DNS Deep Dive And NetDB

Drew Saunders, IT Infrastructure

2022 IT Unconference

DNS Deep Dive (and NetDB)

- A walkthrough of the entire DNS process
  - Assuming NOTHING has been cached!
- From root DNS servers to the final reply
- URL vs DNS
  - CNAME vs. URL Redirect (Vanity URL)
- How does this tie into NetDB?

The Scenario

- You’re at home, using Xfinity/Comcast
- You have absolutely no idea how to configure anything
- Mysteriously, your computer and every DNS server involved have lost all cached data
- You click on a link that enters https://itcommunity.stanford.edu/unconference/ into your browser’s Address Bar
  - **NOT** https://itunconference.stanford.edu (we get to that later)

Your computer

- Using DHCP, your public IP number is 24.125.26.27
- You are using 75.75.75.75 and 75.75.76.76 as your Anycast Recursive DNS Resolver servers
  - What do those last five words mean anyway?

DNS Definitions

- DNS is the “Domain Name System,” which can be thought of as the “411 information system” for the Internet
  - How 411 worked in the ancient times: You call 411, ask a real person for the phone number based on a name, they give you the number, you write it down, hang up, and then call that number
- DNS turns words into numbers. People like words, your computer, at 24.125.26.27, needs to route an IP packet by IP **number** only
- DNS can also hand out useful information, but we won’t go into that much

DNS Server Types

- There are 13 **root** DNS servers, these hand out the information for Top Level Domain servers. Typing "`dig`" without any arguments will show you the list. These don't change very often, maybe every 4-5 years or so.
- Every TLD has a set of **TLD servers**, sometimes referred to as gTLD for the "generic" Top Level Domains (as opposed to country code TLDs).
  - .com, .net and .edu use the same 13 DNS "gtld-servers.net" servers (a.gtld-servers.net through m…)
  - .org uses 6, .mil uses a different 6, etc.
  - `dig +noall +answer -t ns com` (or edu, us, mil, etc.) to see them
- Country codes work the same (there are 6 for .us, and 4 for the tiny island nation of Tuvalu .tv)

DNS Server Types

- The **TLD** servers then hand out **Authoritative** DNS servers for domains within their TLD
- An Authoritative DNS server tells the “world” about your organization (domain). Stanford has 6 Authoritative DNS servers
- A **Recursive** DNS server seeks out information on behalf of an end user, and often is not an Authoritative DNS server
  - 75.75.75.75 is an Xfinity Recursive DNS server, 8.8.8.8 is one Google runs for everyone, and at Stanford, Ice and Iron are the Anycast IP numbers for the M’s, which are the Recursive DNS servers for Stanford Historical Campus users (picture a Marmot eating a Magic Mango Muffin for enhanced Mojo). Umbra and Umami are the servers “behind” Ice and Iron for SRWC.
  - The M’s/U’s are also Authoritative DNS servers but *only* for Stanford users. In addition to the domains served by the A’s, they also hand out .sunet IP numbers

Any what now?

- Anycast is a load balancing system that doesn't rely on a hardware load balancer (like an F5). It can be used for many services instead of DNS
- Anycast is handled by routers. Stanford's routers "pretend" to be the Ice and Iron DNS IP numbers: 171.64.1.234 for Ice and 171.67.1.234 for Iron
- The router will figure out (or be told) the real DNS server that's easiest to route to, and send all your traffic to it. If that server isn't responsive, the router can go to another one (eventually, it's not instant). If a real DNS server needs maintenance, it can be taken out of the pool
  - We could add a 6th "M" server and nobody would have to change any endpoint configurations
  - At SRWC, we have Umbra and Umami as the real Recursive DNS servers behind "Ice" and "Iron"
- 8.8.8.8 is likely dozens of real servers, all over the world
- Cloudflare uses Anycast for many services. Their "Learning Center" web pages explain it well (and DNS too!)

Stanford's "A" Servers

- `dig +noall +answer -t ns stanford.edu` will get you the list and the TTL (86400 seconds, 24 hours)
  - That Time to Live (TTL) tells the world "you can trust that we are in charge of Stanford.EDU for a whole day before you need to check again"
- Argus and Atalante are on the main campus, in 2 different locations
- Avallone is an Anycast service at SRWC. Umbra and Umami take care of this too
- ns5.dnsmadeeasy.com, ns6. and ns7 provide us with offsite redundancy, and speed up resolution for many users
- Our "A" Authoritative DNS servers are non-Recursive, they don't serve Stanford users, they provide information to other, non-Stanford Recursive DNS servers ("the world")

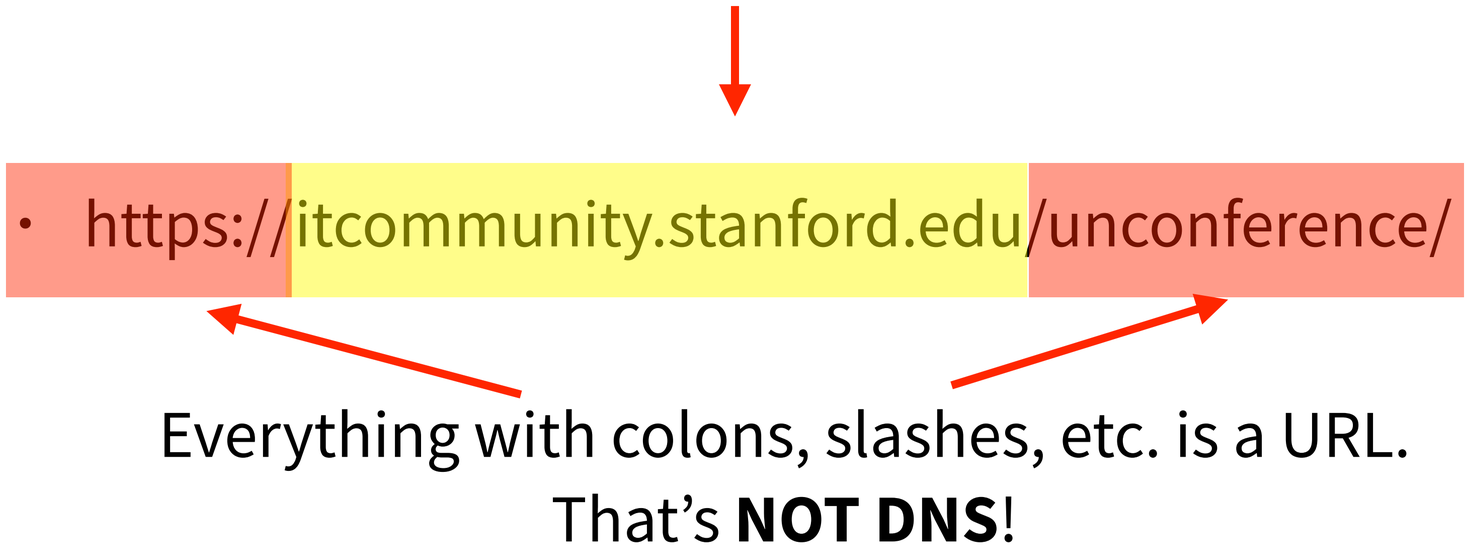
Stanford's "A" Servers

Server	IPv4	IPv6	Location	AS
Argus	171.64.7.115	2607:f6d0:0:9113::ab40:773	Main Campus	32
Atalante	171.64.7.61	2607:f6d0:0:d32::ab40:73d	Main Campus	32
Avallone	204.63.224.53	2620:6c:40c0:0:204:63:224:53	SRWC Anycast	32
ns5.dnsmadeeasy.com.	208.94.148.13	2600:1800:5::1	"Cloud" (Virginia?)	16552
ns6.dnsmadeeasy.com.	208.80.124.13	2600:1801:6::1	"Cloud" (Colorado?)	16552
ns7.dnsmadeeasy.com.	208.80.126.13	2600:1802:7::1	"Cloud" (Where? Doesn't matter, not on campus!)	16552

Back to your computer

- You've clicked on a link that put https://itcommunity.stanford.edu/unconference/ into your web browser's Address Bar
- That creates an "HTTP GET" command that's essentially "I would like to see the contents of the web page at https://itcommunity.stanford.edu/unconference/ please!"
- Your Operating System needs to craft an IP packet, there's no IP address information in that request
- Your Operating System has to ***set aside*** that HTTP GET command and resolve DNS first!

Let’s hammer that point home!



The DNS parts of your computer

- Your Operating System has an application (service) called a DNS Resolver
- Your Operating System also has a network stack, which creates IP packets with the correct TCP or UDP port, and encapsulates them into an Ethernet frame (wired or wireless, still Ethernet)
- Sitting on top of your Operating System is a web browser, it wants to get to a web page, but that request is in a virtual “back pocket” until the IP packet can be created
- The DNS Resolver will get an IP number to put into that IP packet so that the HTTP GET request can be sent to a server somewhere by IP number
- The Operating System will pull the hostname (itcommunity.stanford.edu) out of the HTTP GET request and pass *that part only* to the DNS Resolver software

Your Computer's DNS request

- Remember: You're at home, using Xfinity, 24.125.26.27, and via DHCP your OS was told to use 75.75.75.75 and 75.75.76.76. Your OS has no clue that they're Anycast, nor does it need to know that.
- Those two servers are the Xfninity DNS Recursive Resolver (XDRR) servers for YOU! *(Don't you feel special!)*
- Your DNS computer's resolver software sends a request to one of the two Xfinity DNS Recursive Resolver servers (flips a virtual coin) "Can you give me the IP address of *itcommunity.stanford.edu.* ?" (that last dot means something!)
- Since you already have the IP numbers for your DNS servers, you don't need to resolve their names, your network stack can therefore form an IP packet that you send to your router (inside an Ethernet frame), as those IP numbers are not on your subnet
- You're done, wait very patiently for DNS, then you can pull that HTTP GET request out of your pocket and send it along

Your DNS Recursive Resolver

- Your computer chose 75.75.75.75 (because it's easier to type) and passes the request to it. Your router somewhere figures out the nearest real server and passes it along
- Inexplicably, the XDRR that your router picks has completely flushed all of its cached information mere milliseconds before your request arrived
- It has to start at the very beginning, which is at the end of itcommunity.stanford.edu., that mystery dot at the end. That dot means the Root servers.
  - bind and other DNS server packages boot up with the root servers predefined, but will periodically check just to make sure they're correct
- Your XDRR sends a query to all 13 root servers, and asks them "what are the TLD servers for edu?"
  - After this first query, it will note which has the shortest round-trip time and only ask that one for subsequent queries

Your DRR, TLD to 2nd level domain

- That root domain server passes out all 13 edu TLD domain servers
- Your XDRR asks all 13 “edu” Top Level Domain servers (and notes which has the shortest round trip time for the next query) “what are the Authoritative servers for Stanford.EDU.?”
  - If any chosen “shortest round trip” server doesn’t respond, or just isn’t peppy enough your server will “demote” it and use the next best, periodically checking for which is the best and updating
- That EDU gTLD domain server hands out the 6 Stanford.EDU servers (3 “A’s” and 3 “dnsmadeeasy”)

2nd level domain to hostname, first try

- Your XDRR hasn't asked queried those servers before, so asks all 6 "What is the IP number of itcommunity.stanford.edu.?" and notes the fastest response for future queries

- The "A" servers reply "`itcommunity.stanford.edu. 1800` **`IN CNAME`** `uitcomm-web-1372853055.us-west-2.elb.amazonaws.com.`" Which can also be expressed as "`itcommunity.stanford.edu is an` **`alias`** `for uitcomm-web-1372853055.us-west-2.elb.amazonaws.com.`"

- This says "Stanford doesn't know, please ask the next organization"

gTLD and 2nd level domain, 2nd try

- Your XDRR somehow didn't yet check any ".com" hostnames in those few milliseconds, so has to ask the root DNS servers for the ".com" gTLD servers, and gets the same 13 as .edu! It should use its cached "shortest round trip" .com gTLD server
- Asks the closest "What are the name servers for amazonaws.com.?"
  - Gets 4, `dig +noall +answer -t ns amazonaws.com` to see them
- Asks all four "what is the IP address of uitcomm-web-1372853055.us-west-2.elb.amazonaws.com.?" (and notes the fastest, as before)
- The reply is "we're not that subdomain's DNS servers, use these four servers that are on the us-west-2.elb.amazonaws.com. domain instead"

A subdomain? How tedious!

- Your XDRR is fortunately, very patient.
- The four **amazonaws.com.** servers hand out the four **us-west-2.elb.amazonaws.com.** servers (`dig +noall +answer -t ns  us-west-2.elb.amazonaws.com`)
- Your XDRR then asks all four “what is the IP number of uitcomm-web-1372853055.us-west-2.elb.amazonaws.com.?” (All this time, it’s noting the shortest round drip and caching everything it learns and will hold onto that information until the TTL expires)
- They say:

```
uitcomm-web-1372853055.us-west-2.elb.amazonaws.com. 60 IN A 35.81.153.227
uitcomm-web-1372853055.us-west-2.elb.amazonaws.com. 60 IN A 54.190.219.128
```

  - If there’s more than one IP address, they’ll hand them out in a random order
- The “60” in the amazonaws response and the “1800” from Stanford are the TTL in seconds for those records. If your server gets another request within that time frame, it will just send the cached information

You finally get a response

- Your XDRR is ready to reply:
- “itcommunity.stanford.edu has two IP addresses: 35.81.153.227 and 54.190.219.128. Thank you for your request, have a nice day, come again in a few milliseconds”
- Your computer can now finally pull that HTTP GET request out of its virtual back pocket, and create an IP packet with the destination IP address of 35.81.153.227 (most operating systems pick the first one, which is why the DNS servers randomize them) and a source IP address of 24.125.26.27

The Ethernet frames, simplified

Your initial DNS request, which has nothing to do with HTTP:

Router MAC	Your MAC	24.125.26.27	75.75.75.75	IP address of itcommunity.stanford.edu?
Destination	**Source**	**Source**	**Destination**	

After communicating with many other DNS servers, you get:

Your MAC	Router MAC	75.75.75.75	24.125.26.27	itcommunity.stanford.edu has 35.81.153.227 and 54.190.219.128

You can now create a new frame, completely unrelated to DNS:

Router MAC	Your MAC	24.125.26.27	35.81.153.227	HTTP GET for https://itcommunity.stanford.edu/unconference

And you'll get a whole lot in return!

Your MAC	Router MAC	35.81.153.227	24.125.26.27	"Welcome to the Unconference!"

If dialing 411 was like DNS

- Every phone number in the world would follow the same pattern no matter the country, and be globally unique
- Every person with a phone number would have a globally unique name, like “John-Quincy-Smith-the-third.Hoboken.New-Jersey.USA”
- You’d call 411, be sent to your local information operator, they’d take your request, then put you on hold; look at a list of phone numbers of other information operators, call the USA list to get the New Jersey list of information operators, be given a list, call them, get another list, etc. until they make it to the Hoboken information operators who can look up in their phone book and give out a phone number
- Your 411 operator takes you off hold and finally gives you that phone number and then hangs up
- You still have to write down and call that number in order to have the conversation with Mr. Smith the 3rd! The 411 operator is physically incapable of doing that for you
- After a few minutes, hours, or longer, everyone involved throws away all the slips of paper where they jotted this information down, just in case it gets outdated
- All of this is nearly instant

Key Takeaways

- Your computer's DNS resolver software works with DNS recursive resolver servers
- Your web browser software works with web servers
- DNS doesn't have anything to do with URLs
- URLs don't work without DNS (unless you type in the IP number)
- An organization's Authoritative server will happily pass you onto someone else (CNAME/Alias)
- An organization's Authoritative server may pass you onto other Authoritative servers within that organization (or elsewhere) and not answer your query directly
- Your Recursive Resolver Server handles every DNS request for you

OK, so why so specific?

- We get a lot of requests for “could you send https://some-organization.stanford.edu/ to https://stanford-organization.fluffy-cloud-server.com”
- Strictly speaking, DNS can’t do that, but we *assume* what you mean is “could you create a CNAME of some-organization.stanford.edu pointing to stanford-organization.fluffy-cloud-server.com?”
  - We could, in theory, ask everyone if they really want a CNAME vs. a URL Redirect, but almost nobody would know what we mean
- However, because we will interpret the above into a CNAME request, when someone asks “could you send https://your-project.stanford.edu to https://generic-server.com/projects/your-project” to a cloud provider, we can’t! That’s **impossible**, as that’s very clearly not a DNS request. We get a LOT of those requests

CNAME vs URL Redirect

- Using a CNAME:
  - For “https://itcommunity.stanford.edu/unconference” the hostname, “itcommunity.stanford.edu” is extracted, DNS servers tell you to use 35.81.153.227 or 54.190.219.128, and you send an HTTP request to one of those IP numbers
  - Those IP numbers don’t belong to Stanford, but that’s not your problem. As long as the web servers at those IP numbers are configured to accept HTTPS requests for “https://itcommunity.stanford.edu” then you’ll get a response
  - All your web conversations are between your browser and one of the servers at one of those IP numbers

CNAME vs URL Redirect (Vanity URL)

- URL Redirect, a.k.a. Vanity URL example itunconference.stanford.edu
  - For https://itunconference.stanford.edu the hostname, itunconference.stanford.edu is extracted. Your DNS servers ask Stanford's DNS servers and are told that itunconference.stanford.edu is an alias for stanford.dns.bl.ink., they then figure out the 6 TLD servers for .ink, 4 Authoritative name servers for bl.ink, ask them, and get 54.81.116.232 as the IP number, which is then passed to your computer
  - That IP number don't belong to Stanford, but that's still not your problem. You send an HTTP GET request for "I'd like to see the web page at https://itunconference.stanford.edu" to 54.81.116.232 and the server there says: "Please go to 'https://itcommunity.stanford.edu/unconference' instead"
  - This is a **URL redirect**, and like everything "URL" or "HTTP" it's not a function of DNS. *Note the "/unconference" part!* DNS can't do that!
  - Your computer has to start a new DNS query for itcommunity.stanford.edu instead, which we've already covered
  - Your Address Bar will almost always change with a URL redirect, but that behavior is controlled by the web servers. The DNS servers have long since left the conversation

NetDB and DNS

- NetDB sends A/AAAA records to all our DNS servers, Authoritative (A's) and Recursive (The real servers behind "Ice/Iron")
  - Even if those A/AAAA records aren't Stanford IP numbers, we can still resolve Stanford.EDU (and a few other 2nd level domains), plus all our Stanford.EDU 3rd/4th level domains to any IP number you want*
- NetDB sends CNAME records to our DNS servers, NetDB may then also send A/AAAA information, or that gets passed to some other organization
- NetDB sends .sunet names to the internal, Recursive (M's and U's) DNS servers only
- NetDB is used for MX records, which can't be used with CNAME records
- NetDB will "soon" be used for TXT records too (also can't be used with CNAME records)

**Provided the IP range is in NetDB and if it isn't, just ask!*

What about those CNAME prohibitions?

- We get regular requests to set up TXT and occasionally MX records pointing to existing (or new) CNAME records
- This is prohibited in DNS, see RFC 1034 section 3.6.2
- When you ask the Stanford DNS servers “can you tell me all about www.stanford.edu.?” their reply is “no, we can’t, that’s a CNAME to pantheon-systems.map.fastly.net., please ask them”
- Even if a CNAME points to another Stanford name, think of it as a “I don’t know, ask someone else” even if “someone else” is the same DNS server!
    - Of course, we can put the MX and TXT information on the Stanford Node name record that the Stanford CNAME points to (“list.stanford.edu” is a CNAME to mailman, which has both MX and TXT records)
- Because the initial server replies “I don’t know, ask someone else” you can’t then expect that server to hand out other information, that’s not its information to hand out
    - You’re free to ask the service provider to add TXT or MX records to their entry. Some of them do!
- Because we don’t yet have TXT records in NetDB (which would reject them just like it does with MX now), they have to be made manually, so we can accidentally make a TXT record pointing to a CNAME. When that happens, bind rejects the zone file and doesn’t update!

Putting it all into NetDB

- KB00018222 explains pointing your stanford.edu node name to another organization's IP number (plain old "A/AAAA" record)
  - Problem: some providers want to use one IP number for everyone, that doesn't work well for Stanford, with our hundreds of thousands of DNS records
- KB00018688 explains pointing your stanford.edu CNAME/alias to another organizations node name
  - Problem: TXT and MX records
- KB00018689 explains delegating your domain to some another organizations DNS servers
  - Problem: It's your problem now! NetDB can't feed anyone else's DNS servers

Non-Stanford A records

- We have many AWS, Google, Azure, etc. IP ranges already defined in NetDB, but not every IP range on the planet
- Test with netdb-dev before to see if you can use the IP number you've been asked to use
  - If you can't, either the range is not defined, or your **GROUP** doesn't have access. We need to know the range and your NetDB **GROUP**, not you!
  - KB00016320 explains what rights you get based on group and record type
- Please use "Cyberspace" as the location and an appropriate Make/Model, don't just grab the first 3Com item that's listed

Stanford CNAME to non-Stanford node name

- We have many AWS, Google, Azure, Pantheon, Squarespace, Wix, etc. domains (and subdomains, and sub-sub-domains) already defined in NetDB, but not all of them
- Every subdomain has to be defined, which can get quite tedious for providers who expect 5 or more subdomains
- Again, test in netdb-dev to see if your **GROUP** has rights to create names within that provider's subdomain, and put in a ticket if your NetDB **GROUP** can't
- MX or TXT records? Either ask your provider if they're willing to host the MX or TXT records (almost none are), or you must use an A record
- Don't attach IP numbers to these records, they won't resolve, and you may have your record deleted as "stale." Be careful about "use as template"!

Delegating a Stanford.EDU domain

- You probably can’t create a NetDB domain or make a delegated domain
- You’ll need to know the name servers before you put in the ticket
- Remember, the EDU gTLD servers only hand out the standard 6 Stanford DNS servers, which then say “no, we’re not the name servers for this subdomain, ask these servers instead”
- NetDB then can no longer be used to create DNS records, but can be used for your record keeping, or to point regular stanford.edu CNAME records to your delegated domain node name records (med.stanford.edu, delegated to google, does this a lot)
  - It’s as if you’re your own “cloud” provider

In Conclusion

- DNS gets you an IP number (eventually) which allows you to route an IP packet
  - Many servers help your server get you those IP number(s)
- URLs allow your web browser to communicate with a web server (which requires an IP number, so you need DNS to help out)
- DNS can make all the CNAMEs you want, DNS can’t do a URL Redirect
  - https://itunconference.stanford.edu involves a CNAME, which gets to a web server, which uses a URL redirect to send you to another URL, which starts DNS all over again to get another CNAME, which finally gets you to another web server
- NetDB is used to make those CNAMEs, non-Stanford IP number A records, and delegated domains
- Can’t create what you want in NetDB? It’s probably your NetDB groups.

Thank You!

Questions?