ABU DHABI – How It Works: DNS Fundamentals
Sunday, October 29, 2017 – 08:30 to 10:00 GST
ICANN60 | Abu Dhabi, United Arab Emirates

STEVE CONTE: Good morning, we're going to give it a few more minutes, see if people can get their coffee and get up here. Again, this is going to be a DNS Fundamentals Class, so if you've got experience beyond the fundamentals of what DNS is, you're probably going to fall asleep in this room, which is fine, but if you snore I'm going to come over and give you the mic. Other than that, please make yourself at home and we'll get started in a few minutes, thanks.

Good morning, everybody. Welcome, I'm Steve Conte. I'm the Director of Operations for the Office of the CTO and we put on these How It Works Series tutorials and presentations at every ICANN meeting. The purpose of it is really to give foundational knowledge to people who are new to the ICANN community or want a refresher on aspects of the technologies behind what drives ICANN policy making, things like that.

Today in general, we're going to be doing a DNS Fundamentals, followed by a DNS Abuse primer on how to identify abuse in the DNS. We do training for law enforcement around the world and so it's kind of just an opportunity to show the community what

and how we teach law enforcement to look for DNS abuse. Then we have Internet Networking by Alain Durand, that's going to be about numbering and routing and naming and putting all that together to look at it from a holistic perspective on putting all these protocols together and how does it work on the internet.

And then we'll wrap it up today, this afternoon, with a session from the Root Server Stability Advisory Committee who will talk about the RSSAC, what they are, talk a little bit about the root servers and then also speak about their relation with the ICANN community and how it all fits together on that. If you've got the day to spend with us, I invite you to please spend it with us; if you don't have the day to spend with us and you want to catch some of the sessions, we do a repeat performance on all the sessions tomorrow.

So if there's a time slot that you can't make but you want to see the session, please look at your schedule for Monday and we'll be doing the exact same sessions tomorrow. If you sit through the whole day with us, you'll learn nothing new tomorrow on the sessions, other than different questions coming up, you only have to show up once.

With that, we're going to talk about DNS Fundamentals; this is a introductory level tutorial on what DNS is and the parts of it and how it works, and then we'll touch a little bit about DNSSEC.

There is a DNSSEC workshop later this week, so if you want more in depth knowledge, I would suggest going to that workshop as well. This is just a touch base and be a primer.

Go to the next slide. Before I get going, I'm a non-formal speaker, if you've got questions, please feel free to raise your hand, I'm happy to stop the presentation, have the dialog right then and there. Personally, I think that's the most important time to have a dialog, while everything is fresh in people's head, so don't hesitate to interrupt, pause, whatever, raise your hand; we've got a separate mic I'll bring it over to you and we can have a dialog from there.

With that, we'll go ahead and get going. We're talking about the domain name system, DNS, and if we look at IP addressing either IPv4 or IPv6, it's hard for a human to remember IP addressing. It's fairly easy if you go to the same v4 address over and over again, but when you get to v6, it gets really confusing very fast and hard to remember. There need to be an opportunity for humans to be able to navigate the internet without having to worry about physical numbering and remember numbers.

I remember when I was a kid I used to know every phone number of everybody that I used to know, and then smartphones came around and now I know my own phone

number and that's about it. It's kind of like the same thing, we're using smarter technology to help humans interact on the internet. So way back in the early days of the internet, way, way, way back then, names were simple, there were no domains yet, it was pre-domain space, it was single label names.

The internet was very small, it was ARPA, it was some universities and a very limited number of servers out there. You could use a single label and as long as that was a unique label, that was it, there was no domain behind it. It was a 24-character maximum for them, and they referred to it as host names. That's where we get some of the language, the lexicon of the DNS system out of there.

Next slide, please. At that point, it was mapping the IP addresses to the names, it was called name resolution; that happened even with the host names, with the single label stuff, and each computer on the internet at that point had a file called a host file, and here we see it's host.txt. And that file was a list of every single resolution between a host name and address that was on the internet, and it was centrally maintained by the Network Information Center at SRI, and people would send updates, so if you wanted to change your address for a host name or if you wanted to add a new host name, you'd have to send that update via an email to SRI. And it was centrally

managed and the change release was once a week, and they would release it via download through FTP.

Next slide. There were some problems with that. As the internet grew, it became harder and harder to manage centrally the host name file, the host.txt file, and you started getting some name contention, more and more people wanted computer as their label or something, so there was naming contention and there was no good way to prevent duplicates because people were sending emails and it was kind of a first come first serve but not really, it depended on who was managing it at that point.

There was no process of synchronization, so even though SRI would put this host.txt file available for download weekly, there was no way to ensure that the consumers of that file would actually go and download the latest, greatest release. It's an absolute, no one ever had the same version. I'm sure there was a same version of the files out there at some point, but there was no clarity on the synchronization, we weren't using all the same version at any given time reliably.

As the internet grew and the host names grew, it became larger; and remember, this was the early days where we had at most ISDM lines or dialup, so bandwidth was very small then, so even though in today's terms the host names file was relatively small,

it was very big if you're trying to download it on a 1200 baud modem or something, so we had load issues as well.

It became apparent that a centrally maintained host file just didn't scale. The internet was growing bigger and larger and larger than people thought it would at that time. Basically, like any new community, when you have a town and the town is growing, it's usually the road system that suffers first. You have all these houses that get built and you have smaller roads because the architecture, the infrastructure hasn't caught up to that yet, and that's exactly where we were with the internet at that point.

All these houses, all these servers were being introduced and the architecture, the roadway for those just wasn't up to scale yet, so they had to reengineer and figure out how to do that.

Go to the next slide. Discussions started in the early 1980's on a replacement and the goal was to address the scaling issues of the host.txt file, simplify email routing. Email was a brand-new thing still, it was easy back then to email because you would just do user@host, and because it was a single label, it would just work; but the more and more hosts that were coming out or the more and more people wanted a more robust email system, maybe that user@host wasn't actually the server that was going to serve email.

They wanted a more robust system that was going to build stability and reliability, resiliency into the network. The result on that was the domain name system and through the work of the Internet Engineering Task Force, the IETF -- are any of you familiar with the IETF and the work? I got a handful of people.

Quick tangent there; the IETF is a body of engineers who develop open internet standards to use on the internet. Any time you're using http, ftp, DNS, any standard that's being used on the internet, most standards that are being used on the internet are developed through the work of the IETF, and their product is something called an RFP, a request for proposal. The RFP documents are the standard documents that developers and system administrators and people who operate the internet, they use those to create a predictable environment for the traffic to flow.

The IETF started working on some documents for DNS; I have a couple listed here, but if you go and look, there's probably tons and tons of them out now. Every time there is a new change to the DNS infrastructure, or the way that they want it to operate, or someone thought of a different unique, interesting way to use the DNS, they will probably go out through another internet draft process through the IETF and hopefully turn that into a RFP which creates a new standard on that.

The next slide. Before, the naming system on the DNS was a single file, it was a host.txt file and it was centrally managed. One of the challenges that they wanted to do was the scalability, so they created DNS with the concept that is was going to be a distributed database, that the data is maintained locally but you can get to it globally, and that opened up a lot of areas in which the scalability really improved itself, and we'll go into some that.

They also defined some of the lexicon, some of the language in the DNS, so we have what we call resolvers, which are basically if you have your laptop open or if your phone or iPad or whatever, it's the device that's asking the question, making the query to get that name resolution. Then we have name servers, which are the servers out there that answer that question, and then we'll have a hybrid of the two, the ones that can ask the question and give an answer, and those are called recursive servers; we'll go into in a second as well.

Work was done to optimize the performance of the query response process because if you're asking, I have a couple of slides, we'll go through the whole resolution process, but if you're asking the same question up to a certain point, there's no need to be re-asking that question over and over, so you can cash specific parts of the query to make that query faster and

then you can go and drill down to the unique point on that to make the query and response faster.

Replication, you can have more than one DNS server per domain, and we'll go into the details on that as well. Allows you to have geographic and network diversity as well as replication, so if one server went down and you had more than one DNS server, your consumer can still get the answer from some of the replicated servers as well.

Next slide. The components at a glance, so we talked about the resolver; this could be anything, it could be a phone, it could be your thermostat at home maybe, it could be your laptop, it could be anything. In this case, we're talking about a phone and we're using Safari, and when you type in an address, it's going to make the call to the stub resolver, which is a small bit of code in your phone's OS, or your laptop's OS, that acts as the gateway between what you typed in and the DNS system.

Then we go to a recursive name server; typically, this will be your company or at your company or maybe at an ISP, and it's the hybrid, it's going to answer eventually the queries from the stub resolver, but it's also going to act on the behalf of the stub resolver and act as a resolver to ask the queries in proxy for you to other servers. The recursive name server is a hybrid between

a resolver and a name server, and so we'll get to that in the actual DNS flow.

Then we have authoritative name servers, and all they do is they answer the questions and they're authoritative, so that means that they answer the questions for the zone that they're responsible for and they answer with authority, so when the answer comes back, the recursive name server says, "Okay, this came from the authoritative server," and then it gets the answer back to you.

Go to the next slide. DNS is made up of names space. What it is, it's an inverted tree; if we take this and flip this around, you'll see the tree spot. We call the very top of it the root. If we go back to the tree analogy, there's our roots that are in the ground and then we have branches going up from there.

We have the root, we have top level nodes; each space is called a node, each label on that. We have second level node and then we have third level nodes, and if we look at this from a perspective if we hide the root, hide that end knot, we have www, which is a third level node or a host name could also be used, we have example which is a domain that you might go to a registrar to register, and then we have .com which is a registry. So your URL or your domain name is built into this tree structure

and as we go through the resolution process we'll see how that's done more.

Anyone know when you see and xn--, what that means? IDN exactly, Internationalized Domain Name, and what that is, it's a way for the DNS system, which was built around the concept of passing ASCII characters to utilizing non-ASCII characters, be it Arabic or Hindi or some other script, that script that doesn't use ASCII character. The xn-- is a code that is used to determine what language and what domain that is in the unique script.

Go to the next slide. We call the space inside the characters on these labels, we call them LDH: Letters, Digits and Hyphens. Those are the only legal characters that we could have in the DNS, and that's why I brought up the IDN part of it too, because if we're using an alternate script other than ASCII, it doesn't fit necessarily into LDH when we talk about letters in the ASCII sense of the word.

Each label, each of these are a label, each label can be a maximum of 63 characters, and if we look at a complete domain, www.example.com for instance, a complete domain can only be a maximum of 253 characters long. We have to keep that in mind when as a consumer you probably don't want to type a 253-character long domain; I would typo it, I know that for a fact, but you have to keep that in mind too when you're building

your domains. With some of the IDN, that could be some of the challenges around that, to make sure that it fits into the label size and the full domain size as well.

When you're on the internet and you're typing, and advertising does this a lot, if you're looking through a magazine or if you're up on a billboard, you'll see www.unitedairlines.com and the u and a on United Airlines are uppercase, so capitalized, cause they want to promote their brand. The DNS doesn't care about that, it's case insensitive. It means that it's basically going to turn everything into lowercase before it does any parsing at all, and say, "I don't care about the case, there's only one letter 'a', there's not two, there's not a lower-case 'a' and an uppercase 'a', it's just one letter," so it doesn't care about case sensitivity.

Next slide. And then, every node has a domain name. The domain name is a sequence of labels from a node to the root separated by dots. Here we have highlighted www.example.com, these are all labels; this part here is the node, it has the domain name. Node isn't used in common language, you probably won't hear it again in ICANN meetings after today, so don't get hung up too much on that. We talk about domain names, we talk about domains and top level, TLD's and second level domains and things like that.

The next slide. A fully qualified domain removes the ambiguity identified in a node. If you just went www.example.com, you're not telling the system, the domain system that that's it, there's nothing else, because for the case of UK and some other country codes TLD's, they like to have like a co.uk, which is a separate second level domain inside of their country code. You might actually have a four layer instead of three, you might have four or more layer domain name. If you don't put the dot, it just kind of keeps thinking that there's going to be another referral on there and it's going to keep looking until it thinks it finds it all.

If you put a dot at the end, if you say I'm going to go with host domain top level domain root, the dot it's called fully qualified domain space; it says, "I don't care if there's going to be more, there could be more, but this is exactly what I want to go to; I want to go to exactly www.example.com dot and the root." So if there is ambiguity, you're removing it by being very specific and saying, "I'm ending my query with the root, with the dot right here."

Most of the time you won't do that, it's typically done for lookups and operational aspects and things like that. Typically, if you're going in, you could, you could go to your browser right now and say, "Go to www.google.com.", and it will absolutely

work, but most people don't use that final dot, it's kind of just hidden, it's assumed that it's there.

Go to the next slide. If we look at a domain, it's a node and everything below it. I was saying before that this is typically host, this is typically domain, this is top-level domain and then the root. That's common language but in essence, anything below the node that you're talking about, is the domain, so in this case if we're looking at .com, anything that's inside the com zone is the com domain, which means that all this stuff is related to and managed at least at some level, delegated to the management within .com. If we look at uk, same thing, this is the domain and all the space underneath that would be within that domain.

When we talk about a domain name typically in your everyday life, you're usually looking at it from this point, so your domain would start under example and then everything underneath it all, the host, all the servers would be within that domain space, and we'll talk about the resolution in a second and that might make a little bit more sense.

Next slide. When we're looking at a domain, we're talking about delegations, so there's a certain aspect of responsibility within the domain, responsibility of editing and managing that domain

space, and then they can further delegate that out to other people to manage smaller spaces.

When we divide that up, we allow for distributed administration and that's really good because it gives more control to more people, to more organizations and servers that allows for greater scalability and less reliance on a single point which allows for greater scalability. These administrative divisions are called zones. They're sometimes the same as a domain, but as I said, sometimes you can delegate parts of a domain out and give those zones to other entities for them to manage.

The delegating zone is called the parent and the created zone is called the child, so there's always this parent/child relationship. If you're in IT you might be familiar with that term with some databases and other types of relationships between data points on that.

Next slide, Cathy. If we look again at the name space -- and we're going to go fast on this, next slide -- now we're looking at it from administrative boundaries. These are zones, so we have the root zone; and again, we'll have a session this afternoon about the root serve operators and so they'll go into greater detail about what's different about managing the root zone versus managing another zone.

Then we have the top-level domains, they're all zones, and then we have typically what's called the domain name; when you go to your registrar and you register something, you have that layer here and each of these is a delegation point. The root delegates .com, in this case they delegate VeriSign, to manage the space inside of .com. .com doesn't want to manage each and every domain within that .com space, so they will delegate the management of the domains out to whoever registered them. In this case, if we had example.com, there might be a pointer in .com, there will be a pointer in .com, but the actual management of this zone space here, this domain space here, will be whoever operates the example zone.

Okay, next slide. As we said before, name severs answer queries, resolvers ask the questions, so a name server is authoritative if it manages a zone that has complete knowledge of that zone. It can provide definitive answers to queries about that zone. If I had example.com and I have www, I have ftp, I have all these different labels inside that and they're all pointing to different servers, I, as the manager of the .example zone, my main server is going to be authoritative cause they know all the answers that relates to the example.com zone.

So that's what we call an authoritative server. And there's different levels of authoritative servers depending on how high

up the tree or down the tree you go. For my own instance, I have a domain called conte.net, my last name.net, and I only have email and website on it, and I don't even think my website's working right now. So, my zone management is quite small, there's only two entities inside my zone, but I'm responsible for that. .net is responsible for at least some pointers to the delegations of all the domains that fall within the .net zone, and then the root is responsible for the pointers to refer to, at this point, everything in the DNS system itself, to point to the next point level. When we go through the resolution, that will make a little bit more sense.

Zones should have multiple authoritative servers. Again, we're looking at geographic diversity, we're looking at network diversity, we're looking at redundancy and it also will spread the query load out. On larger organizations, or if you're looking at domain space from a TLD perspective, it's going to lessen the load on each individual server.

Go to the next slide. Before we go into the resolution process, are there any questions at all, any comments? I know there's some people here who aren't DNS Fundamentalists -- that's not right -- they know more about DNS than they're willing to let on to. Anyone want to correct me or ask a question, otherwise we'll jump into resolution itself. Let me bring you a mic.

BARRY LEIBA: Hi, my name is Barry Leiba. I'll just say about the case sensitivity that IDN's throw a little wrinkle into it. If you look for a.com, the a gets turned into a lowercase 'a', no matter what case you started off. If it's a Cyrillic 'a', you're responsible for turning that into a lowercase Cyrillic 'a' before you encode it into the punycode, the xn--, otherwise you have a problem cause if you encode an uppercase Cyrillic 'a' as xn-- something, and send that on, it doesn't match correctly. It's tricky and then you get things like Turkish 'I' that really throw wrenches into everything, it's quite a mess.

STEVE CONTE: This is the iceberg that we call IDN, but at the point of DNS itself, when it gets to the xn, it's not upper or lowercase xn, it's all insensitive?

BARRY LEIBA: Exactly.

STEVE CONTE: Thank you for that clarification, I appreciate that. IDN is an interesting beast because of things like Barry just said, but other instances, too. There's a lot to learn about IDN, we're still

learning about IDN although we have code out there and we have domains or TLD's that are IDN, but there's always new things. I just called it an iceberg, it really is, it's conceptually easy to sail, I want to have non-ASCII characters on the internet and then you start getting into the nuances of like what Barry just said, and then you see that there's a lot under the surface of IDN's.

If you're interested in non-ASCII characters on the internet and you're not yet involved with IDN, I would highly recommend looking into it. I don't know if there's any sessions this week on IDN -- there is a session? So, you know, look at the schedule, find out, otherwise there's a lot of material out there to read. There are working groups I believe still active in the IETF on IDN. There's a lot of places where you as a consumer can be involved, but also you as a potential developer of punycode, which is the code in between the website and the DNS, on how you could be involved or follow IDN on a greater scale.

Any other questions before we get into resolution? Resolution is painfully easy. Let's say you're going to a party and you want to meet that person in the back; it might be your boss's boss and you want to make yourself look good, or make yourself known, so as we enter into resolution, just remember, it's all about who you know to get to where you want to go.

So here I am in my stub resolver and I want to go to www.example.com on my phone in this case, and I'm going to use Safari as my browser, so I'm going to go there, I'm going to type www, and I'm going to mistype it because I've got a big thumb, example.com and I'm going to hit enter.

Next slide. It's going to go through an API from my application to my stub resolver, which is built into the OS of some type and it's going to send that query out to its recursive name server. When you turn on your computer, most of the time these days you use something called DHCP which is a protocol, a RFC standard, that says, "We're going to give you an IP address out of a certain pool and we're going to give you some DNS servers." So you kind of inherit all that stuff when you log into a network.

In this case, when I went to my phone, my default name server, my DNS name server was 4.2.2.2, so when I sent the query it sends it up to there and says, "Hey, I want to go to www.example.com," and this is making the assumption that it's never done any queries at all yet, it just got turned on. My recursive name server says, "I don't know, but I know and we'll go into why it knows but I know the address to the root servers so I'm going to go ask a root server." In this case, I'm going to go ask L root, we'll talk a little bit about how it chooses which root server, it doesn't really matter at this point.

The recursive name sever is now acting on your behalf, so remember where a resolver, on this side we're a name server but on this side it's acting on your behalf so it's a resolver, so it's going to go and ask the question to L root and say, "Hey, I want to go to www.example.com," and L root is going to say, "I don't know, I don't know how to get there but I know how to get to the .com server, so you might want to go there," so it's going to send a referral back to the recursive name server and says, "I don't know, but go check .com, here's some IP addresses."

Next slide. Recursive name server says, "Hey .com, I want to go www.example.com." .com says, "I don't know, but I know the IP address for the example.com DNS servers, so here's some name servers addresses, send it back as a referral and go ask them." So now, my name server, again this in my ISP or my network or whatever, goes to the example servers, and says, "Hey example, I want to go to www.example.com."

Now, if we remember, if we look at zones and if this is an authoritative name server for example, that means it's authoritative because it knows everything inside its own zone; so now ns.example.com will come back and say, "Oh yeah, I have a host or a label called www, it's part of my example.com zone, so here's the address or addresses that relate to that," and it does back up my recursive name server.

My recursive name sever sends it back to my stub resolver on my phone and says, “Here’s the addresses or address,” and that sends back to my application, in this case Safari, and then my application can go and have a direct conversation based on the IP addressing with the www which is somewhere in the example.com zone. After all this resolution takes place, it can go and have a direct conversation. Again, it’s about who you know. In this case, the stub resolver only knows one or two DNS servers, so it says, “Okay if that’s all I know, I’m going to go ask them.”

The recursive name server only knows if it’s brand new, only knows the address to the root servers because there’s a file, we’ll get into that. It will say, “Well, if I don’t know anybody else, I’m going to go ask them.” The root servers have information for the TLD, the Top Level Domain servers, .com, .net, .uk, .ae, any of the top-level domains, so it will refer down to the next person, it’s all about who you know. I don’t know your destination, but I know the next path to get to that destination and so on down the line.

Does that make sense? Any questions about resolution? That’s really at the heart of it, this is really what DNS is, it’s about who you know and how to get there.

Next slide. Cashing is an ability that -- go back one slide; I'm sorry, Cathy. When we did this whole thing, this process, it typically takes a long time, maybe up to 50, 60 milliseconds to do that, and in the internet speed, that's like years, it's a long time. This was for example.com, www.example.com, but now let's say I want to look at the latest footwear and I want to go to nike.com. Well, we know that because of the .com part that it's already asked those questions, so we really don't need to ask those questions again within a certain period of time.

Next slide. And that's where cashing comes into place. When we look at it, we try to speed up the resolution process by cashing certain pieces of the query response chain that took place. In the case of when we looked up www.example.com, the recursive server it now knows the IP addresses and the names of the .com zone, and it also now knows the names of the addresses of example.com servers and the addressed for www.example.

But let' say I'm still in that zone and I want to go to ftp.example.com. We're hitting most of this, we're just going to a different final label. We're going to go through the same process from the users perspective, we're going to go to ftp.com, it says, "Hey, I want to know the address to ftp.example.com." Your recursive name server's going to answer the exact same way, kind of; it's going to say, "I don't know, but before it said I

don't know and I'm going to send you a root server, now it says, but because you already asked about www.example.com recently or somebody did, I have this part of it already cashed, I've already talked to a root server, they already told me I don't know, go talk to .com, .com has gone back and said I don't, go talk to example.com."

Example.com said, "Here's the answer," so because it's already gone through this part of it, those results were cashed inside the recursive name server, so now it doesn't have to do this part. Now it knows I can go right to example.com name servers and ask the question, because I'm looking for a different label, but it's still within the same zone that I'm looking for.

Next slide. This time it goes to ask the name server, and just like before, it says, "Here all the IP addresses for ftp.example.com," sends it down and sends it to the resolver and to eventually the application. We've just removed two thirds of the query process by doing that, which speeds it up dramatically. Now instead of the lengthy 60 milliseconds it might have taken, maybe 20 milliseconds, which is really fast response when you're looking at it.

The higher level you go, the more queries you're going to get. Example.com probably doesn't get a lot of queries, it's a domain, it' a company or something like that, it gets a fair

amount of hits a day or whatever, but not huge. When you get to .com or any of the other TLD's, your query rate around the globe increases dramatically, so there's a lot of people asking about .com zone information. You have a lot of queries coming into here.

When you get into the root server, it's the same, you have a lot of queries coming in because they're called priming queries. We need to know information about a TLD, so the more we can remove the queries by having cashed the answers, the faster the responses are going to be for the consumer but the less load that is going to hit on the TLD and the root servers as well which means it will respond faster to the queries that it is answering for.

Next slide. We talked about synchronization, any questions about resolution? This is the problem about being the first session of the day, everyone's still precaffeinated. Feel free to ask any questions at any time.

In the beginning of the deck, we talked about synchronization issues when we were talking about host names, when it was just single label and SRI was sending out an FTP file once a week, but there was no way to guarantee a synchronization on that. One of the things that DNS as a protocol does is, it builds in the ability for authoritative server synchronization, and that's a

process in which you can keep your data in sync across multiple authoritative servers.

As we mentioned before, you want to have typically more than one name server serving your data, be it if you're a example.com, the zone administrator for that, or especially as you move higher up into the TLD or the root range, you want to have more authoritative servers out there.

This was built in to DNS as a protocol. There is a relationship called the primary and the secondary servers, and it's kind of changed throughout the years but it's still there. It has no bearing on the consumer; as the end user, you have no way really of knowing which is your primary, which is your secondary server because they're serving the exact same data.

The relationship is really about transferring zone data. The primary server, if I had a cluster of three servers, one of them has to be primary, two of them will be secondary. The authoritative data that I have, that I'm modifying, will go into the primary server, and then the secondary servers will call up and say, "Hey…," you know, it's set on a certain time level, every x amount of time they'll go up to the primary server and they'll say, "Are there any updates?" Most of the time they'll say no and sometimes they'll say yes, and they'll say, "Oh great, give me the new data," and so then you'll get it a synchronization off of that.

The server that handles the zone file is called the master server, it's a lot of times the same as the primary server but it doesn't have to be, and we'll talk about that, especially about the root servers and probably many of the top-level domains as well. We'll get into the difference between master and primary as well.

The master also notifies the primary of changes as well, and so it will go through the process. This whole thing is really, primary and secondary, is really about keeping the data in sync, it's not about any one server being more authoritative or having better answers than the others.

For a long time, people at the root server level would think, well a .root servers.net is probably the most authoritative, it's the first, it must be the primary, it must have data faster or get its updates faster and all that, but that's not the case; all of the root servers are equal in the fact that they're serving the same data, they get those changes within milliseconds of each other. It's an equitable way to share the data, to synchronize that data and make sure that it's being served in a distributed manner.

Next slide. We're going to talk a little bit about the file itself, the data inside the authoritative zone. Within the DNS protocol there's standards on how to build that file and what it's called, and that's called the master file format, and the file itself is

mostly -- and times have changed -- but mostly an ASCII file, a flat file that has simple data in there, and we'll take a look at a sample one in a slide or two.

A zone file, if we remember, the zone is the -- in this case, example.com is the zone and it's authoritative and knows everything about all the services inside of example.com, so www, ftp, whatever host name you want to use on that. The zone file is the file within the DNS structure where you manage those labels and you can set those labels.

Cathy, next slide. We can have different kinds of data inside a zone file. We have name to number, it's called an anchor record, an a record, so if you're just doing a host name to an IP address, we'll use an a record. We can have mail exchange records, we can have PTR records, we can have canonical name records, there's all kind of different record types that we could have. We could IPv6 pointing, we call it "Quad A", AAAA, four A's.

Next slide. A zone consists of multiple resource records, multiple entries of that, and that creates the zone for that domain. Every zone has at least one zone file, and you can never have multiple zones inside of one file. There's at least a one to one relationship between the zone file and the zone itself, you can't have one file that manages example.com and nike.com; it just won't work that way, they'd have to have separate files.

Next slide. If we look at the resource records, and you might hear throughout the week things called RR Types; those are Resource Record Types. Resource records have to have five fields, they have to have an owner, they have to have a time to live which is the amount of time that a record can be cashed, you as the administrator of that zone can manage that. It has to have class which is mostly unused. In fact, I've never seen a different class other than IN Internet Class, and DNS was envisioned to be able to run across different types of network and stuff, but internet class is the one that's most widely used these days, I've never seen anything else. I'm interested if anyone ever has or does use a different class, I'd be interested in how it's used.

Then we have resource record types, the different types of data, and we'll go into that. Then we have the data itself.

Next slide. Anything in brackets are optional, they're kind of just inherited into the file. A lot of times in a zone file you won't see owner, you won't see TTL, except at the very top of the file. You won't see class because it just inherits the IN, the internet class into that. What you normally see is the type, the resource record type and the data itself, and then the relationship around that.

Next slide. Here's some commonly used resource record types, we already mentioned a record, an anchor record which is for

IPv4 addressing. We have Quad A, 4 A's, which is for IPv6. We have NS which is authoritative name server IP addressing, we can go into why that needs to be used.

We have what's called an SOA, a start of authority record; this appears at the top of the zone file itself, it appears with the apex. This is what is in there that the other ones can inherit from. We had those brackets in the last slide, there's at one point some of it has to be in there and that's in the SOA record, and when you're looking at a zone file it will tell the DNS server the very basics of what it needs to know to be in that zone.

We have something called a CNAME, which is canonical name; it's an alias basically, it's Steve to Cathy, but at some point, there needs to be an a record behind that.

We have an MX record, which is a mail exchange server, and we have PTR which is a pointer for reverse mapping, and we'll kind of get into that as well.

Go to the next slide. There are a ton; as of last year, there was 84 resource record types in the DNS infrastructure. This slide itself is in the -- either is now or it will be soon, in the schedule, so you can grab this deck and drill down, so don't worry about writing this URL.

This URL points to this, which is really wonderful to see on a screen. This is an example of IANA's DNS resource record page registry. This shows all the different types of resource records, RR types, that are in the DNS. If you look, some of them are obsolete, this one was called an MD, a Mail Destination Record, but they obsoleted it and they use the MX record, but this still records and covers all the RR types in there.

Most of the time you use the handful of ones that we went to on the slide before that, the A, the quad A, MX, NS records; there are some unique ones for very specific purposes, and if you are interested in what those RR types are, you can go to the IANA Resource Record Type Registry, and again, that link is in the presentation and you can grab that presentation out of the schedule.

Next slide. The most common use of DNS is -- when we started this presentation, it's about I can't remember an IP address. There's so many of them that I have, I can't remember which one to go to, or the server changes I want to get to and now it has a new IP address and I don't even know that. DNS was meant for humans to consume resources on the network, on the internet.

The most common use of the DNS is to map names to numbers, and we do that through the A Record, or the Quad A Record where we have a label, we say it's an A Record and we assign an

IP address to that label, and that's really it. When you go to example.com, the DNS server says, "Oh, you're looking for that IP; it's here in my zone file, I'm going to respond, I'm going to answer that query with an IP address."

Or I'm looking for the v6 version of that record, the Quad A, and says, "I have the information here, I'm going to send this record back, this information back."

Next slide. The NS Record type is interesting, it's one of the only record types that appear in both the parent and the child zone. When you're doing a referral, when we go back to that resolution page and we asked the root servers, I want to know where www.example.com is, and the root server says, "I don't know but I know the address to .com," and it gives you a referral, it knows that because it has a listing inside its zone that points to example.com and points to some name server names.

Inside the example.com zone, as we go down the chain, it will have that too, so they will match each other. The left side is the name of the zone that you're looking for, the right side is the authoritative name server that is serving the data for those names.

Go to the next slide. As we just mentioned, if you're looking for .com, this list of name servers will be in both the parent, in this

case the root and the child which is in .com, this list the name servers in both places.

Next slide. When we're looking at .com and we have example.com, that means we have to have the name server records in .com and we have the name server records in example, now there's an issue here. So if .com is the parent to example.com, and it's saying, "Well, the name servers you need to look at are ns1.example.com," there's a problem because the .com is within its own zone, so it's getting itself into a loop.

"Go ask example.com." "Well, I don't have the IP address but here's the name, go ask example.com. Oh wait, that's a .com, go ask example.com;" it starts getting itself into a recursive circle. There's ways to fix that called glue records. And what you can do is, you can put the IP address, once you show that there's a name server with an NS record, you can than add an A Record to give it a hint on where to go to the actual IP address, so that way you're not getting yourself into a circular dependency on that.

So you add a glue record when you're setting you're name server records to the parent, you want to add glue records below that so that it knows if it's serving within the same zone, in this case .com, it knows that it can break that dependency and say, "I'm going to go to these IP addresses because someone gave me glue to get there."

That's kind of weird, does that make sense to everybody? I'm getting some nods, I'm getting mostly snores, but okay.

Go to the next slide. The SOA, the Start of Authority Record, I said that this has to be in every zone file and it does. It's the part where you're kind of giving it the default values within the zone itself. It gives a name server that's considered the primary name server for the zone -- this used to be a way to contact the administrator of that zone, that's probably not as widely used as it used to be, but basically you cannot have the @ sign in a zone record. What they did was, they swapped the first dot and pretend that that was an @ sign. So you actually have an email address here, hostmaster@example.com. Again that's not as widely used as it used to be.

You have a serial number; this is extremely important because the serial number is the item that is being used by the DNS system to create that synchronization between servers. When a secondary server goes and talks to the primary server and says, "Do you have an update?" it does that by saying, "Here's the serial number that I have, do you have a serial number that's greater than that one, that increments that one? And if it's the same or less, it's going to say no update; if it's greater, it will say, "Yes, I have an update."

One of the best ways you can make a serial number that's both human readable but also guaranteed to increment, is if you use the year, month, day sequence issue. If you put the year first followed by the month, it will always increment; if today is 2017, 10, 29 and then sequence 00 or sequence 01, that means that the next time I go and have a sequence number it's going to increase by one. If I update this tomorrow, I'm going to go to 2017, 10, 30 which is going to increment, so this guarantees that as a human you can see when this zone was last updated because it's a date but the way that this date format is put in it also guarantees that it's a sequential number, it's going to increment and it's going to show a sequential relationship between the secondary servers and the master, and the primary server on that.

You have a refresh, various values, retry, expiration, which is related to the cashing; it's called a TTL Time To Live, when that TTL expires at the cashing level when it holds that data, it looks at the TTL and starts a countdown on it and says, "This data is going to be good for that amount of time, and once it passes that amount of time, I'm going to throw this away because I don't trust it anymore."

A lot times the default will be 86400 seconds, which is the amount of seconds in a 24 hour day. So if you ever see 86400, that means that it's typically a 24-hour cash, it will keep it alive

for that amount of time and then my cashing servers are going to say, "That's older than I need," and then it's going to throw it away, it's going to flush the cash and it's going to go and start that query process from the point when the cash was dropped. It might hold the root cash longer than a domain cash based on the administrator of that zone. If my root cash is still good but my TLD cash expired, I won't go talk to the root, I'll go talk to the TLD again to refresh that cash. Yes?

UNKNOWN SPEAKER: Do you think that the TTL is supposed to me managed by the zone administrator only? As resolvers, don't you think they have the right to change the TTL sometimes?

STEVE CONTE: Boy, that's a great question. Do we have any zone operators in the room here? I'd like to get your take on it, if we have any. All right, then you get my opinion. I think that's okay as long as when the operator, if you're managing the recursive name server, if you don't extend the TTL beyond what was in the zone, you can make it shorter because then you're forcing your service to go and re-grab data.

If you make it longer, there's no way to guarantee that if you extend beyond you have a good synchronization of data. If I was

operating a recursive name server, I would stay with either the default TTL that was given to me or shorter, but never longer than what was stated.

UNKNOWN SPEAKER: Yeah, but what will happen if the administrator makes it for one second for example? This will utilize my resolver resources. Imagine huge websites, I will not name any websites, but huge websites with time to live for one second.

STEVE CONTE: That's a great example, and something that I should mention here is that as an administrator of a zone, you need to find the proper balance between a good TTL and one that's causing undo queries, and that's why 86400 is a good standard on that, you're talking about a 24-hour period.

But if I was going through -- let's say you're going to my zone and I'm going through a migration of services, multiple servers; during the period of migration I want to set my TTL short because I want to make sure that new data gets out quickly. If I forget to change that back, then exactly what you just said, suddenly I'm overutilizing your resources cause now I'm asking you to come back and refresh your cash more.

It's a tricky question because there's that chance that you might not know that I'm not done with my migration yet, and you might say, you know, "Well, I'm going to set it longer," but in that period that you're setting it longer and I haven't yet, I might have changed IP addresses on a core service, now that your customers can't get to because you've modified that TTL in that cash. It doesn't always happen, but there's potential there and it gets tricky on how to manage that. Alain's got a comment on this.

ALAIN DURAND: Good morning, my name is Alain Durand. If on the resolver you decide to overwrite the TTL and make it lower, the thing is that you're going to create pain for the authoritative server because you wasted more queries.

UNKNOWN SPEAKER: I'm making it higher, not lower. The problem is with the low values.

ALAIN DURAND: If you make it higher, that will be good and back. It will be good because it will protect you in case of a failure of your authoritative server. If you know for example that your

authoritative is kind of shaky and sometimes it goes down and you still want to be able to resolve it, so if you increase the value than you will protect yourself against failure, so that's a good thing.

But on the other hand, if you make it higher and the authoritative server in the meantime decides to change the record to point to something else, you will not see that change, you will have cash value for way to long, so that's a risk that you will be taking. So, you can do that, there's nothing that will prevent you from doing it, but at your own risk.

UNKNOWN SPEAKER: As a resolver operator, is it ethically right to change the time to live? From a technical point of view I can do it, but from the ethically right to change --

ALAIN DURAND: This is not a matter of being ethically right here, this is a matter of are you going to create pain if you do it? And the thing is you will create pain for yourself if you do it, and then the server has changed on the other side.

UNKNOWN SPEAKER: With the amount of the rate of the number of attacks for example, what happened with authoritative server providers early this year, we faced a huge outage and some authoritative DNS's, so if we had this mechanism to increase the time to live, that would be less pain for the end users.

ALAIN DURAND: If you manage it yourself and you know exactly why you're doing it, I think it makes sense, but -- my friend, Dave, has a comment.

DAVE KISSOONDOYAL: I'm Dave Kissoondoyal from ICANN. Possibly a different way to try to solve the problem that you're talking about is to deploy an Anycast environment for your authoritatives so that you could distribute your service.

UNKNOWN SPEAKER: I'm not an authoritative provider, I am a resolver, so if the authoritative does not deploy Anycast, my users who use my resolver will feel the pain of that authoritative going down, and the problem is sometimes the end user does not understand the structure of the DNS, so he will not imagine that the authoritative is down; he will think that the resolver has a problem, so in the end he will blame me and the end user will

never understand it's the authoritative. Maybe later on they will announce in the media, but maybe not, so I will be blamed for something I'm not able to control.

DAVE KIDDOONDOYAL: Yeah, well welcome to the internet.

STEVE CONTE: Thank you, Alain and Dave for that. You're right , you're stuck in the middle between the consumer and the person who offers the service, and one of the things that you can do, especially if it's the short TTL on the authoritative side, is you could always reach out to the administrator on that domain, assuming that they're nice people and playing the game and they're not malicious encounters, which is my pointless plug to Dave's next session on DNS Abuse at 10:30, will probably go into some form of malicious encounters on that.

Typically, if the authoritative server is there, you can do a Whois or find out some kind of contact point at some point to hopefully get to a person who manages that zone and say, "Hey, your TTL is really small or short, is there a reason why and can you lengthen it," and have that conversation. I think personally, a better way to go, if you can, to have that conversation than to arbitrarily decide to lengthen the TTL yourself because you don't

have all the information. If you have a discussion with the administrator, you might get more information and have a better understanding in that.

We're down to 15 minutes and I've got 15 slides yet left, so I'm going to blow through these. There was some good conversation and I want to have some more, and I don't want to just make this a one-sided thing. I'm going to go through the rest of them pretty fast, but stop me and slow down if we need to because I want to have questions, we have some good resources in the room here and have some dialog, that's part of what the purpose of today is.

Next slide. This is the CNAME, the alias. This is one of the only places -- again, we're inside the zone file, the RR Data Types. Normally, it's going to be label and an IP address of some kind. CNAME is one of the few things where you can actually attribute a label to a label, so if I only had one server, but that server is going to be www.example.com and ftp.example.com, then I could set an A Record for example.com or www.example.com, and I could set a CNAME to -- this is the CNAME on this site, this could be ftp.example.com pointing to the CNAME, the alias on that.

It's just a method to not have to do a lot of work. You set up one anchor record and you could set an alias up for multiple services or multiple labels to that one service, to that one A Record.

Next slide. Mail Routing, everyone does email. Oh, we have a question, hold on. Go back one slide, please. Wait, I have remote users and we're recording, so you've got to use the microphone.

UNKNOWN SPEAKER: For the CNAME, can you have the record belonging to another zone all together? Like instead of example.com, maybe some other website.

STEVE CONTE: I think so. Dave's nodding yes at me, I haven't done that for a while, hold on. Everyone should have a mic.

DAVE KISSOONDOYAL: Yes, and this is actually how blog sites like Typepad allow you to have a vanity domain. For example, my blog is securityskeptic.com, but it's CNAME is Securityskeptic.typepad.com, so if you go into dig or ns lookup and you look at my records, you'll see a CNAME exactly like what you're describing.

STEVE CONTE: Thanks, Dave. Dave, I'm going to keep that mic next to you because you're my authoritative source here today.

Next slide, please, Cathy. Email, everyone uses email and this is actually a great example of what Dave just said too, is that back in the older days you would manage your own email services. I would probably hesitate to ask, but how many still manage their own email services versus you have outlook.com or something like that behind, or are using Outlook services? No one uses email? Okay, then we can just skip this part.

No, a lot of times there's cloud services out there or ASP's or whatever where you might not be managing your own services anymore, you might be outsourcing. Like in this case, email might be using Outlook or might be using some other service, but you still want it to be in your domain. You can go and point your MX Record, your Mail Exchange Record to either the server that you might be managing yourself, mail.example.com or it could actually be the CNAME on that, or you could be pointing it to an outside service, you know, ICANN.exchange.com or something like that, if you're using an outside service.

You can have multiple entries for MX, for the Main Exchange record, and they have something called a waiting. What it's

going to do it's going to try the lowest number first, and if it can't to that number it's going to go up into that number, so what you're building is a redundancy in your mail system. You can have multiple email servers being willing to act on your behalf and it's going to go through a waiting system to say, "Well, I'm going to try this one first, but then I'm going to try this one if I don't get a response," and so it will go through this in a sequential level.

Next slide. Reverse mapping, we've been talking this whole time about names to numbers, but there's also a way to go from numbers to names, so if you're getting an IP address and you want to know what's the related domain or label for that IP address, you could do a Whois, you could do host name, there's various applications and commands out there that you could try to find the name of the IP address, and they do that by having a different zone called an in-addr.arpa zone; don't worry about the next two minutes that I'm going to talk about because you won't ever see this unless you're an operator of a network.

There's another sub zone in there and you basically put all your IP addresses that you're responsible for, if you want, and you associate them with either a broad domain name or you can have them down to a host level, so if you are using like a cable modem service or something like that, if you go to WhatsMyIP or

What Is My Hostname, you could probably go and see that there's a reverse name associated with the IP address on the IP address that was assigned to you.

For the most part, a consumer, an end user will not have to worry about looking up reverse names and there's no mandate that says as an operator that you have to set up reverse names, it's a nice to have, it's really handy from an operators perspective to say, you know, "Something's coming from this IP address, I'd really like to see what that IP address is, and it'd be nice to have that set up," but there's no mandate that you have to do that, and there's no mandate that it's authoritative either.

If I'm managing this IP space, 192.0.2, I could put anything I want as the label, there's no mandate that says it has to be within the specific zone that I manage. It's kind of more of a convenience than authority, you shouldn't ever rely on the data that's in a PTR, you should more rely about the Whois record on the -- who it was assigned to and then you go from that route.

Again, you probably won't ever see PTR stuff unless you're an operator but there is a methodology to go not only from name to number but from number to name.

Next slide. DNSSEC, I'll go very quickly on this, there's a session on DNSSEC at every ICANN meeting and I would recommend if

you want to know more in depth about DNSSEC, that's the place to go. What DNSSEC is basically, is it builds a chain of trust between parent and child and parent and child and parent and child all the way down. It stands for DNS Security, but think about DNSSEC as DNS Authentication. It's not providing what traditionally is known as security mechanisms, it's not doing any kind of encryption or anything like that, it's using the PKI, the Public Key Infrastructure to create key hashes between the two entities, between the parent and the child.

Those hashes build what's called a chain of trust, so at the consumer level, if you have an application that is DNSSEC aware, and that's really the key, and someone is trying to spoof something, they don't have that key hash, they don't have that chain of trust built, it will come back and give you an error. We're trying to get that more and more globally accepted, there's a lot of infrastructure for DNSSEC is in place, it's signed at the root.

You guys might have recently heard that ICANN and Verisign were about to roll the key signing key, the KSK on the internet, but we delayed it because we found some problems. There's an actual session this week about the KSK and the key role in all that, and I suggest you guys go to that, it's actually pretty interesting.

There's infrastructure in place for DNSSEC, there's many of the zones, many of the TLD's are signed, both ccTLD's and generic TLD's are signed, many of the domains are signed these days, but not all of them. The infrastructure is being built and is by large in place. There is still the heavy need for the application to become DNSSEC aware. A lot of applications that you use don't know anything about DNSSEC, so until the full product is built, I don't think the end user is going to see it as much as they probably should or want to.

And then there's the whole question of user education; you know, if my mom's surfing the internet and assuming she's got DNSSEC aware application, and she gets an error or some kind of message that pops up that says, "This isn't trusted," she needs as a consumer to understand what that message means. We can build all the locks and security in the world, but if the consumer doesn't understand what those locks and security does, than there's still a lot more work to go behind that.

I urge you to attend the DNSSEC sessions, I urge you to attend the KSK session, the key signing key session; I don't know what day it is, I want to say Wednesday this week, but it's really interesting, there's a lot smarter people than I am who are there who will be talking about the aspects of DSNSEC and the

purpose of delaying the key and the implications of delaying the key roll around what just took place, too.

Next slide. I told you I'd show you a sample of a zone file; this again, is in the presentation, feel free to download it, basically it's just a text file. We have label, we have RR type, we have data. Here we have label, we have SOA as our RR Type. We have some data; we have the NS Records, we have A Records, Quad A, MX, CNAME. This is typically what you'll see if you are a zone administrator.

Again, times have changed, not everything uses a flat file, and ASCII file anymore; there's some DNS serves that are more of a database type situation, but for the most part, if you can navigate through this ASCII file, you can navigate through most name server applications.

Next -- yeah, Dave.

DAVE KISSOONDOYAL: Just real quickly. The zone files for top level domains are slightly different than the zone file for a delegated name like this. In the top-level domains there is a restricted number of resource record types that you can have, and the same is true for the root name servers and the root zone, so it's interesting to actually go

and get a copy of the root zone so you can see some of those differences.

STEVE CONTE: Thanks, Dave. I'm down to five minutes, four minutes left, and the rest of it is kind of recap stuff. I wanted to point to this page as mostly a plug for the RSSAC session this afternoon, but I did want to leave some more time for dialog. Any other questions or comments, I welcome them, we've got a couple of minutes and let's talk. Anyone? You're going to make me show you more slides, do you really want me to do that?

Alright. So what we have here is the root servers, we have 12 organizations that run 13 letters of the root server. There's something called Anycast, which has allowed us to go beyond 13 instances, which means 13 servers, we can now have hundreds of servers. Anycast is a technology that basically fools the routing tables on the internet to make it think that it's all multiples of severs are all one server.

RSSAC will probably go into detail about that this afternoon, but we've gone past the magic number 13, we have hundreds of root servers out there which creates a very robust, reliable and stable root service.

Root-servers.org is their website. If you have any interest at all, head over there, it's an interesting place. It shows a map and it shows how many nodes, how many instances of those root servers are around the world, and you can see the geographical diversity on that.

Next slide. Root change process, this is a quick rundown on how a change happens within the IANA function. There's a relationship between the TLD manager and IANA and PTI, and then there's a process of databases that are run by the root zone maintainer; in this case it's Verisign as the RZM. They distribute it to the master server; remember we said that the master and the primary are not always the same?

In this case, the master is hidden and only the secondary's, every letter that you see out on the root servers these days are secondary's, they are the only ones who know and can touch the master, that's why they call it a hidden master. So, the distribution and synchronization is exactly the same as it used to be.

Go ahead to the next slide. These are the different types of players in the ecosystem; everyone knows of the registry, those are .com, .uk, .ae. Those are all different types of registries.

We have the registrar who is the agent between the customer and the person who is registering a domain and the registry, and then we have the registrant, which is the holder. In my case, I'm the registrant, I registered the domain conte.net. I used the registrar GoDaddy to do that; I'm not plugging any particular registrar, I'm just telling you what my own service was.

I registered conte.net through GoDaddy, they're the agent between myself and .net, so they asked me for payment obviously, but also for my information and some key name server information and IP addressing for those name servers because they need to populate that and send that to the registry so that that NS record is in the parent and the child.

Next slide. And again, these slides are available online, this is kind of just the relationship between the registry of what I just said and the registrant. So here I am, the customer, I used a registrar to register my domain. Registrar is the agent to the registry, so they send that information via a protocol up to the registry. The registry puts that into their database, they do some processing in there because they also get Whois data, and then they build it into authoritative name servers, and then the consumer, the internet users, go through the recursive name server like we went through the whole process and they go to

the authoritative name servers of .net in my case and then they go down there.

So this is just the chain of how registration process versus resolution process intersect on each other. Looks like that's the last slide.

We still have negative one minute. Does anyone have any final questions or thoughts? Otherwise we have a 28-minute break and then we're going to have Dave Piscitello up talking about DNS Abuse. I highly recommend you stay, it's an interesting talk, and Dave does it all over the world, and it's a great talk. Thank you for your time.

[END OF TRANSCRIPTION]