SAN JUAN – How It Works: DNS Fundamentals
Saturday, March 10, 2018 – 10:30 to 12:00 AST
ICANN61 | San Juan, Puerto Rico

CATHY PETERSEN: For this session Matt Larson will be talking about DNS Fundamentals. Thank you again for your patience.

MATT LARSON: Good morning everyone. We've got a slide issue though, we're clipped off here on the screen. The slides are clipped on the side. Great, thank you. Alright, good morning everybody. Welcome to How it Works DNS Fundamentals. My name is Matt Larson, I'm VP of research in the office of the CTO and I'm happy to be here and we're going to go over the basics of DNS. So I hope by the time we're done today, you'll have a pretty good understanding of how DNS works at the high level.

So IP addresses are easy for machines but hard for people, that's basically why we're -- that's what an IPV4 address looks like. You maybe have some hope of remembering those but IPv6 addresses, that's a rather short one in its written form, those are even harder if not impossible to remember. People need to use names not numbers and this is been the case since the earliest days of the internet.

But early on names were simple. There were no domain names yet, domain names hadn't been invented. All the names were what we would now call 'single label names', that is they didn't have a dot in them, they were just a sequence of characters up to 24 characters and these were called 'host names'. Host is just a fancy word for a computer. A host is some place that you would want to connect to, to log into remotely or to send email to.

Mapping of names to IP addresses we call 'name resolution', resolving a name. In the early days of the internet, this was done very simply. It was done using something called a 'host file' which was very simple, it was just a file of text. If you're familiar with Esty host on a Unix or Lynx system, it's the same idea, not the same format but the same idea. It was literally a text file, it had names of machines and IP addresses.

This was centrally maintained by an organization called The Network Information Center and they had the master copy of this file and the way it was updated was very low tech by today's standards but remember, there were many, many fewer machines on the internet, we're talking like thousands of machines at this point, tens of thousands at the most near the end of the lifetime for host.text.

When an administrator on the network added a machine, removed a machine or changed its name or IP address, they would simply send an email into the nick and say, “Here’s the change I made.” And then the nick would update the master copy, the host file and there wasn’t even a database behind this, it was literally a text file that somebody brought up in a text file and edited.

There was no formal mechanism for distributing this, it was just made available by FTP to download and an administrator would periodically when they thought they needed a new one, they would download a new one. Once a week a file was released, so you would download one once a week and be reasonably current.

There are some obvious problems with this you can imagine. One is naming contention. All the edits were made by hand and there was no good method to prevent duplicates and there were only so many ‘good names’ names got more and more arcane as people tried to all play in this small sandbox of 24 characters. Every machine on the network had to be a unique name in this 24-character name space. Also, nobody ever had the same version of a file because it was released once a week and there was no formal mechanism for downloading, you got a new file when you got a new file and maybe your file was out of date.

Just the traffic and load on the network to move this file around, began to be a real issue. This is a time of course when like a 64 kilobit per second line, that was a big line, that was a lot of band width and this was a big file and it got bigger and bigger and so eventually it got to be a real problem moving this file around. I am told this is a little bit before my time on the internet, but I am told it got so bad near the end of the lifetime of host.text that it took longer to download the file then the update period. It's like painting the Goldengate bridge, by the time you're done you have to start again.

By the time you downloaded host.text there was already a new one. Clearly in a growing network like this, a centrally maintained host file just didn't scale, something had to be done. The discussion started in the early 1980's on a replacement and their two goals. The goal that you might think of as obvious one, is to address these scaling issues with host.text but another goal, which we'll talk about in a moment, was to simplify email routing, people tend to forget that but that was an important early goal for the solution. The result, not surprisingly was the domain name system. The requirements for are in these documents if you're interested spelunking through some ancient history.

This is my one slide summary of DNS. If you had one slide to summarize DNS, this would be it. Fundamentally DNS is a distributed database and this database however is distributed over the entire world. In this database the data is located all over, that's the distributed part. Data is maintained locally, so everybody has their own portion of the data that they maintain themselves but all the data is available globally to everybody else, in other words anybody else can look up anybody else's data.

DNS follows a client server model. Resolvers of a clients and if there's one thing to remember about resolvers, it's that resolvers send queries. Name servers are the sever side and then if there's one thing to remember about name severs, it's that name severs answer queries.

There's some optimizations for this, DNS uses caching to improve performance. Because we're talking about a database distributed over the entire world, the speed of light is only so fast, so a look up can take even in human terms a long time and so you want to be able to remember or catch the results of a look up and a look up could be multiple steps, you want to be able to remember not only the final step that you were looking for but you want to be able to catch all the intermediate steps, information from those steps that you've learned, so that as you

do this in the future, you'll speed things up by having to do fewer look ups.

DNS also uses replication to provide redundancy and load distribution and what I mean by this is that ideally in this distributed database, you have your copy of the data but you don't have just one copy, you have multiple copies of the data. This provides redundancy, if a copy fails there's another copy available for people to look up in and also provides load distribution, if you have popular data that people are looking up a lot, you would want to have multiple copies to spread the load of those queries. I think it helps at this point to have a picture, the old saying a picture is worth a thousand words, there's something to that, so this is the entire DNS echo system components all on one slide.

Here's where I apologize I might go out of frame here. Let's start from the lower left and work to the right and we're going to talk more about what these components are as we go but this is a high-level overview to show where we're headed. On the lower left we have a device that needs to turn names into IP addresses and this device could be anything.

Early on this device was a traditional computer, that was the thing that was connected to the internet, there wasn't much else connected but of course, more and more, now we get smaller

and more interesting things everyday connected to the internet. We use to joke in some circles that my refrigerator will be on the internet, well now your refrigerator might very well be in the internet, you can buy at Best Buy that goes on the internet, so anything on the internet, anything connected to the internet that needs to turn names into numbers, which chances are almost everything on the internet. It has a little DNS client in it called a stub resolver, remember resolvers are clients and resolvers send queries.

The stub resolver acts like a bridge between an application, like say a web browser in this example on this phone, the stub resolver's a bridge between an application that wants to turn this name into a number, into an address and the whole rest of the DNS system. The stub resolver accepts the request from the application and that's an API, if you're familiar with programming terms, the program is calling the stub resolver, it's just a simple piece of code.

The stub resolvers job is pretty simple, it has to take that request from the application, from the program and turn it into a DNS query and send it to something called a recursive name server or a recursive resolver is another term for that. The stub resolver is configured to know where to send its queries and it sends it to the recursive name server or the recursive resolver.

Now this is actually a combination of two roles which is one of the reasons that in the DNS community we can't decide what to call these things. It's a name server in that it accepts queries, it answers queries from the stub resolvers but it's also a resolver itself in that it sends queries. The recursive name server's job is to be able to navigate this distributed database that I'm talking about that is all of DNS and it actually looks up the data on behalf of the stub resolver. The reason here we have two levels clients if you will, is we have the super simple client, the stub resolver and that's at the very edge of the network.

The DNS component, the stub resolver's very, very simple, it send queries to these recursive resolvers, recursive name servers and they are smarter and they can do the heavy lifting, they can track down the answer that the stub resolver wants by sending queries to what we call a authoritative name servers. The authoritative name servers know about different portions of this distributed database and the authoritative name severs, they give answers to the recursive name server. Then ultimately gets the final answer for what the stub resolver asks for, it returns it to the stub resolver hands that back to the application and now you're off and running, connecting to the webpage, to the web server or whatever it is that application is doing. It's super high level, that's how DNS works, those are the important components.

I also didn't highlight it as I went by but there's -- I'm showing a cache there for the recursive name server and that's where the recursive name server remembers the results of the answers it's gotten from queries to these authoritative name servers, so that it can use those to speed up resolution in the future.

I've said that the DNS is distributed database but what's the structure of this distributed database? When you hear database, you might think of a relational database that uses SQL, such a database is made up of tables and every table has rows and columns, that might be the first thing you think of when you hear database but databases can have other structures and structure of DNS is in an inverted tree. Bear with me, this kind of gets sort of computer science 101 here very briefly.

An inverted tree, the root is at the top and the branches grow downward, it should not surprise you that a computer scientist tree is upside down and this inverted tree we call 'the name space' the name space is all of the data in DNS. In this inverted tree, in the name space, each one of these node, each box, has a label. The root node is special, it has a null label, so the label of the root node is not having a label, you can wrap your head around that, that's sort of a mystery, what is the sound of one hand clapping, it's that sort of thing.

We often talk about these nodes in relation to the root, where their position is, so the root is the very top, the nodes immediately below the root we call 'top level' nodes and below that we call 'second level' nodes and so on. Sometimes we use family relationships to refer to this, we might talk and this example about the root being the parent of .UK and .UK being the child of the root, you have parent child relationships in an inverted tree.

We've got nodes that have labels, there's a label inside every box. What are the legal names for labels? They're what we call 'LDH' letters, digits and the hyphen. The maximum length of this is 63 characters and case does not matter, uppercase and lowercase are compared as if they don't matter.

I know that there are nationalized domain names, that's something that we don't have time to talk about today but internationalized domain names are a way, it's like a layer sitting on top of DNS that turns internationalized names back into something that is only letters, digits and hyphens and an example of that is the one in the upper left, that's actually an internationalized name and now I'm realizing I don't remember what it is, I think it's .China but I'm sorry I don't remember. I think it's dot and two Chinese characters. From DNS perceptive all that IDN stuff, it just sort of rides above it. From DNS's

perceptive it's just characters, it's just letters, digits and a hyphen.

Every one of these nodes has a name and its name is what we call a domain name and the definition of a domain name is really simple, you just start at the node we're talking about and you read it's label and then you put a dot and then you move up the tree to its parent and put down that label and put a dot and you keep going until you get to the root. In this case, the one we've highlighted is www.example.com , that is the domain name for that node and that uniquely identifies one node in the name space.

What we call a fully qualified domain name is a name that ends in a dot because it's not relative to any other name, it absolutely refers to one name. In this case, remember the rule I said to make a domain name, www.example.com dot and then the root but the root has a null label, so what a fully qualified domain name really is, is it has the roots null label at the end, after the last dot but you can't see the roots null label, so it just ends in a dot.

If you're familiar with file systems like the Unix file system or for that matter the Windows file system, that also is data structure that can be represented as inverted tree and in that case nodes are files of directories and corresponding to domain names

would be path names in a file system. A full qualified domain name would correspond to an absolute path name, one that starts with a slash. A path name that starts with slash that refers to only one file or directory, it can't refer to anything else.

Domain, what is a domain? Well that has a pretty simple definition too. That is a node and everything below all, all its descendants, remember we use these family terms sometimes. Here I'm showing the .com domain and obviously this is a tiny portion of the .com domain. If you look in the literature that Verisign included in our swag bags, .com is up to a 131 million names and change, obviously I'm only showing three out of the 131 million but .com and everything below it, all those 131 million, that's the .com domain. To make a domain just pick a node and you take everything below it. The top of a domain is what we call the 'apex' of that domain. Apex is just a word that means the top.

Now you do a really important concept in DNS and this called the zones. The name space is divided up to allow distributed administration and what that means is, we divide it up so different people can maintain different portions of this distributed database. Remember, that was one of the goals, we tried central, centrally maintain with host.text, that didn't work, so the idea was, let's distribute administration, let's let

everybody maintain their own data, so we don't have this central bottleneck.

The name space is divided and these divisions are what we call zones and zones are created by a process called delegation. You have a zone that does the delegating, that's what we call a parent and it delegates to a zone, it's child and that's the child zone. This makes more sense with a picture. Here's the name space, the tiny, tiny portion of the name space. Here I'm showing some administrative boundaries, which is what zones are. There's no way to know what these boundaries are, if you just look at then name space like that and I said where are the zones, you can't tell. Just looking at the name space, that's not enough information, you have to know where the delegation boundaries are.

I happen to know and I happen to have used real world examples here, there is a very zone at the top of the name space called the root zone and the root zone delegates to all of the top level zones, every single name at the top level corresponds to a zone and then delegate further on down as you can see. Conceptionally the way to think of it is that a zone has information, has delegation information that creates to and points to a child zone. In the root we have delegation information pointing to the various TLD's. Does anybody know

how many TLD's there are? Too many is the answer, know the answer is about, last I looked it was 1543. In the root zone then, there are 1543 arrows if you will. There's data pointing to the 1543 top level domain zones. Then if we were to look in .com, let me go back to the root.

The root relatively speaking, not very big, there's not very much in there, we only have to know about 1543 TLD's. Let's move to .com on the other hand. As I just said .com is 131 million delegations to second level domains, to second level zones. The .com zone very, very large. But remember, the .com domain is .com and everything below. The .com domain is unimaginably vast, it's like every single thing in .com but the .com zone is just this portion here with dotted line around it. All that the owner of .com, the administrator of .com, all that they manage are these delegations. The .com zone contains nothing but 131 million arrows saying, "Oh, the foo.com zone is over here, the example.com zone is over here."

Remember I said name servers answer queries, so what's the relationship between name severs and zones? That device I showed on the earlier slide with the DNS eco system, the authoritative name server, that is what has complete knowledge of the zone. You the zones to authoritative name servers. When authoritative name server for a zone knows everything about

that zone and can give definitive answers, it can say, “Alright, here’s an answer to something, here’s answer to your query.”

That tell you something is in the zone or it can say, “No, that doesn’t exist, that think you’re looking for, that’s not in the zone.” And I would know because I know everything about the zone. Zones should have multiple authoritative name servers, this goes to the redundancy goal that I mentioned a moment ago. You want to have multiple authoritative servers for your zone because if you only had one and something happened to it, then your zone would be off the air and it wouldn’t be able to -- nobody would be able to resolve queries for it.

If we’re going to have multiple authoritative servers for a zone, how do we keep them in sync? If I decide I have this zone, I have runexample.com, I want to have two authoritative name servers, how do I keep the data in sync on those two cause I need the example.com data in that zone on both of those? I won’t belabor this point here but the DNS protocol has replication built in. There’s something called a ‘zone transfer’ that allows you to move data between authoritative servers.

The place where you make the change is what we call the ‘primary name server’ and then the authoritative servers that get their data from the primary via this zone transfer mechanism we call those ‘secondary’ or ‘slaves’. That’s really about all I

want to say to that except one important point is that the authoritative data on all these servers, the zone data on all these authoritative severs, it's the same, it's all equally good. The primary is not better than a secondary, the primary is just where you make the changes and the secondary's just get the data from the primary.

Let's go down a level now. We've been talking about data in DNS zones but what's that data look like? The DNS standard if you look at the documents that define how DNS works, it's specifies a couple of things. It talks how these DNS queries and responses should look like and we call that 'wire format' which is if you're going to put a DNS packet on the network, what should that packet look like? It also defines something, it's kind of unusual, it defines something we call 'master file format' and this is a standard way to write DNS information, to write it down, to write it in text. There's a standard way to write DNS information and a zone file is what contains all the data for a zone in this master file format.

Every node in the name space, remember every node has a domain name and a domain name can different kinds of data, DNS data, associated with it. That data we store in things called 'resource records'. You sometimes see that abbreviated as RR. Research records correspond to different types of data. This is

the sort of thing that helps with an example and I have examples coming up, don't worry. All of these resource records for the zone, go in a text file we call 'a zone file'.

A zone file simply has all the information for a zone in this master file format. These resource records I'm talking about, they have a few different fields. This is something again that I think helps to just give examples for. I'm going straight to talking about resource records. These are some common resource record types, probably the most common. As you will see there are a bunch of them. DNS can store all kinds of data beyond just the name to address that you think of. The most popular by far and the main reason we have DNS is mapping names to addresses. The first two I have on the slide here would be highly represent an IP version 4 and IP version 6 address.

Then we have some other ones that I'm going to talk about in turn here. Remember there are a lot of them. Last time I counted there were 84 different DNS types and there is an IANA registry for these different types. Of the many thousands of registries that the IANA tracks, one of them is related to DNS types and that's what it's called and that's the RO for it and there were 84 of them. Some of them are exotic and never used, some of them have been deprecated and we can have a lot

more. There's just a screenshot of the website, you can go look yourself, it's not terribly interesting.

Let's talk about some of these resource records one at a time. The most common thing that we use DNS for would be mapping names to addresses. I know I keep saying that but it's true, that's the main we have DNS. These are some examples of that. These are resource records written in this master file format. On the left we have the domain name that the resource record is associated with and then we have a type, the type of the record and then we have the actual data itself that the record carries. That first record says that example.com has the IPB4 address and then what we call a 'quad A record' four A's, that stores an IPv6 address and that is an example of a quad A record that says example.com has that particular IP version 6 address.

DNS is a little bit unusual in that there are record types that are what everybody cares about, that are the reason we have DNS, that would be like address records but then there are some types that are really used mostly by DNS itself to make it work and these are mingled together. The analogy I like to use is a warehouse. Imagine you rent a warehouse and you want to put stuff in, you got goods you want to put in it. You don't just back your truck up and start throwing the boxes of stuff into the warehouse, before you can do that, you need to put up shelving

but you don't really care about the shelving, the shelving is not why you got the warehouse.

The thing you care about is the stuff you're going to put in the warehouse but you can't put it in until you have the shelves. Some of these types are shelves. I'm going to talk about name server types and the SO8 type. That stuff only used by DNS, only DNS cares about shelving. Stuff that the consumers of DNS care about, like address quad Q records, that's the stuff we care about, that's the goods on the shelves.

Let's move on to these name sever records, to a piece of DNS shelving as it were. Right away this gets a little intricate because NS records appear in two places, they appear in the parent zone and the child zone. What these NS records say is, they list the authoritative name servers for the zones. These two NS records say, the zone example.com has two authoritative servers and the names of those authoritative servers are NS1.example.com and NS2.example.com, semantically that's what these records say.

The purpose of NS records is to mark delegations. Remember I had this same slide a moment ago and I had the arrows and I said the arrows are the delegation information. The arrows are how a parent zones says that a child zone exists. If we look at those arrows, what those arrows really are, NS records. Let's

look in the case of .com. In the root zone we have that arrow if you will delegating .com, that creates the .com zone, that says, here's where to go to learn about .com" and that arrow is actually those 13 NS records.

Those happen to be the actual 13 NS records for .com. What that says is, .com has 13 authoritative name servers and they are named a.gtld-severs.net through m.gtld-servers.net and these NS records, they appear in two places. They appear in the parent and that's where they actually are the delegation and then they also appear in child zone. I like to think of NS records or any record for that matter, conceptionally they sort of hang off the particular node that they belong to. We have those 13 NS records, you can sort of envision them as hanging off that .com node in the name space.

I know this is a little complicated and may seem a little arcane but when we talk about how DNS resolution works, NS records are critically important. If I'm going to tell you how DNS resolution works I need to tell you how name server records work. This gets a little more complicated still in that the name of a name server is not enough. If I tell you, "Oh, you want to go to example.com one of the authoritative servers is ns1.example.com", well if I tell you that and you want to go talk to ns1.example.com what's the next thing you're going to want?

The IP address. The delegation also, in some circumstances has to include the IP address of name servers as well. Those are called 'glue records'. Sometimes you'll hear people talking about glue and that just means an address record that is part of a delegation.

Another record that is important to DNS only and that nobody outside DNS cares about is the SOA record. There's one and only one of these per zone. If you go looking for SOA records, at the top of every zone is one of these SOA records. Most of these values are related to those zones transfers that I talked about. These have timers that tell authoritative servers how often they should check to see if the zone is up to date or not and whether or not they need to do a zone transfer. I have an example, no I don't even have the description of what they are but we don't need to talk about SOA records beyond that, just to know that there's one of them per zone and that has to do with zone transfers.

A few more types I want to talk about. One of them is the type that makes an alias called canonical name, a C name record and this creates an alias from one name to another. For example, the C name record there says that if you look up the name male.example.com what you're looking for isn't there, don't look here, what says is go look over here, go look at some-

hostexample.com that's where you want to look. C name creates an alias from one name to another. That's about all I want to say about c names.

Mail routing, remember I said early on that one of the important early goals of DNS was to simplify mail routing and here was the problem. Where does mail for user@example.com go? In the old days the way this would work is you would look up the address of the right side of the email address and you would deliver the mail there.

Imagine back in the host.text days and people would have email addresses that looked that, user@host name, the email addresses were comparatively simple, they were just a username @ and then the host name that remember, was up to 24 characters. What that meant is that your email address was tied to the host where the mail server ran, where you received your mail.

There was tight coupling, your email address basically reflected the name of a machine and there was no way to say, this is my email address but I want mail to go somewhere else and that's where mail routing came, that was an important goal for DNS. What that does is that decouples the email address from where the email should go and this done with something called a 'mail exchange record' and here's what they look like.

They say for a given domain name, where should I send the mail? There are two values there, there's a preference lowers better and then there's the name of a mail sever. In this example what we're saying is, for any mail, if a mail server gets any mail addressed to whatever user @ example.com, it's going to look up the MX records for example.com to know where to send that mail and what these MX records say then is, okay for the mail server that has a mail to user@example.com, it should try to send it to mail.example.com because that's the lowest preference, lower is better. If mail.example.com for whatever reason is not available, well then you should try mail-backup@example.com.

You can see I have my email address is whatever user@example.com but the mail goes somewhere else, it goes to mail.example.com and this wouldn't have to be necessarily in example.com you could have your email address be one thing and the name of the mail server could be in a completely different domain and often is. That's when the case when you have a third posted email provider, which almost everybody has. There's a very tight coupling between email and DNS that you may not have been familiar with but if DNS didn't work email would come to a screeching halt because DNS says, how to route email on the internet.

We've been talking mostly about name to IP, which is what people want to do most of the time. Most of the time what we care about from DNS is mapping names to IP addresses and that's what we call 'forward mapping'. But there are cases where you want to do the reverse, where you want to start with the IP address and know what's the name that corresponds to this IP address?

There are some use cases for that. Imagine you've got a server of some kind connected to the internet, let's say it's a web server and people are connecting to your web server to download web pages and you want to log those connections in a log file, you want to know who's connecting to my web server? Rather than just putting the IP address in a log file, wouldn't be nice to know the domain name corresponding to that IP address so that you could write a log file that had a name in it, rather than a number.

Or let's say that you're tracing a path across a network that the traffic, there's a clever little tool called trace route that says, alright I want to know I'm here, I want to know the path that the traffic takes as it goes to this destination out here. Trace route will tell you the IP address of every hop, of every router that you go through but wouldn't it be nice to know the name rather than the IP address, so that you could have a little bit more information about where it's going. That's why reverse mapping

-- that's what it can be used for. Realistically it's not nearly as important as forward mapping but it does its uses.

How do you do this? How do you do reverse mapping? Imagine things back in the host.text days. We had everything in a text file. If I want to look up a name, I would go through that file until I found the name and I would go, oh there's the IP address. Now let's say I want to reverse mapping, I want to have an IP address and know the name. I go, I look through the file until I find the IP address and then I go, oh and there's the name. In one place the host file, I can do a search on host name or I can do a search IP address.

Let's look at the DNS name space. Let me go back a few slides. This name space is optimized for looking up domain names. You can imagine and we're going to talk about how this works. If I want to look up www.example.com, you can imagine I start at the root and I go down and you can see how you would find your way to www.example.com where you would then find the address you're looking for, whatever piece of information.

But what if I say where is the IP address for 192.0.2.1? Where is it? This data structure is not optimized for looking up IP addresses, it's optimized for looking up names. That's where reserved mapping comes in. There is a special type of record that's designed to map from IP addresses to names but you also

along with that, you need a special part of the name space where IP addresses go. There's this corner of the name space that you may not even have been aware even existed, where every IP address, every possible IP address has a corresponding domain name over there.

For example. The IP address 192.0.2.7, that turns into that domain name up there, we flip IP address around so it's 7.2.0.192.in-addwe.arpa in-adder stands for internet address and arpa is a long story. I'll hold off on the long story. I'll tell a little bit of the long story which is that this whole scheme I'm talking about was not part of DNS at the very, very beginning.

We had host.text, you could do name to address, you could IP address to name, everybody was happy. Then we got DNS and we have the data structure we have, we have the name space, you can easily do name to IP but them people go, wait a minute what about IP to name, I want to do reverse mapping and then everybody said, oh now what are we going to?

We need to make this portion of the name space, we need make domain names for every possible IP address, so that we can look up IP addresses as domain names. Where we going to put them? That's where in-adder came from, short for internet address. Then the idea was, should that be a top-level domain, should in-adder be a top-level domain?

At the time the thinking was, no it doesn't feel important enough to be a top-level domain, let's put it underneath this TLD called .arpa and .arpa was a way to bridge from the world using host.text to the world using DNS. When DNS was first coming online, they took every name in host.text file and they slapped .arpa at then of it and they put it in DNS. This of course is referring to the arpa net which was the early, early, internet.

Now you had way, if you were using DNS, you could still look up stuff that had been the host table because you just took your host name and put .arpa at the end of it. Over time people set up domain names and they no longer needed their host names in host.text, so .arpa was just a transition mechanism. If it weren't to in-adder.arpa we probably would never have arpa if would be long gone but instead, somebody decided, in-adder doesn't seem important enough for a TLD, let's put in underneath .arpa. Maybe I'm the only one who finds that story interesting and that's okay.

We've got these things called PTR records and here's where they go. Here's the name space, I've showed example.com over there, that's the name space we've been talk about for the past hour but then look at this horrible thing. This is how you map this particular IP address, 192.0.2.7 to a domain name and there's a PTR record. This is a convention, the only reason this

works is there's a set of rules and everybody knows what the rules are and everybody follows them.

If you have IP address space assigned to you, if you're the one who is assigned 192.0.2.7 if you want people to be able to reverse map that, if you want them to be able to map that IP address back into a name, you have to put a PTR record at that domain name. The regional internet registries manage the in-adder-arpa name space and they will give you the delegation to the appropriate domain name. You have to on your name servers, you have to run authoritative name servers for that zone with that PTR record.

Then what everybody knows how to do on the look up side, is everybody knows, if I want to reverse map 192.0.2.7 I know what I have to do, I have to turn it into that domain name. 7.2.0.192 and then I look up PTR records and what comes back, example.com, that's the reverse mapping. This whole system only works because everybody agrees how the convention works.

I have one slide on a fantastically complicated topic, DNSSEC. I should probably take it out because I can't begin to justice. But DNSSEC is sort of in the air because there's some important changes to an important cryptographic key that ICANN manages related to DNSSEC. The short story here, as short I can make it,

is that early on, DNS had no notion of security at all, it just didn't go into the design at.

Then what we realized is, a DNS query is one packet and DNS is response is one packet and what's to stop somebody from sending me a response that wasn't to the query I asked for? Long story short, it's very easy to slip in bogus responses. Resolvers have gotten smarter and smarter over the years to figure out when they might be in a situation that somebody might be trying to trick them. Ultimately there's no way around this accept cryptography.

Ultimately what we need to do is cryptographically sign DNS information. Have not only the information in DNS but a digital signature of that information, so that when you get the information, you can cryptographically verify the signature. There's a whole bunch of complicated machinery that was bolted on to DNS to make that happen and that's DNSSEC. If that's all you ever hear about DNSSEC in your life, you probably will be happiest.

There are many, many other resource record types. I have just a smattering of that 84 here, just to give you a sample for the kinds of crazy things that can go in DNS. Some of these are barely used. I don't think I've ever seen an X25 record in the while that maps domain to and X25 address, I'm not even sure how many

X25 networks there are any more. That's just an example that showed how extensible DNS. If you can think of a crazy DNS you can put in DNS you can make a resource for it and you can put it in DNS.

If we pull all of this together, here's of an example of what we call a 'zone file'. If we imagine the example.com zone, if we imagine a really simple zone. Let's imagine somebody gets a domain name, chances are what are the things they want to do with the domain name? They probably want to have a website and they probably want to do email with that domain name.

This is an example zone file with the bare minimum information necessary to do a website an email. Most of the zones on the internet are like this, they're this small, they're just not that much to them. A zone looks like this, it has an SOA record and some NS records at the top, you're always going to have that, that's the shelving of DNS. That says the SOA record has some important parameters about the zone, it has a serial number that changes every time the zone changes.

Then the NS records say, these are the authoritative servers for this zone, here's where you should go if you have questions about the zone and they can give you answers about what's in the zone because they load this file, they have that information. Then we have an address for example.com and by convention

now when you type a domain name like this you expect to the web server so we can infer that when 192.0.2.7 is the web server for example.com and it also has an IPv6 address, you can see there's one of those quad A records.

Then we have some MX records that say, if you're sending email to some user at example.com, you should send it to those mail servers. Then we have www.example.com and that I've made as an alias back to example.com, this means somebody can type example.com in their web browser or they can type www.example.com in their web browser. In the case of www, what an authoritative name server says to a resolver looking up is it says, anything you're looking for www you should go back at look at just example.com. Then we have some glue records, some addresses and name servers. That's it, that's what most zones on the internet are because they're just not that complicated.

This is the big finish. This is what we've been building to which is how do we resolve things in DNS? We've got this name space. It's divided up into zones. The zones have data in them, that data is in the form of resource records. The domain names in the zone have resource records associated with then and if I want to look stuff up, if I want to look up one of those resource records, how do I do it?

Stub resolvers, recursive name severs and authoritative name servers, the components that I had in that slide, that picture early on, they all cooperate to look up DNS data in the name space. If you're going to look something up in DNS, a query always has three parameters, it has the domain name you're looking up. It has the type of data you're looking up, in this case an address record. There's also this thing called class that is a long story that we don't talk about, it's always going to be IN for the internet class. That's another thing that if you don't know any more about it you can just be happy.

Essentially what this says is, if you're going to look up something in DNS, you have to specify the name and you have to specify the type. It's really a look up as opposed to, if anybody's familiar with how to, use an analogy again back to relational databases, relational databases almost always you look things up using SQL, which is a full-blown language.

The QL in SQL is for query language and you can compose complicated queries in SQL, you can say in this database show me everybody who's last name is Larson where number of dogs owned equals 2 and type of dog owned equals pug and they could do this query and they would find because my last name is Larson and I have two pugs. That's an example of a complicated query you can do against a relational database. DNS not like

that. It's very rigid, you have to specify a domain name and a type. This has been one of the things that has let DNS scale so much, is that the query side of it is relatively structured and simple.

There are two kinds of queries. Stub resolvers send what we call recursive queries and what a recursive query means is, I either need you to give me the complete answer to what I'm asking or I need you to give me an error and say that you can't do it or it doesn't exist. Remember, stub resolvers are really, really simple they're everywhere. My phone has a stub resolver. This camera probably has a stub resolver. This TV has a stub resolver. Simple little pieces of code that turn a program request for a name into a DNS query and they launch this DNS query to recursive name server and they wait. They either need to get the answer, the complete answer or an error.

Now recursive name servers on the other hand, they are much smarter, their whole job is to track down data in the name space. They can accept partial answers that we call 'referrals' and they send queries called 'non-recursive queries' that basically say, it's okay if you don't have the complete answer, you can give me a referral and send me somewhere else and I can follow that referral and I can deal with it. This is the high-

level algorithm for processing a query, I'm not going to read that slide.

You have start resolution somewhere. Resolution starts at the root zone. If you don't have any information cached from previous looks up. Let's say this is a resolver that's just been turned on, time to go, let's start answering queries, you have to bootstrap yourself with some information and DNS resolution starts at the top of the name space, at the root and you work your way down.

In order to get going, you have to know, how do I contact name servers for the root zone. Name servers that are authoritative for the root zone and for short hand we call those root names servers. There is no way to find these, there's no way to ask the network and discover them. When you plug a device into the network, it can ask the network and it could say, I'm a new device in the network, can you give me my IP address and some other configuration parameter and we have a protocol for that, it's called DHCP and the network says, sure, here's your IP address and here's some other stuff and the device is happy. In this case a device came up knowing nothing and got an IP address and it's on the network.

A recursive name server on the other hand, it can't do something like it. It can't come up from nothing and start resolving queries,

it has to know how to get to the root and can't discover that, it has to be configured. There is a file called the 'root hints file' that is the list of the root name servers and their names and IP addresses. Every recursive name server has to have a root hints file with the names and IP addresses of the root name servers in there and that's what it looks like.

It's 13 NS records because there happen to be 13 root name servers and they are named A.root-servers.net through M.root-servers.net and why they are named that is another long story. DNS is full of long stories that I'm not telling you today. If ply me with milk and cookies I can maybe explain some of them or peanut M&M's. I once said that at my former job and then the administrative assistant sent me a five-pound bag of peanut M&M's m birthday, that was very nice. Peanut M&M's will do it as well.

We have these 13 NS records that basically say, the root zone has these 13 servers, authoritative servers with these names and then each of those servers has an IPV4 address and each has IPv6 address and those are those addresses. This file, this very file with this text in it has to be on the disk or burned into the code of every recursive name server on the internet of which there are a lot. Depending on how you count, millions. This is everywhere.

Let's take a slight detour and talk about the root zone itself. It would take a lot of peanut M&M's to talk all about the root zone, so we can only talk a little bit. Administration of the root zone is complicated and this is for historical reasons. Two organizations cooperate to administer the root zone. ICANN by way of the IANA functions operator and that is the portion, the hovion subsidiary of ICANN called PTI, they run IANA functions and then Verisign has a roll called the Root Zone Maintainer. This relationship goes back to the early 90's, mid 90's and it's very complicated.

There used to be a third party involved, the US Department of Commerce was the third party involved in authorizing changes, after the IANA transition in 2016 the Department of Commerce no longer has a formal role in that. The administration of the root zone uses to be even more complicated. That's the contents of the root zone. That's making the root zone file. That file goes to those 13 root server names and there are 12 organizations that operate the authoritative name servers for the root zone. These are the corresponding organizations.

Remember, we have A through M, are the names of the root servers and these are the organizations that operate them. There are 13 letters but organizations and that's because Verisign operates two, A and J and the reason for that, many

pounds of peanut M&M's required. This is an interesting group of organizations.

That is a long story. I can tell the story, I'll tell the story. I have time to tell the story. This list of root servers has expanded over time. There weren't 13 to begin with. For a long time, list grew and grew and grew and it grew to 9 and the root servers were not named A through M, they had domain names corresponding to the organizations that ran them.

Then we wanted to add more root name servers because the internet was growing but DSN packets can only be so big, there's a size limit or at the time there was a smaller size limit then there is today and what we really needed was for all the information -- let me go back to the root hints file. We needed all of this information, those names and IP addresses, they all had to fit in one packet, that fit in 512 bits.

When we had 9 servers and they all had these different domain names, that filled up the 512 bits but a guy names, I'm pretty sure I'm giving credit the right person, a guy Bill Manning had the observation, wait a minute, if we change the names of all the root name servers, we can cram more in and the reason for that is this compression scheme that DNS uses. I told you it was a long story.

What that means is that the string root-servers.net only needs to appear once in a DNS response and then you can have other places to refer to that string. The point is, by changing the root name server names to this root-server.net convention, we'd have room for more. We'd have room to go from 9 to 13. At that point we only had A through I, so time to add J, K, L, M.

At the time this was before Verisign bought Network Solutions, so it was Network Solutions and this was before ICANN, we're talking the mid 90's and the IANA functions were run by the University of Southern California, guy name John Postello, who died tragically early in 1998 and so a guy at Network Solutions and John Postello, they were like, we're going to add these four root servers, how we going to do it?

They said Network Solutions, why don't you take J and K and ISIS, the part of University of Southern California the IANA folks, we'll take L and M and what we'll do is we'll have these be like baby birds in the nest. We'll have them close to us and we'll get them running and then when we're ready we'll give them away to people.

The first one to go was K and that went to Ripe, the Regional Internet Registry for Europe. Now Verisign had one, they'd given one away and then IANA still had two. The next one to go was M for the Wide Project in Japan. The IANA folks said, how about we

give J to Japan and the Network Solution folks said you've still got two we've got one, why don't you give M to Japan, okay fine, M goes to Japan.

Now, Network Solutions has J and IANA has L and unfortunately John Postello died at this point and he was the person whom everybody trusted. He single handedly gave out these root name servers to people because the internet was a much simpler place and people trusted John. He helped DNS get to the way it is now.

Unfortunately, things were frozen at that point and that left Network Solutions with J and the IANA function which then went to ICANN with L. That's how we have those root servers where they are. I hope it's okay to tell that story in public because I just did. It's the historical record, it's what happen and it's no secret that this all very complicated politically and this is how we got to where we are and it's very complicated to change it. That's as short as I can the answer.

One thing I will say about this list of organizations is that if you look at them we have a commercial organization, we have educational institutions, ISP, United States Government Agencies, we have a little bit of everything, domestic in the US and International outside the US. Really the only thing that unites these organizations is that they run a root server, they

otherwise have nothing else in common. It's for historical reasons that we still have this arrangement.

It's been very -- no one has been quite sure how to change it. We have these operators who've stayed the way they are for 20 years. I should say that the operators do a good job and do this without compensation. It's something they do for the community. But if makes the arrangement complicated because it means that the root zone is complicated. It takes two organizations to make the root zone contents and then it takes 12 organizations to make the authoritative servers available for the root zone.

There are actually lots of root servers. There are 13 root server V4 addresses and 13 root server V6 addresses but because of a routing technique called 'any cast' it's possible to have multiple instances of root servers. A particular root server IP is not only in one location, it can be in multiple locations and the way internet routing works, is when you send a query to that IP, the internet routing system sends you to the closest instance of that IP that can accept your query. This has allowed the root server system to expand to -- last time I looked it was about 950 instances, all over the world. There are lots of root server instances, the world is blanketed with root server instances, it probably has more capacity than any other zone on the internet.

Super high level, here's how the root zone change process works and please note my disclaimer at the upper right. This is a very, very simplified version of this process. This is just to show the high-level boxes, to give you the 50,000-foot view of this process. When a TLD manager wants to change something in the roots zone, which would be add a name server, add an authoritative name server for their TLD zone. Remove an authoritative name server, change a name servers name or change its IP address.

Those are the only operations that affect the actual contents of the root zone, they send a change to IANA functions operator, which is PTI and then PTI updates its own databases, does various internal check and processes. They send that request to the root zone maintainer to be implemented. The root zone maintainer has another database of the root zone, they make the root zone file and then they distribute it to the 14 root server operators. High level, that's how this process works.

Back to resolution. Ended our detour, we're back to resolution. The important thing to remember here, the whole point of that detour was DNS resolution starts at the root and in order to start at the root you need to know who the authoritative name servers for the root are. You can't discover that, you have to be told, that's that root hints file.

Let's so how this might work. Here we have the iPhone in the lower left. The Safari web browser app up. Somebody types in www.example.com in that Safari web browser. The Safari calls the stub resolver, this is one program basically, having it running a little piece of code that is the stub resolver and it says, I need the IP addresses for www.example.com.

The stub resolver meanwhile is configured to use a recursive name server at 4.2.2.2, that happens to be open recursive server that anybody can send queries to operate by level 3, it's just one of those IP addresses that's good to know. The stub resolver, it's configuration is very simple. The only thing it knows is where do I send queries and it says I know I should send them 4.2.2.2 It sends the query to 4.2.2.2 and says, I'm looking for the IP addresses of www.example.com. The recursive name server, much more complicated. It knows what to do.

To make our example interesting, we're assuming that it's just been turned on. It doesn't have anything in its cache but it does know, remember the root name server names an IP, so it can pick one of the root servers and ask it this question, it asks the same question that it got from the stub resolver and it says, what's the IP address of www.example.com?

We know what's in the root zone, it's only delegations to top level domain zones. What is the best answer that a root server

can give to this query? It can say, I can tell you about .com, it can say, here are the name servers for .com, that's the best I can do and that response is called a referral and now the recursive name server can follow that referral, it can pick one of those name servers. This is that set of NS records, it can pick one and query it.

C.gtld-servers.net is one of the .com servers, now the recursive name server asks the same query again but this time it directs it to a .com server and it says, what is the IP address of www.example.com and in this case, the best thing that the .com server can do, it has 131 million sets of delegations to different second level .com names, including example.com, it can say, here are the name servers for example.com, go ask one of them.

The recursive name server follows this referral, it picks one of the name servers ns1.example.com and it says again for the third-time same questions, what's the IP address of www.example.com but in this case, this name server is authoritative for the example.com zone and it basically raises its hand and goes, I know the answer. It does know and it says, here are the IP addresses that I have for www.example.com and now the recursive name server sends that back to the stub resolver which has been patiently waiting, which sends it back to the application and the web browser now can contact the

web server to get the web page and off we go. All this happens in a few hundred milliseconds or maybe even less.

I've talked early on about caching. Caching speeds all this up. If every time to answer a query like this you had to do all these queries that I showed, things would crawl to a halt, grind to a halt. That recursive name server can cache thing. It can cache to speed things up. After that previous query, it now knows the names and IP addresses of all the .com servers.

The names and IP addresses of all the example.com servers and it knows the IP addresses for specifically www.example.com, all those went into its cache. Now, let's look at what happens if there's another query immediately following that one. Let's say somebody types ftp.example.com, the stub resolver sends the query but this time the recursive name server says, wait I can go straight to an example.com name server and query it, get the answer and return it. You can see how much caching speeds things up.

I super high-level example here of -- talking all about DNS resolution side but there's also the domain name industry side, registering domain names and that's where you'll hear about the three R's, the registry, registrant and registrar and you can even throw reseller there if you want but these are the relationship between those entities.

The registrant is of course the person who uses the domain name, they buy that either from a registrar or sometimes a reseller in the middle and then the registrar gets the domain name from the registry. It's the registry that actually runs the authoritative name servers for the particular name we're talking about, usually a TLD. If we blow this up a little bit, here's a very high lever view of what a top-level domain registry would look like.

You see how registrants usually use a web browser to communicate with their registrar and by a domain name. Registrar's talk to registry's not always but often using a protocol called EPP which is specifically for provisioning domain names. The registrar can say I want to buy this domain name and then the registry, the main purpose of a registry to have a database of what domain names are available for that particular domain, usually a TLD we're talking about here. It can talk back to the registrar and say that domain name is available and sell it to the registrar on behalf of the registrant.

The other thing that a registry has to do is it has to make the information about the domain of the zone available on authoritative name servers. It runs authoritative name servers that answer queries about the TLD. It also makes information available about the domains via other means, protocols like

Whois or Rdap, these are people who want to know who owns that domain or tell me some more information about that domain name. Of course, I've shown how in the upper right, how recursive name servers are what are querying authoritative name servers on behalf of clients. Hopefully that sort of makes everything fit together a little more.

With that, we're done early enough that I could be happy to take some questions. For the benefit of the live stream, could I ask you to please go to the microphone.

UNKNOWN SPEAKER: I've got two questions here for you. The first question is about if I have in my IP address, how to I advertise the new IP address or how do I get to the IP address on to the DNS server? That's the first one and is it the register that does that?

The second question is, for example when you have these web servers, some of these web servers has M dot whatever it is and it has the main M web server, do they have the same IP address? Like the mobile one and the main one?

MATT LARSON: Let me answer your questions in the reverse order. The second question is, it depends, it could be either. It's often the

convention that you have a web server that starts with M for mobile and that could run on the same IP address, there could be a web server that servers one set of content if it sees the domain and another set of content if it sees M dot domain name. If I hazard a guess, it probably often organizations run them on the same web server but you could certainly separate them. Each of those domain names could have separate IP address.

To answer your first question, we didn't really do into -- we don't have time to into administration at all but if I go back to that zone file example, in the simplest example, there is an authoritative name server with -- it's hard to give answers that are good in all cases for DNS but I'll do my best here.

In the simplest example, there would be an authoritative name server that on its disk it would have this zone file, which would have all the information in it for a given zone and if you wanted to add say a new name with a new IP you would add one of those records down there, you would add the name and the IP and the you would have to tell your authoritative sever that, okay I've changed that file and it would load the file again and that's your primary server and then the secondary servers would download the authoritative zone and it would be available.

That's the very manual way to do it, what often is the case now is, you may have a third-party provider and you use a web page

to communicate with them and everything I just described sort of happens automatically on the back end.

Warren, what question could you possibly have?

WARREN: Actually, it's more of a comment then a question. You said you only had a little bit of time to go into the DNSSEC stuff, if people are interested in that there's the DNSSEC for Everybody Session tomorrow from 5 to 6:15 in Room 102, if you found this interesting, that has more details on the DNSSEC side of it.

MATT LARSON: Okay, thanks for pitching that. Is there going to be the skit? Yeah, so there's a skit -- there's actual drama that explains DNS.

GERARD BEST: A couple questions. One about IXP's, if you went back to the slide, the reverse tree, the upside-down tree. Further back, I think you're overestimating. I'm asking a question about the very, very early slide. How IXP's work in this configuration?

MATT LARSON: I'm sorry, I didn't quite hear the question.

GERARD BEST: How do internet exchange affect what's happening here?

MATT LARSON: I guess the shortest answer would be, it's sort of an apples and oranges thing. You can think of DNS as an application that uses the internet to send traffic from point A to point B and then below DNS would be the infrastructure of the internet itself, routers and leased lines and that's where IXP's operate. They're a layer below conceptionally then DNS. They contribute to the infrastructure that lets us move packets from point A to point B and from a DNS perspective those packets are queries and responses.

GERARD BEST: Okay. The second question has to do with CDN's. I think you had CDN's on one slide.

MATT LARSON: I don't think so but what's your question.

GERARD BEST: Just how do they work and how do they contribute to the resolution?

MATT LARSON: I'll kind of give the same answer, which is CDN's and DNS are getting apples and oranges things. CDN's are a way to speed up the delivery of web content and sometimes that's done, there's a tight connection to DNS because you need to resolve domain names of your URL's to correspond to data in CDN's as opposed to maybe on somebody's own infrastructure. It's kind of a complicated topic and I don't really have time to do it justice here. To summarize I could say that CDN's use URL's with domain names in them just like any other thing on the internet and DNS is used to resolve those names and those URL's.

GERALD BEST: Could you do be a big favor, there was a slide that had, I think it was X25. Is that CDN at the bottom there or is that something different?

MATT LARSON: Oh, CDS. That's the DNSSEC is the short answer there. That's for DNSSEC.

GERALD BEST: One more question, it has to do with any cast. How does any cast work and how does help to take us from 13 root servers to root server's instances being replicated across?

MATT LARSON: Any cast is happening conceptionally a layer below DNS in the internet's routing system. From the DNS perspective, from your recursive name servers name perspective it says, I want to talk to the root server at 192.1.2.3, from its perspective it sends that query.

If we look what's happening down a layer lower when the internet actually goes about delivering that query, any cast lets you as part of the internet's routing system, it's let you say, the network with this IP address in it, it doesn't just exist at one place, that same network is available multiple places, so the internet's routing system will go wait a minute I see that this packet could go to 10 different places and it chooses the place close to it and sends it there.

What that means as an operator is you have to configure a name server, listing at that IP address in each of those 10 locations. From a DNS perspective it's just, I'm sending a query to this IP address, down a layer at the internet's routing system, what's really happening is that IP address is available in multiple places

and the internet's routing system decides which one to take you to.

UNKNOWN SPEAKER: Could you go to the slide where the domain name server knows nothing and then it passes on to -- that's the one. Why does my recursive name server send the whole of name to the domain server not just to .com?

MATT LARSON: That's a very good question and the answer is, you could have a single authoritative server that knows about, for example, a parent zone and a child zone and in fact, that's how it use to be. In the early days, the root name servers not only were authoritative for the root but the top-level domain zones.

They were two separate zones but the same server, the same piece of software knew about the root zone and the .com zone, so because you said www.example.com, that one server could go, I know about the root, I also know about .com and so it could hand back the example.com referral. If that makes sense. It allows for an optimization. The recursive thinks it's talking to a root server but it's really talking to a root server and a .com server at once but because it asked exactly what it's looking for

that optimization can happen and that server can say, I can get back even more information.

UNKNOWN SPEAKER: Does that mean that the recursive name server than in its caching would not ask for another .com address if you passed back .com domain?

MATT LARSON: That is correct, it wouldn't have -- you're really putting me on the spot here. Strictly speaking though, as I think about this though, I'm trying to think though, if you get a -- what would a modern recursive do if it thinks -- if the band width is the root and gets back an example.com referral, I'm not sure if it'll think that's poison or if it'll accept it.

I don't know, without going into my laboratory and looking. It's possible a modern recursive server would say -- I think it's going to accept that referral and it's going -- which by-passes over the .com NS records. It's possible it's not going to get those .com NS records cached. I should be speculating in front of a room full of 50 people but I think that's the case.

CATHY PETERSEN: We're running out of time. We're actually at end. Sorry.

MATT LARSON: Can we take the two questions at the mic?

CATHY PETERSEN: Yes, we'll take two questions.

UNKNOWN SPEAKER: Could you kindly explain the process of DNS delegation and how you can take over that delegation and assign it over to another party per say?

MATT LARSON: Are you talking about at a particular level in the tree, like at a TLD level?

UNKNOWN SPEAKER: Yeah, we can take it at the TLD level or below the TLD level, the zone level.

MATT LARSON: Why don't we talk about that afterward and I can get more detail about what you're asking.

UNKNOWN SPEAKER: The first one is very simple. The presentation, how long are they going to stay for one to be able to download to access?

MATT LARSON: I think forever, right Cathy? The presentations never go offline. The slides are going to be available.

CATHY PETERSEN: The slides are actually already in the public schedule. They stay there forever.

UNKNOWN SPEAKER: The second question is to do with the IPv6, is it in use now?

MATT LARSON: Absolutely.

UNKNOWN SPEAKER: Is there any example you can use to show the simple structure of the IPv6? From the slides it looks quite Greek.

MATT LARSON: Not on the slides, I don't have it. An IPv6 address is 128 bits long.

CATHY PETERSEN: Thank you, everyone. Our next How It Works is going to start at 1:30, it will be about DNS Abuse. This will be a primer level session aimed at those who are unfamiliar with how DNS can be misused. It will be very interesting. Have a great lunch and see you at 1:30, same room. Thank you.

[END OF TRANSCRIPTION]