EPISODE 20

[INTRODUCTION]

[0:00:00.6] JM: DNS stands for Domain Name System. This is the naming system that maps the entire internet. It associates information with domain names. More specifically, DNS specifies the mappings between numerical IP addresses and domain names. Most engineers know these basic facts about DNS but they may not know how much engineering a complex company like Etsy or Zappos puts into their DNS configuration.

Dynamic DNS allows for intelligent response so that a resource is served from the most efficient place even in the face of a DDoS attack or just a routine failure of cloud servers. Phil Stanhope is the VP of technology at Oracle Dyn and he joins the show to explain how modern DNS works and the role of a DNS provider. Full disclosure, Dyn is a sponsor of Software Engineering Daily.

[SPONSOR MESSAGE]

[0:01:02.6] JM: Artificial intelligence is dramatically evolving the way that our world works, and to make AI easier and faster, we need new kinds of hardware and software, which is why Intel acquired Nervana Systems and its platform for deep learning.

Intel Nervana is hiring engineers to help develop a full stack for AI from chip design to software frameworks. Go to softwareengineeringdaily.com/intel to apply for an opening on the team. To learn more about the company, check out the interviews that I've conducted with its engineers. Those are also available at softwareengineeringdaily.com/intel. Come build the future with Intel Nervana. Go to softwareengineeringdaily.com/intel to apply now.

[INTERVIEW]

[0:01:55.3] JM: Phil Stanhope is a VP of technology at Dyn. Phil, welcome to Software Engineering Daily.

[0:02:00.6] PS: Thanks for having me.

[0:02:01.1] JM: I want this conversation to go in a couple different steps. First, we'll talk a bit about DNS and what it means for an engineering point of view. Then we'll walk through some simple examples of how people use it and get to some more advanced modern complex infrastructure discussions of how DNS could be useful as a utility.

Starting from the beginning, DNS stands for Domain Name System, and this is the naming system that maps the entire internet. It associates information with domain names. More specifically, DNS specifies the mappings between numerical IP addresses and domain names. If I'm a software engineer, what are the other fundamentals of DNS that we should all know?

[0:02:51.7] PS: That's a good question. Again DNS is what you just described, Domain Name System, and it's for mapping addresses. Now, there're two address spaces. There's the v4 address space that sort of most of us on the internet have grown up, but there's also v6, and they're running concurrently. That's something that you need to consider because some of the very large players on the internet, they're announcing both v4 and v6 space and the path that a packet may take over the internet can be very different for a v4 versus v6. That's one thing to consider sort of at the network level.

I guess the other thing that I've learned to say which some people like — It's a bit pedantic, I guess, but is that there is an internet. There's millions of internets in both v4 space and in v6 space. If I'm sitting in the same room with you and I try to go to a website by typing its common name in it, say, twitter.com, the path that my packets might take to get me to that webpage may be very very different based on my mobile carrier versus yours. IT may be traversing a v4 network purely. It may be traversing a v6 network purely. It may be hopping in and out of both.

Often, you never have to know about that. Most people don't even have to know how that really works but if you're a developer and your mission-critical, high-availability, low-latency API endpoints, these are the one you really need to start to worry about these types of things.

[0:04:26.7] JM: If somebody in China opens up softwareengineeringdaily.com on their mobile phone versus me opening up softwareengineeringdaily.com on my mobile phone, give me a picture for how those two experiences might work at a lower level. What's going on in the routing infrastructure along the millions of internets that is causing those domain names to both map to the same information?

[0:05:00.7] PS: Okay. There're a couple things. One of the things that we do at Dyn also is we monitor the internet with a product that we call internet intelligence. Though we happen to know that on average every packet from your device, to the server you're trying to communicate to, and there may be dozens of servers behind a particular webpage, is on average about 13 hops. That means there're 13 public routers somewhere along the way. Some of those routers might be not performing very well. Some of those routers might have small pipes associated with them. Some of those routers might take you through proxies and firewalls which may induce packet loss or certainly induce latency into it. That's one thing that can happen is that you'll take a different path and the many different paths you take, any one hop can be a problem area, and DNS can't solve that for you. That's a network monitoring level consideration.

There's another consideration, which is we tend to think simply that, "Oh! I got a web server and I've got an IP address," maybe I leased an IP. Say, I'm hosting it out of a cloud provider and a compute instance and I've been given a public IP. I've got a server and I've got one IP. If you're really good with tools like Nginx, you know that you can host many different servers or services through the same HTTP, HTTPS proxy, and that's just configuration issues. How well and good you are at configuring Nginx, but that's one IP.

It's also quite possible that if you've got one server, one IP, and that user is in China and your service is on the East Coast, they've got a long way to get there even though it still might be on average, 13 hops, it's from the other side of the planet and physics gets involved; the speed of light. You can only push packet so fast down a fiber-optic wire. If I'm on the East Coast, short path, I'm probably going to get much better response time. That's one level is consideration.

I'll introduce another term because I'm looking through some of the notes and it's not there. It's Anycast. That single IP in v4 speak that's a /32. It's a single address out of one of a potential of 4 billion addresses, and that's yours. With Any Cast, you have to widen it at a minimum for it to work on the internet to what's called a /24 in v4 space, so 255 potential addresses that you could communicate to.

Then with Any Cast, I tend to refer it to as you can lie to the internet. You can say that you're in more than one place at the same time. With any casting, you could have an Any Cast edge that was very close to China, maybe in Taiwan or in Tokyo, or Seoul. Then if you've done that, you can do with Unicast or Any Cast. You could have two Unicast addresses and have DNS steer you to the one that's closer, or you could have a single Any Cast address or what seems to be a single Any Cast address and the networking routing infrastructure will take you to the closest address.

At Dyn, we're very familiar with both of these techniques and we use them to manager our service and our top customers run a hybrid of Any Cast and Unicasting on the public internet and allow us to provide naming over it.

[0:08:37.7] JM: I think one way that we can look at DNS is it's essentially a database, so anybody who is looking up a domain name is essentially accessing a database and anybody who stands up a website is performing an insert on that database. You set up the website, you assign the domain name to it. This globally distributed database where anytime you enter a domain name in a browser or on a console or whatever, you get an IP address and you can do stuff with that IP address.

Give me an idea for how that big distributed database works. How are things propagated? How does it insert work, and how does an update work, how does a read work?

[0:09:29.5] PS: Okay. That's a great set of questions. DNS has been around for very long time. You're right to refer to it as a database. I would also go further to say it's an eventually consistent database. In some ways, with all of the rage that has been going on for the last 10 years or so around NoSQL, it is the sort of granddaddy of NoSQL databases.

I say that because it is, and so how does that work under the covers. You're right. If browser doesn't understand what example.com is, it's going to have to make a DNS lookup, which is a query. It's a read request to find out what is the actual address of the server that I need to communicate to. That's going to communicate to the edge of a DNS provider's systems. It could be your ISP. It could be Dyn itself. It could be one of our competitors. It could be something where you tend to talk to — People think of Google as a DNS provider, and they are. But there' two core differences or types of DNS systems to get that answer in the first place.

First is the recursive. You, as an internet provider, you must provide a recursive service to your customers because otherwise they won't be able to get anywhere. At Google — For many years, those open recursives or the recursives that you would see, they weren't necessarily open. They might be closed just to your ISP. For Comcast, which is my home provider, they give me a recursive and I can communicate through it. You don't think much about it.

A recursive is actually a cache. It's kind of like a CDN edge node in the sense that it's going to remember answers for a period of time. Every DNS record has a time to live associated with it, a TTL, and that recursive will find out the answer from an authority, cleverly called and authority, the master. That authority is the definitive place where you can store manage your records as a user of DNS systems. You do not manage your records in a recursive. Google 8.8.8.8 is one of Google's well-known Any Casted IP addresses globally, is a recursive endpoint, and they allow you, anybody, to communicate through them.

Dyn and Oracle Net, we offer our own open recursive we have for many many years called Internet Guide, and there a number of others. In fact, there are thousands of open recursives on the internet that you can communicate through. They in turn go to the authority, and the authority is really where the core of the DNS master records are kept. That's typically going to be managed via an API or some form of a user interface. As you make changes to your records, they will be propagated out — They'll be available to be answered from the authority.

I'll actually hold on a deeper answer on how authorities are often structured and how we structure ours here. The recursive, if it doesn't have it in the cache, it's going to hit the authority and the authority is going to give up the answer for, for example, in A record, which is a v4 address. You'll get that answer. The recursive will hold it for it's time to live, which is — Then the past time to lives were very long; hours, maybe even days.

In the modern internet, time to lives tend to be very short. Technically, they could be zero seconds, but in reality, on the open internet, they never really work when they're less than 20 or 30 seconds. By having a short TTL, what you're allowing is the ability to fail over or load balance through DNS mechanisms to another point and location. The authority is the one that's going to tell the recursive how long the answer is good for, and then the recursive is responsible for keeping its cache warm and remembering that answer. Just like a CDN cache however, if there's a lot of traffic, that cached answer might get flushed out of cache to answer the next set of 10,000 unique things that came along, and that's the art of running a recursive versus running an authority.

[SPONSOR MESSAGE]

[0:14:00.6] JM: Ready to build your own stunning website? Go to wix.com start for free! With Wix, you can choose from hundreds of beautiful, designer-made templates. Simply drag and drop to customize anything and everything. Add your text, images, videos and more. Wix makes it easy to get your stunning website looking exactly the way you want. Plus, your site is mobile optimized, so you'll look amazing on any device.

Whatever you need a website for, Wix has you covered. So, showcase your talents. Start that dev blog, detailing your latest projects. Grow your network with Wix apps made to work seamlessly with your site. Or, simply explore and share new ideas. You decide. Over one-hundred-million people choose Wix to create their website – what are you waiting for? Make yours happen today. It's easy and free. And when you're ready to upgrade, use the promo code "sedaily" for a special SE Daily listener discount. Terms and conditions apply.

For more details, go to wix.com/wix-lp/SEdaily. Create your stunning website today with Wix.com, that's wix.com.

[INTERVIEW CONTINUED]

[0:15:25.6] JM: Dyn has been around for a while and you have grown in parallel with this growth of cloud infrastructure, the on-demand — I could request server from Amazon Web Services or Google. How is DNS different for the companies that are built on cloud infrastructure, this on-demand infrastructure in contrast to the decades' old bottle of on-prem architectures?

[0:16:01.8] PS: Another good question. Part of me wants to say there is no difference, but there are some subtle very important working differences. 10, 15, 20 years ago, if I was a company that had been around and now wanted to come on to the internet and had my own data center, had my own internet connectivity, my own slice of address space, in that time it would've probably been v4 space, I would control my domain name.

Invariably, a lot of enterprises control their public facing internet names through systems that they've had internally for many years. Microsoft Active Directory is a DNS server, and the DNS records that you might control for the inside of your organization, you could also control some of your publicly-facing ones with systems like. You have complete control over, saying, "That address, 192.168.32.41 is www.example.com. I know that because I'm the operator of that address space. I'm the operator of the data center. I'm the operator of the server."

Now, let's sort of think about it from a cloud perspective. You spin up a compute instance or a container with a public IP address in one of the cloud providers, you're going to be giving typically two IPs, an internal IP that's useful maybe if you have some other hosts in the same VPN or something that you could communicate with, but you're going to get a public IP, so you know the public IP address.

At that space scenario, just we're trying to map that public IP to an A record. It's no different than the enterprise scenario. There's another thing that happens whenever a cloud provider spin up an instance for you. They typically will also create a DNS name for that server that's accessible on the public internet, but it's going to be in their namespace. You will get an Amazon derived name and you could use that but you've lost your brand if you care about your brand. It wouldn't say softwareengineeringdaily.com. It would say some very long random numbers.amazon.com. It's not exactly amazon.com, but I think you see my point.

There's another type of DNS record you can use to overlay that, a C name, which is just an alias. You could say, "I want softwareengineeringdaily.com to be C named to that other address, the other name that was provided from another domain name. DNS systems will then look up both of those records whenever you're trying to resolve softwareengineeringdaily.com and your browser has a built in stub resolver and it will do that work for you and just help the user get to the site that they're trying to get. That's one thing that's a little bit different with cloud, is the introduction of C names. Whether you care to put a C name in place over the C name they've given you or go write to the A record, that's your, and you do have to think about it because there are sort of ramifications for that.

One last point I'd say is that CDNs invariably also issues C names for you as well, and so it's the same scenario. CDN is doing the same type or — Sort of slight obfuscation. If you know how DNS works and how to use some low-level commands you can figure out all of these chains and where things really are, but it's just sugar coating and wrappers to make it easy for people to navigate and get to where they're trying to get to.

[0:19:43.8] JM: In addition to the changes in how companies are setting up their infrastructure, that a lot of companies are doing it via the cloud because infrastructure as code makes things easier. These economies have scaled that AWS can offer makes it cheaper to run on the cloud for many businesses. It's led to a dramatic increase in web traffic in addition to mobile usage. The fabric of the internet has changed, or at least the density of traffic. Are there any second order effects that the DNS layer of the internet has undergone as this cloud stuff has proliferated? What are the problems that can occur for an application in the age of the cloud that are related to DNS that were less likely to occur in the times of on-prem being the prominent model?

[0:20:53.0] PS: One area that I can think of is that how you start to solve or worry about high-availability problems. If you're in a cloud, you're going to use a load balancer. If you're on premise and you had enough demand against your service, you're going to use a load balancer. Those things aren't changing really that all conceptually. If you also want to shape your traffic and be more available, now, you need to be in multiple regions of a cloud or potentially multiple cloud provider.

Any one cloud provider, they're going to have to have very fat pipes coming in and out of them, and one thing for both the DNS providers and for the cloud providers that's, I think, a little bit different than, say, 10, 15 years ago is that there's this perception and there's obviously a certain reality to it that if you attack a cloud provider you can create big blast radius. If you can saturate the edge of a cloud provider or a specific load balancer for somebody's service, you might actually be causing congestion for all of the neighbors. Some of us call this noisy neighbor problems, and that's also related to virtual hosting and with VM's running on a host. A noisy neighbor might take all the CPU cycles or might take a lot of the disk I/O and you might not get the throughput and performance that you were looking for in your VM or your container. The saving is also true with the traffic management issues.

The club provider's edges attract lots of traffic and potentially malicious traffic. DNS providers, they attract a lot of malicious traffic as well with a simple idea being that if you could impact those you could impact a lot of people. Clouds and DNS providers, we have to invest significantly in DDoS management and mitigation and how we run our systems. That gets to something I mentioned earlier, like Any Casting, making sure that you're available in many places on the internet for your key service endpoints and then using a combination of DNS and network announcements to shape when you need to do maintenance or where you want traffic to go. Those are all challenges that have evolved dramatically over the past decade or so.

For the simple case, you're kind of oblivious to this as somebody who's just spinning up a new API or a new application and you generally don't have to worry about it, but if you become successful, well, you're going to generally start to have to worry about these types of things and shape your traffic, shape where you're answering from, shape your where services are end-pointed from. If you're collecting information that has any privacy with it you, you might be obligated to operate some of your endpoints, specifically, in Europe for example, and a DNS system can help you manage those complexities.

[0:24:05.3] JM: When people think about load-balancing or scaling up to respond to an increase in user traffic, they're typically thinking — At least from my point of view, they're thinking about, "Oh, I need to spin up more machines, more virtual machines," or, "Oh, I need to tell my load balancer to spread the traffic among these things in a slightly different way."

I'm not sure if there's many people think about, "Oh, I should be having my DNS layer be changing how it how routes traffic, the DNS itself."

When you're talking about using DNS as a way to respond to an increase in traffic whether that's malicious or not, what exactly is going on at the DNS layer? Because I've got all these different knobs I can tune for my cloud infrastructure. I can mess with the load balancer. I can mess with setting up more servers and, as you're saying with DNS, I can change the routing of my traffic. How am I looking at these different options as my service is undergoing an increase in load or an increase in failures?

[0:25:18.5] PS: Now, we're getting into advanced DNS use-cases and advanced load-balancing use-cases and they work together with each other, so you're right. I've got more demand on my service. I need to spin up or compute, or bigger instances because of the I/O or the network bandwidth needs that I have. Correspondingly, you also might need to be spinning up and doing some DNS load-balancing and traffic shaping or traffic management.

Let's give a simple example. The one I'm using, a very Any Cast technique. If BGP networks, border gateway protocol, which binds the routers of the internet together, they're always going to prefer sort of the shortest top. What does that mean? It means that if I'm on the East Coast and I ask a question towards like, for example, our DNS edge, the shortest path from where I am would take me to somewhere on the East Coast. If I was on the West Coast, the shortest path would take me to some one of our data centers on the West Coast, provided we've, of course, managed all of our BGP announcements correctly. The same thing would sort of span the globe. On average, DNS providers have six or seven global regions just like the clouds of six or seven global areas of the planet that they break things up into. And you can keep your traffic locally within that.

Often, that is enough and was enough, say, 10 years ago, to do the first generation of traffic management. Keeping answers close to where the users are asking the questions. That's one way of — It's just one technique.

Then the next technique as well, but, yes, I am in the region, but I've got four endpoints that can serve all of my traffic. Then you can start to do load-balancing, round Robin load-balancing. So just like you can round Robin load balance in a load balancer that's picking which compute node to send work towards, you can round Robin load-balance ion the DNS system and we will just randomly pick one of the four potential v4 A records and give that up as an answer.

You can also weight them. I know I'm biasing towards a certain — One of those nodes, and that's the one I want to send the traffic towards, because as the operator, you might know, "Well, two of those four nodes are really big load balancer, load balanced infrastructures, and the other two, they're really for failover and/or fallback, and I only want traffic to go there 10% of the time. Those are the types of things that we let you configure into your answer strategy.

[0:28:00.6] JM: Going back to a discussion of what's changed since the days of mostly on-prem software companies, we've gone from this time where a website request would be served by just a server and then we had servers hosting multiple VMs. We still have that, and then there's some increased layer of redirection that goes on there. Then more recently, we've got the widespread use of containers. A single VM might host a bunch of containers. Have the roles of a DNS provider significantly changed with these increased layers of infrastructure on the same physical server?

[0:28:48.4] PS: That's a great question on a couple of layers. On one level, no. It just creates all the more need for you to use of third-party DNS provider so that you can have better control over naming and where you want things to go. There's sort of a flipside to that answer and it gets to something I didn't mention — I did mention about IPv4 and v6 and carving up address space. I talked about basic traffic management using Any Cast sort of first top, "Take me to the closest thing."

You can't do that without knowing IP address space. Some people talk about Geo DNS or being geo-aware. We have a number of different ways to do this in our DNS systems, but the internet and IP numbers aren't geo, they're just numbers. You can end up in the following scenario, "I am a company. I manage to get from Aron, some address space, and I work with my network providers to be able to announce servers in that address space and it's registered to my company, and my company is in Boston, Massachusetts. Yet, my servers may not be in Boston, Massachusetts at all."

Say, I had — A /24, I mention, is the smallest block that you can announce on the public Internet. Say, I had two /24s, so that's a /23 in sort of network notation. You might think that I've had those two blocks in Boston because that's what the records would say in the registration databases. In fact, I may have chosen to split one of those blocks to serve the West Coast and another block to serve the East Coast. Now, how do you know that one of those 512 addresses — Where is it on the internet? That's another underlying layer of our DNS systems is understanding the geo-location of an IP address so that we can do very fine-grained staring both of where the user is coming from or the resolver that they came through or where we're targeting you to get to. As I say the words, I'd wonder out loud, like that's a hard thing to comprehend if you don't know some of the moving parts.

Its core to this is that understanding how to carve up just those 4 billion v4 addresses and know where they are and know that they change. Here's a great example. Core internet transit bandwidth providers, like, say, a company like Level III. This is a classic case that we talk about with our internet intelligence product. They're a backbone provider in the internet and they have two adjacent addresses. You have those adjacent addresses might be physically at the end of a piece of fiber-optic, one in Miami, another one and Sao Paulo. You can't even assume that two adjacent internet numbers are anywhere geographically close to each other.

We have internal databases within our DNS database that helps us understand this and continuously track these types of a nuances as to where an IP is, because that helps us steer the traffic right where our customers policy wants the traffic to go to.

[0:32:23.4] JM: We talked about how DNS works with a simple website, like softwareengineeringdaily.com. It's more interesting to talk about what happens with DNS on a complex site with a lot of different things going on, like SoundCloud, or Etsy, or just these big companies, they have massive infrastructure. That infrastructure is always under a lot of load.

Give a description for why the DNS requirements of a company like SoundCloud might be different from the requirements of something like softwareengineeringdaily.com.

[0:33:01.0] PS: I think it's a combination of all of the things I've just been talking about, which is how many endpoints can you afford to stand up on the public internet for one. Are they going to be geographically dispersed? Are they going to be concentrated in a particular region? If they're concentrated in a region, are you going to have multiple sort of item potent endpoints that it doesn't matter which one you go, they can do the same amount of work.

Are you trying to serve static content, CDNs? They do all of the things we're describing as well. These are the advanced techniques. I guess I was sort of answering your last question with some of my partial answers earlier. You bring all of these together, geo-location, round Robining, load-balancing, short TTLs. Short TTLs allow you to be resilient under duress, can shift traffic to somewhere else when there's a failure and your downtime might be very minimal, if none, depending on how you might have given overlapping answers or you've limited the number of small number of users that may temporarily not be able to get to you.

The great thing with most browsers and applications is they'll automatically retry. If they couldn't get that connection, the TTL is short, it's going to go get another address. That's going to retry the connection and sort of, Bob's your uncle, you can move along and get to where you were trying to get to. These collection of techniques are core to what our services are offering and core to what keeps some of the biggest sites on the internet all the time.

[0:34:39.5] JM: You mentioned a term there; short TTL. Can you give more color on what that means?

[0:34:45.6] PS: A short TTL would say how long that record is valid for. If I'm doing an active load-balancing where I'm also —

[0:34:53.9] JM: A time to live.

[0:34:57.3] PS: Time to live. CDNs also have the similar thing which is like caching headers of how long the content can be cached in the browser or cached in a proxy that was between the originator of the content and the final recipient of it. Same idea with DNS. DNS sort of predates these notions. The shorter the TTL, the queries that will get asked over all of the people asking the questions spread across the internet. It gives you the ability to sort of heal system or heal your service and provide another place where you might steer users to.

Rather than just sort of round Robining and picking one of four dresses, you could also add an overlay of monitoring, and that's another thing that we do with our advanced DNS services, is we will actively monitor just like you might use a variety or monitoring services to see what your health of your endpoint is and can a user get to your website or get your API. That monitoring can get very sophisticated. Can they actually login? Can they perform a transaction, et cetera. You're probably using that to trigger an alerting system and trigger a pager or whatnot, and so somebody can get involved and mitigate the reason that the monitoring is firing of an alert.

In the case of monitoring that's embedded into our DNS service, we'll adjust answers based on health of target endpoints. You might say I've got four things to choose from. I want you, Dyn, to actively monitor those endpoints. If one goes down, take it out of the answer pool. Even though it could a round Robin between four, if we see that two are down, it's only going to round Robin between two, et cetera. It's a combination of — That's perhaps the most advanced technique that you could use or a policy that you could define against your answer set.

[0:37:05.7] JM: I want to zoom in on this TTL term a little bit more, because TTL implies that there is an entry somewhere and that entry is eventually going to expire once the TTL expires. TTL is a broad term in software engineering where you write an entry to some kind of record database or a cache or anything and the TTL specifies how long is this entry valid for. Once it's invalid, it expires and it's no longer going to be used and so maybe it gets refreshed somehow or it makes a query to get a new record with the new TTL.

When you say you're creating an entry somewhere with some sort TTL, where exactly are you referring to? Is this an entry in a CDN or is it an entry in the DNS provider's database somewhere? Give a little more color on that.

[0:38:02.3] PS: Okay. I did talk about both CDNs and then DNS records or CDN content and DNS answers having TTLs, because they both do. CDNs are also depending upon DNS TTLs to deliver you to their endpoint that's going to delivery you the contents. That's potentially a little confusing. In the end though, when I mentioned earlier, there's a recursive and there's an authority, we happen to operate both. Primarily, we're an authority.

Within our database, under the covers, we'll have something called — We'll have a master database. In our case, it is a database. The records that are in your zone, I'll just keep using example.com. I'll have a record for www.example.com, and www record would say it's in A record, it's got an address, and it's good to a TTL, and it's up to you, the configuror of that record to decide what those values are, and the important one is a TTL. If you set that to 12 hours and you've only got one of those and then your server goes down, no matter what, it's going to be 12 hours that that record is valid and your server will never be reachable for somebody who already obtained a cached version of that answer.

If I come on and I haven't been on the system and gone to that example, www.example.com, and it's the first time I'm trying to go, my browser cache will be empty. It will go get the answer. But if it's the only answer still, it's just going to take me to a dead server or a server that's not responding. The shorter the TTL, the more control, fine-grained control you have over ensuring that you can steer somebody's trying to get to you to work —

[0:40:01.7] JM: Right. If I'm example.com and my site is really picking up in a lot of popularity and I've started to add in all kinds a load balancers and I'm connected to Amazon Web Services, so a lot of my VMs are dying all the time and things are being restarted, I'm going to want a very short TTL on the entry that is associated with www.example.com.

[0:40:32.9] PS: Correct. The shorter the TTL, the more control you have over it, but you get more QPS. QPS doesn't cost a lot in general, so I would not advise people to worry about that. If they talk to us, we can talk to you about things to consider when you're doing that.

I also talked about the master where you're setting those records up. Our masters are never the thing that faces the internet. This is something that you'd asked me about earlier and I didn't answer at this next level that I'm going to go into. A DNS edge, for a provider like ourselves, is always a master slave edge and nobody in the public internet can get to our masters because they're the definitive source. As a change propagates into them, the master sends a special sort of fire and forget one-time message that — It's called a notify. It notifies it's slaves that it knows about that, "Hey, there's a change to this zone." Then the slave can either get a full or partial transfer, so that's a lot of full or partial replica. Maybe just delta is sufficient, just the one record might propagate to the slave, or in certain circumstances, that the entire zone might have to be propagated. It's the slaves that — We have many hundreds of slaves out there on the internet that those are the ones that the recursives actually talk to.

The DNS under the covers actually has a replication protocol designed into it and we use that DNS-based replication protocol to get the answers to slaves and our goal overtime is always to get slaves ever closer to customers or to cloud endpoints or to CDN infrastructures so that we can give high response in those scenarios.

There's a separate replication tier that's under the covers that's database replication and I think many of your listeners will be familiar with what you can do with MySQL or PostgreS or other databases and how they replicate internally. That would all be private replication traffic. In our case, it's private replica traffic as well.

[0:42:44.9] JM: We're getting into why a DNS provider is actually useful, because I can go to any number of sites that can offer me really cheap hosting and they're going to take care of mapping my domain name to an IP address. They're going to host my website particularly if it's a WordPress site on some container that's dedicated to hosting WordPress sites and I can get DNS just with. Why would I want a company like Dyn that's entirely devoted to doing DNS?

[0:43:26.4] PS: I think it's for all of the — Some people might say their edge case, but they're actually pretty common. Scenarios that we were just describing, you've got more than one endpoint and you don't want it to be randomly selected. You add two endpoints, one in Europe and one in Singapore. Do you want to send 50% of your North American users to Singapore to talk to your server? No.

Typically, what you're going to get with this sort of built-in free DNS is the very basic answering; single records; single values, or in the case of — You can have multiple values for an A record. Just pure round Robin; no waiting, no load-balancing, no geo-awareness, no monitoring. All of those things now become something that you need to consider. If you're making money from your service, which I hope everybody is, you need to keep that up. You need to ensure that you're delivering ads if it's a blog type of thing. You need to be able ensure that you're getting a customer all the way through check out if you're doing e-commerce, et cetera. That's where it becomes critical. We're dedicated to surviving and dealing with sort of all of the things that happen on the internet and ensuring that we can give the best answer possible for the questions that's being asked, which is; how do I get to this site?

[0:44:53.7] JM: If I'm an engineer at a site that uses a DNS provider, I I’m going to want to set policies for how my traffic is going to be routed in response to changing conditions. What are some common strategies for setting policies?

[0:45:12.2] PS: I’m glad you used the term policy as well because that's my preferred term, because it is policy. When you're configuring these types of services, you're really configuring policy and what are the criteria by which a particular policy will be applied. We will have policies for health, for performance, reachability, waiting. The policy around the TTL you can even think of as a policy. Those are the things that you start to get control over.

Also, notification policy, being notified that there was a problem with your server or that we detected that we needed to steer around a connection problem. The other thing I’d say is the following. I’ve given a talk publicly a couple of times that it wasn't so much about DNS, but DNS is part of this story.

I remember the day with one of these early clouds were I was in a different company and I thought that spending hundred thousand or a couple of hundred thousand dollars a month with a provider was an awful lot of money. At that point, when you spend $100,000 in hosting, you had somebody to talk to if there was a problem with your services. That day is sort of long gone with some of the clouds.

We had a customer of ours both of a DNS customer and a customer or our monitoring products whose data center or services are running in the Far East suddenly had super high latencies. Effectively, they were off the internet, and it was because of a cable cut. How do you figure out there's a cable cut, and who calls you? How do you know that you’re suddenly having a performance degradation and that your DNS steering needs to kick in get — It’s answering differently because of the threshold you provided on the monitoring. One could be like if you don't see an answer in less than 500 milliseconds on the monitor, send it to the host that's responding less than 500 milliseconds. That’s the type of policy that we can help our customers achieve.

This customer was spending over million dollars a month, and so I titled the talk *Four Ferraris and a Cable Cut,* because they were basically spending four Ferraris a month in hosting and got a cable cut, they were effectively off the internet. How do they know? Who do you call?

The cable cut isn't the cloud provider's problem. That something that's happened external to them. There's a collection of services and capabilities here that you're buying when you opt in to a dedicated DNS provider.

[SPONSOR MESSAGE]

[0:48:05.6] JM: For years, when I started building a new app, I would use MongoDB. Now, I use MongoDB Atlas. MongoDB Atlas is the easiest way to use MongoDB in the cloud. It's never been easier to hit the ground running. MongoDB Atlas is the only database as a service from the engineers who built MongoDB. The dashboard is simple and intuitive, but it provides all the functionality that you need. The customer service is staffed by people who can respond to your technical questions about Mongo.

With continuous back-up, VPC peering, monitoring, and security features, MongoDB Atlas gives you everything you need from MongoDB in an easy-to-use service. You could forget about needing to patch your Mongo instances and keep it up-to-date, because Atlas automatically updates its version. Check you mongodb.com/sedaily to get started with MongoDB Atlas and get $10 credit for free. Even if you're already running MongoDB in the cloud, Atlas makes migrating your deployment from another cloud service provider trivial with its live import feature.

Get started with a free three-node replica set, no credit card is required. As an inclusive offer for Software Engineering Daily listeners, use code “sedaily" for $10 credit when you're ready to scale up. Go to mongodb.com/sedaily to check it out. Thanks to MongoDB for being a repeat sponsor of Software Engineering Daily. It means a whole lot to us.

[INTERVIEW CONTINUED]

[0:50:00.8] JM: What’s the relationship between DNS and my monitoring infrastructure? Is there some way to leverage DNS when I'm trying to get better insights for how my infrastructure is performing monitoring-wise?

[0:50:17.1] PS: I’d say two ways. One; I’ve been saying — One thing I’ve been saying is that we provide advanced monitoring as part of some of our advanced DNS policy capabilities and we do that and we don't charge for that separately. It’s not like a lost leader almost. We need to provide that capability.

We also consume the very same third-party monitoring systems that our customers also depend upon to understand and run their dashboards, like how well is my service providing, or how many are page views am I getting, or am I seeing you 500 errors out of my web servers? We’re using those same monitoring services to monitor ourselves, so it’s kind of a meta-monitoring problem. We need to monitor how well we’re doing. We invariably use the same monitoring more than one monitoring system because our customers use more than one third-party monitoring system. We need to see ourselves from the way the customer sees ourselves or sees our service being provided.

I'd say you need to use both and to the level and extent that you can and always use monitoring no matter what, otherwise you are going to be flying blind. You will not know how things are going. You won't know that your host may be up but I might not be responding because the processes down. Two; it’s responding but it's responding like once a second instead of 10,000 times a second, et cetera. You need to know you when latency is being induced in your services. You need to know that you’re steering around the problems of that's the policy is configured. Third-party monitoring systems are another tool in your tool belt that you need.

[0:51:58.2] JM: Rolled is a DNS play in the event of a DDoS attack or another catastrophic event.

[0:52:05.9] PS: Ideally, we will isolate you from an aspect of it. It really depends on what — There’s not any single type of DDoS attack out there. Many people are aware that there's a whole lot of talk about the IoT and Internet of Things or Internet of Bad Things, or I sort of heard Internet of S, swear word things at a conference last week.

[0:52:38.3] JM: It's a great Twitter account, by the way. They've got some hilarious memes and stuff coming out of that Twitter account.

[0:52:45.2] PS: It depends. Marshaling for, literally, just a few dollars, you can buy a device that properly connected to the internet can generate a tremendous amount of traffic. If you've got thousands of those devices that have been compromised in one form or another, you have a weapon of mass destruction, or certainly a weapon of, "I can perform and send a lot of packets."

If you're sending packets towards DNS provider in the effort to try to — If you take us down and thereby creating a blast radius around us and all of our customers, okay great. We're under some form of elevated packets every single day and we're generally very good at shutting the traffic and dealing with it.

That botnet could also be targeting a specific single IP address, and once you do that, you're not going through DNS at all, and we're just sort of a witness to the same thing that others might be witnessing which could be congestion over a particular pipe and we don't know why. You've got to have those defenses in your system.

The way I would talk about that is defense in-depth, because we provide services to cloud providers and operate some of our services in cloud providers themselves. In our core data centers, we're always operating effectively a private cloud on the public internet. We defend in transit because we can and most people have no ability to even know who to work with to do that. If you're in a cloud provider and hosting a compute instance are you've spun up a container in a container service, you're dependent upon the cloud providers network engineering team to defend for you. That's one level of defenses within the transit itself, upstream from your routing edge.

Then, routers, your routing edge itself is a second tier of defense. Again, if you're running in cloud provider, your cloud provider is providing that capability for you because you don't have access to that. I've gone to a lot of dev ops and monitoring conferences over the past two years and one of the questions I kind of like to ask is, "Who knows what BGP is?" Invariably, I'll get just the tiny subset of the audience will even know that BGP is a protocol that actually makes the internet go.

Then some much larger percentage of the audience knows what DNS is. If you want to talk about Docker, or all of the latest things in the dev opts and in develop world, they'll know that and then they'll, of course, want to bicker about which is the best framework or whatever. Some of the fundamentals of what make it go, they don't know. If you're doing your own routing, you're going to have to know how to manage BGP and defend in your routing infrastructure.

The next layers is your hosts. If it's a big cloud provider and they don't care that your host receives a gigabit or 5 gigabits of traffic because you're paying to get the traffic delivered to you, but that might be traffic that you don't want or it is attacked traffic. It's a SYN attack. It's like a one type of an attack where you're trying to just fake the setup of a TCP session. You're going to have to defend that with how you configure your host. The kernel level configurations and/or — If you're in a container service, you're not even having much of a control over that.

Lastly, the fourth layer of defense is within your processes themselves. Are you seeing a level of traffic? Are you rate limiting? Are you delivering 420s or 429s, because somebody is calling your API too frequently. Each layer has its own set of obligations and capabilities if you want to keep that service up. For how we operate, we defend in each of those layers in a sort of tightly integrated chat ops style services and integrated with that, and you're going to have to build those same things into your container-based services or your host-based services.

[0:57:08.3] JM: As we draw to a close, we talked about the Internet of Things as an annoyance or threat vector but there's also lots of opportunity.

[END]