MONTREAL – How it Works: Understanding DNS
Tuesday, November 5, 2019 – 08:45 to 10:15 EDT
ICANN66 | Montréal, Canada

STEVE CONTE: All right, well, the beauty of having the first session in the morning is you get prime seating. So this is going to be an open discussion, it's going to be a presentation with an open discussion. So if you wish to come up to the table and use the mics, otherwise, raise your hand, we'll get a mic over to you guys. I'd like to introduce you to Paul Hoffman. He's Principal Technologist in our research arm of Office of the CTO and we do a lot with unique identifiers, DNS being one of them.

So we're going to go through a session on DNS here and then the 'How it works series' of presentations will continue in this room at 13:30 today; we're going to have a session on the RIRs, we'll have a session after that on RDAP, the replacement on WHOIS, and we'll have a session on Root Server Tutorial. So after this session, at 13:30 we'll be back here for these other sessions. So if you've got the time and the inclination, please come back and join us.

With that, I'm going to introduce to Paul, and please, join us at the tables. That's fine. Paul, thank you.

PAUL HOFFMAN: Good morning. I'm not intimidated by the small number of people here because that means that everyone else at this ICANN meeting

understands the fundamentals of DNS; you know, you'll be the last ones to learn it and since they all know it, then we're all level set. But thank you for being here, because what I just said was not true and it is really useful for us to have people throughout the ICANN community, even if they aren't technical people, which most aren't, understanding the fundamentals of DNS.

And the reason for that is a lot of the fundamentals actually have policy implications. I'm not going to go into the policy implications today, but one of the things we found is the more people sort of have a sense of, you know, what is this DNS thing that we are doing these higher-level policy things for, better understanding.

Come on in and feel free to sit at the table, so this will be sort of a discussion, feel free to interrupt me. I believe I will finish in time. I'm hoping to finish early so we can even have a more open discussion of it, but if I'm saying something on one of the slides that you don't get, you know, if you're at the table, just hit the mic or whatever.

I'm going to skip over some of the slides. I did not make the slide deck, my boss did, and I don't like it as much as what I would have made, and as we were talking yesterday going through, I'm like, why did you have that? He's like; 'Oh, yeah', so don't worry if I skip over slides. Describing the DNS, to people who are not, you know, day job technical, is actually really interesting because some of it seems really easy and we will have all of those parts of taking it from what I know of what a name is into the DNS and such; and some parts get really arcane.

Steve, why am I clicking and nothing's happening? Thank you. Okay. Nope. I think it's up to you. Oh, wait, oh, wait, it has a left and a right. Oh, I'm sorry, got it, hardware issue.

STEVE CONTE: How many engineers does it take to change a light bulb?

PAUL HOFFMAN: Yes. Right, exactly. So one of the important things to remember as we're going through this, especially as I'm like skipping over some slides and such, is the DNS is one of the oldest protocols on the internet. We have the internet itself, you know, the way that packets move around and then we have these things above it, the web and such like that. DNS is one of the oldest technologies, and in fact, the beginning of the session we'll start with the history.

Older technologies have funnier things in them, like if we were going to create a new DNS today, we would actually make the technology different and probably easier to understand. So if you're like, 'Oh, this seems a little bit weird, I understand some other technologies but I'm having a hard time with this'. Don't think, 'oh, it's because the DNS is inherently complicated', it was developed before lots of other internet technologies were.

So let's jump into the history. Before there was a DNS, before, you know, when we all just had a bunch of -- we, for some value of we if you were here in the '80s, had a bunch of computers -- you had, you

know, each computer has an address. So addresses are easy for machines to understand, but hard for people to understand.

Anyone who's here at an ICANN meeting knows that names are really really important to people; simple names are better than complex names, names that represent something are better than names that are just a bunch of letters strung together and such like that and everyone has known this all along, but because we started with just the simple numbers, it was like, 'Okay, we'll just do numbers' but the first attempt at names don't look like DNS names, there were no dots, it was just a single label, could be up to 24 letters, and these were almost always related to where the machine was at, UCLA computer science department, things like that.

So that was how the naming on the internet started, and there were some problems with that, because you know, you have a name and then you say, 'Okay, I'm going to look up the name, which is called resolving the name, and I'm going to get the address'. So how do you do that? Well, someone said, 'Well, let's put it in a file, just like many of us do, we'll put all of our friends' names and phone numbers in a file'. Then that, you know, that's the list, you go down, maybe you alphabetize it or whatever.

That was the big that was the beginning of the origin of thinking about, how is this service because DNS really is a service, how is this going to work? So that was literally a file.

Remember, in the 80s, we didn't have all the HTML stuff like that, it was like a plain old text file and somebody had to make that file work. You didn't want everyone making their own file, you wanted the file kept centrally, so that was kept at SRI, which was one of the organizations that was pretty much fundamental for making the internet work, but again, this is way before the whole concept of all the naming is worth familiar with, it was just 'How do I reach somebody else's computer by address without having to remember the address just to know their name?'

So somebody kept track of it, it was updated once a week, sent out to everybody, all's good. Except sometimes, you know, like as the internet was getting popular, we're talking now hundreds of hosts not billions, but as the internet was getting a little bit more popular, we had things such as -- well, the first one here is actually the most interesting one. What if there are two computers now at the Stanford Computer Science Department, and you already named one computer, Stanford Computer Science Department Computer? What do you name the second one?

Names have to be unique, if you have two friends named Paul Hoffman, on your phone list, you're going to have to remember 'Oh, well, maybe it's this one, who's in this area code', you know, like name contention has always been a problem, but it's usually a problem for the second person, not for the first person, the first person got the good name, but much more importantly, it's a real problem for somebody who's looking it up.

So there's the question of name ownership, 'Oh, I've got the first name', but really, that's a problem for everybody else and now that there are hundreds of people using this internet, every time there's a name contention, it's a problem, so two ways you might do it, as you might tell the first person; 'No, you're gonna have to change your name to Stanford Computer science One', because now we have Stanford Computer Science Two or you can just tell the second one, pick something else. Well, it's not as representative, so you hear that even though this is in the '80s, they had the same problems that we're dealing with in ICANN today?

On the previous slide, I said that you know, like somebody made the list and everyone grabbed you know, it was updated once a week. Well, it's not that they updated it once a week they were updating and everyone knew to grab it once week. So there was different versions of the list at different times. So that doesn't really work, cause someone said; 'Oh, go here. It's like, well, I don't have that name on my list. Oh, well get that a new list'.

So how often do you get the new list so that you have the coolest, you know, list and such like that? In the early days, bandwidth was really important. So even though this wasn't a terribly big file, pulling it down cost money and time because this, you know, running bits over the network costs time. So, if you were like, 'I always need to have the latest list'.

Well, you were spending a lot of your network budget, just getting the list of names and addresses. That doesn't seem like a really good use,

and then the last bullet is sort of funny because here we are at ICANN, who helps centrally locate things, but having a single list of all of the names centrally managed isn't really good. What happens to the person who's in charge of the list wants to go on vacation for a week? Does nothing get updated? Does that person train somebody else for the week to do it?

Well, that person isn't going to know all of the same stuff and, and may in fact, say things like; 'Oh, you can have the same name twice', which, you know, you can't really do. That kind of thing everyone was like; 'Okay, we're going to have to deal with this' because what they saw was the internet was growing all the time and again, I'm talking growing like 100 to 120, that's the 20% increase in the internet this year, you know, for us, that would be this room, you know, so as they were seeing this were like; 'Okay, how do we fix this?'

So, in the early '80s, they realized, okay, naming is super important, if we get this wrong, we are going to limit the way that the internet will grow, it wasn't even the internet then but our favorite network that we're all you know, really having a good time on. We're going to limit the way it's going to grow. What do we do, what do we do?

So this slide grossly summarizes years of informal discussions among Dweebs about what is the best way to handle naming for a system that we believe is going to grow and be important because by the early '80s, especially by the mid-80s, people understood that the ARPANET was going to be important, by 1985, there were probably small tens of thousands of people, who had accounts on the ARPANET, like I did.

I was going through a commercial service but I didn't understand anything about naming but I knew 'Oh, this is cool, this is going to keep getting bigger' and so the people who were thinking about the Domain Name System weren't thinking about, 'Let's do a cool naming thing' it was how do we make it so that this will grow so that people can be using this not just computers with numbers?

So what came out and again, we just skipped over a whole bunch of late-night discussions and arguments and I know how names should work, mostly among people who were computer science, folks but what came out was the DNS. It's a distributed database. When I was talking about host.txt, which was a single file, one of the most important things that people don't really get is the fact that the DNS is a distributed database. If you distribute out the database, then somebody can go on vacation for a week, and the database still works. In fact, many people can go on vacation and it's still working just fine.

The way that you find out what address is associated with a name is a thing called a resolver, sends queries to the database and says; 'Hey, what is this?' What that resolver sends a query to is called a Name Server. So you have two parties here now, the thing asking, which is a resolver and a name server, which is the answer.

This seems very simple, you know, 'Oh well, yeah, of course, you're going to go ask', this was actually a radical change from the way that any network was working at the time and so the people who invented the DNS, and I say invented sort of loosely because, you know, people

were trying different things at different times, but what became the DNS early in the '80s, was a radical departure.

It was also very different than the way that the IT was doing their naming for their network. They had a very different structure much more heavyweight and what the people who was doing this in the ARPANET realized was, these are names people really, really care about names. Let's make this as lightweight as possible, let's make this as flexible as possible and let's distributed as far as we can and the reason for that is, if people feel comfortable with the name of the thing they are using, or if they have their own, you know, host the name that they have, they're going to like it more, they're going to use it more.

So the fact that the internet had naming, what you know, through the DNS, was one of the main reasons it grew beyond the core geek crowd, was that other people could use it, they could understand what they were doing.

So two other things that came into play very early on, which I'll talk about later in the presentation, one is caching, which means that, remember, I said that people were spending a lot of money just getting the list, it's nice if you can remember things because then you don't have to ask again, so caching was built in, that was a very geeky thing at the time, turns out to be the savior for the DNS and the fact that you could have many name servers, many of the answers with the same answer in different places.

So for example, if you had if you didn't have replication and I'm the name server for ICANN, and I'm having a problem, you know, like I'm down, no one would be able to ask any questions until I came up again, but if there were three copies of me, then you could say, 'Oh, that one seems to be down, I'll go ask here'.

So by putting both caching and replication in, what happened was, they made it less like a database, where you have to go to the database system as such, and more like a schmoozed out, you know, group of things, that still got you the answer, you wanted and again, that made it a lot friendlier and made things go.

So let's get a little bit into the technical part. I'm going to cover these bits later but this is sort of, in fact, many of you who have taken sort of DNS 101, you know, courses before have seen a picture like this. This is just the players, we're going to actually walk through what this really means in a bit, but the players in this thing are, is this pointer two or not? Steve? It is cool, okay? I just blinded myself.

So this is you, this is the person at the end, this is your phone or your browser. In your computer, these are computers, those of us in '85 would not believe this, but this is your computer. It has a program called a stub resolver, and remember I said Resolvers are the things that ask questions. So when you type in a domain name into your phone, you cause this software to ask the question of, 'what is the address associated with this name that this person's just typed in?'

That goes to a recursive resolver and again, don't worry, I'm going to be covering these things and we'll have more examples later. So first player is you and you have a stub resolve or a recursive resolver is a system that you're associated with, that will actually do all the queries for you. You say your sub resolver says to the cursor resolver; 'hey, can you find this out for me?'

The recursive resolver then talks to authoritative name servers, notice that this is a plural we have lots of authoritative name servers. It does things, we'll talk about in a few minutes, that essentially lets it get the answer, so you're not getting the answer, you're just asked the question to this 'better answer getter' it asks all these, then it says; 'Okay, I've got the answer, now, we'll come back there'.

So these are the major players the stub resolver, a recursive resolver, authoritative name server. It's all software, I mean, these are all of course pictures of computers and such. It's really software. These look like big computers, they actually can be tiny Raspberry Pi's but you know, the other sort of cool thing about the DNS is that it was built very lightweight, so you don't need giant computers to be running any of this. So those were the players, let's go back and do, sort of the most important part is, what is a name?

You know, my name is Paul Hoffman. My name is also Paul Eric Hoffman. Some people call me Mr. Paul Hawk, you know, like, even though these are all the same names, different ways of looking at that name. In the DNS, we have this thing called the Name Space and what a Name Space is, is a set of names where every name is unique. You

never have two people with the exact same identifier and the reason for that is, if you did have two people with the exact same identifier, and someone asked, well, what's the address, they wouldn't know which of the two answers to get.

So a very, very important part of DNS is the fact that in the Name Space, there's one name, you know like each name is unique. So the Name Space is this hierarchy and we'll go through this in the next couple of slides and so each node in the namespace has a label so, this is the most confusing label, so we're going to skip over it, but these are labels like that you are familiar with, they're probably people here who are from the gTLD community, you know.com there's people here, certainly from the ccTLD community, you know, UK, I guess this is a new gTLD community kind of thing.

So there is these nodes in this tree, and we'll go into this a little bit later but the root of the tree actually has a name that we mostly don't see, but as you know, ICANN, as you probably heard ICANN manages the root. The root of the tree is not everything down there.

When you go and get a name called www.example.com, you don't come to ICANN for that, you only come to ICANN if you want to have a name in the top level of the tree, we can tell you where to go for the other ones. So the labels in a name, they have to be letters, digits and hyphens, they can only be 63 characters, most of them are much shorter because again, these names are supposed to be used by humans; and in these examples which are ASCII, we've totally ignored

IDNs in this presentation, which is one of the reasons I'm not so happy with it, because I'm a big IDN fan.

So these are all ASCII, so each label comes under; so under com, you have the label of example, under that you have www. So at each level, the names can be repeated as long as there's no repetition in a particular level under another one. So just in the same way that Paul Eric Hoffman is different than Paul Eric Smith. I could also be Paul Eric Hoffman Smith. I mean, you can still see how those are different.

This part of the tree is completely independent of this part of the tree. You have full independence, one of the things, one of the reasons they did that is even very earlier on, was the internet was already quite International. So you couldn't have the Americans telling the Brits; 'Here's how you're going to name things under your part of the tree', nor with the Brits be able to tell the Americans nor could Stanford tell UCLA and such like that.

So at each level in the tree, it's completely independent and the way you do that is you have these labels, and each label indicates another layer of the tree. So a domain name is the combination going all the way up from the bottom of the tree to the top so www, example, com, root.

Notice that when we talk about domain names, we never say www example com root. The root is completely implied here and the reason is the root actually doesn't have a name. It doesn't say ROOT, it just says that's it, that's the root of the tree. So again, names are

done from the bottom up. So mail.example.com.root, www.bar.com.root, that is a full name; that is what's called a fully qualified domain name.

And notice here that there is also www.food.com, there could easily -- oh look, I'm sorry, I forgot he did actually put an IDN in there. You could also have, in a different tree, you could also have something called www.food.uk. That's a completely different name than www.food.com because it's in a different part of the tree. There's no problem with having similar looking names that have one label different. They're still completely unique, and as I said before, this is called a fully qualified domain name.

So before I go on to that, how are we so far? Just doing okay. You all feel like you still have names? I haven't lost you on this? Okay. This was something that the folks, going back into the history, this is something that the folks understood of what they wanted to happen. All of this makes perfect sense if you have taken a bunch of math classes, which I haven't, by the way, but they did you know, computer science in '70's and '80's mostly was happening in math departments and such like that. This is graph theory, things like that.

They understood this and they were able in their heads to say people are going to like this. People understand that 100 Main Street, in Evanston, Illinois is not at all the same places 100 Main Street in New York, New York and they also understood that 100 Main Street, Evanston, Illinois, in the United States, if there was an Evanston in Germany, no one would have the problem of thinking; 'Oh, this

doesn't work'. So by creating names like this, and then figuring out how to actually be able to run the protocol, people would be able to do their own thing quite well.

So let's go into how that own thing works. We call it the Domain Name System, well, what is a domain? A domain just as an English word means, sort of a single collection of things, you know, that you can identify. So in the DNS, a domain is everything that is underneath a particular part of the graph. So the com domain includes www.bar.com. it includes www.example.com.

The blue line here shows the domain, shows here the com domain, which is again, completely separate from the UK domain. This also shows the www bar is also its own domain, that is completely separate from www foo, okay? So a domain is everything below a particular label in the tree, and that labels called the Apex. The reason why we do that, the reason why it's not just a free for all at the bottom, is because you want layers of control.

If you have a company called, you know, Jones company and you're going to want to have names under that, you don't care about just www you want mail, maybe you want to have department names, each having their own domain name underneath it, somebody there has to be able to administer that and so they will be, you know, the administration is always done at the apex of a domain.

And that means two things, one is that administrator has complete control over everything underneath them; even if they cede partial

control to the people underneath, they can remove that control so they can control everything or they can delegate down and so delegation, the idea of saying well, I own this domain but I delegate administration of a part of it, underneath me to you, is fundamental.

So VeriSign who is -- you know, the root has delegated com to VeriSign. VeriSign delegates control of each Name Space underneath it, so example, .com, VeriSign delegates control of example, so if whoever owns example.com wants to create www, they don't have to ask VeriSign anything, because VeriSign has delegated control to them. The flip side of that is VeriSign can take that control away at any time.

Now, this is ICANN, we all have lots of contracts, VeriSign has agreed not to do that but in the idea of the Name Space, someone at the top can delegate away and can un-delegate and that's really, really useful because maybe you don't want to delegate each sub name to somebody, maybe there aren't people who know how to run it in your organization. So you want the top administrator for the organization to be running all of them and then maybe after a while, your organization gets big enough, where you want to be able to delegate off, this gives you the advantage of control and the ability to give away partial control.

So, when you delegate a zone, the entity who is delegating is called the Parent, and the one that is delegated is the Child. So now we've gone from trees and roots to parents and children. I know it gets

confusing, but it's not like these people were always terribly consistent about this.

So, here is a picture of the whole Name Space again and then let me go into where you can see the administrative boundaries. So, this is the administrative boundary of the root. This is what ICANN true IANA has management over. COM is its own administrative boundary because each time com by agreement, so com has an agreement with the root and the agreement, is that it will delegate to other people, sounds good.

This is a Bound Administrative Boundary, this is an Administrative Boundary, and this isn't an Administrative Boundaries because it doesn't have any subdomains under it yet. Once it does, you will see other Administrative Boundaries. So we've gone from the idea of there's this whole tree to there are a bunch of islands within the tree, that are delegated down and again they can be undelegated. Feeling okay? Good.

You can see how the delegation happens, delegation always is pointing down. You never get to delegate up, if you're a child, you never get to tell your Parent what to do and you never get to tell your Parent killed by my sibling. It's always up to the parent about that, it's fortunate for some of us.

So now let's talk about how that all happens, we're starting to get out of the picture range now and we're going to get into the sort of mechanics of how all this happens. Name Servers, as I said earlier, are

the things that answer queries and so a Name Server for a zone is authoritative for it, just as we saw here, that you have the Name Server for this zone is authoritative for it, the name server for this zone is not authoritative for com, it is authoritative for saying; 'This is where you will find com, but it doesn't answer for com', which now sounds good for the Parent/child thing. The kids get to speak for themselves or the grandkids get to speak for themselves.

So an Authoritative Name Servers, it gets to give definitive answers, and there are two kinds of definitive answers. One is, here's the answer and the other is, 'I've already delegated that off somewhere else, so I'm not going to answer for you but I'm going to tell you who can'. Basically that's it, that's the entire DNS answer system is 'I'm authoritative, I know or I'm not, but I am authoritative to tell you who is' and so that'll works well but, again, if that server went down, you wouldn't want to say; 'Oh, there's no more answers until the server comes up'.

So we have redundancy. Redundancy gives you Authoritative Name Servers that all give the exact same answer. So redundancy is not, 'I'm going to spread out my load of some answers are going to go here and some answers go here'. All Authoritative Name Servers for a zone will always give the same answer and that's nice because then the person who's asking questions doesn't have to go; 'Well, which is the best one?'

I've got a list of three Authoritative Name Servers, you know, to go to, I'll just go to any of them or if I've been looking at my previous queries,

it seems like this one's faster for me, I'll go to the fastest one. I'm going to skip over this one, this was the first one I was going to skip over this is a little bit nerdy.

So far, we've been talking about queries that always come back with answers that are addresses, remember when I started, for those who came in late, I started off by saying, this is all about trying to have a name that gives you the answer, where's the address of where I'm going? So that was a gross simple simplification, because even in the early days, the folks realized, if you have a name, there's a whole bunch of information that might be interesting.

So you know me, I'm Paul Hoffman, and you might only care what's my gender? Good luck with that but somebody else in the room might actually care, what's my favorite color? Or you might care what my gender is, and what's my favorite color? So for every given name, there's a lot of information, there might be one piece of information that is the most commonly asked, which is what's my favorite color but that doesn't mean that my name is only associated with that.

So again, when they invented the DNS, they realized there's going to be the common query, which is what the address is, but why not allow each name to have lots of information associated with it. So all of that information are stored in something called resource records and a resource record says, if a person asks about this type of information, here's the answer. If the person asks for the domain www.example.com, what is the address? Here's the answer, but if

they ask for the domain www.example.com, who's its administrator? Here's the answer, lots of things like that.

So I'm going to give a bunch of examples here and I'm going to skip over that one because that was also not necessary, as is this one, so here are some common kinds of things that can be associated with a domain name.

So the first two are the ones that you hear about all the time, the addresses, okay, and there are two kinds of addresses hopefully, I'm not telling anyone new here that IPv4 and IPv6 are different, good. Don't have to know what they are, you just have to know that they're different and because they're different, they are kept in different resource records. So when you make a query to www.example.com, you have to actually say what kind of data you want back. You cannot say, show both of the addresses, you actually have to ask twice and you can say, what's the IPv4 address? What's the IPv6 address?

Those are the ones that we're mostly familiar with, but here are a bunch of other things that were defined very early on and I actually have examples of these on the following slides of other kinds of things that again, associated by name, any name can have all sorts of resource records and the interesting thing is, it's unlimited. These are standard types of resource records that have been standardized by the IETF, Internet Engineering Task Force, the people who are sort of the protocol masters here, but you could if for something that you're working on, if you want to associate certain kinds of data with names,

you can make up your own Association, the ability is up to have like 65,000 different types.

So don't feel like, that what I'm giving here in the next couple of slides is the entire list. The types of data are being added too slowly, but anyone can do this and that's again, one of the wonderful things that they figured out early on in the DNS is once you've got a name that people are excited about 5 and 10, and 20 years later, they're going to want to add other kinds of information.

Okay, so let me go through some of these because I actually have better examples. So as of now, which was actually two years ago, there were 84 types of these and again, you can have 10s of thousands. This URL tells you how to see the ones that are currently assigned but a whole bunch of these right here actually say, here are numbers that you can use, you know, types that you can use on your own, you don't even have to ask permission.

Now, other people might use the same one, so there would be contention but if you want to experiment you can as well, I'm not advocating that you do. I'm saying this is how they figured out how to make names useful into the future because the DNS is already 30 years old. Lots of things that were created 30 years ago just aren't really that useful anymore. The DNS was well designed from the beginning for this.

So this is actually an example. This is what you would see if you went to the URL on the previous page and you can see here, these are the

types that we're about to talk about. All of this is kept by IANA, so for those of you who are, you know, at the ICANN meeting today with some concerned about IANA and its management, this is one of the things you know, ICANN sort of has three remits domain names, IP address and technical identifiers. The types of information in the DNS are actually technical identifiers that are managed by IANA.

If you go wandering around the INANA website, you will see lots of registry like this but this is the one that we're talking about now and it's one of the literally thousands of registries that IANA maintains, this one's really important to me because I care about the DNS, many of them are important to other people but if you are the kind of person who would say I have a new kind of information that I want associated with the domain name, and you went through the process of getting the acceptance such yours would appear here in the registry as well.

So let's go through some of those types. First is address records, these are the most common and again, here's an example example.com has a, sorry, I skipped over this, each data type has a sort of a pneumonic name and in the case of IPv4 addresses, it's A, which stood for address because there was only one kind of address at the time, so A seemed good. So, this in a zone you would see if you asked for the A type, for example.com, you would get an answer back like that.

Then the other kind of, you know, later on, another kind of address came up so they had to give it a different kind of name which is Quad A, because it's four times as long as A, nerds don't do really well on the jokes here, you would get this address, so these are these two again,

these are types of data associated with domain names. These are the ones that most people think of.

There are two other types, are NS and SOA. NS stands for Name Server and SOA's, don't worry what it stands for, basically, it's a serial number. If you think of domain names types as having sort of two flavors. One is types that help domain name exist and the other is the real data you want. NS and SOA are like the shelves in a warehouse. So they are the ones that help the rest of the data exist and then you would put A and Quad A and other things on those shelves. I think yes, okay. So let's go to NS, which is the next biggest one.

Remember earlier, I said that names are delegated, the way you delegate down is you create an NS record, a Name Server record for the thing below you and what that says is if someone comes and asks me about something below me, so I'm not authoritative for it, I've already delegate it NS is where have I delegated it to? What is the Name Server that I'm going to answer back and say; 'I'm not answering for you, that name server is answering for you'.

So in this example, the Parent would, for example.com would say; 'Here are the name servers for example.com. This is the kind of answer you would get if you asked the Com Name Server for example.com; com would say; 'don't ask me, ask this Name Server over here'. So it's a way of referring off and delegating if you get an answer that has an NS's in it, it says; 'Okay, not me go ask them'. The answers are always at a layer or two below where you are. So the left-

hand side of this is, that the name that you asked for and then the right-hand side is the name of the Name Server.

Now, one funny thing here with NS's is you actually give another name, you don't give the actual address of it. That was a very questionable decision, but we're stuck with it, would have been easier if it was addresses, but again, when you they were developing the Domain Name System, they were really into names, so they said names of names, they didn't really worry about loops and such like that maybe they should have. So again, NS Records, mark the delegations, if here is when you ask dot, 'tell me about com'. This is the answer we're going to give and notice that these are all answers where it says com has a name server here such like that. Notice that these addresses actually are not in com. So you can delegate down and say; 'But go ask over there'. The DNS is very, very flexible this way and one of the reasons you'd want to do this is in fact, you wouldn't want to have all of your Name Servers in one zone, because if that one's down, then you know the queries going to stop. So a lot of flexibility here.

So again, an NS record says; 'I am actually delegating down'. I'm going to skip over this one, that one's super confusing. Actually, so the slide I just skipped over is not only confusing for folks in this room, you could get 100 techs, you know, domain name techies together, and about half of them will have forgotten what was on that slide. So don't feel bad. It's something that we always ask each other.

So now, again, com delegates down as well, you can have as many levels of delegation as you want and when you delegate, you can also give what is called Good Notice that there are answers here. So if in the delegation, I can say, example.com, here's it's NS Records I've delegated down here. You can also give hints. This is not definitive, because I'm not supposed to be gaming on answers, but I'm going to give you a hint of where to look so that you don't have to do another query, makes things go a little bit faster.

The other one I told you about the SOA which again, the only part of the SOA we're going to look at us here's it's a serial number. A zone has information in it about everything in it. So in a database, you always wonder, well, how up to date is this database, the way that the DNS folks decided to make their database work is every time you change any information in a zone about the information underneath, you bump the number of the serial number, so that you can tell at any given moment, I've got a complete set and such like that.

So this other stuff that you can ignore, but this is a serial number and the next time anyone makes any change in the zone, this serial number would go up, doesn't have to go up by 1 it will just go up. So now let's get into some of the other things that people thought of early Actually, you know what, I'm going to go through a couple of other ones. The first other kind of data that appeared in the zones other than addresses were these things for mail routing.

So imagine you're trying to send mail to somebody, and you know, you have their email address, you know, Paul@proper.com or

whatever. The way that this happened before the DNS was you would look up the address of proper.com and there had better damn well be a mail receiver there. Early on, people realize; oh, you know, for something like stanford.edu you couldn't always assume that there was one host@stanford.edu that was able to accept mail. Mail takes a lot of processing and such like that you might want to have another computer.

So instead of saying every host that has an address has to be able to receive mail. They said; 'Let's allow there to be special male hosts that did nothing other than mail'. So they made they did the MX record, the mail exchange record to decouple the mail server from the address. So when you send mail to me at gmail.com, you're not actually sending to the one host whose address is @gmail.com what you get is what we have for these records, notice the type is MX, just as you saw as there before, notice that there are two addresses here with different numbers.

That's a priority number, what that says is try to send at the lowest number, but if that doesn't work, you can also send mail to another one. So this allowed, all of a sudden, the mail system to be much more flexible, by looking in the DNS, you could get this information that said; 'Here's how to send mail, but here's how to send it to many places'. And you could in fact, have two different records with a 10 in them. You can say; 'I don't care which of these two you start with, start with there'.

So this made everything much easier for sending mail and now notice how this is really different than the A and the Quad-A record, the A in the Quad-A record was, you wanted this address, here's the address. Now we're giving you a list of choices. Completely radical idea, very, very different than anything anyone was doing at the time and networking but what it allowed to happen is that mail service all of a sudden became much more reliable and much more available.

If you set up a computer that had you know a name, you didn't have to put a mail server on it, it could be specialized for something else. You could have all of that mail, go over to this, you know, one organizational mail server. For those of you who have been around for a while, you've noticed that email is still sort of important after all these decades. This from the DNS not from the mail service itself from the DNS is the reason we all get to communicate as well as we do over email, was that one thing that they thought of in the '80s.

I'm going to go over this one briefly, if 30% of the people in the room get this, that's great, I'm not expecting to be on it. So, so far we've been saying, 'Here, let me just go back that this name is associated with this address and so the lookup is always what is the address, for example.com?' How many people have been thinking; 'Well, how do you look up the name that's associated with the address?' typical things people like are never happy enough with going in one direction.

So people said; 'Well, everyone's going to ask this, let's come up with a way of doing this' and so what they have is called a Pointer Record so that you can have and I'll show on the next slide, a little bit of a better

example, a way to look up and an IP address and find out if there's a name associated, there might not be a name associated with it.

So name to IP is Forward Mapping, that's what we're always used to, this is called the Reverse Mapping Tree, excuse me, let me silence this guy, it's making me think that people are communicating with me. So it's really, really, really hard to do address to name mapping and so the way they came up with, like I say, I'm not expecting more than 30% of you to get this right.

If example.com has an address of 192.027 as a v4 address the reverse lookup is 7.2.0.192.in-addr.arpa; you take the address, and you reverse it, and you tack on in-addr.arpa, you have to have someplace to tack it on, and that happened to be -- arpa was a domain name a lot of people could be using, an in-addr means internet address. But the reason why you reverse it is 192, in fact, is sort of the highest part of the tree in the address. So you want to put that closest to the root of the DNS. I'm done with this slide, don't worry about it.

Another thing though, that you can find in the DNS, another use of the DNS by having additional record types that are interesting. I thought the mail examples more interesting, so here are some other things that also were defined, txt records, you can put arbitrary text about your hostname. It's not clear why people would look it up because it's sort of arbitrary. You could be saying anything you want, but you can actually have a text description of the hostname.

You can actually map domain names to URLs. There are record types that allow you to do that. If you use DNSSEC authentication for names, you can actually associate a hostname with, for example, a certificate or certificate authority as a way of saying, for example, if you're going to do a web access to this domain name here is the certificate for it, you don't have to necessarily wait for it in the TLS handshake, and that record is called TLSA.

I did that one, and then we get into the obscure one, so we won't really go into that. So this one was sort of funny. What is the location of this address? There's a weird thought, but you can actually look up on some of them or the geographic position, you can actually get the latitude and longitude. Turns out, that made no sense at all, but they still exist.

Okay, so now let's combine all of that, all of these records, when you say; 'well, what does the example.com zone look like?' It has an SOA record, which has a serial number, and again, that's sort of a shelf description of this serial number. This is the serial number for this set of information at this moment, if this information changes, the serial number will become higher.

There are some NS records, which says that if you ask example.com, if you ask for information about example.com here is where you're going to be delegated to. Name Server is delegation down. You can still also have addresses even though you're delegating down that machine has addresses, so here are some addresses. You want to send mail to example.com, here is how to send mail to anyone at

example.com. Ignore that one and then again, this is additional, it's called glue information, it's sort of voluntary.

It's so that if you were about to go look up one of these names, I'll just tell them to you now, not authoritatively, I'm just giving you a hint, if you really cared, you're going to go ask for authoritative information, but if you don't care so much, here's a hint. So this is what is called a zone file. This is all of the information about a zone. The Zone file for .com is massive, it's got a quadrillion entries in it.

Most of them are just -- I mean, there's one SOA because there's always one SOA record with the serial number for his own -- most of them are just NS Records, a lot of them are also DNSSEC, special information for the zones, but even then, because there are so many records in .com it just goes on and on and on. Typical for a small, you know, domain, it would be about this big, actually even smaller, because you usually only have like two or three name server records.

Okay, so now this is the data that you can get, this is what we got, so let me take a moment here. Yes, let me take a moment here. Questions now, as far as I can tell, I'm going to have time for questions at the end. Okay, forging forward. How do you look this stuff up? I've just told you what the stuff is. How do you look it up?

A stub resolver, as I said before, is the thing that on your computer that sort of starts the query going. When a stub resolver sends out a query, it actually has three parameters, the name that it's asking for, the type that is asking for, and the thing that we're going to ignore,

which is called the class. This was the computer scientist adding another level of 'Ooh, well, what if we want to expand this in the future' turned out no one ever expanded. There's basically just one class, but at the time, they were like, 'Ooh, people might want to try different ones'. It got too confusing, so no one used it.

A stub resolver sends recursive queries saying 'I need a complete answer. I'm dumb. I only know how to ask simple questions and I want a complete answer', but when it sends a query to a recursive, remember, we showed the stub was on the left and the recursive operator was in the middle. That recursive operator can send iterative queries saying, 'I'm going to ask you something but if you don't know the whole answer, tell me where I can find it'.

Your stub doesn't say that your stubs, it says; 'Give me the whole answer or tell me to go away' but the recursive can ask the authoritative; 'I want this information from you but if you don't know it, but you know where I can get it, tell me and then I will keep going' and if you think about it, well, who do they ask first? It's the root zone, the part of ICANN that everyone loves to love.

The first thing that a recursive resolver would do, if it doesn't already have the information, is it would ask the roots servers. 'How do I find dot com?', 'Or here's a name, my friend the stub asked me for www.example.com so I'm going to ask the root'. What's the answer for www.example.com? The root doesn't know that but the root knows how to tell you where to go find com at least. Okay?

So first question is well; 'How do you know how to ask the root?' so they have to be configured and basically a root hints file like, that every resolver has it when you start-up and you don't know anything and you don't know how to ask anything, where do you go? Is this root hints file. Very, very important because if the root hints, if you give a different root hints file, they're going to start at different places. So this is completely agreed to by everybody, this is the addresses of the root servers, and I'll talk about the root servers in a second. So this is what that file looks like.

Notice that there are 13 entries and they all have names a.rootservers.net. The really important part is right here, these are the addresses, okay? At some other point in time, this file might be bigger, because there might be more root server operators, this file might be smaller because there might be fewer roots or operators and this file might be different because these addresses sometimes change. Not very often, fortunately, but they do sometimes change. So it's always important for the recursive operator to have the current root hints file, so they know where to go.

Yeah, so that top bullet is probably one of the biggest understatements ever made, was, 'Boy, is this complicated!' This is not a DNS 101 slide, but this is an ICANN 101 slide. The root zone is actually not controlled by one entity, it's controlled by two, it's controlled by the IANA functions operator for filling in 'what are the answers', but VeriSign is the maintainer, you may ask why? If you do want to know why, there's a great document that RSSAC, the Root

Server System Advisory Committee has put out called RSSAC's 023, better known as the history of the root servers.

Dozens of pages, arcane stuff, sort of interesting to look at, it's being revised now, but go ahead and read it, if you want to know why are there two and things like that? So, there were 13 root server operators is actually there are only 12 organizations because one organization VeriSign runs two of them. So here are the servers and the operators.

Now, again, notice none of this has anything to do with the DNS, as in the protocol, I'm going to get off this slide soon, but the thing to know is you got to start somewhere. If you're a recursive resolver to know who to ask. You're going to ask one of these folks and it doesn't matter which one you ask because they all are authoritative for the root zone. So they all have exactly the same information.

If you ask, you know, USC, and you ask VeriSign or RIPE, you will get exactly the same answer. It's super important to understand that when you have multiple authoritative servers for one zone, even the root zone, they have to answer exactly the same or else you're going to get really confused really quickly and you're going to have preferred ones and such like that. So all 13 of these root server operators, they'll give you exactly the same answer.

Why do you pick one versus the other? Well, you might have noticed you get answers faster from one of them or you might say, 'Heck, I don't care, I'm getting fast answers from all and let's just keep asking around'.

This is a cool site, because it's got a map that's interactive on it, shows you where all the root servers are located, approximately, it doesn't actually show you street addresses, but it shows you, for example, that there are 13 root servers about there in the world and 39 there, whatever but you can zoom in and things like that and what this slide doesn't show is at the bottom here is there's a list of all of the root server operators, and you can find out where they have all of their servers.

Some of the root server operators have literally hundreds of identical copies. So the root server system itself is literally over 1000 servers, with identical information. Operated by 12 organizations with 13, sort of, identities just to make it a little bit more confusing but all of those will give you exactly the same answer and that's where it's different. So, Steve, when is the root server session this afternoon cause I'm going to skip over slide?

STEVE CONTE: Root server session is this afternoon, I believe it's at five o'clock on, I'll check my notes and recap before the end of the session.

PAUL HOFFMAN: If this looks interesting to you, I'm skipping over this, but we have a whole session that's actually given by the root server operators on this. Okay, so oh, good. We're doing well on time, there will be time for questions at the end. But now I'm into the funny pictures again. So, I just told you, this is how all the questions go, let's walk through

this one by one to see how all this goes and Steve, I'm at 10:15 is when I am? Great. Okay, we're doing great again, this is you, oops, no, I'm sorry. I'm not good with buttons, Steve. You should give me ones with like bigger buttons. I'm a software guy. This is your phone or your computer, this is your computer, this is your computer, and this could be your computer. Your computer wants your browser, wants you to go to www.example.com. So that's a name, it's not an address, you need to find the address in order to go there.

So it says, to the stub resolver; 'This is a name I want' stub resolver goes; 'I know how to receive names from applications', like your browser, or any application running on the computer can ask for, what is the address of this name associated, stub resolver says; 'Great, I got the query and I know what to do, I have a resolver that I know I can ask'.

The way that the stub resolver finds the resolver is actually usually set up by the operating system, the operating system will know a friendly resolver, that's often a resolver that's on the same network you're on. So you're the network administrator will set up resolver they'll say; 'You know what, I don't have resolver go over here', but every stub resolver will have the address of a recursive resolver, it will answer questions?', so it sends this query 'What's the IP address of this', right, you sent in this previous slide, you said; 'What is this to the stub resolver?'

Stub resolver says the same question to the recursive resolver and this recursive resolver happens to be at 4.2.2.2, it's just an address. Okay,

so that's the query, but the other part of this query is 'What is the exact address?', only give me the exact address or tell me to go away? Don't make me go somewhere else.

So the first thing the recursive resolver does, let's assume the resolver just started up, and the reason why I'm saying that now is I have a second example of after it's been running, but and you'll see why so it just started up and it goes; 'I don't know anything about the internet, let me go ask a root server'. It has that list of addresses, that I gave you of the root server, so it picks one out and it happens to pick L, but it can pick any of them because they all have the same information, remember, every authoritative server is going to have the same information as every other authoritative server for his own.

So it says; 'Well, what's the address?' So the recursive resolver says; 'What's the address of this?' now we here in the room, now know that the root server is not going to give it a full address because the root server has a limit amount of information, but it sends it off and then the next thing that happens is it says; 'Here are the name servers, here's the DNS records for .com', so that's still an answer, it's an answer of; 'I'm not authoritative for www.example.com but I am authoritative for .com, I have delegated any questions about .com to that name server or this set of name servers'.

So the recursive resolver now knows this thing of 'What are the name servers for .com'. So it goes to one of those that it got back as an answer and it says; 'Okay, let me try again, what is the address for

www.example.com?' So, these are the name servers for .com, and it says; 'don't know, but here are the name servers for example.com'.

So now we've had two delegations make sense because in fact, we would not want the root server to know the address of all the billions literally, of names out there, that would be silly, then we're back to the first slide that I gave, for those who came in late, of the history of, you know, a tight piece of paper with names and addresses, that wouldn't work, so we are delegating down.

So the authoritative name server for com says; 'Here are the authoritative name servers, for example.com', this is why it's called recursive resolution, it is recursing down the tree, remember the early tree example, it's now moving down the delegations in the tree, and it says; 'What's the address for www.example.com?' and it says; 'Here it is', right? Because this is the authoritative, for example.com and it knows the address, so it says; 'Here are all the IP addresses for example.com', sends it back and then the last up the happy step is here they all are.

So this stub resolver got exactly what it wanted. Even though all of this other work happened, it doesn't even know about the other, it can't see what's happening. It can sit there and wait but it doesn't actually know this. I'm going to stop here for a moment. Did these last set of steps make sense? Does anyone have any questions over that? Was that a yes it made sense? Okay. Any questions on how this happens?

Because this is the meat of everything in the DNS. You know, there's all these things I talked about earlier, oh, there are all these data types? How does it happen? This is how it happens and you are here. Okay. So and then, of course, the last step the stub resolver tells your application, now your application has an address, and it's able to go out and connect and show you pictures of cats. I'm assuming there are cat's people here anyway, sorry. Okay.

What I just showed you is what you often get in DNS 101 classes, and they stop there and that is wrong to stop there because what I just showed you, remember I said; Let's say that that resolver, the recursive resolver, had just started up and it doesn't know anything, now, after what we just did here in this state, this recursive resolver knows things, it knows that it's not going to send more queries to root server for .com because it remembers the answer it got, it's not stupid, it has memory in it.

We all have memory and sometimes we remember lots of things and if at some point I asked you, 'Who lives in your house?', and you gave me a list, I'm not going to ask you again, five minutes later, I remember that and if I'm asking like, 'What is their favorite color?', you might say; 'I don't know but here are all the people who live in the house' but I don't have to ask you each time. So this resolver now knows this, and knows this and knows this, so we can remember it. So why is this great? It speeds up the next set of questions that comes to it. It already knows this.

Now, the interesting thing is, it doesn't know it forever, just in the same way if I said; 'You know, what is the favorite color of all the people in your house?' And you said; 'Well, here are the people who live in my house', they aren't going to live with you forever, this is a good thing, it will change, some people will come in and some people will leave, but you can tell me, by the way, and this information is good for at least the next half hour. I know who's going to be in my house for the next half hour.

So in the next half hour, I am not going to ask you again, I will go directly to them. After a half-hour, I better ask you again because maybe it has changed. So the recursive resolver caches, caching is a fancy word for remembers, the information that it has seen. So now let's do this again. Notice that this is not www.example.com because it might have remembered what the answer for www. FTP.example.com, which is a very archaic domain name.

So let's walk through it again. 'What's the IP address for FTP.example.com?' My favorite thing, one arrow because it remembered this and says; 'I know the name server, for example.com', gets the answer, boom. Okay. So, that is actually the end of the presentation, but that speed up is what happens the vast majority of the time, in the DNS.

So again, if this is the picture that you are used to seeing, I'm sorry, went back too far, if this is the picture you're used to seeing at the end of the DNS 101 class that's true, a very, very small number of times, what is true, the vast majority of the real world is that picture, the

recursive resolver already knows stuff and it remembers it so that the next time you ask a question, it's going to give you a faster answer and not only that, it will save, remember, all of these queries took up network bandwidth, doesn't have to use those.

Questions, anything by the way, so we have about 15 minutes/20 minutes, yeah, anything that I gave or just, you know, I've always been wondering about blah, blah, we can do that. So anyone at the -- oh here.

STEVE CONTE: So we're going to let Paul rest so I can steal his microphone and bring it to the questions.

BENJAMIN AKINMOYEJE: Hello, good morning, Paul, thank you for the presentation. My name is Benjamin and I'm an ICANN66 fellow. Here's my question, what makes it so fast that you don't even know that these things are happening? You know, you read the browsers, you see going all this way, but what is responsible that time factor that makes it happen so, almost invisibly? And how do you notice, like also what is the difference when it's a DNS server that has a memory of what has been asked before and the first time, how do you notice that like, okay, I I even noticed something is going wrong.

PAUL HOFFMAN: Right. Good question. So to answer your last question, how do you know? You don't know, it all seems like magic and that's good because, you're all sitting here taking a DNS 101 class. If you had to take a DNS 101 class in order to use the internet, the internet would not be very big. So this is one of those -- I don't like to call things magic, because in fact, I just explained to you all the technology, so it's clearly not magic but the fact that you don't need to know how names turn into addresses, and your first question, you know, how does it happen so quickly?

It happens quickly, because people early on realized, the quicker that this happens, the happier people will be. We hear these days about 'Oh, Google is always trying to optimize so that you stay on Google or Facebook or whatever', but in fact, in the mid '80s, these were people who had very slow internet connections, I mean, wasn't even the internet, but their connections were literally 1,000th as fast as what you have in your home or actually 110 thousandth as fast as what you have in your home.

So they cared about the speed of the queries as well. So they put in this caching as a way to help them, they didn't know that you were going to be looking at pictures of cats 30 years from now, or, and this is the more important thing, they didn't know that you would be getting medical advice, possibly in an emergency 30 years later, but what they knew was, getting faster answers back was going to help them and therefore the them 30 years from now would also be helped. So they built that in very early.

Early on in the presentation, I talked about how the fact that the DNS was created in the mid-80s, before there was really much of an internet. The DNS technology has moved forward since then, but surprisingly little, if you think about how every other technology has changed in 30 years. DNS technology has changed but only a little bit relative to, even if you look at how the web started in 1993, and you look at it now, huge changes, lots of different mediums such like that. DNS is actually a lot the same, so that's good, very stable and as you point out, still very fast, sort of bad in the sense that you have to take a DNS 101 class to understand it because it's sort of arcane, but does that help?

STEVE CONTE: Did you want to follow up?

BENJAMIN AKINMOYEJE: Yeah. I mean, that that solves it for me. I mean, for me, as a scientist of some sort, I would have just known like, what I wanted was, with all the DNS risk and all of those things, is it humanly possible to know that, 'Oh, somebody is interfering with my connection right nnow'.

PAUL HOFFMAN: Ah, you're asking about interference.

BENJAMIN AKINMOYEJE: What are the timing factors like, and more importantly, I've seen, based on places I've worked in different places, that would it have been fair to start saying, I'm championing for DNS server around me. One of those -- to be around my vicinity somehow, will that improve my experience or it doesn't matter where it is?

PAUL HOFFMAN: Okay, so completely different question, but a very good one and let me summarize. So does having an authoritative name server for something that you're asking questions about a lot close to you help? Yes. Does it help a lot? No. How can you tell the difference between this and this? You can't and it changes over time because everybody is conspiring to make your internet experience faster and there are basically two ways to do that: the DNS and everything else that's in your web browser.

The DNS folks, pretty much they're squeezing out everything they can from this washcloth, but as everything gets faster, they keep up. The other things that are speeding you up. That's outside of ICANN's remit, but the DNS is getting faster without getting technically different, it's just getting faster.

So then you also asked about attacks and DNS abuse and such like that. We're all people in this room. We all have human tendencies, every criminal out there is also a person, they have human tendencies, their human tendencies is to see where things are weak and they

attempt to exploit that. The DNS does not get weaker, but as it gets bigger, there are more places to exploit.

So the reason why there's more DNS abuse now, and I assume a bunch of you are going to be going to the DNS abuse session after this one, the reason that there's more DNS abuse now is not the DNS has gotten weaker, it's because it's gotten bigger along with their In the same way that for those of you who grew up in a small town, and then those of us who are in our '60s go back to our small town and it's a lot bigger of a town, there's probably more crime in that town, not because the towns worse this is because it's bigger.

So, DNS abuse will continue to grow with the DNS, of course, we're all trying to limit that, without limiting the DNS, 'How do you do that?' Dunno, go to a different session, that's not a DNS 101 question. Other questions, folks?

STEVE CONTE: Paul, I want to follow up on that really quickly, one second. I do urge you if you have the time today to join us at five o'clock to listen to the root servers tutorial, too, because it does address some of the speed, latency, or lack latency issues with the concept of a protocol called Anycast, or use of a protocol called Anycast which allows for greater geographic diversity for root servers and things like that and that might help address your question.

PAUL HOFFMAN: Yes, please go ahead.

UNKNOWN SPEAKER: Hello, I just wanted to know a little more detail about the record exchange happening in the delegation; for example, when the records you server calls the root server, what kind of information does it get with DNS records? Does it get the IP address of the name server as well?

PAUL HOFFMAN: Ah, very good question. Yeah, yeah. So I brushed over that because it's, um, but I'm going to come back to, so this will help. When you ask an authoritative question, you ask somebody; 'Tell me everything that you're authoritative for'. It is not authoritative for the addresses as well, but it can give them as hints. So if you don't care for an authoritative answer, you just want to keep going. You can use the hint that was given to you, but if you if you want an authoritative answer, instead of saying I'm going to trust this hint, you go actually look this up itself, so what you're trading off here is more round trips to get an authoritative answer versus speed.

Normally, if you're on a fast recursive resolver, it's always going to want authoritative, because this is an area where people can fool you, they can lie to you, in the hints, or an attacker who's in the middle can lie to you. This is one of the reasons DNS SEC is so important, is, it's slower adding security always slows you down and I didn't cover DNS SEC in here, because it's definitely not a 101 topic, but then you're

sure you're getting the right answers in exchange for having to take more time.

Now, going back to what I said at the end, though, about all the caching, you only have to take that time, occasionally, you don't have to do it for every query, because you remember all of the answers you got and how long they're good for. So an authoritative server that knows it's not going anywhere, is going to give you answers and say; 'You can believe this answer for an hour or for a day, you don't need to come back and ask me'.

So queries that are coming from the stub resolver to the recursive for anything that already knows, it gets it instantly. In exchange for the work that the recursive server did, for somebody else, earlier in the day that the answer took longer. Does that makes sense?

UNKNOWN SPEAKER: I didn't get it actually. I just wanted to know, when you get the initial court does it give the IP of the server also?

PAUL HOFFMAN: Oh okay, so yeah, so here's where an answer that would be like this, where you got this, you might or might not also get the address. If you get the address, it's not an authoritative answer for the address. It says; 'This is just a hint, if you like my hint, you can use it, but if you don't trust the hint, that's okay, go and ask for the actual address for

the name servers of this guy'. Yeah, it could be yeah, yeah. Yes. So go ahead and say that in the mic.

UNKNOWN SPEAKER: I'm just pointing out that in this example, it's different. So the hint is about one and two, but if you look at the authoritative answer, it's about seven.

PAUL HOFFMAN: And the reason for that is, again, decoupling from the parent to the child to delegation. The DNS would be even faster if the parents could answer for the children, but as we all know, when someone says; 'What's your kid's favorite color? You're going to probably answer wrong, you know, they are going to be; 'No that's not the real'.

So, when they designed the DNS, they actually said the parents can tell you where to find the children to ask the question and they might even say, and the hint is their favorite color is blue and so you might say; 'Okay, I'll just trust the parent', but really, if you wanted to know the answer, you have to find the kid, you know, the parent tells you how to find the kid and then you ask the kid, and as it was just pointed out, the kid can say; 'No, they don't know what my favorite color is, or they knew my favorite color yesterday, but now it's pink'.

So that adds complication and it adds delay, but because of the caching, it only adds delay for the first person who asked as long as the answer is correct. So and that leads to an important part of the

DNS, which I did gloss over is, 'How do you decide how long to make the answers good for?' and that's a black art. Make it too long, you can't change it. Make it too short, people are going to keep asking you and there's going to be a lot of delay. Yes?

UNKNOWN SPEAKER: I'm a business person. So a lot of this stuff is new to me.

PAUL HOFFMAN: Yep.

UNKNOWN SPEAKER: I'm in charge of running my company's new dot brand registry, so this is helpful, I have a sort of a basic question. So, you know, you talked about the process, but just can you walk me through, in like a minute, when somebody registers a new domain or redirect, I always hear well give it a little bit of time, it has to propagate through what does that mean? What is propagating through the internet mean?

PAUL HOFFMAN: Ah, so, what they're saying is, give me a little time it needs to propagate it does not propagate through the internet, it's propagate to the Name Server, for the exact reason that the answer I gave him, there's a time to live for even for negative answers. So let's say that I asked the dot com Name Server, what is the address of www.doesnotexist.com and it comes back and says; 'I can

authoritatively say there is no such domain name', but it's also going to say, and you can believe that answer for the next five minutes, right? Because it doesn't want you coming back and asking all the time.

So during that five minutes, if one of your folks registered doesnotexist.com, you need to wait the five minutes or else people at various resolves will come back and say; 'Oh, I'm not even gonna ask that'. So you need to wait for that time live, which goes back to the black magic, do you set a long one? so that no one's asking you these repetitive questions, but then if that information change, you're going to be lying to people until the end of that or do you give them a short one, and then people are going to keep pounding on you for this. You see the balance there.

UNKNOWN SPEAKER: It's primarily a matter of that waiting for that time first for somebody to make a new entry into the DNS system and that being reflected.

PAUL HOFFMAN: And to change the serial number of the zone, which changes every time.

STEVE CONTE: Paul, just to follow up on that a little bit that's, that's per resolver questioning that, so the TTL is 12 hours, one resolver might ask that question right now and 12 hours later, they'll, you know, it'll refresh

one resolver might ask that question eight and a half hours from now and that 12-hour clock starts there. So the propagation is, is a range because it's per resolver asking the question and holding that cache.

PAUL HOFFMAN: And just to make it a little bit more confusing is some resolvers actually ignore what you told them for how long to wait. They mostly ignore it on the short end, they say; 'Ah, he told me to wait, you know, to not ask again for an hour, I'll ask again and a half-hour', which is the tragedy of the commons, right? Is that; 'Hey, it's his CPU cycles, I'm going to use them more'. So, you can't say exactly how you know, you can't say exactly how long to wait, both for the reasons Steve gave and the reason I just gave. It's all a guess, it's a messy database. We still have time for a couple more questions, please. Do you have -- yeah you got a mic?

UNKNOWN SPEAKER: Abdulaziz from UAE [inaudible]. I have two questions and sorry, maybe you answered them in the beginning because I came late two minutes, but just a follow up for that question, so what is the longest time that I will wait for, like when I used to create websites before and I will add the domain name, then for my country, I will wait for like 24 hours, if I use the VPN and enter some other country I'll immediately get the new website. So for my visitors, I used to tell them like wait for two days, for example, and then try again. So what is the longest time that I would expect to wait for?

PAUL HOFFMAN: The longest time possible or the longest time you would expect?

UNKNOWN SPEAKER: Possible?

PAUL HOFFMAN: Possible is 38 years? No, I'm sorry, 37 years because it actually one bit is off. Normal there's all sorts of normal, look around the room, there's all sorts of normals. Some people are like; 'Oh, keep it short, I don't mind if people keep asking me', other people are like; 'No, keep it long for stability'. You're weighing stability against availability. That's what we do in our daily lives all the time. 'Do I have a snack now at mid-afternoon?' or 'Do I wait for dinner?', 'Am I going to be hungry at five, if I if I don't have a snack?' It's the same trade off, it just happens to be in the internet.

UNKNOWN SPEAKER: Oh is that's decided by the registry, right? The registry will decide.

PAUL HOFFMAN: Now we're into the registry. It's decided by the authoritative name server. The authoritative Name Server may or may not be run by the registry.

UNKNOWN SPEAKER: Ah, yeah.

PAUL HOFFMAN: But it is always decided by the authoritative name, I'm sorry, it is specified, not decided by, by the Name Server because there was recursive resolver might say; 'I don't care what you said, I'm going to ask sooner'.

UNKNOWN SPEAKER: But the registry Name Server is -- actually the root will always point to the TLD, registry whether the name exists or not.

PAUL HOFFMAN: Yep.

UNKNOWN SPEAKER: And the registry --

PAUL HOFFMAN: No, no, the root will say that name doesn't exist, if the TLD doesn't exist.

UNKNOWN SPEAKER: How does the root know?

PAUL HOFFMAN: The root has a complete copy of the root zone, so they know two sets of things. One is 'Who are all the TLDs and therefore the second thing they know is 'who is everyone else who I'm just going to say that doesn't exist'. So, if you say to you know if you ask your recursive resolver for the address for www.doesnotexist.doesnotexist, it will ask the root zone ago does not exist you know, like that, that is a nonexistent name and it will immediately come back to you and your cursor resolver will remember that answer for five minutes.

UNKNOWN SPEAKER: I understand that the authoritative Name Server for a domain name determines what the time to live is, but when a new domain is registered, what determines when it's resolved? That would be up to the -- different registries have different policies or that, right?

PAUL HOFFMAN: Correct. And some of them are faster, slower, but that's a business question, that's not a DNS question. And we're out of time and I think we need to stop at the right time. So if you have other questions, or someone in the room right after us?

STEVE CONTE: I don't know.

PAUL HOFFMAN: Okay, so I will sit here until other people come in at 13:00, and if not, I'll just meet you out in the hall. Okay.

STEVE CONTE: Paul, thank you very much for this. It was a very, very helpful, please, everyone.

PAUL HOFFMAN: Thank you for knowing things.

STEVE CONTE: Thank you all for your time here to join us and for your good questions. And I invite you to please join me today on the journey of how it works. We have a next session at 13:30 with the Regional Internet Registries, followed by a session on RDAP, which is the replacement for WHOIS, and finalizing at five o'clock with the session on the Root Server Operators and RSSAC. Those all three will be in this room, in 512G. So please join us again at 13:30. Thank you, have a great day.

[END OF TRANSCRIPTION]