DNS 101

Hands-on

Yazid Akanho

DNS training
March 2025

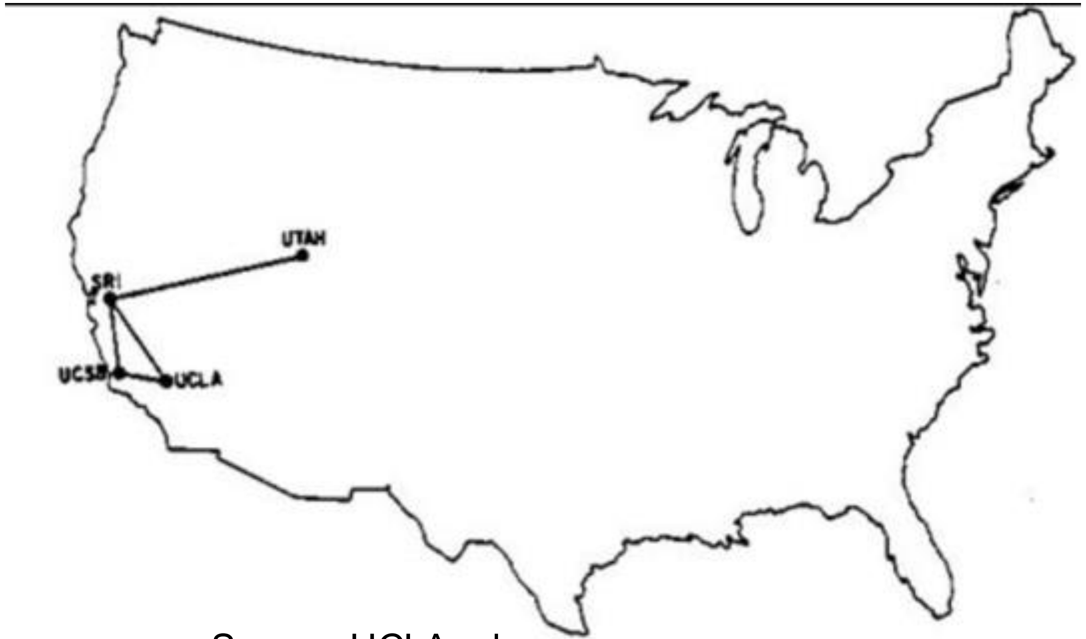
Agenda

- Once upon a time
- Rise of the DNS
- DNS Database and Data
- Resolution process
- Caching
- DNS Resilience

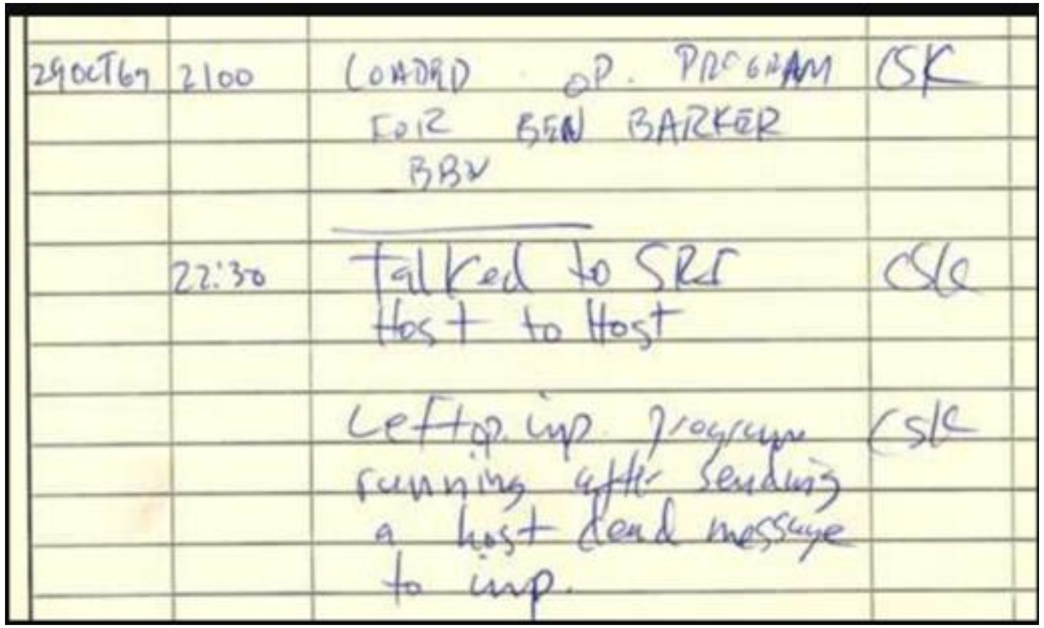
Once upon a time…

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

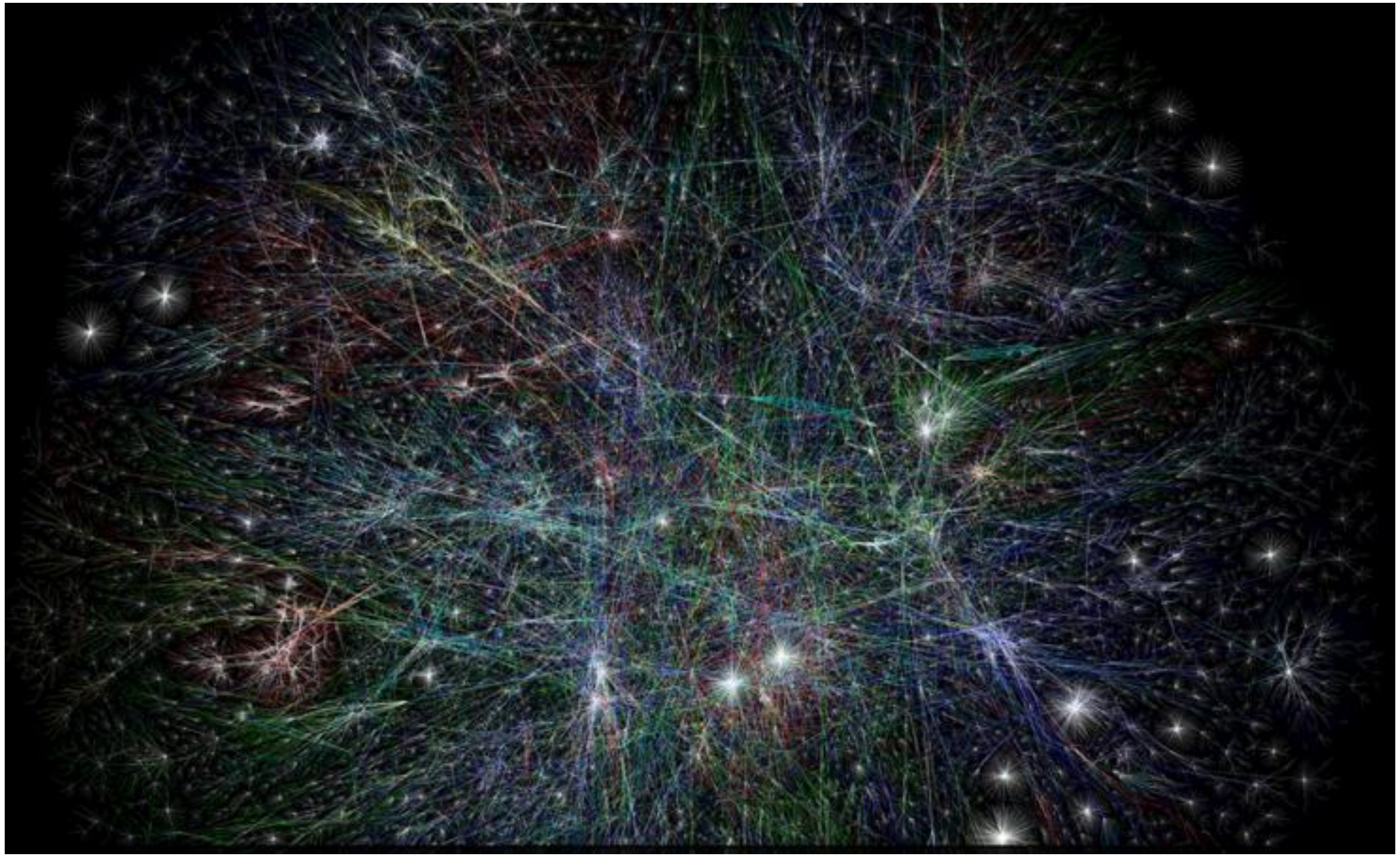


Source: UCLA.edu



Source: edn.com

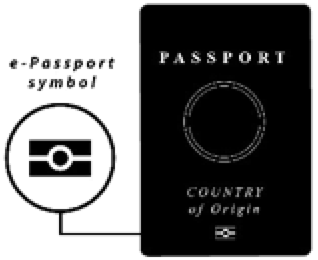
The Network of Networks



Source: sri.com

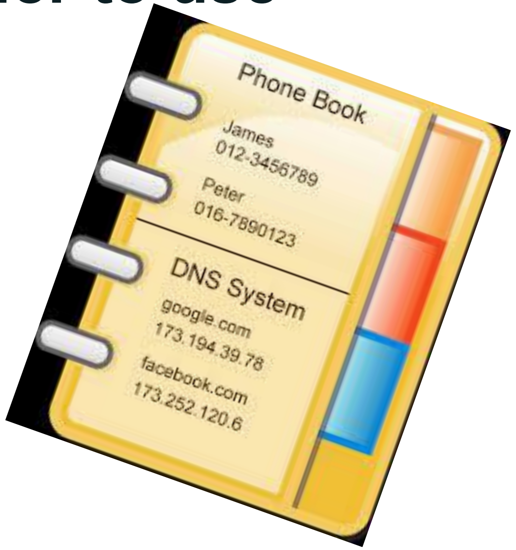
Source: Kaspersky.com

The Network of Networks: +100.000 networks; plenty of services



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

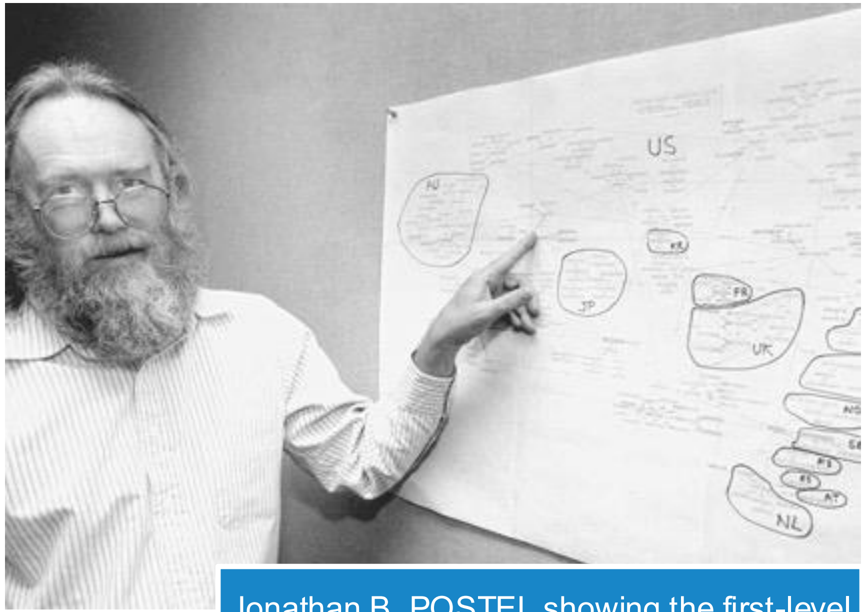
- Naming contention
  - Edits made by hand to a text file (no database)
  - No good method to prevent duplicates
- Synchronization
  - No one ever had the same version of the file
- Traffic and load
  - Significant bandwidth required then just to download the file
- **A centrally maintained host file just didn't scale**

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:
  - RFC 799, “Internet Name Domains”
  - RFC 819, “The Domain Naming Convention for Internet User Applications”
  - Most referred to: RFC 1034 and RFC 1035

Paul MOKAPETRIS & John POSTEL: inventors of DNS

Paul MOKAPETRIS



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

Rise of the DNS !

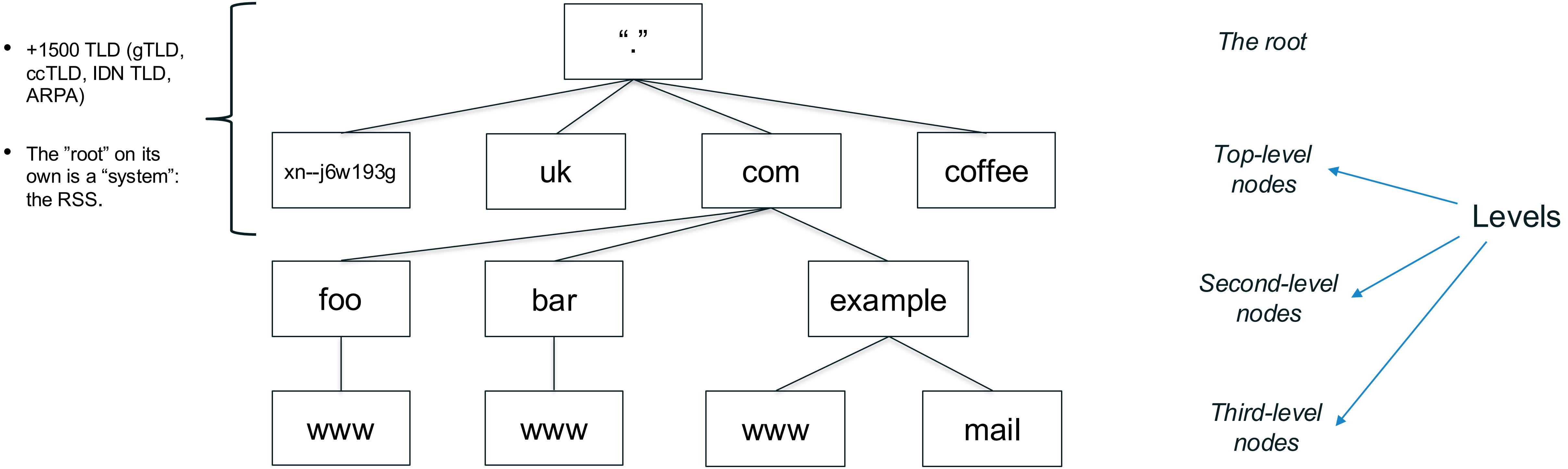
A kind of phonebook of the Internet

DNS in a nutshell

- DNS is a globally **distributed** and **coherent** database
- Components:
  - ”**name space**”: the hierarchy of the nodes and resource records.
  - ***recursive* resolvers:** run a specific type of service; receive – retrieve – respond to DNS queries. Stub resolver, caches, forwarders.
  - ***authoritative name servers*:** run a specific type of service to serve DNS zones it knows and “authoritative” answers to DNS queries. Primary and Secondaries.
- <u>A critical infrastructure</u> of the Internet, optimization and resilience are necessary: <u>caching</u> to improve performance and <u>replication</u> to provide redundancy and load distribution.

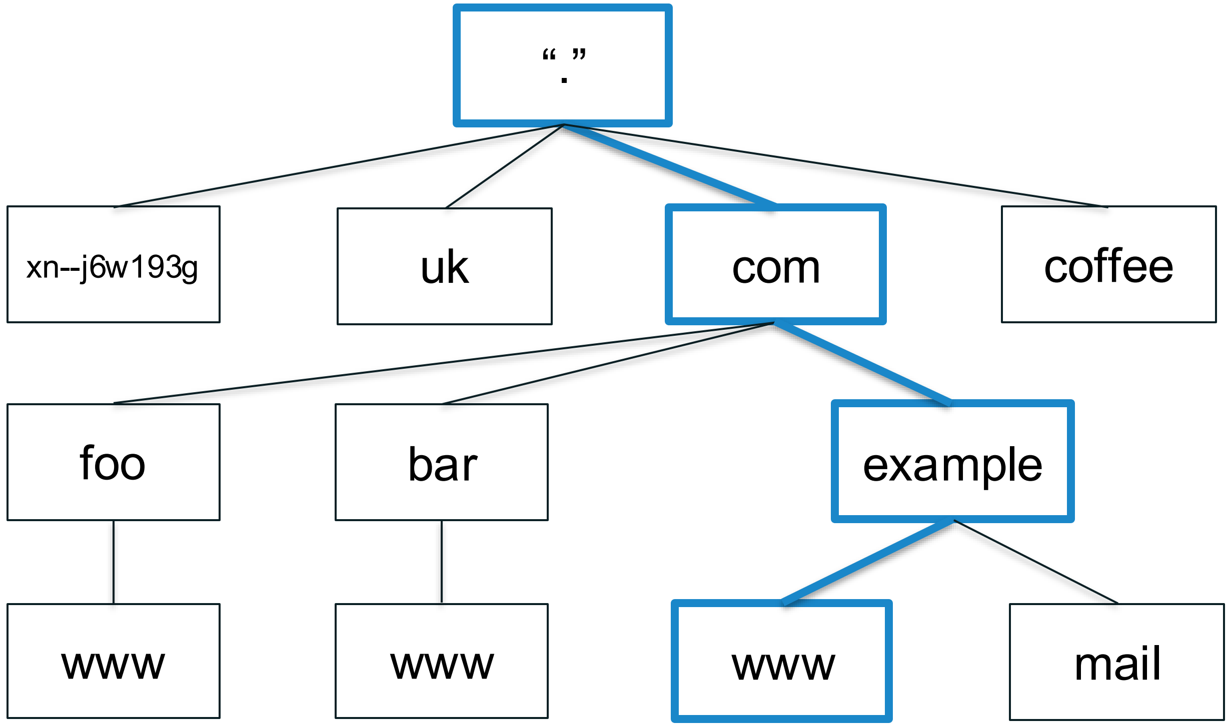
The Name Space and label syntax

- DNS database structure: an **inverted tree** called the ***name space***
- Each node has a label; root node has a null label
- **Legal characters** for labels are "LDH" (letters, digits, hyphen)
- Maximum length 63 characters



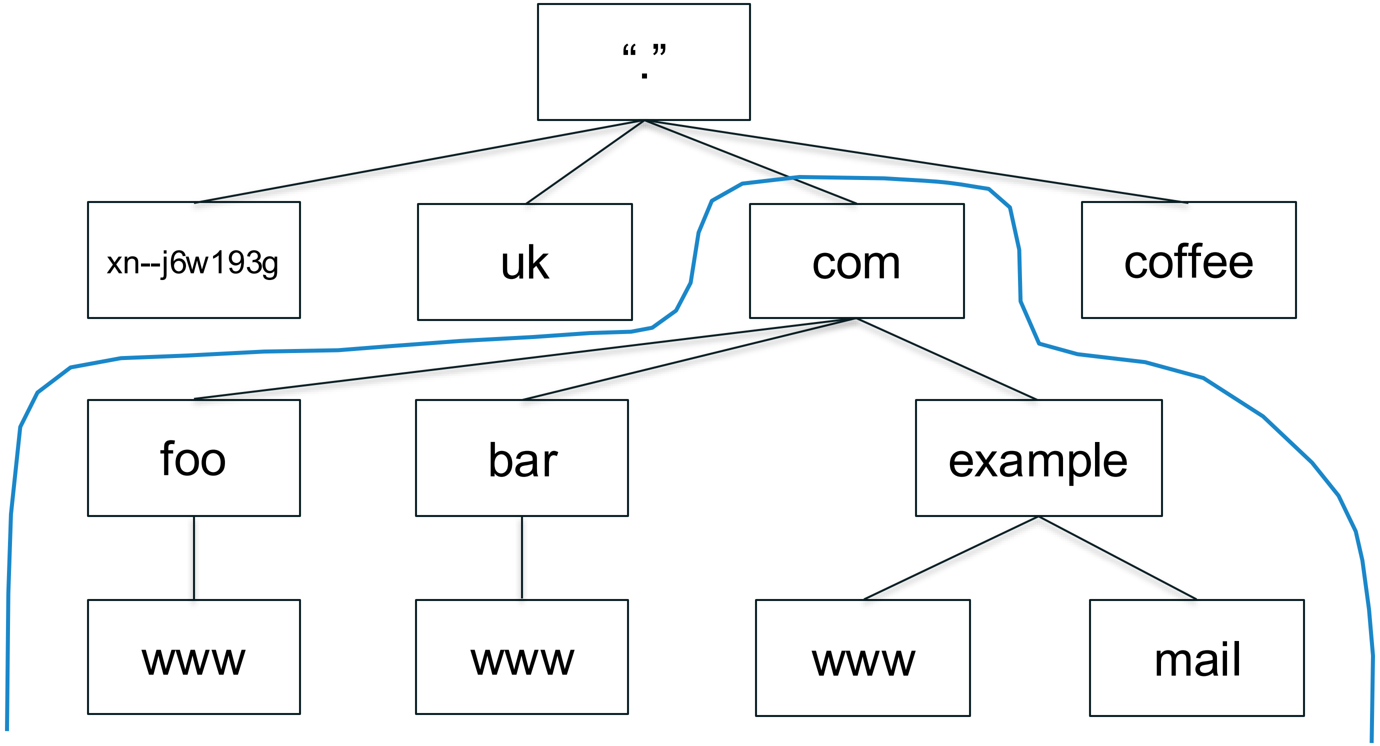
Domain Names

- 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.*



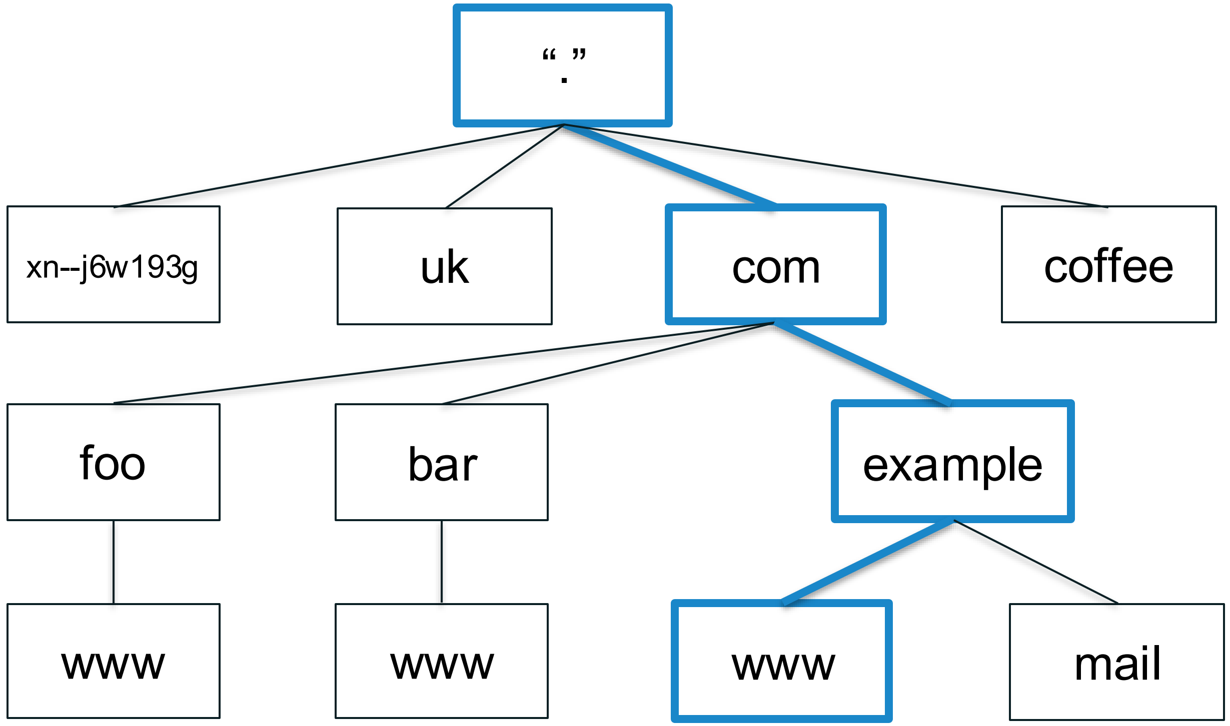
Domains

- 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

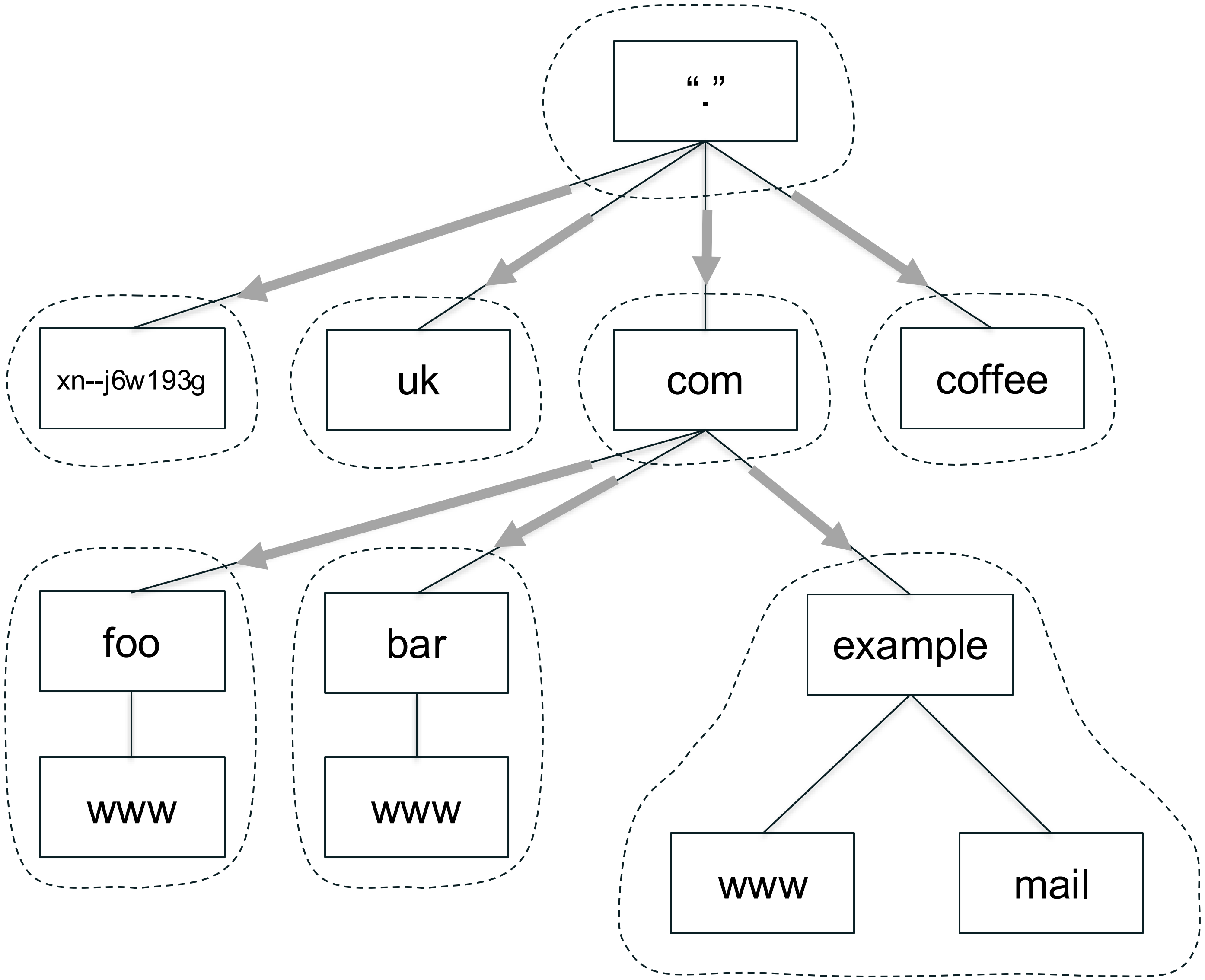
- A ***fully qualified domain name (FQDN)*** unambiguously identifies a node
  - Not relative to any other domain name
- An FQDN **ends in a dot**
- Example FQDN: *www.example.com.*



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***
  - Ex: .gh zone is delegated from root zone. Both zones share (same!) information about the authoritative nameservers in charge of the zone.

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**

Resource Record and Types

- **A and AAAA** IPv4 and 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)

- And many others at : http://www.iana.org/assignments/dns-parameters/dns-parameters.xhtml#dns-parameters-4

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
```

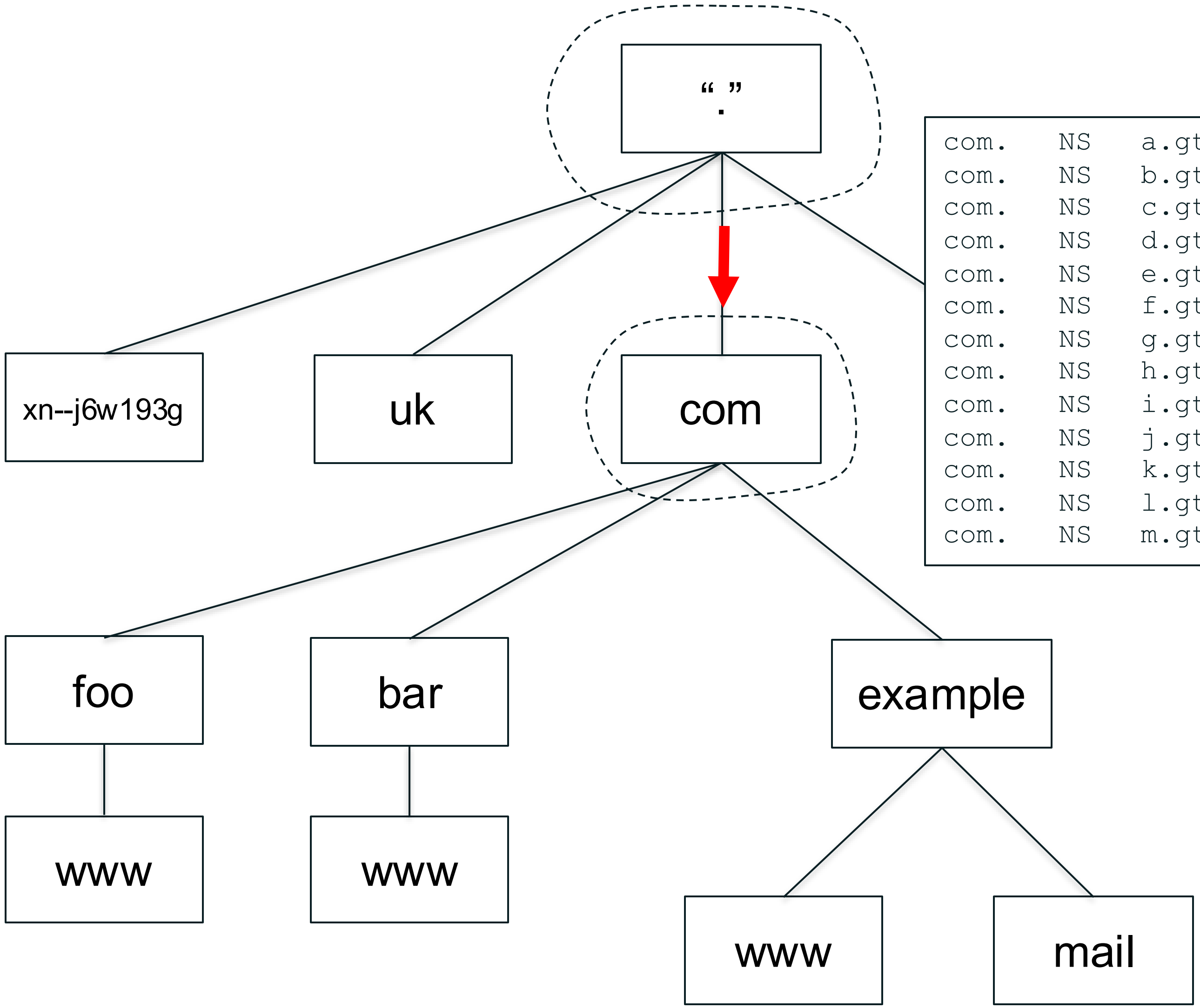
(Authoritative) Name Server (NS)

- Specifies an authoritative name server for a zone: servers that are expected to provide answers with “authority” about a domain.
- 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.
```

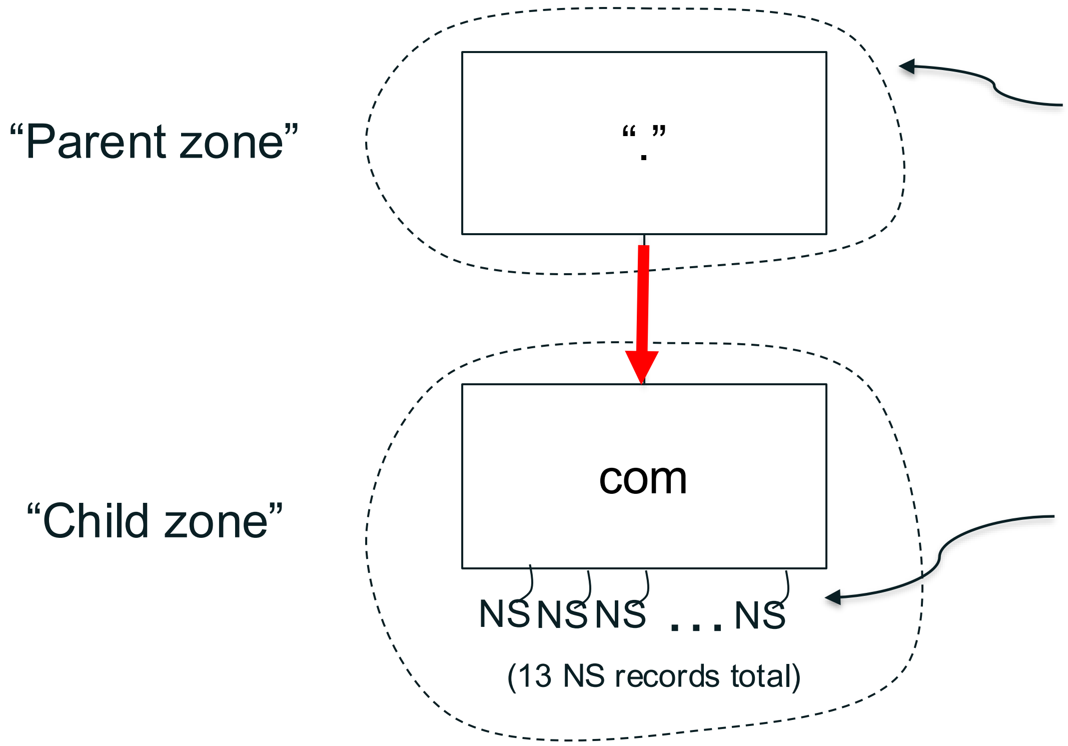
- Left hand side is the name of a *zone*
- Right hand side is the *name* of an authoritative name server for that *zone*
  - Not an IP address!

NS Records Mark Delegations




```
com.   NS   a.gtld-servers.net.
com.   NS   b.gtld-servers.net.
com.   NS   c.gtld-servers.net.
com.   NS   d.gtld-servers.net.
com.   NS   e.gtld-servers.net.
com.   NS   f.gtld-servers.net.
com.   NS   g.gtld-servers.net.
com.   NS   h.gtld-servers.net.
com.   NS   i.gtld-servers.net.
com.   NS   j.gtld-servers.net.
com.   NS   k.gtld-servers.net.
com.   NS   l.gtld-servers.net.
com.   NS   m.gtld-servers.net.
```

NS Records Appear in Two Places



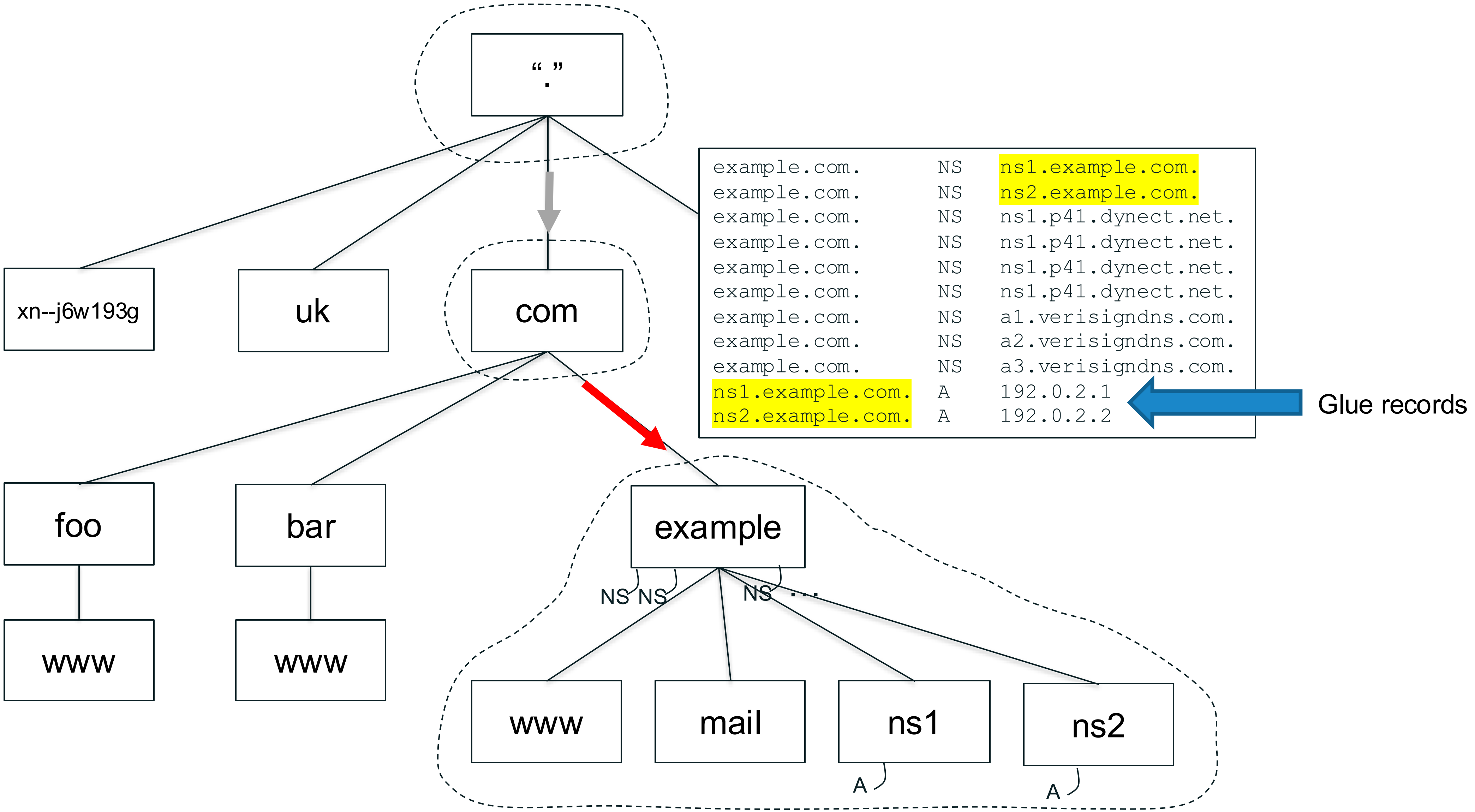
com.	NS	a.gtld-servers.net.
com.	NS	b.gtld-servers.net.
com.	NS	c.gtld-servers.net.
com.	NS	d.gtld-servers.net.
com.	NS	e.gtld-servers.net.
com.	NS	f.gtld-servers.net.
com.	NS	g.gtld-servers.net.
com.	NS	h.gtld-servers.net.
com.	NS	i.gtld-servers.net.
com.	NS	j.gtld-servers.net.
com.	NS	k.gtld-servers.net.
com.	NS	l.gtld-servers.net.
com.	NS	m.gtld-servers.net.

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.
```

More Delegation, Including Glue



Time for practice !

Getting familiar with “dig”

Lab dig

Time for practice !

1. Check NS records for your ccTLD
2. Check the IP address of those NS
3. Check the IP address records for your organization web site and mail servers.
4. Are the TTL for the above records the same ? Please use the authoritative answer (aa flag!) to answer.
5. Check if delegation information for your ccTLD in the root zone matches with the zone information.

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 (*primary* 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***: 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***: mailbox of the responsible party for the domain. For mailbox *john.doe@example.com* this field will be john\.doe.example.com.
- ***serial***: 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.
- ***refresh***: number of seconds before a secondary should check for zone updates.
- ***retry:*** number of seconds before a failed refresh should be retried, normally set to less than *refresh*.
- ***expire***: upper limit in seconds before a secondary NS should stop answering requests for the zone if the master does not respond.
- ***minimum***: TTL for negative caching purposes (for example, how long a resolver should consider a negative result for a subdomain to be valid before retrying).

Time for practice !

1. Look for the SOA record for:
   - your ccTLD
   - your organization domain.
   - Few other ccTLDs and other organization domains
2. Comment on the respective values of the various fields.

Reverse DNS entries (PTR)

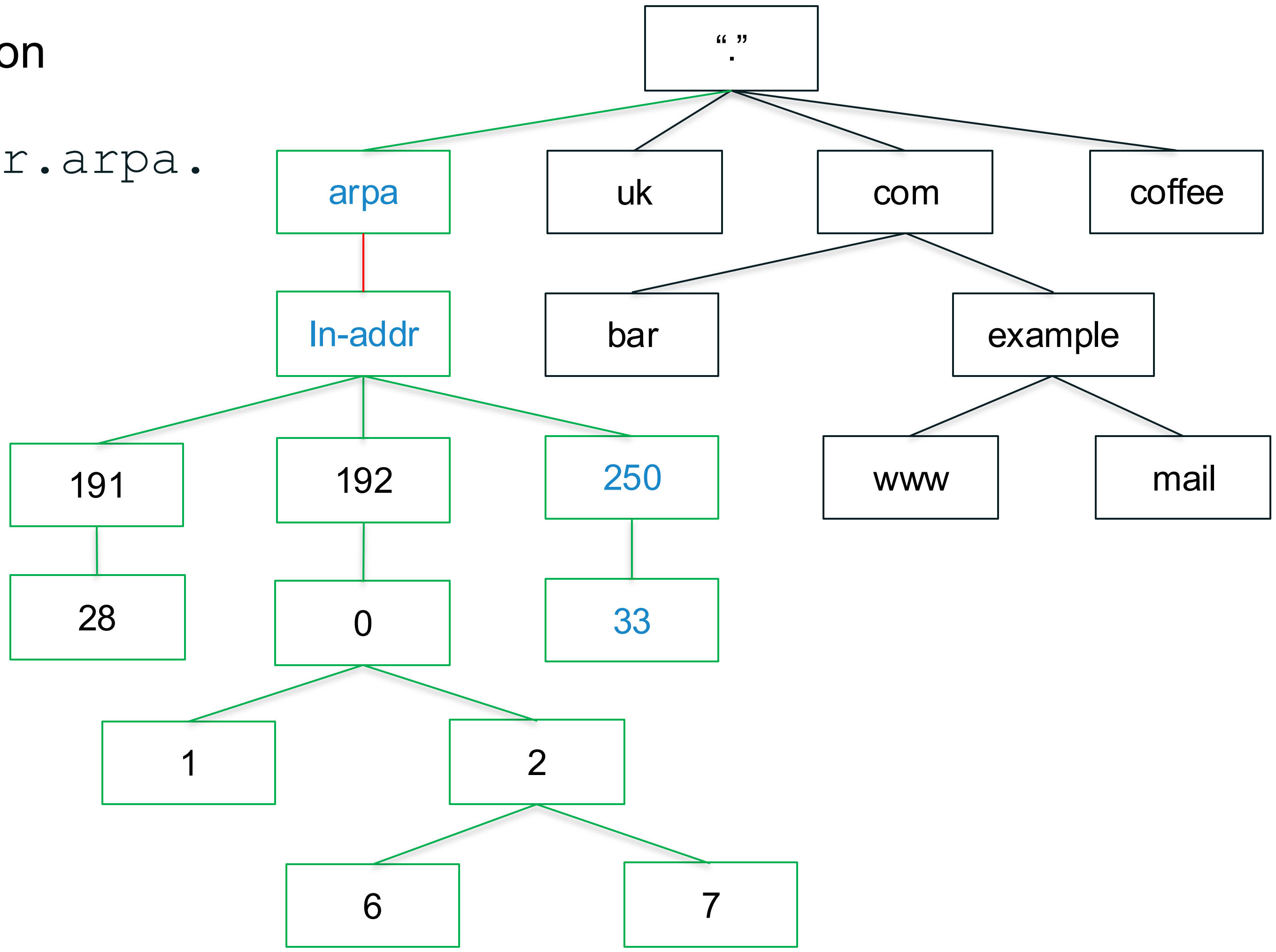
- The most common use of DNS is mapping domain names to IP addresses.
- DNS also maps IP addresses to domain names. This is called *reverse DNS* and it uses the PTR RR type.
- IPv4 reverse DNS is mapped via a special domain (subtree) called ***in-addr.arpa****.*
- IPv6 reverse DNS is mapped via a special 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.`



Time for practice !

1. Identify if reverse DNS entries exist for your organization:
   - ○ email servers
   - ○ Web server
   - ○ Authoritative nameservers.
2. If reverse DNS entry does not exist, how to create ?

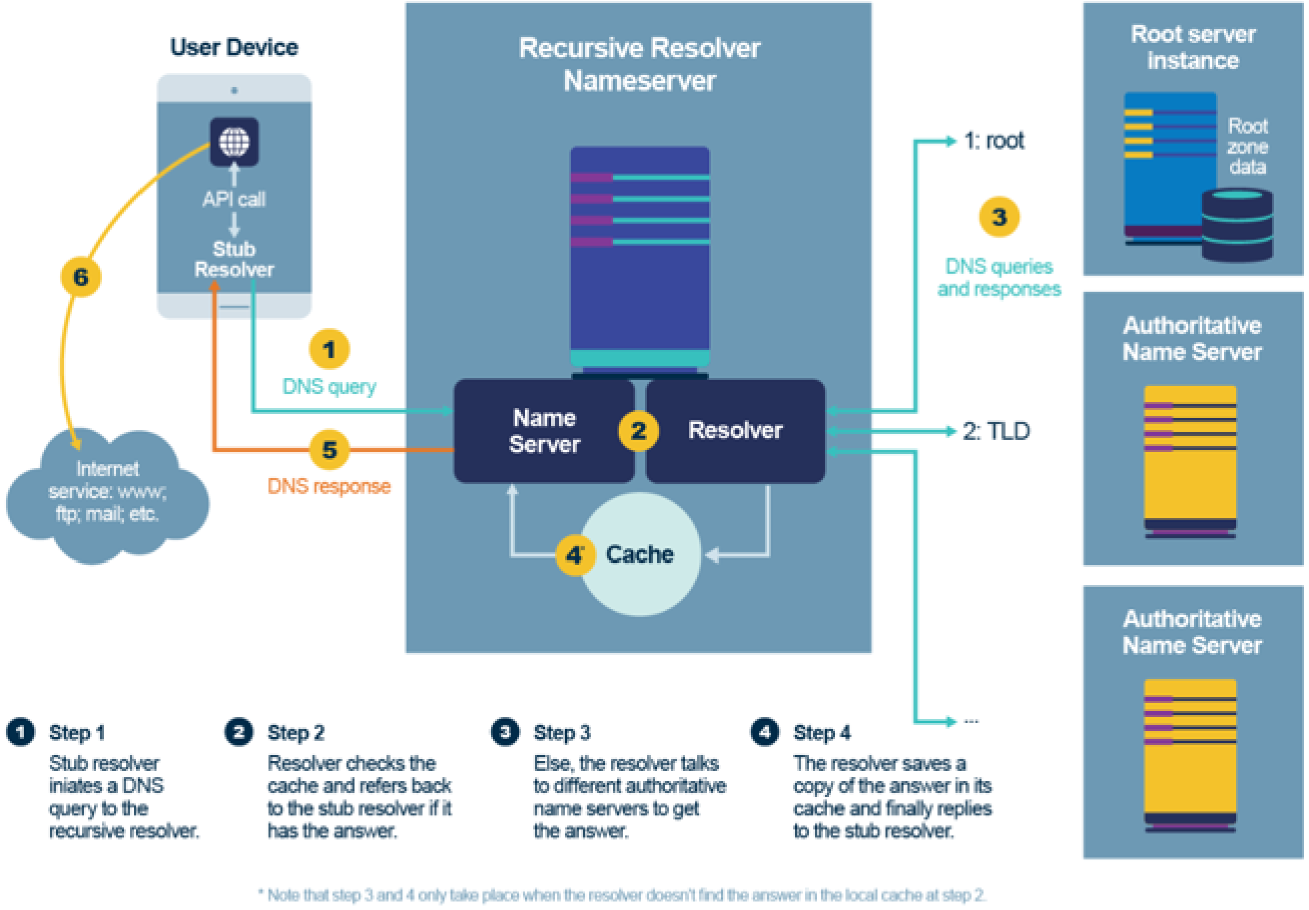
Sample Zone File: *example.com*

```
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
```

Resolution Process

The Resolution Process

Operation of getting the answer for a specific DNS query.



Let's go though resolution process step by step…

Resolution Process

A user types *www.example.com* into Safari, which then calls the stub resolver function to resolve the name

www.example.com
Web site

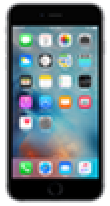
Recursive Resolver
4.2.2.2

Resolution Process

A user types *www.example.com* into Safari, which then calls the stub resolver function to resolve the name

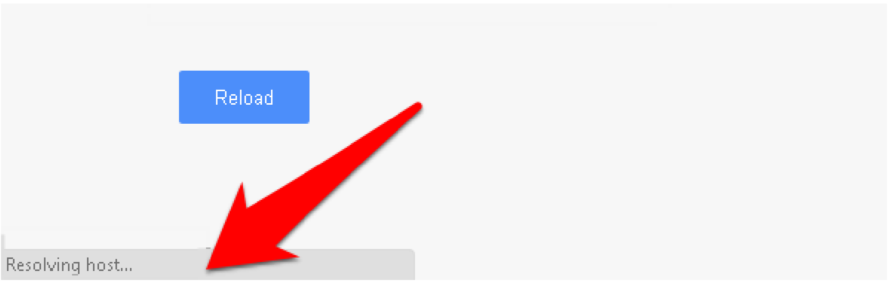
Recursive Resolver
4.2.2.2

www.example.com
Web site



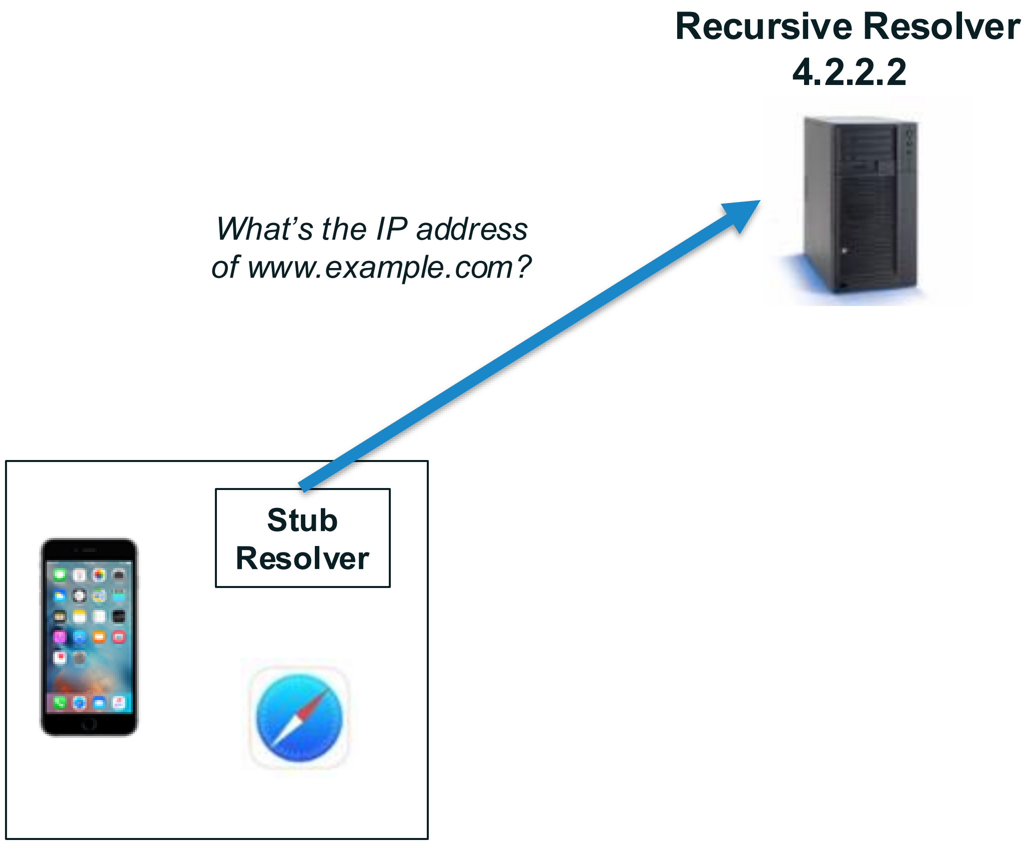
Stub Resolver

"www.example.com"



Resolution Process

The phone’s stub resolver sends a query for *www.example.com*, IN, A to 4.2.2.2

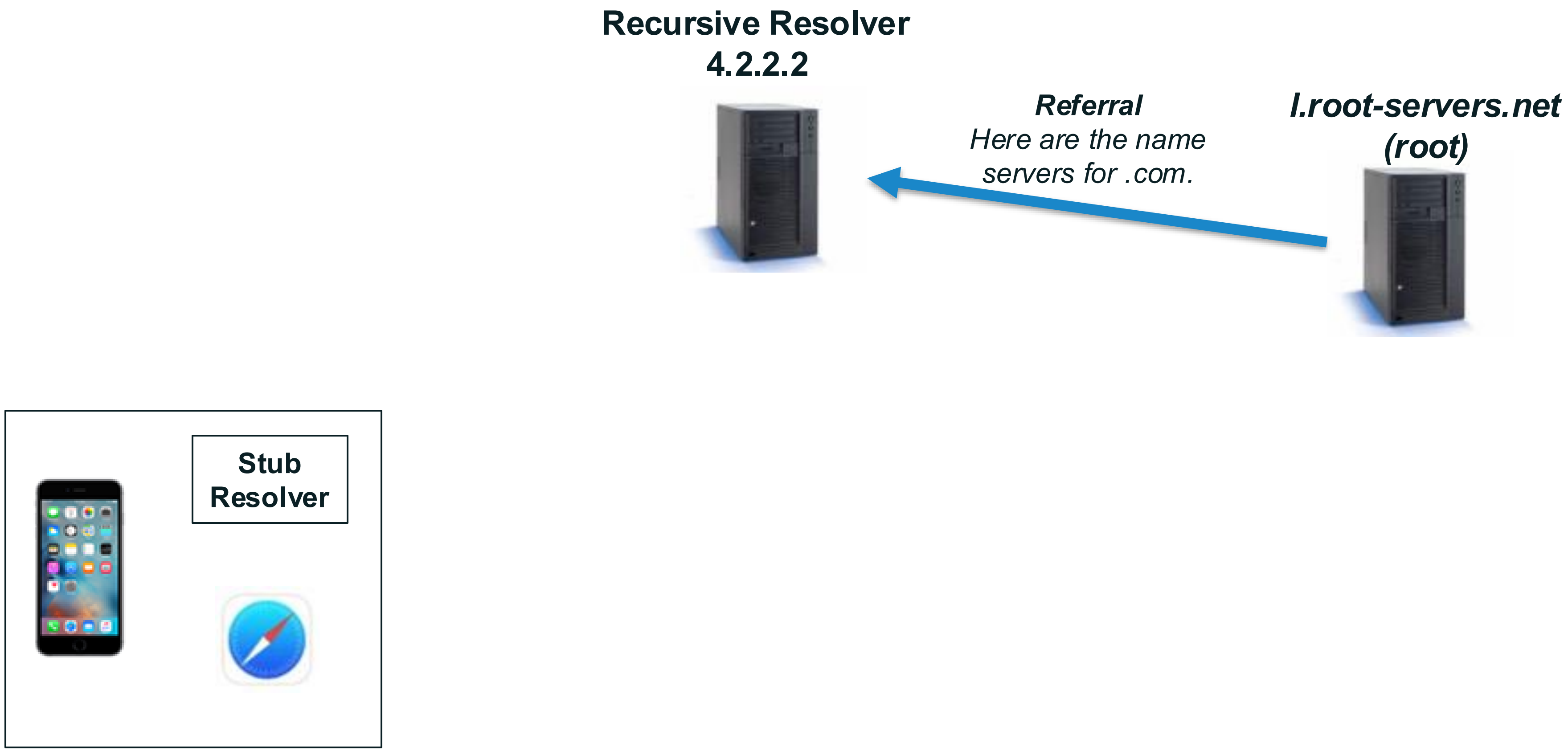


Resolution Process

Recursive resolver 4.2.2.2 has no data cached for *www.example.com*, so it queries a root server

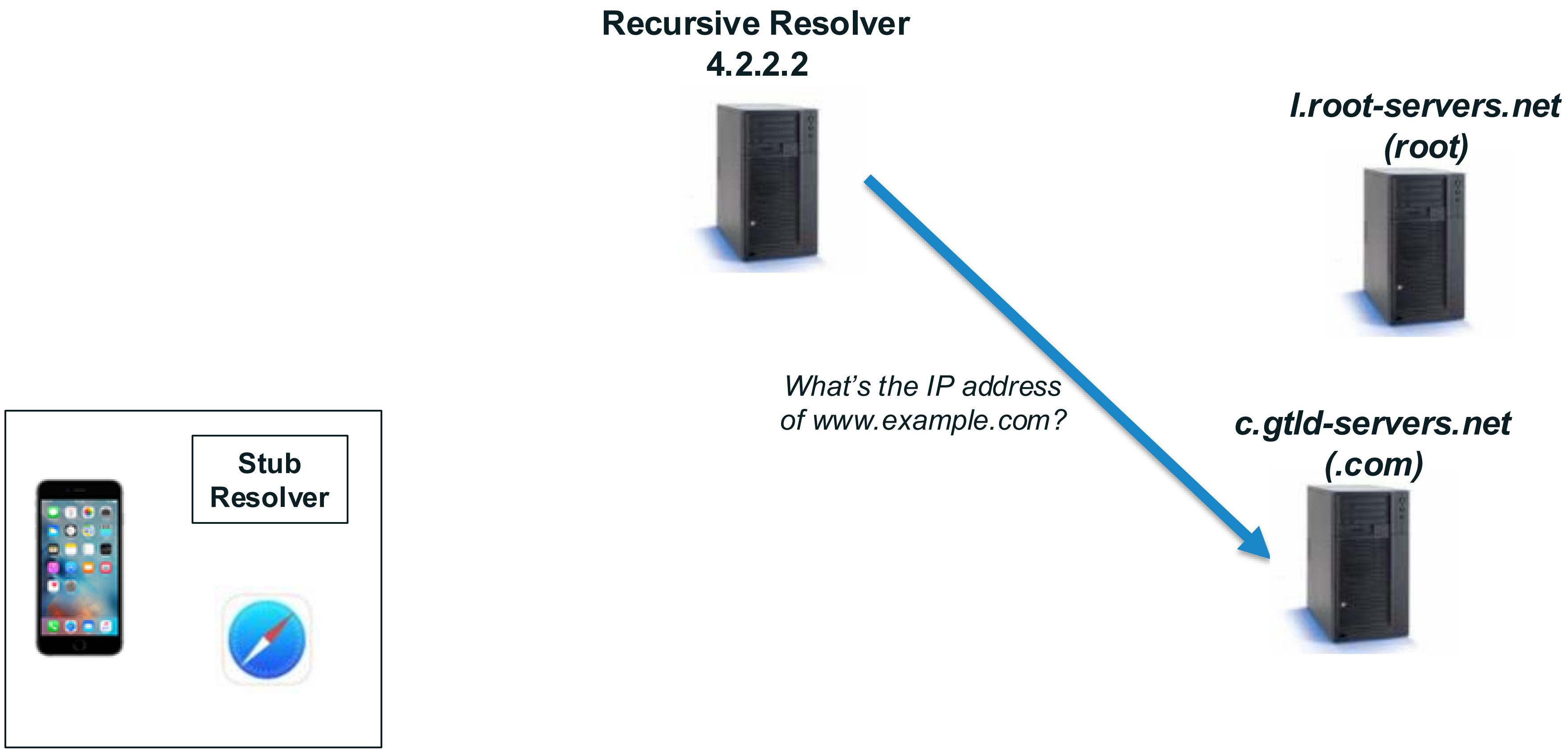
Resolution Process

Root server returns a referral to *.com*



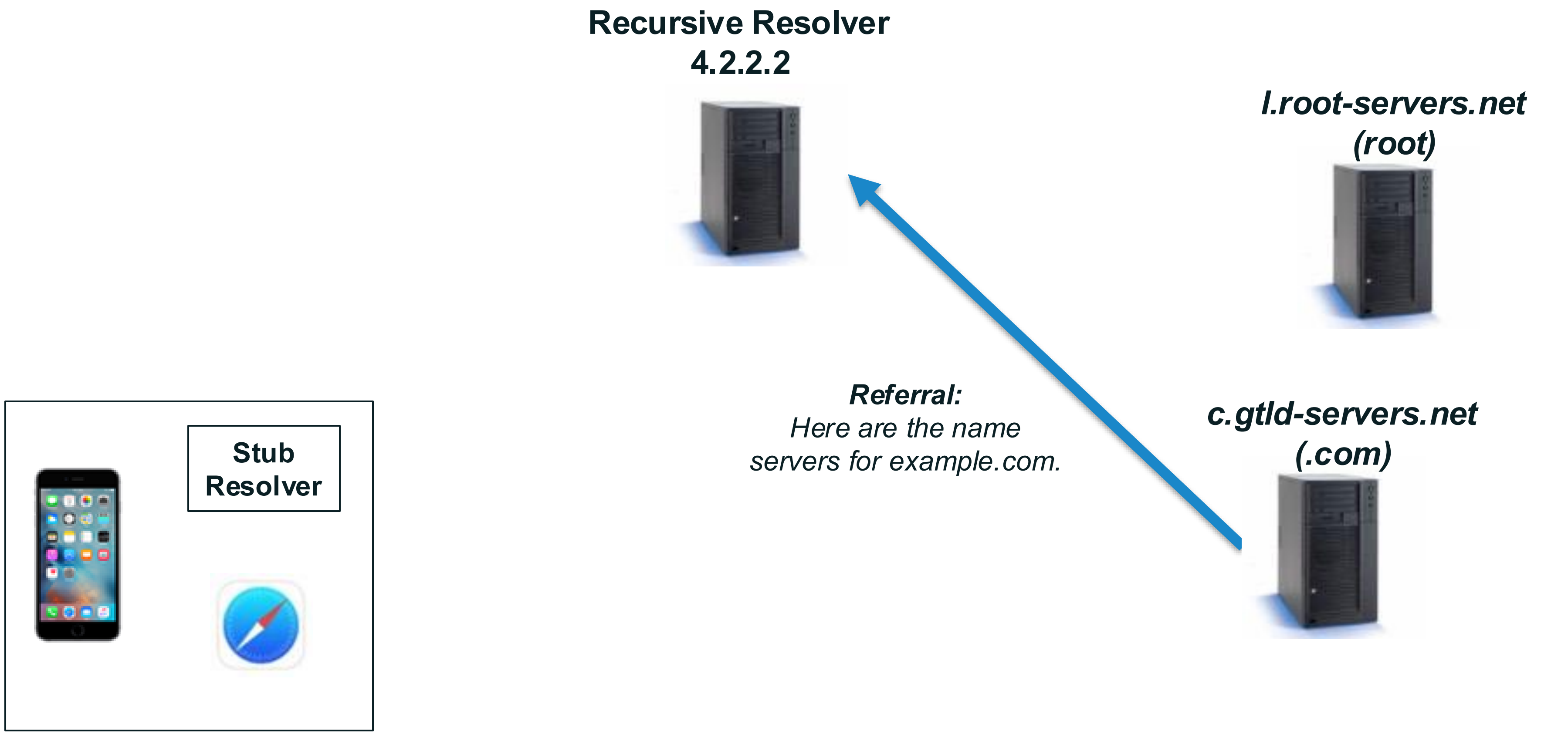
Resolution Process

Recursive resolver queries a *.com* server



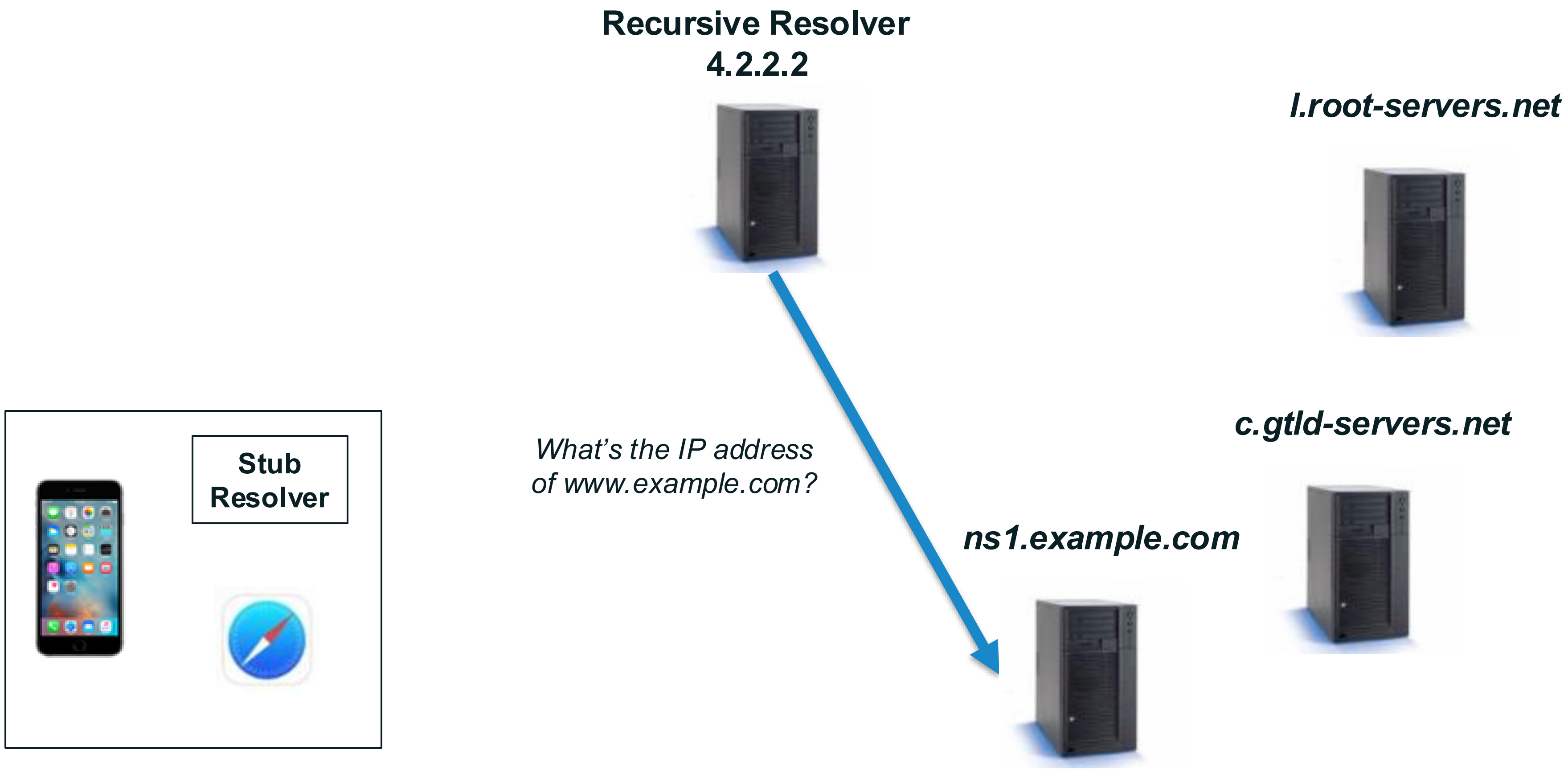
Resolution Process

.com server returns a referral to *example.com*



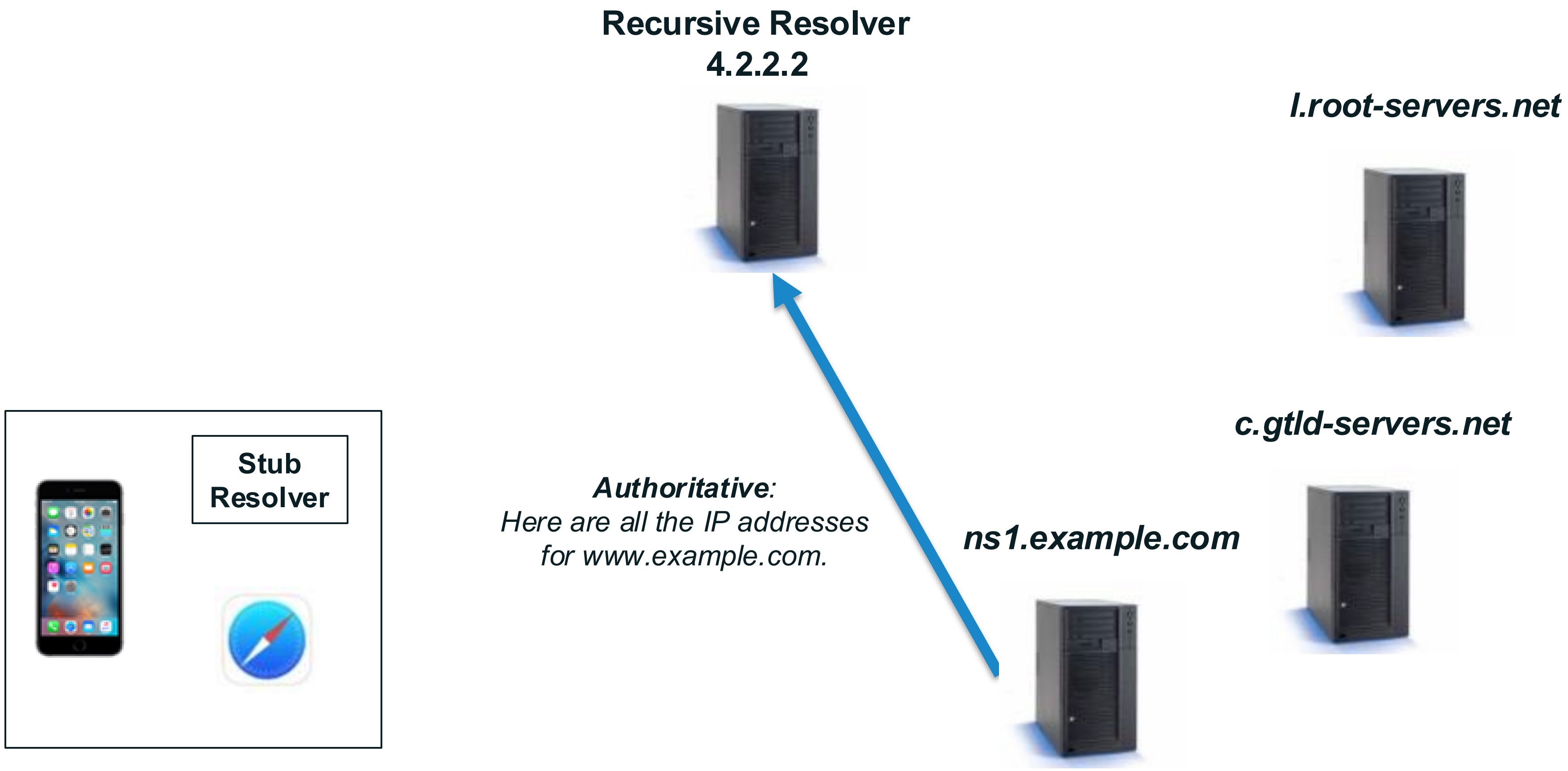
Resolution Process

Recursive resolver queries an *example.com* server



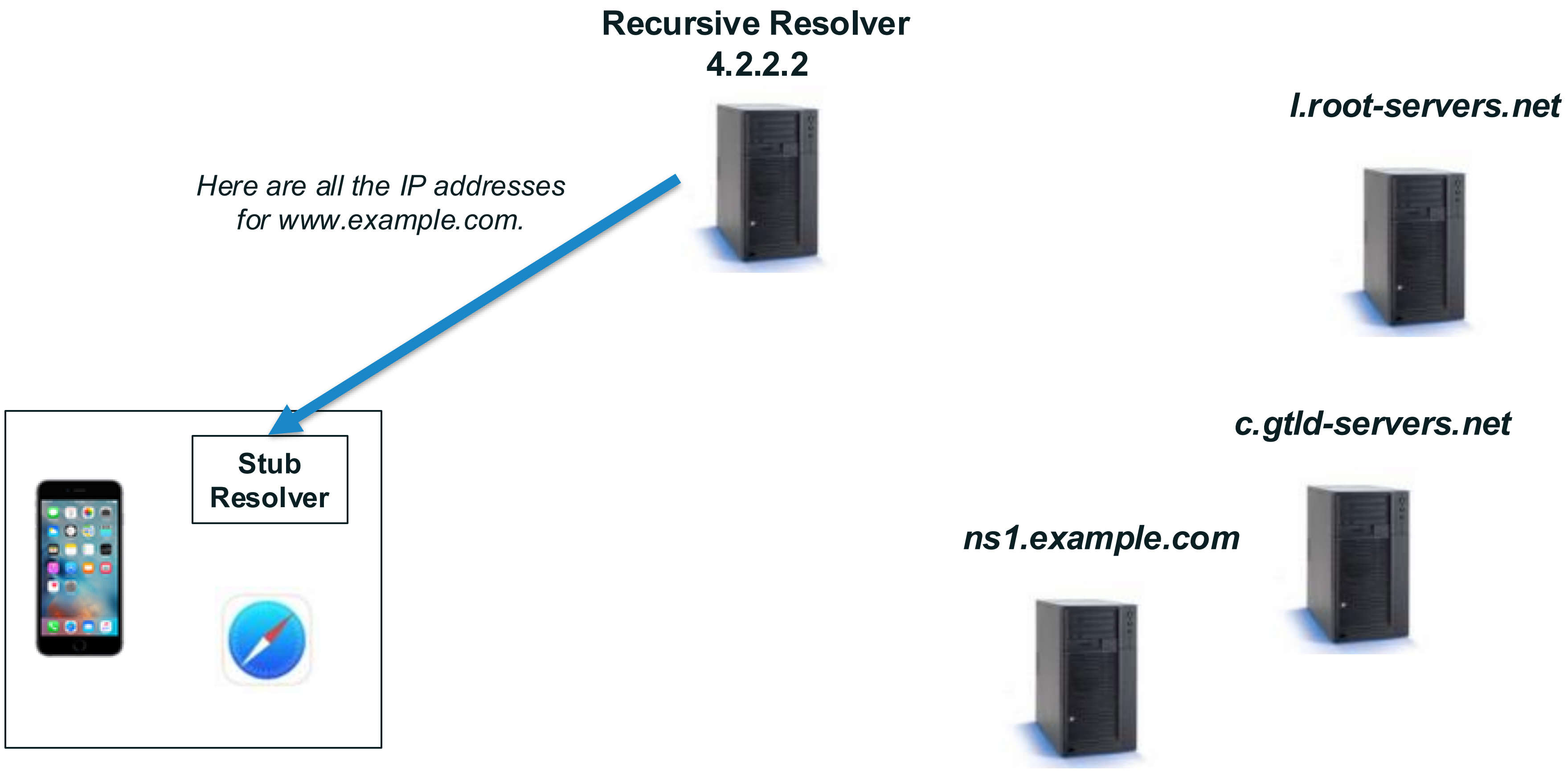
Resolution Process

example.com server returns the answer to the query because it is the authoritative for example.com



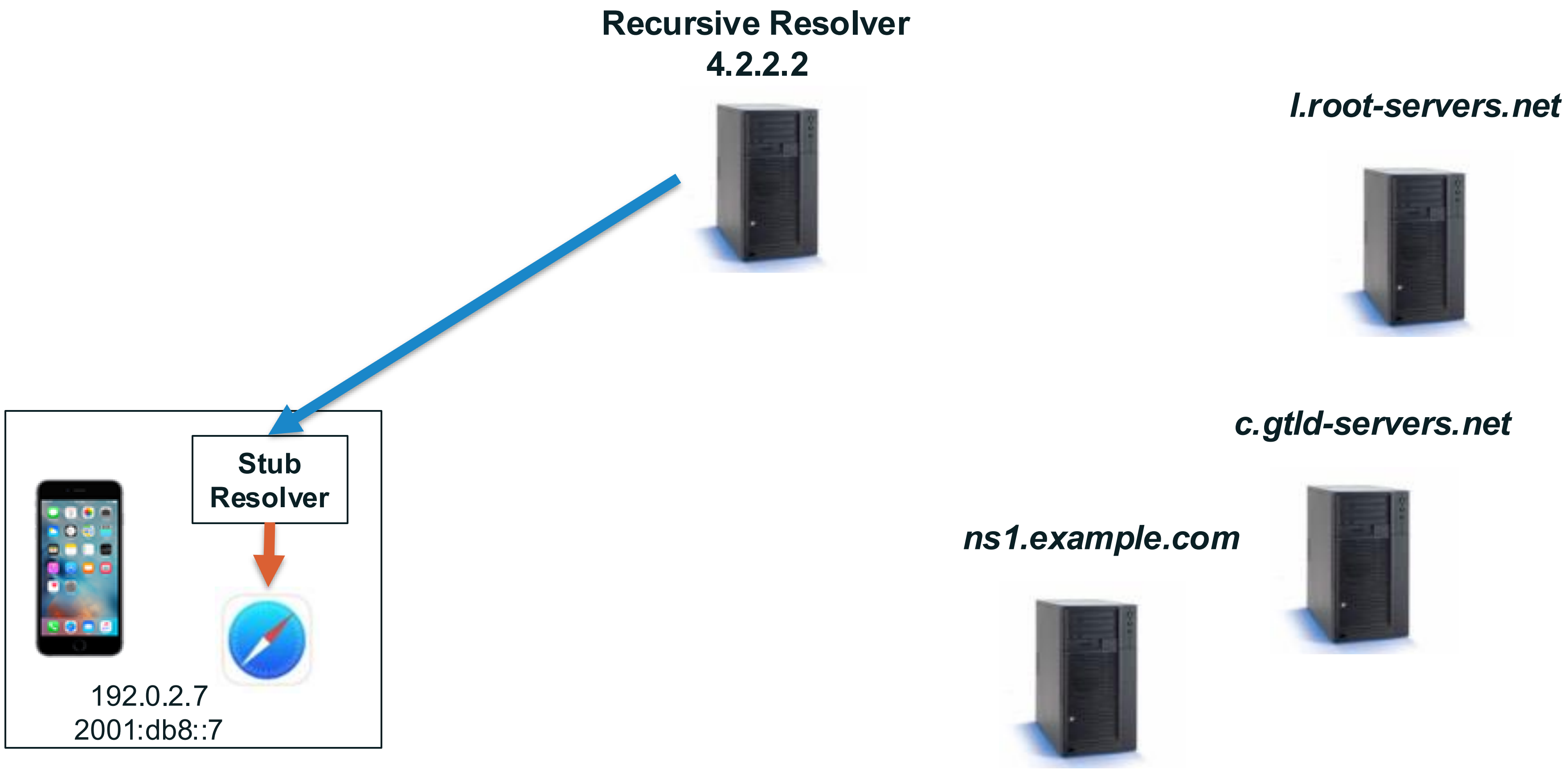
Resolution Process

Recursive resolver returns the answer to the query to the stub resolver



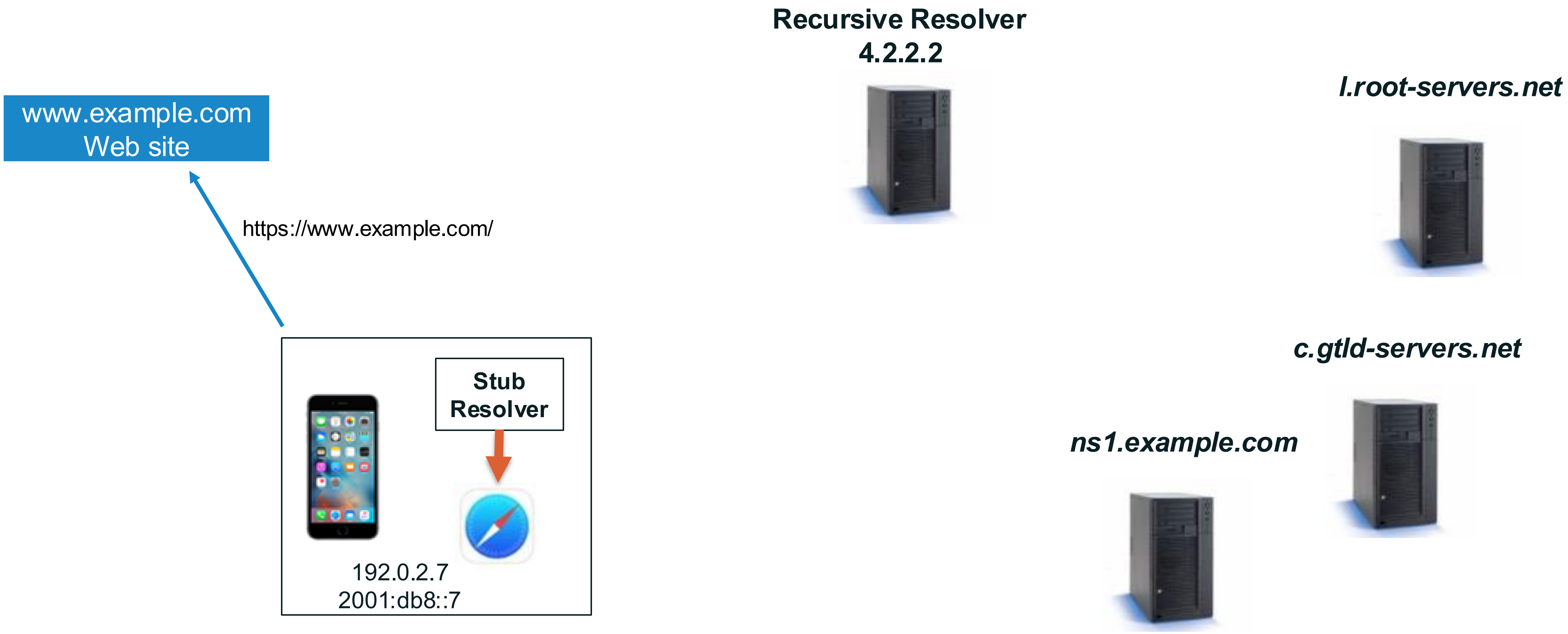
Resolution Process

Stub resolver returns the IP addresses to Safari



Post Resolution Process

Safari can now open the HTTP(s) session with example.com web site.



Post Resolution Process

Example.com web site should reply to the user

Caching

Understanding 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.

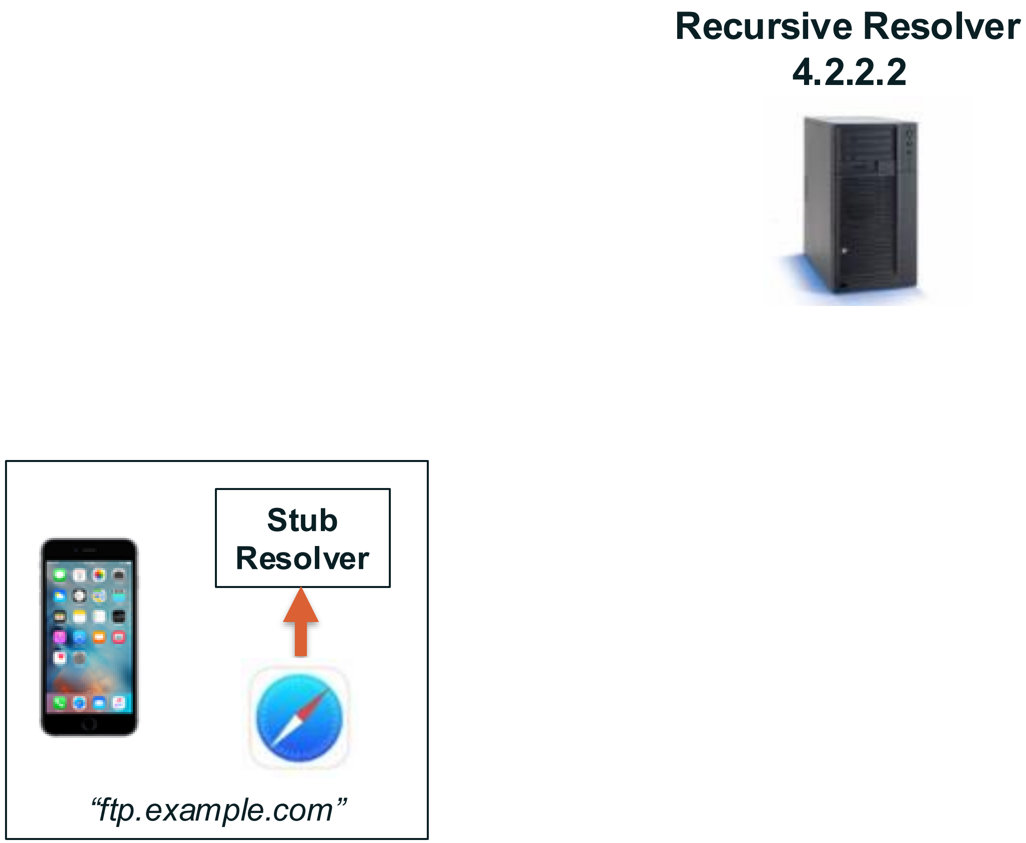
Resolution Process (caching)

- After the previous query, the recursive resolver at 4.2.2.2 now knows:
  - Names and IP addresses of the .com servers
  - Names and IP addresses of the example.com servers
  - IP addresses for www.example.com
- It caches all that data so that it can answer future queries quickly, without repeating the entire resolution process.

Let’s look at another query immediately following the first query . . .

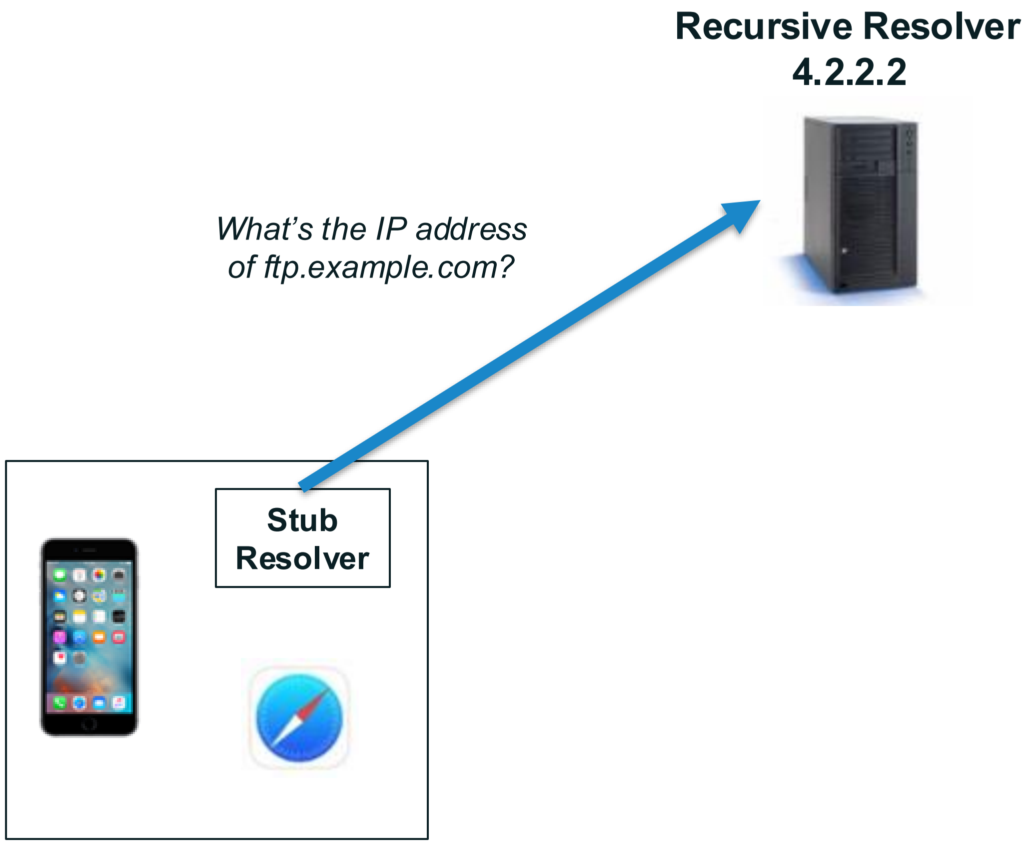
Resolution Process (caching)

A user types *ftp.example.com* into Safari, and it calls the stub resolver function to resolve the name



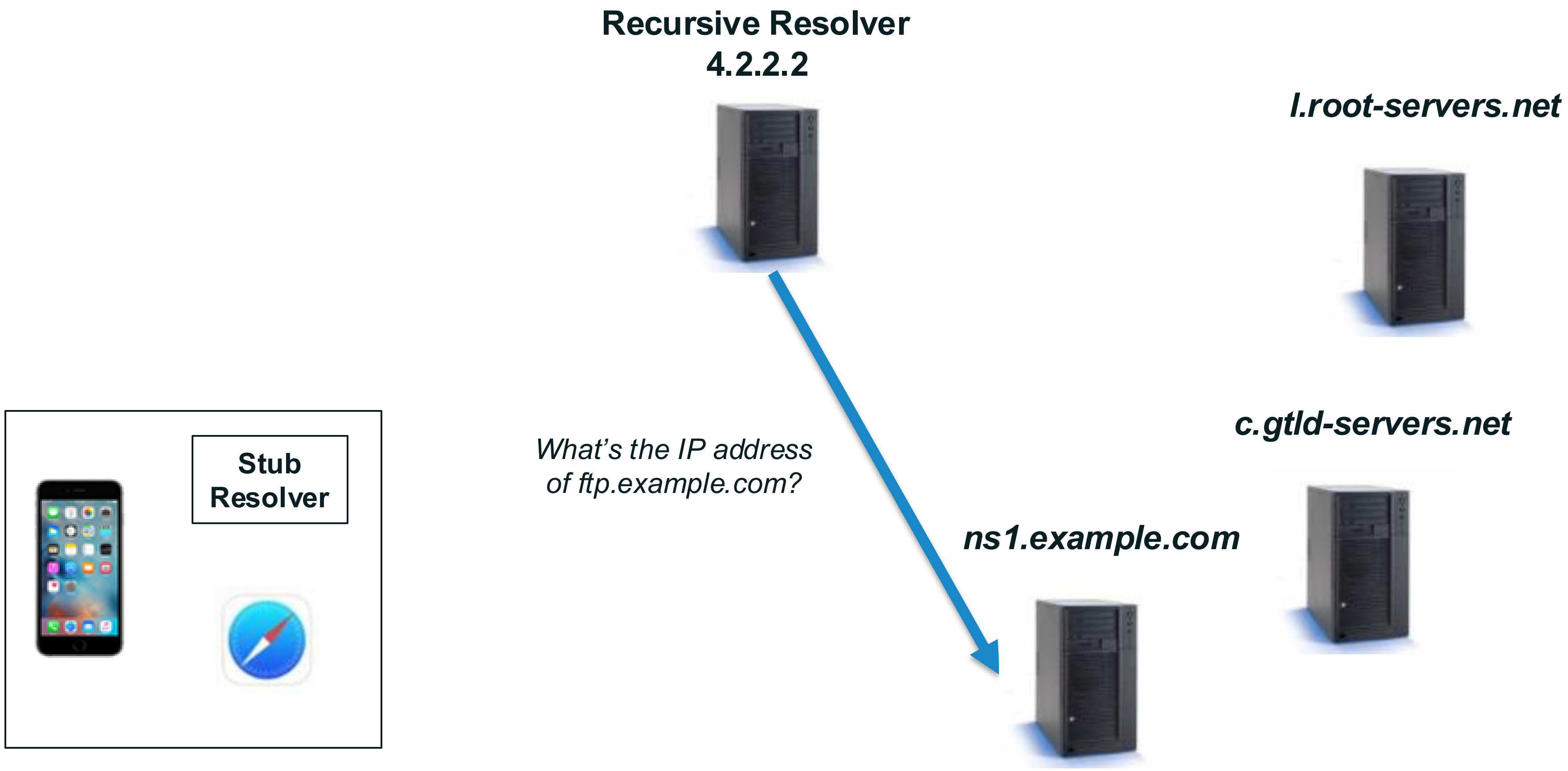
Resolution Process (caching)

The phone’s stub resolver sends a query for *ftp.example.com*/IN/A to 4.2.2.2



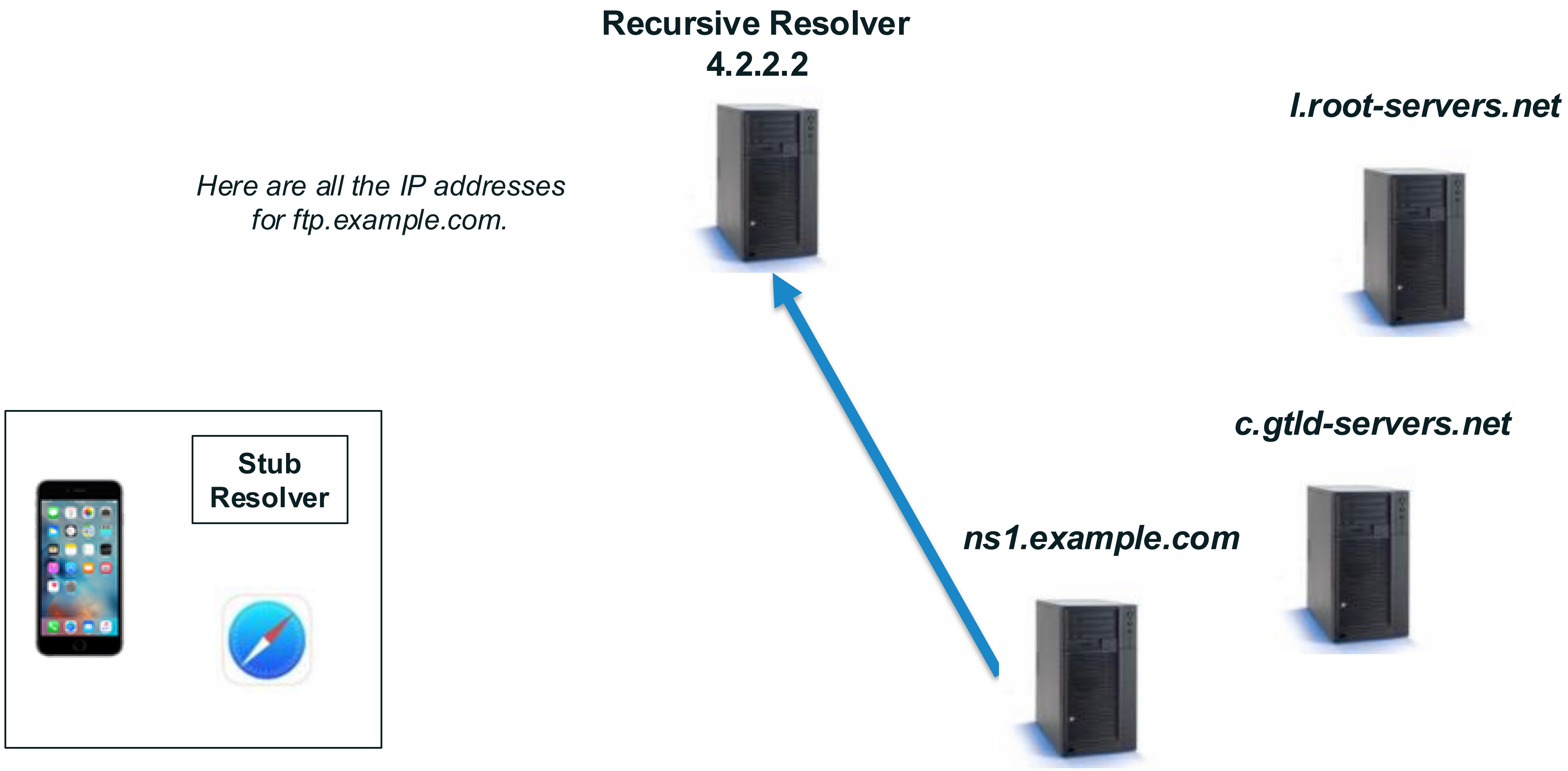
Resolution Process (caching)

Recursive resolver goes directly to example.com servers because it has that data in its cache



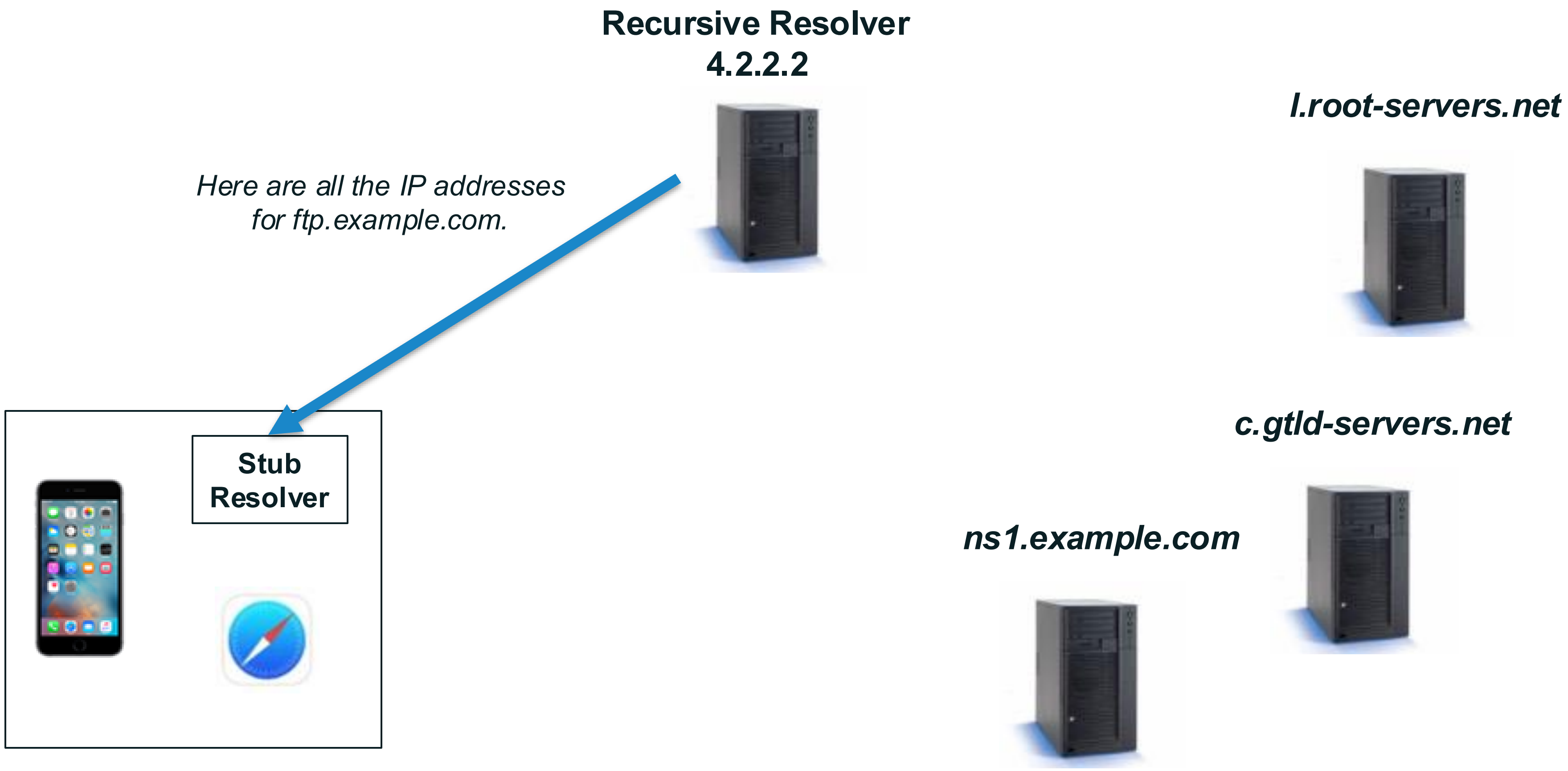
Resolution Process (caching)

example.com server returns the answer to the query



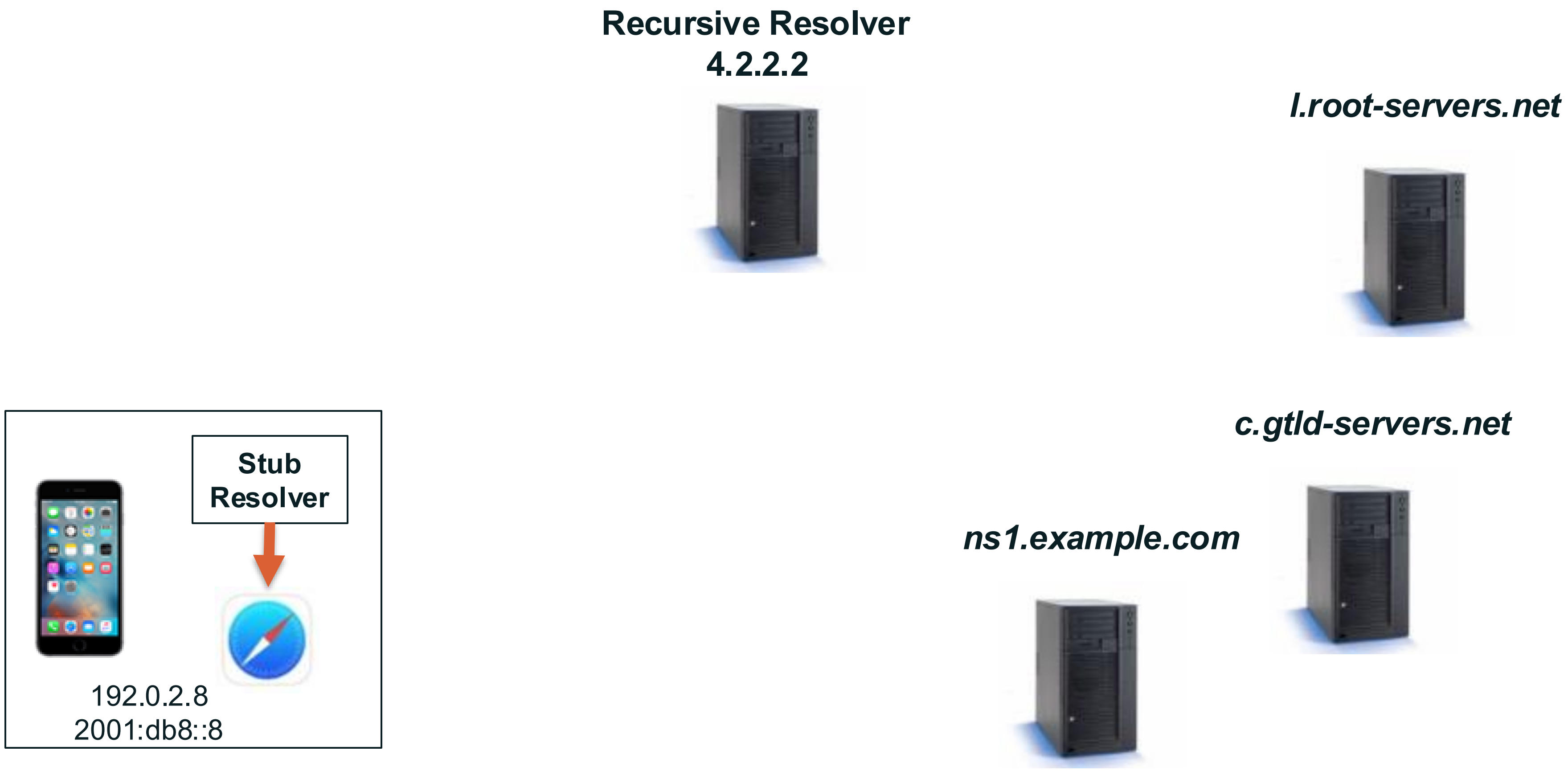
Resolution Process (caching)

Recursive resolver returns the answer to the query to the stub resolver



Resolution Process (caching)

Stub resolver returns the IP addresses to Safari which can now initiate the FTP session to that IP address.



DNS Software

DNS Software overview

- Diversity of software platforms and vendors: commercial and open source.
  - Open Source: BIND, Unbound, Knot Resolver, PowerDNS Recursor, DNSMASQ, …
  - Commercial: Windows, Nominum Vantio (now part of Akamai), Secure64 DNS Cache, …
  - Overview : https://en.wikipedia.org/wiki/Comparison_of_DNS_server_software
- This list may be incomplete, and does not imply endorsement of any specific package

ISC's BIND

- Authoritative server and cache all-in-one
- http://isc.org/
- Always changing, see current version on web site: https://www.isc.org/downloads/bind/
- Longest track record in DNSSEC

NLnetLab's Unbound

- a caching-only name server with DNSSEC built in
- http://www.unbound.net/
- "unbound" is a play on the word "bind"

cz.nic's Knot Resolver

- a caching-only name server with DNSSEC built in
- http://www.knot-resolver.cz/download/
- "Knot" is a play on the words "bind" and "unbound" (see a trend?)

PowerDNS Recursor

- Caching resolver
- Supports DNSSEC validation
- https://www.powerdns.com/documentation.html
- Also an authoritative server
- name is not related to BIND, unbound, Knot

DNSMASQ

- provides network infrastructure for small networks: DNS, DHCP, router advertisement and network boot
- authoritative and cache
- Supports DNSSEC (validation)
- main page: http://www.thekelleys.org.uk/dnsmasq/doc.html

DNS Software overview

- Popular resolvers: BIND and Unbound. PowerDNS recursor and Knot resolver are quite new.
- Popular authoritatives: BIND9 and PowerDNS. Others like NSD and Knot are raising as well.
- We will mainly use BIND, Unbound and NSD in our labs.

Software	Auth	Recursive	DNSSEC	DB / API
ISC BIND9	X	X	X	
PowerDNS	X		X	X
PowerDNS Recursor		X	X	
NSD	X		X	
Unbound		X	X	
Knot DNS	X		X	
Knot Resolver		X	X	

DNS Software: BIND

- Version 4 released with BSD 4.3 in 1986
- Currently at version 9.18
- BIND 10 was once in development, but has been abandoned
- Most feature rich DNS implementation out there: ACL, views, DB API, dynamic DNS, DNSSEC signing and validation, etc.
- Often considered “the reference”
  - BIND zone format is the de-facto notation
- More details at: https://www.isc.org/bind/
- Used in many commercial products

DNS Software: NSD

- Developed by NLNetLabs
- Authoritative only
- Developed to mitigate risk of a single bug
- Taking out all BIND implementations
- Several root servers use NSD
- Zones are “compiled” into a precalculated “on the wire” format
  - all possible answers are calculated, then stored into a binary DB, ready to send out
  - very fast

DNS Software: Unbound

- Developed by NLNetLabs
- Resolver only
- Developed with performance in mind
- More lightweight that BIND
  - More efficient memory usage
  - More features to control caching
  - Fast...

DNS Resilience

ICANN

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

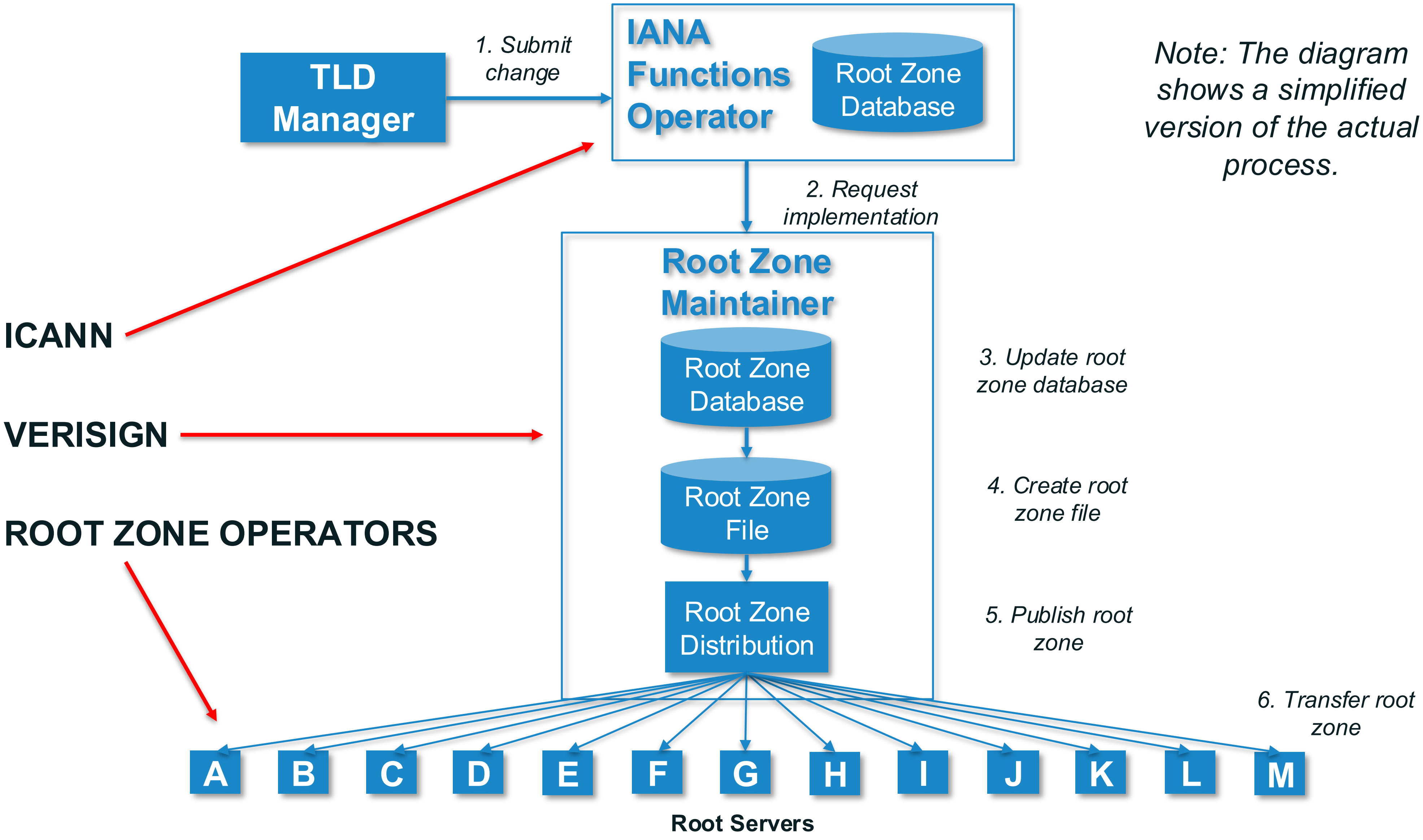
DNS Resilience #2 – (Root Server System's Resiliency)

- 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

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)
- **F** Information Systems Consortium (ISC)
- **G** United States Department of Defense (US DoD)
- **H** United States Army (Aberdeen Proving Ground)
- **I** Netnod
- **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)

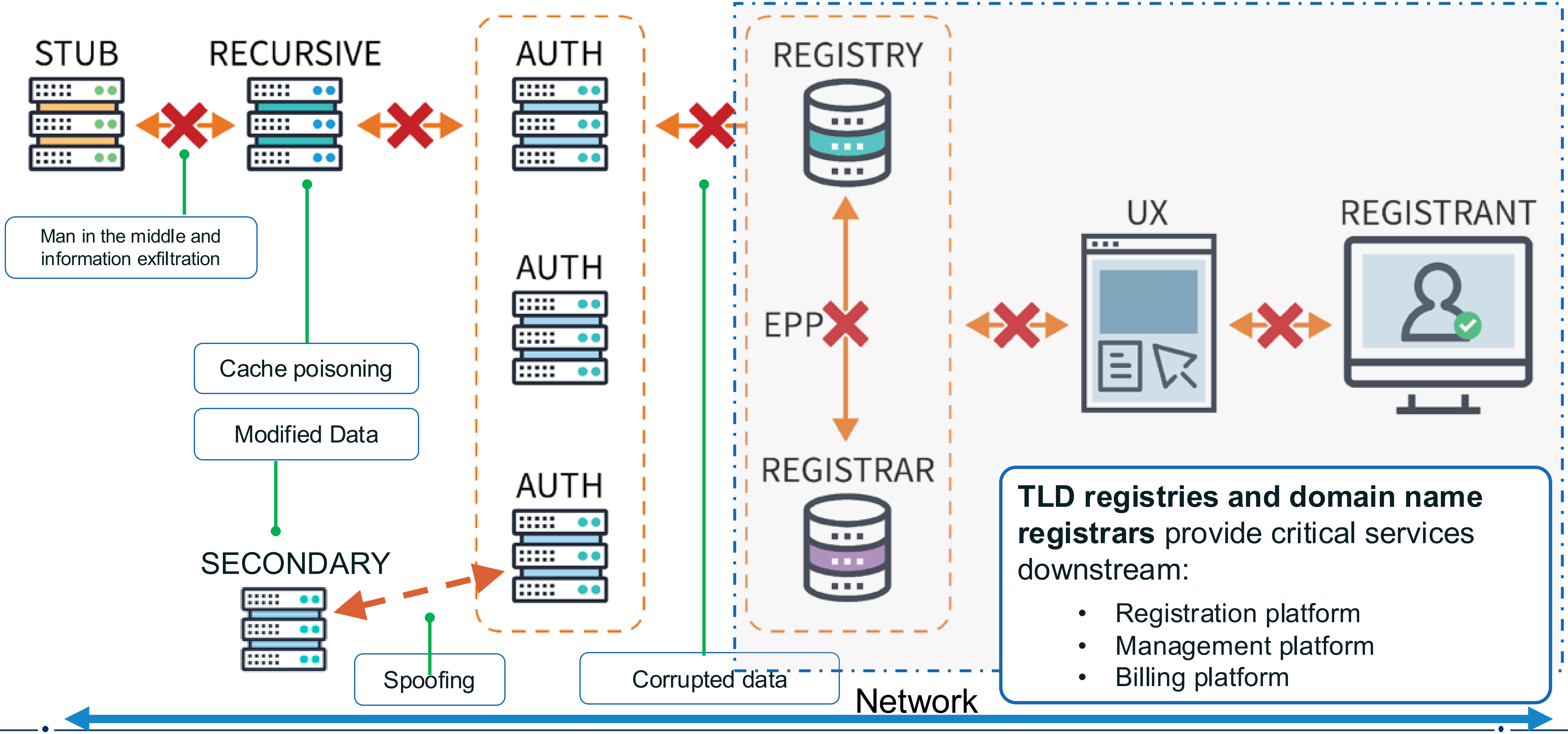
Root Zone Change Process



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

Discussion: Which root server instance(s) are hosted in your country ?

Potential Target Points of the DNS Infrastructure/Ecosystem



TLD registries and domain name registrars provide critical services downstream:

- Registration platform
- Management platform
- Billing platform

Time for practice !

1. Configure your own zone’s primary and secondaries NS
2. Confirm they are all in sync, serving and responding well for the zone.

Thank you

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