COPENHAGEN – How It Works: Internet Networking
Sunday, March 12, 2017 – 13:45 to 15:00 CET
ICANN58 | Copenhagen, Denmark

ALAIN DURAND: I will have to get my own fiber there, all the way through the ocean. People don't really do that. What they do is they lay out a bunch of fibers and then they connect them to create what's called a fiber path. So it's not dedicated fiber just for me, but I'm going to create a wavelength on the fiber from here, let's say, to the next central office, maybe to be green. And that will be converted into another wavelength, into another fiber from the central office all the way to maybe an oceanic station, but will be again converted to yet another color to go underneath the ocean, converted several times until it arrives to my office in Washington D.C.

From my perspective, I don't see any of that. All I see is simply a point-to-point circuit between here and my office. Speed of this? Well, 10 GigE here is fairly common. 25 GigE, 40 GigE is fairly common now. 100 GigE – sorry, GigE means Gigabit Ethernet. 100 gigabit per second is fairly common in service providers, and we are seeing multiple of the above if we want to go faster. There's some standardization effort to go to 400 gigabit per second, up to maybe a terabit per second.

This enables to have a point-to-point circuit between two distant cities, but the world is not really flat and we cannot simply think about having like point-to-point circuit with that technology from everywhere to everywhere. The whole idea is to say we are going to connect different segments that are built on top of the technology that I just explained, and create an IP network. And that's why I say that if you think that IP stands for intellectual property, you may be wrong, but you're actually in the wrong in the [right wrong] because we're going to explain really what this is about.

This is the Internet Protocol that is there to reconnect all those different circuits that have been put together. On top of this Internet protocol, we are going to put transport protocol that essentially are there to [fan] out packets to different application, because I can have multiple application running on top of the same IP address.

There are two most known protocol on the Internet: TCP – Transmission Control Protocol – and UDP for User Datagram, TCP being so far the dominant protocol.

Kathy, I would like to send you some data, and I want to make sure that you get the data. So let's say that I send you a bunch of data and I'm going away. How do I know that she has received it? Have you received it? Maybe, maybe not. That's what UDP

does. Send traffic, walk away. TCP, we are going to make sure that she actually receives the traffic. I'm going to send a bit of data – not a whole file but a chunk of a file, essentially a datagram that is maximum 1500 bytes. I'm going to send this to Kathy.

Kathy, here is a part of my message. Did you receive it? No. I'm going to try again, there was some problem in our communication. Kathy, here is the same part of my message. Did you receive it?

KATHY: Ack.

ALAIN DURAND: Now I know that she has received it. First one didn't make its way through. I don't know why. Maybe there was a problem, a glitch somewhere in the network. But second one went through, so I know that she can receive it. But at this point, she doesn't know that I know. All she knows is that I've been able to send thing to her. But maybe I was away when she responded to me and doesn't know that her message went back to me. So I have to respond to her. Yes, Kathy, I've received your ack. Thank you.

And when I do that, she actually knows that I've heard that she can receive my packet. So both of us now at this point know that

we can communicate together. That's really what TCP does. So I'm going to start sending some packets. Packet number one, packet number two, packet number three, packet number four.

KATHY: Ack.

ALAIN DURAND: Now I know that she received actually all those packets. Actually, no. She's going to say ack for packet number one, two, three, four. But at some point, there might be too much traffic happening and I will have sent packet five, six, seven, eight, nine but she can respond to me that she has only received packet number five.

KATHY: Ack packet number five.

ALAIN DURAND: I know that I have sent up to number six or seven or eight. She has only received five, so it means that I need to retransmit six, seven, eight. But why are those packets lost? Fiber cut? Not very probable. Most probably it's going to be somebody else is using the network and there's some congestion somewhere so I need

to slow down. So instead of saying packet one, two, three, four, I'm going to send now packet number six, seven, eight.

KATHY: Ack six, seven, eight.

ALAIN DURAND: Thank you. Now I know that when I slow down, she can hear me. Maybe I can go a little bit faster next time. Nine, ten, eleven.

KATHY: Ack nine, ten, eleven.

ALAIN DURAND: Thank you. So by doing this, now we know what is the maximum speed at which I can transmit my data to her. That's really what TCP does for you. That's why it's really useful when you're trying to transfer a file, a big file or a small file, because you want to make sure that the person on the other side will completely receive it.

Now, if you transfer not a file but if you have a real time communication like a voice communication and you're watching a movie for example or you're talking to somebody on the phone may be a better example, when you're talking about something to somebody over the phone, it makes no sense for

the network to pause and ask you to repeat five times what you have just said because the person on the other side is going to think you're crazy.

So it's actually better that we that we have observed on those real-time communication that we drop packet, that we don't retransmit them. So we don't try to have integrity of communication but we try on the contrary to make sure that communication happen even if bits are lost according to appropriate timeline.

When we describe on top of those protocols the content of what is being streamed, we have protocols [defined] at IETF like RTSP that essentially detail this is what the stream is going to be, this may be the speed at which it's going to be streamed, so get ready for that, and this is how you can control me to say pause or other. So that's the layer five.

But for more general communication, what you need is to describe exactly what is being transferred, what are the capabilities. And there have been a number of protocols description that have been proposed over the years. It goes from yes, anything you like. Could be some fixed width text strings. ASN 1 was very popular at some point in time, it was binary encoding.

Then folks tried to use XML which is a markup language, and the stuff that the "cool kids" are using now is called JSON, which is kind of a variation of XML. It proposes a dictionary of items, and I have on the screen here an example of how you can have a JSON description of a menu.

So this is a presentation layer that explain how the data is going to be structured and formatted. But none of this is really useful to kids or to the general population. What they care about is watching a movie on their iPad or reading something and going to YouTube, going to Facebook and things like that. But what we need to think, to keep in our mind as technologists when we talk about all those layers, none of them matter if you cannot deliver the application to the user.

So the application layer protocols are HTTP for Hypertext Transfer Protocol, or the secure version of it, HTTPS that essentially encrypts things end to end. Those are the seven classic layers, but there are more.

Next slide. Oops, what have I done here? Here we go.

There's another layer which is a financial layer. If you do all this, your user happy, but you don't collect any money from this. You're not going to do it very long.

Like any business, you need to make sure that it's a sound business and you have some kind of return on investment that matches your expectation. So every time you build a network, just doing it for technology's sake will never work. It has to be associated with some kind of a business plan that makes sense.

On top of that, there are the political layers, and this is where you are over there today in the ICANN meeting, and many of our standard bodies in other meetings that are trying to understand how these things get together.

So that's my approach of the seven layers, extended from zero all the way to nine. This is just the foundation, and I'm sure most of you already know all of that, I'm not teaching you anything here. But let's go now to really the meat of this presentation, which is naming, addressing and routing. And all this started about a week ago when I had a serious tooth problem. It was so bad that I really needed to do something. I couldn't wait to go back home, take a flight for 25 hours.

So I went to see Kathy. She was there at the meeting. I said, "Kathy, my tooth really hurts today. I don't know what to do. You have been there before, you have lived in this country, in this city. Is there a dentist that you can recommend?"

KATHY: My dentist is Dr. Johnny Walker.

ALAIN DURAND: Okay, so now I know that I need to find Mr. Johnny Walker. She just gave me the name of a dentist, right? I don't know where he is, I don't know how to go there, but I know his name. That's the first thing I have. So what's a name? If you look up the dictionary of a name, a name is a word or a set of words by which a person, an animal, a place or a thing is known, addressed or referred to.

Essentially, you have already two things in this definition. That is a way to address somebody or to refer to somebody. What Kathy just did here, she referred me to a dentist, Johnny Walker. And now we can have a conversation about this Johnny Walker. It's not any dentist in the world, it's just this particular one, this Johnny Walker. So if I know your name, I know who you are. I know that this dentist is Johnny Walker. That's the first thing I would like you to remember form this tutorial. Only three things I will ask you to remember. That's number one. If I know your name, I know who you are.

As I just said, we can use his name to have a meaningful conversation about that person. So we can talk to someone – I can talk to Kathy – or we can talk about someone, we can talk about Dr. Johnny Walker. So, did you really like Mr. Johnny Walker? Was he a good dentist?

KATHY: He was a good dentist.

ALAIN DURAND: Okay. So you recommend him.

KATHY: Yes, I do.

ALAIN DURAND: So we just had really this conversation about this person. But names have scopes.

For example, my name is Alain Durand. It's spelled with an I because that's the French spelling. In my family, I'm the only one who's named Alain. But when I was in elementary school, Alain was quite a popular name. There were about six or seven kids every year who had the same first name as I had. So when the teacher stand up and was really angry and said, "Alain, go to the whiteboard!" All of us looking at each other says, "Which one?" And then the teacher got even more angry because nobody was going to the whiteboard, or blackboard at the time. So we need to qualify this. It's only when the teacher said, "Oh, Alain Durand, go to the blackboard" that I knew it was me, not the other ones.

That's why this notion of scope is really important. But it's still not enough to do something, really. Okay, I know I have a tooth problem, I know that Dr. Johnny Walker in town is a good guy, but I have no idea where he is. So what do I do? Kathy, I think you convinced me. I need to go see Dr. Johnny Walker. What's his address?

KATHY: His address is 125 Root Canal Road

ALAIN DURAND: Thank you. Now I know where Mr. Johnny Walker is: 125 Root Canal Road I can go there. I can map the name to an address. What Kathy did was simply go to her address book, her rolodex or whatever technology we use now and looked up Dr. Johnny Walker. The result was an address. So when we use the Internet technology, the DNS is doing just that. It's taking a name, looking it up and finding an IP address.

So now I have his address. Where is my dentist? Going back to the dictionary, an address is a particular of a place where someone lives or an organization is situated. Second thing I would like you to remember. First thing was if I know your name, I know who you are. If I know your address, I know where you

are. I know that Mr. Johnny Walker is 125 Root Canal Road and I can go there.

So, going to make a little detour before going to my dentist. As I mentioned, I live in Washington D.C. There's a very famous place, a small house, white, that is in a nice neighborhood and everybody wants to go there for some reason or another. The address of that place is 1600 Pennsylvania Avenue Northwest, Washington, D.C., 20500-0003, USA. That's the address. That's the street address. Full street address. There's hierarchical structure to this, and you have to read it from the end, from the right to the left.

Most important part is USA, then D.C. D.C. is not a state in the U.S. but it's the District of Colombia. It's kind of like a state but it's not. Northwest is one of the quadrants of Washington. There are four quadrants: northwest, northeast, southwest, southeast. So it's the northwest quadrant. Pennsylvania Avenue is the street, 1600 is the number on that street.

But that's a postal address. Not all addresses are organized like this. For example, in the U.S. you have the 1-800 toll-free phone system numbers. So you want to call the number and it's reverse charge. You don't pay to call, but the person that's being called is paying for the call. When you call a 1-800 number, you don't know if the person picking up that phone is in the same city, in

the same state, the same country. Maybe it's somebody in India who's picking up the phone.

So the number is associated with a service, not associated with actually a person. And we have something similar in the DNS. For example, in the early days of the Internet, all machines had a name and people chose names of dogs, names of cats, names of wine, whatever they like, to identify their machines. But when we started to add services like a mail service, a file transfer service or a web service, we didn't want to associate this with a name. We associated with a set of letters that identified the service. So it became www. and the name of the domain instead of being Romanée-Conti. the name of the domain. This is one of the very famous French wine.

Cellphone numbers have the same property. When you call somebody on a cellphone number, if you have normally a phone number, the first part of the phone number is the area code and that tells you globally where that phone number is in the country. If you have a cellphone, you are not at that particular location. My cellphone number is a number associated with the Washington D.C. area, but I'm here in Copenhagen today. So looking simply at my phone number doesn't tell you where I live or where I am.

IP addresses are the same. If you look up an IP address, simply by looking at the address you do not know where this address is. You don't know if it is in the U.S., in Europe, in China, in Africa, in Australia, you don't know that. What you can do is to look up some other directories that will say, "Oh, this address has been allocated by an IRR that is in Europe. That's the RIPE database. But there, you can see maybe it had been allocated to a country or service provider in that country. Maybe the service provider is using it somewhere else.

So there are services that have been built by people who observe a network and say, "Oh, this particular address is actually in that city." Those are geolocation database, and accessing those geolocation database usually is a paid service where you need to pay them to access the data.

So the point I'm making is sometimes there is a structure in addresses, sometimes there is not. As I mentioned about names having scopes and having multiple Alain in elementary school, the same thing with addresses. For example, Paris. If I'm here and I'm asking you how far is Paris, most probably you will say it's about an hour, hour and a half by plane, maybe 10 hours driving. We think Paris, France.

In the U.S., there are 29 cities called Paris. Sometimes I make a joke with my kids. I say, "Oh, let's go to Paris this afternoon and

we come back for dinner," because there is a little town in Virginia called Paris and maybe like 20 houses there. There's a restaurant, that's all there is. So similarly with an address, you need to have a scope and you need to qualify exactly which one you mean. So in Paris, I would have to say, "Oh, I'm going to Paris, Virginia" instead of Paris, France, whereas the famous movie talks about Paris, Texas. Same thing, there is a small city there.

So, similarly with names, I can use addresses directly or indirectly as a reference. For example, I can send a postcard and I'll write the address on the postcard and the postcard will go where it's supposed to go. Or we can talk about an address. So Kathy, 125 Root Canal Road, is it a good neighborhood? Can I go there, or do I need to take a taxi?

KATHY: You can go take a taxi.

ALAIN DURAND: Okay, so we just had a discussion about 125 Root Canal Road. That's the second property of our addresses. But that's still not enough to communicate. I still don't know how to go to 125 Root Canal Road. So back when you send a postcard, let's say that you sent a postcard, a nice postcard and you send it to 1600

Pennsylvania Avenue northwest, Washington D.C., 205000, USA from here, or anywhere else. Then you want to say, I don't know, whatever you'd like to say to the current residents of the White House.

Why will it get there if you post it here in Denmark? It will get there because there's an agreement between the Denmark post office and the U.S. post office so that they know how to exchange letters, packages, parcels and postcards. There's a cooperation in place between those organisms, and you can be anywhere in the world, there's cooperation in place. Sometimes those are bilateral agreements, sometimes they are multilateral agreements.

For example, I was in Hanoi in Vietnam just two weeks ago and we were talking about prisoners of wars in 1970. American prisoner of war receiving packages from their family. How did the package arrive there? Because U.S. was at war with Vietnam. Well, came through Russia, because there was a bilateral agreement between Vietnam and Russia, and they had a bilateral agreement between Russia and the U.S., and that's how the package made it.

The point I'm making here is it only works because there's a number of those collaboration in place. So, now I want to go to my dentist, 125 Root Canal Road, Mr. Johnny Walker. How do I

go there? Well, I need to find a route to go there. Or if you have a British accent, you may say route. So looking at the dictionary, a route or route is a way to get somewhere, a starting point to a destination.

That's the third thing I want you to remember. So backtracking, first thing I want you to remember is if I know your name, I know who you are. Second thing was if I know your address, I know where you are. And the third thing now is if I have a route for you, I know where to go. I know how to get there. Those are three different things. Name, who you are. Address is where you are. A route is how to go to your place. Those are the three different things.

So let's explore route a little bit. Obviously, it's built on the internet. Remember this address the U.S., what I said, postal address, you have to read it from the end. That's the same thing here. So when you want to go somewhere, you drive your car and you want to go to, let's say from here to Paris, you're going to follow road signs. Those road signs to tell you go on this highway or go on that other highway have been put in place before you start your car, before you start driving your car.

Same thing on the Internet. The equivalent of the road signs on the Internet have been put in place before you start sending your packets, before you start sending your traffic. They have

been put in place by routing protocols. The way they work is the destination is going to send an announcement saying, “This is me, you can reach me there.” So if you look at the diagram here, this is a point in the [inaudible] that says, “You can reach me there.” And who does he send this to? To his provider. The first node in yellow. That node in yellow is going to advertise to his peers, “If you want to talk to this guy, send me your traffic. I know how to talk to that person.”

The second hop – I don’t know if this is a pointer that works. Maybe not. Okay, the second one that’s one towards the bottom is also going to also talk to his peers and says, “I know somebody who knows how to get three.” So this is about, “I know a guy who knows a guy how to get there.”

And all this is going to be propagated across all the Internet, all the way to the source. So the source essentially is going to receive messages like, “I know a guy who knows a guy who knows a guy how to get there.” That’s all it is. You never know really things. All you know is there’s a guy who knows a guy how to get there.

So when you send that traffic, you just reverse this and you send the traffic to the first hop that has told you, “I know how to get there. I don’t know the details, I don’t want to know the details ,but I know how to get there. Send me your traffic.”

You send it to them. That router is going to do exactly the same thing. He has remembered when he built all those routes who sent him announcement saying, "I know how to get there and I'm somewhere closer than you." "Okay, I'm going to send the traffic. I'm going to trust you. Send the traffic."

All this will go all the way to the last router, which typically would be the service provider router that's going to take the packets and send them to the destination. Same exact thing on the way back.

So now I have a path to go to the dentist. Now I've arrived to Dr. Johnny Walker, we can take some office tools and do something about my tooth on Root Canal Road. Thank you, Kathy, it was a good referral.

So, that's all I have for now, and I'm going to hand it over to Geoff who's going to talk about something maybe more in-depth. We will take questions at the end of the second part.

GEOFF HOUSTON: Good afternoon. My name is Geoff Houston, I actually work with the numbers registry, which makes me logically the right kind of person to talk about the root system or the domain name system. Not. But I do know one or two things about it, and Alain asked me to sort of go through at an introductory level what is

so special about the root of the DNS. Why do we have this sort of special advisory committee for the root servers? What do they actually do in terms of the way the name system and the DNS actually works? That's what I want to talk about here.

The namespace is kind of special. If you think about other naming spaces, like Alain's dentist, that physical address normally has some structure. So one would have a street number, followed by a street name, followed by a locality, followed by a city, followed by a country. In other words, it's got an order to the way information is presented, and there's a structure to how we talk about postal addresses. Domain names have a very similar structure, and the way their structure works is that it's a series of labels separated by an ASCII dot.

So in my example, www is a label. It's just a string of characters. Originally – and for a very long time – the characters you could use in labels was the so called LDH rule. You could have letters, a through z – I'm Australian – you could have digits, zero through nine, and you could have hyphens. With one qualification: the label had to start with a letter – which worked well until 3Com came along. The company 3Com uses the digit 3, they complained bitterly, 3Com got its domain name 3com.com. Now you can start with a digit too. Okay, fine.

That worked well until we started thinking about all the folks who don't use Latin styled characters. Japanese, Chinese, all these other scripts, and so now the rule is much more convoluted than that. Much more convoluted. It basically says it's a series of characters separated by ASCII periods. And there's an order. There's a very particular order, because this namespace is a hierarchical namespace.

In some ways, if you reverse the postal address, country, city, locality, street, street number, you see that every single item there progressively refines that location. The same thing happens in the domain name system, and that naming system actually goes backwards. The most generic thing – and everyone has it – is the bit we always leave out: the dot at the end. Every name ends with a dot if it's a fully qualified name.

The next label in, in my case com, is one of the so called top level labels, and there are currently 1522 – give or take a few – of these labels that are actually being instantiated in the domain name system. Some of them are country codes, which is why we have a ccNSO here somewhere. Others are more generic. It started with .com and .net and .org, and kept on going. And we now have I suppose about 1300 of them.

Below that, each of those registrars and registries populate. So .com is a generic name, and Example is sort of being given out

by com. And www is being given out by whoever owns Example. So we write it – oddly enough – backwards, from right to left. It was a 1980s thing. The English wanted to do it the other way, the Americans wanted to do it this way. I guess the Americans won that particular fight. You had to be there for it to be important.

The other way of writing it is actually as I've done on the right-hand side there. It is indeed a hierarchy, from the root, to .com, to example, to www. So it's a sequence of labels. That's a domain name, but it also is a bit of a hint. In the same way that a postal system, a postal address is a bit of a hint about how to get there, get to the right country, get to the right city, find that street, look for the number. It's an ordered search.

Well, the same thing happens here in the domain name system. Because the domain name system is actually one of the most amazing databases we've ever built on this planet. It's a single collection of data that is widely distributed. The way it works is that in each point of that hierarchy, there's a service that talks about all the names in the next immediate level down.

So that root is actually a zone, a point of information. And what it can tell you is all of the names at the next level down. Can't tell you anything else, just the names at the next level down. And if you look at that level, in my example .com, there's a zone file, and in that zone file – hopefully – is Example. Example is being

further delegated, someone else is actually operating that part of the database. And if I look inside example.com, I should hopefully find my word www.

Now, many other folk use the word www, but no one else has it inside Example. And many other folk might use the word Example, but no one else has it inside .com. So that's what makes that particular name unique. And what it reflects – in the same way the postal system talks about how to get there, that name actually talks about how to find it in this massively distributed database.

I don't know how many lookups we do per second in this database by the way, but my suspicion is that it numbers in the 20 million per second, 100 million per second. The DNS works extraordinarily hard, and the fact that answers come in milliseconds is a modern-day miracle.

So, what do you use this name system for? Typically, the most usual thing is to map the name system to a protocol address. Because down there on the wires, we don't route names. Packets don't have names in their header. Down there on the wire where your packets are flowing, it has one thing: either a 32-bit or a 128-bit binary address. Ones and zeroes. What is the binary address of example.com? I don't know. You don't know.

Let's ask the DNS. What's the binary address of Google.com? I've forgotten, let's ask the DNS.

No matter what you do on the Internet, if you want to send packets there for whatever reason, the first step always is you ask the DNS. Now, we don't use the word "ask" because we're just weird. We use the word "resolving." So this is resolving a name to find some associated information. Let's just have a little bit of a look at how this works, because what we need to do is to wander down the tree.

Because the root has no idea about example.com, but I start by asking any one of those root servers, "Hi, can you tell me anything about example.com?" And what the root will say is, "Look, I'm sorry, I have no clue. But I do know that .com is in my zone. So tell you what, I'm going to refer you to someone who knows all about .com. So here are the computers, their IP addresses, that can answer questions about .com. Go ask them."

So my resolver does exactly that. Same question. www.example.com, what's its address? And it says again, "I've got no clue. But I'll tell you what, I know about Example, and here are the nameservers for Example." So the third time, I'm kind of lucky, because that server for example.com has the label www as a terminal and I can get back its IP address.

Every single time, that's the process you go through in theory. So if you think about it, every single time a query starts at the root – and there are a lot of queries – if the root didn't exist, everything wouldn't work. Now, it's not as bad as that, because we use computers and computers have memory. So when I get an answer back from the root, I remember it, and the ancillary information says, "Tell you what, when you get an answer, keep it in your computer memory for some period of time." Sometimes hours, sometimes days.

So it's not that bad, but when the cache expires, when I lose my memory – as I will – I'm going to have to ask the root again. So the root is special. If we didn't have it, we wouldn't have a DNS that's working. If all the root servers – God forbid – turned themselves off, the DNS would slowly die as the caches expired, and there'd be nothing. You couldn't resolve names.

So how do we use these root servers in this root service? Every single resolver needs to sort of start itself up in the world. It's this single piece of software without any particular initiating state. And so what comes from the vendor of the software is a hint file that says, "You might try asking these people. Send a DNS query packet to them, and hopefully one of them will give you back the current list of root servers."

So off goes a priming query. Very important query. Back comes a list of IP addresses that service the root zone. At that point, the resolver is now kicked in. It's able to now answer questions using that top-down search. So that gives us the first real role of the root. If it can't answer priming queries, we're all dead. Virtually speaking.

So the first role, answer those priming queries. Answer them quickly, answer them accurately, answer them well. It all starts by asking the root, and as I said before, in practice, they don't. Once it learns, it remembers, so caching is extremely important. That's why it works so well, that all those computers that help resolve names remember the answers for a period of time.

So once I ask the root for the root servers, I won't ask for another day or so. However long the root server said, "Keep it in your brain," I will. That cuts down the query rate quite significantly. So if you actually look, once you do the one priming query, you probably won't ask the root again for some time.

The next issue is I don't ask the root for everything it knows. I don't ask for all 1500 names. I ask it a question about a name I'm trying to resolve. So every time I get as a resolver a new name, my.favorite.car.is.a.maserati, might well be a name. I don't know. But the one place I start is the root, going, "Do you know anything about this name?"

Every single previously unknown name that your resolver encounters, that you type in on a browser bar, send a question to the root. If it's not answered from your cache, it goes to the root. So the resolver queries the root, and what comes back is, if you will, the answer that is put in my cache. The corollary is that once I know that answer, I don't ask the root anymore.

So it should come as no surprise when the root servers say, "Most of our queries are for unknown, nonexistent names." That's the truth. But the root servers are really garbage operators, and what they're doing is spending a lot of time and effort answering queries about names that never existed, the garbage of the Internet because the ones that do exist find their way into caches and don't get continually re-queried.

So the role of the root server is to answer when the caches can't help. New names that haven't been remembered. So root servers between 75 and I think up to about 90% sometimes of the queries at the root are garbage. So yes.

The next thing, root servers must answer. It's almost an ethical or a moral statement, but that's the idea. They can't play favorites. They can't say, "No, I'm not going to answer Alain today. I don't like him."

You can't do that if you're a root server. Every single question gets answered if you possibly can. Because they don't know why

you're asking. They have no idea when you ask the question what you are or why you're asking it. So because they don't know, they have to answer everyone. And the root servers – and there are a lot of them around the planet – always give you the same answer, no matter which one you ask.

So they're promiscuous askers, a fact that has been exploited by Denial of Service attacks, because the roots can't stop answering. It's their job to try and answer every last query. So if you present a question to the root, the task of the root servers is to make sure you get answered, and the other task is to make sure no matter which root you ask, no matter how you get the question to a root – and there are a lot of servers – the answer is always the same.

Those root servers need to synchronize what they're doing and always come back with precisely the same information. If your resolver can't reach any root, your resolver can't answer any questions. Once its cache dies, it dies. No more questions.

Now, there are two ways for this to happen. One, you pull the wire out. The resolver is now off the net. Oops, it can't answer any more questions. That's pretty obvious.

But the other way – which is more insidious, and these days incredibly difficult – is to take all the root servers out. To launch a truly horrendous attack that would simply overwhelm them

and kill them. And if you manage to do that, all of us have a problem. So the root servers must be available. Not all of them, but at least one of them somewhere need to be able to answer questions. Preferably all, preferably all of the time, because every query starts at the root, and once the caches expire, they have to ask the root again.

So if the root dies, there is no more DNS. So that aggregate system is one of those "Five nines isn't good enough, it's 100% of the time available." Somebody has to answer somewhere somehow for every single query for this to keep on working.

How fast? We don't know. We don't know how fast it should be. Quick is good, but don't be – [I'm like], billionths of a second? Yes, physics gets in the way. Milliseconds? That would be pretty good. Weeks? Not interested.

So somewhere between taking a few weeks to answer and answering that nanosecond is the right answer. It should take as long as it needs, but no longer. And that's about as good as we can phrase it at engineering level. So it's good that they're quick, it's good that they answer promptly, but it's not particularly required. But they should answer relatively soon.

Your software that you run normally runs its timeout around one to eight seconds. So when your application says, "Is this a name? What's its address?" Typically, it starts drumming its little virtual

fingers on the desk. By the time about eight seconds goes past, up comes a gray screen or whatever the screen that says, “I’m sorry, that name isn’t there. I don’t think you’re on the Internet anymore.”

So typically, roots need to respond within the timeframe that the applications we use. The timeframe of the applications, around one to eight seconds for most applications most of the time, which means the root needs to also answer promptly, whatever that means. Don’t take forever, not a good idea.

We have 13 root server letters. A all the way through to M. And for some time, that actually corresponded to 13 pieces of silicon in 13 locations, which was okay. I’m not sure there were any incidents where all of them were out all of the time. Maybe others can correct me. But there were a few close incidents.

We then discovered and got used to a concept called Anycast. Anycast is a strange trick where you put up multiple pieces of machinery, multiple parts of the world, to give them all the same IP address. So when I go to 1.2.3.4, the routing system will probably get me to a machine somewhere close to Denmark. But when I fly home to Australia and go to the same address, if that is Anycast, I might go to a different machine, possibly in Australia, maybe in Singapore. Somewhere that was close to where I am.

Almost all of the root servers these days use Anycast, and rather than having 13 machines, there are now some hundreds of machines all running identical copies of the root. They use 13 v4 addresses and 14 v6 addresses. I think that's the case. And those addresses are Anycast across the Internet. Anycast is quick, because the resolvers, the root servers can be close. And it also, oddly enough, gives us greater resilience, because if some person says, "I'm going to attack 1.2.3.4" and they happen to live in Australia, that might be a problem for that Anycast instance for Australia, but here in Denmark wouldn't notice a thing.

So you tend to localize the problem with attacks by simply Anycasting the service and simply putting the service up in multiple locations. So Anycast is now – if you will – part of the fabric of the way the root system operates. So I think I've touched on this already. I should have looked at these slides earlier, I'm sorry. There are – as far as I can tell when I did these slides a week ago – 1530 distinct labels sitting in the root.

Now, there are many more names that don't exist. Those ones do. And as I said before, if you look at the roots, the majority of the queries they get, they simply get answered with a response that says, "That's not a name. Forget it." And that's somewhere around – as I said – between 75% and up to as high as 90% of the queries are simply names that don't exist.

So which root do you use? You actually can't send a query to a root. There's no such generic address. So your software will pack a letter: A, B, C, whatever. Some pieces of software try and pick the letter that they think is fastest. They'll send a few queries, measure the time and latch onto the one that they think is closest. Other DNS software goes first query to A, second query to B and just round robin, sending a query to each one in turn.

It doesn't matter. It honestly doesn't matter. It's not clear that one behavior is better than another. The whole reason why there are 13 is that if one doesn't work, there's another. So if you wait for a while and you don't get an answer, try another letter. And that's why there are 13 letters. You can do this up to 13 times before you conclude, "Oops, I'm really in a pickle." So if there's no response, you just switch to another letter. That's why we have that number of distinct Anycast constellations.

The only change in the root server system over the years has actually been the introduction of Anycast and that took us some years to figure out it was a safe change. But it's not going to stop there. We are seeing a huge amount of Internet traffic out there. As we populate today's world with what, seven billion mobile devices, seven billion things, and [inaudible] constantly predict we're going to have 20 billion things in a few years' time, all of those things use the DNS. And networks will grow, the DNS query

rates will grow. There is more traffic, more queries. This is part of the problem.

Do we just keep on building bigger machinery, or will we try and get a bit smarter? Do we actually go, "Well, that's okay, we'll just add more machines, add more bandwidth?" Or do we actually look at the properties to see how we can make life better?

One of the things that happened – and it happened five and a half, close to six years ago now – was that we made one step that I think is truly fundamental and very important. We introduced security into the DNS.

When I asked this IP address what's the address of Geoff.com, I get back an answer from the root. How do I know the root gave me the answer? If I'm a suspicious person, I could be very worried, because there's nothing there that says, "That's the right answer." It's just an answer, and so what happened in the standards world is that they introduced digital signatures and cryptography. A very similar form of cryptography to what you use when you go to your online bank or any other form of Internet security.

What we're doing is using the RSA algorithm and doing signing across the DNS. If you have the master key – which is public, everyone can have a copy – then when you get an answer, you can actually ask for the credentials and apply your master key to

your credentials. If they unlock correctly, that's the real answer. No one is trying to mislead you.

If you get the credentials and you can't unlock it, it doesn't work, perhaps you shouldn't go to that address. Perhaps you're being misled. Because if it's signed and your key won't open it, you've not got the real answer. You've got something else. We can use this in various ways to make the system better.

Why do we have 13 root name servers and not more? Well, a bunch of reasons. But if the answer is signed, I don't care where I got the answer from. You can tell me the answer, and if you give me the right credentials and I can unlock it with the key, that's the real answer. You don't need to be a root server to tell me it, you just need to give me enough credentials that I can check that it's genuine.

When I give you a piece of colored paper and say that's money, typically you'll accept it if it looks okay. The same kind of thing could be used in the DNS. Anyone can give you that answer if they also are able to give you the appropriate credentials, the digital signatures. And so some ways, instead of making those 13 root servers bigger and bigger, maybe we can look at the larger infrastructure and say, "Well, maybe anyone can be a root as long as they're able to deliver the signed data."

And that means the resolvers, the end people like me, we just have to be a little bit suspicious and apply our key to the answer. If it unlocks, it's good. It's the real answer. Because my key unlocked it, it's the real McCoy. I didn't need to ask a root to get the right answer. So, DNSSEC gives us some hints about how we can get rid of those 13 single letters and create a broader environment that allows many folk to be authoritative about the root zone, as long as resolvers are willing to use DNSSEC to validate the answers that they receive.

That's a fascinating direction as to how to make the Internet bigger without necessarily making the black holes in the middle even bigger holes than they are. A number of places to do this. The IETF has been looking at it, the Internet Engineering Taskforce. They published technical documents, they called requests for comments. They're not interested in your comments, they're standards and this one is a standard. They number them from 1 to – I think they're up to about 8000.

This is a pretty recent one, 7706, talks about how you can configure your local resolver – and if you're an ISP, I'm talking your language at this point – how you can configure your local resolver to serve the root as if you were a root server. So all your customers then get a root service from you rather than going further out and around the Internet, increasing your resiliency and reliability. And because you're using DNSSEC, you've got to

serve the real thing. You can't lie and customers should be checking you anyway.

And there's another way too, which is we're just getting around to, and it is fascinating. Because normally when you ask a question of the DNS, it tells you, "Well, that name doesn't exist. Try me again." And it's kind of a guessing game. A, B, C, D. But in DNSSEC, we actually send back a subtly different answer. There are only 1530 names, and between beer and Bentley, there's nothing.

So what if I asked for the name beg? That's between beer and Bentley. If I get back an answer that says, "Nothing between beer and Bentley is good," when someone asks me for beg that sits in that range, I go, "Hello, I've been given a range. That's not a good name." So instead of simply using single names, we now start to talk about ranges of names and scopes.

This makes the entire root service really resilient against attacks, because the most common forms of attacks we're seeing right now is that folk generate random names. Random string, ask the DNS. New random string, ask the DNS. And the resolvers faithfully carry the question, and if you can do ten questions a second, so what? If you can do a million, we're all worried.

One of the ways to stop that form of attack is exactly this approach. Using DNSSEC to sort of say, "This whole collecting of

names between these two points do not exist." Any random string does not exist. The answer is always, "No, I don't need to ask the root." So there's a lot of power in this, and hopefully it will be implemented in the coming months – too optimistic – years. And it will certainly make the DNS a better place, there is no doubt about that.

There's more we can do, and like everything though, it does depend on open data. If we can peer inside these interactions and actually understand better what the software is doing, we can make it better. How queries are spread, how they're answered, what's a better way of doing this? We're trying to introduce v6 in all parts of the Internet, including the DNS. How do those two protocols interact? What's going on between the two? What location parameters are used? How does DNSSEC work today? How many folk use it?

All these are really good questions, and the more we can gain insights into the way the DNS actually works, the more I think we can make a more resilient and stronger DNS in the future. So my plea is, as ever in the research community, I need data. Open data is your friend and my friend, because with it, we can do better answers. So that's a brief introduction to the top of the DNS, the root zone as it's seen from you and me, from the point of perspective of the resolvers. If there are any questions, I would be happy to try and answer them. Walking microphone?

UNIDENTIFIED MALE: Thanks, Geoff. I'll relay the mic around.

GEOFF HOUSTON: Thank you.

[SUBHASH SUBRAMANYAM]: My name is [Subhash Subramanyam] and DNSSEC is implemented at the root server level and at the registry server level, but the average user, when he sends an inquiry, does his computer always look for a validated answer? In other words, are all computers uniformly – even phones and other devices – equipped to look for a validated answer?

GEOFF HOUSTON: That's a really good question, and thank you for asking it. I love it. We've done a lot of work in measuring this and interestingly in APNIC, I've actually enlisted Google Ads and using the ad network to help me understand how the Internet looks from the edge inward. Because ads can have scripting in them, and the scripts can do things.

I started measuring the amount of v6 that is out there, and around the 10% of the world's 3.6 billion users are able to retrieve using v6 today. Great achievement. In America, there's a

lot of it. Geo in India has just turned it on. 44 million users. It's happening. 10% of the world. Great. We also looked at DNSSEC. How many users will validate a name if it's been signed? 15%. 50% more than use v6. 15% of the planet will validate a signed DNSSEC answer.

What's actually weird is that double that number start the validation process. So, one third of the Internet will try validation, but half of them if they see a bad signature go, "Oh, bad signature? I'm going to use a resolver that doesn't validate, because I really want the answer." And it's kind of, "No, that wasn't the way it was meant to work." So there is an awful lot of DNSSEC validation out there, far more than v6. So I wouldn't get dispirited about this, we are very close.

If you run a nameserver, if you publish names, think hard about signing it. It protects you, it protects the folk who are using your name, and a lot of folk are validating already. So it's a really good thing to get onboard.

Tutorial at the end of today, DNSSEC workshop on Wednesday. There's a lot of DNSSEC here at this meeting. By all means come to the tutorial this afternoon for an in-depth look at it.

[SUBHASH SUBRAMANYAM]: At least do the ISPs and others who maintain cache servers, do they look for a validated response?

Another question is, most of my queries as a user gets answered by the ISP or his cache server. What if the data in the cache server is misleading and I'm misdirected to go to a malicious site? What safeguards are taken? So this is where it counts.

GEOFF HOUSTON: If the name you are looking up is not signed, you and I have no idea if you're being misdirected or not. There's no truth out there. You can't sort of tell, can you? If it's signed, the signatures might work if you're being misdirected. So that's why signing is always a good idea.

Now, interestingly, almost all of your resolvers ask for credentials. 80% of all the DNS queries out there start the DNS process of validating it. So if you signed something, all of a sudden you're delivering your credentials to almost everyone. But quite a number of folk who run resolvers go, "I'm not actually going to validate the answer." I've equipped your engine with a turbocharger and you're not going to use it. It's kind of, "What are you doing here?" So around one third of users actually validate the answer, and half of them go, "Oh, if that's a bad signature I'll just try someone who doesn't:"

Most of the abuses and problems in the DNS can be solved immediately if folk A, signed their zones, and B, resolvers checked those signatures. It's as simple as that. It's not hard. Go to the tutorial this afternoon, it's really good.

UNIDENTIFIED MALE: Any other questions for Geoff or Alain? Alright, well, thank you very much for coming.

[END OF TRANSCRIPTION]