# DNS caching

- Performing all these queries takes time
  - And all this before actual communication takes place
  - E.g., 1-second latency before starting Web download
- Caching can greatly reduce overhead
  - The top-level servers very rarely change
  - Popular sites (e.g., www.cnn.com) visited often
  - Local DNS server often has the information cached
- How DNS caching works
  - DNS servers cache responses to queries
  - Responses include a “time to live” (TTL) field
  - Server deletes cached entry after TTL expires (OR SO THEY SAY)

# Negative caching

- Remember things that don’t work
  - Misspellings like www.cnn.comm and www.cnnn.com
  - These can take a long time to fail the first time
  - Good to remember that they don’t work
  - … so the failure takes less time the next time around

- But: negative caching is optional
  - And not widely implemented

# Exercise

Perform a DNS query for nyu using first the authoritative DNS server (ns1.nyu.net) and then your local server.

Note: When using nslookup on Windows, you need to specify the -debug flag to get the relevant information for this task. For example:

```
nslookup -debug <Domain Name> <DNS-Server>
```

Compare the ANSWER SECTION of the responses. Can you see differences between the answers from your local DNS server and the authoritative server? Run the query to your local server multiple times to make the differences more obvious.

# Exercise

Perform a DNS query for nyu using first the authoritative DNS server (ns1.nyu.net) and then your local server.

Note: When using nslookup on Windows, you need to specify the -debug flag to get the relevant information for this task. For example:

```
nslookup -debug <Domain Name> <DNS-Server>
```

Compare the ANSWER SECTION of the responses. Can you see differences between the answers from your local DNS server and the authoritative server? Run the query to your local server multiple times to make the differences more obvious.
The answers differ in the time to live (TTL). While the TTL is constant in the replies from the authoritative DNS server, it varies in the replies from the local server.

# Exercise

What is the reason for this difference?

DNS can be used to balance the incoming load. What are the considerations one has to make when using DNS load balancing with respect to the TTL?

# Exercise

What is the reason for this difference?
Solution: The local DNS server caches replies to requests. To ensure that it does not keep outdated information in its cache, each authoritative name server attaches a TTL to its replies. The TTL tells the local DNS server how long it can store the reply in the cache and use it to reply to requests.

DNS can be used to balance the incoming load. What are the considerations one has to make when using DNS load balancing with respect to the TTL?
Solution: With low TTLs we can ensure that we can shift the load quickly. However, low TTLs also mean that our authoritative DNS server will get many more requests.

# DNS Resource Records

DNS: distributed DB storing resource records (RR)

RR format: (name, value, type, ttl)

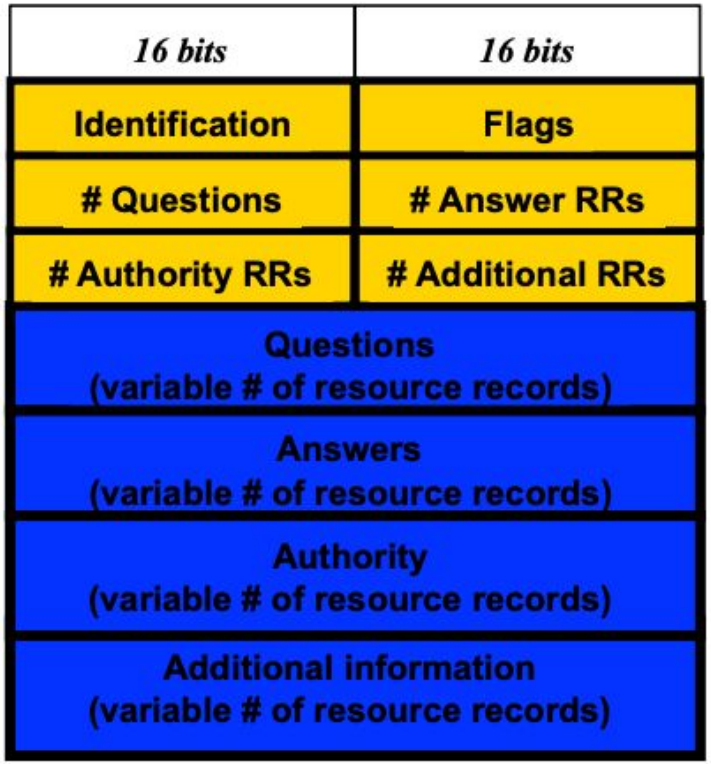
- Type=A
  - Name: hostname
  - Value: IP address
- Type=AAAA
  - Name: hostname
  - Value: IPv6 address
- Type=NS
  - Name: domain (e.g., foo.com)
  - Value: hostname of authoritative server for this domain

# DNS Resource Records

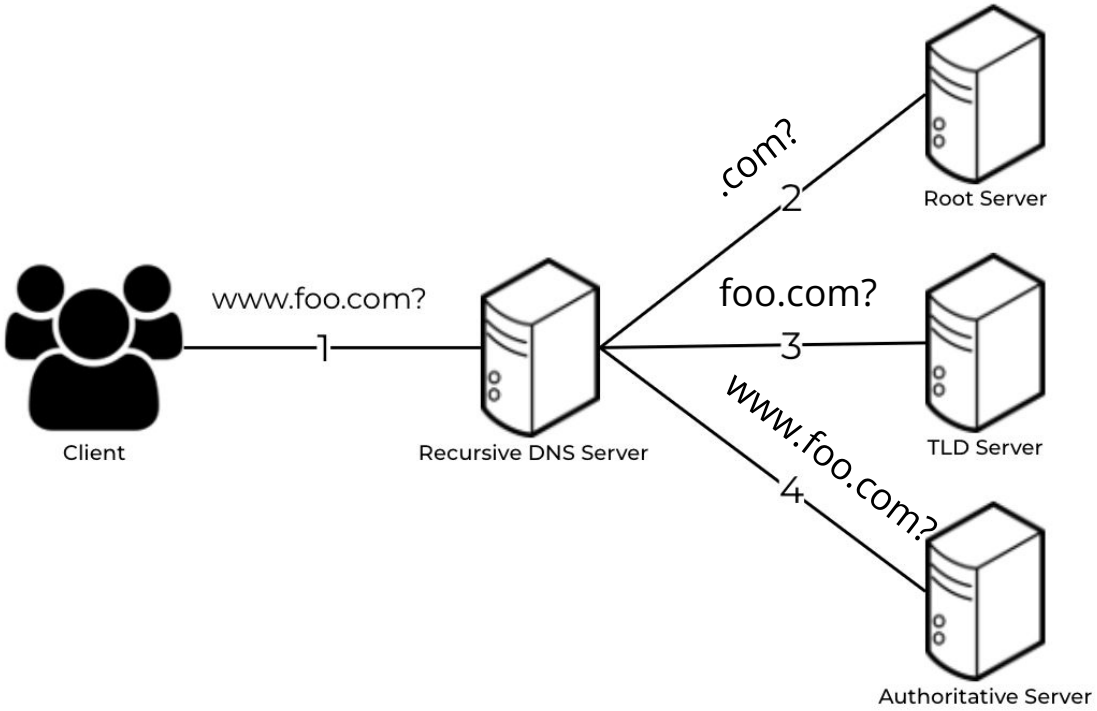
- Type=PTR
  - Name: reversed IP quads (e.g., 78.56.34.12.in-addr-arpa)
  - Value: corresponding hostname
- Type=CNAME
  - Name: alias name for some “canonical” name (e.g., www.cs.mit.edu is really eecs.mit.edu)
  - Value: canonical name
- Type=MX
  - Value: name of mailserver associated with name
  - Also includes weight / preference

# DNS Protocol

- DNS protocol: query and reply messages, both with same message format
- Message header:
  - Identification: 16 bit # for query, reply to query uses same #
- Flags:
  - Query or reply
  - Recursion desired
  - Recursion available
  - Reply is authoritative
- Plus fields indicating size (0 or more) of optional header elements



# Any security concerns about this architecture?

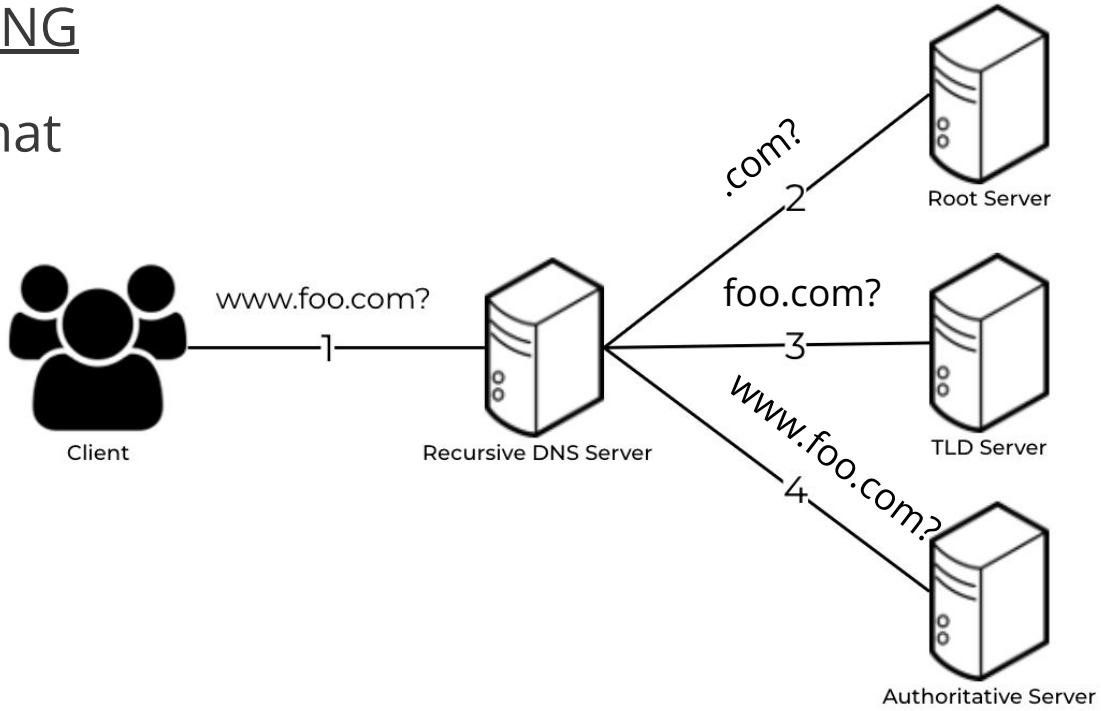


# Any security concerns about this architecture?

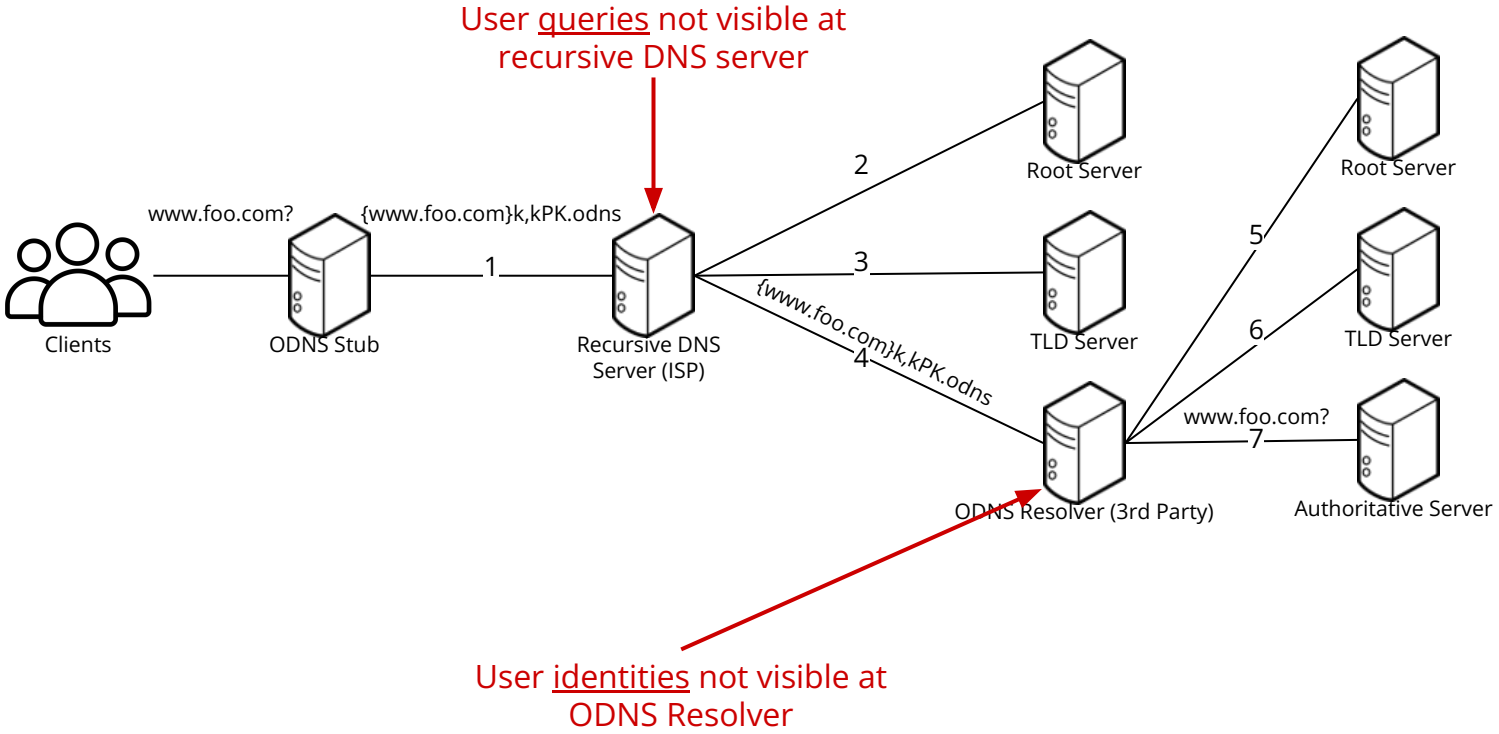
Recursive learns EVERYTHING

No surprise - companies that enjoy data run public DNS recursives

- Google (8.8.8.8)
- Cloudflare (1.1.1.1)

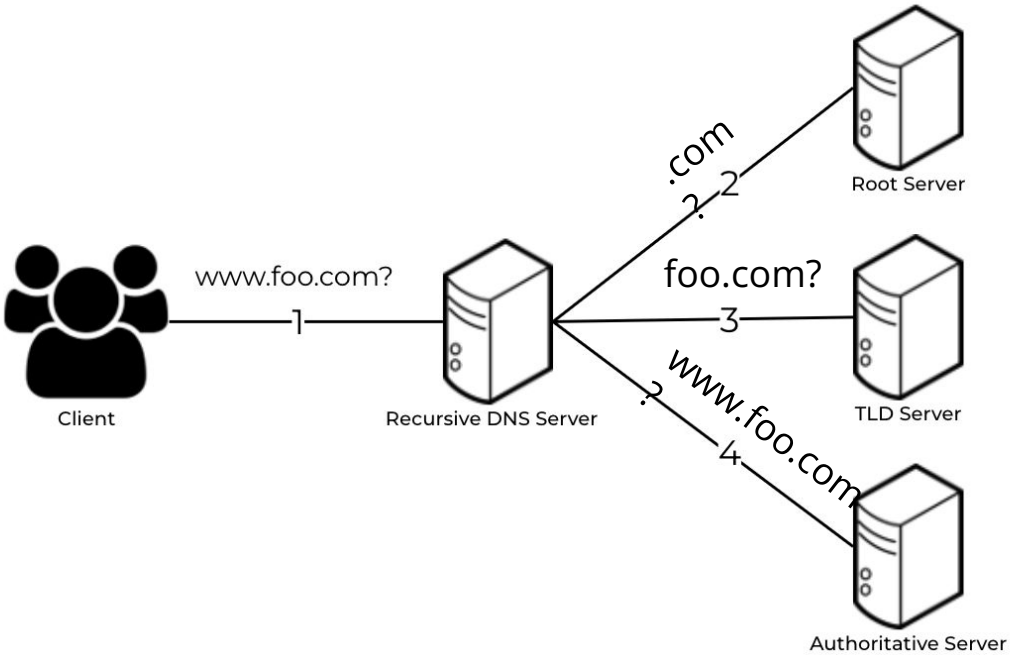


# Oblivious DNS



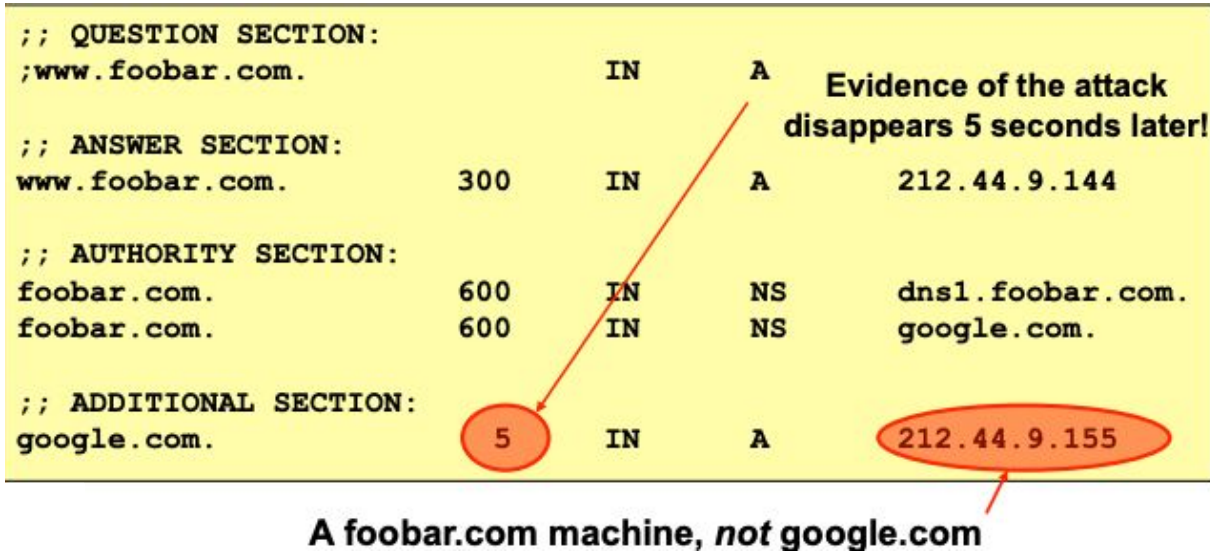
# Security problem 2: Starbucks

- As you sip your latte and surf the Web, how does your laptop find google.com?
- Answer: it asks the local name server per Dynamic Host Configuration Protocol (DHCP) …
  - … which is run by Starbucks or their contractor
  - … and can return to you any answer they please
  - … including a “man in the middle” site that forwards your query to Google, gets the reply to forward back to you, yet can change anything they wish
- How can you know you’re getting correct data?
  - Today, you can’t. (Though if site is HTTPS, that helps)
  - One day, maybe: DNSSEC extensions to DNS



# Security problem 3: cache poisoning

- Suppose you are a Bad Guy and you control the name server for foobar.com. You receive a request to resolve www.foobar.com and reply:



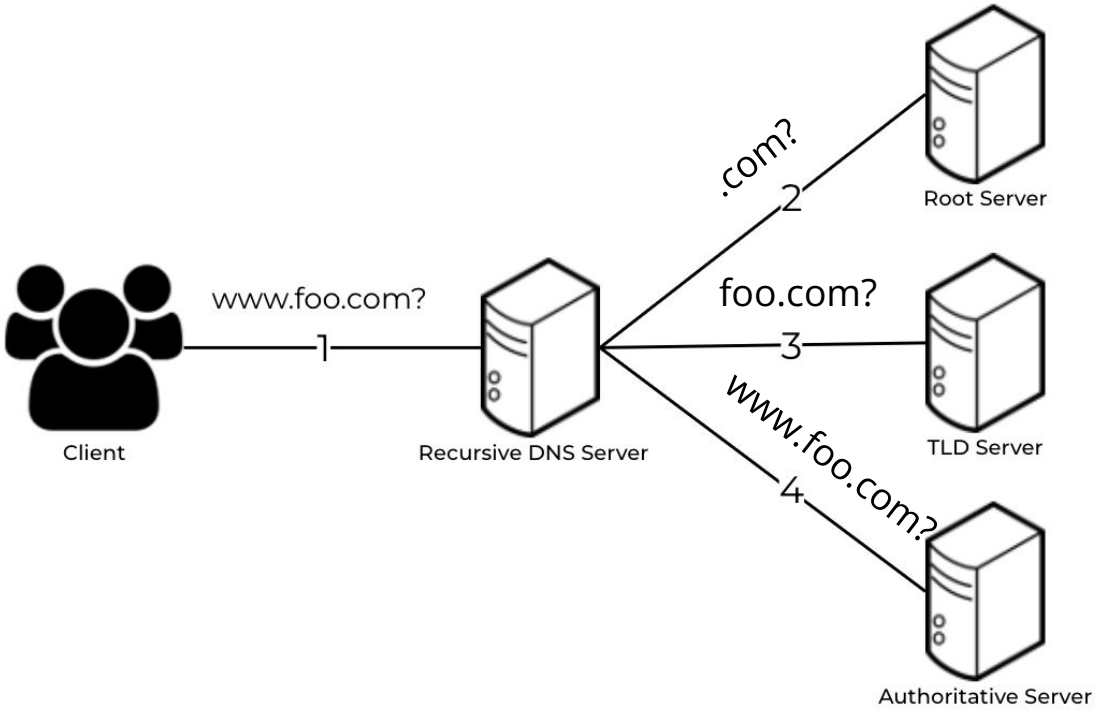
# Security problem 3: cache poisoning

- Okay, but how do you get the victim to look up www.foobar.com in the first place?
- Perhaps you connect to their mail server and send
  - HELO www.foobar.com
  - Which their mail server then looks up to see if it corresponds to your source address (anti-spam measure)
- Note, with compromised name server we can also lie about PTR records (address - name mapping)
  - E.g., for 212.44.9.155 = 155.44.9.212.in-addr.arpa return google.com (or whitehouse.gov, or whatever)
    - If our ISP lets us manage those records as we see fit, or we happen to directly manage them

# Security problem 3: cache poisoning

- Suppose Bad Guy is at Starbuck's and they can sniff (or even guess) the identification field the local server will use in its next request:
- They:
  - Ask local server for a (recursive) lookup of google.com
  - Locally spoof subsequent reply from correct name server using the identification field
  - Bogus reply arrives sooner than legit one
- Local server duly caches the bogus reply!
  - Now: every future Starbuck customer is served the bogus answer out of the local server's cache
    - In this case, the reply uses a **large** TTL

# What transport should we use for DNS?



# What transport should we use for DNS?

Traditionally, UDP used for queries

- But we need reliability: must implement this on top of UDP

Try alternate servers on timeout

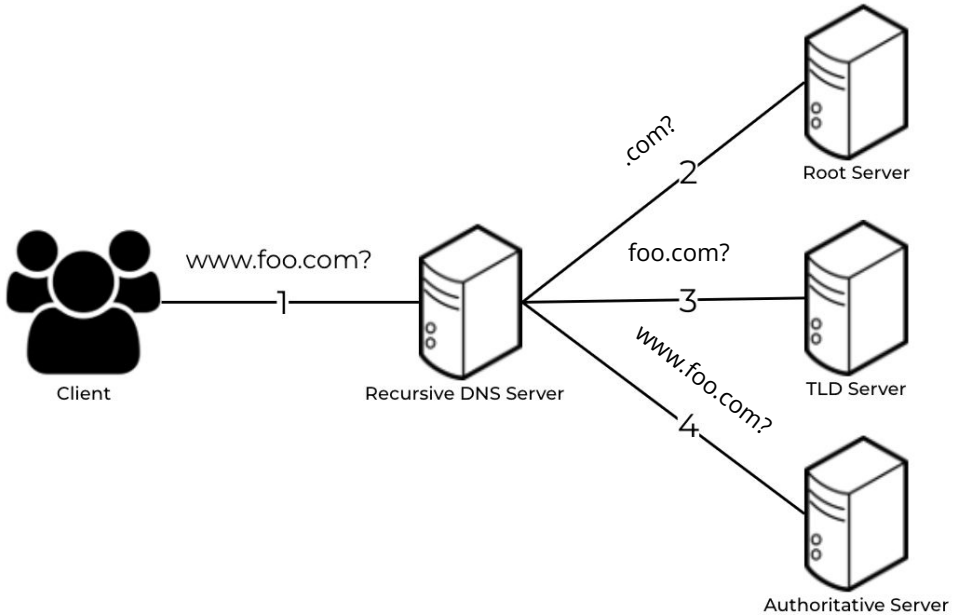
- Exponential backoff when retrying same server

DNS servers are replicated

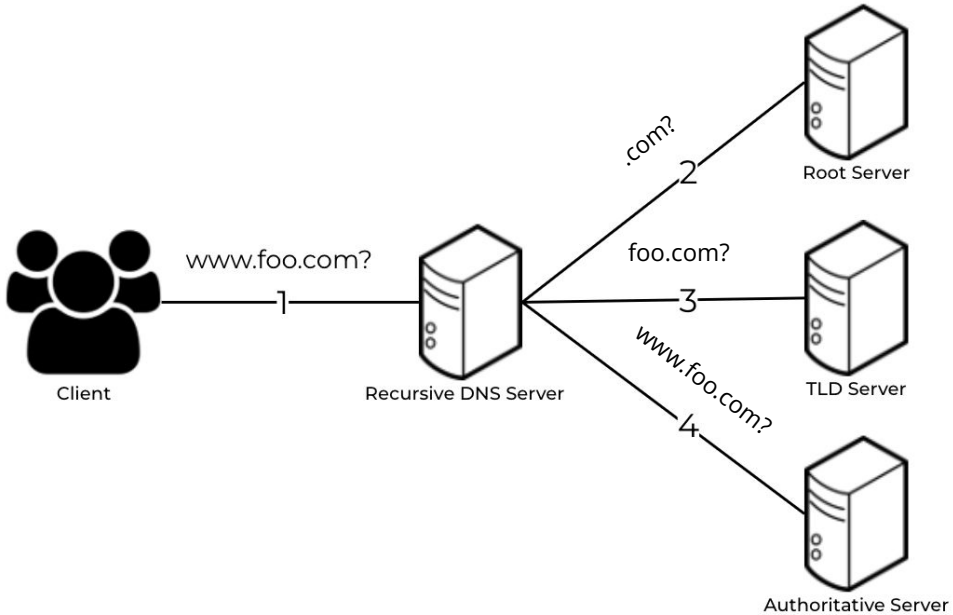
- Name service available if at least one replica is up
- Queries can be load-balanced between replicas

Same identifier for all queries

- Don’t care which server responds



# We’ve started moving towards TCP. Why?



# DNS Privacy

- Large arguments over DNS in recent years
- Current versions of Firefox are using DNS over HTTPS by default
  - No longer uses your own ISP
  - Uses third party for DNS resolution — Cloudflare
  - Do you know/trust them, or are you just securely giving your DNS info to an untrusted third party?

# DNS Privacy

- 

- info to an untrusted third party?

# DNS wrap up

- Full distributed, hierarchical system
  - Root -> TLD -> Authoritative
  - Anycast
- Underpins the modern Internet
  - Old way (/etc/hosts) couldn’t scale
- Recursive vs. Iterative modes
- Many different types of records
  - A / AAAA - used to translate domain to IP
  - PTR - reverse of A / AAAA
  - CNAME - name redirection
- Lots of work on DNS privacy / security these days