Google Prodcast Season Three Episode Eight

[JAVI BELTRAN, "TELEBOT"]

STEVE MCGHEE: Welcome to season three of *The Prodcast,* Google's podcast about site reliability engineering and production software. I'm your host, Steve McGhee. This season, we're going to focus on designing and building software in SRE. Our guests come from a variety of roles, both inside and outside of Google. Happy listening, and remember, hope is not a strategy.

Hey, everyone. Welcome back to *The Prodcast*. This is Google's podcast about SRE and production software. This week, we have two guests, Andy and Wilmer, and we're going to be talking about a software system that was developed by a bunch of people, including these SREs here today. And this system is something you've used before almost assuredly. It's a little thing called public DNS or, as I like to call it, 8.8.8.8. If you know, you know. As always, I'm accompanied by Jordan. Welcome back, Jordan.

JORDAN GREENBERG: Hey, everybody. Thank you.

STEVE MCGHEE: And this week, we're going to talk with Wilmer and Andy. Why don't you guys introduce yourselves?

ANDY: Hi, I'm Andy. I'm an SRE at Google. I've been here for nearly nine years, working on networking systems, and I was once the TL for Google Public DNS.

STEVE MCGHEE: Cool. Welcome.

WILMER: And I'm Wilmer. I've been here for 17 years in a bit. And, yeah, like Andy, I like networks.

STEVE MCGHEE: [CHUCKLES] Don't we all?

JORDAN GREENBERG: That's a good way to say it.

STEVE MCGHEE: Yeah.

JORDAN GREENBERG: I like when my network works.

STEVE MCGHEE: [LAUGHS] We also happen to work together. I'm wearing my Eng London sweatshirt

today just to point that out. We're all in the same office at the same time. I think you guys probably have the same sweatshirt. Mine's too small. I don't know about you guys, but they're a little bit dodgy, as we would say back then.

OK, so this week we're going to talk about, like I said-- the official title is public DNS. We call it a couple other things. Most people will just know it as these numbers, 8.8.8.8. So why don't we first mention, like, what is it? [LAUGHS] Like, what's the deal? Why would someone even use this thing?

ANDY: I think there was a perception that, generally speaking, you use the DNS service provided to you by your ISP. And, at least back when we built this thing, they were generally not that reliable and generally not that fast. People had a bad experience of the internet because of that, because if your DNS server is slow, then everything else is slow. I'm sure you've experienced that. And we were kind of uniquely positioned to do something about that, right?

We could build something at scale. We had places to deploy it all over the world. We believed we could make the internet better for many of these people who had ISPs with maybe not great DNS servers through no fault of the ISPs themselves. And I think you can see that because, for a long time, the recommendation for, my internet was bad, was a description of how to open your network settings and put 8.8.8.8 in one of the fields. Never at 8.8.4.4.

STEVE MCGHEE: [CHUCKLES]

ANDY: [CHUCKLES] Everyone forgets about that one.

JORDAN GREENBERG: Mm, yes.

ANDY: Including you, Steve. [CHUCKLES]

STEVE MCGHEE: That's right.

JORDAN GREENBERG: Oh, no. [CHUCKLES]

STEVE MCGHEE: Sorry.

ANDY: [LAUGHS SOFTLY]

JORDAN GREENBERG: So for me, I know that I play a lot of video games, especially online video games. So I'm always switching my DNS on my router to make sure I'm using it because it's faster for

me to get that resolution to wherever I need to go. Why do customers use this today?

WILMER: Like Andy said, and above all, I think one cool thing about it is it's just a very, very, very memorable IP address, of course, which we'll talk about later. But yeah, it's fast. And in fact, the reason we built it originally was because even the founders had a pretty firm belief that bad DNS provided by ISPs was actually why the internet was slow and why Google was slow. So one major reason to even build this product to begin with was to actually make Google faster for everyone.

JORDAN GREENBERG: Wow, that's interesting.

ANDY: With the nice side effect that it affects everybody.

STEVE MCGHEE: That's right. Wilmer, you were there at the beginning of this project, right? Can you tell us a little bit about how it came about? Did you get an email from Eric Schmidt saying, Wilmer, please make a new DNS for the internet. Signed, Eric, or how did how work?

WILMER: How unnatural would it be? So the idea was coined by David Presotto, I believe, to actually-- let's build a resolver ourselves. And he took it to our team at the time-- traffic team, traffic SRE-- because we were already responsible for running various DNS servers at Google at the time, like, well, the authoritative servers and all of that. So it seemed like a good fit.

And it happened at the right time because I was just doing my second visit to the Bay Area and doing the rounds, just checking how everyone's doing, what are you working on? And someone said, hey, I kind of would like to work on this HonestDNS thing and I don't really have time, so would you like to help me? And I grabbed that because I felt like if we do this, I can talk to my friends about it instead of just saying, yeah, I work on internal infrastructure, which hey, that seemed cool. And in general, I just like doing something that wasn't just web stuff, but actually some infrastructure. So it was actually an SRE product from the beginning, which was really cool.

STEVE MCGHEE: That's awesome. So you were on this traffic team that managed DNS already, and I guess the idea was, like, what can we do to make the customer experience better? And one interesting thing about this problem is it's hard to give DNS to a couple customers. You have to kind of make it big to begin with. [CHUCKLES] It's hard to launch this in a small beta or something like that. And so this is kind of one of those problems that I think you guys were set up to do at scale, and it would be hard to do otherwise. Is there anything that you had to think about at the time around that dimension of the problem?

WILMER: We didn't know what to expect yet because, as you say, you cannot launch it, you cannot beta it, you cannot do an invite scheme, which we used to do a lot at the time. So we really just had to make a random wild guess on how much is going to draw. And, well, we just did it from memory because it's been a while, of course, but we just followed our usual DNS, yeah, provisioning philosophy of just overprovision because DNS tends to be cheap. Well, mostly. So just make sure we can serve crazy amounts of traffic and just have a footprint in every cluster, in every data center that we had in the world at the time, which I think at the time was like 12.

So we did that and we actually got by for a long time. So we launched, and within the next year we would add machines here and there as we saw demand actually coming up. But I don't remember in the early days scramble, at least, thankfully.

JORDAN GREENBERG: What sort of challenges did you face when starting this up?

WILMER: So there are two things that come to mind. So the first one is a writer at the time. The Dan Kaminsky story came out where he said, I think I found a way to poison your DNS cache much faster than you think you can.

STEVE MCGHEE: Mm-hmm. Yeah.

WILMER: As in, I can do it within hours instead of within weeks. And the idea was, well, actually 16 bits is not that much entropy. And really, if you don't randomize your port numbers, then, really, shouldn't be running a DNS service. And the initial prototypes of Google Public DNS were actually based on bind, which at the time, from memory, wasn't randomizing its port numbers for external resolution yet. So actually, if we were to launch it the way it was, it would be a service that is highly susceptible to cache poisoning.

So that was the first thing, where we had to fix that and make sure that the port number randomization doesn't get ruined or much less entropic for our load balancers, for example. So that was a major thing, and really just because the timing was unfortunate.

And the other thing was around load balancing, where one benefit of people using their ISP's resolver is around how load balancing on the internet works when you rely on DNS for it, where--

I don't know if all listeners understand enough about DNS to know what the resolver and what an authority is, but the problem is when the Google name server-- so we say, ns1.google.com, gets a

query that Steve wants to go to Google. ns1.google.com doesn't get Steve's IP address. It gets the IP address of Steve's name server. Now, if that's a name server of a resolver at Steve's ISP, then cool. Then, probably, we know exactly where he is. But if the IP address is actually of a name server, a resolver that Steve uses anywhere else, including ours, that obscures a little bit where Steve, the end user, actually is. So the ns1.google.com isn't actually able to provide a perfectly suitable answer to send Steve to the nearest Google location sometimes.

So this was a problem that plagued all these public resolvers, including Public DNS. And while we had a way to-- we had an internal protocol within Google to deal with this already. We could encode Steve's IP address-- sorry using you all the time Steve--

STEVE MCGHEE: That's OK.

WILMER: --in the query too, as one, and get a good answer back. But of course, that was internal and wouldn't work if we were talking to say, Akamai for example, or to any other CDN. So one thing we try to do as soon as possible after launch is write an RFC on adding a little header to DNS, using EDNS0 option protocol, where we encode a little bit of, well, of your IP address in the query from public DNS to the authority, to ns1.google.com, so that we can actually give you a response back that we know is proper for you instead of only roughly proper.

And this was hard because the idea of using DNS for load balancing like this is-- not everyone likes it and it's absolutely a hack. But in the end, we thankfully agreed that it is so far-- and we're speaking about 2010-- the best hack available to us, and this RFC, we needed that to actually, yeah, make it work, even for people who use public resolvers or people who are with big ISPs that don't have resolvers at every place where they have peering

STEVE MCGHEE: Cool. And so this is RFC 7871, I think.

WILMER: Correct. Yes. Some people know it as EDNS Client Subnet.

STEVE MCGHEE: [CHUCKLES] If you want to look it up, you can look it up. And so, Andy, you said you have read this many times. How would you describe reading such an article? Do you recommend it for bedtime reading or is it--

ANDY: [LAUGHS SOFTLY]

STEVE MCGHEE: --more of a keep-you-up thriller?

ANDY: Like--

STEVE MCGHEE: What do you think?

JORDAN GREENBERG: Is it a toilet read?

STEVE MCGHEE: [LAUGHS]

ANDY: [LAUGHS]

JORDAN GREENBERG: What makes the most sense?

ANDY: [LAUGHS] Most DNS specifications follow Postel's law-- be strict in what you emit and loose in what you receive. It contains some edges, Wilmer, which we have had to cross from time to time. You must not apologize to me.

WILMER: You know how hard it is to be internally consistent.

ANDY: It's extremely difficult, yes. So yeah, I mean, the problem that Wilmer was talking about was a major problem for public resolvers, and I think it's still largely an unsolved problem on the internet. But we were keen to be-- per the founders' sort of definition-- we were keen to be good citizens on the internet. We wanted to be able to make sure that, if people use public DNS, that services that needed to do this kind of location-based load balancing, were still able to do that. We didn't hurt Akamai or break Akamai in any way because, if we'd become very successful as the resolver has, we could be very impolite to people, which is something we really try to avoid.

STEVE MCGHEE: Cool. Backing up just a second. We've described the problem. We've described some of the solutions, some of the documents that have been written. Can you tell us how was or how still is SRE specifically involved in the building of this, the design of it? Is it like, oh, SRE just holds the pager, and it's cool because it's big. Or was it part of the original plan? Wilmer mentioned that. Like, he was very early on in this. Just how involved were SREs throughout this entire process. And how does it work today?

JORDAN GREENBERG: Mm.

WILMER: So, yeah, at the time, yeah, it started in SRE and it started with bind deployments. And the funny thing that happened is, so proper SRE, one of the first things I did for having a test setup is I load test it.

STEVE MCGHEE: Mm. Nice,

WILMER: And when you load test the servers, you need logs to send to it because you cannot just send random strings to the DNS server, of course. So I fetch some logs from our own name servers, mostly internal traffic, and I ask around. I asked Corp for some DNS logs from there. But one thing I did as well is I reached out to the crawl team because I knew that the crawl team downloads the internet roughly every day, and therefore, need to also resolve all the names of the internet every day. So I knew they would have interesting logs and, wow, they did, yeah, because the cache hit rate there is very low because there's a lot of typos in HTML files all over the internet.

So anyway, I asked them for their log files, and they sent those. But also, we then had-- a sign of the time-- we had an actual phone call where they told me, Wilmer, we're working on a new resolver because we have this stack that is built on something else and it's not working great, and we're building something ourselves instead. And we heard about the HonestDNS thing you're working on, and we would like to work together. So we have this. Now, we have a backend that can resolve. But how about we put a DNS front end on it as well so we can use that instead of bind? And of course I was interested because it would be nice to have a Google-scalestack with features we can add ourselves and, as I say, cluster, right, cache various things.

So that's how it started off as an SRE project. And eventually, the SWEs found us. And it's been a very, very good cooperation during my time. And I haven't worked on it for the last 10 years, so Andy knows better how it's going now, I think.

STEVE MCGHEE: Yeah, Andy, how did you get involved? What is your current involvement, I guess, as well?

ANDY: I got involved, I think, around 2016, 2017. The service had gone through absolutely massive growth. And SRE was still quite involved, but there was a feeling that maybe we had grown, maybe a little complacent with the management of it, and I saw a bit of a gap. I mean, I'd used it from the outside for years and was really curious about it.

And one of the first things I sort of did was sort of try to look at the system holistically. It had grown very organically over the time, right? It now consists of three separate components that talk to each other. We had some deployments on the edge of our networks, some deployments in the central part. We didn't really have a great capacity planning story, and it had definitely hit that size where losing a

bit of capacity can cause a kind of a stampeding herd problem, right? Lost a bit of capacity here. Oh, no. OK. Well, this bit's overloaded now. Oh, no. Someone drained that. And now, you're having a really bad day.

And one of my really early realizations here was like, DNS on the internet is very, very, very sensitive to dropped queries. It's extremely sensitive. People think that clients retry et cetera, but they don't retry for three, four, five seconds. And so you really are hurting pretty much the entire user experience of thousands of people when you do this.

So I sort of started poking around. And then, we had a number of very large DoS attacks. The botnet's called Mirai, which has been written about publicly. You can go read Krebs On Security about Mirai. They turned up with an absolutely huge attack. I think they attacked us, they attacked Dyn, a bunch of different people. And we suddenly had to sort of bring the whole SWE and SRE team together to be, OK, all right. This was really bad. We were down for, I think, seven minutes or something during the Mirai attack.

STEVE MCGHEE: [CHUCKLES] Yeah, wow.

ANDY: How are we going to prevent this from happening again? And there was a lot of back and forth across that boundary, right? I think you could have said, oh, well, SRE would take the capacity planning and the SWEs would go and look at the code and see if they could make it faster. But that's because, I think, of the legacy of the development of the system to which I owe to Wilmer. That was not how it played out, right?

I started poking around in the code along with a few of my colleagues. I think we have a few bottlenecks here. I think we can do something about it. And we spent quite a lot of time with the dev saying, look, this isn't a capacity planning problem, right?

STEVE MCGHEE: Right.

ANDY: There are traffic patterns that are innately harmful to the service that we are now seeing being exploited. We need to adjust for that. Initially-- and I don't mean to drop my SWE colleagues in it-- one of the responses was, can't we just have more capacity?

STEVE MCGHEE: [CHUCKLES]

ANDY: No, I'm afraid we're at a size where that is becoming-- [LAUGHS] challenging. And so what the

way that's continued through the development of the service is we've launched many new features for Google Public DNS, I think DNS64, DNS over HTTPS, DNS over TLS, DNS of QUIC, a whole bunch of things. And it has always been this back and forth between dev and SRE. Dev have largely wanted to drive most of the feature development. Although, I think DNS over HTTPS mostly came from SRE saying, it would be really good if we did this. And then, I think the Chrome people and the Firefox people talking to us and saying, it'd be really good if you supported that.

And then, we had to go and think about, OK, if we are going to do this and Chrome users-- 1% of Chrome users-- can opt in to this, what does that mean for the system? What does it mean for how we qualify releases? What does it mean for capacity planning? How do we link all these things together? And so we've always had this back and forth across it. And Wilmer has contributed code to Public DNS in its early era. And here are my-- well, I guess it would have been nearly 10 years later? Nine years later-- my code is in Google Public DNS as well because I needed to fix a few bugs. And I talked to the dev team. I'm like, hey, do you if I-- and then, I fix it.

And I've always kind of held this philosophy as well. Maybe it's an incorrect philosophy. It is difficult to support a system as an SRE when you don't at least have some understanding of the structure of the code.

STEVE MCGHEE: For sure.

ANDY: I mean, that you know this piece connects to this piece, and that is how it's conceptually modeled, OK? For a long time, I didn't know there were three caches inside of it, right?

STEVE MCGHEE: [CHUCKLES] Yeah.

ANDY: And that meant we made odd decisions. I would think, oh, we have enough memory. Why is it saying that it's-- why am I cache missing? And so I've always wanted to encourage that. And a really fine way to do that is to contribute, to fix a bug, to go performance hunting, things like that.

JORDAN GREENBERG: So that's really cool. So--

STEVE MCGHEE: That's awesome

JORDAN GREENBERG: --you have it, this claim to fame, saying, you edited Google Public DNS. And that is now written into your life story of how you made this early on, how it changed, how you support it. But now, we want to know, if you had to build it again, less ugly, what would you do differently?

STEVE MCGHEE: [CHUCKLES] And you can't just say whatever it is right now. Like, you can't just assume precious--

JORDAN GREENBERG: Yeah. yeah,

STEVE MCGHEE: --completely.

ANDY: I would have thought about denial of service attacks earlier, I think. It's quite difficult to pull that in. I mean, Wilmer definitely wants to tell me I'm wrong and tell me that he thought about it a lot at the time.

STEVE MCGHEE: Wilmer's got ideas.

ANDY: [CHUCKLES]

WILMER: We did, but you need to experience them before you really learn how to deal with them. We absolutely thought of it, and also, amplification attacks and other things. So we did what we could predict with the 2009 knowledge we had, but we learned a lot, obviously, by actually running the service, and you cannot beat that.

JORDAN GREENBERG: Mm-hmm.

STEVE MCGHEE: So it turns out that it was a trick question just to get you to say exactly that, Wilmer. This is the answer. It is like, yeah, we don't want to go back and say like, well, if we were to actually waterfall it properly, we would have done the following. But no, we learned from how we built it and how it worked, and we didn't predict the DoS two attacks and the emergence of QUIC and all these things.

JORDAN GREENBERG: Yeah.

STEVE MCGHEE: And they happened, and we evolved over time.

JORDAN GREENBERG: It sounded like it needed to be ugly.

STEVE MCGHEE: That's what I would imagine is probably the case for lots of services.

JORDAN GREENBERG: Yeah.

ANDY: Yeah. I think I'm fundamentally an incrementalist when it comes to this kind of thing. I think

there are many paper designs you could make that would, on paper, be better. But I go and look at the code, and yeah, it's a system from 2009. Is that 15 years? It's a long time now — of stuff in it.

WILMER: 2008, even.

ANDY: Yeah, I mean the thing I always remember is, almost all of those lines are there for a really good reason. I definitely experienced a bunch of-- I'll call it growing up, working on that, and working on a different system. Google's authoritative DNS server, which I was also a TL of for a while.

I definitely felt that urge to be like, no, sweep away all this historical detritus. And in each case, hubris came, and it taught me a terrible lesson. No, I can't remove these simple rate limit things here. They're there for a reason. Those are important. I removed them, and the performance got substantially worse.

OK, well, that's an important lesson in being an SRE is learning when to admit that you are wrong. And I was wrong a lot early on.

STEVE MCGHEE: You mean you can't just understand things perfectly and burn it all down because you don't like it?

ANDY: I mean--

WILMER: Come on.

ANDY: We can barely describe to each other how we feel about very simple things, like food. I don't know how we would describe how code works. This language is used, I think, as Terry Pratchett said, to tell the other monkeys where the fruit is. It is very difficult to communicate about complicated topics using that language. The good fruit is here.

STEVE MCGHEE: So one of the important things about setting up DNS when you're like an end customer is, it's not dns.google.com. You literally can't do that. You have to use this crazy thing called an IP address.

And one of the hard things about this is, normally, when you get an IP address from your DNS provider, it's some random set of numbers, and it's hard to remember. But Google is this magical beast that got this awesome thing. 8.8.8.8, 8.8.8.8. Also, 8.8.4.4, I forget this time. But what was the story behind that? Did we just have this sitting around? How lucky did we get?

WILMER: So fun thing is, initially, it wasn't. So initially, the IP address was 74.125.125.125, which is very repetitive but also very long.

STEVE MCGHEE: Yeah. What did it start with?

WILMER: Yeah, exactly. And I thought like you, these IP addresses need to be memorable. But also, I was constrained by no one caring what I'm doing. So I just went shopping. And I searched for all the IP space we owned at the time. And I tried to find a /16, where I just knew that if I just repeat the second exit twice more, then I have a pretty nice IP address.

So that's how I found that IP address. And unfortunately, it was reserved for a new cluster somewhere. And there was some back and forth. And can we please get it? It's nice. Believe me, we need it.

And eventually, we got it. And then we had an eng review with Larry, one of the founders. And his verdict at the end was basically-- and this was when we were about ready to launch, his verdict was, really cool, but our IP addresses are terrible, so please fix that.

So there went our launch plan. And we had to go to other friends. And they got us some IP addresses. So that was 8.8.8.8. And they are just the most memorable IP addresses I could imagine.

JORDAN GREENBERG: Yes, I would definitely agree to that.

STEVE MCGHEE: Iconic even. Yeah.

WILMER: Graffitied on walls here and there every now and then. It's very interesting to see.

STEVE MCGHEE: Yeah, that is pretty awesome when you can see that just out in the world. It was not a marketing stunt, people just did that. That's pretty awesome. I hope you guys are proud to have been a part of that.

WILMER: It was a highlight of my career.

STEVE MCGHEE: Yeah, that's pretty darn cool.

JORDAN GREENBERG: Yeah.

STEVE MCGHEE: I have one final question, which is, of course, silly. And you can choose to ignore it. But is it always DNS? Andy's nodding.

ANDY: Disappointingly, surprisingly, alarmingly large amount of times it is. And it's not that the record doesn't exist. It's always that the DNS server was slow or the stuff gets slow as a result every single time. GetAddrInfo should not be slow. Every system assumes it's fast. And if it's slow, weird stuff starts to happen.

WILMER: Yeah.

JORDAN GREENBERG: I have a follow up question. If I am using a service that interacts with DNS, it's a network service, and I get a Retry button, do I press the Retry button or do I wait for it to retry? And are these things built with retries? Does the button actually work? Or am I making everybody's--

ANDY: It actually works.

JORDAN GREENBERG: OK.

ANDY: But I would recommend pressing the button, to be honest with you, because otherwise, you have to wait for whatever the OS-level timeout, which is generally, many seconds more than you're willing to wait, 5 or 10. I would press the button.

It's very interesting to note that sometimes, when we have a problem with Google Public DNS, sometimes, the queries per second we receive goes up, because there are lots of queries that are just abandoned because someone has hit Repeat.

So we get a query, we don't answer it in a timely fashion, and we get another one from you because you hit the Refresh button. So sometimes, going up is an indication that something bad is actually happening, because we're seeing all the retries come from people who didn't get their first question answered. They ask a second one.

WILMER: This reminds me of the fun observation we made a long time ago, which is that the cache hit rate to 8.8.4.4 is noticeably lower than to 8.8.8.8, because you sent a query to 8.8.8.8, and you don't get a response because the query you sent is actually a thing that doesn't exist. And therefore, you try again. And you send all your retries to 8.8.4.4.

So if your query was easy and was answering cache, it comes from 8.8.8.8. And if your query was so weird, you just send it twice, once to 8.8.8.8 and once to 8.8.4.4. So the cache hit rate is half of the one of 8.8.8.8, I believe. It's crazy.

ANDY: Yeah. Noticeably different. And the amount of traffic is noticeably different as well.

STEVE MCGHEE: Yeah, I think that's a great example of an emergent behavior that is unpredictable. Well, I mean, maybe someone very clever could have predicted it, but I certainly wouldn't have. And it's only really observable after the fact and with a bit of head scratching. That's a good one. That's pretty cool.

WILMER: Yeah. You learn it when 15% of the internet uses you.

ANDY: Yes. [LAUGHS] You learn a lot.

JORDAN GREENBERG: OK. So hit the Retry button maybe 600 times. Refresh my queries whenever they don't go through. And do that at a scale of eight billion people. And I swear, everything will be just fine.

ANDY: I mean, if we didn't answer your query the first time, you should ask us again. I mean, we're rapidly working to fix whatever the problem is. Certainly, I've been paged.

JORDAN GREENBERG: [LAUGHS]

ANDY: But yeah.

JORDAN GREENBERG: So last question for you both, how do you see SRE changing in the next two years?

ANDY: I think more and more in the outside world, I am seeing SRE evolve from a DevOps plus plus, which I think would be the worst way that the SRE could be implemented in the outside, and more towards the ethos that I observe on a day-to-day basis, which are SREs are basically, frankly, miserable generalists. We'll specialize when required, when the time needs it. We are generally fairly cynical about systems, and skeptics, and looking for places for things to go wrong, which is a very difficult thing to carry around in your personal life.

And I observe more and more of that happening in the outside world. I think, probably, until fairly recently, I have friends who've gone to SRE roles outside. I've seen a lot of them do the Terraform. I mean, I don't mean to denigrate Terraform at all. It's a wonderful tool. And there was as well, in Google, a period of time where SRE was used to do SRE work. And it was almost sort of a division between the two.

I'm seeing those walls come down. I think, because once you get systems over a certain size, it is not possible to consider these two things in isolation. Reliability is not a product shipped by SRE. It is a product shipped by SRE and the SWE teams they work with.

And I'm seeing that attitude become increasingly more prevalent. And I think, to be honest with you, people like Wilmer and I have always walked around with that in our heads. Certainly, I look to Wilmer as an example when I joined for how to be an SRE. Wilmer did not see boundaries. Wilmer just saw problems. And occasionally, solutions.

And I'd love to see that be more prevalent in the outside world. The SRE book is great, but I think it's a little light in this particular section, this miserable generalist. Go off and find where the problem is, and start digging, and take people with you. kind of ignore the role aspect. That's just my specialty. I'm here to solve problems.

STEVE MCGHEE: Yeah. Right. How about you, Wilmer?

WILMER: Yeah, hard to beat that answer, really. What I've just been seeing over the years is it just makes everything more complicated. And people specialize, and the thing gets more complicated. And we shard the team into two, so that everyone can reduce the number of things they need to understand.

I would like there to continue to be many horizontal people who actually do understand how the whole thing works, roughly, because it's so much more fun. It's hard, but it's also so much more fun.

ANDY: Well, I mean, just for being more fun, I think with the horizontal people, you can narrow in on solutions faster to problems. If you have a bunch of people who have approximate knowledge of how many things work, they can help stop you chasing blind alleys early, so you can keep the SRE team from wasting their time and energy in a place where you know this is not going to work out.

And to be honest with you, I've been an SRE for nine years. I'm relatively senior at this point, although that is a very weird thing to say. That is about 70% of my job is to keep the blind alleys from being explored again.

We don't want to do this one. We have done this one before, and it has never worked. Well, you can only really get that through experience. And you can only get that experience by being given the opportunities to go down the blind alleys, which means you need to have an approximate knowledge

of everything. If you end up very specialized, it can be very difficult to create these new people.

STEVE MCGHEE: Awesome. Well, thanks to you both. Before we sign off, is there anything that you'd like to add to our listeners that are, again, their SREs, or pro to SREs, or SRE-adjacent folks way out on the internet. There's several gazillions of them. Any advice? Or just places to find you on the internet and your rantings or insights.

ANDY: My advice would be, be curious. Be curious about why things break. Be curious about that weird thing you saw happen. By being curious, you would normally learn something you didn't know before. And then later, when you don't expect it, that piece of knowledge will be interesting to you. It will help you out in some other place.

I always think about the example of the physicists studying string theory, and they suddenly discovered that Calabi-Yau theory, which is some theory of how space topologies interact. Actually, it was exactly what they needed. And the mathematicians just picked it up and dropped it along the way. SREs are a bit like that. I remember something about that. I can help you with it.

You can't find me on the internet. My rantings are all private, I'm afraid. You'll have to come to the Google London office and buy me lunch. And I will happily explain things to you.

STEVE MCGHEE: Excellent.

JORDAN GREENBERG: Hopefully, the listeners aren't cats.

ANDY: [LAUGHS]

WILMER: I have no idea, honestly. Indeed, learning, and being curious, and look at TCP dumps every now and then, because actually, it's a lot of fun, stuff like that. I have no rantings or ramblings online. The only social network that matters to me is Strava. So that's all.

STEVE MCGHEE: Excellent.

ANDY: I need to add you on Strava, Wilmer, clearly. [LAUGHS]

WILMER: Yes, let's do that.

STEVE MCGHEE: All right. Thanks very much. That was great. Thank you again, Jordan, as always, Thank you both for coming today. And have a great day.

[JAVI BELTRAN, "TELEBOT"]

JORDAN GREENBERG: You've been listening to Podcast, Google's podcast on site reliability engineering.

Visit us on the web at sre.google, where you can find papers, workshops, videos, and more about SRE. This season's host is Steve McGhee with contributions from Jordan Greenberg and Florian Rathgeber. The podcast is produced by Paul Guglielmino, Sunny Hsiao, and Salim Virji.

The podcast theme is “Telebot” by Javi Beltran.

Special thanks to MP English and Jenn Petoff.