DNS

The not so basic basics of DNS…

Yazid Akanho & Paul Muchene

ISOC Liberia DNS Webinar

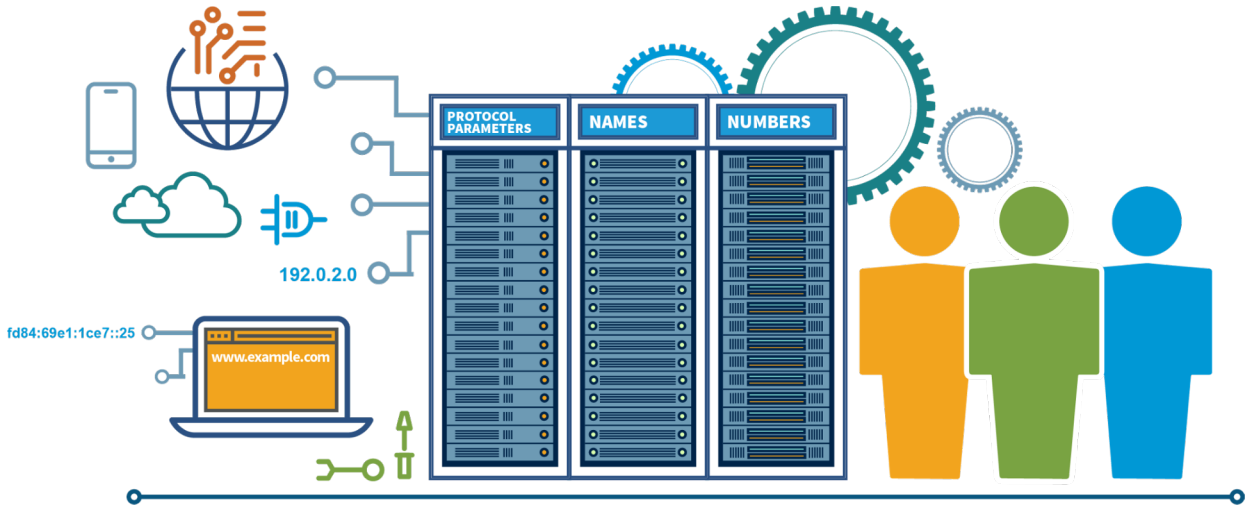
07 July 2021

Agenda

- ICANN: Who we are
- Once upon a time
- The Rise of the DNS
- DNS Database and Data
- DNS Resolution process
- Root Zone Administration
- DNS Resilience
- DNS Privacy

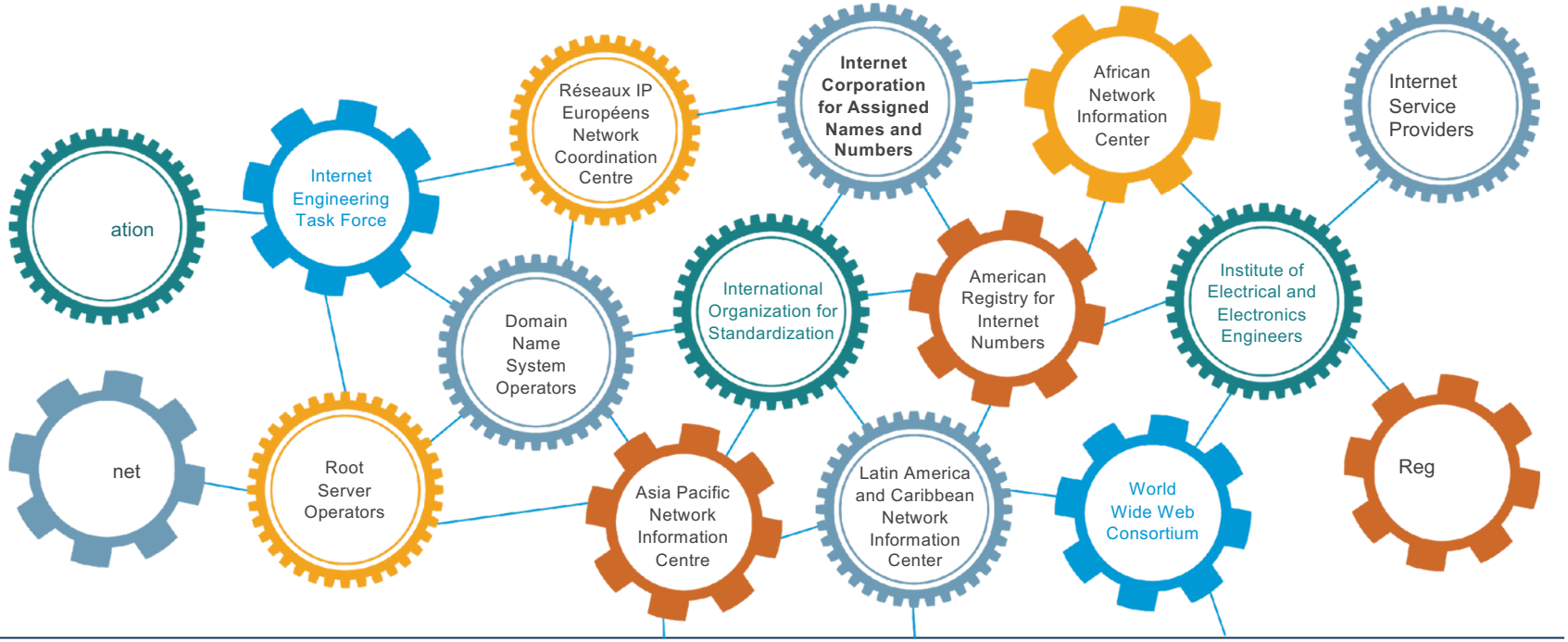
ICANN: Who we are

Coordinating with our partners, we help make the Internet work.



Our Technical Partners

Coordinating with our technical partners, we help make the Internet work.



Our Other Partners

We all work together in different ways to help make the Internet work.

ICANN's Mission

The mission of the Internet Corporation for Assigned Names and Numbers (ICANN) is to **ensure the stable and secure operation of the Internet's unique identifier systems**

Specifically, ICANN:

1. Coordinates the allocation and assignment of names in the root zone of the Domain Name System
2. Coordinates the development and implementation of policies concerning the registration of second-level domain names in generic top-level domains (gTLDs)
3. Facilitates the coordination of the operation and evolution of the DNS root name server system
4. Coordinates the allocation and assignment at the top-most level of Internet Protocol numbers and Autonomous System numbers
5. Collaborates with other bodies as appropriate to provide registries needed for the functioning of the Internet as specified by Internet protocol standards development organizations

Unique Names and Numbers

Anything connected to the Internet – including computers, mobile phones, and other devices – has a unique number called an IP address. IP stands for Internet Protocol.

This address is like a postal address. It allows messages, videos, and other packets of data to be sent from anywhere on the Internet to the device that has been uniquely identified by its IP address.

IP addresses can be difficult to remember, so instead of numbers, the Internet's Domain Name System uses letters, numbers, and hyphens to form a name that is easier to remember.

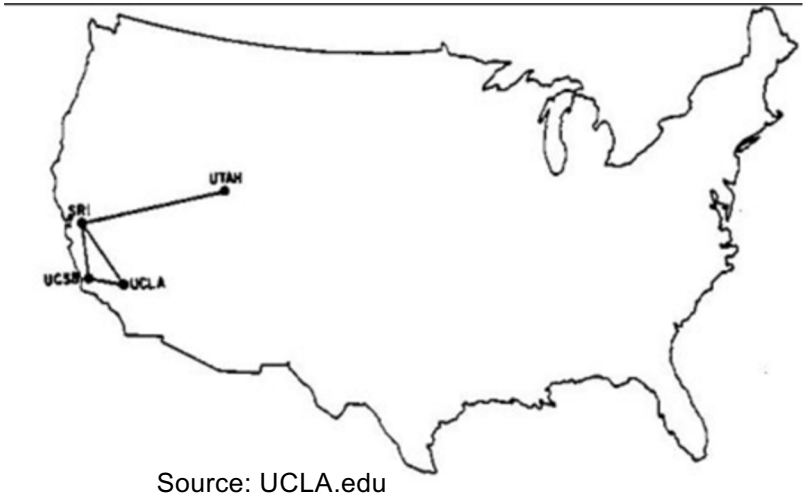
Questions and Feedback

Questions & Feedback

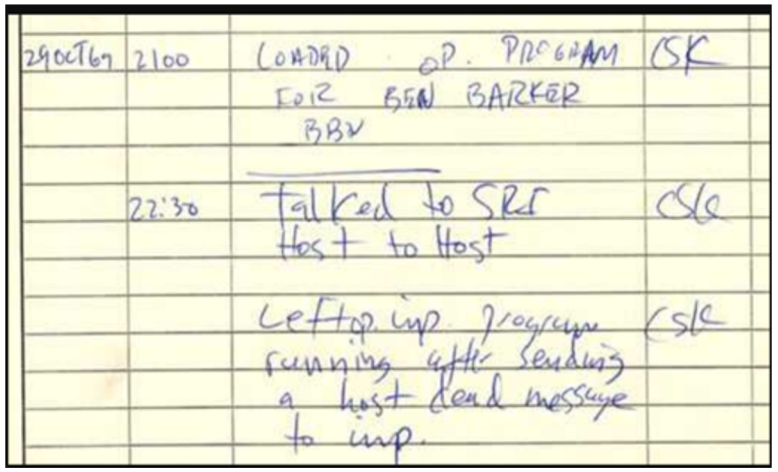
Once upon a time…

The Network of Networks

- 1969 - ARPANET is Born on October 29th – 4 Participating Institutions:
  - University of California, Los Angeles (UCLA)
  - Stanford Research Institute (SRI)
  - University of California, Santa Barbara
  - University of Utah



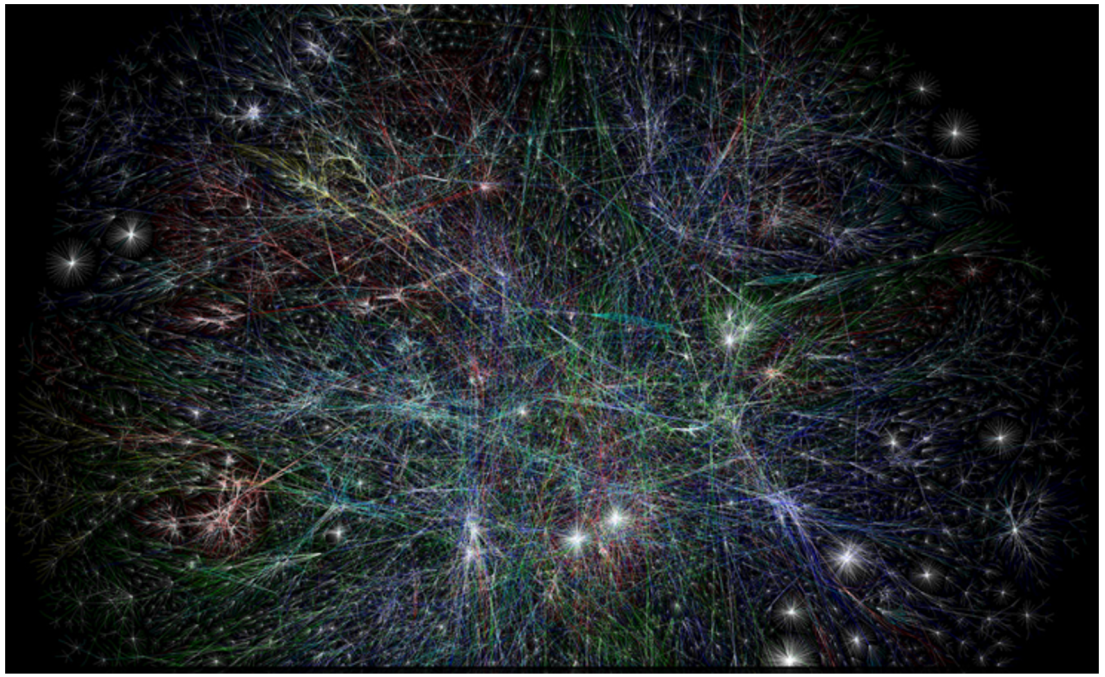
Source: UCLA.edu



Source: edn.com

The Network of Networks

Source: sri.com



Source: Kaspersky.com

The Network of Networks Today!

Names and Numbers

- Devices are identified over the Internet using IP addresses.
  - IPv4: 192.0.2.7
  - IPv6: 2001:db8::7
- While IP addresses are easy for machines to use, people prefer to use names.
- In the early days of the Internet, names were simple
  - No domain names yet
  - “Single-label names”, 24 characters maximum
  - Referred to as ***host names***

Name Resolution

- Mapping names to IP addresses (and IP addresses to names) is ***name resolution***
- Name resolution on the early Internet used a plain text ***file*** named HOSTS.TXT
  - Same function but slightly different format than the former */etc/hosts*
  - Centrally maintained by the NIC (Network Information Center) at the Stanford Research Institute (SRI)
  - Network administrators sent updates via email
- Ideally everyone had the latest version of the file
  - Released once per week
  - Downloadable via FTP

Problems with HOSTS.TXT

- Naming contention
  - Edits made by hand to a text file (no database)
  - No good method to prevent duplicates
- Synchronization
  - No one ever had the same version of the file
- Traffic and load
  - Significant bandwidth required then just to download the file

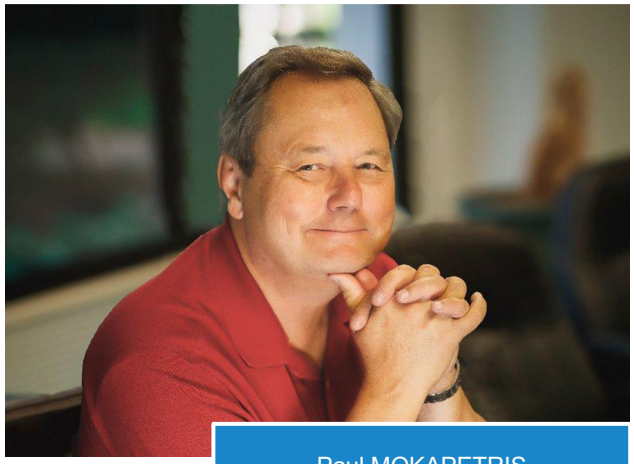
- **A centrally maintained host file just didn't scale**

The Rise of the DNS

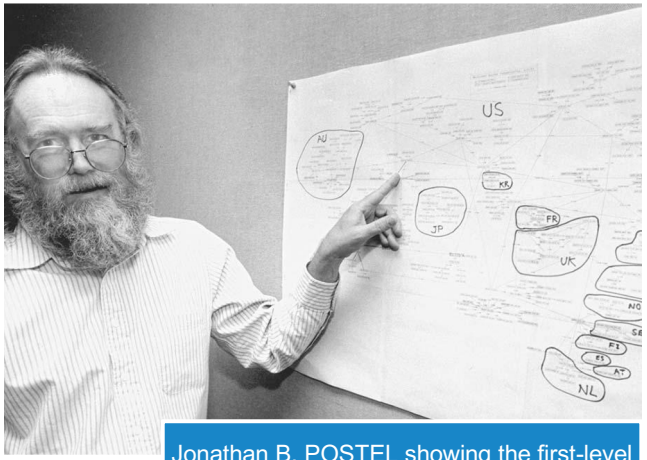
DNS to the Rescue

- Discussion started in the early 1980s on a replacement
- Goals:
  - Address HOST.TXT scaling issues
  - Simplify email routing
- Result was the ***Domain Name System***
- Requirements in multiple documents -> https://www.statdns.com/rfc/
  - RFC 799, “Internet Name Domains”
  - RFC 819, “The Domain Naming Convention for Internet User Applications”
  - RFC 882, “Domain names concepts and facilities”

Paul MOKAPETRIS & Jonathan POSTEL: inventors of DNS



Paul MOKAPETRIS

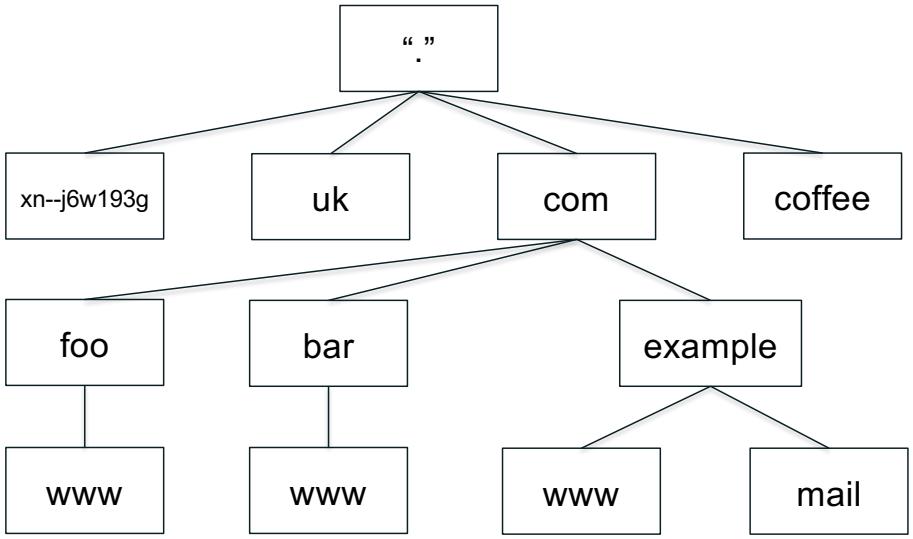


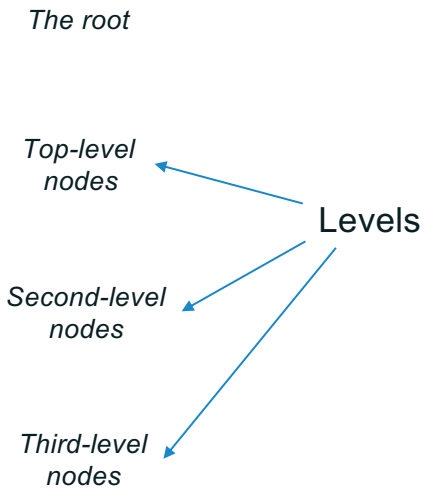
Jonathan B. POSTEL showing the first-level domains on a map in 1994

DNS Concepts

The Name Space

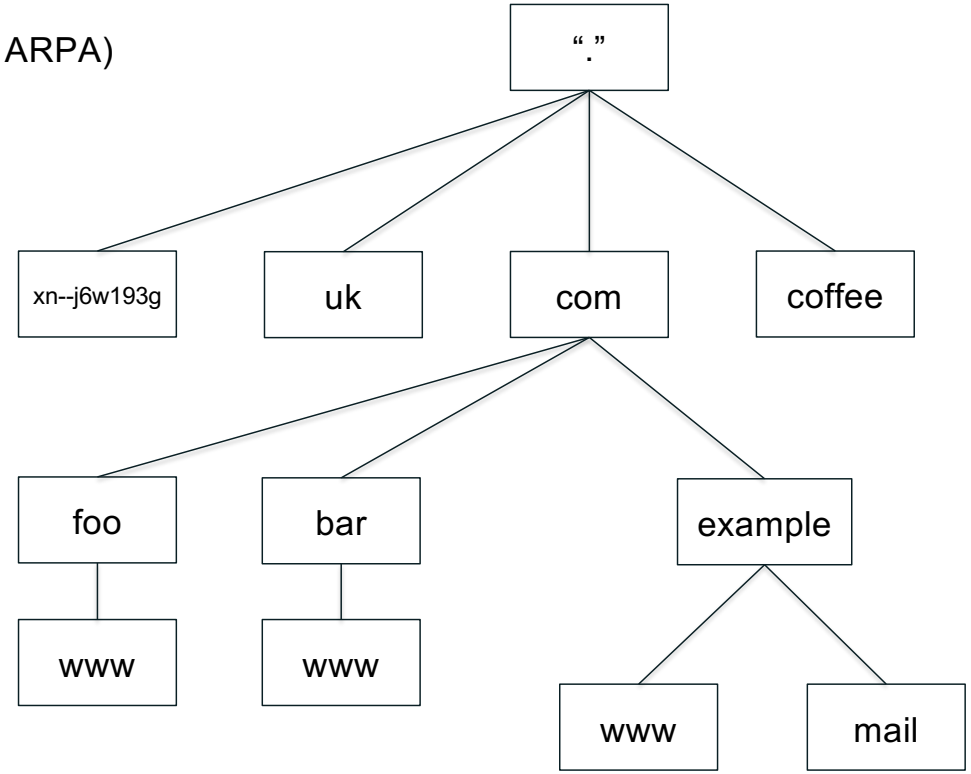
- DNS database structure is an inverted tree called the ***name space***
- Each node has a label
- The root node (and only the root node) has a null label





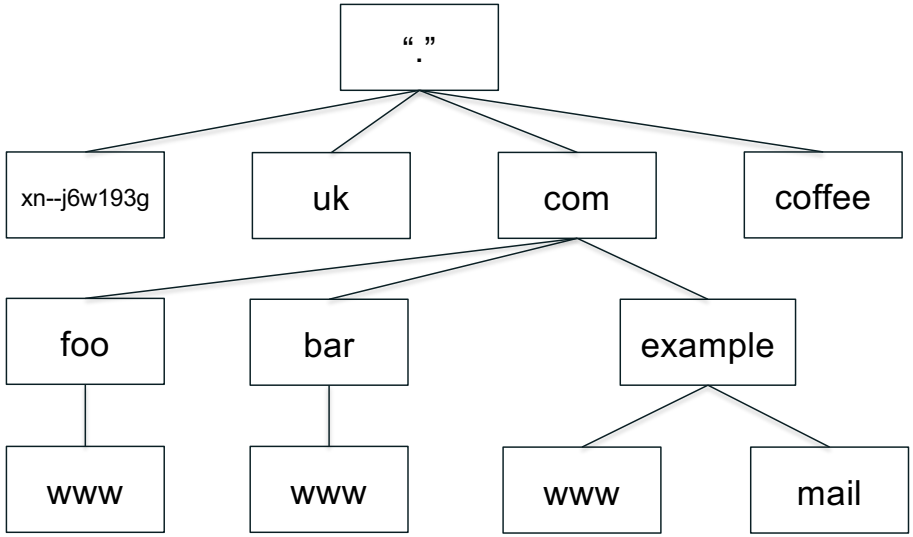
The Name Space today

- +1500 TLD (gTLD, ccTLD, IDN TLD, ARPA)
- The ”root” in its own is a “system”.



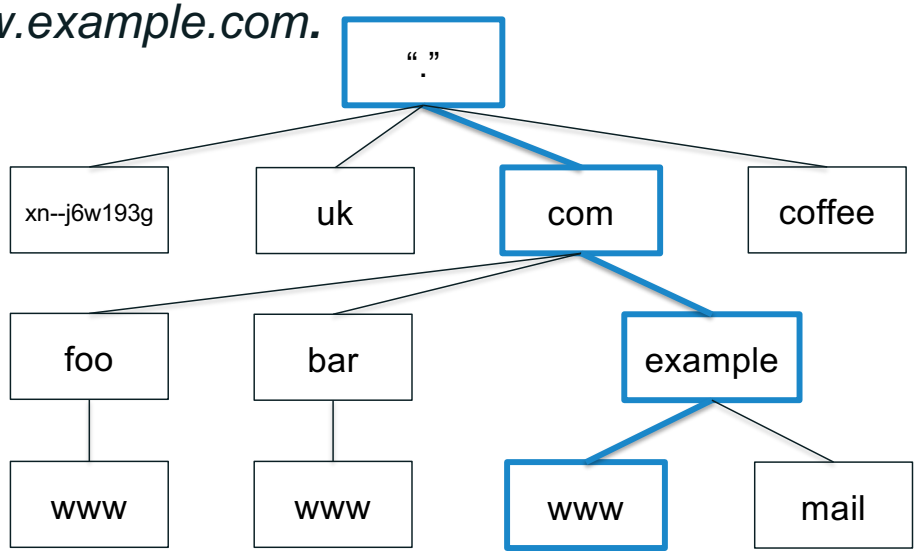
Label Syntax (before IDN)

- Legal characters for labels are “LDH” (letters, digits, hyphen)
- Maximum length 63 characters
- Comparisons of label names are not case sensitive



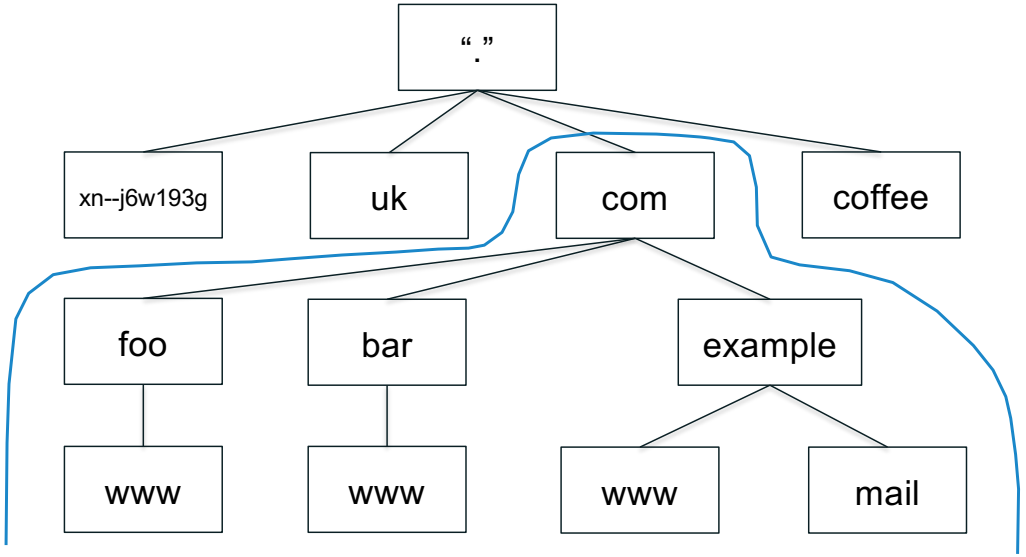
Domain Names

- RFC 8499 - An ordered list of one or more labels
- Every node has a ***domain name***
- That ***domain name*** is built by sequencing node labels from one specified node up to the root, separated by dots
- Highlighted: *www.example.com.*



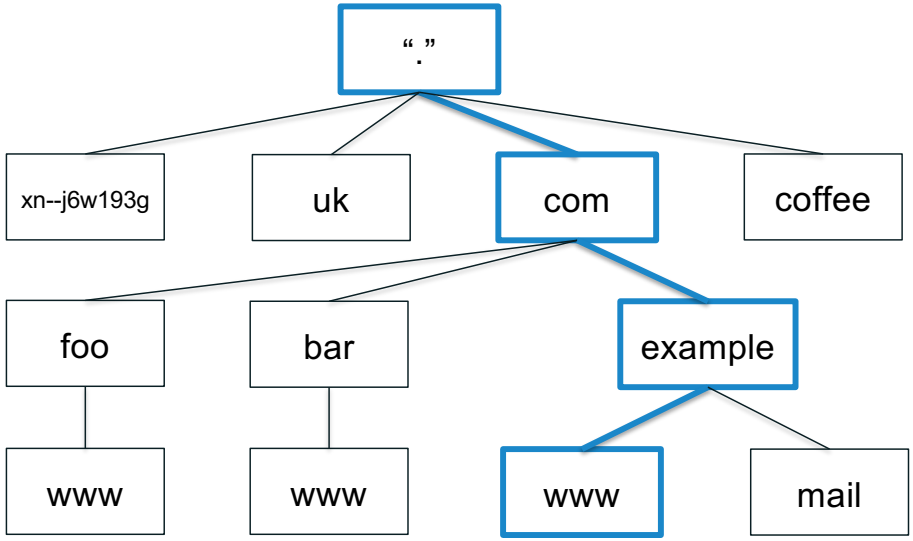
Domains

- A ***domain*** is a node and everything below it
- The top node of a domain is the ***apex*** of that domain
- Shown: the *com* domain



Fully Qualified Domain Names

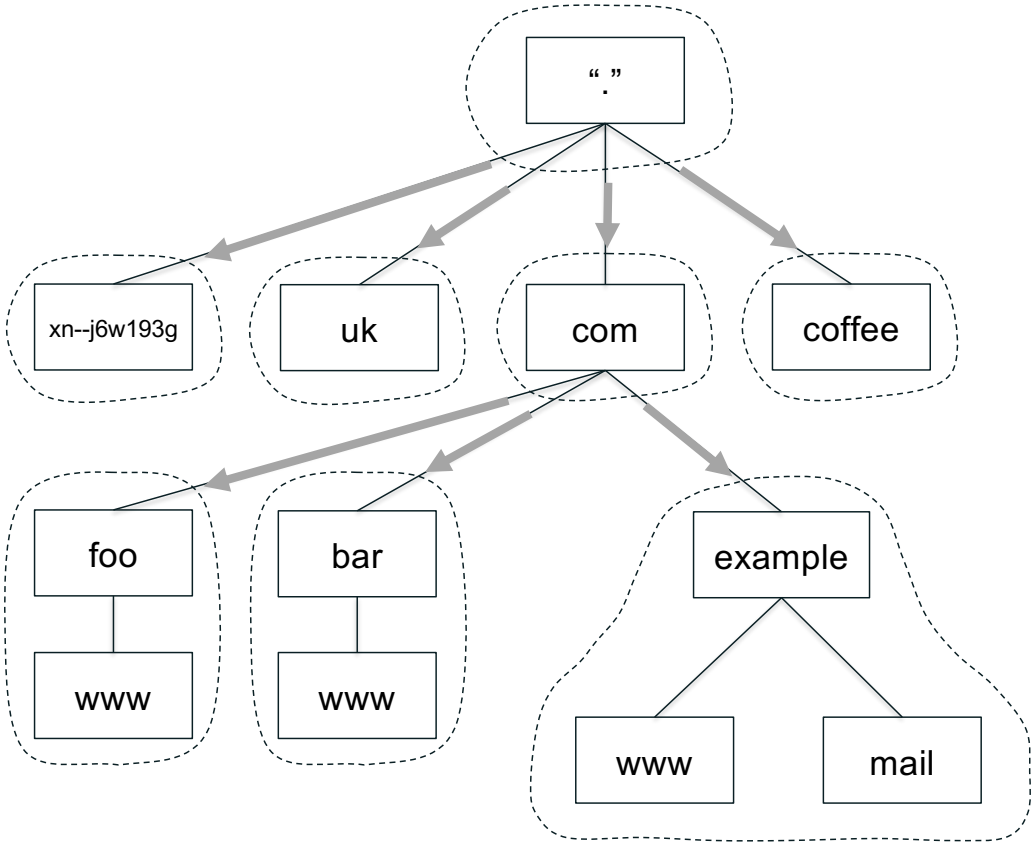
- A ***fully qualified domain name (FQDN)*** unambiguously identifies a node
  - Not relative to any other domain name
- An FQDN **ends in a dot**
- Example FQDN: *www.example.com.*



Zones

- The name space is divided up to allow distributed administration
- Administrative divisions are called ***zones***
- An administrator of any zone may delegate the administration of a subtree of its zone, thus creating a new zone
- ***Delegation*** creates zones
  - Delegating zone is the ***parent***
  - Created zone is the ***child***

Administrative Boundaries - Delegation Creates Zones



DNS Database and Data

DNS Data

- The DNS standard specifies the format of DNS data sent over the network
  Informally called “wire format”
- The standard also specifies a text-based representation for DNS data called ***master file format,*** used for storing the data (much like tables in a database)
- A ***zone file*** contains all the data for a zone in master file format

DNS Resource Records

- Recall every node has a domain name
- A domain name can have different kinds of data associated with it
- That data is stored in ***resource records*** (this are the records in DNS database)

  Sometimes abbreviated as ***RRs***
- Different record types for different kinds of data

Zone Files

- A zone consists of multiple resource records
- All the resource records for a zone are stored in a ***zone file***
- Every zone has (at least) one zone file
- Resource records from multiple zones are never mixed in the same file

Format of Resource Records

- Resource records have five fields:

  Owner: Domain name the resource record is associated with

  Time to live (TTL): Time (in seconds) the record can be cached (will see later what caching is and how it works)

  Class: A mechanism for extensibility that is largely unused

  Type: The type of data the record stores

  RDATA: The data (of the type specified) that the record carries

Master File Format

- Resource record syntax in master file format:

```
[owner]   [TTL]   [class]   <type>   <RDATA>
```

- Fields in brackets are optional
  - Shortcuts to make typing zone files easier on humans
- Type and RDATA always appear

Discussion: What type of Resource Record do you know ?

Common Resource Record Types

- **A** IPv4 address
- **AAAA** IPv6 address
- **NS** Name of an authoritative name server
- **SOA** “Start of authority”, appears at zone apex
- **CNAME** Name of an alias to another domain name
- **MX** Name of a “mail exchange server”
- **PTR** IP address encoded as a domain name (for reverse mapping)

Warehouse Analogy

- Most types are used by consumers of DNS

  A, AAAA and almost everything else

- Some types are used mostly by DNS itself
  - NS, SOA

- DNS is like a warehouse

  NS and SOA are the shelves you build…

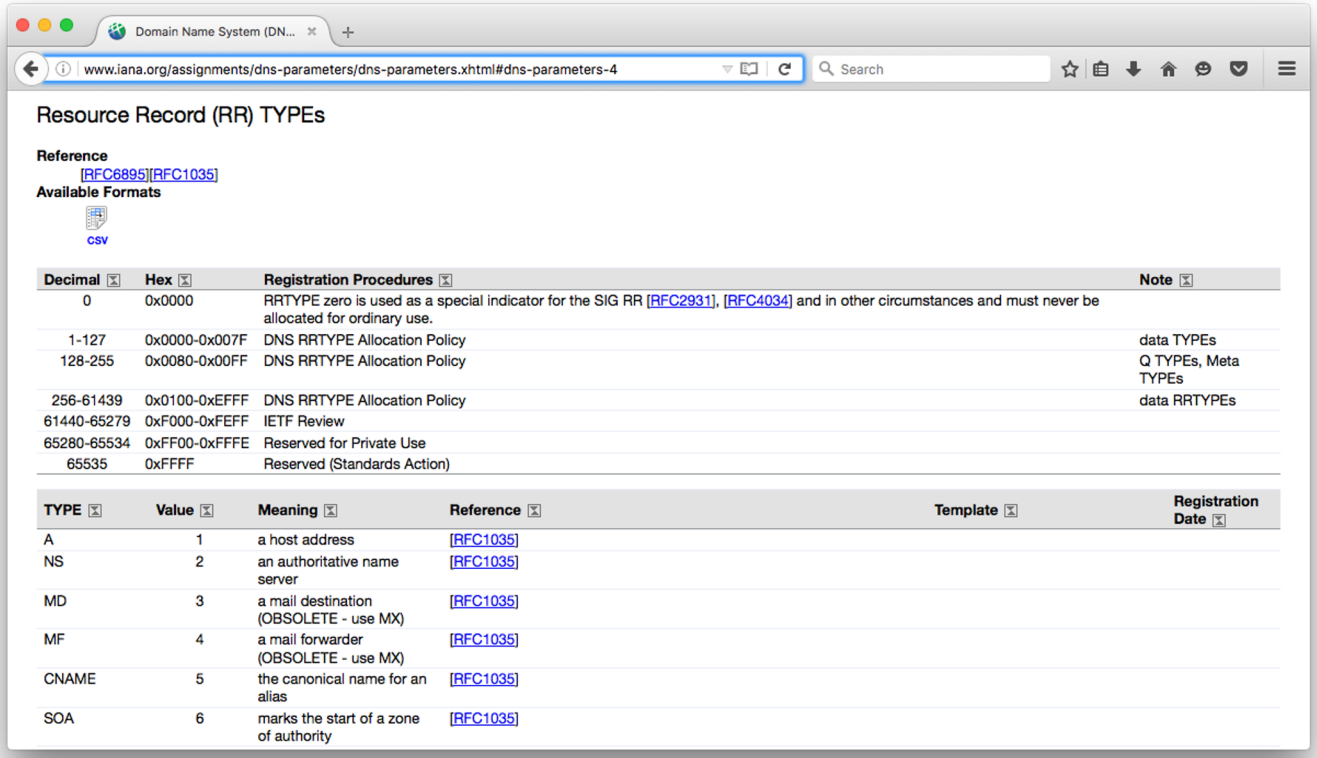
  ...so you can store stuff you care about (A, AAAA, etc.) in the warehouse

Lots of Resource Records

- There are many other resource record types
- 87+ types allocated
- IANA “DNS Resource Record (RR) TYPE Registry” under “Domain Name System (DNS) Parameters”

  http://www.iana.org/assignments/dns-parameters/dns-parameters.xhtml#dns-parameters-4

IANA DNS Resource Record (RR) TYPE Registry



Resource Record (RR) TYPEs

Reference

[RFC6895][RFC1035]

Available Formats

csv

Decimal	Hex	Registration Procedures	Note
0	0x0000	RRTYPE zero is used as a special indicator for the SIG RR [RFC2931], [RFC4034] and in other circumstances and must never be allocated for ordinary use.	
1-127	0x0000-0x007F	DNS RRTYPE Allocation Policy	data TYPEs
128-255	0x0080-0x00FF	DNS RRTYPE Allocation Policy	Q TYPEs, Meta TYPEs
256-61439	0x0100-0xEFFF	DNS RRTYPE Allocation Policy	data RRTYPEs
61440-65279	0xF000-0xFEFF	IETF Review	
65280-65534	0xFF00-0xFFFE	Reserved for Private Use	
65535	0xFFFF	Reserved (Standards Action)	

TYPE	Value	Meaning	Reference	Template	Registration Date
A	1	a host address	[RFC1035]		
NS	2	an authoritative name server	[RFC1035]		
MD	3	a mail destination (OBSOLETE - use MX)	[RFC1035]		
MF	4	a mail forwarder (OBSOLETE - use MX)	[RFC1035]		
CNAME	5	the canonical name for an alias	[RFC1035]		
SOA	6	marks the start of a zone of authority	[RFC1035]		

Address Records (A & AAAA)

- Most common use of DNS is mapping domain names to IP addresses
- Two most common types of resource records are:

  Address (A) record stores mapping for a domain name to an IPv4 address

```
example.com.            A           192.0.2.7
```

  “Quad A” (AAAA) record stores mapping for a domain name to an IPv6 address

```
example.com.            AAAA        2001:db8::7
```

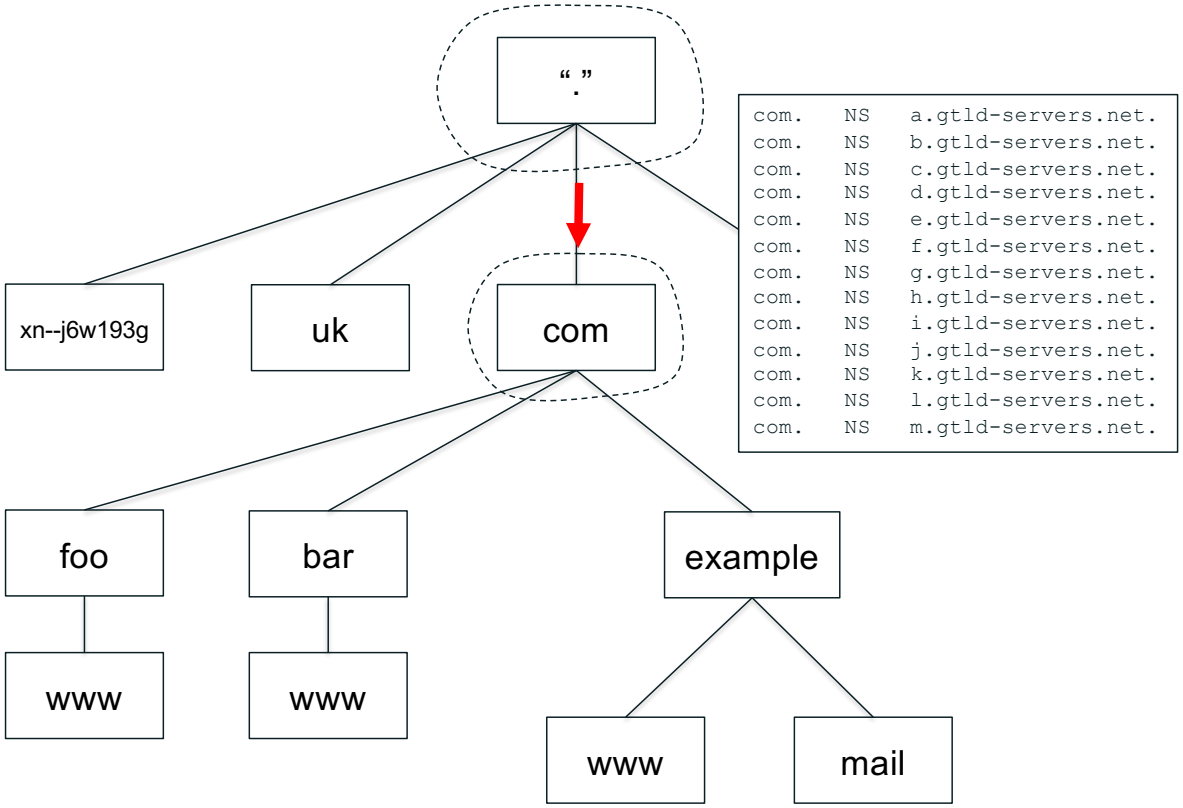
Name Server (NS)

- Specifies an authoritative name server for a zone
- The only record type to appear in two places
  - "Parent" and "child" zones

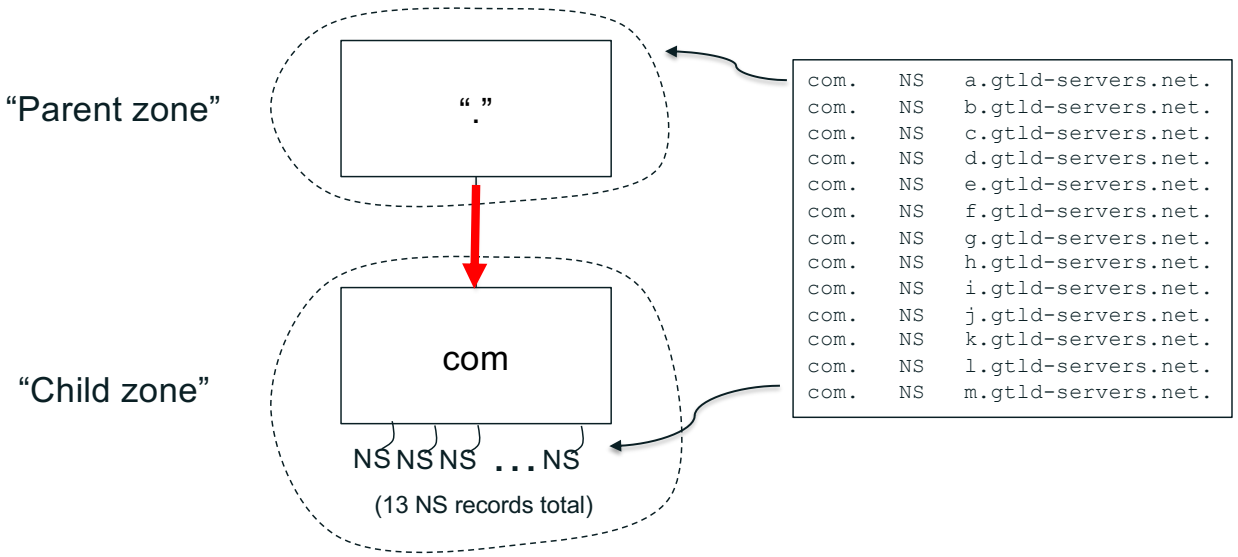
```
example.com.   NS   ns1.example.com.
example.com.   NS   ns2.example.com.
```

- Left hand side is the name of a *zone*
- Right hand side is the *name* of an authoritative name server for that *zone*
  - Not an IP address!

NS Records Mark Delegations



NS Records Appear in Two Places



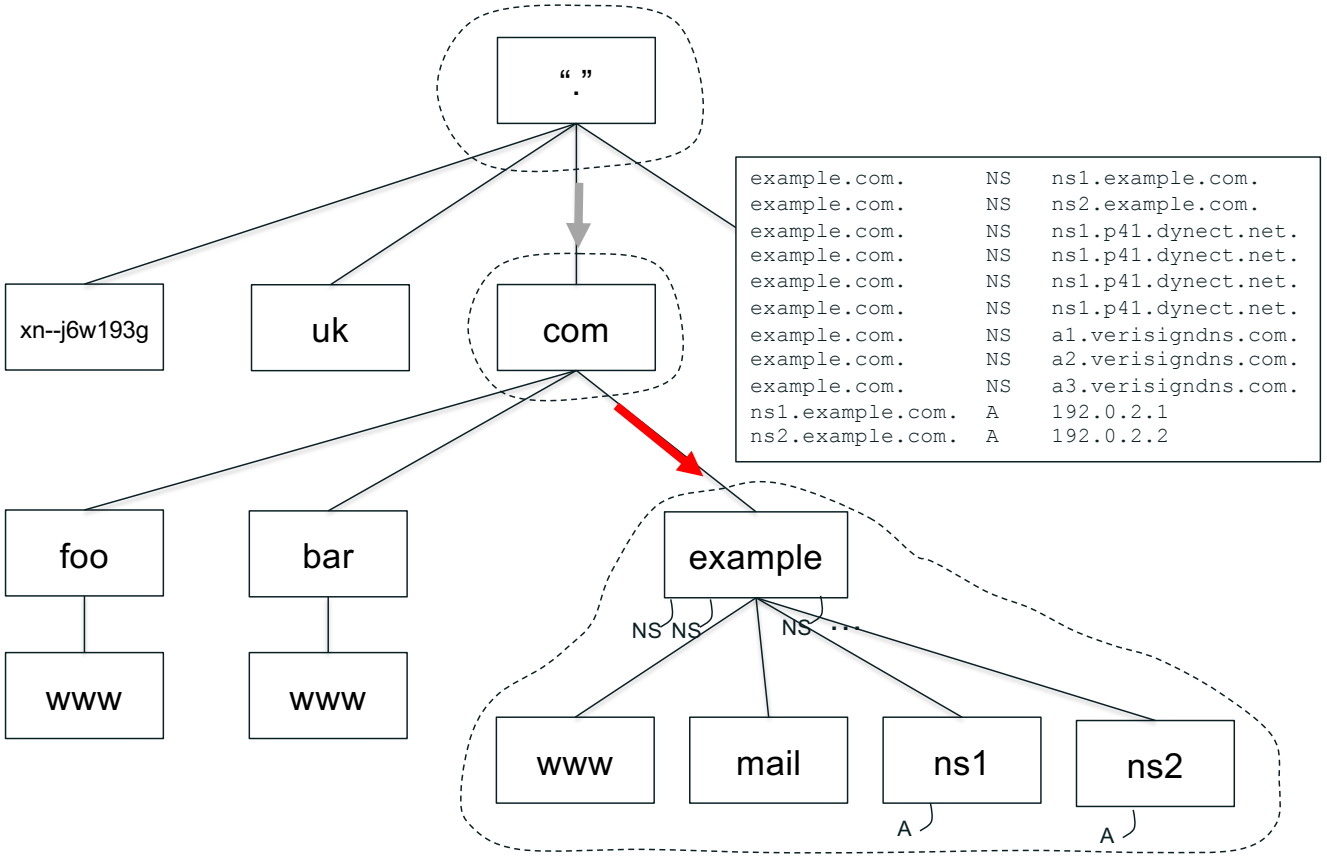
Glue Records

- A glue record is:
  - An A or AAAA record
  - Included in the parent zone as part of the delegation information
- Glue is needed to break a circular dependency
  - When the name of the name server ends in the name of the zone being delegated

```
example.com.    NS    ns1.example.com.
```

- Also for breaking for complicated dependencies not described here

More Delegation, Including Glue



Start of Authority (SOA)

- Contains administrative information about the zone.
- Every domain must have a Start of Authority record at the cutover point where the domain is delegated from its parent domain.
- SOA indicates that a name server is authoritative for a domain. If we do not receive a SOA RR in a query response from a server, that indicates the server is not authoritative for that domain.
- DNS name servers are normally set up in clusters (*master* and *secondaries*). The database for each cluster is synchronized through zone transfers. The data in a SOA record for a zone is used to control the zone transfer.

Start of Authority (SOA)

SOA records contain following fields:

mname: The primary name server for the domain, or the first name server in the name server list. For *example.com*, the primary might be *ns1.example.com*.

rname: The mailbox of the responsible party for the domain. For mailbox *john.doe@example.com* this field will be john\.doe.example.com.

serial: The version number of the original copy of a zone (preserved in zone transfers). If a secondary name server slaved to this one observes an increase in this number, the slave will assume that the zone has been updated, and it will initiate a zone transfer. Zone updates are usually denoted by the date and time stamp (e.g. if updated on 19 March 2020 at 15:55:00hs then the serial would be 20200316155500).

Start of Authority (SOA)

refresh: The number of seconds before a secondary should check for zone updates. Common practice is 24hs (a value of 86400).

retry: The number of seconds before a failed refresh should be retried, normally set to less than *refresh*. Common practice is 2hs (a value of 7200).

expire: The upper limit in seconds before a secondary name servers should stop answering requests for the zone if the master does not respond. Common practice is 1000hs (a value of 360000).

Start of Authority (SOA)

minimum: The TTL for negative caching purposes (for example, how long a resolver should consider a negative result for a subdomain to be valid before retrying).

```
example.com.        SOA   ns1.example.com. John\.doe.example.com. (
                           20200316155500 ; serial
                           86400          ; refresh (1 day)
                           7200           ; retry (2 hours)
                           3600000        ; expire (1000 hours)
                           172800 )       ; minimum (2 days)
```

CANONICAL NAME (CNAME)

- The canonical name (CNAME) is normally used to alias one name to another (but do not confuse it with an ALIAS). In the case of CNAME there should be no other records on the same name.
- As an example suppose we want to have both *example.com* and *www.example.com* pointing at the same server *example.com*, the record should be:

```
www.example.com.     CNAME      example.com.
```

- Note that a CNAME always points to a name (not an IP address).
- So somewhere else there should be a record like:

```
example.com.                A          192.0.2.7
```

Mail Exchange (MX)

- Specifies a mail server and a preference for a mail destination

```
example.com.   MX   10 mail.example.com.
example.com.   MX   20 mail-backup.example.com.
```

- Owner name corresponds to the domain name in an email address, i.e., to the right of the “@”
- The number is a preference, **lower is more desirable**
- Rightmost field is the domain name of a mail server that accepts mail for the domain in the owner name

Reverse DNS entries (PTR)

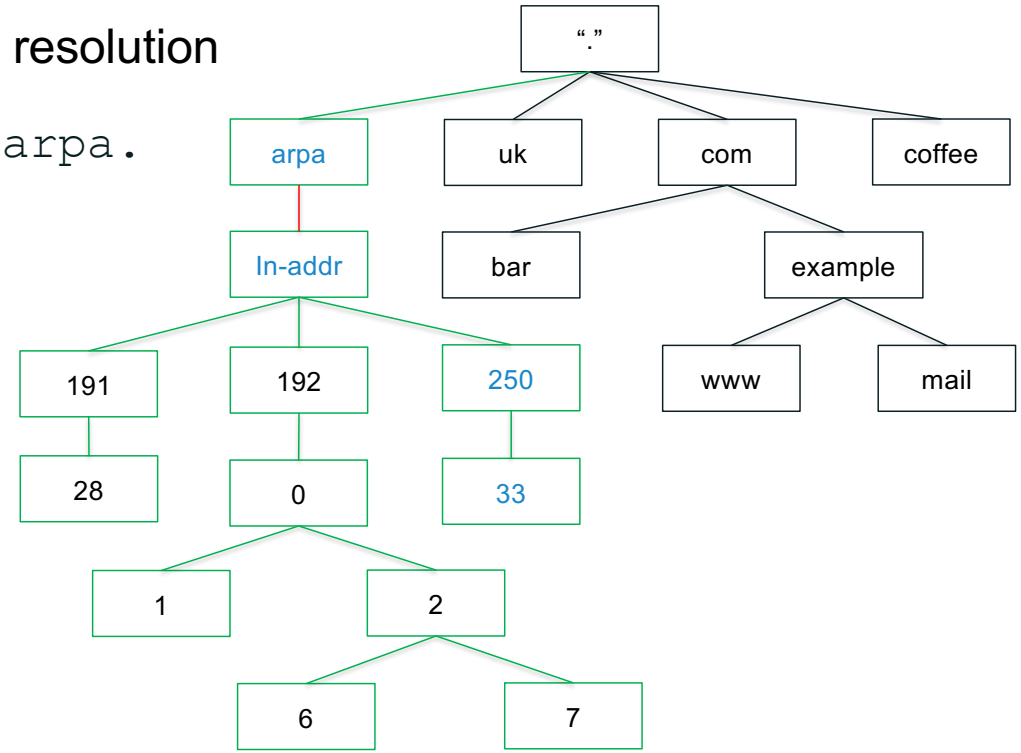
- The most common use of DNS is mapping domain names to IP addresses.
- DNS also maps IP addresses to domain names. This is called *reverse DNS* and it uses the PTR RR type.
- IPv4 reverse DNS is mapped via a special domain (subtree) called ***in-addr.arpa***.
- IPv6 reverse DNS is mapped via a special domain (subtree) called ***ip6.arpa***.
- To represent the IPv4 address *192.0.2.7* of *example.com* domain name, we reverse the IPv4 address and append the second level domain suffix ***in-addr.arpa.*** at the end, resulting in:

```
7.2.0.192.in-addr.arpa.
```

Reverse DNS entries (PTR)

- Subtree for previous reverse resolution

```
7.2.0.192.in-addr.arpa.
```



Sample Zone File: *example.com*

```
example.com.        SOA   ns1.example.com. hostmaster.example.com. (
                           20200316155500 ; serial
                           86400          ; refresh (1 hour)
                           7200           ; retry (2 hour)
                           2592000        ; expire (4 weeks 2 days)
                           172800 )       ; minimum (2 days)
example.com.        NS    ns1.example.com.
example.com.        NS    ns2.example.com.
example.com.        NS    ns1.p41.dynect.net.
example.com.        NS    ns1.p41.dynect.net.
example.com.        NS    ns1.p41.dynect.net.
example.com.        NS    ns1.p41.dynect.net.
example.com.        NS    a1.verisigndns.com.
example.com.        NS    a2.verisigndns.com.
example.com.        NS    a3.verisigndns.com.
example.com.        A     192.0.2.7
example.com.        AAAA  2001:db8::7
example.com.        MX    10 mail.example.com.
example.com.        MX    20 mail-backup.example.com.
www.example.com.    CNAME example.com.
ns1.example.com.    A     192.0.2.1
ns2.example.com.    A     192.0.2.2
```

DNS Resolution Process

DNS in a nutshell

- DNS is a distributed database
  - Data is maintained locally but available globally
- ***Resolvers*** send queries
- ***Name servers*** answer queries
- Optimizations:
  - Caching to improve performance
  - Replication to provide redundancy and load distribution

The Resolution Process

- Stub resolvers, recursive name servers and authoritative name servers cooperate to look up DNS data in the name space
- A DNS query always comprises three parameters:

  Domain name, class, type
  - E.g., *www.example.com*, IN, A
- Two kinds of queries:

  Stub resolvers send ***recursive*** queries
  - "I need the complete answer or an error."

  Recursive name servers send ***non-recursive*** or ***iterative*** queries
  - "I can do some of the lookup work myself and will accept a ***referral***."

The Resolution Process

- The resolution process is the implementation of translating from a domain name to an IP address , or more general getting the answer for a specific query.

We will go though resolution process step by step…

The Resolution Process

But first…

- How do you start the resolution process if there is no local data (resolver just booted up)? : Empty cache, and/or Not authoritative for any zones
- No choice but to start at the root zone

  The ***root name servers*** are the servers authoritative for the root zone
- How does a resolver find the NS, A, and AAAA records for the root NS?

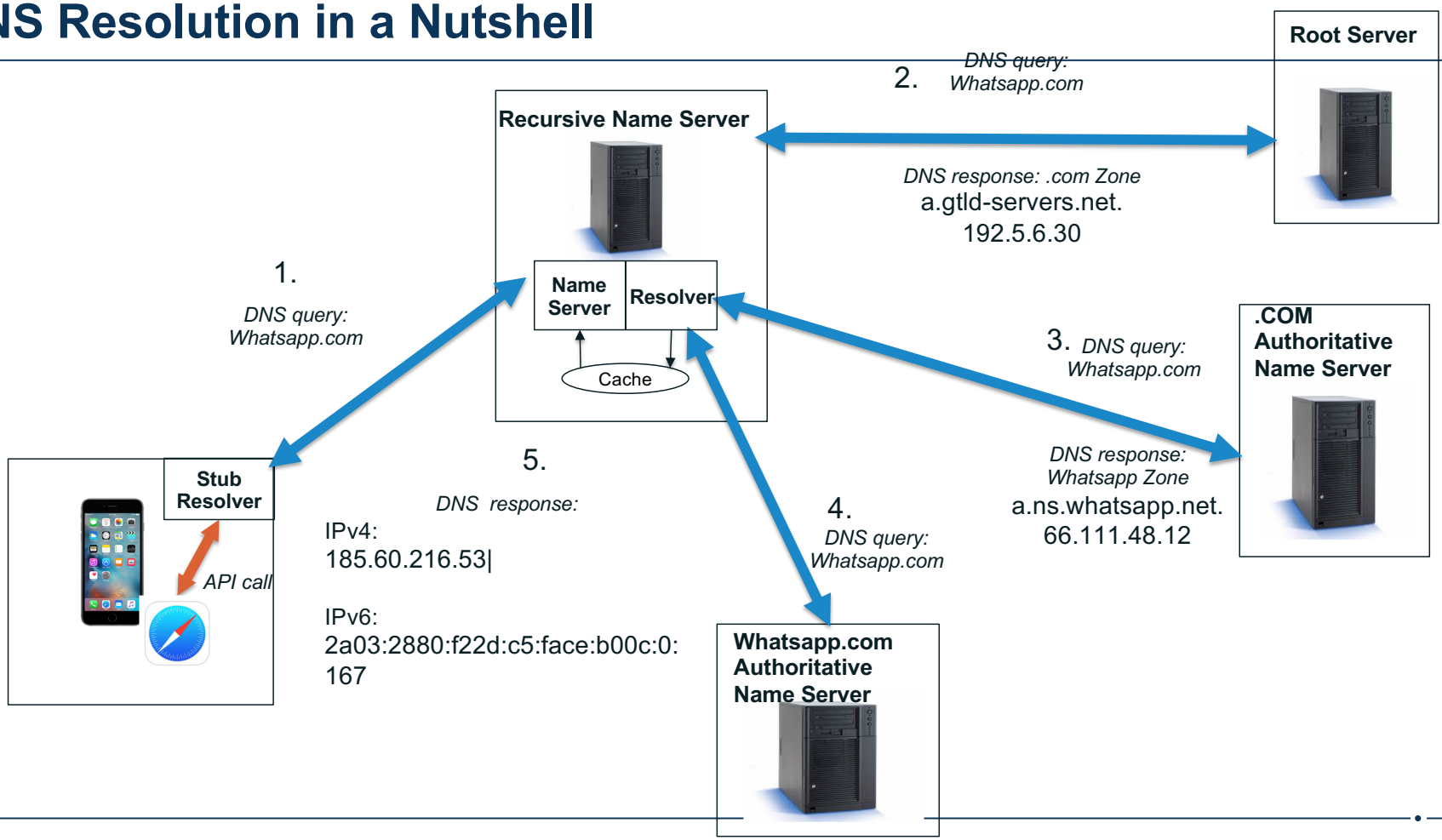
  They must be configured (in fact, most of DNS software come preloaded with an up to date version of the file called ***hint file***)

  No way to discover them
- The ***root hints file*** contains the names and IP addresses of the root name servers: https://www.iana.org/domains/root/files

List of Root Name Servers and Root Hints File

```
.                       NS      a.root-servers.net.
.                       NS      b.root-servers.net.
.                       NS      c.root-servers.net.
.                       NS      d.root-servers.net.
.                       NS      e.root-servers.net.
.                       NS      f.root-servers.net.
.                       NS      g.root-servers.net.
.                       NS      h.root-servers.net.
.                       NS      i.root-servers.net.
.                       NS      j.root-servers.net.
.                       NS      k.root-servers.net.
.                       NS      l.root-servers.net.
.                       NS      m.root-servers.net.
a.root-servers.net.     A       198.41.0.4
b.root-servers.net.     A       192.228.79.201
c.root-servers.net.     A       192.33.4.12
d.root-servers.net.     A       199.7.91.13
e.root-servers.net.     A       192.203.230.10
f.root-servers.net.     A       192.5.5.241
g.root-servers.net.     A       192.112.36.4
h.root-servers.net.     A       198.97.190.53
i.root-servers.net.     A       192.36.148.17
j.root-servers.net.     A       192.58.128.30
k.root-servers.net.     A       193.0.14.129
l.root-servers.net.     A       199.7.83.42
m.root-servers.net.     A       202.12.27.33
a.root-servers.net.     AAAA    2001:503:ba3e::2:30
b.root-servers.net.     AAAA    2001:500:84::b
c.root-servers.net.     AAAA    2001:500:2::c
d.root-servers.net.     AAAA    2001:500:2d::d
e.root-servers.net.     AAAA    2001:500:a8::e
f.root-servers.net.     AAAA    2001:500:2f::f
h.root-servers.net.     AAAA    2001:500:1::53
i.root-servers.net.     AAAA    2001:7fe::53
j.root-servers.net.     AAAA    2001:503:c27::2:30
k.root-servers.net.     AAAA    2001:7fd::1
l.root-servers.net.     AAAA    2001:500:9f::42
m.root-servers.net.     AAAA    2001:dc3::35
```

DNS Resolution in a Nutshell



Root Zone Administration

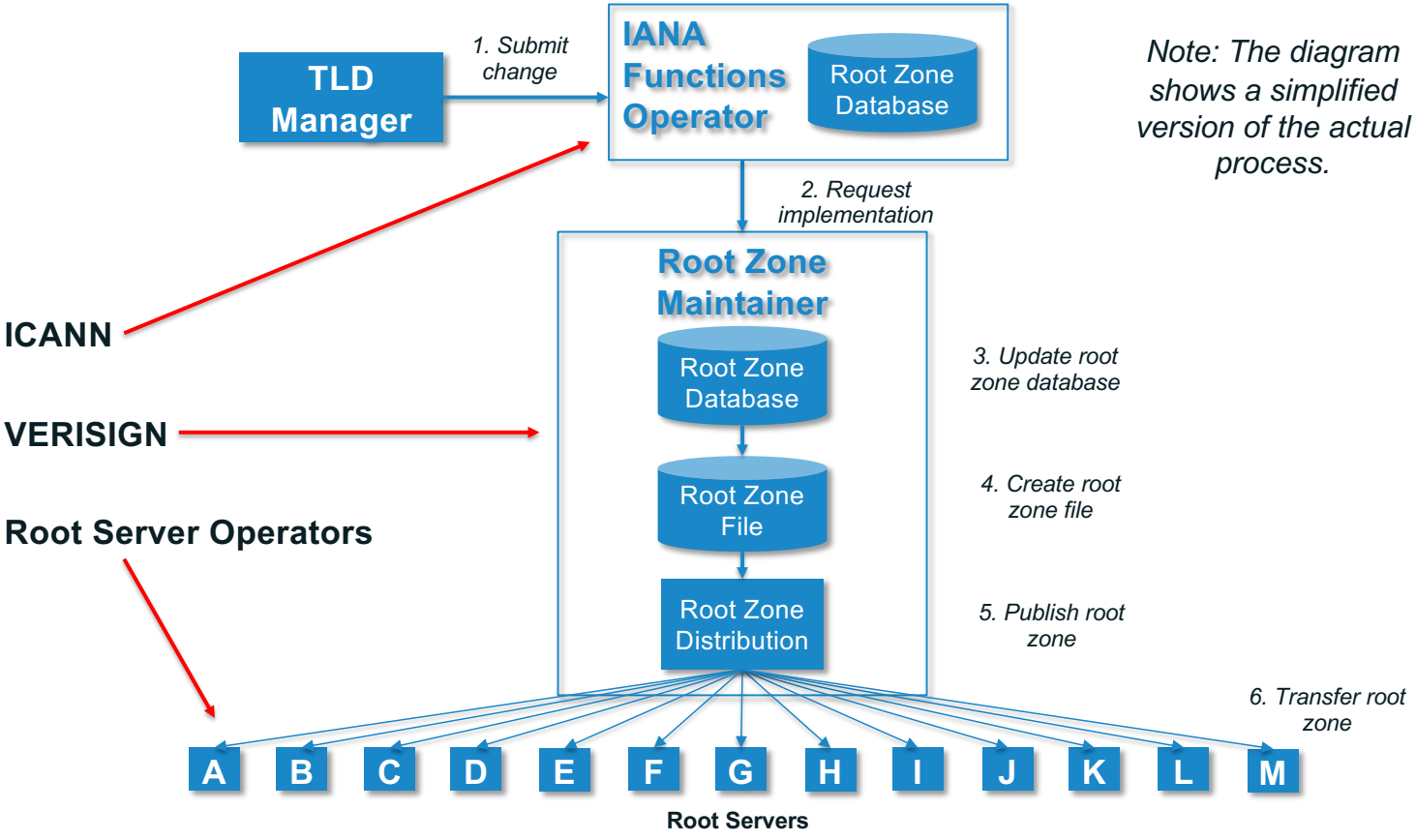
Root Zone Administration

- Administration of the root zone is far from a trivial task
- Twelve organizations operate authoritative name servers for the root zone

The Root Servers Operators

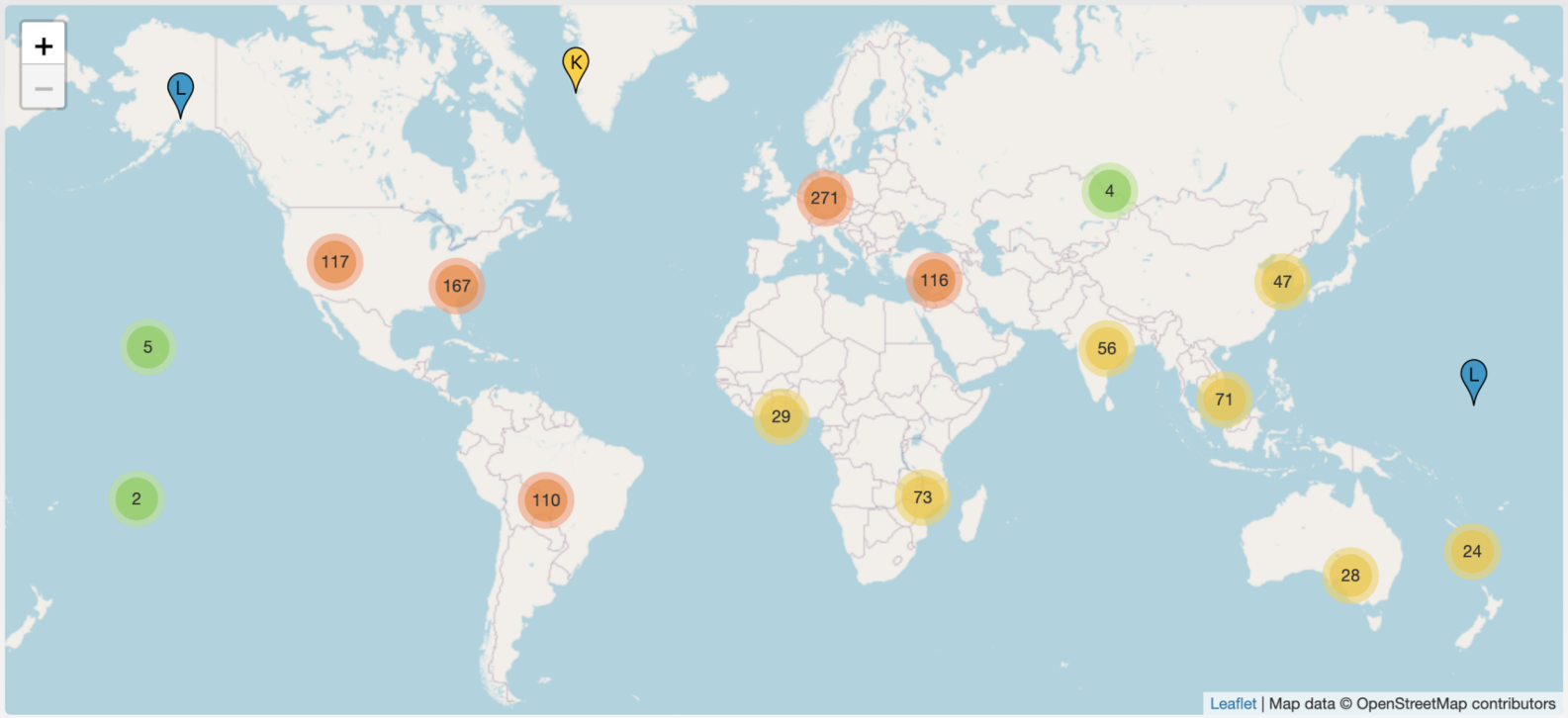
- **A** Verisign
- **B** University of Southern California Information Sciences Institute
- **C** Cogent Communications, Inc.
- **D** University of Maryland
- **E** United States National Aeronautics and Space Administration (NASA) Ames Research Center
- **F** Information Systems Consortium (ISC)
- **G** United States Department of Defense (US DoD)
- **H** United States Army (Aberdeen Proving Ground)
- **I** Netnod Internet Exchange i Sverige
- **J** Verisign
- **K** Réseaux IP Européens Network Coordination Centre (RIPE NCC)
- **L** Internet Corporation For Assigned Names and Numbers (ICANN)
- **M** WIDE Project (Widely Integrated Distributed Environment)

Root Zone Change Process



Note: The diagram shows a simplified version of the actual process.

The *root-servers.org* Web Site



As of 11/22/2020 12:29 p.m., the root server system consists of 1341 instances operated by the 12 independent root server operators.

Discussion: Which root server instance(s) are in Liberia? –> root-servers.org

DNS Resilience

DNS Resilience #1

- Zones may and should have multiple authoritative servers
  - Provides redundancy
  - Spreads the query load

Authoritative Server Synchronization

- How do you keep a zone's data in sync across multiple authoritative servers?
- Fortunately, zone replication is built into the DNS protocol
- A zone's ***primary*** name server has the definitive zone data
  - Changes to the zone are made on the primary
- A zone's ***secondary*** server retrieves the zone data from another authoritative server via a ***zone transfer***
  - The server it retrieves from is called the ***primary server***
- Zone transfer is initiated by the secondary
  - Secondary polls the primary periodically to check for changes

DNS Resilience #2 – (Anycast)

- A root server operator may deploy copies of the root server it operates anywhere in the world using a technique called ***anycast***
  - Provides **redundancy and resiliency** to global DNS infrastructure
  - Spreads the load on its root server
- Each of those copies are called ***instances*** of the root server
- All instances should have identical DNS data to ensure they all give the same answers

DNS Resilience #3 - Caching

- When a recursive resolver boots up, it has no DNS data for specific domain names (except the root name servers, which are in its configuration files).
- Each time the recursive resolver learns the answer for a query, it *caches* the data to re-use for any future identical queries.
- It only caches the answer for a limited time: the TTL of the RR.
- When the TTL expires, the resolver clears that data from its cache. Any future query results in a fresh lookup.
- Caching speeds up the resolution process and lowers potential load throughout the DNS.

DNS Privacy

Encrypting DNS Data: Benefits

- Data privacy is good for end users
- Encrypting DNS traffic protects users from observers on the path between the stub resolver and the recursive resolver
- Encryption also prevents attackers from altering responses
- Analogue: using DoT and DoH increases the security of the DNS in a similar way that HTTPS helps secure web traffic

DNS Encryption: Where?

- Until recently, stub resolvers appear only in operating systems
  - All applications call the OS for DNS service
- In the past few years, browsers (and other browser-like applications) have added their own stub resolvers
- The standards for DNS encryption assume that the client is acting as a stub resolver, and the server is acting as a recursive resolve
  - Note the “acting” part

DNS Encryption: How?

- Two Standardized Protocols:
  - **DNS-over-TLS (DoT)**
  - **DNS-over-HTTPS (DoH)**
  - There are other non-standard protocols e.g. DNSCrypt
- DoT: RFC 7858 https://datatracker.ietf.org/doc/rfc7858
- DoH: RFC 8484 https://datatracker.ietf.org/doc/rfc8484
- DoT and DoH have a large amount of overlap, but the differences are important to network operators

DNS-over-TLS (DoT)

- DoT (RFC 7858) takes advantage of Transport Layer Security (TLS) to encrypt DNS traffic between the stub resolver and the recursive resolver, giving users authentication and confidentiality for their DNS queries
- Runs on TCP/853 instead on UDP/53 (making it easy to discover and filter)
- Stub resolver establishes state with the recursive resolver

  Authentication of the resolver is optional, but is necessary to prevent on-path attacks
- DoT is useable by any application taking advantage of TLS
- Client setup requires only an IP address or domain name of a DoT resolver

DNS over HTTPS (DoH)

- Basically: the stub resolver starts an HTTPS session on port 443 (like normal web browsing) with the resolver, and when the session is established, starts sending DNS traffic that has been wrapped in HTTP queries over it.
- If the HTTP is version 2, the server can also push DNS content to the client, which the client can use or discard.
- Authentication of the resolver is mandatory because it is mandatory in HTTPS
- A bit harder to set up than DoT: need a URL
- DoH can re-use existing HTTPS connections because the service is based on the URL

DNS-over-HTTPS (DoH)

Protocol Goals (RFC 8484)

- Who do you trust?
  - “I trust my bank to give them my money.”
  - “I trust my bank enough to do online banking with them.”
  - “Maybe my bank is the most trusted vendor I should use for recursive resolver service.”
- The user decides who she trusts the most with her DNS traffic, and she configures the DoH application to use a trusted DoH resolver.
- Runs on TCP/443 and is co-mingled with *web traffic* in a single HTTPS connection, making it much harder to discover and filter

DNS-over-HTTPS (DoH)

- The stub resolver is part of the application
- The stub establishes an HTTPS session (just like normal web browsing) with the DoH resolver
- If the client is using HTTPv2, the DoH server can also push DNS content to the client, which the client can use or discard
- Authentication of the resolver is mandatory because it is mandatory in HTTPS
- Client setup requires the URL for a DoH resolver
- DoH can re-use existing HTTPS connections because the service is based on the URL

But This New Model Prompts Some Concerns

Service providers have a new paradigm to negotiate:

- No longer able to rely only on DNS to meet regulatory compliance and filtering goals
- What happens if it does not work?

  The user configures her web browser to use a DoH resolver. For whatever reason, DNS resolution stops working properly. A major concern for service providers is that the user now calls them and asks for help.
- ISPs do significant business working with parents on parental controls. When applications do their own DNS, a lot of these parental controls no longer work.
- ISPs often receive court orders to block certain sites. DoH/DoT resolvers do not know about these court orders, and still resolve these sites.

Technical and Policy Implications

- Increased privacy for users' DNS traffic
- Increased assurance for users' DNS traffic
- Circumvention of DNS filtering for security
- Circumvention of DNS filtering for local policy
- Circumvention of DNS filtering that is mandated by governments
- Unwanted centralization of DNS resolution cannot be detected
- Speed of DNS responses

Taking a Further Step Back: Policy Concerns

Stepping back from the service provider concerns, "applications doing their own DNS" (ADD) introduces all new challenges for broader public policy:

- Which laws apply?

  In the traditional model, the recursive resolver providing the answers to the end user device is generally found in the same country as the user. Applications doing their own DNS will often mean that the recursive resolver is in a different country, which means a different legal jurisdiction.

  Very important when you think about complex topics like content filtering laws and end user data privacy regulations

Taking a Further Step Back: Policy Concerns

Stepping back from the service provider concerns, ADD introduces all new challenges for broader public policy:

- Who gets to determine the resolver?

  The DoH protocol was designed to allow the end user to decide who they trust most for recursive DNS service. But nothing stops the application maker from deciding *for the user* what resolver will be used.

  What if the application maker is not honest and is purposely using a resolver which steers queries away from their intended sites and instead, provides DNS answers to sites that the application maker can profit from?

  What if the resolver operator is monetizing DNS data without the user's consent?

  Bad or dishonest implementations of ADD *disempower* end users, and do so in a context that most end users know nothing about: recursive DNS resolution

ICANN's Position

- Privacy is good
- Filtering of DNS can be beneficial
- Applications and OSs have insufficient information to make network control decisions, enforcement of legal mandates, and so on…
- DNS data should be protected
- OCTO 003: Local and Internet Policy Implications of Encrypted DNS https://www.icann.org/en/system/files/files/octo-003-30apr20-en.pdf

Current Developments

- Mozilla has partnered with Cloudflare to provide DoH and has embedded DoH into their Firefox browser
- Google through their public DNS service (8.8.8.8) supports DoH and can be activated on Chrome

One World, One Internet

Visit us at **icann.org**

@icann

facebook.com/icannorg

youtube.com/icannnews

flickr.com/icann

linkedin/company/icann

slideshare/icannpresentations

soundcloud/icann