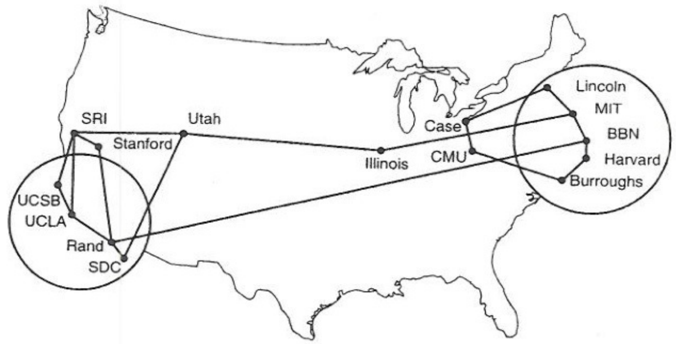


Domain Name System (DNS)

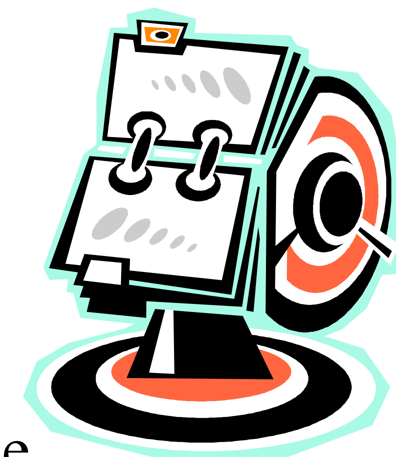
Reading: Section in Chapter 9

RFC 1034, STD 13

Name Syntax and rules for delegating authority over names
Specify implementation of a distributed system that maps names to addresses
Protocols to accomplish the above

Goals of Today's Lecture

- Computer science concepts underlying DNS
  - Indirection: names in place of addresses
  - Hierarchy: in names, addresses, and servers
  - Caching: of mappings from names to/from addresses
- Inner-workings of DNS
  - DNS resolvers and servers
  - Iterative and recursive queries
  - TTL-based caching
- Web and DNS
  - Influence of DNS queries on Web performance
  - Server selection and load balancing



Names: Overview

- What do names do?
  - identify objects
  - help locate objects
  - define membership in a group
  - specify a role
  - convey knowledge of a secret
- Name space
  - defines set of possible names
  - consists of a set of name to value *bindings*

Host Names vs. IP addresses

- Host names
  - Mnemonic name appreciated by humans
  - Variable length, alpha-numeric characters
  - Provide little (if any) information about location
  - Examples: www.cnn.com and ftp.eurocom.fr
- IP addresses
  - Numerical address appreciated by routers
  - Fixed length, binary number
  - Hierarchical, related to host location (network)
  - Examples: 64.236.16.20 and 193.30.227.161

Separating Naming and Addressing

- Names are easier to remember
  - www.cnn.com vs. 64.236.16.20
- Addresses can change underneath
  - Move www.cnn.com to 64.236.16.20
  - E.g., renumbering when changing providers
- Name could map to multiple IP addresses
  - www.cnn.com to multiple replicas of the Web site
- Map to different addresses in different places
  - Address of a nearby copy of the Web site
  - E.g., to reduce latency, or return different content
- Multiple names for the same address
  - E.g., aliases like ee.mit.edu and cs.mit.edu

History: Global Namespace gathered in Local File

- Original name to address mapping
  - Flat namespace
  - /etc/hosts
  - SRI kept main copy
  - Downloaded regularly
- Count of hosts was increasing: moving from a machine per domain to machine per user
  - Many more downloads
  - Many more updates

Global Namespace : Central Server

- Central server
  - One place where all mappings are stored
  - All queries go to the central server
- Many practical problems
  - Single point of failure
  - High traffic volume
  - Distant centralized database
  - Single point of update
  - Does not scale

Need a distributed, hierarchical collection of servers

Global Namespace: Domain Name System (DNS)

- Properties of DNS
  - Hierarchical name space divided into zones
  - Distributed over a collection of DNS servers
- Hierarchy of DNS servers
  - Root servers
  - Top-level domain (TLD) servers
  - Authoritative DNS servers
  - Local DNS server
- Performing the translations
  - Local DNS servers
  - Resolver software

Domain Name System (DNS)

- Elements
  - Resolver
    - Stub: simple, only asks questions
    - Recursive: takes a simple query and makes all necessary steps to get the full answer.
    - Caching: A recursive resolver that stores prior results and reuses them
  - Server
    - Authoritative: the servers that contain the zone file for a zone, one Primary and one or more Secondaries
    - Caching: A recursive resolver that stores prior results and reuses them
  - Some perform both roles at the same time.

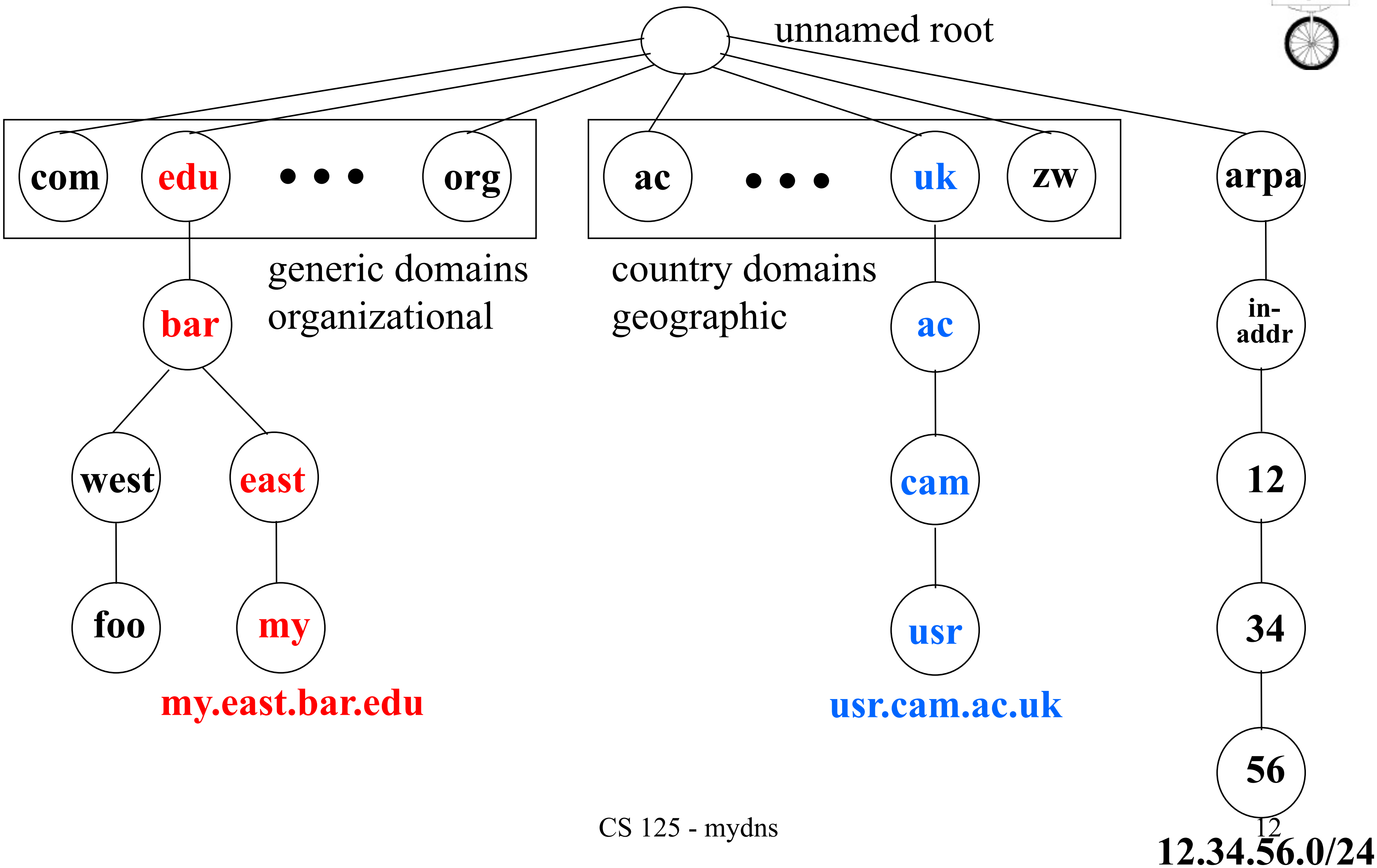
DNS Root Servers

- 13 root servers (see http://www.root-servers.org)
- Labeled A through M

TLD and Authoritative DNS Servers

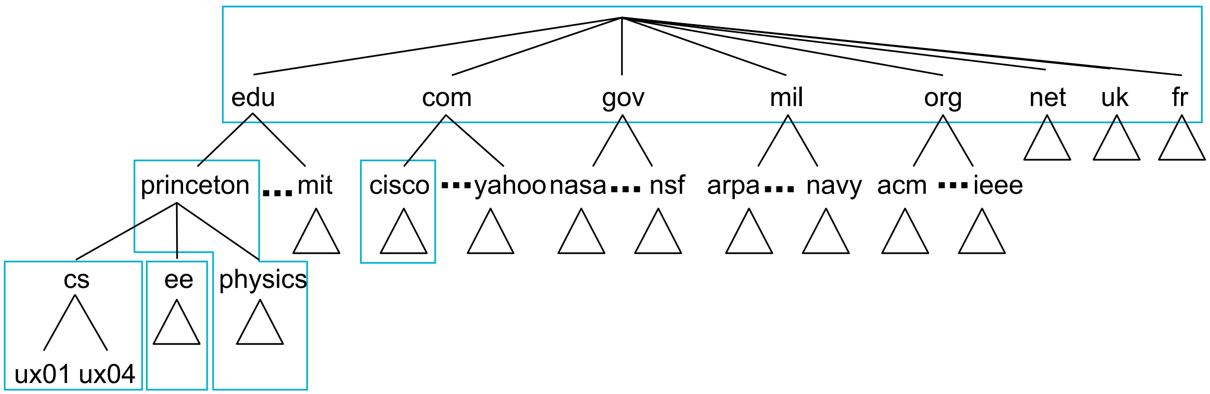
- Top-level domain (TLD) servers
  - Generic domains (e.g., com, org, edu + new ones)
  - Country domains (e.g., uk, fr, ca, jp)
  - Typically managed professionally
    - Network Solutions maintains servers for "com"
    - Educause maintains servers for "edu"
- Authoritative DNS servers
  - Provide public records for hosts at an organization
  - For the organization's servers (e.g., Web and mail)
  - Can be maintained locally or by a service provider

Distributed Hierarchical Database

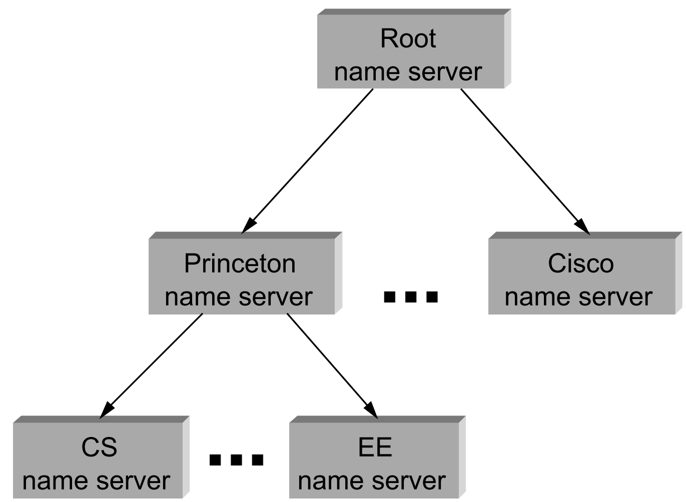


Name Servers

- Partition hierarchy into *zones*



- Each zone implemented by two or more *name servers*

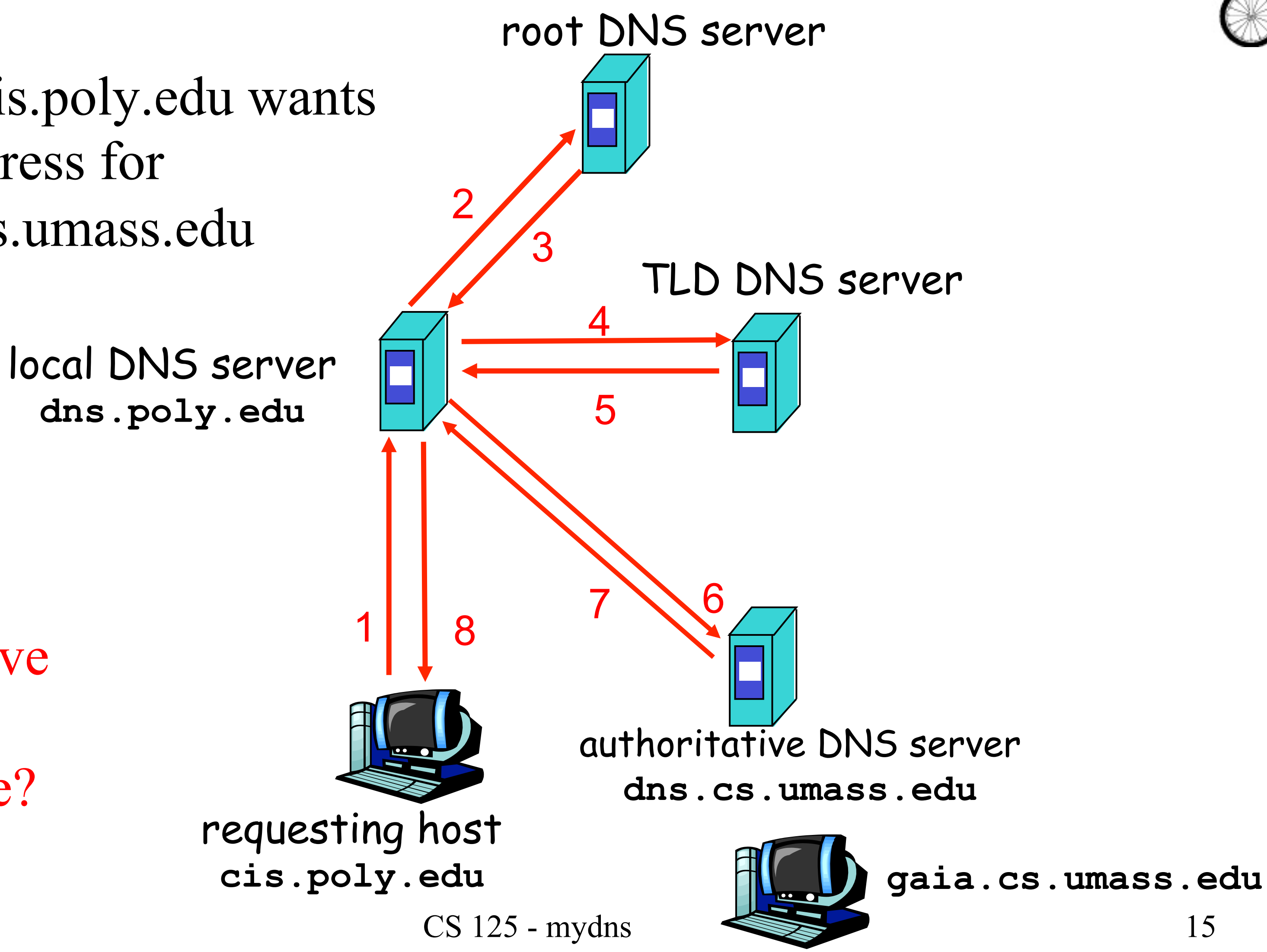


Using DNS

- Local DNS server ("default name server")
  - Usually near the end hosts who use it
  - Local hosts configured with local server (e.g., /etc/resolv.conf) or learn the server via DHCP
- Client application
  - Extract server name (e.g., from the URL)
  - Do *gethostbyname()* to trigger resolver code
- Server application
  - Extract client IP address from socket
  - Optional *gethostbyaddr()* to translate into name

Example

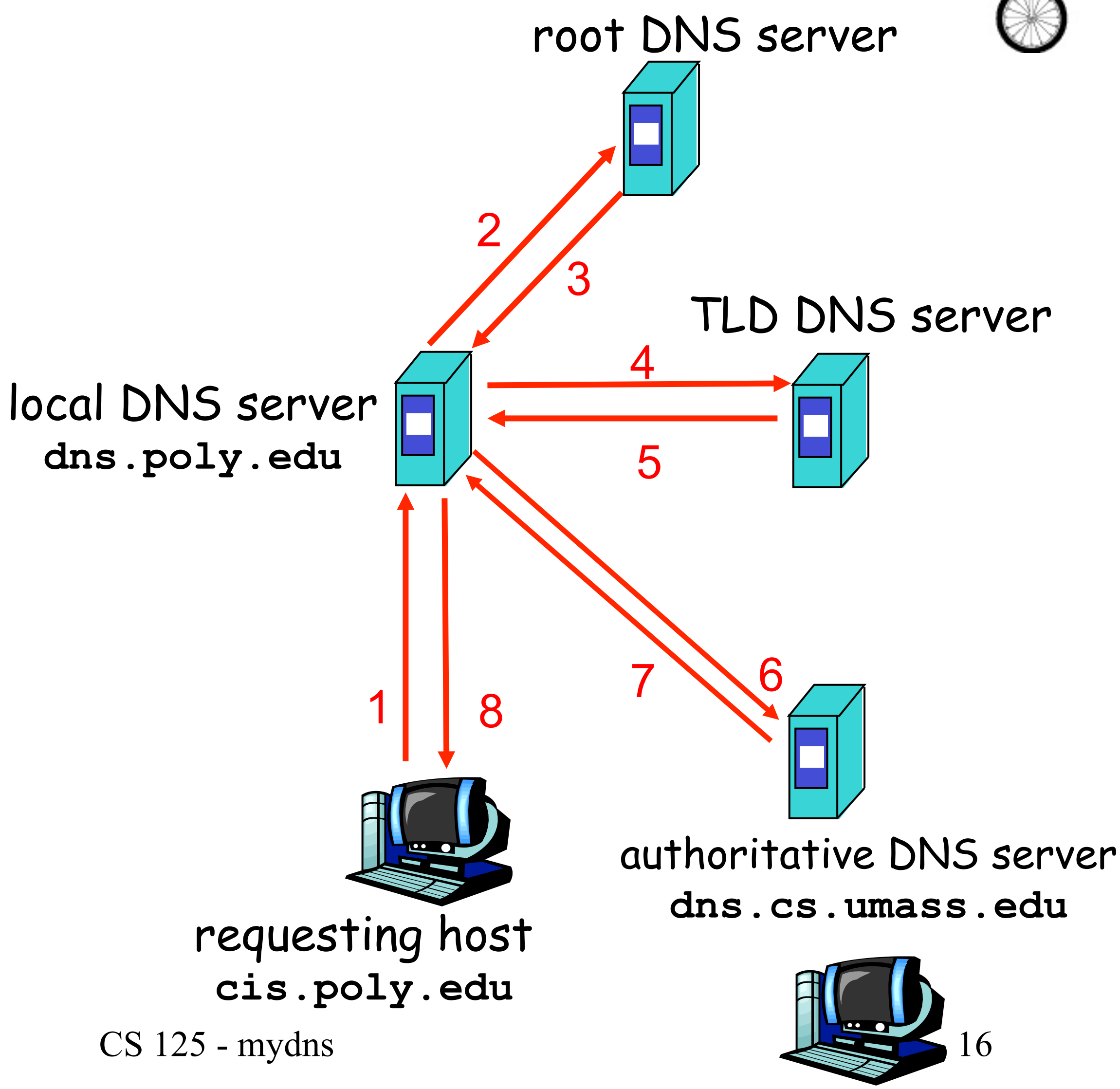
Host at cis.poly.edu wants IP address for gaia.cs.umass.edu



Recursive
Or
Iterative?

Recursive vs. Iterative Queries

- Recursive query
  - Ask server to get answer for you
  - E.g., request 1 and response 8
- Iterative query
  - Ask server who to ask next
  - E.g., all other request-response pairs



DNS Caching

- Performing all these queries takes time
  - And all this before the actual communication takes place
  - E.g., 1-second latency before starting Web download
- Caching can substantially 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 the cached entry after TTL expires

Negative Caching

- Remember things that don't work
  - Misspellings likewww.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

DNS Resource Records

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

DNS: distributed db storing resource records (RR)

- Type=A
  - **`name`** is hostname
  - **`value`** is IP address
- Type=NS
  - **`name`** is domain (e.g. foo.com)
  - **`value`** is hostname of authoritative name server for this domain
- Type=CNAME
  - **`name`** is alias name for some "canonical" (the real) name

    `www.ibm.com` is really

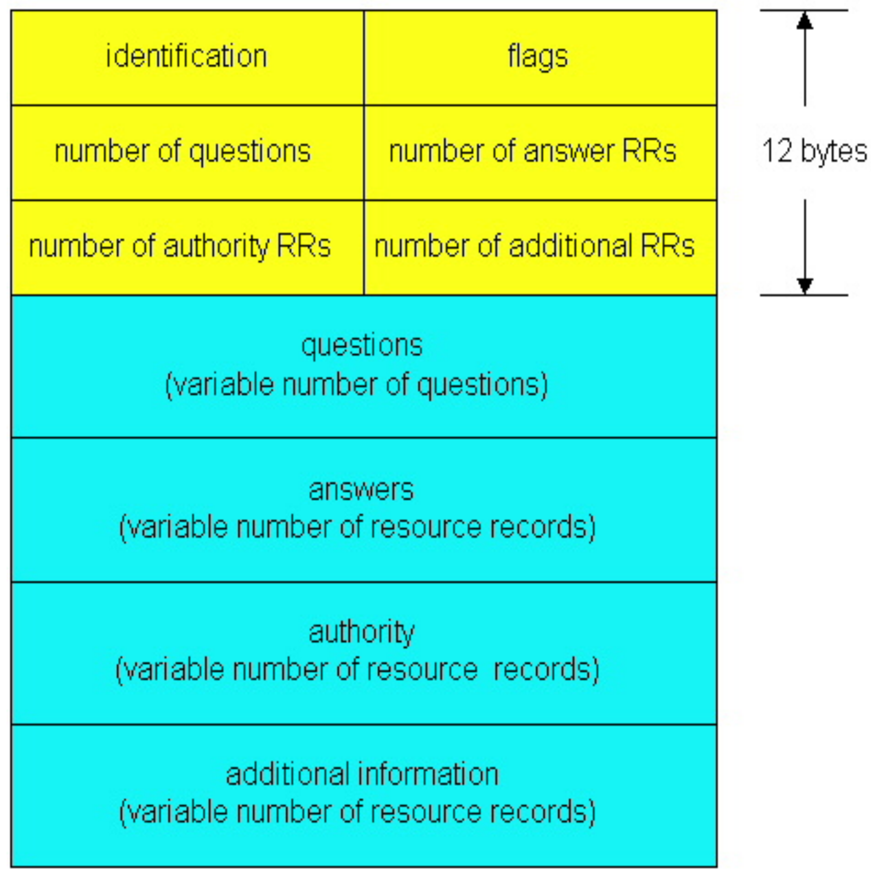
    `servereast.backup2.ibm.com`
  - **`value`** is canonical name
- Type=MX
  - **`value`** is name of mailserver associated with **`name`**

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

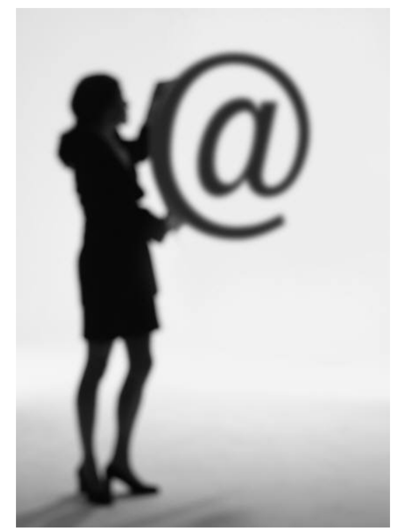


Reliability

- DNS servers are replicated
  - Name service available if at least one replica is up
  - Queries can be load balanced between replicas
- UDP used for queries
  - Need reliability: must implement this on top of UDP
- Try alternate servers on timeout
  - Exponential back off when retrying same server
- Same identifier for all queries
  - Don't care which server responds

Inserting Resource Records into DNS

- Example: just created startup "FooBar"
- Register foobar.com at Network Solutions
  - Provide registrar with names and IP addresses of your authoritative name server (primary and secondary)
  - Registrar inserts two RRs into the com TLD server:
    - (foobar.com, dns1.foobar.com, NS)
    - (dns1.foobar.com, 212.212.212.1, A)
- Put in authoritative server dns1.foobar.com
  - Type A record for www.foobar.com
  - Type MX record for foobar.com



Playing With Dig on UNIX

- Dig program
  - Allows querying of DNS system
  - Use flags to find name server (NS)
  - Disable recursion so that operates one step at a time

```
unix> dig +norecurse @a.root-servers.net NS www.cs.princeton.edu

;; AUTHORITY SECTION:
edu.                2D IN NS        L3.NSTLD.COM.
edu.                2D IN NS        D3.NSTLD.COM.
edu.                2D IN NS        A3.NSTLD.COM.
edu.                2D IN NS        E3.NSTLD.COM.
edu.                2D IN NS        C3.NSTLD.COM.
edu.                2D IN NS        G3.NSTLD.COM.
edu.                2D IN NS        M3.NSTLD.COM.
edu.                2D IN NS        H3.NSTLD.COM.
```

DNS and the Web

DNS Query in Web Download

- User types or clicks on a URL
  - E.g., http://www.cnn.com/2006/leadstory.html
- Browser extracts the site name
  - E.g., www.cnn.com
- Browser calls gethostbyname() to learn IP address
  - Triggers resolver code to query the local DNS server
- Eventually, the resolver gets a reply
  - Resolver returns the IP address to the browser
- Then, the browser contacts the Web server
  - Creates and connects socket, and sends HTTP request

Multiple DNS Queries

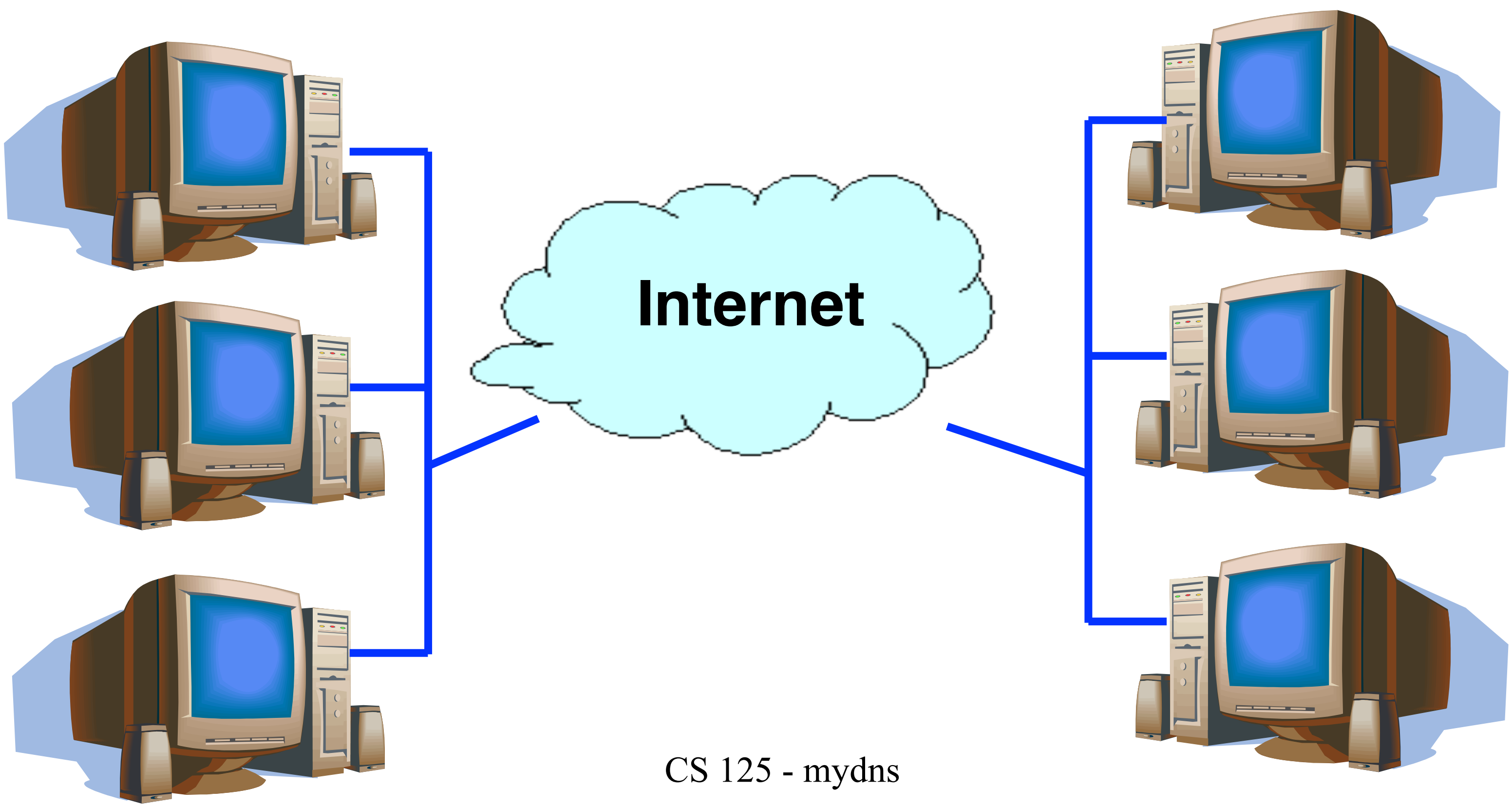
- Often a Web page has embedded objects
  - E.g., HTML file with embedded images
- Each embedded object has its own URL
  - … and potentially lives on a different Web server
  - E.g., http://www.myimages.com/image1.jpg
- Browser downloads embedded objects
  - Usually done automatically, unless configured otherwise
  - Requires learning the address for www.myimages.com

When are DNS Queries Unnecessary?

- Browser is configured to use a proxy
  - E.g., browser sends all HTTP requests through a proxy
  - Then, the proxy takes care of issuing the DNS request
- Requested Web resource is locally cached
  - E.g., cache has http://www.cnn.com/2006/leadstory.html
  - No need to fetch the resource, so no need to query
- Browser recently queried for this host name
  - E.g., user recently visited http://www.cnn.com/
  - So, the browser already called *gethostbyname()*
  - … and may be locally caching the resulting IP address

Web Server Replicas

- Popular Web sites can be easily overloaded
  - Web site often runs on multiple server machines



Directing Web Clients to Replicas

- Simple approach: different names
  - www1.cnn.com, www2.cnn.com, www3.cnn.com
  - But, this requires users to select specific replicas
- More elegant approach: different IP addresses
  - Single name (e.g., www.cnn.com), multiple addresses
  - E.g., 64.236.16.20, 64.236.16.52, 64.236.16.84, …
- Authoritative DNS server returns many addresses
  - And the local DNS server selects one address
  - Authoritative server may (should) vary the order of addresses

Clever Load Balancing Schemes

- Selecting the "best" IP address to return
  - Based on server performance
  - Based on geographic proximity
  - Based on network load
  - …
- Example policies
  - Round-robin scheduling to balance server load
  - U.S. queries get one address, Europe another
  - Tracking the current load on each of the replicas

Challenge: What About DNS Caching?

- Problem: DNS caching
  - What if performance properties change?
  - Web clients still learning old "best" Web server
  - … until the cached information expires
- Solution: Small Time-to-Live values
  - Setting artificially small TTL values
  - … so replicas picked based on fresh information
- Disadvantages: abuse of DNS?
  - Many more DNS request/response messages
  - Longer latency in initiating the Web requests

DNSSEC

- RFC 3833, Summary of DNS Weakness
- Role: Protect DNS
  - DNS Rrset is signed by the zone it belongs to
  - Zone DS Rrset is vouched for by parent zone.
  - DNSSEC is intended to protect DNS clients from forged DNS data
- What DNSSEC does not do:
  - Make data in DNS any more current….

Conclusions

- Domain Name System
  - Distributed, hierarchical database
  - Distributed collection of servers
  - Caching to improve performance
- Readings
  - DNS Related RFCs > 100
  - DNSSEC - 4033, 4034, 4035
  - Original - 1034, 1035