Great Educators in Computer Networking: Bruce Davie

Bruce Davie
VMware
bdavie@vmware.com

Matthew Caesar
(interviewer)
University of Illinois at Urbana-Champaign
caesar@illinois.edu

This article is an editorial note submitted to CCR. It has NOT been peer reviewed.
The authors take full responsibility for this article's technical content. Comments can be posted through CCR Online.

KEYWORDS

Education, Computer Networking, Network Virtualization, Software-Defined Networking

This interview is part of a series on Great Educators in Computer Networking, where we interview some of the most impactful and skilled educators in our field. Here, we interviewed Australian Bruce Davie, the self-described computer scientist/engineer/runner/cyclist, who agreed to talk to us about his thoughts on computer networking education, his role in it, his thoughts about the big ideas in our field, and how the pandemic is changing our work. Bruce has over 30 years of industry experience and is well known for a broad spectrum of educational initiatives such as co-authoring several textbooks, as well as his contributions to many networking standards and technologies, including IP quality of service, network virtualization, software defined networking, and more.

Bruce, what do you think about textbooks these days? Do you think textbooks have a future given how much education seems to be moving online?

I'm a pretty big fan of the textbook for a few reasons. I think the thing that Larry[1] and I have tried to do from the very beginning is to give people a sense of perspective about what is and is not important, I think it's really hard to get that perspective if you just go Googling around, seeing what's online. And so there's lots of ways to get information, but I think the benefit of a textbook is that it's been put together by people who have some experience and have a viewpoint on what's important in the field. You're going to get something that is different from just say, looking something up on Wikipedia or googling Stack Overflow, or whatever other kind of ways you might try to get the information. And so, speaking for myself, I'm always trying to find answers to questions that are available online, but the problem is that when you try to find the information online you struggle sometimes: Is this authoritative? How do I know this person's perspective is appropriate? So I think for me that's the big thing that a textbook brings.

Another thing is, very concretely, Larry and I very strongly believe in this systems approach. Our idea of the systems approach is that you need to look at how all the pieces of the system fit together and you can't simply look at little individual boxes in isolation. I think that's one of the areas that if you just go around looking at stuff online you'll often get that very isolated view of things. So you can understand how some part of congestion control works but you don't understand how it fits in the bigger picture of end to end application performance. So that's really the thing that we try to do and we've tried quite hard to cut stuff out because that's part of what it means to give perspective. So LAN emulation on ATM[2] is probably not that important for people to understand anymore; it's interesting history but it's not teaching you anything fundamental and so let's not have the textbook just tell you every single fact about networks for the last 30 years.

Very few educators have such extensive real-world experience with computer networking. What are universities doing well in terms of computer science education? What could they do better?

One of the things that I found interesting when I was at Cisco was a lot of people didn't have much networking education. They typically had a software development background more than they had a networking background, so a lot of people were trying to learn networking while they were in industry as opposed to having learned it at a university. That presents its own set of challenges because you'd kind of like people to understand what they're doing. It makes a huge difference if you are thinking about it in the context of industry. You want somebody who is working on a feature within a bigger product to understand where their feature sits within that overall product as opposed to just thinking "my feature sits in a box and I make it conform to the specifications, but I don't know how it fits in the bigger scheme of things". I would generally like people to have that bigger picture view so I think that to the extent you teach that at university, that will help them when they are going to industry.

For example, I'm actually quite intrigued by the way they[3] taught networking when I was at MIT. It came pretty late in the undergraduate computer science education curriculum because they taught systems as its own separate class. So I don't even know if there was an undergraduate networking course. You would have gotten networking as part of a systems course as opposed to having a dedicated networking course. I co-taught the graduate networking

[1] Larry Peterson, co-author with Bruce Davie on "Computer Networks: A Systems Approach,"[1] one of his widely-published textbooks. https://www.cs.princeton.edu/~llp/

[2] LANE (LAN Emulation on ATM) is a protocol that lets an ATM network function like a local-area Ethernet network. ATM (Asynchronous Transfer Mode) is a layer-2/layer-3 protocol which was widely viewed as a competitor to IP in the 1990s but is no longer very widely used. https://networkencyclopedia.com/lan-emulation-lane/ , https://en.wikipedia.org/wiki/Asynchronous_Transfer_Mode

[3] Bruce also performed instructional work himself when he was at MIT, including teaching the graduate networking class (6.829 Computer Networks) with Dina Katabi and Hari Balakrishnan. http://web.mit.edu/6.829/

course with Hari Balakrishnan and Dina Katabi, but that was a graduate level class. But I believe for undergrads in those days they were still using their intro to systems class as a way to get people thinking about networking, and I actually really like that approach when you think of networking not as its own special field but as a particular type of system. You learn about things like scalability, which is one of the fundamental properties of systems, or learning about separating policy from mechanism[4], things like that, so there's a whole lot of general principles that apply to systems that are broader than networking.

If I could extrapolate a little bit into my experience at Nicira[5], we had more people with a distributed systems background than we had people with a networking background. You know obviously at Nicira we were trying to do something that was quite different than what was being done at traditional networking companies, because we were a startup trying to disrupt the networking business. So we had people who understood networking, but we had a much greater concentration of people who were systems people and, in some cases, not a lot of networking background. I think that was really positive because first of all, it gave us fresh perspectives—there's research that shows that teams with diverse perspectives produce better outcomes. The fact that we weren't just a bunch of networking people probably helped us do the kind of disruptive things we were trying to do. I know there's a lot of stuff in that answer but I tried to boil it down to what you need to teach people at university to help them succeed in industry. It's kind of that—knowing how all the different pieces fit together and having that broader view as opposed to "this is what the seven layer model[6] looks like", you know?

One other thing that I'll just highlight again from Larry's and my viewpoint that I think also touches on how you might teach at universities is that something like the seven layer model is often taught as if it was "handed down by God"; as if this is the way networks are supposed to be—but of course it's something that was made up to help people to understand networks long after networking had been established as a field. It's a pedagogical tool; it's not the answer to how networks are supposed to be built. And so again, I think we very much wanted people to learn what are the principles that underlie networking? And maybe those principles lead you to the seven layer model, or a different set of people will tell you that it leads to a four layer model, and some people will tell you (and I would probably sympathize with this view) layering is only helpful for some percentage of the time and there are times when you really need to not use a layered model at all and have a different model for what you are trying to do with networking. So, all of that makes me think you are trying to teach people a set of fundamental principles that go much broader than "this is how networks have been built by the previous generation".

[4]The idea that "mechanism", or the system implementation, should be separate from "policy", or what the system operator/user would like the system to do, is a principle which has proven useful in the design of many computing systems. https://en.wikipedia.org/wiki/Separation_of_mechanism_and_policy

[5]Nicira was one of the first companies to really make substantial inroads in getting network virtualization into industry. VMware acquired Nicira (and Bruce Davie's employment) in 2012 for $1.26 billion.

[6]The Open Systems Interconnect (OSI) model, which is composed of seven "layers", is commonly used by instructors as a mechanism to describe how the various protocols in the internet fit together. https://en.wikipedia.org/wiki/OSI_model

I like how you are talking about these fundamental principles. If an educator wanted to give a list of these fundamental principles, where would they go?

Ha ha, yeah, so we did try to highlight some of them in the book but I don't think we've done it in a way that makes it easy to find them. This is an idea for another book we should do because, like the end-to-end principle[7], is probably one of the most widely cited fundamental principles of networking and of course it's actually a systems principle, not a networking principle, but it's mostly referred to by networking people. So that's one that we've definitely highlighted in the book. Separating policy and mechanism is another one we highlighted. In the early edition of the book we had this idea of trying to make these principles very clearly visible, and making them pop up in various places but you couldn't just pick up the book and say "Oh, these are the seven fundamental principles of networking". That would be an interesting challenge, but I'm not sure that you could come up with a definitive list. There are a few out there that are very well known like the ones I just mentioned.

That's interesting. Do you have an opinion of how much students should be exposed to the details of systems? How deep do you think networking instructors should go with their teaching? I know there are various trade offs there.

I think you only really want the details to some extent. They are an artifact of the way things have played out. If you're an engineer trying to implement IPv6 forwarding in a router, you need to know all the details of IPv6, of which there are quite a lot. And so for somebody like that you need everything. But the average person is never going to do that. It's really helpful to understand why IPv6 came about and what was the fundamental problem with IPv4 that caused us to have to come up with IPv6—that teaches you something about scalability. It teaches you something about designing for the future and being able to predict where your system might end up. I'm pretty sure Vint[8] said there were two mistakes he made in the IP—one was that the address space was too small and the other was no security. Ha ha, but aside from that it was great. So, I don't even know if you'd call them mistakes, but anyway those are the things that we are living with, of course.

And yes, I wouldn't necessarily want to teach people all the gory details unless those details help to illustrate some broader point. Say you're that hypothetical engineer trying to implement a forwarding pipeline—if it's been done properly the set of RFCs will tell you all the details you need to know. So part of what you're teaching people in a networking class is how to understand an RFC and how you know when it's time to go to look at the RFC because ultimately the very low level details are going to be there. And there's another part of this too, I think, that after you've read the RFCs on IPv6 you probably have actually no idea how to implement an efficient forwarding pipeline. That's all about algorithms, which realistically have very little to do with IP and have a lot more to do with search algorithms and that sort of thing and so you also have to give people what they need to know. Because if I wanted to find a really efficient forwarding algorithm for IPv6, where would I go

[7]See the paper "End-to-End Arguments in System Design" [3]

[8]Vint Cerf, developer of TCP/IP and widely considered one of the pioneers of the Internet. https://en.wikipedia.org/wiki/Vint_Cerf

to find that? It would probably require looking into some Sigcomm papers and so this is kind of the "teach a man to fish as opposed to just feeding him for a day" argument. You want to give students the tools so that when they are faced with a new problem, they know how to go tackle that problem. You can't possibly give them everything they need to succeed, but you can give them a set of tools that will help them know that this is the time to go to the RFC, this is the time to go to an academic paper to find an interesting algorithm, this is the time to go to an algorithm textbook. I should probably do a shout out to George Varghese[9] because his book on network algorithmics is really a good one in this space because it tells you how to think about network algorithms. He's one of the people who's probably published more papers about algorithms to improve networking than almost anyone I can think of. What he tried to do in his book was not to say "Here are the algorithms" but rather "Here's how you think algorithmically", which is quite a challenging thing to teach. That's ultimately what you are trying to teach people, how to be creative enough that you can create algorithms or at least how to know when you should go look for one and see what someone else has published.

There's a bunch of concepts that are taught in networking, like congestion control, quality of service, and things like that. Do you see any new concepts emerging from your industry perspective?

Yeah, I've got a pretty strong view that we've moved to a world where distributed systems and networking have kind of converged. We're in a much better position, I think, in terms of what's possible with networking now than what was possible 10 years ago. Software defined networking is one outcome of that blending of distributed systems with networking. Networks now have APIs where you can specify what you want the network to do as opposed to the old model of networks where you had this extremely low level distributed system where you had to go and configure every box correctly, and if you did it all right a sort of magic would happen and the routing protocols would converge and the internet wouldn't go down. But if you got a single box wrong it was chaos. I literally just sent out a tweet yesterday[10] about an outage at Cloudflare that was caused by a single misconfigured router and my takeaway is: proof again that networks should not be configured by humans. So the way I see things going is that, and this is really what SDN brought us, you don't configure boxes. You express what you want the network to do through higher level APIs that can then get mapped onto low level implementation.

So in the case of SDN systems that I've worked on you can say "I need these 5 virtual machines to be on the same L2 segment, I need them to be connected to a router, I need them to have some firewall policy—these are all the things that I want.You specify all those intents through an API and then underneath the covers, a bunch of distributed systems technology takes my intent and maps them onto an implementation that is programming a bunch of forwarding tables somewhere. A virtual machine could move and all of the forwarding behavior that it needs moves with it. So all of this to me is how networks become more and more things that are controlled by software, not humans; things where the primary way to interact with a network is through an API and you don't interact with the individual boxes or forwarding elements. You interact at a higher level with the network itself. Whether that's your wide area network where you say "I've got these 10 branches and they need to be connected in a private mesh which is like software defined WAN" or you take Google's B4[11] where you say "we've got this big backbone that connects all of our data centers together and we need it to provide connectivity between the data centers but we have centralized control of it to make sure it balances out the traffic—we don't go configure individual routers to do load balancing". There are countless examples and so this is sometimes talked about as "intent-based networking". Unfortunately, every time you come up with a good term for some new approach to networking it is co-opted by people in industry, by someone who starts slapping it on their products trying to declare success. The real high level point is that we move from box level networking to network level abstractions and then a whole lot of new things become possible—things like network verification. You can say at a high level "my network should do this" and you can now go down and see "does my network do that?". So you compare the thing you had specified as your intent to the reality and see whether they match. That's been another great trend in the last half dozen years. And then networking becomes increasingly automated. Look at how most people experience networks today, well particularly take people developing applications. By and large they don't think much about networking—they build a bunch of microservices and they specify that those microservices have to be able to communicate with each other, and under the covers all the networking happens. And again, I actually think Kubernetes is really interesting as a system that embraces this same idea that you specify what you want, you say what your intent is—I need 10 copies of this microservice and then Kubernetes makes sure that the reality matches your intent. It's actually a really nice fit between this new model of how we deploy applications matching up to this new model of how we deploy networks. And so you'll see lots of work going on to make networking systems that fit well into these new modern application frameworks.

You've written several textbooks. Do you have any interesting stories about them? Are you able to tell us anything about how they came about?

Yeah, I've certainly got some good stories here. So first of all, Larry got the initial contract to do our textbook without having talked to me about it. He figured that he was due for a sabbatical and decided to work on a textbook; I don't know how many months in he was when he realized that he had kind of bit off more than he could chew. He and I had been collaborating for several years at this point on this research project on gigabit networks and I think we had done a Sigcomm paper together so we knew we could work together. He reached out to me and said "Do you want to help me on this textbook?" I had done one book prior at that point and I was definitely interested in doing a networking textbook and so I jumped on that opportunity. So we worked on that first edition

[9] See the book "Internet Algorithmics: How to Build Fast Routers and Servers," George Varghese, 2004.

[10] https://twitter.com/_drbruced/status/1285499195948888064

[11] See the paper "B4: Experience with a Globally-Deployed Software Defined WAN"[2]

together. It was just an absurdly large amount of work and while Larry was on sabbatical I think, I had a day job. I probably did a quarter of that book, but I was working evenings and weekends for a long time.

So that was the first edition and then every edition subsequent to that the publisher would come back and say "It's time for a new one", particularly 2nd and 3rd editions, there was a ton of stuff we wanted to improve. We had the book out long enough to figure out there were some big gaps, like we had nothing on security in the first edition. We had nothing on wi-fi because there virtually was no wi-fi in the early going. So big chunks we had to write in the 2nd and 3rd editions. Plus, we were learning what worked for people in terms of instructors using the book. So, one thing I remember, the first book had a pretty heavy dependency on x-Kernel[12], which, you know, Larry quite rightly considered to be a great teaching tool. But it was very specific and not every instructor wanted to go and invest the effort in getting x-Kernel up and running, and so in the 2nd and subsequent editions we kind of downplayed the x-Kernel part of this. We still used code snippets to illustrate things, but we didn't have such a big focus on x-Kernel to try to make the book more accessible.

And then the other thing I remember very vividly, is I wrote most of the first security chapter, I think, and I literally knew nothing about the topic when I started. This was one where I just went and read RFCs, and I read other people's textbooks because we didn't have Wikipedia. I just read everything I could find and then wrote what I thought was important, but I was definitely not writing from a position of expertise. And I just completely mangled the description of message authentications, I mean not mangled badly, but it was an obvious flaw in what I wrote that got pointed out as soon as the book was in print. It was one of those things that was like "Oh right, so you can't actually become a security expert that quickly". I've used that story over the years to illustrate how long it takes to be an expert at something because I reckon it's, I don't know, maybe 10 years, to get expert in any topic. Oh, this is probably Malcolm Gladwell's 10,000 hours[13]. Even before I could consider myself a routing expert, I reckon I was probably in the field for more than 10 years. I was working alongside pretty legendary people like Tony Li and Yakov Rekhter who wrote BGP and so you have to be pretty brave to call yourself a routing expert when you are around people like that. It takes a long time to get really good at that stuff and when you are writing a textbook you are going to have to write about some things you are not an expert. That's one reason you have reviewers and thankfully, reprints, and newer editions when you make mistakes.

Do you have any observations in terms of how people use your textbook?

Probably there's not a lot I can say there. I think one thing people have comments on with the book is, not everyone loves the order that we go through the material. And this is something that comes up a lot—the top-down versus bottom-up approach[14] to networking. We wrote the book absolutely bottom-up and around the same time that our book came out, Jim Kurose's book came out going top-down. I think his book was more successful than ours, which may or may not be due to the top-down approach, but I think they both have merit. Some people look at our book and think "I can take this book and teach through it in a top-down way" and in fact, I think we even wrote something in one of the editions in the preface saying here's how you can use the book to go top-down. The reason why we came bottom up was partly because of how we learned networking. My first experience using e-mail was 1985 and my first experience using FTP was probably 1988. Bear in mind that the world wide web didn't exist until 1991 and it didn't really become popular until about '94 and so we were people who experienced the internet in its kind of early days with regard to applications. We didn't think about the internet primarily from an application user perspective. We thought about it more from the perspective of the people building it. I was building ATM network interface cards and things like that. Our experience was kind of bottom up and so we wrote about it from our experience, but by the time our book came out in 1995, millions of people were experiencing the internet through the world wide web and in many cases that was the internet for many people. The natural way to explain networking to this new generation of people who were maybe 10 years younger than Larry and me, was to explain it through applications; hence, the top down approach made a ton of sense. So I think that's one area where I reckon probably most people who use our book go bottom up with it, cause it's just so much easier to take a book in the order it's written. I'd be delighted to know if there are more people using it top-down because we did have the view that it could be taught that way. There's no doubt if you want to go top-down, ours probably isn't the best book.

One of the things we did have quite a strong view about, and we said this pretty clearly, is that layering is not your master. Layering is a tool. It helps you understand the way networks work and it can help you separate one part of the system from another while you want to focus your energy on one part of the system, But ultimately you have to think about the whole system and the whole system goes from top to bottom so we wanted people to think about the network not really top down or bottom up. We kind of wanted people to think about it end to end but inevitably you've got to cover all the material somewhere. If you actually look at our book there's long entire chapters that are not related to any layer. Congestion control is an example where it's not really in a layer; it's in layer 2, and it's in layer 4, and it's in the application layer, and so that's one example where we pulled something out and made it into its own chapter; security is another example that cuts across layers and gets its own chapter. There's a bunch of things that we felt are best not treated in a layerist way, but the fact is you've still got to help people get there somehow. The other thing was really trying to go through a series of mental exercises like what's the first thing you'd have to solve if you wanted to build a network?

[12] See the paper "The x-Kernel: an Architecture for Implementing Network Protocols". https://www2.cs.arizona.edu/projects/xkernel/

[13] In the book "Outliers", author Malcolm Gladwell mentions a "10,000-Hour Rule", which claims that the key to becoming an expert is largely a matter of just practicing the right way, for a total of around 10,000 hours.

[14] Bruce is referring to the fact that computer networks are often taught by making a single traversal either up and down the OSI model. "Bottom up" means you start at the lowest level (physical layer) and successively introduce higher layers one by one; "top down" means you instead start at the application layer and iteratively teach layers down until you finish down talking about physical layer protocols.

Then what is the next thing to solve? How do you solve that? That was more the things we were trying to do, not to say "well, let's start at the bottom because that's the way it's always been done" but rather, what's the first problem you have to solve? Well, I have to connect 2 things together;I need something physical to connect them together. There you go, physical layer. What do I have to do now? Well, I should make sure the bits are intelligible at the other end. OK, encoding. What's the next thing I need? Get the bits there reliably. Ok, so framing. We work our way through all these problems and eventually you end up with applications. Anyways, I'm not really answering your question at this point. I'm just giving you my philosophy on the book. There is definitely no right way to use the book, that's for sure, and I think we're generally happy if people are finding ways to use it teaching people the systems approach, teaching people to think about the big picture, teaching them to think that just because this was done in the past doesn't mean that's the right way, it's just this is the way it was done. What are the principles that led it to be done in that way? How would those principles be applied going forward? Those are the kind of things we want people to take out of it.

Is there any advice you have for somebody who wants to write their own textbook?

Ha-ha, make sure you've got a lot of free time. So, I think the most important thing is to make sure you've got some fresh perspective beyond what's already out there. When we wrote our book there were a couple of other books out there and we felt that the existing books were either very strict about layering to the point of not really reflecting the world as we experienced it, or were ignoring the internet, you know, which was fair enough because there were lots of other networks competing for our attention. By 1995 we felt that the internet was going to be the world's most important network. That obviously turned out to be true, but it was less obvious in the early 90's that that was going to be true. And then the other thing was that the encyclopedic approach was pretty popular. So the idea was any given networking textbook would have a little bit of everything but you couldn't necessarily tell what was important. So we kind of wanted to have this sense of perspective that you don't need to know every single thing about how bit encoding works but you need to understand why bit encoding matters and then if there are various encoding schemes that we haven't covered, you can go and look them up somewhere. So this idea of don't be an encyclopedia; give people a perspective on what's important. Those were the things we felt were our fresh perspective. If you've got a fresh perspective that's different from what's out there in existing books, that's a good reason to do it. I guess the other thing I'd say is the idea of having a co-author does go a long way both in sharing the workload, but also keeping you honest. For example, I remember putting something in the first edition draft that was very encyclopedic and Larry kind of called me on it and I said "Yeah, you're right. I didn't really pick out the perspective on why that's important", so let me just go trim that section and make it more consistent with our overall viewpoint—here are the important takeaways from this technology as opposed to here's the quick summary of the RFC, which was what I was a little bit prone to falling into.

Related to that, one of the things about you is that you have a very interesting background for someone who is so passionate about education. A lot of people who are interested in teaching take the university track in a professor job, however you pursued industry labs and things like that. Is there a reason for that? What are your thoughts about industry as a nice platform for teaching?

Yeah, it turns out it really is. I don't think I would have fully had this perspective even at the beginning of my career, but I absolutely love teaching and I literally was talking to my wife this morning about this. I just recorded a video on quantum computing[15] which I don't really know that much about, but I know enough about it to help people get a bit smarter about it. I've invested a huge amount of time just learning enough to be able to give a half credible talk about the subject. And I just absolutely love doing that. You find audiences always hungry for information when you're in industry so it is a great platform for people who like teaching, provided you can find a way to make that part of your job in industry. I definitely didn't have a well formed plan early on. I will say, when I finished my PhD, there was no part of me that thought that going into academia looked like a good idea. I didn't study networking in my PhD. I did hardware description languages and formal methods, but did not get really passionate about that topic. The natural thing if I'm being kind of academically inclined would have been to try to apply for a research position furthering the research I did in my PhD, which I had zero interest in doing.

What I was able to do was to land this research job in Bellcore[16] based purely on the fact that I apparently knew something about how to do computer science from having done a PhD. I knew nothing about networking and thankfully Bellcore took a gamble on me and that sent me on my way in my networking career. The big decision point that I reached when I decided that it was time to leave Bellcore was, do I go for another research job or do I go for a more industrial job? I actually applied for both, and there were a couple of teaching jobs that I applied for. At this point I was seven years out of my PhD and my resume probably didn't look that great for somebody trying to hire a professor because I didn't have a great publication record for somebody seven years out of school. I had an ok publication record but not a strong academic one; I had no teaching experience, so I kind of looked like competing directly against somebody straight out of school, but I had this 7 year period in industry. So I wasn't actually looking like a great fit for academia at that point so nothing came of that, but I did get job offers from a couple of places that built routers. And in 1995 going to a place that built routers looked like a pretty good idea. The big question that I had was "Am I giving up something by going from research to industry?"

One of the ways that Cisco made me feel good about the move was they said "We'd like you to build bridges between Cisco and the

[15] See the talk "Quantum Computing: How Will It Impact the IT Industry?", Bruce Davie and David Ott, VMworld. https://www.vmworld.com/en/video-library/search.html#text=%22quantum%22&year=2020

[16] In 1984, the US Government broke up ATT, which had a monopoly on telecommunications in the United States, into a number of smaller companies (called RBOCs or "Baby Bells"). The RBOCs needed some place to coordinate their technologies and standards—the Bell Communications Research company (Bellcore) was formed to provide this function.

research community, since you were in the research business. So, do you think you could come to Cisco, work on our products, and also build bridges into research?" I was like, well that sounds great. Consequently, I've had this kind of lengthy career now where I've had one leg very firmly in the industry side and maybe a toe in the research side and I've been able to maintain that throughout my career. Where I've been, people have been supportive of doing things like writing textbooks, being involved in Sigcomm, doing all the things academics will do. I had support for doing teaching at MIT when I was full time employed at Cisco. So I've had that opportunity to straddle the two worlds, and I think what's worked really well for me in industry is , I may not be the most inventive person in industry, but I'm pretty good at explaining stuff to people and that goes a long way. A CTO is a person who can articulate a technical vision. You don't necessarily have to create the vision, although that can be part of a CTO's job, but it's really great if you can explain the vision to people. Those people could be your co-workers, they could be junior engineers, they could be the CIO of a customer—a pretty broad range of people, but a lot of that is variations on teaching. So I guess that's how I've seen an industrial career play out.

Related to that, I was wondering if you had any insights into education as related to RFCs. What is difficult to express your thoughts in terms of RFCs or are RFCs the right mechanism? Are you able to talk at all about how you educate consumers?

One of the things I always say to anybody who's asking me for career advice is to get better at your verbal communications. So that means writing and speaking. Being able to write an RFC is definitely a skill. The number one thing you need to write a good RFC is to be able to write proper grammar and express your ideas clearly and, of course, you have to understand the technology you are writing about. I think RFCs are all over the map in terms of their quality. A good RFC will definitely be a good way of getting an idea documented so that other people can implement it. I think the process by which RFCs get built today has gotten a bit heavyweight. There was definitely a more lightweight process in the early days of the internet where it was a smaller community of people. Often somebody had an idea, they put together an implementation, and then they documented their implementation. Other people could now go and implement something that would interoperate with that other implementation. I think in the early days RFCs worked really well. These days they tend to be more designed by committee with all the drawbacks of that. Not that that's a bad thing necessarily. You kind of want to have some sense of democracy in how standards get built, but there's a lot of overhead today. I think this is the whole question: How do you educate the people who are your target audience when you are in industry? With RFCs you are really targeting implementers.

With a lot of what I do, I'm often targeting technical decision makers and so a lot of the time I want somebody to understand for example, why should they pay attention to the way I think their work should be done rather than the way that it's always been done in the past? For the last 8 years I've been trying to explain to people, "don't just keep on buying boxes and configuring them by hand. That's not the right way to do it." That's a technical argument that I've spent the last 8 years putting together. I literally did that for a customer yesterday using a very quick summary of my 8 years of history with Nicira and VMware. To say this is how the world looked in 2011 and this is how it looks today. This is how we got there. This is why it is better. Here are all the side effects that were created by this new approach to networking. The whole idea is to take the customer on a journey where they see "Yeah, this would be a better way of doing networking". So, ultimately I want them to come and buy our product, but I actually want them to understand that they will be better off if they change their approach to networking. A lot of the time when I talk about networking with customers I don't mention VMware at all because we are just part of this bigger movement, the SDN movement, the distributed systems meets networking movement, intent-based networking, all of these big areas and trends are important and people who want to improve their systems should be paying attention to those trends. Frequently if you are in an IT organization you don't have the bandwidth to go and learn about trends in the industry. You simply keep doing what you've always done which is, alright, just wait for the next generation of router to come out, I buy that and I configure it and then you know, it hasn't broken badly in the last 10 years so what's the worst that could happen? So that's definitely an education process and I've been to the VMworld customer conference every year since 2013 precisely because I like doing that kind of education of the technical audience.

We also wanted to ask you about your thoughts on different participants in the educational process. You've taught at universities; you've worked with people in industry. Do you have any thoughts on the things different people in the teaching process, for example high schools and K-12 teachers, should do to prepare students for systems oriented careers even that early in the educational process?

I certainly don't want to claim to have a lot of knowledge in how to teach, let's say, K-12. I think that I started getting exposure to computers when I was in, I don't know, middle school or high school. And you know, I was pretty fortunate given the era that I went to school that I could get access to computers. I got interested in them because they were a good place for problem solving, they were logical, they conformed to rules—I was naturally drawn towards mathematics because it's how my brain worked. I liked physics. I didn't do as well in English. I really enjoyed Latin because I thought grammar was really interesting because it conformed to rules, particularly classical languages have really strict rules of grammar. So for me, I think computers were something that I got interested in because I was exposed to them. Nobody was really teaching me any principles of computer science. I just learned how to program. Getting that exposure was a great way of stimulating my interest. Obviously that led to the career that I had. I think if I would be thinking about how I would want to bring that into the curriculum for kids K-12, I would definitely want them to have exposure to computers in a way that makes them think these things are interesting and cool. Often that's done today through letting kids do Minecraft or something that's reasonably creative and I think I'd teach them how logic works in computers. Like the idea

that the computer is doing exactly what you tell it to. Sometimes they don't do what you want; they do what you told them to.

Learning that kind of stuff seems very valuable. And then I think the systems approach stuff—you probably can teach that in high school at least. You can teach people that the world can be divided up into little problems that are well scoped and you solve that problem and don't worry about everything else—that's kind of the opposite of the systems approach. The alternative is that the world is complex and things interact with each other and if you really want to understand how things work you look at how things interact across the end to end system. I'm trying to think about whether I learned any of that in school. I suppose that's something I probably wasn't exposed to until I got further into my university degree at least. To me that seems like they could be taught in some way. It could be taught through a computer science type degree without a strong focus on coding. I look at what people play with. I think that playing with a Raspberry Pi doing interesting stuff seems, at least for some kids, to be very motivating. I think the other challenge is that it's something they feel positive about and they see computing as interesting rather than just something for geeks. It's often about how you tie into something that kids care about—obviously a lot of kids like games—and so that feels like a way that you can make it feel like something that's creative and interesting as opposed to just boring and geeky,which unfortunately, I think too many people still feel that way about computer science.

Did COVID-19 affect you or your educational efforts at all or if you have any thoughts on best practices?

I just mentioned that I normally go to the VMworld customer conference. This year we will be doing this online for the first time and so I'll be pre-recording my talk. In fact, the talk that I just mentioned that I had recorded on quantum computing is an early version of one of the talks I'm doing at VMworld this year. So it's going to be a really different experience for me because I like being in front of a room of a few hundred people and hearing how they react to what I'm doing, having them ask questions and that sort of thing. There's obviously pluses and minuses here. The plus is we can reach a lot of people we couldn't reach before so people who maybe never could afford to go to VMworld in the past now can register for free and get the content. Similarly we've got an internal conference going on at VMworld this week. I've got a short talk in there. It will be available to everybody in the company whereas last year when I gave a similar talk it was only available to a couple hundred people. So I kind of like this democratization of access that's coming about through being forced to do everything online. I think the challenge is how do you get people engaged? I am nervous about asking people to watch a one hour presentation where they have zero chance to interact with me. I always try to interact with my audience and that for me is a key part of giving presentations; so I think that for education in the more classical sense we've got to find good ways to make it more interactive given the challenges of keeping people engaged when they're doing stuff online. So absolutely we are in a different world now in regards to education. I think that, thank goodness we have the internet and that it's working so well and I think that we're going to be doing a lot of experiments in the next 12-24 months of how we make better use of connectivity that we have to reach people.

Are there any final things you would like to say to the networking community?

Maybe I'll just wrap up by saying something about communications. I touched on it in one of my answers, but it's something I'm very passionate about. It's actually the thing I'm talking about in our internal conference this week and I'm going to release that video[17] next week after our conference. For me, communication is so frequently neglected (and I mean verbal communications, not bits over wires!). How you communicate your ideas is so frequently neglected by engineers and scientists and David Clark is kind of my role model on this. I saw him speak in 1990, when he was getting the Sigcomm award[18] and I've been a fan ever since. His ability to communicate his ideas in a way that is both technically engaging and personally engaging is something I've aspired to and I've put a lot of effort into both getting better at writing and getting better at speaking. I feel like that's something that every engineer and computer scientist should be doing if they desire to have an impact. Maybe that's the last thing I'll say too. Part of the reason I've enjoyed being in industry is, on occasions, you really get to see a big impact from your ideas that is not just that my paper got cited a thousand times, but my idea got implemented and deployed into hundreds of networks. I think for me those two things go hand in hand. It's great to see your ideas have impact. You've got to figure out for yourself how to do that, but if you can't communicate them well then you are really operating with a handicap so that's something that's gotta be a big priority for anybody as an educator—you have to be a good communicator, as a student you should be trying to figure out how to develop your communication skills.

REFERENCES

[1] L. Peterson and B. Davie. Computer Networks: A Systems Approach. (Computer networking textbook) *Elsevier*. https://book.systemsapproach.org/

[2] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu, J. Zolla, U. Hölzle, S. Stuart and A. Vahdat. B4: Experience with a Globally-Deployed Software Defined WAN. *ACM SIGCOMM*, 2013.

[3] J. Saltzer, D. Reed and D. Clark. End-To-End Arguments in System Design. *ACM Transactions on Computer Systems*, vol. 2, no. 4, November 1984.

[17] See the talk "How to CTO, or, Putting the 'Talk' in Chief Talking Officer", Bruce Davie. https://youtu.be/svahAz02aJo

[18] http://www.sigcomm.org/awards/sigcomm-awards