraction. I believe that when we start looking at large distributed systems as networks, as opposed to as interconnected hosts, there will be new abstractions and new laws that let us predict behavior and do so in a way that is not directly related to first principles. We need laws for the various levels of abstraction that exist in a distributed system, and we don’t yet have those laws.

So solutions such as systems of systems and hierarchical organization may or may not work, subject to what jump in level of abstraction we make?

Exactly, although whenever I have heard the term systems of systems, it’s been uttered in a way that’s led me to believe the speaker understood that a system of systems was a different animal—a different level of abstraction—than a system. These speakers are suggesting we’d need new laws and new abstractions. But certainly there are people who take a reductionist view. They believe that one can decompose systems and understand system-wide properties from properties of components. I believe that approach is no more realistic than trying to derive the properties of modern materials from the properties we know of the hydrogen atom.

**What are the trends of distributing computing systems?**

The biggest trend I have noticed is a movement toward applications. Distributed systems are an outgrowth of research in operating systems and system software—work concerned with defining common functionality for use by all applications, and implementing that functionality in a way that achieves good performance for all. But, increasingly, the semantics of individual applications in a distributed system must be considered and supported individually. Application-specific functionality becomes more important than broad functionality. For example, electric power distribution is changing in the US due to deregulation, which introduces competition. And the way to compete is to control your power distribution network using some sort of distributed system. That distributed system must instruct power generators to produce enough power for the loads, and it must do so in a way that is trustworthy. But in addition to security and fault tolerance, the control theory for the distribution grid becomes an issue for the distributed system designer. This control theory is specific to the application (power distribution), and what types of control it can exert depend on what qualities of service the distributed system implements.
As a second example of the importance of applications, we can see the Web as being an application driving work in networking. Web(-page) caching, for example, wouldn’t attract interest if it weren’t for the Web. Third, some security problems make sense only in connection with particular applications. Supporting the Principle of Least Privilege requires knowledge of what each application is doing, so that suitable rights (and no more) can be granted to that application.

All that said, I do not believe we’re moving to a mode where distributed systems researchers and practitioners become even further compartmentalized (along application-specific lines) within the distributed systems community. Rather, I believe that our community will use specific applications as an excuse to look at a vertical slice of the picture and then take a step back and have a discussion of what needs to be done differently given the new, bigger picture.

One concern I have is from the practical perspective: who would be the entity to take this broader step back and look at a potential solution? My concerns are related to economics and also to the widespread use of computers, making it very hard to make any changes, for example, to protocols or distributed solutions. I am putting in a bit of a practical spin.

The computer-systems research community tends to pay more attention to their principals than to identifying new principles. The leaders in the field (with a few exceptions) build a system, report on that experience, and move on to the next. Clearly, particular distributed applications would be prime candidates for those artifacts of study. Thirty years ago, system researchers scoped their area quite broadly: computer architectures, operating systems, networks, and databases were all considered part of systems. And one finds the same principles do appear in different guises in all these areas. The situation need not be different today. There is no reason why researchers couldn’t take a broad view yet, at the same time, study a series of applications to acquire that broad view.

I worked awhile on the next US air traffic control system, and I learned a lot about air traffic control, but I was also able to make fault-tolerance contributions (applicable beyond air traffic control systems). More recently, I have been working on electric power distribution and, although I have learned something about electric power distribution, I am working to make contributions with broad application to distributed systems. For me, the application becomes an excuse to exercise discipline-specific expertise and extract broad principles.

What about the large embedded infrastructure base? How well does any work you do have to play in that existing infrastructure?

There is good news and bad news. Considerable leverage comes from interoperability, so it makes sense to build things that interoperate. The good news, then, is that we live in a world that is almost a monoculture: a small number of hardware processor architectures, operating systems, and protocol stacks are prevalent. Interoperability abounds. And, because of Moore’s law, every five years the entire content of a heavily loaded desktop PC will fit in 5% of the new machine—every five years you have a completely empty canvas to paint. It is a canvas that has all the functionality of the old systems and yet you can still paint an entirely new picture. Lots of functionality can be added without sacrificing what is already there.

The bad news is that we must work within the constraints of the existing base, and that often prevents revolutionary change.

What is the impact of the fact that operating systems don’t evolve at the pace we would like? There is Unix, which has been here already for 20 years, and then NT, which seems to be remaining, and networking stacks do not change. I’m curious about the growing needs for new distributed systems solutions.

It is certainly true that desktop systems are evolving quite slowly. But the main elements of our
distributed systems are not always going to be desktop systems. What people are excited about in
distributed systems laboratories is investigating systems comprising palm-top class devices,
portable sensors with local processing, and Java rings. Those platforms are quite different from
today’s desktop systems.

Distributed systems built from these newer, small devices will need new solutions. For example,
we’re used to naming all the hosts in our distributed systems. When our distributed systems
comprise many small devices (for example, a processor in each light bulb—although I don’t think it
will go that far), we won’t be able to or be interested in naming each host. It might be interesting
to talk about the light bulb in this room or the next, but I doubt we’d be willing to pay the cost of
giving each light bulb an IP address. With such a significant change to the rules, problems that we
think of as solved today will again become hard, and problems that we have worked hard to solve
might no longer be relevant. For example, routing could once again become an interesting
problem. Migration of computation may take on a new meaning in settings where there are
embedded devices. There will be more transient connectivity when we get into the world of
wireless devices. Changing functionality as a function of available power will become important. So,
the action in distributed systems will move on, and the evolution of our desktops will not impede
research.

What are the most important technologies for the future of
distributed systems?

As I said earlier, I believe that applications will become a bigger driver than they have been. Also,
the new modes of networking—such as wireless and ad hoc networks—will change the rules. I see
programming languages as having been neglected by the operating systems community, but that
neglect is undeserved and won’t last much longer. Java has penetrated the operating systems
community in the sense that researchers are worried about what systems problems it solves or
creates. We see new security solutions that arise either from analyzing programs (for example,
proof-carrying code) or from modifying programs and rewriting them (for example, the in-lined
reference monitor work done here at Cornell). Program analysis and program rewriting solutions all
are based on programming language technology.

We haven’t mentioned real-time and resource management. It has been somewhat avoided in the
distributed community as a very, very hard problem. Do you see an increased need for the support
of resource management—not just for quality services, but in general?

I think that quality of service is getting short shrift, and that has been a mistake. Implementing
quality-of-service guarantees is extremely important if we’re ever to build systems that are
trustworthy. If you want to build a system that is guaranteed to deliver some service—because it is
controlling air traffic, or the power grid, or a financial market—you need both components that are
guaranteed to deliver service and glue to combine those components. But if you start with
components that don’t have service guarantees, your only recourse is stochastics, and probabilistic
guarantees are all you will get. Probabilistic guarantees may suffice in some settings, but not in
others, and systems intended to control critical infrastructures may require stronger guarantees.

The Internet is built on a religion of statelessness. Internal nodes don’t manage per-connection
state. This is a real strength of the Internet’s architecture, because statelessness facilitates scaling.
But the religion of statelessness prevents using certain kinds of resource-reservation schemes and,
consequently, prevents implementing certain kinds of service guarantees. There have been
proposals to implement quality of service by keeping state at the edges of the network (which is
permitted by the religion of statelessness) or by doing things statistically. None of these proposals
have been deployed, and we don’t know what guarantees they can deliver. If the guarantees are
not strong enough, we’re not going to support the trustworthiness that some distributed
applications need.
What are the roles of academia in the advances of distributed systems today compared to the past? Will startups and the gold rush permanently impact the balance of research and development, or is this just a temporary trend?

Startups and all these IPOs have certainly had a significant effect on many computer science departments. Systems faculty and students are the most likely to be attracted by the gold rush, because for them there is the most money and opportunity. One hears about students attending graduate school not with the expectation of getting a PhD, but with hopes of getting involved with a faculty member whose startup will make them rich.

And the culture in academia is very different today than it was 5 years ago. Faculty today are more inclined to talk about seed money and venture funding, whereas in the past they would talk about grant funding. And faculty are reluctant to talk about the details of their research, hoping to transition that research into a startup. There is just not the sort of openness that there used to be. Ideas are rushed into companies instead of into print, and that means there is no longer the kind of discussion and criticism that fosters scientific progress. It is really unfortunate.

On the other hand, the gold rush is certainly a driving force—people are being driven to make advances, and they strive for advances that they believe will be marketable. Unfortunately, I think this situation aggravates the schism between theory and practice, because the problems theorists are working on are not as directed by practical concerns as they could be if systems researchers were available and willing to interact with their theory colleagues.

Do you think that this situation is long term? Or do you see that it will revert back and improve?

I don’t think the gold rush will abate as long as Moore’s Law holds, because that law provides the opportunity for new functionality to be deployed in our computers without compromising old functionality. Faculty will succeed in their startups, get rich, and computer science departments just might end up being populated by a new class of gentlemen professors—that is, people who are faculty because they want to teach students and not because they need to make money. And computer science departments have long had multiple classes of faculty. For example, for the last two decades, research funding levels have varied across different research areas. Systems faculty typically have—and need—higher funding levels than theory faculty. Computer science departments overcome such class distinctions by being careful about how internal resources get apportioned. I see no reason why this approach wouldn’t work with a new class of gentleman professors.

What is the model of today’s researchers? Can faculty today succeed and have an impact?

There are many examples of systems faculty having impact. In fact, given the current industrial climate, I think it is more likely that something significantly new would emerge from a university setting, where a researcher started with an idea and no expectation of commercialization. Companies are under too much pressure to produce short-term results. Today’s corporate research staffs are not given the free reign their predecessors in the early eighties had. It is at the universities where the flexibility to take real risks exists. So, I would be surprised if a company came along today and revolutionized the computing paradigm the way Xerox did with the Alto and distributed computing. I think the development of such new paradigms is much more likely to happen today in a university. And I think the area where such new paradigms will next develop is nomadic computing—wireless, small devices. Such devices are easy to experiment with and cheaply.

http://dsonline.computer.org/archives/ds100/ds1intprint.htm
to buy. And research doesn’t require large investments in infrastructure.

**Do you have anything to add with respect to any specific area of your experience?**

I would comment that a popular perception is that systems work is about performance. It isn’t. Systems researchers are concerned with how you might combine a collection of components with given characteristics to obtain a system with some interestingly different characteristics. Dealing with performance mismatches can be a significant piece of this picture. But, increasingly, the interestingly different characteristic is going to concern qualitative dimensions, such as security and fault tolerance. These subjects must now emerge from their niches and become mainstream systems topics.

**What about the reliability of the systems, networks, and solutions?**

*We are talking about the use of computers in hospitals, where reliability is of extreme importance. Do you see any significant advantages? I mean, we can’t just provide solutions.*

We mustn’t deploy insufficiently reliable systems in life-critical and mission-critical settings. I disagree that we can’t deploy our systems in such settings, because we are doing so increasingly. But I do agree that such behavior is irresponsible.

I think most folks don’t appreciate just how untrustworthy the computing systems we deploy are. And there is unfortunately no incentive for people who discover system trustworthiness problems to make these problems broadly known. Neither the financial institutions nor the military, for example, have anything to gain by widespread reporting of a compromise of their systems. Without widespread reporting of system compromises, citizens can’t build intuitions about what can and cannot be expected of computing systems in the way of trustworthiness. Today, fault tolerance—hardware fault tolerance—is extremely good. But there are lots of bugs in software, and that generally leads to system failures and the creation of vulnerabilities that allow successful attacks. Furthermore, there is a real structural problem in the industry. Software producers do not have an incentive to produce and distribute higher-assurance software.

Feedback? Send comments to dsonline@computer.org.

This site and all contents (unless otherwise noted) are Copyright ©2000, Institute of Electrical and Electronics Engineers, Inc. All rights reserved.