Cisco Cloud Unfiltered Podcast Series, Episode 13: Dave Barach and Ed Warnicke

Got questions about VPP? Dave and Ed have answers. In this episode they explain what it is, how it works, why it's such a big deal, and which technologies will leverage it in the near future.

Niki Acosta: It is Cloud Unfiltered episode 13. I am your cohost, Niki Acosta.

Val Benincosa: I'm Val. I'm here with Niki.

Niki Acosta: Yeah, and we've got some awesome guests today to talk to us about VPP, and we've got, what he has described himself as the guy who just sits in a dark room and writes all the algorithms, and then what I would describe as his Google translator [inaudible 00:00:26] and help us understand the value from a business point of view, so why don't we let you guys introduce yourselves? Dave, go first.

Dave Barach: Okay. I'm Dave Barach, and one of a dozen Cisco Fellows. I've spent the last 15 years of my career working on the technology we're going to talk about today called Vector Packet Processing, and I am the guy who counts clock cycles and generally hides under his desk when being a normal social human is required. I guess 100,000 feet, I've been programming since 1968 when I was a 14 year-old kid. A little subtraction will tell you I'm slightly older than dirt. At any rate, with that having been said, why don't we throw it over to Ed Warnicke, my colleague who we won't accuse of being the Google translator. I think I can hold my own, but at any rate, Ed, go for it.

Ed Warnicke: Yeah. My name is Ed Warnicke. I'm a distinguished consulting engineer at Cisco Systems. I'm also the TSC chair at FD.io. I segued into tech because my misspent youth involved moving towards being in string theory and physics, and what I sort of woke up one day and realized, that wasn't really how I wanted to spend my life. Programming had been an interesting hobby, and so I moved that direction with networking, and I've been involved ever since. Largely, I'm entirely evolved in the last half decade to decade of my career interacting in the outside world with open-source communities, and basically figuring out how to fill gaps and needs there so we can move the industry forward.

Val Benincosa: That's awesome, and Ed, you totally strike me as physicist, somebody doing string theory, too.

Niki Acosta: I suddenly feel unqualified to host this podcast. I'm like, "Hm."

Val Benincosa: And Dave, you totally look like exactly what I think of somebody who's doing what you've been doing. You fit the stereotypes. That's a compliment. That's a compliment.

Niki Acosta: Speaking of physics, Dave, you had given us a little bit of a background, [inaudible 00:02:36] work with folks at NASA. How did you get down this path of now working on VPP at Cisco?

Dave Barach: Okay. Well, let's see. I come into Cisco in the mid '90s, working one of the platform teams with an acquisition in what's later become the NEBC. Fast forward to, you can go through a number of jobs, I'd ended up randomly asked to work on WAN protocol, performance, and scaling, and one of the things I have done over the years, for probably 30 years, is to work on performance analysis and tuning of large code bases, and I ended up as a distinguished engineer largely because of getting maybe a factor of 300 WAN protocol scaling back in the day, using some tooling that I still to this day use.

At any rate, that gave me an opportunity to jump into an advanced development group. Some are under [inaudible 00:03:34] and the 7200 team. They had mixed in recently an acquisition from the outside, including my earliest collaborator in this work, a guy named Eliot Dresselhaus, who was a physics guy who worked at NASA Ames and had done a lot of finite element modeling. The two of us rather hit it off immediately. We're not identical in any way, but we have some of the same passion for counting clock cycles and making things go really fast, and Eliot and I sat down and figured out some algorithms to start doing what we call VPP, or Vector Packet Processing, which is to say rather than processing one packet at a time, you process a bunch of packets at once.

And you can think of it almost as you're a dealer at a card table, and you've got three customers in front of you, and you deal the 100 cards, the 100 packets out into piles for them to play with, where the dealer and the players there, rather than ... I hate to anthropomorphize it to a ridiculous extent, but rather than being Joe, Sam, and Jane or whatever, they're Ethernet input, they're IP4 input, they're IP4 lookup, and what you end up finding is that restructuring packet processing in that way lends the computation to getting really good performance on the computers that we actually have on hand, on commodity hardware. So that's kind of a 100,000 foot of how I got into it at Cisco. It was just a lucky break, something that was real interesting, and 15 years later, we have something that I think is way cooler than I would have ever guessed it would be from the beginning, and it's due to not just my work, but a lot of guys.

I mean, Ed's helped shape the way we've put it into open source and stuff, and it's just been a lot of good work around the edges. I mean, it shipped on billion dollar Cisco products in various forms, the ASR 9000, the primary Punt/Inject path is exactly VPP-based, and the techniques work, and it's actually fun stuff for someone like me who really enjoys the act of programming probably more than most things.

Niki Acosta: We'll ask you the same question, Ed. How did you get down to your path? And you mentioned, too, FD.io, FD.io, right?

Ed Warnicke: Correct.

Niki Acosta: Tell us about how you got into tech and your work with open source communities because that's really interesting.

Ed Warnicke: Yeah. So as I mentioned, when I decided that I didn't want to be a string theorist, I had to figure out some other useful thing to do with myself to keep me off the streets, and I wandered into Cisco because they were there, and they were a well-reputed company that I liked, and from there, wandered around, did a whole variety of things inside Cisco, and eventually stumbled into the problem of being deeply frustrated at how difficult in

the early 2000s it was if you were employed by Cisco to go and participate in open source communities, and it was very clear to me that this was a crucial thing for us as a business, and did a lot of education work inside the company. Changing the way we structured our processes, changing our attitudes around engagements with open source communities, worked a lot on how open source is consumed.

I think I was one of the very small number of engineers who were involved with the GPLv3 drafting process on Committee B, and can point to parts of that license that are personally my fault. And then from there, wandered ... At Cisco, got the message more about the strategic importance of open source. I did a lot of work helping us move into engaging with open source communities, starting open source communities. I was very involved in the starting of Open Daylight. I am currently the only remaining original TSC member from Open Daylight's launch who's still on the technical steering committee there, and there's a certain art to building an open source community that is long-term successful.

It's underappreciated because no one remembers the communities that have failed, and so when we were looking to open source FD.io, it was a natural choice to be involved in structuring how we would do that in a way that would be successful that would involve many different players and that would engage with the outside world. I also have a particularly, in this low-level bit banging space, a somewhat unusual passion around consumability, which is to say that it doesn't really matter how good your code base is if it's hard for people to pick it up and use it. At the end of the day, the history of tech and the history of open source is littered with superior technical solutions that have failed, that have lost, and they typically have lost for one of two reasons, and sometimes both.

Either A, because they failed to build good communities. It turns out that communities are really crucial for the success of open source. Or B, they were a massive pain in the ass to use, and so I work very hard on engaging people, keeping the community open and pluralistic, and on making sure that we just do the basic blocking and tackling of being easy for people to use.

Val Benincosa: So I got to ask, there's VPP that we're talking about, and there's FD.io, which is FD.io. What's the relationship between these two things? Are they different projects? I understand VPP is part of FD.io, but please tell me, explain that to me.

Ed Warnicke: Yeah. So one of the critical things that you have to start out when you're building an open source community is how to productively disagree because if you have a situation in an open source community that you do not tolerate differences of technical opinion in a reasonable way, or even differences in focus of interest in a reasonable way, then you eventually fail because you end up excluding really cool ideas, and so the way FD.io is structured is it is a multi-party, multi-project, open source consortium, meaning that there are many different companies involved in producing things there, and there are many different projects under FD.io, of which VPP is just one. We're very blessed that Dave Barach had the foresight when he built VPP to provide it with an incredibly flexible plugin architecture, and the underlying architecture is such that you can accomplish pretty much anything you need via plugins.

And so the net result is, you will get VPP where a lot of the basic blocking and tackling infrastructure work is done, but you can also have projects that come in like the NSHSFC project, which simply produces a plugin to VPP that does work around the network service header protocol, and they can do their work. They can have their own committers, and even though we love them dearly, you don't have to put yourself in a position where a VPP committer has to take enough interest in your patches to review them in order to function.

You also get this with ... We have a very active community around something called information-centric networking, which is looking at a different way of approaching how

you look at the network, and resources, and whatnot, and it's a very interesting area, but it's also very specialized, and by having a separate project, they can produce their plugins for VPP, and have their own sub-community that makes sense. So FD.io in its entirety is really looking at, "How do we accelerate the network data plane?" And in a broader sense, you're starting to see people looking to it to accelerate storage as well, so it is actually scoped more broadly than just networking, and we tend to think about this in three layers.

At the very bottom, you have what we call network I/O. Network I/O can be thought of as, "How do I get a packet from a NIC or a VNIC onto a thread on a core?" And it turns out there's all kinds of black magic in doing that well, and the people who are focused on that tend to be fairly exclusively focused on that. And then you get where VPP lives at a packet processing layer, where okay, now I've got a packet, I'm sitting at a thread on a core, now what do I do to process it? How do I classify? How do I possibly transform the packet? I might want to put it in a encap for VXLAN. I have to do some very pedestrian things like [inaudible 00:12:17]. I need to make decisions about how to forward it. How do I route it? How do I switch it?

All these things fall into the category of packet processing, and VPP does those very well, but if you're the fastest, most feature-rich data plane on earth, and nobody has a way to actually manage and control you, you're still not very interesting, so we have a third layer that we usually refer to as data plane management agents. You can kind of think of it as the control plane, although it's a little bit broader than that, and one of the beautiful things about VPP is it's completely agnostic as to what that control plane looks like, and so there's plenty of space within FD.io for different kinds of projects for data plane management agents.

We have one called Honeycomb that provides NETCONF, RESTCONF, soon it’s gonna provide BGP. We have a project called GoVPP that's providing language bindings for VPP for Go. We have a lot of these different kinds of activities going on, and even though they're all part of the broader FD.io community, they all have their little sub-communities with the people that are focused on that task and gather together for warmth.

Val Benincosa: Wow. So I think I understand it. So FD.io is a pretty big project. VPP is one of the structures of it, and it has all kinds of different plugins where different people can have it, and in essence, you want to create a community so that it's not just, "This is the way Cisco's doing it." So I got a question for Dave. Why is this better than anything else? What's the big deal about it? You talked about some of your inspiration with one of your collaborators about using some magical science, but what's the big deal about this, and why do we even open source it anyway, if it's such a good thing?

Dave Barach: Okay. Well, why we open source it, I think I can give you a reasonable insight. One of the technologies that accomplishes some of the same things that the VPP engine itself does is a thing called OVS, which turns out to be partly in the Linux kernel, partly above the Linux kernel. It turns out to be very, very difficult to develop new features in that environment.

Niki Acosta: Time out. OVS is open vSwitch?

Dave Barach: Yep.

Niki Acosta: All right. Thanks.

Dave Barach: That's the one. Sorry, sorry.

Val Benincosa: And we use that in OpenStack sometimes, right? Some people do.

Dave Barach: Yep. Yeah. Some people do, and the story there is that OVS is ... You don't want to do much in the way of tit for tat, what I think is good or bad about the software architecture, but as a matter of development velocity of the network industry moving forward, it gets largely in the way in some ways, mostly because you can't get patches into the Linux kernel rapidly. It's a good thing. Linux is real stable, and Linus does a marvelous job keeping it that way, but they inherently have to move rather slowly. In our world, if you add a plugin that you got from the back of Farmer Jones's pickup truck, and it smells like it came off the manure truck by mistake, you just don't run that plugin again for awhile until it cleans up its act so to speak, that it's not something that affects core technology that everything else in the world depends on.

There's some people you don't want writing the firmware for your grandmother's pacemaker, and a plugin that breaks a lot, people will just not load until it stops doing it, so we have some real development velocity advantages in VPP. The other thing is recent measurements are that a quad-socket, latest generation, xa664 can push north of one terabit full duplex on not exactly a home desktop computer, but not much bigger than that. Compare and contrast with the CRS1, it actually blows away our CRS8, an eight-line cart configuration pretty easily now, and that's, God, I don't even want to think what one of those would cost or did cost just a small number of years ago.

So it's very, very high performance on commodity hardware, which is also kind of a big deal because you can deploy such things in the data center. You can move network functions that have been traditionally done on closed proprietary boxes costing, measuring millions of dollars, you can do that same set of tasks in a data center for not monkey nuts money, but not a huge amount of money, so it's a big deal in moving the industry forward at a greater velocity kind of sense, and it's also a big deal in the performance space because you can start doing what a core router did eight years ago and a mid-level aggregation router was doing a few years ago, so those are why I think it's a big deal.

Val Benincosa: So were you doing this all on x86? Are you down inside the assembly language of the x86, or where are you doing this stuff at? Where's the optimization at?

Dave Barach: It turns out the code is actually pretty amazingly portable. We've run it on PowerPC 32, PowerPC 64, [inaudible 00:17:51]. We actually had one of the guys in the community crank it up on a Raspberry Pi where it'll do 300 plus megabit. It's pretty scary on a $35 computer. Think about how many bits per cent you're getting there. I won't try and do the arithmetic off the top of my head because I'll mess it up, but we also have run over the years on [inaudible 00:18:14], on MIP64 effectively. The code base is explicitly pretty ... It's 32 bit, 64 bit clean. It's [inaudible 00:18:25] clean, and it's run on a number of processor architectures, but our main focus most days is really on the x86/64 these days.

Val Benincosa: So you're doing that ... So it's just written in C? Sorry. I have to get all technical. I want to know these things. I'm totally curious.

Dave Barach: Yeah. No, that's totally fine. Yeah, the code base is largely C, with the occasionally one or two lines of Inline assembler. How would you put it? There's a hand-rolled set jump long jump for all the architectures, which ends up being effectively assembler code, but there's not much assembler code, and one of the portability things is to not do any more than you absolutely need to in that way. I started programming when assembly code was all you could do, once you got past wires, so I don't mind it, but it's not the right thing to do for portability, so we don't.

Val Benincosa: Right. So cool. Sorry, Niki. I keep jumping in with all the questions.

Niki Acosta: No. Can you hear me?

Val Benincosa: Yeah. Sorry.

Niki Acosta: I was thinking about VPP. I was talking to a colleague earlier because we have actually integrated it with Metacloud for a project that we've done, and I was like, "Hey. Help me wrap my head around this," and he was explaining that basically you could think of packets coming in sequential order and how that creates a bottleneck, and so if you put it on an analogy to a highway, it would be like basically opening up all the lanes at the same time to be able to route the traffic to where it needed to go. Is that relatively accurate? I'm trying to dumb it down here because I'm not a networking expert. I'm a cloud girl.

Dave Barach: Let me try a little bit on the main coding techniques. Fast backwards to about 1985. If you remember the Cisco 7500, the washing machine sized thing that would do about 50,000 packets a second, the way the packet path worked in the 7500 was you take a single packet off of an RX ring, and you'd mess it through a whole mess of C code, and one of three things would happen. You'd punt it to the control plane. You'd rewrite it and throw it out the back door, or you'd throw it on the floor. Those are really the three things that a packet processor needs to do.

Now the observation we did was to say, "Look. Rather than taking one packet at a time, grab as many as you can from the input ring, forming a vector," a vector from size one to size 255 turns out to be the limiting size we use. It's an empirical result that that's about as many packets as you can do at once without running into second order reasons why it doesn't get any faster. What happens in a single packet at a time path is imagine that that packet runs through enough code that all of a sudden the first number of instructions you ran fell out of the instruction cache. Instruction caches are typically measured in kilobytes, not huge amounts, and what happens there is that every packet goes through. It's sort of like running around, and you get to put it ... It's like going through the toll booth once per packet, where what we do is we stuff the packets in a big old truck, and pay the toll for 100 of them at once.

What I mean by that, that's not a bad physical analogy. It's like one person in a car versus 100 people in a bus going through a toll booth. What happens is in any one of the pieces of the directed graph that we use to process packets, we've broken the computation up into a bunch of nodes in a directed graph. Let me not get overly computer sciencey on you, but we've broken the computation up, and what we do is we say, "Okay. I have 100 packets. Now run them through one little piece of the full story, all 100 of them." The first packet around the track warms up the instruction cache, and all the rest of them are free at that level. You don't have to be constantly pulling the same instructions in.

What you've done is you've given the program what we call a phase behavior. We say, "Okay. First I'm going to do this for awhile, then I'm going to move onto something else, warm that up in the instruction cache," and all of a sudden you realize when you've done that and you've created a graph structure that the code often runs 10 times faster than it would if it was done one packet at a time. That's the main trick we use to make it go real fast. It's do the same thing over and over again, and then go do something else over and over again, rather than doing A, then B, then C, then D, then E, and then F. You do A a bunch of times, B a bunch of times. It's a vectorization trick. That's it. And it's amazingly effective.

Niki Acosta: Go ahead, Ed. You were going to say something.

Ed Warnicke: I was going to say, one thing we've lightly touched on but haven't really gotten into detail about that gets to be very interesting, VPP runs entirely in user space, meaning that it doesn't run in the kernel at all, and this gives you a number of huge advantages, particularly as the world moves towards containers.

One of them is that, as Dave mentioned, it takes a very long time to get something into Linux kernel, and because we're not going into Linux kernel and we have a good plugin architecture, it means you can maintain a much higher development velocity, but it also means that a lot of your stories around high availability, upgrades, and everything else have also gotten to be much better, because rather than having to go, if you want to upgrade a feature that's in the kernel where you have to bring the entire box down to upgrade the kernel, and where you're looking at a reboot cycle and the time involved in a reboot cycle and the loss of state of all the processes running, if you want to go in and upgrade a VPP, all you have to do is kill the existing process and start a new one.

So you go from minutes, sometimes 15 minute reboot cycles, to milliseconds in order to do that upgrade and that path, and you also don't lose the state for all the existing applications that may be running on the box. So that's a very big deal, but the other place that gets to be huge is when you start looking at the world of containers and microservices, because now that we've moved the network stack into user space, you can actually treat networking and pieces of networking as microservices, and this makes it incredibly easier to do the kinds of things you want to do with networking and container space, and to break the world up into network microservices.

Val Benincosa: Wow.

Niki Acosta: You just blew my mind on that.

Val Benincosa: So we're Cisco right? And we've made a lot of money as Cisco as a company on selling boxes that have hardware that's optimized for this, and so now that we have this software that we're giving away for free, what does that do to the company?

Ed Warnicke: That gets to the business case discussion around this, and it's pretty straightforward because while you can do some amazing things in terms of software packet processing on a server, you still tend to lose very strongly on certain things like [inaudible 00:25:49] for example, that go into the operational costs of this, but at the same time, the market is very clearly demanding being able to do certain kinds of very sophisticated features networking-wise on servers.

Val Benincosa: On the servers, yeah.

Ed Warnicke: Right. So you get into a place where if I have things that I need to do on the server anyway, then you need this kind of performance, you need density, you need scale on the server. So today, when you look at containers on a server, you're looking at a density of maybe 100 containers on a server, but as you look forward, you have people who want to talk about doing 1,000 containers on a server, 10,000 containers on a server, so the world becomes complicated enough that you really need this kind of sophistication, performance, and density on that server.

Now does it make sense operationally if for no other reason than pure power constraints to go through a server into a place you might have otherwise used a core router? Probably not. But one of the things that we found as a business at Cisco is there are lots of business plays where in order to enable the value that we bring in other parts of the network, or the value that we bring in higher order products, we need to be able to have a data plane that can do these things that can run on the server and many customers are demanding that that kind of common infrastructure actually be open and open source, and so it makes huge amounts of business sense for us to open source the VPP data plane for that reason.

The other thing that you run into that's a fascinating fact about modern software is in the modern era, 80% of the value of almost any software that you have running on the

server is the ecosystem of things that it connects to, and so being open source plugs you into that ecosystem in a very strong way, which enhances the value of the overall system.

Niki Acosta: So it's no different than cloud? I mean, open source like OpenStack. It has the same sort of value, the black box, vendor lock-in stuff goes away if what you're providing is open source and available for anyone to use.

Ed Warnicke: Yeah. It's more data plane than what you see in something like OpenStack, but yes, and it's much more flexible in terms of the kind of network models that it supports. Typically, in OpenStack, you're looking at this very linearly built L2 networks, L3 routers, etc., and with VPP, you can do pure L3 routing. You can do complicated features like IPv6 segment routing that also allow you to do really interesting things. You can do all the different kinds of encaps and decaps that make sense, and you can build very small, specialized network microservices, which you can deploy in any container just like you deploy any other microservice, again, because of the user spaceness of it all.

Val Benincosa: So Ed, I got a question. So let's say that I'm a big time Kubernetes user, love it, just got it running, and I'm now looking at all my different types of networking stacks. There's Weave, Calico, Flannel, and Cisco's Contiv right? How can I use VPP with this today, or what's that look like, or the future of that?

Ed Warnicke: Yeah. So one thing I do need to point out is that VPP is providing the data plane, and I mentioned earlier that it's sort of agnostic as to the data plane management agent that's running it, so basically anyone can take their control plane piece, any of those guys you mentioned, Weave, Flannel, Calico, any of those guys can use the VPP data plane for enhancing the service that they offer. Contiv is expecting, I believe in its 1.2 release in September, to be providing VPP support so that you can actually use it there in order to support your container networking, and I expect as we move forward, we'll see a lot more interesting things coming out in that Kubernetes space by using VPP because one of the things that is non-obvious is that a lot of the limitations that you run into in the container space around networking are actually limitations of the underlying data plane that's currently available.

If you're going to go use something that was designed as a host stack for a host computer, it's going to have real limitations in terms of what you can do with it as a data plane to support your containers, and so the more you get VPP into the container space, the more interesting networking things I think you will see being trivially possible for Kubernetes that at this point, are hard to impossible.

Val Benincosa: I don't even know ... I'm not smart enough to really ask a question of what would be possible that isn't possible because I don't understand-

Ed Warnicke: I'll give you a really straightforward example. One of the things that VPP, and in fact, Dave is spending a lot of his time on this right now, that VPP is bringing online is a user space host stack, right? So right now, if I'm a process, I grab an interface or some collection of interfaces and I listen for a socket, or if I'm a client going out, I open a socket to talk to some remote thing. Now Dave can say much more detailed things than I can, but what we've built there is a TCP host stack that was designed to operate at cloud native scales, so out of the box, it gives you single core half-million incoming TCP connections per second performance. It was designed to be able to move into the kind of world where you want to go from 100 containers per box to 1,000 containers per box to 10,000 containers per box, which is a very different way of approaching the problem than if you've designed your host stack in order to support a server that may have a couple of interfaces on it. It's just a very different design.

And so that ends up being a very interesting approach to the problem, and when you realize that you are going to have these kind of container densities, and that a lot of the

traffic between the containers is going to be east west traffic on the same box instead of north south traffic that leaves the box, suddenly certain optimizations become very obvious, like why would I go through a TCP stack to talk to a container that is sitting right next to me?

And so one of the things that Dave has done, and again, he can say a lot more about this is if we realize that the route that we have is route this traffic to another container, we don't do TCP at all. You still see a socket as the application in the container, but instead, you get much higher performance at much lower resource cost. First in first out, FIFO passing of the traffic.

Val Benincosa: Because you're not going all the way down the stack.

Ed Warnicke: Exactly.

Dave Barach: And that path, to be honest about it, is we benchmark it north of 50 gigabit full duplex now. It's probably not every pair of programs want to shovel that much data between container and container.

Niki Acosta: Not yet. Not yet.

Dave Barach: But the idea, as you can imagine, terminating quite a collection of incoming flows, running them through a service chain where going to the service chain elements, and then to the first guy in the service chain, A, B, C, and then back out the back door, that turns out to be really pretty incredibly inexpensive. People have tried to do that with SRIOV technology where you're actually using hardware DMA engines, the trouble being, if you've got a 40 gig NIC, it's going to have 40 gig of DMA bandwidth, and each hop from container element to container element to service chain element, rather, each of those hops is going to chew up DMA bandwidth, whereas in this case, you're pretty much saying, "Well, the real limitation is how much memory bandwidth and how much ... " You know, a multi-socket world, how much can you get through the QPI bus if you're using an Intel thing?

So it's a way of really lighting a fire under local chained cases, and VPP at a certain level, it's almost being an introduction interface. It says, "Okay, know that you think you really want TCP to a guy that I happen to also be talking to, so why don't I just introduce you two and get out of the way?"

Val Benincosa: So that's why it's better than SROIV then, is because now we're not going all the way down to the NIC, and we're actually staying up in user space to do that transfer. So it's just based on the memory bandwidth of the processor.

Dave Barach: Right, and rather than pushing data in and out of a PCI bus N times, if you've got N service chain elements, you get the data once off the wire, through the PCI bus, you shove it up into user space, and then when somebody wants to transmit back out the trunk, effectively, you do one more transaction with the PCI bus, so if you have a 4, 6, 8 deep service chain, you're going to really hugely profit from that.

Val Benincosa: Oh, yeah. So a question on it. Does this mean, could I say that with VPP, are we betting on a future of boxes that have thousands of containers as opposed to maybe the boxes today that have maybe 100 containers per box? It is it meant for something like that?

Dave Barach: I'd say that you're talking about really getting, minimizing the amount of the box that's devoted to doing networking, as opposed to revenue generating whatever those service chain elements are doing, whatever the containerized workloads are doing. We're trying to very efficiently handle the problems that have heretofore, in a lot of cases, been done

by the Linux kernel. Certainly functional, but not tuned for performance, and not friendly when you start chaining together microservices phrased in containers.

Niki Acosta: So I noticed that ... Tangentially, I work on the private cloud side, and through that work, we do a lot of work with service providers, and it seems like service providers are very eager to have VPP. Like they're very, very interested in this type of technology. I don't know why. Is it because they can suddenly facilitate a ton more packets with fewer resources? Is that what it boils down to? Is it allowing them some kind of edge over other service providers?

Ed Warnicke: Yeah. So I mean, this is one of those things where embrace the healing power of "and", right? Because service providers are operating at a scale where anything you can do to actually reduce to them the cost of providing services is going to be a huge deal for them, and in some of those costs, it's just the raw silicon they're consuming, the number of cores, etc., and one of the things that you tend to find is that the flip side of performance is efficiency. So when we talk about being able to push a terabit of traffic through a server, by the way, using a minority of the cores on that server to do that, then you're also talking about the fact that if you're pushing less than a terabit of traffic through that server for whatever reason, you can do so with far fewer resources, which of course translates to dollars and cents if you're a service provider.

The development velocity and the feature-richness also end up being a very big deal because if you're a service provider, you want to do more sophisticated things than somebody who's developing a host stack for a mere server would want to do. You want to be able to do things like IPv6 segment routing, which we already support. You've got a bunch of features related to 5G that you want to be able to support, which are coming fast online inside VPP, and you also very much want to get to that containerized world because containers end up being much more resource-efficient than VMs, and so if you have a data path that is living in a kernel space, you're sort of forced to go into a virtual machine, which both gives you lower performance, it tends to be harder to manage, and it also consumes a ton more resources.

If you can go purely to user space with your data plane, then what you can actually do is take advantage of the lighter footprint of containers, the easier management of containers with things like Kubernetes in order to get a fairly big advantage as you split things up for your NFV workloads. Now, that's all good news, but you also wind up with other kinds of things that play into it as well.

So when you want to do hardware acceleration, and many service providers do, and there are many vendors in the space of providing hardware-accelerated NICs of various kinds, you need a highly modular plugin-oriented architecture like VPP to be able to get from A to B really quickly. If you're a service provider, you don't want to have to go and do bespoke development based upon which NICs are in the box, and have to go build from source code and increase radically your management expense by having to track which binaries you're running on which box.

With VPP, a vendor of hardware-accelerating NIC can provide you with a plugin that you drop into your plugins directory, and if your accelerating hardware is there, it takes advantage of it, and the world goes fast.

Niki Acosta: Because it's abstracted right? It's completely abstracted.

Ed Warnicke: Well, it's less about abstraction and it's more about modularity. The very thing that Dave mentioned in terms of breaking the problem down into small chunks so that you can go very, very fast also means that you can break the problem down into small chunks in terms of where the demarc is between your accelerating hardware and what it does, and what VPP does. So you'll get scenarios where, for certain use cases, the accelerating hardware can do all the work, but when it hits more sophisticated features

that maybe it doesn't support, it can pass them up to software, and this, of course, means that you can do this in a much cleaner fashion, but the modular nature, being able to say, "For every hardware vendor, I'll have a plugin that enables their hardware acceleration," and if their hardware happens to be on the box I'm running my VNF on, then it goes very, very fast.

But if I go beyond the boxes that I have with the magic NICs, it still runs. It still works. If I have multiple different vendors of hardware-accelerating NICs, they all still work, and so it gives you an ideal environment for building this out because it gives you an ideal environment for innovation, and at the end of the day, that's really what it comes down to is how we solve problems involves innovation and shortening the latency between ideas and execution, deployable execution, and VPP has the ideal infrastructure for that.

Niki Acosta: Got it. Badass. That was exciting.

Val Benincosa: Got it. So I got a question. So we've been talking about how awesome this is. What are the haters saying? Who are the haters? Who is the competition? What else is going on at this stage?

Niki Acosta: Does this replace OVS? Can you run it adjacent to OVS within the same footprint?

Val Benincosa: Yeah. What are the haters saying?

Ed Warnicke: You'd want to replace OVS with it, and when you look at things like the host stack, you're, in many ways, replacing the entire kernel networking stack with it, and so in terms of what the haters say, I would characterize it less as haters, and more of differences in opinion about how the world works. So if you are someone who strongly believes that the world should be nothing but match flow action tables, then you're going to have a natural affinity for something like OVS. Now-

Val Benincosa: Why would you even say that? I mean, everything you guys have said when you describe this, to me, it's like, "Well, yeah. It's obvious. You put it in a vector. Of course that's the best way. Why wouldn't it be?"

Ed Warnicke: Not everyone agrees with everyone else about technology direction and what they consider to be the best approaches to technology. That's a natural fact in the world.

Niki Acosta: Clutter. It's hard to change. People don't like to change. They have to ... People are naturally resistant to change.

Ed Warnicke: So I grew up in physics, which is the hardest of the hard sciences, and we have a saying there, "Old physicists don't change their minds. They die." And when you have people who have very strong personal investments in having learned something and getting close to it, they develop a very natural human affection for it, and there's a learning curve for learning something else. So if you just spent the last half decade, of which I spent too much of my life doing this, by the way, learning how to try and take a semantically meaningful network thing and flatten it into a reasonable match flow action table world, which by the way, I personally found insanely painful, then you're going to have a natural affinity for that just by virtue of having done it a lot, right?

And then you also get back to the ecosystem stuff that I was talking about, where 80% of your value derives from the ecosystem you plug into. People like Open vSwitch have been in the ecosystem for longer. There are more things that have attached to them, and so there are still places where the ecosystem is in the process of attaching to VPP. So you may find in places ... In many places, you will find there is a natural ecosystem

attachment for VPP that's already there, but you may find corners where there aren't, and that's just something that comes with time and adoption.

Niki Acosta: Very cool. There's a lot of analogies here. In the realm of open source, especially, when you've got communities of bright people, I mean, in OpenStack there's this big tent idea where they were just going to say, "Okay. Everyone's welcome," and now they're dialing that back and saying, "We need to be very specific about what we say is core, and everything else," because people want consistency. In order for them to be able to feel confident and build on something, they want to know that there's going to be support for it, and I really think that's probably the value that Cisco provides. I've been thinking about how Cisco monetizes this, and my guess is that A, we have some of the plugins, and B, we probably have the right expertise to be able to support customers who want to go this path.

Ed Warnicke: Absolutely.

Niki Acosta: Am I off?

Ed Warnicke: No, you're completely on, and we also build various products on top of it, and it also enables a bunch of technologies that we have in the network that allow people to actually do things better than they could do if they were stuck with older legacy networking technologies on the server. So I mean, these are all paths to a really sane business case, and why it was pretty much a slam dunk for Cisco to open source this from a business point of view, but one of the things that's always been lovely in my experience, and I've been working in open source for the last 10 to 15 years, is that if you've been intelligent about your technology, then it is very often the case that doing good in the world is profitable, that you can do well by doing good, and this is a classic example of that.

Val Benincosa: That's awesome.

Niki Acosta: I was reading up on the FD.io site, and there was some good information about having VPP being used as a basis for a load balancer, a firewall, intrusion detection, host stacks, and creating combinations of applications. For example, you could add load balancing to a vSwitch.

Ed Warnicke: Yep.

Niki Acosta: Does that exist already?

Ed Warnicke: Some of them do. So for example, we do have a [inaudible 00:46:07] style load balancer in VPP. We do not yet have DPI. I've talked with various parties who may be interested in that, but that hasn't come to fruition yet. In terms of firewalls, we have a lot of things that look like ECLs and that do that at various levels of classification in order to allow you to police what's going in and out of the box. Do you want to comment a little bit more on some of these things, Dave?

Dave Barach: Well, I can certainly point out that we have, at times, done ... Some guys at Cisco Israel did an NBAR prototype, which I think we're just about getting ready to dust off again because it's getting more important for-

Val Benincosa: What's an NBAR?

Dave Barach: NBAR. What is it? Network-based ... Something or other recognition. It's deep packet inspection. It's a DPI technology, and you can ask yourself, "Well, how do you do DPI on HDPS?" And that's a fair question on the other hand. Yeah, those sorts of things are there. We do quite the set of combinations. Going back to my home gateway again for a

minute, we're doing what amounts to integrated routing and bridging and a bunch of carrier-grade NAT effectively. Those are two things that we've sold as products, that if we choose, we could ... we've had a product for many years on the CRS1 that was actually VPP-based called Carrier-Grade NAT, and VPP at this point in the game is good enough in that space that you could begin to do a shrink-wrap solution to sell.

The kinds of things I expect to see in Cisco product development moving forward are take the VPP engine as of a certain release, and here's a bunch of special sauce that does 5G mobility work, that does virtualized CMTS, that plays again ... The cable example is in the works, frankly. I don't want to say too much about it because we don't want to be preannouncing products and so on, but that scenario is definitely in play of taking things that have been traditionally done on proprietary hardware and splitting them into the little bit that does the electronics part. It's just hardware. You have to do it that way, and then tunnel all of that stuff back over 100-mile back haul network to a data center that can actually do low-level protocol bashing and then the routing functions that's needed in that space.

But those are the sorts of products that we will be shipping, and ways that Cisco can really monetize. Yeah, the little detail is of course, if you're selling a Cisco shrink wrap solution, we do sell computers. We test all the time on UCS, and some of our smart NIC technology, you know, the [crosstalk 00:49:23] is an area where we can do things to the point about hardware acceleration. We can work a little bit with guys to get some microcode in those sorts of devices and make it so that, yeah, this stuff will run perfectly fine on Micron or Dell or whatever server you want to throw at it, but if you buy one of ours, it'll be factors faster, but that's another path to monetization we see.

Niki Acosta: Cool. And you're running this at home, yes?

Dave Barach: Oh, yeah. We're talking and my ugly mug is showing up through a VPP home gateway, so assuming that I don't look any worse than normal, well, it's obviously working today. I will say that the home gateway's been up for about two months, and it's probably pushed ... My family loves to watch videos, so it's probably pushed almost two terabit over that period of time, and it's running on an Atom-based six by one GigE piece of commodity hardware that I just bought off the internet.

Niki Acosta: Crazy. That's so awesome. So I'm going to ask another question because it's something that I've been wracking my brain about, but it seems to me, then, at least for, I'll say a majority of people around the world, we are now able to take advantage of technologies that are really intuitive, really easy to understand, really easy to use. A lot of the complexity has been abstracted, and it's funny. There's an inside joke at Cisco about the fact that we're really good at making things complex, and whether that's because of networking's hard, or stuff is so descriptive that you have to shorten it to an acronym, I don't really know what that is, but I look at all of the products and services from a consumer level, from a business level, and it seems like these things are starting to become very intuitive.

They don't require a ton of expertise. I can go up and I can now get a domain and provision a website in five minutes. I don't have to know HTML. I don't have to know any of the underlying database stuff that happens. I just go and I click a button and stuff works. Do you foresee having that kind of intuitive simplicity when it comes to the network and some of these technologies? Are they-

Ed Warnicke: So for the audience, for the particular audience that you're dealing with. A good example of this is around containers. There's a ton of stuff about networking that nobody in devops should ever have to understand.

Niki Acosta: Yes. That's what I'm saying.

Ed Warnicke: And if you try and make them understand it, you're doing them a disservice, so if you would just want to deploy an app, you don't want to know about L2 network segments, and programming routes, and all the rest of this. You want to be able to say, "Hey. Here's my app. I want something that looks like a Kubernetes service so I can go reference it, and maybe I want something that looks like a reverse proxy behavior, like Kubernetes Ingress, and I want it to work maximally. That's really what I want. I don't want to worry about anything else." And so this floats into issues around consumability, which is again, one of the things I'm very passionate about, and where that consumability is, what simple means to you, depends very much on who you are as an audience.

So for the container guys, it's going to be being able to talk about things that are semantically meaningful to them, like services, and ingresses, and network policies. If you're somebody who is in a service provider space, and you're trying to stand the thing up, then consumability may mean something slightly different. It may mean having good packages you can apt in Yum where that stuff just runs without you having to think about it.

Now you then may think incredibly complicated thoughts about how you want to configure it, because there is no end to the creativity in complicated thoughts in the service provider space, thoughts that you would never want to expose to a devops guy. So it really comes down to ... You're talking about consumability, you're talking about simplicity. What you're really talking about is who is the audience and where is their locus of interest? And I talk to people across all these different fronts all the way down to people where their locus of interest is really the code itself, and where for them, what's provided simplicity is having this graph node architecture because they're writing code that plugs into it.

And then as you bump up the line all the way to something like Kubernetes and their descriptions of the world, at that point, simplicity means something different, and you have to hit the right element of simplicity at each of those layers, because if you try and feed somebody something that is too simple, you've made it impossible for them to do their job. Like the service providers would crucify me if I tried to make them live exclusively within the Kubernetes layer, just like the devops guys would chase me down with pitchforks if I tried to make them understand packet processing graphs, and they would all be right to do so.

Niki Acosta: So in that sense, it's clear to me that this is a very valuable technology, but it should also be something that just works. Like you may not see this technology directly, but you will definitely experience the benefits in some way, shape, or form somewhere else.

Ed Warnicke: Yeah. Think of it from the point of view of somebody who's standing up a relatively large Kubernetes cluster, right? And even step back from the devops and the technology guys. You're the guy who writes the frigging check for the boxes, so you're writing the check for the boxes, and for you, what simplicity means is when your guys deploy this technology, the size of the check that you have to write declines by X percent, right?

Niki Acosta: Awesome. It's true.

Ed Warnicke: And that's all the more you care about, right? And so you need to also be able to plug into that layer because we all like checks. They're very nice. I'm very happy with the one that I get every two weeks, and the guys who write them are all doing very, very hard calculations about value per dollar.

Niki Acosta: It's a great way. I hear about VPP and all these networking technologies, and I'm biased definitely because I come from the cloud world where we just expect the network to be there and we expect it to work.

Ed Warnicke: No, absolutely.

Niki Acosta: And it's a different perspective.

Ed Warnicke: Yeah. Well, the thing is in the cloud world, you do expect it to be there, you do expect it to work, but you still want things from it, and you want the right level of abstraction of expressing those things, and so yeah, you expect it to be there and you expect it to work, but if you start having people telling you that you have a limit to the number of containers or VMs you can deploy on a host because of it, or if you have a workload that requires a certain amount of bandwidth, and you simply can't get it out of the system that you're running on, or if you discover that you have an application that's latency-sensitive, and it's breaking, and you realize that you're spending half of your latency budget running in circles in whatever is providing your networking, that matters to you, too, and it gets back to your point about just working [inaudible 00:56:45].

That definition shifts over time, and the network that just worked for you yesterday is the bottleneck in your system today, and so VPP is providing that network not just of today but of tomorrow so that you never actually notice those bottlenecks because they aren't there.

Niki Acosta: Well, you all have done a phenomenal job in explaining this. I was really nervous. I was like, "Val, you're going to have to help me here because I'm reading this stuff and it's like Greek to me," but you all did a really good job.

Val Benincosa: [crosstalk 00:57:16]. Yeah, no. I think I've come away with a great appreciation of VPP and FD.io. It's pretty awesome.

Niki Acosta: So it is an open source community. You can find it on FD.io. Your wiki's actually really good. Did you write that, one of you? The wiki stuff? The "What is VPP?"

Ed Warnicke: Like any good wiki, it was done by a cast of hundreds to thousands.

Niki Acosta: Wow.

Dave Barach: It's grown up over 18 months or so that we've had it in the open. It started off with I guess a little bit of wiki that I had had on an internal server for years and years before we decided to open source the technology, but yeah, there are a ton of people. I mean, we probably ought to do a laundry list of folks we haven't mentioned here. [inaudible 00:58:15], Keith Burns, sort of my best buds down in packet bashing land. Neil [Rands 00:58:23] did a tremendous job on building a world-class FIB, the structure that actually says, "Okay. I have an IP4 packet. It wants to go somewhere. Get it to happen." There have been a ton of contributors.

My original collaborator, Dr. Eliot Dresselhaus, all these guys have done just yeoman's duty over the years. I'm kind of the [inaudible 00:58:51] run around behind the circus parade cleaning up after the elephants, pushing the broom, and kept the thing moving forward for long enough that now you have the sensation that you've gotten to the top of hill. The huge boulder that you've rolled up and had it roll back on you any number of times has finally gone over the top of the hill, and now you're hoping it doesn't hit your house.

Ed Warnicke: And Dave, don't forget [inaudible 00:59:15] and all the amazing work that happens [crosstalk 00:59:17].

Dave Barach: Yeah, yeah.

Ed Warnicke:	One of the things that we do that is quite a bit different than any other data plane community you're going to find is we do CICD for performance on real hardware, and so literally, every release we do, you get an automatically generated report that says, "This is all the performance testing we run. This is the performance delta that you got from the last release." It's really quite astounding.
Niki Acosta:	Wow. And it's all automated. Yes?
Dave Barach:	One last shout out I absolutely want to do is to David Ward, Cisco's CTO, chief [inaudible 00:59:53], head of the architecture. David has been a huge supporter of this technology for a decade, and without his alternate coddling and kicking and coming up with budget when nobody else could to keep the lights on and the wheels spinning, we wouldn't be where we are today.
Niki Acosta:	We need to get him on as a guest. I would love to talk to Dave Ward. He has his hands in a lot of things, and he keeps coming up. I'm like, "We should just talk to him." I see there's mailing list. I see there are IRC channels that people can join at FD.io. Where can people find each of you? I know you, Dave, are @vppguy on Twitter, and Ed, are you on Twitter?
Ed Warnicke:	Yeah. I'm the very cryptically named @edwarnicke.
Niki Acosta:	Cool. And that's W-A-R-N-I-C-K-E.
Ed Warnicke:	Correct.
Niki Acosta:	Awesome. Well, thank you so much guys for explaining this. This has been hugely helpful for me, and I hope it has been for our audience as well. We've got Vikram Hokasote next week on the podcast. We have a ton of guests lined up. Please subscribe, leave comments, let us know what you think. You can find us on iTunes. You can find us on SoundCloud. We love hearing from you. Val is @vallard, V-A-L-L-A-R-D, and I'm @nikiacosta, one K, four letter word, so hit us up, let us know what you think, let us know who we should talk to. We love hearing from our listeners, and we thank you so much for your support.
Val Benincosa:	Yeah. Thanks, Dave and Ed. Awesome.
Niki Acosta:	Everybody say "bye".
Dave Barach:	Bye.
Ed Warnicke:	Bye.

For More Information

Find more Cisco Cloud Unfiltered podcasts.

Learn more about Cisco Cloud solutions.

Americas Headquarters
Cisco Systems, Inc.
San Jose, CA

Asia Pacific Headquarters
Cisco Systems (USA) Pte. Ltd.
Singapore

Europe Headquarters
Cisco Systems International BV Amsterdam,
The Netherlands

Cisco has more than 200 offices worldwide. Addresses, phone numbers, and fax numbers are listed on the Cisco Website at **www.cisco.com/go/offices.**

Printed in USA 07/17