HYDERABAD – How It Works: DNS Fundamentals
Friday, November 04, 2016 – 09:00 to 10:30 IST
ICANN57 | Hyderabad, India

STEVE CONTE: I want to welcome you guys, and thank you for getting here so early for the session on, we're going to do DNS introduction, DNS fundamentals, which I've just realized I've misnamed my slide. Okay. This is… If you're familiar with DNS at all, this is probably not the session you want to go to, so I don't want to waste your time.

This is really going to cover the basics and the fundamentals of the domain name system. So, I will pause and see if anyone is going to walk out the door.

All right, so if you know DNS, no heckling, because this is basic stuff. Go to next slide.

I am Steve Conte. I'm the director of programs for ICANN's office of the CTO. I've been involved with ICANN, started in 2002. I've been a network engineer, I've been, I headed up their IT department. I have run the ICANN meetings. I have worked at IANA. I became their chief security officer, and then I left and went to Internet Society for a couple of years, and ran their fellowship to the IETF program.

And then about three years ago I came back to ICANN. So, just a brief introduction of who I am, and why they actually decided to put me up in front of you all. We're going to talk today about DNS. DNS was really because, made because it's difficult for a human to remember a cluster of numbers.

So, if we're looking at IPv4, it's not so hard because there is only, the dotted quad, there is only four subsets of numbers you have to remember, but even that one, just trying to remember multiple servers and multiple machines, it gets more difficult. And when we go to IPv6, and we talk about 128 bits in the address space, in hexadecimal, then it becomes near impossible to remember.

So, back in the early days, it was pretty easy to do mapping between a label, a text label, and machine, because it was just a couple of machines on the internet at that point, and there was a centralized location that would handle the host names and the host dot text file. Go ahead to the next slide.

This was managed by SRI, the NIC, and it was all manually managed. So, you know, the hand full of servers out there, and this was before TLDs, the dot coms and dot nets and stuff, so it was a single label field that would go in there. So, you would have a signal name for a machine and you would submit it to SRI, and they would put it in, and that was it.

And now you had mapping. And then SRI, you would have to go use an operative of that machine, or a host machine would have to go and download that file every week. And it was working, but it was not really that scalable. Go ahead.

Because it was made by hand, and because it was kind of, you know, requests were made by email and on the fly and stuff, there was naming contentions. And because it was a single label, also, if I wanted a host name named Steve, and somebody out there named Steve also wanted a host name Steve, it kind of became whoever grabbed it first won, and got that host name.

So, it wasn't great as far as scalability goes. It wasn't great as far as, because it was manually entered and manually managed, you know, you'd have issues with typos and things like that. So it was very difficult as the internet started to grow, to maintain that.

We also had problems with synchronization, because it was manually downloaded by the person who wanted to have that host dot TXT file, there was no to ensure that there was a synchronicity between the different machines. So, some machines had a more recent version of the text, of the host files.

Some machines had an older version of the host files, and there was no way to guarantee the same data was in every single file. And there was a traffic and load balance issue too. Don't forget,

this was the early days of the internet. Dial up was still the thing. We are talking probably 1200 baud modems.

Some people, if they were really well… Some institutions, if they were really well off, they had X11 lines or stuff like that. So the bandwidth really was an issue back then, unlike today where we have broadband going into the house, and you can download gigabyte files in less than an hour.

So it became quite clear that an essentially maintained file to map names to numbers was unmanageable. And especially as the internet was starting to grow, it was becoming more and more difficult to do that. So go to the next slide.

So the sort of discussions in the early 80s, they being the technical community, early predecessors to what became the IETF, the Internet Engineering Taskforce, started discussions in the early 80s to talk about a replacement for the host file, and for the host file system. And that basically, they wanted to address the scaling issue. They wanted to simplify email routing. So before, when you had the one label, if you wanted to send an email, you sent it to user at host, and that was it, but there was only one machine, so that machine was pretty much…

The assumption was made that that machine was also handling email services, and maybe gofer or other services at that time. But they wanted to disconnect the concept of email from the

DNS itself, or from the host file itself. They wanted to be able to have robust email that might not necessarily be handled by a particular server, and might not necessarily be handled within that last cluster, or within that name space too.

So they were looking at that. And so the result was the domain name system, the DNS. I'm showing two ROCs here. There is a multitude of ROCs around the DNS since then. It has continued to be worked on, almost every single IETF. There is more meetings about DNS, about how to make it more robust, or to work on the DNSSEC, or other pieces of DNS.

So it's one of our oldest protocols. It works at a core level for the internet, and it's also, it continues to be worked on, and continues to be modified, and so it's still pretty important, arguably speaking, it's so very important, at least to us because our whole industry is DNS, but to the internet itself. Go ahead.

So what they did instead of having a centrally maintained location for management of the resource files, they decided a distributed database was a way to go. So we would give… You know, there would be a core piece that would be managed, and then we would delegate management of name space to other people.

And that became easier to manage because you're managing a smaller cluster of names within that zone, or within that name

space. A couple of definitions. When we talk about resolvers, those are the machines that send queries. So any time you type in something on your laptop, or something in your telephone, or whatever, it's sending a call, if you're using Safari, or Chrome, or whatever, Firefox, it's sending a call to a resolver DNS application on your device, and then application is working on your behalf and is asking the queries of the mapping from the name that you typed in, to get the IP address that your machine is wanting to go to.

Name servers answer queries. So those are servers out there that will look and listen for the queries from the resolvers, and they will answer the query to the best of its knowledge. And I say it that way because there is other things that we'll go into. Also, really quick, I'm a very non-formal speaker.

If you have questions, please ask them on the fly. I'm not going to have a question and answer period. I'd rather answer the questions right away and have a dialogue rather than have you guys listen to me for the next 90 minutes.

Some of this also, we're looking at ways to improve performance. So name servers and resolvers, they do something called caching, and they hold an answer for a certain period of time, that's definable by a number of metrics, and we'll go into that as well as we talk about DNS today.

And we're looking for replication and redundancy. So the internet… The old story was that the internet was built back to withstand nuclear war and stuff like that, so you could lose portions of the network, and yet the network would still function, and would still operate.

The built the DNS in the same resilient model. They wanted to make sure that if pieces of the DNS stopped functioning on the network, on the internet, that the whole DNS system wouldn't go down. So we might lose portions of it, we might lose some namespace for a while, but DNS in general will keep going.

And so all of the other namespace will keep operating. And part of that is because of the distributed database model, and some of that is the distributed and replication model of the DNS servers themselves. So if we look at the components at a glance, I mention that every device has some kind of resolver, some kind of DNS mechanism on there.

In today's cases, we'll be using a telephone in our examples. They could be your laptop. It could be your refrigerator. It could be whatever device is connected to the internet. It has some sort of DNS built into it. We talk about recursive name servers, and in this instance, you notice that the recursive name servers actually doing both. So if we look at this from a model of, you being at home.

Your stub resolver is on your phone, your laptop, your recursive name server would be in your ISP, or in the case of the conference here, we have recursive name servers physically here on the site that answer DNS queries. But you'll notice it also acts as a resolver, because it's also acting on your behalf, on your, much like your stub resolver, the recursive name server is also proxy your query, because it's going to refer that outward, and we'll go into details on that.

And then we have various authoritative name servers. And we'll discuss the term authoritative, and we'll discuss why they're different then a recursive name server as well. Go ahead, next slide.

So, we call the database structure an inverted tree, because if you look at a tree, and it grows up, it grows branches, and it gets wider and wider, but we're going to take the root, oddly enough, and we're going to dangle it upside down from the root. And so now, we have different name spaces within each branch of the tree.

We do have the root, which is the top of the tree. And you've heard root name servers, and we'll go in detail about that. There is a presentation this afternoon from the root server stability advisory committee. They'll talk more about the root servers as well. We have top level domains, TLDs, and these can be dot

com, dot net, dot ccTLDs, such as dot UK. We have a gentleman from dot PR here.

It could be IDN, Internationalized Domain Names. So this domain at the left, the X, M, dash, dash, J6W whatever, that's a mapping to dot HK's IDN, Hong Kong's IDN of its two letter, two character namespace for Hong Kong, in Chinese. Then we have second level nodes, second level domains. These are the domains that you typically, as a user, those are the ones that you are going to register.

So, ICANN dot org, or for my case, Conte dot net, I have my own domain. Nike dot come. You know, those are the domains where you typically see, you'll typically surf to, if you're navigating the internet, or navigating the web. Those are the ones that you'll be registering as a user as well.

And then we have third level domains, and those are typically host names. So when you go to www dot ICANN dot org, www typically means that I'm going to, this is the host name, the actual machine I want to talk to. Now, there are some things around that, and as technology has increased, you know, more than one machine will answer the call to www, but at this level, that's typically a single machine, or a single cluster of machines that are answering to that.

Go to the next slide. Legal characters for labels, they call them LDH, letters, digits, and a hyphen. There is nothing else that can go into the domain space other than letters, ASCII letters, digits, zero through nine, and a hyphen. Everything else, and we look at IDN, it becomes much more challenging, but everything else is not allowed.

The at sign is not allowed because that's for email. The exclamation point, there was one company at one point, popular search engine, it starts with a Y and ends with an exclamation mark, they wanted to have the exclamation mark in the domain space, and we can't because that's a character that will parse differently, and it usually says leave this application, I'm going to run up a command outside of the application and then come back.

So because all of these different characters have different parsing mechanisms on the computer and with computer programming, they didn't want to have any of that in the namespace, so we allowed the hyphen and that was it.

Maximum link for any label is 63 characters, but that's per label. So I could have a 63 character, third level domain, so www www www, whatever, 63 characters dot, another 63 characters on top of that. Typically you don't do that because it's very, very hard to navigate, but it is possible.

And it's not case sensitive. So you might see a business card or a website out there, where they're capitalizing specific characters, and it's marketing, it's show, it is completely not case sensitive. You can type any domain in all uppercase, any domain in all lowercase, any domain in a variety of upper lowercase, if you're one of those weird, crazy people who like to do, you know, upper and lowercase throughout your whole sentence, it will still resolve to the same spot.

Every node has a domain name, and you know, I actually like this slide a lot, because if you look at any domain name, and in this case, we're looking at example dot com, and if you think about it, when you look at a domain name and you try to map it to a tree, and you're trying to say, well, which is the top level domain, which is the domain, second level, and which is the host name and stuff, if you take it from the end, where the dot is on the dot com there, on the example, and you kind of like teeter it down, suddenly you see the tree mark here.

So you see, www dot, the highest branch, lowest branch, you see example, you see com and then you see the root, dot. Now, a lot of times, you won't see the dot at the end, it's an implicit dot. If you type any domain in your URL, or in your browser, there is going to be a hidden dot at the end, because that's how the domain name system works. Go to the next slide.

If you put the dot, it's called a fully qualified domain name. That means the system, the DNS system itself says, you put a dot at the end, this means I know exactly what you're going for. This is… There is no ambiguity at all, what you're trying to surf to. So you can go to www dot ICANN dot org right now, and put a dot at the end, and you'll get to the same location as if you went to www dot ICANN dot org, but there is no ambiguity.

The system knows exactly what you're asking for. Now the implicit dot at the end, if you don't put it, it kind of inserts it there for you, to say we think you're going to this site, but since you didn't fully qualify it, we're just going to take, you know, the best guess. And you know, for the way that the DNS operates, the best guess is, 99.987 whatever percent correct in getting there.

But if you wanted to absolutely make sure you were going to the correct space, put that dot at the end. Go ahead. So a domain, is a node… So here we have dot com, a top level domain. A TLD, it's the apex of that, and the domain space is everything that falls inside that node. So in the case of dot com here, we see food dot com, bar dot com, example dot com, and then we see the host names, the third level, underneath that.

That all constitutes namespace, and in this case, the namespace for dot com. Go ahead. Name space is divided up to allow

distributed administration and maintenance. So even though, go back one slide. Even though dot com, this is the domain of dot com, VeriSign, who operates the dot com top level domain registry, doesn't manage the stuff underneath it. They don't manage example dot com, or bar dot com, or any of that, but they manage the pointers to that. And we'll go into some more detail on that. Go ahead to the next one.

So, because the name space is so large now, they distribute out the administration to it. So every apex is an administrator for the domain space, the name space, below it. And then they delegate, and that space is called zones, and then they delegate those out to other particulars as well.

So if we look at Conte dot net, my domain, we have the root, that delegates dot com to, in this case, VeriSign, I'm sorry, dot net, in this case to VeriSign. Dot net, I've registered Conte dot net, so dot net delegates the Conte dot net space to me, or to my administrators to administer and to maintain. So, anytime you're doing delegation, the person delegating is called the parent, and the person receiving that delegation is called the child on that relationship. Go ahead.

And go ahead to one more. So if we look at this, these are the boundaries of zones. So again, we talked about, the root has authority on some part, and then it delegates authority to other

aspects of the top level domains. So it relies on the operators of the top level domains to manage their zones, their namespace, and then those operators delegate down.

So for example, dot com delegates somebody else to manage example dot com, and it goes down from there. Go ahead. Delegation creates zones, I just said that, so let's skip this slide.

So as I said, name servers answer queries, and recursive servers ask the queries. So an authoritative name server is a server that answers queries for the zone in which it is built to handle. So I have my Conte dot net domain, and I have authoritative name servers, and I have two, because it's just my personal domain.

But those authoritative nameservers know everything there is to know about Conte dot net space. And they're called authoritative. So when a query comes in and you want to go to www dot Conte dot net, it will go through and it will finally get to my name servers, and the name servers will answer back with an answer, and they'll say, and I'm answering with authority.

And it will say, and that's just to tell everyone that there was no referral, no recursion or anything else taking place. This is the final say in who is managing the zone and what data is in that zone. Go back one. Did I…? Zones should have multiple authoritative servers. This is good, I mean, so there is a diversity question in that.

Part of it is you want to have multiple servers that are outside of its own network. You want to make sure that if you even have two name servers, that they're not sitting next to each other in the same network, because if that network is attacked, or that network goes down for any reason, you lose all your name servers.

By putting a single, even if you have two machines, and you put a single name server outside of your network space into a different network space, you've just increased your reliability tenfold on that. So if your network goes down, you still have another server out there on the internet that's answering queries for your domain.

A lot of places, especially when we're talking about TLDs and the root, and things that are much more important than my personal domain, they'll have name servers all over the world. We also use a technology called Any Cast, which we'll be talking about this afternoon in the RSSAC session, which allows you to basically mimic your nameservers globally, and give yourself a multiplier in the number of nameservers and location that they are. Go to the next one.

Busy slide. I hate busy slides. I'll see what this says and I'll say it in three words or less, well maybe not. So, we have a concept… So, we talked about the concept of parent versus child, and

that's within delegation of zones. So the parent is, in my case, Conte dot net, the parent is dot net, and the child is Conte.

When we talk about that kind of relationship with the servers themselves, it has to be one server out there, if I'm modifying my information within Conte dot net, there is one server that gets that information first. And then it will propagate that information out to other servers, that are handling that Conte dot net namespace.

The primary one, the first one that I touch where I put the information is, and that's called the primary server, also called the master, and then anything else that goes and talks to it as authoritative, is called a secondary server. So, you know, you might have heard on the root servers that a root is the master server, and you know, everyone else pulls their data from A.

Well, that's not true. A root is exactly equal to any of the other 12 letters out there, 12 root server letters, the master server in the root is actually a hidden master that is only accessible by the root servers themselves. So, and that's a good practice to have if you're running a business, or running a TLD, or whatever, you want to put your master somewhere that it's protected.

That not anyone can get to and touch. You want only the secondary servers to be able to touch that. That's a good security mechanism, because if the master is compromised, and

the data is changed within the master, all of the secondary ones are going to grab that data, and suddenly you have bad data in your zone.

So, having a hidden master, or a protected master, is a good practice to have.

The only one I'm going to touch on in this slide is the zone file. So, a zone file, for the most part, DNS applications, the final product is a flat file. A text file that then the DNS applications will use and load into their zone. That's called a zone. And you can manage your own zone file with a simple text editor, if you have a Microsoft product, you can use Notepad, and manage your zone file that way.

Easy when you have just a couple of hosts, like my Conte dot net, because I'm only managing a couple of machines, and I can go into a text file and it's very easy. But when you're managing a very large domain, like dot com. You know, millions and millions of entries in dot com. Having some poor fool sitting there, editing in a text file, finding the right line and not typo it, is going to be really hard.

So they have database applications that you can then manage your zone data through that application to the interface, and then when whatever button, when they say go, that converts it to the text file, then the DNS service pulls off of that. So if you

recall, every node has a domain name. A domain name can have different kinds of data associated with it. We kind of touched on this, and we'll go into details now.

The data store is called resource records. Resource records, go ahead to the next slide. A zone consists of multiple resource records, and we'll touch on that, those are ending from host names to mail exchange information, to name servers themselves. And there is something like 84 different resource record types that you can put into a zone.

Typically, in almost any zone, you'll see maybe four resource record types. But there is a plethora of things that you can put in. Go ahead. So a format, we're not going to go into too much detail on this. The format of a resource record is that you specify an owner, and that's the domain name of the resource record, which the resource record is associated with.

For example, dot com, we'll use, a poor example, we'll use example dot com. And the owner would be example dot com. Time to live, TTL, you'll hear that often when talking to DNS guys, that really effects the caching of the DNS, of the answer that's coming through that. And when we do the resolution portion of this, we'll talk about the caching and the TTL, but basically, from a zone file perspective, you're telling any kind of

caching server, recursive server, that this data is valid for X number of seconds.

The default is 86,400 seconds, which if you do the math, that comes out to 24 hours. But you can set the TTL to anything you want. If you're doing a lot of changes to a server, or changing server, you'll want to set the TTL for that server name to be very small, so that way it will expire in the caching server, and it will force a new query to happen. The longer you set the TTL for any specific resource, the longer it's going to sit in someone's cache.

So if you're making changes, let's say you set your TTL to a week for www dot example dot com, and your physical machine that runs the web server, for example, dot com, something happens, it goes bad, the hard drive fails or something, and you set up a new machine and you give it a new IP address, and why isn't anyone getting there?

Well, the cache around the world, everyone who has tried to get there before, is saying, this data is good for a week, so I don't need to ask that question again, I don't need to go through the whole recursive process of query and response process. So, I'm going to hold this data for a week.

Well, anyone who is holding that data for a week or less, at that point, is not going to go and know about your new machine. So that's why you kind of want to find a balance between setting

your TTL too low, which causes more and more queries to happen, that might not necessarily be relevant, balance between that and balancing between setting it too long, which would mean that no one would be able to get to it if you made changes. And I've got a question here.

NAOMI: Hi. My name is Naomi. And I'm wondering when, who is handling these resource records? Like, at what level?

STEVE CONTE: That's a great question. At the level of whoever the zone administrator is. So, there are multiple places that resource records are handled, and that's every… Go back, if you can find the slide about the zone boundaries? The ones with the dotted lines around everything. There. That's good.

Any time you see a dotted line, that's a zone boundary.

NAOMI: So, each one of those would manage resource records.

STEVE CONTE: Each one of those has a zone file, and within that zone file are various resource records, and they're managed within that boundary. So if we look at this, you know, there is management,

we'll talk about root management, that handles resources relating to referrals to the top level domains.

Dot com handles referrals relating to the third level. Dot example, example dot com rather, handles the resource records for anything, any space inside of example itself.

NAOMI: Thank you.

STEVE CONTE: Any other questions? Okay. Next. Next. Yeah, that's where we were.

Okay, so time to live, we'll go over, we'll talk about that in more detail when we do the recursion process. Class, back when DNS was formed, it wasn't clear that the internet was going to be the thing, it wasn't the internet, capital letters, it was just another network.

And so, there were other types, there were other class types as well. However, as the internet exploded and became more and more popular, the class type of IN, internet, became the primary one, and that's mostly what you see on the network today.

So it's there, but it's mostly unused and it's kind of just inferred. Now type is the type of resource record. And we'll have some

examples coming up in the next slide or two. This will tell you different types of, tell the servers what types of data they're asking for.

So if you're asking for an IP address, for IPv4, your type is going to be A. It's called an anchor record. If you're asking for an IPv6 address, you're going to get a quad A, four As, that just refers to IPv6. If you're looking for a mail exchange, and your SMTP server is trying to, someone's SMTP servers is trying to communicate with yours, they're looking, you're asking the DNS for a MX record, and that's asking the DNS which I should be talking to, to deliver mail and have that mail exchange.

And then the R data, which is actually the data that you're assigning, the data in which you're assigning the label to itself. Go ahead to the next slide. Can you guys still hear me? The sound changed. Did we? Test, test. Test. We lost the speaker here, no sound.

So, what we have here, I'll try to speak a little bit louder for you guys here. What we have here is just a small example of format, anything in the brackets are optional. So inside a zone, if you don't put the owner, if you don't put a TTL, if you don't put a class, it's going to inherit some default values when you build that zone file.

And the type, and the data, always has to appear. And we'll have some examples of that too. Go ahead. So, here are some common resource record types. This is what you'll see, barring DNSSEC, this is the most common ones you'll see inside a zone file. As I mentioned, there is A records, anchor records, which is IPv4 address space. We have quad A, which is IPv6 address space.

We have SOA, start of authority, or source of authority. This kind of defines to other operators, some key characteristics of the zone file itself, and also points to who manages that zone. We'll have a sample zone file coming up soon, I'll show you that.

We have C name, C name is an alias to another domain, within your space. So if you set www as an anchor… We'll see an example in a second, but it would be www, which is your A record, you will have A because you're telling me that this label is an anchor to this IP address, 192.0.27.2, or whatever we have here on these slides.

And now I wanted to say, www three, but I want to use the same server instead of setting up another A record, which could cause some ambiguity, would set up a C name, a canonical name, or an alias to it, and would say, www three C name, and then we would put www here, and that's just basically saying that www three is also known as www. It's just an alias.

You use that if you're a small business, and you have one server, and that server is running your web space, your email, and maybe your FTP, instead of setting up three A records with the same IP address, you'd set up one A record, whatever primary is called www. You would set up an alias for mail dot, and an alias for FTP dot.

I still have no sound on this side. Okay. Go ahead. The other important one is MX, mail and exchange record, and we'll talk about that as well in the next coming slides.

All right, we're having technical difficulties here. It seems that online we went to the next slide, but not here.

All right, so as I said, there are 84 resource record types out there. There is a list of them at IANA, you can go if you have any interest at all. Here is a horrible slide to put up in a presentation. That's a picture of the IANA slides, or the IANA website that shows the different ones. And I have no connection on this one.

And I've got my audio back, though.

I'm sorry, give me one second.

Sorry, we had a network drop.

All right, this machine has got to be rebooted. That's how we fixed it last time. I'm going to talk without slides for a minute,

until we get that fixed. We have everything breaking. It's like you're turning 50 and suddenly everything doesn't work.

So we were talking about resource records. And you'll have, there is 84 types, honestly don't care, you shouldn't care about what those 84 types are, if you have any interest at all, go to IANA, search for resource record types. We just do a visual on where we are.

All right, so we're just going to go into delegation, or to the process of resolution, looking for a domain name, and how to resolve to that, because I can do that without slides. So, let's say, let's make some assumptions. Let's say you're on a network, and you're on a machine, and you've just turned this machine on for the very first time, and the network itself, the name servers there… I lost again. Okay.

The name servers themselves… Can you guys hear me?

We're having so many problems with things.

Testing, testing, okay, excellent. I'll still have a voice by the end of the day now. So, it doesn't know anything about that, except it knows how to get to a root server. So in this case, it's going to send it to K root, which is operated by RIPE NCC in the Netherlands. And so, it goes to the RIPE NCC, to the root server

there, and says, hey, I want to go… I want to get the IP address for www dot example dot com.

And the root server says, I've no clue. I have no idea how to get there, but I can refer you to the name servers for dot com. So it sends back the IP address list for all of the name servers for dot com, back to your recursive name server. Your recursive name server then takes those addresses and picks one, and goes to the dot com address name servers, and says hey, I want to get to www dot example dot com. Can you give me the IP address?

And dot com name server says, you know, I don't know, but I know the address to the example dot com name servers, let me refer you to that. So it sends the list of example dot com name servers, back to your recursive server. Now, your server goes to the example dot com name servers, and says, hey, I need the IP address. I want to go to www dot example dot com.

And example dot com server says, I know that answer. Here is the IP address for it. It goes back to your name server. It goes from your name server back to your machine, and then plugs it in, and your machine will talk directly to the IP address for www dot example dot com. That's called a referral system. It's all… It's kind of like going to a business meeting, or a business party, and it's all about who you know.

You know, if you're trying to get to someone, and you don't know them, hopefully you have a friend, who knows a friend, who knows a friend, who knows that person. So you start working through that process of, hey, how are you doing? I want to go meet that person. I don't know that person, but I know this person, and starts moving down the line.

It's exactly what DNS is. In a nutshell, that's really what DNS is. It's how you know and how to get there. If we can get to the recursion process, we're going to bounce around, I think, today. Sorry, give me one second to figure out where we are. While we're finding that, is there any questions about the resolution process at all?

Does it make sense? I don't know how to get there, so I'm going to ask somebody else. That might have a piece of that information, they're going to refer me to another one.

All right. So this is what we're talking about here. We're looking for www example dot com, they went to your recursive name server. Recursive name server says, I have no idea. So I'm going to go to the root server. The root server says, I have no idea, but I can point you to dot com, to the TTLD. So, the recursive name server goes to the TLD server and asks them, they have no idea, but they know where there name server, for example, dot com.

[Inaudible] says back up, and comes back to example dot com, and example dot com gives you the address. So, that's a pretty complicated process, and it takes a whopping, you know 42 milliseconds to get all of that information out, which you know, is short in the breath of a human, it takes a long time for a computer. So how can we make that faster? Go to the next slide from here.

And we go that, we talk about caching. So caching speeds up resolution process, because now there are pieces of this referral process, that we don't have to ask again, until the TTL expires, until the data is no longer fresh. So after the previous query that we just did, our recursive name servers already have the address to the dot com servers, it already had the address to the dot example servers, and it has the address for www, the IP address for example dot com.

So let's say now, go ahead to the next slide. Let's say now, we want to talk about a different server in the example name space. We want to go to FTP dot example dot com, and so let's start the whole process, go to the next slide. And we'll ask the question, hey, I want to go to FTP dot example dot com, so we're in the same namespace, but we're in a different, we're looking for a different host.

And so now, you're in recursive name server, go to the next slide. It says, I know how to get to example dot com already, to the name servers there. So, I don't need to go and ask a root server, and I don't need to go to ask a TLD server, because I already have that information in my cache. So now, I'm going to go directly to the name servers for example dot com, and I'm going to ask the question, I need the IP address for FTP dot example dot com.

Next slide. And it responds and says here is the address. Next slide. And it goes down, it gives you the address. Much faster because you're taking out two steps of query response. So let's take this same example, we'll stay on this slide, and now we're looking for, stay right there. Wherever. Either one of those slides. Don't move.

We're looking… We're done with example dot com, and now we're going to go for the latest, and look for the latest footwear, and we want to go to Nike dot com. And www dot Nike dot com, and if we go through this, we already have cache in our recursive name server, because we've already asked some questions.

So now if we're going to Nike dot com, www dot Nike dot com, we don't need to ask a root server that question, because we already have the information of the dot com servers. So we're going to skip one step, instead because we have that

information in the cache, we're going to jump down to the TLD server, to dot com name servers, and ask the question, hey, I want to get to www dot Nike dot com.

They say, I don't know, but here are the name servers for Nike dot com, go ask them. And so, we'll replace example down at the bottom with Nike, and so, we still have to more resolution, but we're skipping a step because we have the cache for a root server already, so we know how to get to the dot com servers. Does that make sense? Everyone? Any questions?

I've got people online and we're recording.

UNKNOWN SPEAKER: Why is the first name server called recursive?

STEVE CONTE: So, it's recursive because, and we'll actually have a slide on that too, it's a great question, because it's not authoritative, is the answer. Recursive basically means, I'm willing to give and accept referrals, and I'm not acting as an authority to the answer, or to the data that I'm serving itself.

Whereas all of the servers down here on the right hand side, those are all the authoritative. I'm telling you with authority, we'll start at the root. I'm telling you with authority, that the IP

addresses for the name servers for dot com are these. And I'm not referring, I'm not taking that from a referral, I'm taking that from my zone data.

Same with dot com. I'm telling you with authority, that the name server IP addresses for example dot com is this subset, and I'm telling you that with authority because I'm taking that out of my zone data, and likewise for the data within the example dot com. When you're recursive, you're taking it on someone else's authority. So when the answer finally comes back from example dot com into the recursive name server, it will say, here is the data, but I have no way to tell you, at all, whether this data was good or not.

I can't tell you if there was a man in the middle who was poisoning my data, or anything else, this was the data that was given to me, I'm going to give it to you, use at your own, you know, risk. And we'll talk about DNSSEC and how that modifies that a little bit, and that's a good lead-in into that section. So, thank you.

Any other questions? Yes.

UNKNOWN SPEAKER: So how he selects the [inaudible] servers, when first time [inaudible] dot com, and as he explained in the [inaudible]

name, when there is no cache here, so how it selects root server from [inaudible]?

STEVE CONTE: So how does it select which root server to talk to the first time? It handles that… It depends on what DNS system you're using. Mine will select in a various different way than NSD does, and it's a combination of, basically going that one, closing your eyes and picking. It's a combination of doing a quick, they call it priming query, they'll go and send a query out to all 13 root server letters, whichever one answers first will get a precedent, and it also depends on available bandwidth, and speed of the network, and things like that.

So, there is no sure-fire answer for that because it differs depending on who is building the DNS server, who is the programmer, and how they decided to build that algorithm. Okay? Any other questions? Okay. We're going, let's go next and see what's coming up.

I'm sorry. We went out of order here. Go ahead. All right, actually we'll stay on this one and then we'll go back to root server stuff. So we have three players, primary players, in internet space, and domain name space.

We have the registry. Registries are those who manage their zone file. In the case of the ICANN community, we have the root zone, which is a registry, and it has all of the data for the name servers pointing to the top level domains. We have the registries, which we talk about traditionally in the ICANN model. Those are dot com, dot net, dot org, dot UK, dot whatever.

Top level domains, we call those, in the ICANN world, we call those the registries. Those are the ones that manage the big TLD space. We have the registrar, and we're going to go quickly over the registrar, and then I'll come back to it. The registrar acts as the intermediary between what we're going to talk about next is the registrant and the registry. And then I'll jump back to registrar.

The registrant is me. The registrant is my mom. The registrant is your mom, whoever, who wants to go and register a domain. Typically, a registry, if we're looking at the gTLD space, some, if not most, ccTLD space, country code space, most registries won't speak directly to the registrar themselves, the registrant themselves. They have the registrar in between, acting as a proxy for the registrant.

This is done because there is different ways to put data into the registry, to the zone. And so, as a registrar, if you're handling data for, and able to sell domains for a particular TLD, you'll

have some kind of communication method with them that's known and standard and secured, in some ways, too. A lot of registries and registrars use a method called EPP, which is just a mechanism where one database can talk to another database, and send data in a predictable fashion to that.

Registrars also handle, typically, more than one registry. So going to a registrar is kind of like going to a market for some. And let's say I wanted to register Conte dot net, Conte dot org, Conte dot UK, and Conte dot IN, all at the same time, I might pick a registrar that can handle all of those, and sell all of those domains in one spot, that way I only have one place where I have to manage my domain space.

So registrars act on behalf of that. And if we go to the next slide, perfect, okay. We also have… Registrars can also be, also have resellers. So, when we talk about agreements, registrars have to have agreements with ICANN, resellers don't. Resellers have that agreement with a registrar, but they inherit the agreement to ICANN.

So even if I'm a reseller to Go Daddy, I have to follow the same rules that Go Daddy does, because Go Daddy has signed an agreement with ICANN about policies and how to handle things like that. So there… Resellers like a subset of registrar. So if we

look at the registry, and we put the pieces in together here, we have the registrant.

So if we look at the left side first, this is acquiring or managing your domain. We have the registrant, that's the person who registers the domain, who wants to have and manage that domain, so that name space. They will usually register that domain over HTTP via website, they'll go to a registrar, and they'll pay for a registration of that domain.

The registrar speaks to the registry, the top level domain, and they will do that through applications, like I said, through a protocol called EPP, or other protocols. They'll handle that registration data up to the registry. The registry will take that data, and delegate that registration out to the registrants. So they're saying, basically, you're putting all of your information into the registrar, your payment information, your user and your password, and your name server, because you need to have at least one name server.

Two is better, and more is better. So at some point, you're going to register your domain, and you're going to say, here are my name servers. The registrar will take that information, and they'll mark it to the registry. They have to bring it upward to the parent, and say, here is my name server. Here is the name

servers for example dot com, and the registry only has name server information.

The only RR type, the only record type in a registry is NS, name server, because they're basically going to… Their whole job as a registry is to refer recursive servers to go and speak to the one in which they've delegated to, administration to. Now, we have WHOIS and R-DAP as well, we're not going to talk about that at all.

There are different types of WHOIS, different heaviness. There is a thick WHOIS and a thin WHOIS. It depends… Depending on what type of WHOIS, the data might live at the registrar or the registry, so we're talking about only DNS here today. I'm not going to talk about WHOIS, but there is a question, so we'll see if this is a question about WHOIS. And I'll tell you again that we're not talking about it.

UNKNOWN SPEAKER: [Inaudible] stops… [Inaudible]…

STEVE CONTE: Say again?

UNKNOWN SPEAKER: [Inaudible]…

STEVE CONTE: Oh, cyber-squatting. Cyber-squatting is more of a concept than it is…

[SPEAKER OFF MICROPHONE]

Yeah, someone still registered it. So the fact that it's… It might be registered to somebody other than…

[SPEAKER OFF MICROPHONE]

Right. Or use like McDonald's or something like that. If I register, I'm not Steve McDonald, I'm Steve Conte. So if I register Steve McDonald, I'm sorry, McDonald dot com, and now the big fast food chain says, well, you're squatting. You've registered my domain, yeah, there is a policy issue around that.

[SEPAKER OFF MICROPHONE]

It's a policy problem. We're talking about the technology. So as far as the technology goes, someone registered the domain, it was properly handled it through there. Whether or not the domain was properly registered to the, or registered to the proper entity, is a policy issue.

Now, from a technology perspective, everything was right. There is no such thing as squatting from a technology perspective. It's in the records. It's maintained. They system,

the network doesn't care who the owner is, as long as the data is the same. So when we talk about ownership, then we talk about policy, and I'm not a policymaker.

I'm not about to talk about that aspect of it. So, there are sessions in here about those types of policy issues, and I would suggest looking at the schedule of events and see if you can look at… WHOIS is… There are a couple of sessions on WHOIS that might play into that. But as far as the technology itself, there is no such thing as cyber-squatting or domain squatting, or anything like that.

It's either registered or it's not. If it's registered, then there are name server referrals in the network, and it's going to work. And again, the network doesn't care who has registered, or who that network is pointing to. The network cares if there is data. If there is data, hey, it's working, we're good.

If there is no data, it's either an error, or it's a natural occurrence of no data. No one has registered that domain. And that's it. So, the policy is the next level up, and one I try not to touch. Go to the next slide. Is there any other questions at this moment? I have a half of a hand. Is that a hand? No? Okay.

So we're going to jump back to, give me a second here.

I apologize for jumping all over this deck, but we're going to talk about the root.

So we talked about how to start the resolution process if you have an empty cache and no data. We talked about the whole host file, the hints file, that every name server has. So it says, even if I have no information at all, I have a file called a hints file that will point me to the IP addresses of the root servers. Go to the next slide from there.

So, two organizations currently administer the zone's contents. We have ICANN, who acts as the IANA functions operator, and we have VeriSign which acts as the root zone maintainer. Before 35 days ago, we had another entity in there called, NTIA, US Department of Commerce. They had a role in this too, but that's changed.

Now we have solely the IANA functions operator, that is, has also links to the community as well. There are no mechanisms if you've been following the IANA transition process at all. I'm not going to talk about that, that's policy. Not my cup of tea. And there is the root zone maintainer, which is VeriSign.

What happens… We actually have a process on that, so I won't go into detail yet. Go to the next slide. We have 12 root server operators. So if you would have asked me 15 years ago, how

many root servers there are, I would say there is 13. And there is one letter for every server, and that's it.

We had 13 servers. And it was robust. That's pretty good. There are 13 servers, they're all serving the exact same data. If we lost L root for a couple of minutes, or a day, whatever, it actually is 12 other root servers who were handling the same information. And the hints file says if you can't get to one, well there are 12 other ones, pick another one and go there.

But we decided, or we figured out that 13 actual machines wasn't actually quite enough. Attacks were getting more robust, bandwidth was increasing. Broadband was increasing. You are getting high speed access into home networks. There was more and more bot nets out there, and so the attack vectors were becoming bigger and bigger. And even though 13 is a pretty robust number, it was still becoming clear that it might not scale, it might not be robust enough.

So the… And I'm going to be brief on this, but it will, if you have more interest in any cast, please come to this afternoon's session at 3:15, I think? They'll talk about any cast. The root servers will be here themselves. But what happened basically is the root servers devise a technology called any cast.

This allows, in a nutshell, allows you to mimic entirely the presence of your root server, same IP address, same everything

else, across different aspects of the internet. So you can have different locations on the internet. So, you might have one server. Back when I was running L root, it was 198.32.64.12.

We might have that one IP address and that was our one server, but if we wanted to put that out elsewhere, we would put that out in a different network, and we would tell the network, the routing system, that this is the same IP address, and the same routing and all of that.

So from a router's perspective, it just looks at things and says, I can get to that single IP address from multiple vectors. And so from a routing perspective, it only thought there was one server. That's super basic. Root servers will be here this afternoon, please come back for that, because they say it much more eloquently than I can.

Hey, I said I can. But what we have now in today's network, even though we still have 12 operators of root servers, now I say 12 because VeriSign runs, we have 13 letters, VeriSign runs letter A and letter J. Even though we have 12 operators of root servers, we have over 600 instances of root servers themselves out there.

So we've gone from 13 physical servers to over 600. So the robustness and the resiliency of this has increased, 10, 20, 100 fold on orders of magnitude. So, if we lose one machine, one instance of a single letter somewhere, there is still over 599

other instances out there that are willing and able to respond with authoritative data. This makes for an extremely robust network.

And the root ops, if you come to this afternoon's network, they'll show you how that robustness acts, and reacts, to such things as attacks and network flaps, and things like that. Go to the next slide?

Root servers has a website, the root ops have a… Root operators have a website. It's called root dash servers dot org. This is a great site to go to, just for pure interest. You can't see it on this slide, but all of these circles represent the number of instances in a specific location. And you can click on it and drill down and see how many instances.

So, some of them you might have one or two instances, like it looks like Iceland has two root server instances out there, but if you look at South America, I can see one looks like 27, there is, you know, locations out there that have multitudes of root server instances out there. And so if you go to root dash servers dot org, you can see where the various instances have been placed around the globe.

Most of the time, those instances are being placed in exchange points around the world, because the concept on that is that exchange points consolidates traffic. Hopefully, that local traffic

is staying local. That's one of the purposes of an exchange point, is that you don't actually necessarily want to leave your geographical boundaries.

So if you're [inaudible] between networks, an exchange point is a great way to do that without bouncing to another country, or another locale. Having a root server in that exchange point is great, because that means any type of query that actually hits the root server, doesn't have to leave, it doesn't have to long haul, back haul over the expensive networks.

It will stay in the local networks, so your speed is quicker, but it's also cheaper too. Go to the next slide. So now, we look at the root zone change process. What we have now is that we have a system in a IANA called the root zone management system, or ZMS. It's a website, it's a secure website.

If you're a top level domain, you would go in. if you need to change your name server, let me finish this process up and then I'll come to you. If you want to change a name server IP address, for instance, you would log in to the RZMS, you would submit your change, the IANA functions operator will go and verify that change.

One of the things that they verify is that, especially if you're putting in a new name server, they want to make sure that that name server is up and running, it's reachable, and it's serving

good data already. The root servers, root system doesn't want to put what they call lame delegations.

IP address in there that are either serving bad data, or no data. So they want to verify that process first. They will also verify that the person, or the entity making the request, is the entity that is allowed to make the request. So there is various verification authorization steps that take place before any kind of request goes to the maintainer.

Once it is authorized, it does go to the root zone maintainer, VeriSign in this case, for implementation. This gets put into their database. It then, I mention, you can edit a zone file with a text editor, but if you're looking at a root, or if you're looking at a TLD, it's difficult because there is a lot of entries. So it gets put into a database.

When the database is ready to be published, they hit the go button. And it makes a text file a root zone file. That text file is then published to a name server. In this case, they call it the root zone distribution server, but in this session, we called it the master. And specifically, the hidden master.

So that root zone distribution server is really just another name server, but the difference is that it can only have communications outside to the internet with the 13 IP addresses of the root server instances. So only those IP addresses can talk

to the hidden master. And so what happens is, it puts out a notification out to its secondary, all 13 letters, all 600 plus instances, are considered secondary, they're equal secondary.

They will serve the exact same data as one another. They all get a message from a hidden master, saying hey, I've got a change, come and get it. And they will go within seconds of each other, and come and pull the new data. We have a questions in the back.

UNKNOWN SPEAKER: [Inaudible]. If I'm installing a root server internet exchange. So, what should [inaudible] server should I prefer? If I am having a lesson instances in my country, like I'm only having K root servers, or yeah. Or some other… So I should go with the condition of like, [I've replaced?] using, or using the round trip [inaudible] on some kind of a concept.

STEVE CONTE: Let me rephrase that question and make sure I understand it. If you have an exchange point, and you want to bring a root server in, which one should you choose? Okay.

The answer to that is really whichever one works best for you. And I'll say it that way, and I'll explain it. The root server data is

exactly the same, right? So between K root and B root, the data, no difference. You up for a question? Okay.

UNKNOWN SPEAKER: My question is that, if I'm putting some root server exchange point, and I want to use, I want to reduce the latency. So which root server, technology should I go for?

STEVE CONTE: Okay. So, the…

[SPEAKER OFF MICROPHONE]

Right. So, I'm still getting to that question. So because the data itself is the same, and it doesn't matter from a query response perspective, what matters then is a financial perspective, different root servers, operators, have a different model in which to provide a root server to an instance. You'd have to, as an operator of that exchange point, you have to determine which one makes the most fiscal sense to you, economic sense to you.

Now, there is no good answer. Basically, if you can get a root server, any letter, multiple letters, in there, you're still benefitting your constituencies, your customers, on that IXP. You're benefitting the entire globe, because you're adding another root server to that. They're all… Come back to the

RSSAC session this afternoon, they'll tell you flat out, they're all committed to the better good of the internet.

They're out there. They act responsibly, they're meant to make sure that the data that they serve is exactly the data that should be served. So there is no one root server who is better than the other root servers. They're all… You know, the numbers of instances out there. But that's really just a question of finances.

As far as the instances that they have out there, all of them are professional and robust instances. So I can't tell you, even if I wasn't working for ICANN or anything like that, I couldn't tell you which one was better, because there is no one better. They're all very professional, they're all very serious when it comes to serving the global internet community.

Root server… I will tell you this, back in the days when I used to run L root, it's not a way to make money. It's a huge suck and a huge drain of financial resources to do it right, to do it robustly. The fact that, and because of the fact that there are different financial models that they offer to bring in a root server to ISP. Some try to recoup it in different ways, because the costs are significant.

So, I don't have a good answer for you. But I can say that whichever one, if you're doing this, whichever one you choose,

will be the right one, because they're all serious players. Any other questions while I'm in a pause mode?

Okay, go ahead to the next slide. So I'm going to jump one more time. We're going to briefly talk about DNS, and then I think we're in a good wrap-up section. DNSSEC.

So, I'm not going to talk a lot about DNSSEC. I am going to say today, at 5:00 in hall two, is a session specifically about the beginner's guide to DNSSEC, DNS security. They're going to give you way more than one slide. And then later on this week, there is also hands on tutorials about implementing and then selling DNSSEC. But I am going to give you at least a slide's worth of data. DNSSEC stands for DNS security extensions.

I like to think, actually, that it was misnamed. It's not about security. It's about authentication. It doesn't encrypt your data. It's not going to do anything to increase security of that data. It is going to authenticate the response to the query, the person who made the query. So I like to call DNS off, standalone. No one goes that path with me.

We have a question.

UNKNOWN SPEAKER: We have a question online from [inaudible]. Which level of security is deployed in root servers? And how does it ensure that this is safe from cyber attack?

STEVE CONTE: So, each root server letter is their own entity. Each root server letter handles the, it makes autonomous choices on what root server software they're going to use, and on what type of security they're going to put in. So there is no single answer to that question. I would say, re-ask it at the RSSAC session this afternoon, and you'll probably still get no answer.

Part of that is because we're talking about security. Good security is you don't talk about security. So, they clearly have security mechanisms in place, and each root server operator has their own mechanisms, but they're not going to go into detail about what is in place, and how, because if there is every a compromise to that security method, they don't want to broadcast that that's the method they're using.

They will also make choices on which DNS software they're using. Could be bind, could be NSD, could be other software too. Could become custom software. A lot of them will use multiple types, so one instance might be using bind, another instance might be using NSD. That's purposeful.

So in case there is a compromise on one of the applications, on bind, or NSD, or something like that, that they won't completely go down. They might lose a couple of instances, a couple of clusters, they'll bring those down, but for the most part, some of their root instance network is still operating.

Each… What was the second part of that question?

[SPEAKER OFF MICROPHONE]

How to ensure it's safe from cyberattack? The root is attacked every single day. The root is probably being attacked right now, as we speak. It's really, root attacks are mostly a question of denying service. If you compromise a single instance of the a root machine, you might compromise data for a second, on that instance, but it's only going to be only that instance when we're talking about 600 plus instances out there.

Compromising a single machine isn't going to benefit the attack very much at all. And then we'll jump back, actually, into DNSSEC because that's, we'll touch on that. So most attacks against the root seller system, is DDOS and denial of service attacks. And when you have something of machines of magnitude attacking the network, and you have 600 plus nodes on that, you're going to lose a couple of nodes.

You're going to saturate them, and you know, they're going to drop. But you're going to have a lot of other nodes out there, that will nod. You know, the ones that drop are basically, you know, we say in the US, we're taking it for the team. They're going to soak that attack, and accept that attack, and probably go down, but while they're being attacked on those couple of instances, a couple of those nodes, the rest of them are operating fine.

We had, there was a DNS attack against a rather large provider a couple of weeks ago. And you know, a couple of major websites had some connectivity issues. You couldn't get to them, or it was hit or miss whether you can get to them, but for the most part, the internet worked fine, because the way that any cast works, and the way that the network resiliency, when you have more than one DNS server.

And that's why the more DNS servers you have, up to a point, the better and the more resilient you are, because if one is being attacked, you still got others that will answer those questions. So, there is no way to ensure against attack from a root server. And even at a TLD level, there is no way to say, I am not going to get attacked.

And if you are, that's an invitation to get attacked. Really, what you're looking about is how to best use your resources to

distribute that attack, to soak it, and also maintain an order of excellence, a level of excellence to service the internet community. Any other questions on that?

Okay, let's jump into DNSSEC for a second, because one of the questions was, one of the items I mentioned is, what if an instance is compromised? What DNSSEC does, and like I said, it's more DNS off, DNS authentication, what it does is using cryptographic keys and key pairs, it builds a chain of trust between the root, and the domain that's being queried.

And the DNSSEC beginners, this afternoon at five, will go into a lot more detail on that, but the basics on this is that, parent holds part of the key, the child holds part of the key, on that key pair. When the key pairs are compared, they both say, yup, I've got this, and the child says, yup, I've got this. All right. We trust each other. We made that authenticated connection. That builds a link in the chain, and you look at the chain goes down to the host level.

So when I'm querying for www dot example dot com, and I am DNSSEC aware, that means that I am using software that, I understand that there is something called DNSSEC out there. And it's looking for specific flags from the DNS, to know whether or not that was a trusted response. So I'll ask the question, the

query response question, just like when we went through the process.

Do I know you? Do I get my referral? Get to the next level? Go down there, and I get my IP addresses, but along with every single response, I give you a key, and I give you a piece of the trust in the chain, and say, here is… If I'm a root, and I'm giving an answer for dot com, to the dot com servers.

I say, here is the dot com servers, and look at signed, and then you go and you compare that signature with the dot com servers, and its cryptographically generated, which means you can't just come up with another set of strings and say, yeah, look, it's signed. You know, it has to compare, and there is a hash and an algorithm behind that.

So I get through the resolution process, each answer has a signed response from there, and it builds the chain of trust. So when I finally get the answer to what the IP address is for www dot ICANN dot org, I'm sorry, example dot com, I can also look and say, look, it has been…

The chain of trust is built all the way from the root. So, I can be assured that the data I've asked for is coming from the location that I am assuming that I was expecting it to come from. Whether or not that data is good, is a different question, because there could be typos in the data. There could be other things in

there. But you know, even if you've got the bad data from a typo, you know you got bad data from the source.

Now if a machine was compromised, if we are looking at an attack on a root, and one of the instances was compromised, and someone is trying to put data in, they're breaking the cryptographic key, and there is no way they could generate, or re-generate to tell the parent that that's good data, or tell a child that's good data, even.

So if someone compromises a single instance on the root, on one of the root letters, and they put in dot Steve, because all of the Steves unite and we want to rule the world. There is no way we can sign that, there is no way we can tell the world that we're authenticated, and that the IANA has blessed the dot Steve domain that we're actually going to actually take over the world.

We just have to do it in other methods. Let me finish this thought, okay. So, DNSSEC works really well against compromises and attacks like that, and I think we're almost at a time… There is a couple of different records that we'll see. We talked about A records, quad A's, DNSSEC introduces other record types.

And I'm going to open up a couple of more minutes for questions. I really do apologize for the bouncing around today.

We had some technical difficulties, and I hope that this was helpful. We do have a question in the back.

While I'm walking back there, our next session will be at 11:00. And it will be on networking and routing and naming, and it will be from [inaudible], and it's a really good session and I recommend staying for that. Sir?

UNKNOWN SPEAKER: Thank you. There was [inaudible]… Is there a global body who maintains RR? Record [inaudible]?

STEVE CONTE: Anyone who… Record sets are maintained in the zone itself. So the zone operator, the zone maintainer, will put in resource record types. If you're trying to introduce a new record type, that has to go through probably an IETF process, a RFP, I'm sorry, a RFC, and has to get acknowledged by the community.

And I think I have another question here, and then I'm going to stop it because we are out of time.

UNKNOWN SPEAKER: What is the progress in different languages?

STEVE CONTE: DNS with different languages? So that's called IDN, internationalized domain names. Again, that's above the technology level, so and you're asking good questions, and the right questions. From a technology level, like I said, we can only have ASCII characters, digits, and hyphen. Which at one point, you saw XN, dash, dash, whatever, that's a mapping between scripts, different language scripts, and usage in the DNS. And there is applications out there called Puny Code, which does that mapping and figures out, you know, let's say we had Cyrillic in there, which is a different script.

There is a mapping out there that says, I'm going to convert the script using Puny Code and the coding set, into ASCII and numbers and the hyphen. And so that way, you know how to get to, that way the DNS system knows how to get to IDN domains. Okay? And the last question.

UNKNOWN SPEAKER: Where is the idea that we are depending on caching instead of creating [inaudible] whole registry data into each of the DNS servers and use them?

STEVE CONTE: I'm sorry, I'm not catching the question out of that.

UNKNOWN SPEAKER: If we are relating and querying into each IP address our DNS name, instead we put all of the data in registry into each of the DNS servers, and use it? Because we are [inaudible] relying on caching and [inaudible].

STEVE CONTE: Okay. So you're saying the entire zones from the TLDs, and you cache those, the entire data set. Now, for some TLDs, you can do that, and they allow that, and some, let me rephrase that. From some domains, from some zones you can do that, and they allow that. And some zones don't allow that. They call that DNS working. And some zones allow you to walk the entire zone and pull everything out, and some say, no, only ask the specific question, and I'll only give you the specific answer, and we won't give you all of the data.

So it really depends on managing that zone and how they set that up. All right, so we're past time. I'm getting the evil stink eye. I want to thank you all, first and foremost, for your patience as we struggled through these technical difficulties this morning.

And most importantly, for coming and joining me this morning for DNS. We are here all week, if you have any further questions, please feel free to find me in the hallway, I'm happy to answer anything I can, unless it's policy. I'm not going to talk about

policy. I'll talk technology only. But, thank you all, and please join us in 28 minutes for [inaudible] next session. Thanks.

[END OF TRANSCRIPTION]