ABU DHABI – How It Works: DNS Fundamentals
Monday, October 30, 2017 – 17:00 to 18:30 GST
ICANN60 | Abu Dhabi, United Arab Emirates

STEVE CONTE: All right. We're going to go ahead and get started then. We've got six people, so first of all come on up if u want. We'll treat this as an opportunity. I still feel weird about talking in a mic with six people, but we are recording it so I guess I have to.

This can be as short and sweet or as long and detailed as you guys want, so I'm really going to encourage questions. I can stop at any time, and we can talk. How many of you guys are tech sector? Wow, okay, more than half. That's great. I believe you said you were academic.

UNIDENTIFIED MALE: [inaudible]

STEVE CONTE: Okay, governance?

UNIDENTIFIED MALE: [inaudible]

STEVE CONTE: Okay. So the tech group, you understand that this is totally DNS 101. It's DNS Fundamentals. If you know about DNS, you're going to fall asleep. As exciting as I am and can be, you'll still fall asleep, so think about that. Those non-tech people, feel free to stop me at any time. It's really a fundamental class about understanding the underlying technology of what DNS is because it's what the rest of the week is going to be. People are going to be talking about the different aspects of DNS, so this is really all about what that is.

I'm Steve Conte. I'm the Director of Operations for the Office of the CTO at ICANN. I guess I'll give you a little bit of history because we have some time and some space. I started with ICANN in 2002. I worked in the IT department, ran their IT department, was the root server operator for L-root at one point along with my colleague John Crain. Then I left and went to Internet Society for a couple years. Then I came back and they said, "Great, now talk about everything." So here I am.

We're going to talk about DNS, Domain Name System. As most of you were in the last session with the root server operators, you'll see first of all this is backwards because we had a conflict. This was supposed to the first session of the day, but we had a conflict with the opening session. So it's the last session of the day because we have a conflict with the public forum and that works well.

A lot of what you saw with RSSAC is going to be covered here but maybe in a little bit more detail. I know Andrew was talking about the conversion of IP addressing to numbers, and that's really what we're going to be talking about. We're going to walk through step-by-step the process of resolution.

As Andrew had said before, IP addresses are easy for machines to remember or identify, but it's really hard for people to do it. There are not many opportunities or not many places where you would actually type an IP address into a web browser or into an FTP server or anything like that unless you had it written down somewhere because they're just hard to remember. It becomes much harder to remember when we look at IPv6. It's such an expanded number space there that it just becomes nearly impossible to remember a single address much less multiple addresses. People need to use names. It's how we remember things. It's easier than remembering numbers.

Way back in the early days of the Internet, it was pretty simple. There was a single file. There wasn't any such thing as a domain name yet. They were called hosts. There was a single file. There were machines that they decided to put labels on and translate them to numbers. We called them hostnames. It worked really well in the very beginning.

People would map these hostnames to their IP addresses, and that's called name resolution. It would have an application that would look at that HOST.TXT file, and it would do the resolution. It would say any time you want to talk about "Steve" you're really talking about this IP address.

It served the same function in a slightly different format than what we see today. At that point, it was a centrally maintained file. I don't even want to call it a database yet. It was maintained by the Stanford Research Institute (SRI). It worked pretty well. They would update it fairly regularly, once a week, and they would put out a new file for availability to download via FTP once a week.

But it was getting hard as the Internet grew more and more. This was even before it became a commercial enterprise. The Internet started as a U.S. Department of Defense project, DARPA project. It expanded out to universities. So even before it became commercially successful or interesting, it started growing more and more within the academic sphere. So it was [becoming] problems with various parts of this being centrally managed.

We had naming contention. Edits were made by hand. It was kind of first come, first serve. If someone else wanted the hostname "Steve," whoever got that e-mail in to SRI first kind of won.

Synchronization had challenges because it was reliant upon the consumer, the network administrator to go and grab via FTP the newest latest and greatest version of the file. So there was no guarantee of the versioning on it. Some people might have had an older version. Maybe the FTP broke. Maybe it was automated and no one caught that it didn't grab it. So there were issues with synchronization.

Traffic and load, which sounds kind of silly these days because we're talking about a very small text file, but remember back in the late '80s and early '90s that the main medium to get on the Internet. So as this text file grew, it became significant bandwidth both at SRI but at some of the consumers too to get that new file. Now we look at it and we're in the world of 75 Mbps at home and stuff and you think that takes 500 milliseconds. That's no problem. It was a problem back then.

The technical community at that time started to realize that a centrally maintained file and system just didn't scale. The Internet was starting to grow commercially, it was getting more and more interesting, and people were doing more and more things on it. They had to come up with a way to do it.

The technical community started a discussion back in the early ‘80s on a replacement. The goals on that were to address the HOST.TXT file scaling issues. That was the centrally managed file.

Also, now that more and more entities were joining the Internet at that point, there were some routing issues with e-mail and things like that. So they also wanted to simplify the mail routing on that.

Some of that also is that there were services – I don’t know, how many of you remember AOL or other services like that? CompuServe? Things like that? There were services that might have had a gateway to the Internet but weren’t necessarily on the Internet at that point. That caused some mail routing issues too because if I was on AOL and Cathy was on CompuServe, I couldn’t necessarily send her an e-mail because we were on two unique, diverse systems. When those systems became more and more prevalent in using the Internet gateway, they wanted to have ways that they could route mail back and forth.

The result was the Domain Name System (DNS). Here are a couple of the starting RFCs that were developed through the Internet Engineering Task Force (IETF). Is anyone familiar with the IETF here? A handful of people. This is a group of engineers that are located around the world. They volunteer to work on creating protocols and standards, open Internet standards on the Internet.

Steve Crocker, our Chairman of the Board here, started the series of – the product on this is called an Request for Comment (RFC). Back in 1969 when Steve Crocker was at UCLA, I believe – USC – they needed a way to track unique identifiers and stuff. So they started this RFC series documentation with RFC 1. It grew and grew, and in the '80s a body of engineers took that over called the IETF, the Internet Engineering Task Force, and they decided that they would be the home where the RFCs and the protocols once they were developed and the standards were built, that's where they would be kept.

The IETF started in about '86, so this was about the same time it was transitioning as well. These were some of the early RFCs on domain name system. There are a lot of them out there now. As the protocol has matured throughout the years, there have been new RFCs applied that would enhance or change the behavior of that protocol. If you ever go to IETF.org, for instance, you could probably find their RFC page and then do a search for DNS and you would get pages full of documentation on that.

These were the starting RFCs that described what DNS was and they started to use it. DNS is a distributed database, which means it's no longer centralized. As we saw in some of the other presentations, they call it a hierarchal database. Basically what it means is that there are a whole bunch of ASCII files out there and the root, the central point, and they start pointing to other file out there as well. It was a way that they could delegate maintenance and management of various pieces of the DNS system.

We'll have a visual on this soon, but Andrew talked about this a little bit in his slide deck too. Resolvers send queries, and nameservers answer queries. This is really what DNS is. Someone asks a question, and someone else answers that question. We'll go through that in a second, and I have a visual for the resolvers and the nameservers.

One of the other things they wanted to do is they wanted to work on optimization. Again, we're talking about low bandwidth days that the traffic going over the Internet was necessary traffic only. If I was asking a question about Nike.com and then I wanted to go ask a question about Adidas.com, it didn't need to go through the whole resolution process. They wanted to introduce caching because some of that information could be kept closer to the end user, closer to the query, and allow for faster response at the end user but also less traffic overall flowing on the network. Again, in today's Internet that kind of thing from a traffic perspective is less of an issue because we have such large amounts of bandwidth out there. But the caching still has a noticeable impact on the end user.

They wanted to also come up with replication to provide redundancy and load distribution. We talked about this in RSSAC. They introduced Anycasting. But even before Anycasting was a standard, it was enabled geographically and topographically diverse locations to be able to have nameservers out there that were serving the same data, allowing faster response times but more resiliency too in case a fiber was cut or a backhoe dug up a cable and stuff like that.

At a glance, this is the visual. If we look at the different types of DNS, we have what we call a stub resolver. This could be any end user machine. All your laptops that are open right now have some sort of stub resolver on them. At the very basic level, it's the one who asks the question.

In this case, it's a phone and it's attached to an application. In this case, Safari on an iPhone. It has a stub resolver. Any laptop has a stub resolver mostly built into the operating system, but occasionally it is overridden by additional stub resolvers that are on applications. Like Chrome comes packaged with a stub resolver as well, so it will use its own resolver to ask the questions.

Now in the middle, we have something called a recursive nameserver. This is going to be the nameserver that typically you'll get when you turn on your computer at your work or at your home. It automatically gives you an IP address. It's a protocol called DHCP. With that IP address, it also gives you a DNS server or maybe two or three of them. That's what the recursive nameserver is. It's the one that's going to sit in the middle of the process and act on your behalf in the resolution process.

So in some ways, it's going to be a nameserver. It's going to answer to you when you ask the question. But in some ways also, it's going to act on your behalf. It's going to be a proxy, and so it will be a resolver. It's going to ask a question as well but outward. So this is the guy in the middle. As the end user, you're going to ask a question, and he'll eventually answer it. So he's a nameserver at that point. But until he answers it, he's asking the question out to the Internet on your behalf. So we call that a recursive nameserver.

Then over here we have authoritative nameservers. Authoritative nameservers will only answer questions. They will never ask the question. These are the nameservers that are either in the root or the top-level domains - .com, .net, [inaudible], whatever. It could be the domain name itself: example.com would have to have a nameserver that's authoritative to answer within its sphere of knowledge as well.

They should never be able to ask a question on someone else's behalf. That's a misconfiguration. It happens occasionally because it's easy to misconfigure, but authoritative nameservers should only answer questions.

When we look at the namespace, and a lot of people call this an inverted tree and if you flip it upside down, you'll kind of see. We do call the very top-level the root, so we kind of see the tree format here. If we flip it around, we see the roots at the bottom and then we get branches coming up.

At the root, that's the part that IANA and PTI manage and distribute to the root server operators through the root zone maintainer process that has been developed through the IANA transition.

Next on that tree is the top-level domains, the top-level nodes. Those are the .com, .net, [inaudible]. We have something over here that looks kind of funky: xn--j6w193g. Does anyone know what that represents whenever you see something that starts out with xn--?

UNIDENTIFIED MALE: It's an IDN.

STEVE CONTE: It's an IDN, exactly, an Internationalized Domain Name. Typically, the end user will never see xn-- because IDN is a methodology in which it uses a non-ASCII script to be able to navigate the Internet be it on a website or some other way. So there's a piece of software in whatever application you're running that's going to do the conversion from xn-- to whatever language or script that this is actually representing. But it still qualifies as a top-level domain. The reason why it's in as that we'll go into in a slide or so, but the answer quickly is that the DNS can only handle ASCII characters.

You have a question? That was it? I'm going to give you a mic because we are recording.

UNIDENTIFIED MALE: So the IDNs are only international in the second instance and, in fact, they are still Roman script in terms of how they fit into the system?

STEVE CONTE: I don't want to call it a second instance. I want to call it a second layer. It's more when we looked at the OSI layer earlier, it's kind of like that. The DNS can only handle ASCII, but there's a place in between that called punycode which then can convert that ASCII into a different script that the application, Chrome or whatever, could handle.

Then what we have next on down that three is what typically everyone knows as the domain. If you go to your registrar GoDaddy or Network Solutions or whoever you're using and you register a domain name, in this case example.com, this is the domain name that most people refer to when they're talking.

You can also have additional layers below it. Most of the time, these would be hostnames. We see mail, we see www, whatever. Traditionally if you look at it, it would be host.domain.tld. That's your traditional thing. The very first part, the www or mail or whatever it is, typically refers to a machine or a service.

However, you can do multiple delegations. So, for instance, .uk has a layer instead of having their domain names here, they have a further delegation layer. They have co.uk and they have gov.uk. Then below that is where you as a consumer would actually start being able to register domain names on that. So you can have as many dots and as many layers as you want as long as you don't surpass some limitations, which we'll talk about in a second here.

In fact, we're going to start talking about it now. Legal characters, as I said, are only ASCII commonly known as "LDH" (letters, digits, and hyphen). Those are the only legal characters that will fit into the DNS system at the point of resolution: A-Z, 0-9, and the hyphen. Those are the only characters that the DNS knows how to operate around.

Label: that's a new term here. If you go and register a domain, it would be like example.com. But each point on the tree, we call each of those a label. This would be a label, this would be a label, this would be a label. It's just the way that we try to recognize the individual points within a fully qualified domain.

Maximum length of each label can be 63 characters. Maximum length of the entire string can be 253 characters. It cannot exceed that. In the last slide, I said you could have as many labels between dots as you want. The limitation is that it cannot exceed more than 253 characters as a total string name.

I'm sorry. Let me get you the mic. Total length is 253 characters. 253, yes. Yes, in fact, that's a good question because in most computers it's divisible by 8 and stuff, and that should be 64, right? Because 8x8x8 whatever. It's only 53 because we're taking the dot into account. So 253 is making some assumptions about how many dots you're going to have. You can go farther, but you have to make more assumptions on those dots.

UNIDENTIFIED MALE: [inaudible] which is the root server. Okay, is the first dot which is the root also counted along with it?

STEVE CONTE: Yes. We'll talk about that. The first dot is typically hidden when you type in a domain name, but it is there and it is a limitation of one character.

UNIDENTIFIED MALE: Sorry, this is really 101 for me. What is the difference between a label and a string?

STEVE CONTE: That's a great question actually, and thank you for that. A label is the individual part: example is a label, uk is a label. When we talk about a string, we talk about all of them put together. We string them together. So www.example.com would be the string. Again, most people call it the domain name. It's more acceptable or more widely known to be using that terminology, but within the DNS world we call it the label and the string so we have more preciseness in what we're discussing.

UNIDENTIFIED MALE: I would like to know how the 253 is arrived at. Is it just a convention or that's a limitation that [inaudible] to having that the maximum of a domain string cannot have 253?

STEVE CONTE: That's a great question, and I don't necessarily have an answer. I suspect it has to do with the fact that DNS runs traditionally over a protocol called UDP versus TCP. UDP is historically within a packet that's 255 bytes long, or maybe 512 – 512, I think. But it had to be able to fit into that small datagram packet. I suspect that the limitation are historically based around that, but I'm not entirely sure.

I actually appreciate that question. That's something that I will research so the next person who asks me that question I can answer with authority. I can look that up. If you try to find me this week, I'll see if I can get you an answer and something with authority more than "I don't know."

One other point on this slide is that DNS is not case sensitive. A lot of times when you're looking through a magazine and you see someone is trying to advertise something and they want their organization name inside the domain or inside the domain name that you'll type, the URL, they'll upper case their letters. If I was going to UnitedAirlines.com, I would see the upper case U and the upper case A. DNS doesn't care about that. It basically does a conversion and converts it all to lower case first before it does any kind of resolution. It's called case insensitivity on that.

Someone yesterday brought up the fact that IDN is absolutely case sensitive. But as we just discussed too, IDN happens at a layer above DNS. So that would take place in the punycode translation, but by the time it goes down to the DNS the xn-- whatever would still be case insensitive. It would be lower case for all intents and purposes.

Every node has a domain name. This is kind of just where we were talking about labels and strings. Again, a label, we use a lot of different words for the same language. A label is a single point in this, and a node is also a single point. "Label" is what we're calling it and "node" is just the positioning. You can swap the terminology out equally.

Sequence of labels from that node is what we call the domain name. In this highlighted one, we have starting at the bottom here "www." There's a barrier here that's a different node, ".example.com" and then the root on top which as you pointed out is there but most of the time it's hidden and just inherited in that.

A fully qualified domain name (we call this an FQDN) makes sure that where you're going there's no ambiguity on what you're asking the server to look up. For all of you with your laptops open right now, you could go to "www.google.com." and put that final dot on and it will absolutely respond back positively. But what it does is it removes all ambiguity. It tells the DNS system that this is where I want to stop looking. There's nothing else.

Let's say "google.co.uk" and we'll say "google.co," which is Colombia - .co is Colombia. If you were looking, once the recursion happened, it might start looking beyond the .co and start looking and say, oh, there's also a .co.uk and it might give you multiple answers or some ambiguity in what answer you're looking for. When you put that final dot in there, it fully qualifies it. It says, I don't care if there's anything else. This is what I want, and remove all that ambiguity out of that.

The dot is always there kind of. It's called the hidden master or hidden root. It will be assumed when you hit ENTER on your browser that you're putting a dot at the end. But you could absolutely tell it this is what I want by putting that dot at the end.

When we talk about domains, again, most people think domains are at this level of the labeling system here. But a domain really means in the DNS world you take the node or the label and anything underneath that that's attached to it is the domain. In this case, .com, we might have an example.com, we might have a bar.com, we might have a foo.com. They are all part of the .com domain.

What that really means is that within that domain space, the apex of that node, the top part of this node has some information about everything within its sphere. It might not know, because of delegation, it might not know the IP address for www down here, but it will know the IP address to get to the nameservers for example. So it's just creating boundaries of delegation, boundaries of responsibility.

Now zones are the space divided up to allow distributed administration. When we look at domains and what I just said is that the domain is the apex of that label, the top uppermost part of that label. It should and will know about everything within itself. When we talk about zones, we're talking about the same thing but in an administration perspective. Here's the namespace, and here are the administrative boundaries, for example, of this namespace.

The root is an administrative boundary. If we look at it from a domain perspective, it's the top. It's the apex. So it knows something about its whole space. We don't know exactly how much, but it has some responsibility of the entire space. It delegates through zones the next level there.

In the last example, we did com and the domain space was everything underneath that. But it can delegate that delegation or the administration to the next level below it. Then in that, you can have spots that it's authoritatively responsible for or it could further delegate that down. Like in the example of the co.uk, uk would be here and instead of looking like example here where it would be all this, it might have [co.co] underneath it and that would be a delegation so it would be a separate zone. It might have gov underneath it and it would be a separate zone. It's a point of delegation where someone else is going to maintain part of that domain space.

UNIDENTIFIED MALE: There is another example, .com.pk. Who is the administration because nodes are basically ccTLD and .com is basically TLD.

STEVE CONTE: I don't know the specifics on pk – pk is Pakistan, correct? I suspect it's much like uk. I know that pk is the top-level domain. You probably have a gov.pk as well then?

UNIDENTIFIED MALE: Yes, .com.us, .com.sa, .com.uae – this is a common phenomenon. So in this regard, who is the administrator: .us or .com?

STEVE CONTE: So .com.pk, .com.whatever is different than .com. That's just common language being spread across multiple TLDs. Who the administrator is on .com.pk is whoever pk wants to delegate that administration to. Whoever the administrator of co.uk – which is the same thing, they just took the m off – is whoever uk has decided to delegate that administration to. So there's no central com other than Verisign which is the top-level com. This is .com, and we look at this thing, this would be .pk up here and then com down here. So they would delegate that administration out to another body and it's not a standard body across the globe. Does that answer your question? Kind of?

UNIDENTIFIED MALE: A sort of clarification, [inaudible] call it labels, so each label is [inaudible], right?

STEVE CONTE: Each what?

UNIDENTIFIED MALE: Here you said that com which is top level then like the bar is a label.

STEVE CONTE: It's the next level down, yes.

UNIDENTIFIED MALE: Okay, which means the www is also a label, right?

STEVE CONTE: Everything on that is a label.

UNIDENTIFIED MALE: Okay. The question I asking is you've used the word label and used the word zone. I want to know the difference between the two. Are they synonymous? Can we call let's say now we have from com is delegated to bar, bar is a label, right?

STEVE CONTE: Yes.

UNIDENTIFIED MALE: And from bar you also delegate to www, www is also a label, right?

STEVE CONTE: So a label is not synonymous with that. A label in the case of this is any entity in the DNS architecture. This is a label. That's a label up there. This is a label. This is a label. It's just defining an entity point on the tree. When we talk about zones, we're talking about the delegation points of this domain space. The domain is the highest point, the apex, and then everything is its domain. But the labels are just a definition of some kind of entity there. We call it a label because we have to call it something. We don't want to call this "bar" all the time because it might not be "bar" all the time. We want to talk about the label at this level. So it's just a definition point.

UNIDENTIFIED MALE: Oh, okay. You earlier said that label has a maximum of 63.

STEVE CONTE: Say that one more time.

UNIDENTIFIED MALE: An old slide, it is stated that each label has a maximum character of 63.

STEVE CONTE: 63 characters, yes.

UNIDENTIFIED MALE: Is there any reason for that?

STEVE CONTE: As we said before, it probably fits into the fact that there was originally the size that it had to fit into a UDP packet. So 63 characters per label was probably some number that a scientist came up with and said no more than 253 characters for the entire string of labels, the entire domain name on that. I don't have that answer, and that's what I'm going to be looking up for you on that.

Any other questions while we're still on this slide?

Okay, this is kind of what I was talking about. Delegation is just the same picture with arrows of who is delegating what, and it's pushing it downward. Again, we talked about nameservers answer queries. An authoritative nameserver only answer queries. It doesn't do any kind of asking. Recursive servers ask the questions; nameservers answer questions. Authoritative servers only answer questions.

Zones should have multiple authoritative servers. That's called redundancy. So you can have a server here in UAE serving your domain, and you can also have a server in the United States as long as it's serving the same data, such as what we went through with the root servers last session. Then it's called a redundant server, and that's good to have. It covers both geographic and topographic distances but it's also in case one drops or is unavailable for some reason. Then you'll have some redundancy there.

UNIDENTIFIED MALE: Is it better to have primary and primary server or primary and secondary server?

STEVE CONTE: That's a great question. Actually, we'll talk about primary servers in a couple of slides, if that's all right.

All right, so let's step through the resolution process. This is really at the bones what DNS is. I like to attribute it like if you're going to a party and you want to meet that person at the other side of the room. You've never met them before but you want to be introduced to them. You probably know somebody who knows somebody who knows somebody who can eventually bring you that introduction on that. That's exactly what DNS is. At the heart and the soul of DNS, it's all about who do I know and who do I need to know to get to that next level.

I see you taking photos. We do have these slides available online as well.

UNIDENTIFIED FEMALE: [inaudible]

STEVE CONTE: Okay.

In this case, we're going to be looking for a movie on my iPhone here with my Safari browser, and I'm going to be looking for www.example.com. We're going to make some presumptions. We're going to presume that I've never, ever looked for this before. I'm going to presume that the recursive nameserver has just been turned on for the very first time and configured. It has not done any other query ever before. That's the only way this example is going to work because we'll go into caching after that.

I'm here on a network with my phone as I type in www.example.com and I hit ENTER. It sends that to the stub resolver right there on my phone, and the resolver asks the question. So the resolver is going to take that URL that I put in, send it to the resolver. The resolver is going to ask the question, and it knows that it can go ask the question to the recursive nameserver because that's the IP address of the nameserver that was given to it when DHCP assigned the data to it. My phone was given an IP address. My phone was given one or more DNS servers that it says go ask them because there's no one else.

My stub resolver sends the query and says, "Hey, nameserver, what's the IP address of www.example.com?" My recursive nameserver says, "Gosh, I don't know. But I know the app IP address because I have a file" – and we'll talk about that file – "I've got a file on my system that's called a hints file. Even though I've just been turned on, I have this list of root servers that I can go ask the question for. I'm going to go ask one of the root servers, in this case L-root.

Remember we said the recursive nameserver is both a nameserver because it's going to give you an answer but it's also a resolver. It's going to ask the question. In this case, it's asking the question on your behalf. It goes to L-root and says, "Hey, what's the IP address of www.example.com?" Then L-root says, "Well, I have no idea, but I have the information for .com, so go ask them."

Now we're going to go down that level in the tree. It takes the IP information for the nameservers for .com, sends it back to my recursive nameserver. That's called a referral, by the way. Now the recursive nameserver takes that referral and says, "Okay, I'm going to go ask .com. Hey, .com, what's the IP address for www.example.com?" And .com server says, "I don't know, but I know the IP address for the nameservers for example.com, so go ask them. Here's the information."

Now my recursive nameserver goes down to the example.com nameserver and says, "Hey, what's the IP address for www.example.com?" This nameserver says, "I know that because it's in my system, it's in my domain." So it says, "Here's the IP address for www" – which is the label in that – ".example.com." It sends it back up to the recursive nameserver, and now the recursive nameserver finally sends that IP address back to my phone and I can go to www.example.com.

That's DNS. Thank you guys for coming. It has been great. Oh, there's probably more, huh? That's really what DNS is. It's all about asking the question, "Who do you know?" If I wanted to meet the gentleman in the back, I would start here. "Hi, I'm looking for…," and you would say, "I don't know, but I know this person behind me" and go down the line.

It works really well, but it takes a long time in Internet time. This whole process, asking all the root servers and the TLD servers and the domain server and all that, it probably took maybe up to 100 milliseconds. That was a long time in Internet time. It was almost one-tenth of a second.

There are ways that we could be doing it to make it faster, and we'll talk about caching in a second. In fact, we'll talk about caching now. Caching speeds up the resolution process because I just got the IP address for www.example.com, but now let's say I want to go to ftp.example.com. In our heads we would say most of that path we've already followed, so all we're really asking for is the last part of that, the last label on that.

Now let's take a look at going ftp. Now what happened the last time if you remember when we were looking at www.example.com, we went first and we asked a root server. It could have been any of the 800+ instances of the root servers. It doesn't matter which one. They all give the same answer. We went and asked a .com server. We went and asked one of the nameservers for example.com.

Every time an answer was responded to, my recursive nameserver cached that response. It said, "Oh, I probably need to know this for the future. I bet someone else is going to ask for .com, so there's no need for me to go ask the root servers again where the .com address is. I'm just going to go directly. I'm going to hold that information because someone else is going to want that.

Now when we look at going to instead of www.example.com, we go to ftp.example.com. So let's look at it this way. Now I type in ftp.example.com. I send it to my recursive nameserver, and it says, "Hey, I want to know the address to ftp.example.com."

Instead of saying, "Gee, I don't know. Go ask a root server," it says, "Okay, I already the .com part of that, so I could go ask .com. But in this case, I also already know the example part of that, so I'm going to bypass those two. I'm not going to ask the root server because I already have that information. I'm not going to ask .com because I already have that information. So I'm going to go right to the nameservers for example.com and I'm going to say, 'Hey, I just need the address to ftp.example.com."

So I can speed up that process by 60% by caching data inside my recursive nameserver. Now my query for ftp probably took me 20 milliseconds if not less. It was probably a very quick trip for the end user experience. But also equally it saved some load on the other nameservers. They didn't have to answer yet another query that you already had a cached answer for, so it made the total bandwidth a little bit faster as well. Then once it sends up to ftp, it does the same thing. It goes back to my stub resolver, gives me the address, and then I can go and speak directly to ftp.

Yeah?

UNIDENTIFIED MALE: [inaudible] short question.

STEVE CONTE: Yeah.

UNIDENTIFIED MALE: Can we name the L-root server an authoritative nameserver?

STEVE CONTE: Yes. All the servers on this side when you're asking the question, they're all authoritative. All they can do is answer the question. They can't ask that question on your behalf. Root servers will only answer questions and they know some information about everything below them. When we look at that tree again, the dot, the root was on top. It knows some information mostly about the next level down about that. So it sends a referral back in.

Same with in this case it says gltd-servers.net, but it's really representing .com. It knows and will only answer, not ask a question, but it knows everything about the .com zone where to go to as the next referral point. That's what authoritative does. It says, "I can't ask a question. I can only give answers. And the answers I give are only in the space that I know."

So on the apex, if we go back to domain space, we remember the highest point, the apex, represents what the domain is. So when we were looking for .com, we went to the .com servers and it knows something about this whole space. Again, it knows mostly about the referral, how to get to that next level. It doesn't know anything about underneath that next level, but it will answer with authority about everything underneath that next point and it gives a referral on where to go. So the authority either gives a referral to say go ask the next guy or it gives an authoritative answer. It says, "I know www is beneath." If you're asking the example nameservers, instead of referring it says, "I have that information. Here's that information."

All right, synchronization of authoritative nameservers, this is where the question about primaries comes in. There are three terms that are used commonly. There's called the primary server, the secondary server, and a master server. This is really all about who has within – let's say you have example.com and you have some diverse nameservers. They're serving the exact same data across the network. You have either geographical and/or topographical diversity. So they're not all sitting in one server room in the same cluster. That doesn't do anything. You have to have them on different networks and things like that.

One of those is probably the one that you're editing your data in your domain on, and that's called the primary server. It's the one that you're going to put the updated information in first. Secondary servers will serve the same data, but they have to get that data from somebody and somewhere. They used to be called slave servers and if you think master/slave relationship, they have to talk to the primary to get that data. We, the DNS community, decided that's a really awful way to describe servers, so we went with primary and secondary servers.

Primary servers are the focal point on where the data originates within the domain. It's where the domain administrator probably goes and edits or uploads the zone file first. Secondary servers will query the primary server and ask, "Hey, is there an update?" It's either a yes or no question. If the primary server says yes, then the secondary server will try to pull that new data down to its server.

Now there's another term called the master. Primary and master are pretty much the same thing. Oftentimes, they're exactly the same thing. The only difference is sometimes the primary server might be hidden behind firewalls and stuff like that on the Internet and probably doesn't act – it can't act like a traditional nameserver. It's just acting as a distribution point for the other nameservers within that domain space.

In that case, the primary server doesn't answer to the public and the only servers that the public Internet sees are secondary servers. Again, they all serve the same data, so it doesn't really matter that relationship. But if that's the point where you as the administrator are actually uploading your file or making edits and all that, you want it to be a little bit more secure so you put it behind firewalls and you only allow your secondary servers to come and touch and touch it. That way, people aren't going to try to hack it or compromise it or corrupt your data and things like that.

So most of the time these days, you put the primary server behind firewalls and protect it, and it becomes the master server. It just becomes a distribution server. It's not used as a traditional nameserver.

As I mentioned, the zone transfer is initiated by the secondary. The secondary servers always go and ask the question, "Is there an update?" The primary server never goes and pushes and says, "Hey, there's an update." They always wait for the secondary servers to come and ask it. It's just the way the process works. There's really no reason that I know of that they do that except that way the primary server doesn't have to necessarily know how many secondaries you have. So you don't have to reconfigure it every time you add a secondary. It just waits for someone to ask and if it has the right credentials, it will say yes there's an update or no there's not an update yet.

I'm going to move a little bit fast because we're behind and I'm halfway through my deck. I'm going until 6:30, right, Cathy? Oh, then I've still got time. Okay.

The file that is in the domain name servers is called a zone file. Again, if we looked at zones, zones were the delegation points within the DNS structure. That means that the data needs to reside in those nameservers somehow, so they reside in something called a zone file. That's just a fancy word to say there's an ASCII file on this server that has the translation data between names and numbers. Now we've matured in DNS. Some DNS servers use databases to determine that, but for the most part it's still a regular old ASCII file that you can pull up and edit on your laptop if you wanted to.

Inside that zone file though there are different types of data that a nameserver can answer. The most common data is what's the IP address for www or for whatever host you have. But there's a pretty high amount of resource records that you can define within a zone file. And different types of applications use different types of resource records. We'll go through a couple of those and have some examples there.

A zone file consists of multiple resource records. It has to have at least one record in there, and that's call an SOA (start of authority). But it probably has a number of other records in there for translation or for other purposes.

Resource records from multiple zones are never mixed. You can have one nameserver running more than one domain. I have a personal domain, Conte.net. My nameserver runs Conte.net, but it also runs the nameservers for my wife's bookkeeping business, BookkeepingSCV.com. The same machine runs both, but it has to have two different files to manage those zones, those domains with because you can't mix them both together.

There are some formats of the resource records. Many of these are optional and no longer used, but we'll go through them anyway. Owner is the domain name of the resource record. When we went through the other ones, the example.com is the owner. We're not talking about a person. We're talking about the zone, the domain that we're in.

The time to live is in seconds. Remember when we were looking for ftp, we were caching example.com nameserver, we were caching the value for .com, we were caching the value for dot for the root. The time to live tells the caching server in the amount of seconds how many seconds can I hold onto this data before I consider it no longer valid data and I'm going to dump it. I'm going to flush it out of my cache and start asking the question again.

So at some point in time, your recursive nameserver, the guy in the middle, is going to start that whole process again starting at a root server and moving on down the chain because it wants to know that the information path, the query path is still valid because things might change. An IP address might change on a server or something else. So there's a time to live on that.

Class, when DNS was developed, it was thought that there would be something maybe other than Internet as a transport mechanism. We had a couple different network transports at that time, but Internet won out kind of like VHS won out over Beta and HD DVD won out over Blu-ray. No, it was the other way around. Blu-ray won out over HD DVD. So because Internet type class is mostly used, the other ones kind of faded away. But they're still there. They're still defined. You could absolutely use them if you were using a different transport class.

Type is the part that we're going to see most. Those are the different types of resource records. We're going to look at some resource records in the next slide or two.

And then the RDATA is actually the data, the values that you put into these translations within the types.

To bring that a little bit closer to making sense, here are a couple of different resource record types. An A record stands for anchor record. This is common to translate a name to an IPv4 address, so www into 192, 168, whatever. A quad-A, AAAA in that address, represents IPv6 address space. NS defines an authoritative nameserver. SOA as I mentioned was start of authority. We'll see that in a second too.

CNAME is an alias. You can have one A record defined as a machine. So like example.com, you can make that an A record. Then you can have www aliasing example.com. So it's just a quick way to add pointers to a single record and not have to redefine that IP address. It's mostly for convenience for the editor, especially when your zone file becomes very long and you have one server or maybe a handful of servers acting on behalf of ftp and e-mail and website and things like that.

We have an MX record which is a mail exchange server. It's the way that e-mail talks and utilizes the DNS to find out information about that domain's e-mail system. And we have a PTR record which is the reverse. Instead of translating a name into a number, this is a way you can translate a number into a name. Not often used as much as it used to be.

As of August 2016, there were 84 record types. Here we have six. Those are the six most widely used – well, there are a couple more now – resource record types, but there are 84 of them all in all. I'm going to just remind everyone that the slides are already on the schedule. Don't worry about writing down that URL. You can just grab the slides and grab it from there.

What that URL points to is this page that you can't read at all. This is the IANA page, and it's the IANA resource record types. IANA records these record types so people who are developing

DNS applications can see what they're doing. You might be able to see that. Like here is type MD and it says mail destination but it was obsoleted. So at some point MX took over that. That doesn't mean they're going to remove that old record type because there might be some legacy systems that still look for that. But they're going to say don't use it. Use MX. It's more reliable. There are more ways you can use it.

So at some point if you're interested, go to IANA, go to that URL in the presentation and take a look. Most of them probably will have zero meaning to most of the world except for a handful of really nerdy engineers who want to play with stuff.

Most common records are the A record and the AAAA record. Again, DNS was really initially about translating names to numbers. So the A record is the anchor for IPv4 and the AAAA for IPv6. What you have here is the definition with the answer. We're going to define example.com. We're going to anchor it to this IP address. We'll do the same thing. We can use the same space here, the same definition, but we're going to anchor it to an IPv6 address of this. It can have both entries in the domain space because we're talking about two different transport protocols.

Any questions?

UNIDENTIFIED MALE: The previous slide, can we retain both records, the A and the AAAA, IPv4 and IPv6?

STEVE CONTE: Yeah, absolutely. When a query happens actually when you ask the question of you want to go to example.com, many times it will give you both. The response will come back with the IPv4 and the IPv6 address. And depending on if your machine is IPv6 capable, it will say I'll take preference over IPv6 and I'll go to that address first. If it times out or if you are only IPv4 capable, then it will just take the IPv4 as a default.

Was that a question back there? No, I don't think so.

UNIDENTIFIED MALE: Excuse me, Steve. Only one. It will be a short question. You know that IDN is possible to calculate or recalculate to use [inaudible] number [inaudible] letters and so on, so it mathematically is developed and we used it to move from the [inaudible] letters [inaudible] script to [IC code system]. So my question is, is there mathematical conversion from the IPv4 to IPv6? Is it developed already or it's not clear?

STEVE CONTE: That's a great question. I'm not sure I fully understand that. Are you talking about converting IPv4 to IPv6 or IPv6 to IPv4?

UNIDENTIFIED MALE: Yes, [inaudible].

STEVE CONTE: I know there's some tunneling software out there, but I am not an IP guy and I don't have that information. Someone might in front of you. Go ahead. There's a mic there.

UNIDENTIFIED FEMALE: For me, I want to ask what the difference between regular DNS and DNS64.

STEVE CONTE: The 6 to 4 is the gateway so people can do the translation between IPv6 to IPv4.

UNIDENTIFIED FEMALE: The DNS [working] for IPv6, yes.

STEVE CONTE: But again, I'm not an IP guy, so you're way out of my realm on that one. I'm sorry. Do you want to add to that answer?

UNIDENTIFIED FEMALE:	No. I'm asking what's the difference between regular DNS and DNS64.
STEVE CONTE:	Regular DNS is either IPv4 or IPv6; 6 to 4 is saying that my service is not IPv6 ready yet, but you still might want to run services on IPv6. So it uses that tunneling process, the 6 to 4, to say here's my IPv4 addresses. Use this tunnel because I'll pretend that I have IPv6 addresses and the rest of the world can respond to that as well. So it's kind of a way to fake IPv6 on an IPv4 network.
UNIDENTIFIED FEMALE:	But there are three mechanisms to migrate from IPv4 to IPv6. There is another two types: dual stack and also NAT-PT.
UNIDENTIFIED MALE:	[inaudible] three mechanisms.
UNIDENTIFIED FEMALE:	Yeah, three mechanisms. Also, the tunneling mechanism.

STEVE CONTE: This is where I smile and nod at you because I'm not an IP guy on this. I'm sorry.

UNIDENTIFIED MALE: Just a little bit contribution to the IP stuff, actually it's not advisable that you be translating from IPv4 to IPv6. The two are not compatible. What is advisable is that your [own parallel], that is your server, can take both IPv4 and IPv6.

STEVE CONTE: Yes, dual stacked.

UNIDENTIFIED MALE: [inaudible] IPv4 traffic goes on that interface and IPv6 traffic goes to IPv6 interface which is a dual stack. But at the same time, you can [run internally] just as you've said, but actually trying to translate from IPv4 to IPv6 and from IPv6 to IPv4 is not advisable.

STEVE CONTE: So not being an IP guy, I would agree with that in the sense that we also don't want to do necessarily network address translation on DNS. I think it causes the same types of problems.

UNIDENTIFIED MALE:	I want to make [inaudible] for basically the last case [of the] DNS in terms of delegation and in terms of [regards the source keeping].
STEVE CONTE:	Okay.
UNIDENTIFIED MALE:	So can I make a [inaudible] and you advise who is the delegation authority and who is the [regard source management]?
STEVE CONTE:	I could give you a best guess on delegation. I can give you an absolute delegation on the top-level domain, but anything underneath that I couldn't without looking up that information myself. I couldn't give you any kind of idea of who is delegating it. Going back to your com.pk example, they might just have that broken up, but the people who….
UNIDENTIFIED MALE:	If I draw a [table] that it's easy for you?
STEVE CONTE:	If you want to draw a table, I can try to take a stab at it. While you do that, I'm going to keep on with this. Okay.

Remember we're talking about resource records. There are only a couple of them that can be in multiple places. In this case, we have a parent-child relationship. When we look at delegations and zone, the parent is always the zone above the other one. The one on top is the parent; the one below is the child relationship.

In the case of the NS record, the nameserver record, you're going to put them both in the child and the parent because when we did that resolution process when they were referring, when the [root] nameserver said I don't know but I know the addresses to .com, it has to have these nameserver addresses in its zone, in the parent zone, in order to make that referral to.

So you put that in the parent zone and you also put it into your own zone because that's telling your own zone that it is giving it the authority to make those answers. When you put the nameserver, the NS records, in your own zone, you're telling your own nameserver I'm authority because I have these records in there. You put it in the parent zone so the parent zone can make the referrals back to you as that nameserver.

So there's something else in there. Let's say from com it's referring to example.com. We did this in the resolution period. It's saying I don't know the address to www, but here's the example.com domain. Go ask them. Well, that's kind of a vicious circle there because example.com or the nameserver for example.com is NS1.example.com and NS2.example.com. Well, both of those fall in the example.com domain. So if com only knows about referring to the next level, it has no clue about the level below that because the zone, the delegation is within example. Now it has caught itself in a vicious circle. I need to refer you, but I don't know how to refer you.

So what they developed was something called "glue." That's a way that it can give hints to the parent on how to get to the next point so you don't have this recursive loop. We went back to this. You put the NS records in both the parent and the child, but you also put in some glue. In this case, the glue is defining A records, anchor records, in the parent that point to the IP addresses of those nameservers to the child. That way, it breaks the recursive circle, the loop, and allows the parent to make that referral because it now knows the IP address.

Does that make sense. That's kind of a weird and complicated thing. Does it not make sense anyone? Okay.

You can do it for an A or an AAAA record. It's just used to break circular dependency.

Before we talk about SOA, let me look at your table here. Grab that mic and come over here and talk to me about this.

UNIDENTIFIED MALE: Yes. Here is one. This is simple: abc.pk. Basically, this is a ccTLD. I think this registry operator will delegate and this registry server will keep these records. Is it clear?

STEVE CONTE: If I'm understanding this correctly, abc.pk, the delegation for this….

UNIDENTIFIED MALE: Yes, for the operator .pk and the record is also maintained at .pk.

STEVE CONTE: Well, see, I don't know. Is this an actual domain? If this is abc.pk, then pk delegates abc to somebody to manage.

UNIDENTIFIED MALE: Yeah.

STEVE CONTE: Who keeps that record? Are you talking about the zone file or the record of delegation?

UNIDENTIFIED MALE: The IP address.

STEVE CONTE: So then abc is who keeps the zone file, so this is not pk. This is abc.

UNIDENTIFIED MALE: No, basically this is the name of the domain. Basically who is maintaining the domain is maintaining .pk [inaudible].

STEVE CONTE: pk is a top-level domain. It delegates that authority for somebody else to delegate abc.pk. So the record, the zone file is maintained and kept by whoever is running abc, not by pk. If I run Conte.net – I do, I run Conte.net – .net delegated me to run through the process of going to a registrar and I registered my domain name. That's the delegation process. So .net delegated me to run Conte.net. I am responsible for the zone file for maintaining Conte.net - .net has nothing to do with the data in my zone file.

Same for here: pk might have delegated abc to run the abc.pk domain, but pk doesn't have any – for the most part. I mean, there are cases, but they're not the ones who are editing that zone file.

UNIDENTIFIED MALE:	Basically, when in your case you presented that when the recursive server sent the request where is the IP address of this website, basically this is a website. So maybe there IP address is 203.601.1.1. So it's a real IP.
STEVE CONTE:	Okay, so when I go, it will ask the root servers. The root server will say I don't know. Go ask pk.
UNIDENTIFIED MALE:	Yeah, pk will give this address.
STEVE CONTE:	pk says I don't know. Go ask the abc.pk nameservers. That's a different entity though. So they're not keeping that record. The zone file is kept by the registrant who registered abc.pk.
UNIDENTIFIED MALE:	And who is maintaining the IP address of this registry?
STEVE CONTE:	Whoever registered abc.pk. I don't know.

UNIDENTIFIED MALE: I think if I get what he is saying right, he wants to know the delegation process. I think what will happen is that pk points to the DNS. That is the delegation process if I'm right, pk points to the DNS server of abc.

STEVE CONTE: Correct.

UNIDENTIFIED MALE: That is the delegation process.

STEVE CONTE: Right.

UNIDENTIFIED MALE: So if abc wants to further delegate, I don't know, but the delegation process is pk pointing to the DNS server of abc.

STEVE CONTE: That's absolutely correct.

UNIDENTIFIED MALE: So this one has its DNS record that has the files, but this one just keeps that DNS service name in its zone file that points to this.

STEVE CONTE:	This one only has IP addresses to abc.
UNIDENTIFIED MALE:	Okay [inaudible] the zone file.
STEVE CONTE:	Right, and that's abc. It's not pk.
UNIDENTIFIED MALE:	So this one just keeps the IP address. The pk just keeps the IP address of the abc DNS server.
STEVE CONTE:	To the nameservers, yes.
UNIDENTIFIED MALE:	That is delegation.
UNIDENTIFIED MALE:	So IP address basically is stored by the pk?
UNIDENTIFIED MALE:	Yes.

STEVE CONTE: But only for the nameserver.

UNIDENTIFIED MALE: And IP is basically in zone.

UNIDENTIFIED MALE: IP, the DNS server of abc is what is kept by the pk.

STEVE CONTE: Everything underneath.

UNIDENTIFIED MALE: Everything else is maintained by abc.

STEVE CONTE: By the registrant.

UNIDENTIFIED MALE: [inaudible] on the DNS server, pk only points to you.

UNIDENTIFIED MALE: And the pk also maintains the IP address?

UNIDENTIFIED MALE: No.

STEVE CONTE: To the nameserver only.

UNIDENTIFIED MALE: [inaudible] abc DNS server IP address is supplied by abc to pk, pk just points those IP address to the server.

STEVE CONTE: As pointers.

UNIDENTIFIED MALE: So pointing to this, abc maintains the server. So this one just keeps the IP address, thereby pointing to this.

UNIDENTIFIED MALE: So the server redirected to this site?

UNIDENTIFIED MALE: Yes, every query that asks for this that comes to pk is sent [through this].

UNIDENTIFIED FEMALE: Ten minutes.

STEVE CONTE: What?

UNIDENTIFIED FEMALE: Ten minutes.

STEVE CONTE: Ten minutes? Wow, that went fast.

Okay, so that's the same thing that we're just talking about here and with glue is that the parent, pk, has the referral information to the child, to abc, abc is the child. So there's a list of nameservers that relate to abc at the parent, at pk. And there's glue, those IP addresses, also in the parent. But that's the only referral point. That's the only thing that it knows about abc is that the authoritative answers can be found at those nameservers. Everything within abc, everything else is managed in the zone file at the child, at abc.pk. That's it. So pk is only a list of nameservers [and glue].

UNIDENTIFIED MALE: My question is that this is basically the registry. What it maintains about the registrant?

STEVE CONTE: About what?

UNIDENTIFIED MALE: This is basically the registry .pk, .com. Basically these are the managers who manage these domain names?

STEVE CONTE: It's the level above it. It's the parent. It's not the manager. So it's not managing necessarily the data within abc. It's managing the delegations data to say I'm going to let you run abc.pk. I'm going to let [Sunday] run def.pk. But I don't care. I don't know anything about what's inside it.

UNIDENTIFIED MALE: So how does the registry maintain the record of the registrant? My question is that. How does the registry maintain the record of the registrant? This is just an example.pk [xyz].

STEVE CONTE: The history of the transaction you mean?

UNIDENTIFIED MALE: No the records in their registry servers.

STEVE CONTE: It's in pk has a zone file, pk's zone file only has nameserver information and glue and that's it.

UNIDENTIFIED MALE: That’s what I’m asking. It’s a nameserver.

STEVE CONTE: Yes. So the higher you go in the tree, root, top-level domain, etc.

UNIDENTIFIED MALE: Basically we say that who is maintaining the nameserver information? Nameserver information.

UNIDENTIFIED MALE: [inaudible]

STEVE CONTE: Okay, we’re going to have to move on. I’m not sure. We’ll touch base on this after, but we have ten minutes left and I want to get through more and I’m not sure if we’re getting anywhere on this.

UNIDENTIFIED MALE: Okay.

STEVE CONTE: So let’s talk offline or after the session.

UNIDENTIFIED MALE: Okay.

STEVE CONTE: Thank you. Sorry about that.

We're still talking about resource records here. I'll be fairly quick on this so we can try to get a couple more questions in. SOA, the start of authority, basically tells anyone asking or querying that zone some basic information. Again, we have owner. Remember the owner? We have owner. We have the type, SOA. We have the primary nameserver. This is mostly historical. Again, the primary and secondaries don't really matter. This is more of a definition point than anything else.

This used to be the e-mail address to the host master, the person responsible for editing that zone file. But the zone file can't have characters, can't have the @ sign. That @ sign is a character that parses and does other things. So what the people who DNS said is we'll put dots in there but really take that first dot and pretend it's an @ sign so you actually have hostmaster@example.com. It's just a way to have a reflection in there.

Then you have some values. You have a serial number. The serial number is used for synchronization. When the serial number is higher, that means that when the secondary is queried [then the primary] and the primary's serial number is higher than the secondary server's, that's what initiates a synchronization process. One of the great ways to do that is to do a date. So we have a year, we have month, we have day, we have sequence.

UNIDENTIFIED MALE: [inaudible]

STEVE CONTE: Sorry?

UNIDENTIFIED MALE: [inaudible]

STEVE CONTE: A lot of times, when you set the serial number, one way to guarantee that it's going to increment, and it's a great way to let you as the administrator know when the last time you updated that zone was, is by putting the year, the month, the day, and then the sequence number. I guarantee no matter how many times you draw that out, it will always be higher the next time you write it.

So if we wrote it today, it would be 2017103000. If we updated tomorrow, we would have 2017103100. So it would be incrementing already. If I made two updates tomorrow, it would be 2017103100 and then the next one would be 2017103101. It would be higher. It guarantees an increment when you change the serial number when you use that format. You can use whatever format you want. You can say 1. The next time you do it you can say 2 and all that. But it really doesn't give you much information on what's going on. The other values in here are for refresh, retry, and default TTL values.

Again, Alias is just a way if you set one label with an anchor with an IP address and it does multiple things, you can set aliases to that: www, ftp, all pointing to one IP address.

Mail, there was a problem before, as I mentioned, if I was using a service that was only a gateway service into the Internet like AOL or CompuServe way back when, it wasn't always on the Internet. If my mail server from that kind of service wasn't on the Internet in the old days, it couldn't deliver the mail. There was no way to do that.

So they wanted to make sure that there were ways to have flexibility and move through it, so they came up with the MX record, the mail exchange record. This is a way that it could define different places that would handle the e-mail services for that domain. So you can have multiple MX entries in there, and basically you set it by a weight, a number, a value here. Basically the lower the number, the more preference the e-mail servers will take to try to go that one first.

In this case, we had 10 and 20. If another mail server tries to send you e-mail, it's going to try 10 first. If the value mail.example.com isn't responding or is otherwise occupied or something else, then it's going to time out and it will say 20 is the next on the list. I'll try that one next. That way, it builds some redundancy into your mail service. It allows you to have more than one mail server. It allows you to have a gateway because you could have a store and forward path with somebody else. They'll maybe store your e-mail until your mail server comes online and then they'll forward it to you. So it gave you some more….

UNIDENTIFIED MALE: This 10 and 20, this is number what?

STEVE CONTE: It's not the amount of e-mail. It's the weight in preference. When another e-mail server comes, it's going to look for the lowest number first and say I'm going to send an e-mail. Out of these two, 10 is lower than 20 so I'm going to try to reach that server first. If it doesn't, then I'm going to go up the list to the next lowest value and then try that one.

UNIDENTIFIED MALE: It's prioritization.

STEVE CONTE: Exactly, it's a prioritization.

Reverse mapping, I'm going to go really fast on this. It's just a way to, as we did before, we did a name with an A record to a number. This is a way to set up a number, it's written weirdly, but if we look here we have 192.0.2.7. It's a way to map that number to a name. It's not used often anymore. It's not accurate and it's not authoritative. I could set that to be Conte.net if I wanted to. I could set it to whatever value I wanted, so it's not really an authoritative way. It was just a helpful way to define IP addressing.

DNSSEC, I'm actually going to skip. There are some sessions this week, workshops I believe tomorrow. I recommend you guys go to DNSSEC. Basically in a nutshell, it's called DNS security but I like to call it DNS authentication. It's not really going to provide you a secure method as we think of secure to do a query. It's going to provide you with a chain of trust to authenticate that the question you're asking is actually coming from the place that you think you should be asking the question from. I'm really going to leave it there because we are out of time and I'm going to blow through the rest of these slides.

Here's a sample. I told you that the zone file is text, it's ASCII. This is a sample of what one would look like. Here we have our SOA up here. It has to be in everything. We have the authoritative nameservers. So we're telling the zone file that it is authority, that it's primary. It can answer with authority because it's listed.

Let me go through this. Hang on.

We have an A record. We have AAAA record. Here we have two mail exchange records. Here we have an alias. We set example.com to this IP address. This time we're going to set an alias of www.example.com. But instead of putting an address in, we want to say it's whatever IP address example.com turns into. That way, if I ever change that IP address, I only have to change it once. I don't have to change it all the way down the line. Then I have some IP addressing for ns.example.com.

Quick question because I have a lot of stuff.

UNIDENTIFIED MALE: [inaudible] this file is stored [inaudible]?

STEVE CONTE:	This file is stored on the primary nameserver of that domain.
UNIDENTIFIED MALE:	[inaudible]?
STEVE CONTE:	example.com, not .com; .com only has referral information. This is at example.com.

You all sat through the RSSAC presentation, right? The one just before this for the root servers? Okay, so this is exactly what Andrew and Steve were talking about: 13 entities, root-servers.org, root zone change process. They talked about the distribution, all of that.

It's easy to get confused between the registry, the registrar, and the registrant. The registry is .com. It's Verisign. It's those who run a certain domain space on the Internet. If we look at it all the way down, Conte.net is the registry for anything below Conte.net. But when we talk about registry, we're typically talking about a second level, the top-level domain. So we have registry.

Registrar is the entity between the customer and that registry. So GoDaddy, Network Solutions, whatever. There are hundreds and hundreds of them out now.

Then the registrant is the customer. If you wanted to go and register a domain name, you become the registrant. You go to the registrar who is the agent between you and the registry and register that name. Then they put that information into the registry.

In some cases, the registry and the registrar are the same people. Mostly in the ccTLD world.

Download the slides. This is a basic look from the end user perspective and the registrant perspective. If I register a name, it goes into the registry through the registrar agent. It gets added. But as the end user, I can get to that domain information through that process.

I hate to wrap it up so fast. You guys hit me with some great questions in the middle though, so we blew through the end of it. If you see me this week, we should talk, sir, and make sure we're understanding but later this week. I have another meeting that I'm pending on.

I want to thank you guys for sticking. I want to thank you for blowing off the public forum and preferring to come and talk to me. I appreciate that. I am here all week, so if there are any further questions, please find me or if you see me in the hallway or whatever and grab me and ask questions. I don't mind. I love talking about DNS. Thanks.

[END OF TRANSCRIPTION]