EECS 489

Computer Networks

CDN and DNS

Announcements

- Please form teams for Assignment 2, 3, and 4
- Working without a group?
- Late days
  - 3 late days for the duration of the semester


ASSIGNMENTS ON SYLLABUS ARE
CLOSER THAN THEY APPEAR

Agenda

- CDN: Content Distribution Network
- DNS: Domain Name System

Recap: Improving HTTP performance

- Optimizing connections using three "P"s
  - Persistent (latency)
  - Parallel/concurrent (bandwidth and latency)
  - Pipelined over the same connection (latency)
- Caching
  - Forward proxy: close to clients
  - Reverse proxy: close to servers
- Replication

Replication

- Replicate popular Websites across many machines
  - Spreads load across servers
  - Places content closer to clients
  - Helps when content isn't cacheable

Content Distribution Networks (CDN)

- Caching and replication as a service
- Large-scale distributed storage infrastructure (usually) administered by one entity
  - e.g., Akamai is in 130 countries and 1700 networks
- Combination of caching and replication
  - Pull: Direct result of clients' requests (caching)
  - Push: Expectation of high access rate (replication)

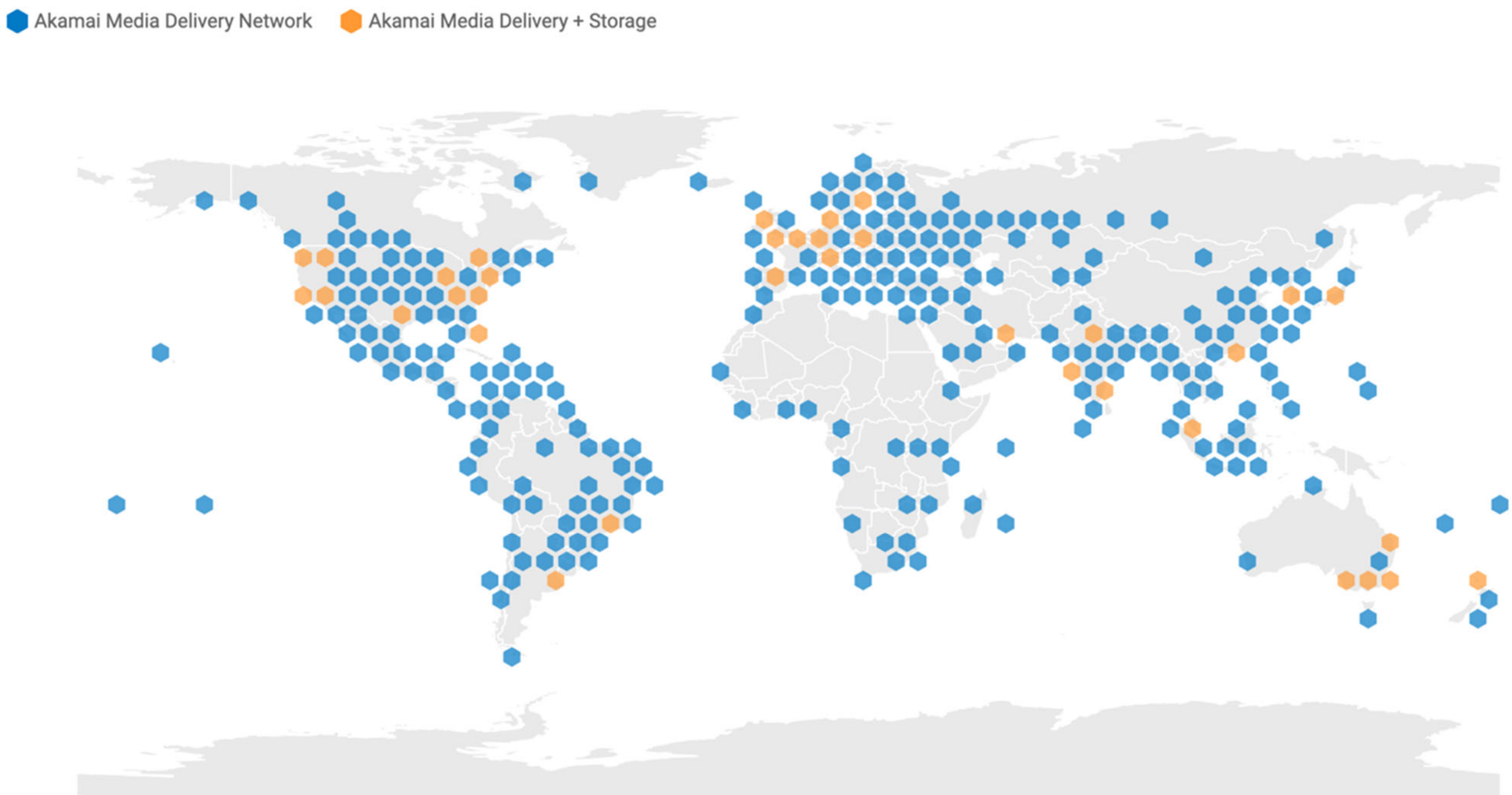
Cost-effective content delivery

- General theme: multiple sites hosted on shared physical infrastructure
  - Efficiency of statistical multiplexing
  - Economies of scale (volume pricing, etc.)
  - Amortization of human operator costs
- Examples:
  - CDNs
  - Web hosting companies
  - Cloud infrastructure

CDN example – Akamai

- Akamai creates new domain names for each client
  - e.g., a128.g.akamai.net for cnn.com
- The client content provider modifies content so that embedded URLs reference new domains
  - "Akamaize" content
  - e.g., http://www.cnn.com/image-of-the-day.gif becomes http://a128.g.akamai.net/image-of-the-day.gif
- Requests now sent to CDN's infrastructure

CDN example – Akamai



Retrieved on Sep 10, 2021 from https://www.akamai.com/visualizations/media-delivery-network-map

Why direct clients to particular replicas?

- Balancing load across server replicas
- Pairing clients with nearby servers to decrease latency and overall bandwidth usage

Packet Generation

DNS: Domain name system

Internet names & addresses

- Machine addresses: e.g., 141.212.113.143
  - Router-usable labels for machines
  - Conforms to network structure (the "where")
- Machine names: e.g., cse.umich.edu
  - Human-usable labels for machines
  - Conforms to organizational structure (the "who")
- The Domain Name System (DNS) is how we map from one to the other
  - A directory service

Why?

- Convenience
  - Easier to remember
- Provides a level of indirection!
  - Decoupled names from addresses
  - Many uses beyond just naming a specific host

DNS: History

- Initially all host-address mappings were in a `hosts.txt` file (in `/etc/hosts`):
  - Maintained by the Stanford Research Institute (SRI)
  - Changes were submitted by email and updates downloaded periodically from SRI
- As the Internet grew SRI could not handle load
  - Names were not unique anymore
  - Hosts had inaccurate copies of `hosts.txt`

DNS: History

- In 1983, the first stable operational DNS implementation included
  - The associated query protocol;
  - A server implementation; and
  - Initial root servers.

- Since inception, DNS scaled from 1000s of queries/day to 10s of billions queries/day

Goals

- Uniqueness: no naming conflicts
- Scalable
  - Many names and frequent updates (secondary)
- Distributed, autonomous administration
  - Ability to update my own (machines') names
  - Don't have to track everybody's updates
- Highly available
- Lookups are fast
- Perfect consistency is a non-goal

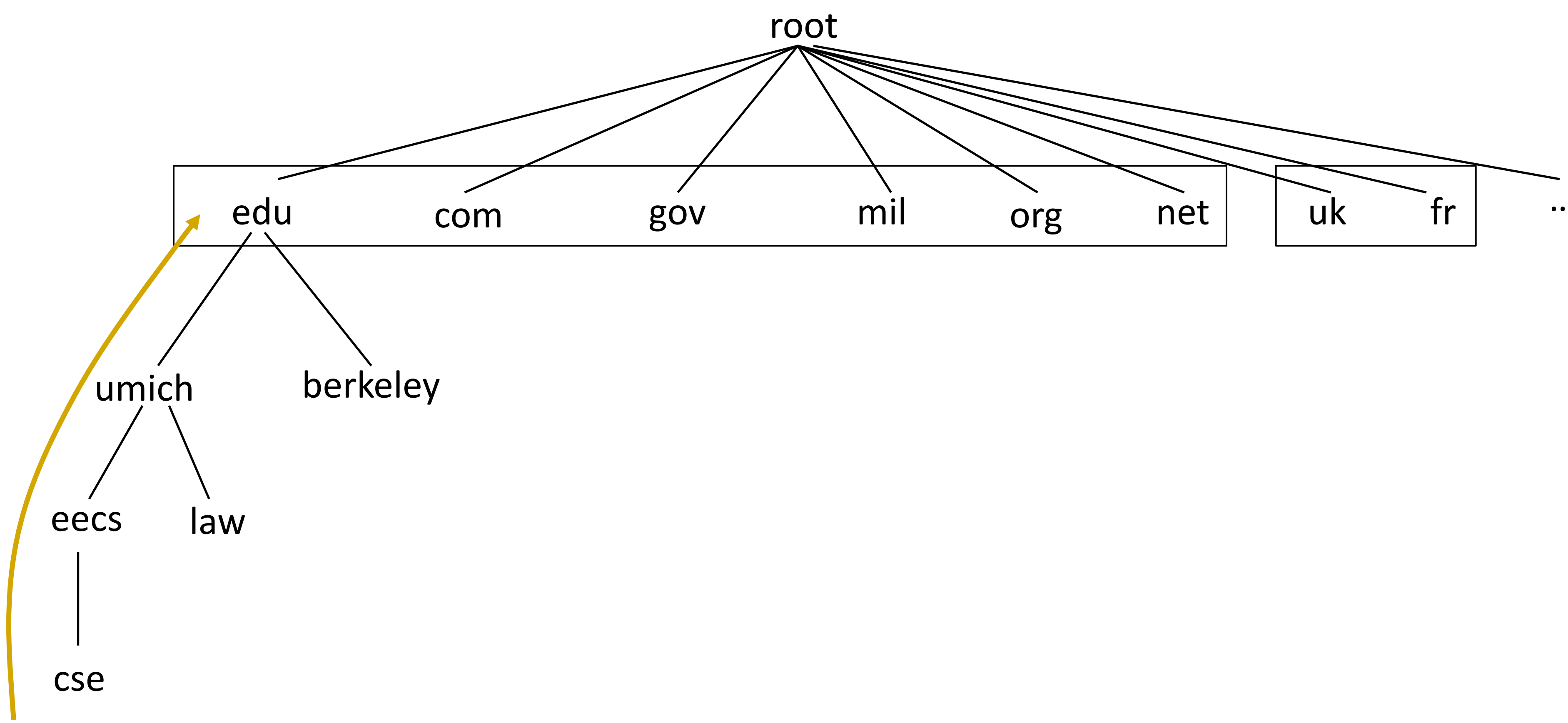
How?

- Partition the namespace
- Distribute administration of each partition
  - Autonomy to update my own (machines’) names
  - Don’t have to track everybody’s updates
- Distribute name resolution for each partition
- How should we partition things?

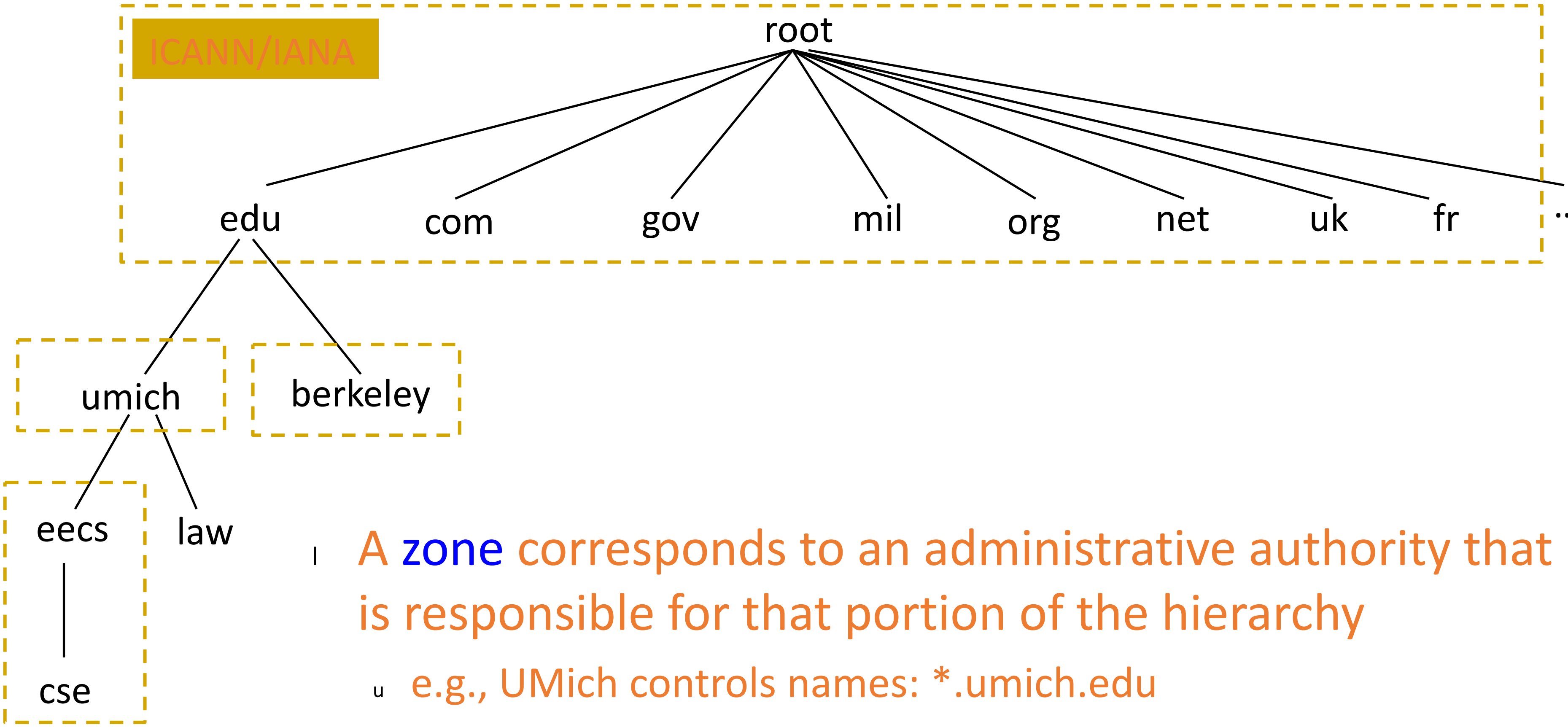
Key idea: Hierarchy

- Three intertwined hierarchies
  - Hierarchical namespace
    - As opposed to original flat namespace
  - Hierarchically administered
    - As opposed to centralized
  - (Distributed) hierarchy of servers
    - As opposed to centralized storage

Hierarchical namespace



Hierarchical administration



- A zone corresponds to an administrative authority that is responsible for that portion of the hierarchy
  - e.g., UMich controls names: *.umich.edu
  - e.g., EECS controls names: *.eecs.umich.edu

Server hierarchy

- Top of hierarchy: Root servers
  - Location hardwired into other servers
- Next Level: Top-level domain (TLD) servers
  - .com, .edu, etc.
  - Managed professionally
- Bottom Level: Authoritative DNS servers
  - Actually store the name-to-address mapping
  - Maintained by the corresponding administrative authority

Server hierarchy

- Each server stores a (small!) subset of the total DNS database
- An authoritative DNS server stores "resource records" for all DNS names in the domain that it has authority for
- Each server needs to know other servers responsible for other portions of the hierarchy
  - Every server knows the root
  - Root server knows about all top-level domains

DNS root

- Located in Virginia, USA
- How do we make the root scale?

13 DNS root servers

As of 2023-09-13T17:51:42Z, the root server system consists of 1751 instances operated by the 12 independent root server operators.

https://root-servers.org/

DNS records

- DNS servers store resource records (RRs)
  - RR is (name, value, type, TTL)
- Type = A: (→ Address)
  - name = hostname
  - value = IP address
- Type = NS: (→ Name Server)
  - name = domain
  - value = name of DNS server for domain

DNS records (cont'd)

- Type = CNAME: (→ Canonical Name)
  - name = alias name for some "canonical" (real) name
    - e.g., cse.umich.edu is really cse.eecs.umich.edu
  - value = canonical name
- Type = MX: (→ Mail eXchanger)
  - name = domain in email address
  - value = name(s) of mail server(s)

Inserting Resource Records into DNS

- Register foobar.com at registrar
  - Provide registrar with names and IP addresses of your authoritative name server(s)
  - Registrar inserts RR pairs into the .com TLD server:
    - (foobar.com, dns1.foobar.com, NS)
    - (dns1.foobar.com, 212.44.9.129, A)
- Store resource records in your server dns1.foobar.com
  - e.g., type A record for www.foobar.com
  - e.g., type MX record for foobar.com

Using DNS (Client/App View)

- Two components
  - Local DNS servers
  - Resolver software on hosts
- Local DNS server ("default name server")
  - Clients configured with default server's address OR learn it via a host configuration protocol (e.g., DHCP)

```
Wireless LAN adapter Wi-Fi:

   Connection-specific DNS Suffix  . : adsroot.itcs.umich.edu
   Description . . . . . . . . . . . : Intel(R) Wi-Fi 6E AX211 160MHz
   Physical Address. . . . . . . . . : 00-A5-54-3B-A7-B3
   DHCP Enabled. . . . . . . . . . . : Yes
   Autoconfiguration Enabled . . . . : Yes
   Link-local IPv6 Address . . . . . : fe80::b07e:59d8:3fed:5cb1%24(Preferred)
   IPv4 Address. . . . . . . . . . . : [redacted]referred)
   Subnet Mask . . . . . . . . . . . : 255.255.0.0
   Lease Obtained. . . . . . . . . . : Wednesday, September 11, 2024 9:56:36 AM
   Lease Expires . . . . . . . . . . : Wednesday, September 11, 2024 1:29:06 PM
   Default Gateway . . . . . . . . . : 35.3.0.1
   DHCP Server . . . . . . . . . . . : 141.211.147.232
   DHCPv6 IAID . . . . . . . . . . . : 167814484
   DHCPv6 Client DUID. . . . . . . . : 00-01-00-01-2B-34-E2-F8-00-0C-29-15-E0-FE
   DNS Servers . . . . . . . . . . . : 10.10.10.10
                                       10.10.5.5
   NetBIOS over Tcpip. . . . . . . . : Enabled
```

Using DNS (Client/App View)

- Two components
  - Local DNS servers
  - Resolver software on hostsadd
- Client application
  - Obtain DNS name (e.g., from URL)
  - Do `getnameinfo()` to trigger DNS request to its local DNS server

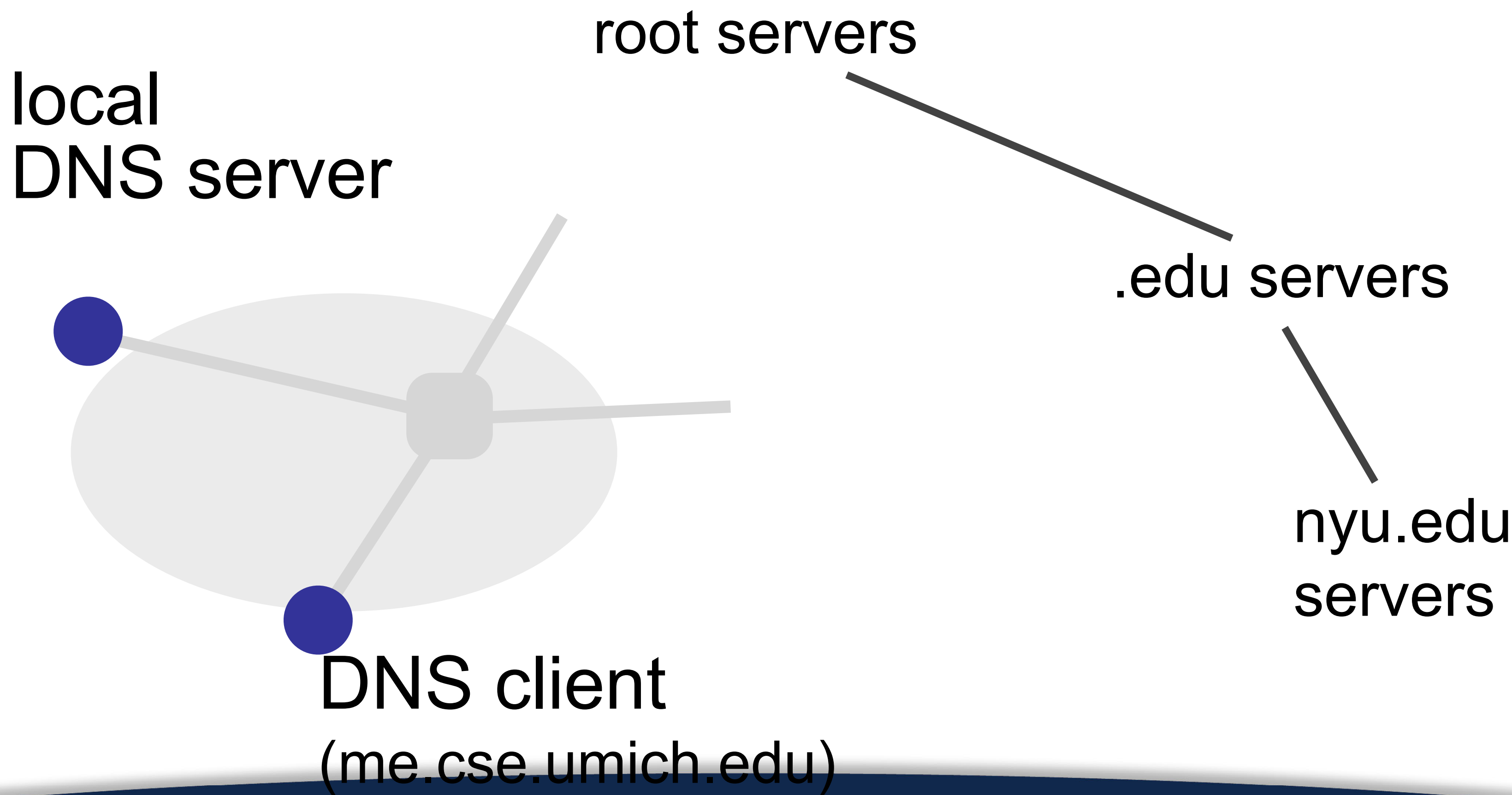
```
import socket
s = socket.getaddrinfo("www.google.com",80)
print(s)
```

```
[(<AddressFamily.AF_INET: 2>, 0, 0, '', ('142.250.190.132', 80))]
```

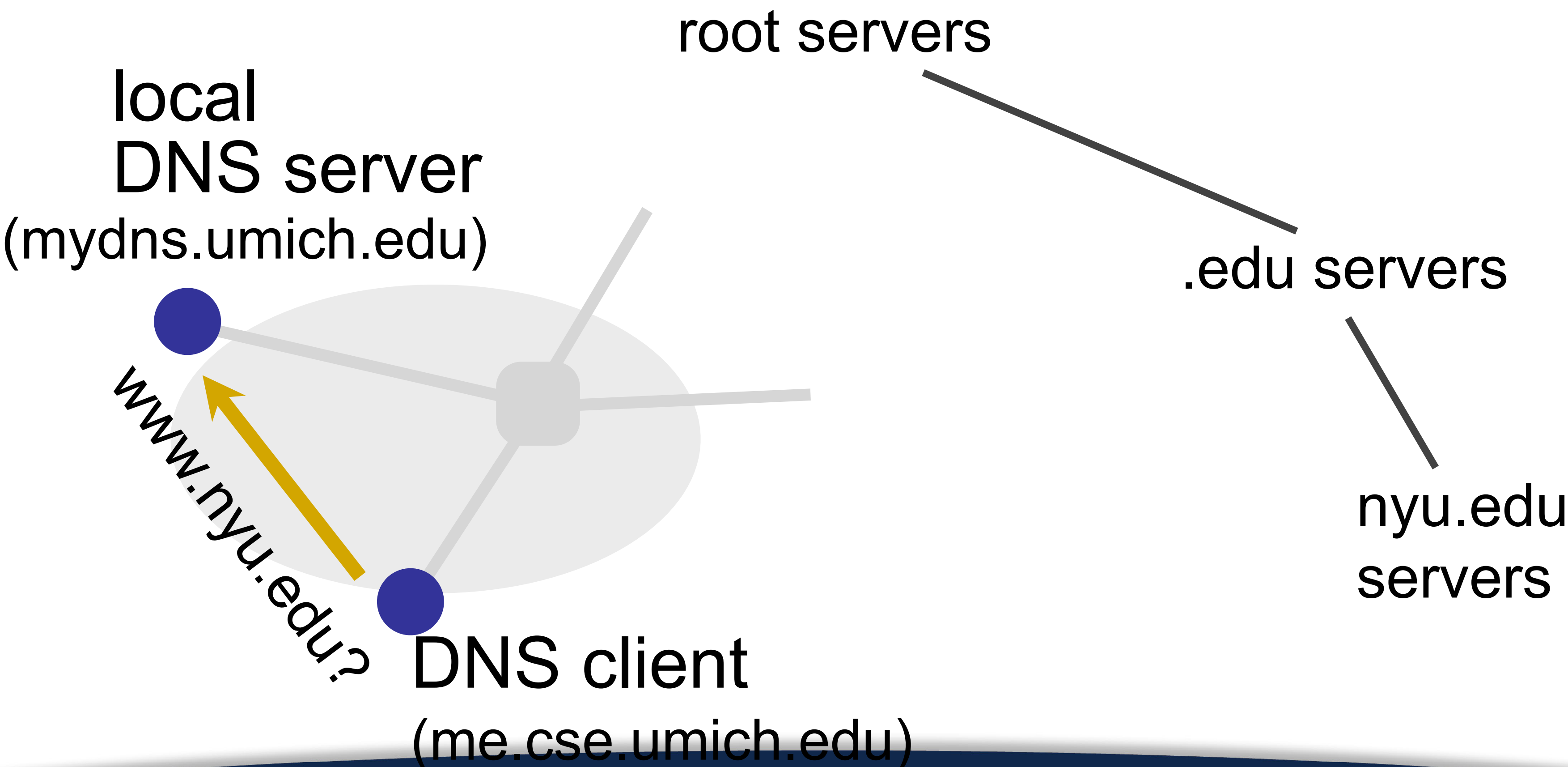
Dig (domain information groper)

```
dig www.google.com

; <<>> DiG 9.18.1-1ubuntu1.2-Ubuntu <<>> www.google.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 26427
;; flags: qr rd ad; QUERY: 1, ANSWER: 9, AUTHORITY: 0, ADDITIONAL: 0
;; WARNING: recursion requested but not available

;; QUESTION SECTION:
;www.google.com.                        IN      A

;; ANSWER SECTION:
www.google.com.         300     IN      A       142.250.190.132
ns3.google.com.         0       IN      A       216.239.36.10
ns2.google.com.         0       IN      A       216.239.34.10
ns4.google.com.         0       IN      A       216.239.38.10
ns1.google.com.         0       IN      A       216.239.32.10

;; Query time: 9 msec
;; SERVER: 172.20.64.1#53(172.20.64.1) (UDP)
;; WHEN: Wed Sep 11 13:22:13 EDT 2024
;; MSG SIZE  rcvd: 350
```

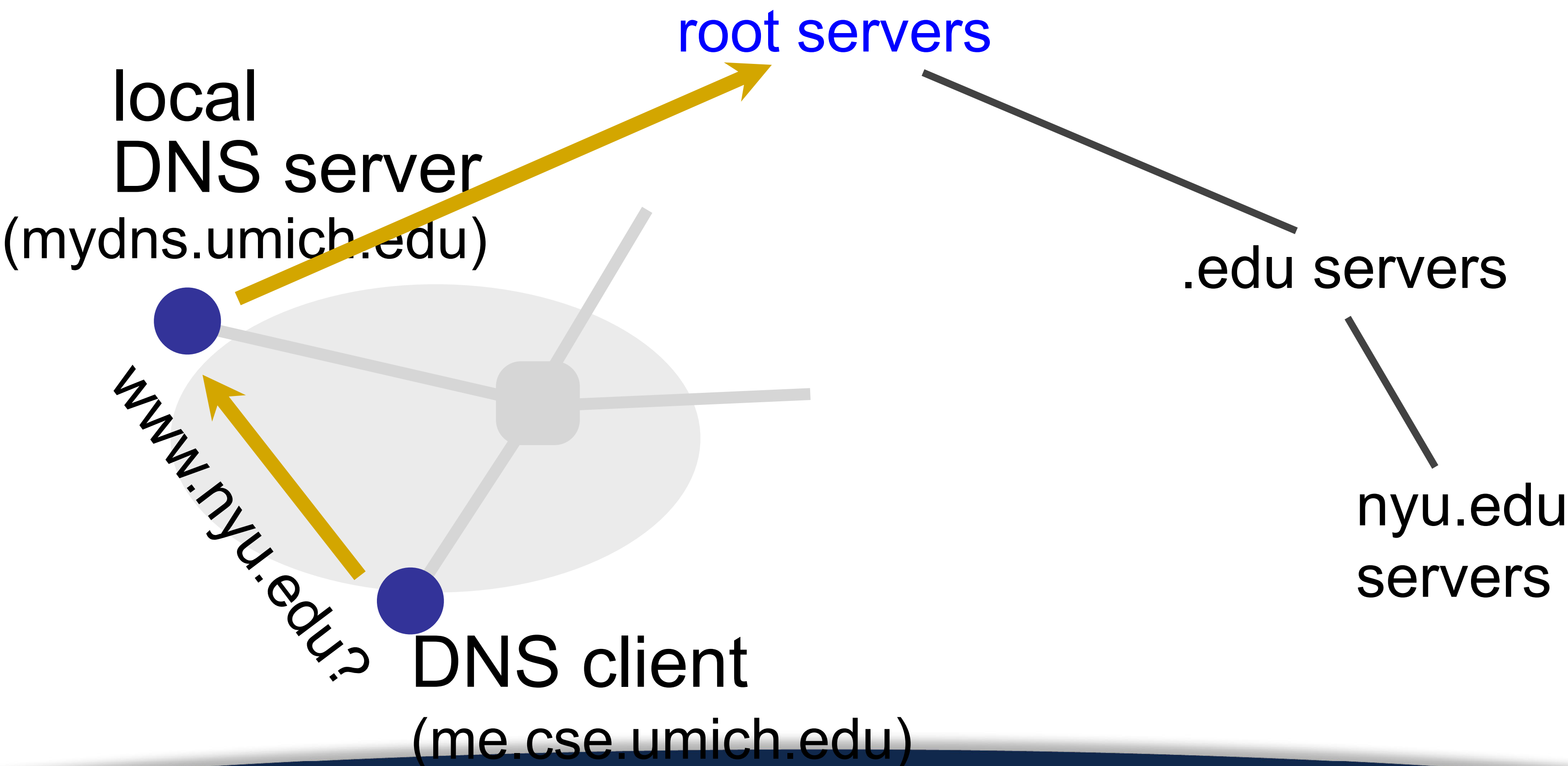
Name resolution



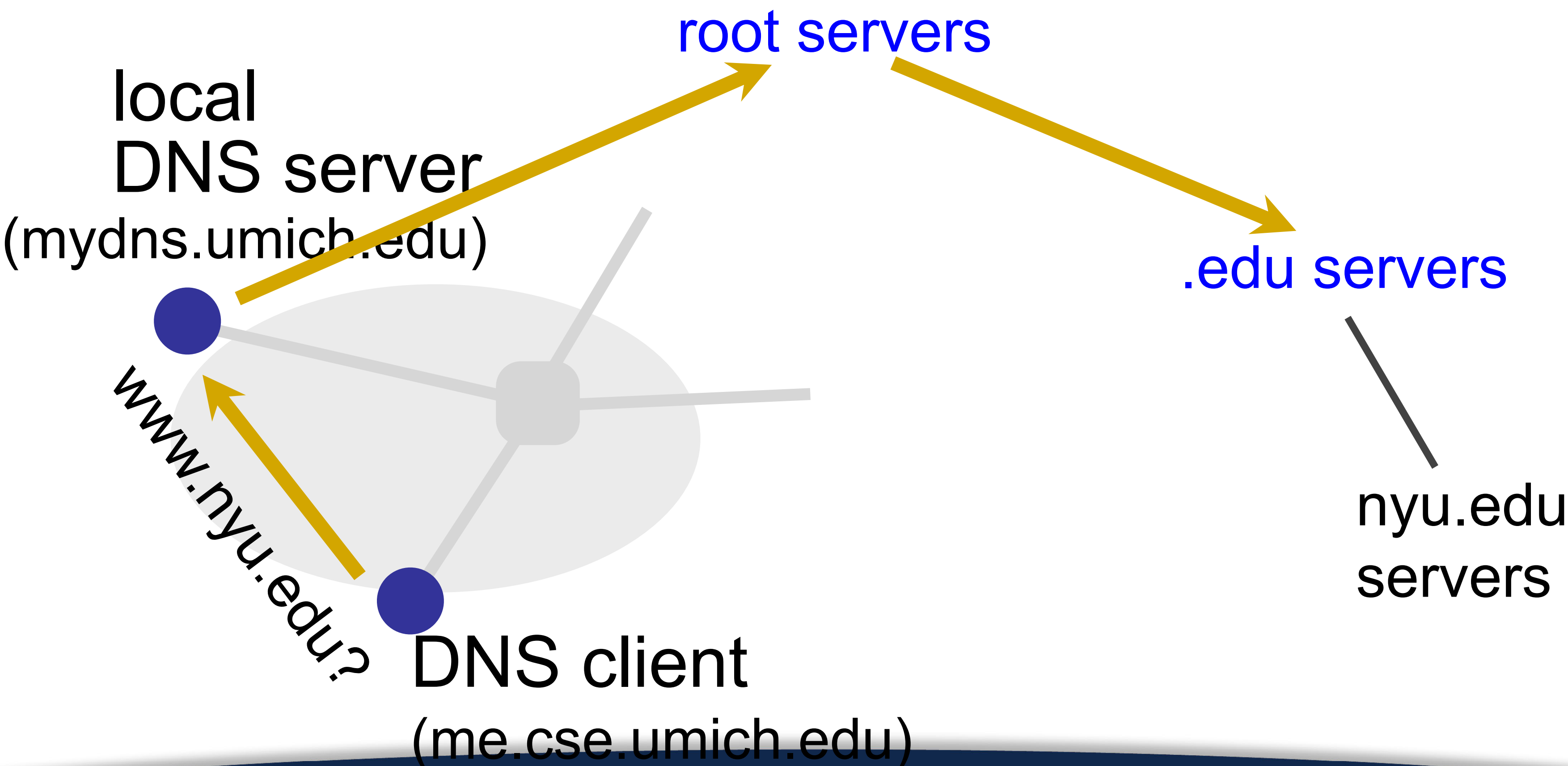
Name resolution



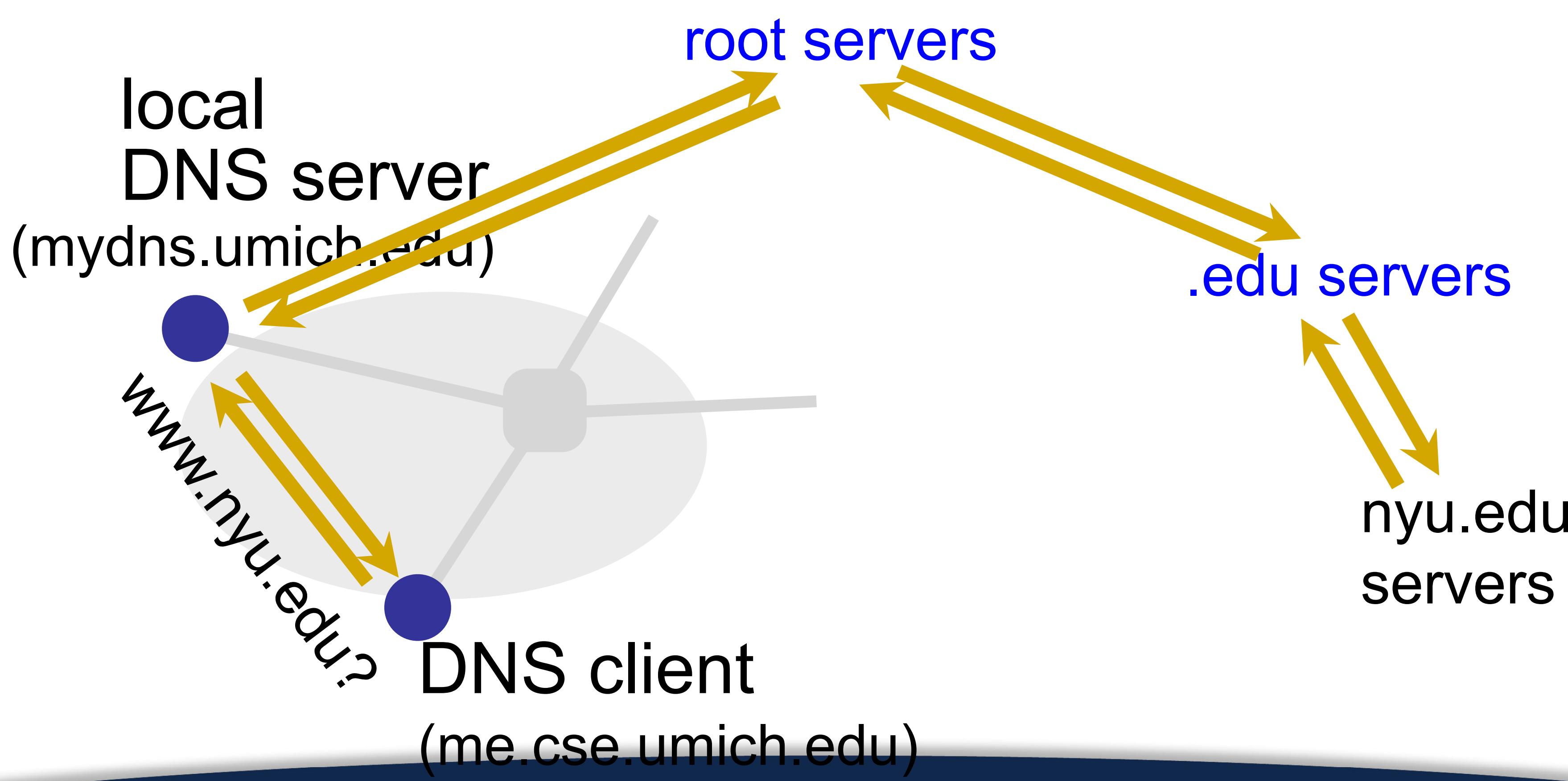
Name resolution



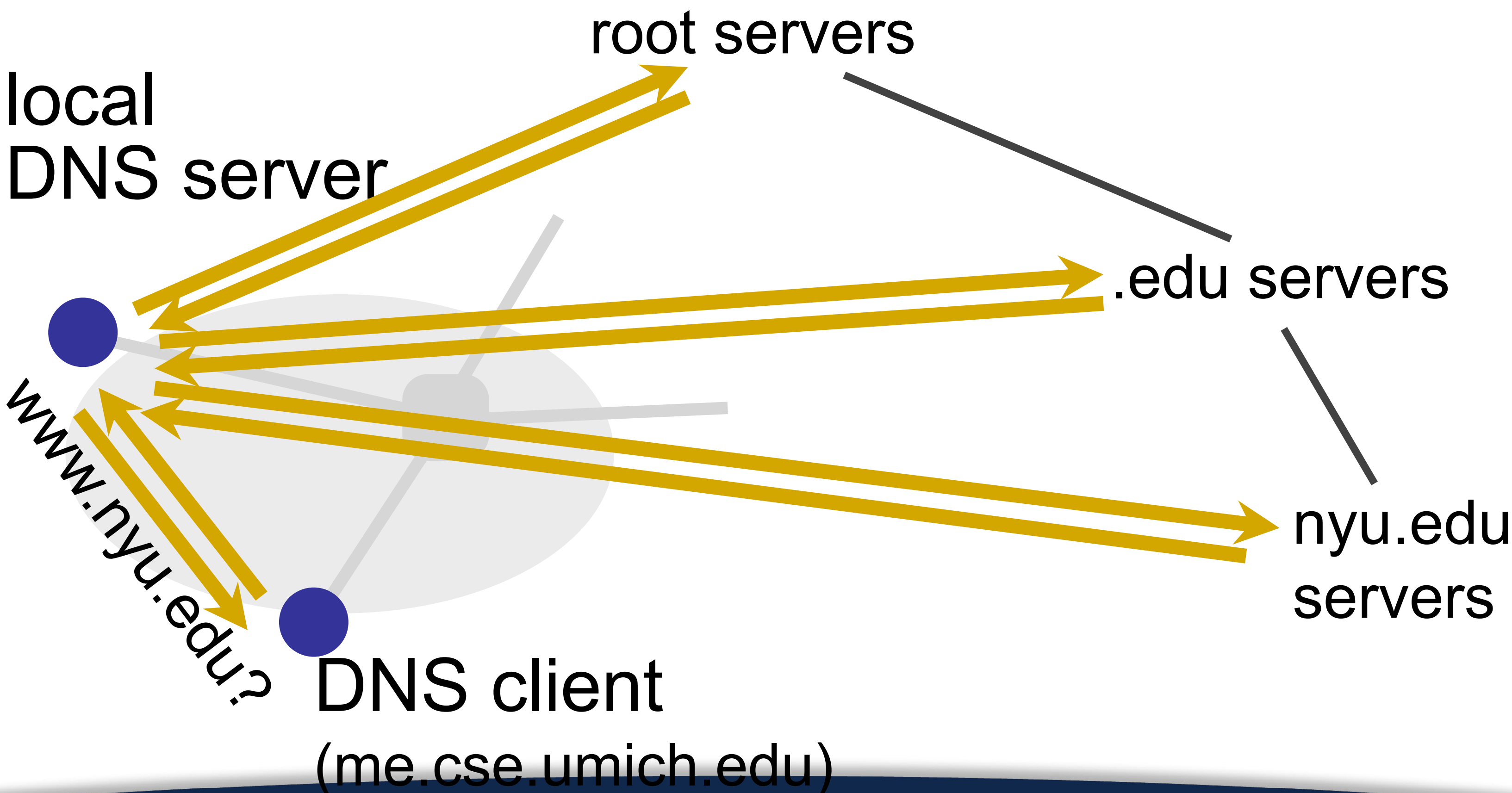
Name resolution



Name resolution: Recursive



Name resolution: Iterative



Two ways to resolve a name

- Recursive name resolution
  - Ask server to do it for you
- Iterative name resolution
  - Ask server who to ask next
- The iterative example we saw is a mix of both!

DNS protocol

- Query and Reply messages; both with the same message format
  - Header: identifier, flags, etc.
  - Plus resource records
- Client–server interaction on UDP Port 53
  - Spec supports TCP too, but not always implemented

Goals: Are we there yet?

- Uniqueness: No naming conflicts
- Scalable
- Distributed, autonomous administration
- Highly available?

Reliability

- Replicated DNS servers (primary/secondary)
  - Name service available if at least one replica is up
  - Queries can be load-balanced between replicas
- Usually, UDP used for queries
  - Reliability, if needed, must be implemented on UDP
- Try alternate servers on timeout
  - Exponential backoff when retrying same server
- Same identifier for all queries
  - Don’t care which server responds

Goals: Are we there yet?

- Uniqueness: No naming conflicts
- Scalable
- Distributed, autonomous administration
- Highly available
- Fast lookups?

DNS caching

- Performing all these queries takes time
  - Up to 1-second latency before starting download
- Caching can greatly reduce overhead
  - The top-level servers very rarely change
  - Popular sites (e.g., www.google.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

TTL in dig output

```
dig www.google.com

; <<>> DiG 9.18.1-1ubuntu1.2-Ubuntu <<>> www.google.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 26427
;; flags: qr rd ad; QUERY: 1, ANSWER: 9, AUTHORITY: 0, ADDITIONAL: 0
;; WARNING: recursion requested but not available

;; QUESTION SECTION:
;www.google.com.                        IN      A
```

TTL

```
;; ANSWER SECTION:
www.google.com.         300         IN      A       142.250.190.132
ns3.google.com.         0           IN      A       216.239.36.10
ns2.google.com.         0           IN      A       216.239.34.10
ns4.google.com.         0           IN      A       216.239.38.10
ns1.google.com.         0           IN      A       216.239.32.10

;; Query time: 9 msec
;; SERVER: 172.20.64.1#53(172.20.64.1) (UDP)
;; WHEN: Wed Sep 11 13:22:13 EDT 2024
;; MSG SIZE  rcvd: 350
```

Negative caching

- Remember things that do not work
  - Misspellings like www.google.comm
  - These can take a long time to fail the first time
  - Good to remember that they do not work so the failure takes less time the next time around
- Negative caching is optional
  - Not widely implemented

Important properties of DNS

- Administrative delegation and hierarchy enables:
  - Easy unique naming
  - “Fate sharing” for network failures
  - Reasonable trust model
  - Caching increases scalability and performance

DNS provides indirection

- Addresses can change underneath
  - Move www.cnn.com to 4.125.91.21
- Name could map to multiple IP addresses
  - Load-balancing (CDN)
  - Reducing latency by picking nearby servers (CDN)
  - Try out: `dig google.com` a few times
- Multiple names for the same address
  - E.g., many services (mail, www) on same machine
  - E.g., aliases like www.cnn.com and cnn.com

Summary

- CDNs improve web performance
  - Via replication and caching
  - Good server selection
- DNS allows us to go to webpages without having to memorize IP addresses
  - Allows a level of indirection that enables many functionalities including CDN server selection

Quiz 4

- https://forms.gle/2rjqsa3SoGWdbdL58