Ben Pfaff

Derick: [00:01:00] How's how's 2021 treating you so far.

Brandon: Yeah, I think I heard this in a tweet that 2021 already told 2020 to hold its beer.

Derick: Right. 2020 has, uh, turned 21 and can legally drink now and is already winning at beer pong.

Brandon: What do we have today? Well, maybe you've heard of something called Open vSwitch.It's the virtual network switch for Linux. Uh, we got to sit down with Ben Pfaff, one of the original and core contributors to this open-source project and talk to him about how Open vSwitch came about.

Now you might think a pre-2000 Linux developer who lives and breathes the C language who got a computer science PhD at Stanford in the hardcore area of file systems who was advised by the founder of VMware would be a bit intimidating. Uh, but Ben is the polar opposite of that. And you can tell quickly that he's someone who's open to any question, whatever level of detail.

And that makes having a good conversation with him - pretty easy.

Derick: Yeah, it'd be, even before I met Ben, uh, I had found his guide in how to write in C on his website and, and had been using it. And during the episode, when I brought it up, he was like, oh yeah, my website, I should probably go back and revise that.

Brandon: Yeah, here's my quick Ben story. So I think this was for the OpenFlow project back in 2007 or so we were trying to decide which source code system to use. And everyone had an opinion, but we heard Ben chose git for OVS and everyone I knew trusted them technically. So there was no question we all installed git for the first time and we used it there.

Banter with Ben

Brandon: [00:02:56] Ben, welcome to the podcast.

Ben Pfaff: I'm happy to be here. You guys are - you guys are fun. And I at least know you Brandon from, previous experiences. I'm sure I've met, uh, the other two as well, Derek and Aditya.

Brandon: I feel like in other times we'd be doing this over a beer,

Ben Pfaff: Yeah.

Brandon: So why don't we get to know you a little bit better Ben, who are you? What do you do?

Ben Pfaff: Well, I work at VMware, I'm in, uh, the VMware research group, and, uh, up until now, uh, since roughly 2007, I've been known for a piece of software called, uh, Open vSwitch.

Brandon: Great. Well, we'll talk about that plenty more later today. Uh let's let's do some one-liners. Let's get to know you a little better. What was the last thing you changed your mind on?

Ben Pfaff: Ooh, last thing I changed my mind on, uh, gosh, I guess the, uh, the first thing that, that comes to mind is I've been letting my hair grow out a lot this year.

Brandon has it always been that long while I've known you or, is that something new for you too? I don't remember.

Brandon: There was a period deep into grad school where I didn't care, and I let it grow. And I did the locks of love thing. And there's a picture of me, that's in a field of goats and it's affectionately named the goat Jesus picture because when your hair gets long enough, you kind of start to resemble Jesus.

And, 10 years later, and now I've got. I'm rocking a ponytail. I can't quite do the locks of love 10 inches for a charity thing, but it's getting there.

Brandon: You changed your mind on his haircuts are required, regular thing? They're optional, or what was the change? Yeah.

Ben Pfaff: Well, the change initially was that I was scared to go get my haircut, starting in March, with the whole COVID thing. And then after a while I decided, Hey, this is actually kind of fun, and I'm going to let it go for a while and see what I think of it ultimately.

Brandon: I don't know what Derek thinks of this, but I think there's like a certain geek cred when it gets beyond a certain length.

Derick: I used to have very long hair as well. Um, it doesn't happen anymore. I'm thinning out.

Ben Pfaff: Well, now it's all in the front.

Derick: Yeah. See that's where I'm growing it out now. So, I'm, I'm jealous. I wish I could grow my hair. I wish I could have hair like yours.

Brandon: For the listeners, Derek is rocking, what, five inches of beard?

Derick: Yeah. Yeah, I got to let it go this time. I got to get it back up. It was like seven or eight inches.

Brandon: ZZ Top got nothin' on you.

The Rust Language

Ben Pfaff: [00:05:09] So, I don't know if our audience is technical enough to appreciate this, the more technical thing I've, I've started changed my mind on recently as I'm getting, I'm getting to be okay with longer lines of code, uh, probably moving out from about 80 columns before to about a hundred now.

And that's partly because I just this year, got this enormous monitor that has room for lots of columns of text across the screen. So, now it feels like I'm just wasting the space. If it all goes vertical instead of horizontal.

Derick: Have you ever seen Scala code where it's like dot, dot, dot, dot, dot, and it's, it can be lines, right? It was one line, but it's like a huge series of nested classes and functions.

Ben Pfaff: I don't think I'm to that point yet. I've been writing more Rust than, uh, I don't think I've ever worked on Scala.

Derick: Isn't it like number one now on that language survey?

Ben Pfaff: Well, for me, uh, Rust is the language that I've been looking for for a long time. I have -been a serious C programmer since the 1990s and C is great because you have total control. But on the other hand, you incur so much risk, any sort of mistake and things go wrong, your, your whole machine explodes, it's only memory protection that saves you, and with Rust, you have to do a lot of work upfront to really figure out what you want.

But on the other hand, when it runs and your bugs are not ones that blow up the machine. Your bugs are ones in, what people refer to as business logic, they're bugs in your logic and not in your memory management.

Brandon: Does that make you question your identity a little bit? Cause I think of you as a hardcore C guy. I think of you as someone who has literally written "How to write good C".

Derick: I want to point before he answers. I just want to say we'll put a link into Ben's website. You have written I've and I've promoted some of the things you've written on Twitter, some really great guides in how to do things in C.

Ben Pfaff: Oh, I know that I had things like a C coding style guide, that was one that I wrote in the 1990s. And I'd have to go back to it and see how much of that is something I still support.Like on the other hand, I guess things like the

Brandon: Do you say "hi, I'm Ben. I'm a Rust programmer".

Derick: Or is it a Rustian? What do they call themselves?

Ben Pfaff: They call themselves Rustaceans. Um, so, the thing is, it's hard for me to say, hi, I'm Ben, I'm a Rust programmer, because there's a lot of Rust that I'm still kind of a novice on. I don't think I'll be able to say that until I've gotten my, my rust up to, you know, somewhere in the ballpark of where my C is.

That's going to take a while.

Brandon: I wonder if it's kind of like C people going into Java, where they would port the idioms, even if the idioms weren't necessary.

Ben Pfaff: Oh, Oh yeah. I'm, I'm, I'm definitely guilty of that. So, Open vSwitch, which I'm sure we're going to talk about a lot. Um, has a ton of libraries in C and then at one point we wanted to, do some work in Python. So, I spent probably what you call a long weekend, porting a very large portion of the C libraries in OVS into Python, which basically consisted of me typing as fast as possible, writing out all of the C stuff in Python.

And I've been informed by people who are really Python programmers. That there's a lot of stuff that is not very, uh, idiomatic for Python there.

Brandon: I feel like we could do an entire episode on languages and it's one of those bike-shedding things where everyone's got an opinion, but I think for this episode, we're not going to go too much into languages. We're going to go into users, systems and people a little bit more than the deep-tech dive that we could do.

Ben Pfaff: Yeah, let's dive into it. Uh, where do you want to start?

What is OVS?

Brandon: [00:08:50] Yeah. So maybe give us a sense of what is OVS beyond the one-sentence description? What is it? How did you find your way into it? Maybe some backstory there.

Ben Pfaff: Yeah, so, we ended up writing OVS at Nicira, which was a, a startup that I joined as one of the first employees in 2007. And we didn't really set out to build a software switch. Uh, what we set out to do was to build a business that would be, based on flow-based networking based on being able to get packets to where our customers needed them.

Um, when we started out, we didn't have a very good idea of what the application was going to be, that we were going to try to sell to people.We knew that we wanted, to be able to, do something impressive, do something that would, uh, be at high scale. Butwe didn't have a good idea of how to do that, or even if it would be software or hardware based,and so, uh,here I am, uh, plopped down in an office and told, okay, we're going to do something it's going to be based on networking. We don't, we don't exactly know, um, what we're going to need to do with the network.

So, build something that'll do that. And, when somebody gives you a problem like that,what you end up saying to yourself - and of course I'm saying this as if it was me, but this was a collaboration between a large number of smart people, so forgive me, uh, all of my collaborators, when I make it sound a little bit like it was me, it was, it was all of us -but when that happens, what you do is you build something programmable so that, when you figure out what your real problem is, you program it to do that. And so, what, uh, OpenFlow and Open vSwitch ended up being from our perspective was a programmable packet processor.

It takes in packets, uh, on one end and it spits them out on the other, and, uh, in the middle, you have some other pieces of software out there, a controller, uh, telling it what to do with the, uh,the packets in the middle. So that's sort of the OVS story, in a nutshell, from my perspective.

OVS - how a non-network-operator would design a switch?

Brandon: [00:10:50] One of the things that stood out for me was - OVS is almost like how a non-network-engineer might design a network switch. Can you talk us through a little bit of the differences between maybe a virtual switch and a physical switch?

Ben Pfaff: So. The difference between a physical and a virtual switch from a programmer's perspective, it's just a difference in the device drivers, you have, uh, a device driver that talks to a virtual interface or one that talks to a physical interface.Um, from a user perspective could design both of them to have exactly the same interface.

There's, there's no reason that you couldn't use your IOS CLI to configure either one of them. but, uh, we wanted things to be, more, intimately, controllable, by a controller so that, um,maybe things at the individual packet level, could be programmable, maybe, uh.If you go to sort of a traditional network switch, then, uh, what you're telling it is very much sort of generalities.

Um, this port is on this VLAN, um, maybe, uh, apply this ACL that sort of thing. but. the OVS interface was something you can implement that sort of thing with,but you can also tell it, things about, uh, arbitrary combinations of network fields, of headers and so on.

And you can instantly reprogram it,at any time, through the API. we didn't think of it as an API. We thought of it as a network protocol,but,these days that's the appropriate term.

Brandon: So I think you hit on something that's a point worth repeating here - which is, users could come to Open vSwitch and treat it like a physical switch. They could configure ports, they could configure VLANs, they could configure static routes, they could configure access control list entries. But it didn't have to stop there. There was this enormous set of other functionality that was enabled by it being extensible. And by, as I just learned, actually for the first time, you didn't know early on exactly what form the networking side would need to take.

Early Days at Nicira

Brandon: [00:12:38] Can you tell us a little bit more about what it was like? Uh, so I remember what was it, 2007? I think Nicira had just formed and it was what, four or five people, right in that size? And you were in this small office on California Avenue in Palo Alto. And it was terrible.

Every time I went to the office and then I left and got lunch with people, I remember someone saying the Internet went down or it's like a meg and a half - it's something so basic that it seemed like a wonder that OVS got created at all.

Ben Pfaff: Yeah, the physical circumstances, uh,were not that, uh, inspiring. It was,uh, a small office with inadequate air conditioning for holding, uh, racks of equipment. Um, And, almost all of us, who were working there had had some sort of a difficult circumstance, we were dealing with.

It seemed like something like three of us had new babies within that first year or two. Honestly, it's a wonder that we managed to produce anything at all. Much less, something that influenced the industry in the way that it did... so it was, it was pretty, pretty humble beginning, you might say.

Systems Programmers vs Network Engineers

Derick: [00:13:51] So that's crazy. That's a lot of stress right - A new baby and a new startup, and you're from a file system background. Was there ever a point when you're in that first year where you're like, am I doing this? Like, is this crazy?

Ben Pfaff: You know, I tend to think of myself as a systems programmer. but I've done a little bit of work in storage and file systems. I guess I never really thought of myself as a file systems guy. I have a lot of interest in most parts of systems programming.

Um, if you look at one of the things I produced when I was at Stanford, it was an operating systems course. The educational operating system I built as part of that called Pintos, I still get emails from faculty at the beginning of every semester, all around the world, um, saying, Hey, could I have access to the sample solutions for this thing?

So, try to think of myself as a generalist. I've become a little more of a specialist in what we call SDN now. But, someone says, okay, go build me a scheduler, or, a thread switcher or, I don't know, a virtual machine monitor, I would like to think that I could rise to the occasion and actually get really excited about that.

Brandon: I want to talk a little bit about your background because I think it's relevant to where OVS went. So, your advisor was Mendel Rosenblum. And Mendel's, well-known for creating this little thing called virtualization with x86 and a company called VMware. And you aren't a network person, you're a programmer, but you're building a network switch - were there any kind of impedance mismatches, any differences between the things you valued and the people around you, or was that a whirlwind of learning protocols and interactions?

Ben Pfaff: I don't know if we're, if we're going with the illusion that I'm a filesystem guy. I would just say the packets are just really small files. But let's see here.yeah, so the impedance mismatch between networking and systems is a real one, It's a little hard for me to exactly put my finger on the difference.

When you talk to networking people, they instantly start talking about protocols, um, uh, BGP, um whatever they're, they're using to manage their network. Systems people are

usually a little more interested in, maybe how you get things done at a higher level or maybe, how you implement things.They tend not to focus so much on the protocols themselves. I guess the jokey way that I say this sometimes is that if you're a networking person, you do things in multiples of 12.You have a 12 bit Vlan and a 24 bit VNI and a 48 bit Mac and systems people like multiples of 16 and 32 instead.

Derick: I never actually thought about that, in multiples of 12,

Ben Pfaff: I think maybe that they like to, take the leftover bits and, and put some extra metadata in them.

Brandon: IP addresses are 32- bit aligned, but I know what you mean.

Ben Pfaff: Only a networking person would invent a 14 byte, ethernet header.

Brandon: And you're being nice with ATM. And what was it? 53-byte cells.

Ben Pfaff: I think those are 48 bytes, aren't they? But, I don't know, ATM.

Derick: There's a really terrible story about how that came out

Brandon: How encouraging it is to learn how we can compromise.There's a great lesson there about the political process, how we can all come together,right?

Ben Pfaff: Right. Let's just take the average.

Brandon: To achieve the most mutually disagreeable outcome that we can all say yes to.

Ben Pfaff: Yes.

Derick: So as, as, uh, okay. So as a systems person, right, coming into the network space, um,what are some of the quirks that you found, or some of the observations you made about the mindset of network engineers?

Ben Pfaff: They're always tending to talk about what such and such chip on such and such a piece of switch hardware can do.they seem to be very focused on what specific pieces of hardware and software can do.It was kind of hard for a while, um, trying to figure out, how do you make people think about how we could design things to work - how, uh, how, how we could make the software do things?uh, they also tend to be very focused on what happens in the middle of the network. So things like on your top of rack switches on your, your core switches, that sort of thing.They tend to, uh, to think about changing things based on, reprogramming and, and changing those components and putting new boxes into the network.

Focus on what you can change

Ben Pfaff: [00:18:04] And, we realized pretty early on that if you, as a startup, try to influence things by focusing on those bits of the network, you just get bogged down because

every change to one of those is a negotiation, with some network equipment manufacturer or with some, ASIC manufacturer and they don't want to talk to you.

We actually brought in, one of these manufacturers pretty early on and, uh, their first question was how many thousands of these things are you going to buy? And, as a startup, that's not what you do.So, uh, within a fairly short amount of time we realized that if we were going to have any real influence, then the way we were going to do it, at least at first was by putting software on the edges of the network.

And that's actually where virtualization came in. I just sort of by coincidence had a background in virtualization because of what I'd done as a grad student.But we didn't come into Nicera thinking what we were going to do had anything to do with virtualization. It was something that...

Brandon: So I remember that it was what, 2008. And you were trying to get everything running on these, uh, I think it was a Taiwanese ODM, uh, Quanta running on their boxes. I think I deployed a bunch of those myself and installed the OS's for those things. Those things were pretty rough in the early days when the firmware hadn't quite matured yet,

Ben Pfaff: Well they were supposed to come with some sort of Quanta-specific firmware. But if I remember right, Quanta actually shipped them to us with Google's own switch OS on them which was a bit of a surprise.

Virtualization creates an opportunity

Brandon: [00:19:27] Yeah tip the hat a little bit in an unintentional way. Uh and I remember the was it 08-09 or so around that point Nicira looking internally and going is this a sustainable approach? Is deploying to hardware ever going to work when you're asking so many people to change and they don't have a strong incentive to change. So when did the focus really shift to Open vSwitch?

Ben Pfaff: So The focus shifted to uh to Open vSwitch and we invented the name uh Open vSwitch when we started working along with Citrix on Xen and Xen Server - that was the name of their product we figured out that we needed to get an inroad into virtualization and we looked at VMware We didn't think they would be interested in working with us - and we were right about that as we learned from some folks we hired out of there later - and VMware had most of the virtualization market Xen and Citrix had a another little chunk , so we went to them and said Hey what if we built you something that would compete with VMware's virtual distributed switch. And they said Okay. Uh we don't have anything like that would be uh, good for us competitively if we had something that could match up to it, so we said okay we will do this by building Open vSwitch which can do anything basically. And then we will build a controller on top of that for you that will do VMware VDS and you have to let us sell our own product that does different stuff that will work as a controller as an alternative to this thing we're selling you.

Brandon: Well, Ben, when you were partnering early with Citrix, what functionality were they looking in your direction to enable?

Ben Pfaff: Right. What they wanted was the ability to do something a little bit like VLANs but they don't require you to use the physical VLANs of the actual network And they would support gracefully following around the virtual machines as they migrated from one host to another.

Brandon: Um in other words you take a bunch of virtual machines and you designate them as these can talk to each other over the virtual distributed switch and no matter where they actually go in the physical hosts that follow them around . It sounds very simple at this point but it was functionality that wasn't readily available at the time

And just those machines in that virtual LAN could speak to each other, right? So this was an early example of segmentation, of splitting the network into pieces and getting security benefits from that?

Ben Pfaff: Yeah networks are a good example of what we have in computers - we invent new ways to share things like networks and then we instantly need some new way to limit that sharing to partition things away again and this is a pattern that shows up over and over again in in computers and this was just one example of it in networking.

Early Customers

Derick: [00:22:16] I talked to Martin over beers one time and I told him he must get a positive response from people who've worked in service provider environments because I remember I came from a service provider environment and went into a company that does financial stuff and I was actually shocked at how poor networking actually was in a virtualized environment - It's something like VLANs but not, and then it would just be VLAN tags getting shoved out of this interface , and I thought how come there aren't VRFs or other things ,inside of the host it's essentially a networking stack at the bottom of this host, that is the actual edge. Did you find service providers were more open to these concepts in the beginning?these concepts?in the beginning?

Ben Pfaff: Service providers were our big customer base. the problem that network virtualization solves is largely a problem of scale - if you have a small environment then it's not hard to manage your ACL's; it's not hard to manage your VLANs it's not hard to keep track of these things

but some of our early customers were service providers that were hitting real limitations Um I think in some cases they might actually have been uh hitting up near the limit of the number of possible VLANs around 4,000 and, they were finding that it was a mess trying to manage those that was a big part of our story uh sort of mastering complexity. Um, another big part of our story for service providers was self-service. We repeatedly heard the story where if you wanted to deploy some computers and you needed them networked together then you had to talk to the networking team.

The networking team uh had to set up a VLAN for you and it could take weeks or longer in some cases just doing that sort of thing but once you make this virtual it's really just a matter of clicking a few buttons and saying I would like these machines to be able to talk to one

another and things happen and you don't have tough, have this lengthy negotiation with another team.

Brandon: And I want to jump on something there, so, this is a network switch, a virtual switch, that it sounds like you're going to target to server administrators as opposed to network admins because you're completely bypassing yet again uh an entity that can say no on the process of getting this deployed and getting to this functionality that users wanted to make their service provider network more manageable, or to support higher levels of scale.

Ben Pfaff: Yes these service providers tend to have something like a server administrator team and a network administrator team and it seems like they don't always have the same incentives they don't necessarily cooperate all that welland once you've got two layers once you've got your host and once you've got uh your virtual machines within that host in a sense you have a nesting of [00:24:00] responsibilities. In theory that network team might have some responsibility for the hosts network administration because at least in theory there's a network inside it. That doesn't seem to be how things actually worked, and it made things simpler for everybody I think when the servers were implementing an overlay network they were implementing network virtualization and then the job of the network and the network administrators was just to get packets from physical machine A to physical machine B. I'm sure I'm simplifying that I've never been in charge of a non-trivial network but that's my understanding.

Software Switch Performance

Brandon: [00:25:26] So Ben we've talked about extensibility initially for Nicira, building Open vSwitch, we talked about manageability. I want to talk a little bit about speed as a roadblock to getting this thing deployed and getting it meaningful and then I want to talk about availability, And I don't mean in the distributed systems sense, I mean getting things into the kernel. So, when you're building OVS and its meeting the bar for required functionality you've got this partnership with Citrix, did the speed even matter in the early days?

Ben Pfaff: Performance has always been important and there's a few axes along which it was important.There is sort of the raw network performance where, people start running packets through your switch as fast as possible when trying to see if it can keep up with line rate and that sort of thing. Um that's important for benchmarks because people are going to do it I don't think it's actually the most important for our metric because,when people do that they tend to forget that When you're dealing with this at a , at an edge , where you've got virtual machines running, those virtual machines have to actually be able to do something with the packets uh to provide something useful.

If you can get a hundred gigabits or initially one gigabit because networks have gotten faster over the time we've been working on this If you can get that in and out of your VMs but you can't do anything useful with it in the middle, it's not that uh exciting So there's that dimension and performance. The other dimension in performance that comes [00:26:00] up over and over again and for good reason how much CPU time are you using to generate this performance And so initially we were talking about uh primarily about machines that had

one core or two cores. The sort of approach that we can use these days of DPDK where you assign a core to service a NIC can't work.

Brandon: DPDK?

Ben Pfaff: So DPDK is Intel's library for network processing And it tends to use this approach where you give a core to a network processing component and then it uses the entire 100% of that CPU core.

And this allows it to run very fast but Initially, this sounded ridiculous - how am I ever going to get anything useful done in a virtual environment if I'm using all of my one or two cores just to move packets around.

Brandon: Definitely. I mean, OVS is getting created around 2007, 2008. You don't have every desktop having many cores - maybe two, maybe four, but you don't have many cores for sure.

Ben Pfaff: Right. Um,

Brandon: So this wasn't even an option?

Ben Pfaff: No one of our performance goals was to use as little CPU time as possible and in fact we would get complaints if we use more than about 5% of of a CPU

So that's sort of the direction we started from.

Brandon: And you had to pull out every trick in the book that you knew but packets are just tiny files to move as you mentioned earlier so you knew a few tricks to make things fast. You knew a few low-level C hacks to get your lookup tables to be really efficient to avoid unnecessary copies that might take up that time. Can you talk a little bit at a high level about how you get performance out of a software system that's sitting that close to the hardware?

Ben Pfaff: Yeah in a software switch there are uh three phases with which you deal with a packet. One is the packet comes in from somewhere you have a device driver there the last part the phase three is you've got the packet going out somewhere and you've got a device driver there.

So. Open vSwitch - the device drivers Aren't part of it, the device drivers are part of DPDK they're part of the Linux kernel there somebody else's problem. We can't have much of an effect on how fast they are. The part that we can influence is the phase in the middle part two that's the phase where we decide what to do with the packet where we figure out um this is where it needs to go, and these are the changes that we need to make to it. So the whole history of Open vSwitch is how do we make this decision How do we make this this network packet classifier as efficient as possible And uh we've come up with new techniques along the way and our publications,have been fairly oriented around that in fact I'm still working with uh, some researchers on uh how do we come up with even better network packet classifiers .

Brandon: And I remember that was one of the things that surprised me that the Open vSwitch kernel module was doing things faster than the previously built Linux one even though you could do a whole lot more with OVS, right - it had all this extensibility built in - but you put a lot of work into it right?

Ben Pfaff: Right and uh a big part of that story is uh is caching if you [00:29:00] look at how most software switches work even today uh they work by taking each packet and putting it through a whole list of decisions. They might first look at where is this coming from? They might look at different things that help it decide, “does this look like a valid packet?” and eventually they figure out ,"Oh this is where it should go?" And you have to do that at some level but with open vSwitch uh we came up with some clever schemes that allowed us to quickly make all these decisions in one go for packets that are similar to ones that we've seen recently.

If you look at our flagship publication on Open vSwitch it's almost all about

Brandon: Yeah, I think this is the mega flow stuff, right?

Ben Pfaff: Yeah.

Brandon: You've seen a packet before How do you very very quickly figure that out the next time and the next time and then the million times after...

Ben Pfaff: Exactly. And with caching there's always the question of how do you do cache invalidation and that part of the story is one I've never been satisfied with and it's probably the uh the part of that paper and the project that I'm least happy with over time. That's the hard part.

Getting into the Linux Kernel

Brandon: [00:31:06] So Ben I feel like we could dive into the tech details for a long time and really geek out,But That's not why I wanted to bring you to the podcast , actually it's...

Ben Pfaff: Yeah, pull me back! Pull me back from the brink!

Brandon: Let's talk about people, what every engineer wants to do, let’s talk about human processes. Let's talk about something that I think is very remarkable, which is getting code of any kind into the Linux kernel. And in your case, it's not like some little bit, it's not a single patch that you have your name on. A lot of us have our name attached to a little patch here or there, it's a monster, it's a beast, it's a big amount of code. And this isn't a six-month process. This is a multi-year process to build, to refine, to get over that barrier.

And I wanted, I wanted this to be the focus of our discussion - about what is it like? What is the human experience for a top-grade developer with a big chunk of code that they're shepherding as part of their job, or [00:31:00] even outside of it - getting that code into the kernel? So, let's, let's start with the most basic elements.

What is Linux? What is the Linux kernel?

Ben Pfaff: Sure so Linux is the lowest level code that most programmers think of running on a computer it's what we call the kernel It's what sits between the hardware and the applications and Linux is actually a pretty important kernel. It's what runs on every Android phone It's what runs on every Chrome book it's important enough that even Microsoft is building support for it into windows.

Brandon: And is this the largest open-source project in terms of active code or contributions?

Ben Pfaff: it must be up there. I've heard people claim that Kubernetes is actually more active; they're probably number one and number two - that would be my guess.

Brandon: Great - So how do we get to it already there in their OS distribution, but I have to imagine there's more to it than that.

Ben Pfaff: Yeah,the reason that Open vSwitch wanted to get into the kernel was that the kernel is the thing that's right next to the hardware. When a packet [00:32:00] arrives from a virtual machine or a physical NIC that's where it ends up first. And so if you do packet processing right there uh you get higher performance than if you do it anywhere else because if you do it anywhere else and there's a layer between you and the kernel, and so you get a higher latency this is by the way no longer completely true because there are ways that you can completely bypass the kernel

Brandon: But this isn't the code that most people get started with - this isn't Python scripting or Java code this isn't high level stuff - this is the opposite. What's kernel code like in comparison to the rest of the system?

Ben Pfaff: Right kernel code is difficult to write because it's all written for maximum performance and there's not very many barriers between you uh, and completely screwing up the computer. If you write ordinary code then if, if you do something wrong it crashes the operating system kills it, and you start it again uh, with the kernel usually if you make a mistake then your machine as a whole crashes and you have to reboot the whole thing. There just aren't very many guard rails so it takes careand, as a result the people behind the project are very fiercely protective of almost everything in the project for good reason because if they let just anybody contribute any code then that machine is just going to explode It, well not literally, it...,

Brandon: Just got a mental image of you walking with an explosion behind you.

Ben Pfaff: There's a mythical instruction in the uh the hacker literature uh the halt and catch fire instruction. But fortunately those are few and far between they don't normally really exist. Anyway.the people uh behind the kernel throw up a lot of barriers to getting things in there for good reason because they want their stuff to work.

Derick: Yeah, you know when I started in the nineties I came in at the tail end of sort of the way they used to bring people into an actual you know on the ground coding shop where

you were sort of an apprentice and then at that time the one I went into I mean all the mentors were these crusty old Unix people, and as somebody who was young I was like so excited to be working on these systems and I would have these ideas and we would have these meetings - I'm like I think we should do this And I got shut down so many times by these just - this image of these people with these long gray beards being very stubborn and obstinate right Like these Unix gatekeepers, I guess. Right?

Ben Pfaff: Now you're one of them with the uh with a Unix spirit yourself.

Getting OVS into the kernel - not Ben's first rodeo

Derick: [00:35:23] But the thing is you went up against that right? First of all, you made something that needed to be made right, that didn't exist before and it's huge It's not tiny It's not like, Oh I'm going to make this small change to this little part of the kernel you introduce something huge that was fundamental to the kernel I remember when I heard that Open vSwitch went into Linux I was like Holy shit how did they do that?

Ben Pfaff: There's a couple of things here Uh one is that this wasn't my first rodeo before this I actually had two other device drivers that I had contributed to Linux I contributed the 16 color VGA frame buffer driver way back in the nineties and I had also contributed a driver for an FM tuner card back in the nineties as well So I had some experience with Linux, but I also had a long experience with free and open source software in general. I actually have a project that I'm still working on, a 25 year project, anduh, you know collaborators come and go And you learn to work with them. The other a superpower we had here was actually another person on the team Jesse Gross Jesse fits right in with those kernel people by which I mean that he is incredibly technical and sharp and very cantankerous

He's a great guy but he fits in just perfectly with the kernel folks. So, he started out as a fairly junior person at Nicera but he very quickly showed that he was an amazing programmer.

When we found him the right part of the project which was the kernel module, he took ownership of it instantly and he was the one who did the heavy lifting of actually convincing people in the kernel community to take it.

The other thing that we had going for us as you say there was a lot of code there there was, but it didn't interlock with most of the kernel very much; it was more or less, in its new isolated directory. There were only a few places that it connected to everything else.

As somebody who manages a project and accepts things, that's actually a very good sign to me because it means that if something goes wrong and somebody fails to maintain something or there's a terrible bug you can remove it without breaking other stuff, and I think OVS fit in that category.

Why Code Gets into the Kernel

Brandon: [00:37:40] So Ben I want to jump in here. Why would anyone ever say yes on the Linux kernel team to a new chunk of code, because you are stuck with that until the end of

time? And okay maybe I'm being a little hyperbolic - you're not stuck till the end of time - you can remove things – but the bar is pretty high because people, once they're using it, don't like to have things taken away. And that extra code is something that you have to maintain, even when someone else's code changes and it forces changes elsewhere. Can you talk a little bit about that?

Ben Pfaff: Usually it's because the users want it. We had users at companies that have some influence over the Linux kernel; they're the contributors to it it's not like Linux is some uh some company off somewhere whereas a customer you pay for features, Linux is the people who work on it. And, as those people change as they continue to contribute to it then, they have some say over what goes in. They actually have some incentive to get it in, because if you're using a feature like open vSwitch and it's not in the kernel then you have a lot more work to do because when, whenever Linux changes you have to sort of go in and change your stuff to go along with it, and that's a pain -that's work. So, if you get it in there, it's less work.

Then I love this notion of Linux as a user centered design organization, because I don't think there are any turtleneck wearing designers talking about delight here, these are kernel guys, right? These are people who do code that is really critical to the experience. And if they get it wrong, the effects are potentially life-threatening.

Brandon: These are people who know that what they're doing is meaningful because it's so critical to the experience despite the higher bar for getting anything in and testing this.

Ben Pfaff: I don't think people who aren't technical realize how much of a thrill programmers and developers get out of seeing things work and making things but, that's part of it too. Yeah.

Brandon: Do you still get that thrill?

Ben Pfaff: Sometimes the things that I really like these days are when I've built something that teaches me something new.

I've written a lot of code so the opportunities to do things that are truly neware a little farther between but when I've got some new concept and I've finally embodied it in something that works, that's fantastic! And then the other thing is when I get to talk to users who've used my uh my [00:39:00] software for something especially if it's for some use that I couldn't have foreseen in advance.

We have uh one or two of those at the Open vSwitch conference every year. And are always some of my favorite talks.

Brandon: I'm wondering if Derick if you have a story about nervous excitement around coding or anytime you'd done it recently and felt that spark of joy.

Derick: Um wow. I wasn't expecting that, but I was working on a project where we had to write code that automatically strung up a set of containers with networking between them for the purpose of presenting a lesson to someone visually on the screen that they could

interact with. Like, they would select the lesson and then everything gets spun up, it went together, and we had to write code to do all that. And it was one of those things where there were so many independent pieces that were getting spun up - like and they had to be spun up with an initial configuration - that was the whole trick. You know there was an IP PBX and there's switches and there's Linux command line softphones which are a thing - you can actually from a command line make a phone call [00:40:00] and i'm all that had to be done, and there was a lot of coding Python and Go that went into it and using a lot of very new like 0.1 version libraries that were not... we'll say well-written so ...

Ben Pfaff: Sometimes it's a thrill when things work at all.

Derick: Yeah exactly. I call it yak shaving like you know how all the pieces are supposed to go together, and then once you think you've strung them all together now you got to go through and figure out why it doesn't work piece by piece you know yak shaving one nasty yak dread at a time off the yak.

Brandon: But I think it's actually a lot broader, right? It's not just coding here. It's any kind of creation where there's a user involved and you don't know exactly what their experience is going to be.

Ben Pfaff: There's a lot of responsibility I guess when you run into things as sort of a serious programmer um I know that when I find myself exploring something new I usually end up reporting and maybe fixing bugs and things that seem completely unrelated.

Brandon: Yeah, I think the more critical a system is, the more time you have to put in to getting it right. And the higher that bar has to be because of all the places the Linux kernel is used. And yeah, your laptop may reboot, but Linux isn't just controlling your laptop, right?There are a lot of embedded devices that are connected to people's lives, that are connected to rockets heading up - In fact this is part of what motivated Forward - which is this notion that if something matters and the cost of a failure is very high, then you should invest all the resources possible and the highest tech to make sure that thing is going to work in every circumstance.

Not just throwing it over the wall

Brandon: [00:42:21] So kind of coming back to OVS in the kernel, moving past Linux as a user-centered design organization, which was in no way where I thought this conversation was gonna go - what was that process like for you? Maybe walk us through the initial throw it over the wall or maybe pull someone aside and say, Hey I got this code." I think you might want it. I think it's going to be really good for the users..." What's the next step?

Ben Pfaff: So we knew that we had a lot of homework to do because we had you might say cut some corners in some ways. We knew that some bits of the system would not be acceptable to the kernel networking community, as-is. So, in the run-up to contributing this thing to Linux uh we had to go through and make a lot of big changes to interfaces especially the interfaces between the kernel and the applications that would use it. We re-wrote that layer probably multiple times until we got it to a point where we thought it would be

acceptable, and the way that we knew this sort of thing was by spending a lot of time over a period of years hanging out on this networking mailing list. We had a fairly deep knowledge of how the folks already involved in the project think and that's really important, because if you throw something over the wall and there's a lot of things that just don't look right you probably will get a bad response and you may not even understand why. I've had this happen to me any number of times as a maintainer where somebody sends some great big wodge of code and it's just completely wrong in so many ways. And it's hard to even properly write a review of that because you have a conceptual impedance mismatch with the authors of it

So, we built it we did all this work to ensure that we wouldn't have that impedance mismatch, and the goal was to make it as easy as possible for Dave Miller, the ultimate maintainer who would have to accept us to say yes, we wanted getting to yes to

Brandon: So you did the best you could in terms of getting this code as clean and as understandable as possible and you submit it... what does it look like? Is it an email?

Sending The Email to Getting To Linus

Ben Pfaff: [00:44:29] It probably seems a little comical to people these days but the way that you submit things to Linux at the time and I believe still largely done this way is by sending an email that has a patch a set of changes to apply to what's already there in Linux.

And, also very importantly there's a commit message, there's an explanation of here's what you're looking at, this is why we're doing it, this is uh you know, what's come before, this is what's coming next - it's a very important explanation of your purpose and your reasoning.

Brandon: And was that it. They said, "Yeah that looks good, and you gave us a lot of context. Thanks! We'll accept it, happily!"

Ben Pfaff: Oh gosh so that that's the way it works for something like a trivial bug fix but, for anything big it's going to be a back and forth. You've done your best to make it easy for people to say yes, but. People are people and everybody has a different viewpoint, and uh, the people with the power to put things in have a higher-level viewpoint than you do. They know about how things work in other pieces of code that are connected to this.

They know how the people above them are likely to behave. I know that sometimes Dave Miller, the networking maintainer might reject something on the basis that Linus, the guy that he is in a real sense that even if not in a pay sense, um, reports to, that Linus might reject it.

So honestly, I don't remember how many times we went back and forth with Open vSwitch. I bet Jesse does, but it's definitely a back-and-forth; you make changes and people make suggestions. It takes time and it takes effort, and you have to work on it.

Brandon: I mean, the people at the top care about evolvability and risk and interface boundaries like you mentioned, whereas the chunk that you care about is - you just want your piece in, so you can get work done, so users can get the benefits.

Ben Pfaff: Right, and so you get things like people saying well why didn't you do it this other way?

In some cases, you have an answer, and in other cases, you find yourself thinking "What way?" And then you have to go find out you know - what are they even talking about? And and then you have to explain why it's, you know, then you have to explain why it's, you know, not the right choice.

The Internet Is Character Development

Brandon: [00:45:39] I think back when I was in Scouts as a kid, we'd look back at some of the tougher hikes and go that was a character-building experience, and it was like this - I don't know eighties meme for something that sucked at the time, but you felt like you were going to get something out of it? Do you feel that way about kernel development? Has it made you tougher?Has it made you better?

Ben Pfaff: So being on the internet is character development. The internet is full of people and the people out there are often not very nice.

When I started out on the internet, I think I was kind of one of those not very nice people but I got better. If you look at my Usenet posts from the late 90s / early 2000s you'd probably find a more disagreeable person that you might not want to want to work with and there's always going to be people out there who are hard to deal with and some of them are on the Linux kernel mailing lists, and you just have to have a thick skin and address the bits of their comments that are technical and relevant and ignore the bits that are character attacks.

Brandon: All right, so there's some back-and-forth. Eventually someone says yes. It gets adopted. Do you celebrate with a cake? What do you do?

Ben Pfaff: I think there was cake there was probably beer. It was a huge victory if we were a public company, I think the stock would have gone up at that point.

It's a big deal. It means that the way forward is going to be easier, that when we get our customers to use our software, they have to do less work... it makes things easier for everybody.

Open Source Communities

Brandon: [00:48:03] All right. Any other thoughts on OVS, if you have a soap box to get on here - what would you share with the listeners?

Ben Pfaff: I think one of them is that for any open-source project don't assume that you're going to get a community built up around it quickly.

I've seen people put up slide presentations where they say okay, we're gonna make an initial release in May, and by September it will be supported within the community. That's not how it works. It's a slow process over a period of years at one point I was doing something where I would figure out on a yearly basis what fraction of the commits in Open vSwitch came from

internal at Nicira slash VMware versus external from people not directly connected with us. And the first couple of years it was near zero Yeah single digits percentage And then over a period of years several years five years or more it started growing. So it was double digits and these days it's probably mostly people who are not associated with VMware.

You have to work hard and you have to make sure that your project is relevant. Uh, you have to make sure that you have a good uh standard setup and guidelines. You have to hold people's hands sometimes. It's not magic.

Brandon: I had a very similar experience with Mininet where the bar is much lower for contribution, but there was still a few years between when you had lots of people using it for classes to demo their networks, and then when you started to get meaningful contributions to make the system better.

Ben Pfaff: Sometimes it feels good though.

I've had people go to a lot of trouble sometimes to say thank you. There was notably a guy who bicycled through the rain Sarajevo just to show up and say you know thank you for writing the software.

Lightning Round

Derick: [00:49:54] Yeah. So, let's do a quick lightning round Ben, for things that you're really excited by that you've contributed to more recently. OVN, what is it?How does it relate?One sentence.

Ben Pfaff: Yeah, OVN is software that allows you to connect up your virtual machines in a data center or another virtual environment and it'll network them and follow them around as you move them around the network.

Brandon: Cool. Okay And then in one sentence eBPF Why are you so excited by it?

Ben Pfaff: So eBPF is the way to extend the Linux kernel, you can write code in C or another language and then push it into the kernel in a safe way that performs well,the reason it's exciting is that you can finally extend the kernel without having to actually write kernel code in the same way. If it existed when we built the open vSwitch kernel module maybe we wouldn't have had to build it all, because we would have been able to get the extensions we wanted without writing

Brandon: That’s very interesting. So, would you like to hear next on this podcast? If you could get Bruce Davie on, have you already

Ben Pfaff: If you could get Bruce Davie on, have you already talked to him?

Brandon: I have not.

Ben Pfaff: He's a character

Brandon: I would love to speak to him. What do you think we should talk about with him?

Ben Pfaff: Networking? Bruce had a lot to do with the early MPLS So um I'm curious how he would change it if he was designing it today - that's a protocol I have a bone to pick on, so I'm curious about his thoughts.

And we're out...plus how to contact Ben

Brandon: [00:51:30] All right Well Ben thank you so much for joining our podcast. If listeners want to learn more about Open vSwitch and the projects you're working on or even reach out to you, how should they contact you?

Ben Pfaff: Sure so these days I think Twitter is probably a good way to start.

My handle there is Ben underscore Pfaff that's P F a F F you can also reach out to me via email or other ways um,if you type my name into Google I'm like the top page of hits so I'm not hard to find.

Brandon: And are there any podcasts you might recommend?

Ben Pfaff: Podcasts. Well, funny you should ask. I have or at least had my own podcast about Open vSwitch - it's called OVS orbit. You can find it at ovsorbit.org.I have 70 some episodes dating back a number of years. The first episodes have pretty bad audio but if you skip forward a little bit it gets pretty good. So, if you want to know more about Open vSwitch that's a great place to look and then coming up in 2021, I expect that I'll have a new podcast with folks from the VMware research group.

Brandon: Great.. We'll have to, we'll have to plug that once it's out.

Derick: Well, Ben, thank you very much for coming on the show. This has been a great conversation and we'd love to have you again.

Ben Pfaff: All right. Thanks for having me. I enjoyed it too.