



# Domain Name Service



# The role of names and name services

- Resources are accessed using *identifier* or *reference*
  - An identifier can be stored in variables and retrieved from tables quickly
  - Identifier includes or can be transformed to an address for an object
    - E.g. NFS file handle, Corba remote object reference
  - A *name* is human-readable value (usually a string) that can be *resolved* to an identifier or address
    - Internet domain name, file pathname, process number
    - E.g. /etc/passwd, <http://www.cdk3.net/>
- For many purposes, names are preferable to identifiers
  - because the binding of the named resource to a physical location is deferred and can be changed
  - because they are more meaningful to users
- Resource names are *resolved* by *name services*
  - to give identifiers and other useful attributes



# Names and resources

Currently, different name systems are used for each type of resource:

| <b><i>resource</i></b> | <b><i>name</i></b> | <b><i>identifies</i></b>        |
|------------------------|--------------------|---------------------------------|
| file                   | pathname           | file within a given file system |
| process                | process id         | process on a given computer     |
| port                   | port number        | IP port on a given computer     |

Uniform Resource Identifiers (URI) offer a general solution for any type of resource. There two main classes:

URL

Uniform Resource Locator

- typed by the protocol field (http, ftp, nfs, etc.)
- part of the name is service-specific
- resources cannot be moved between domains

URN

Uniform Resource Name

- requires a universal resource name lookup service - a DNS-like system for all resources



# Name services

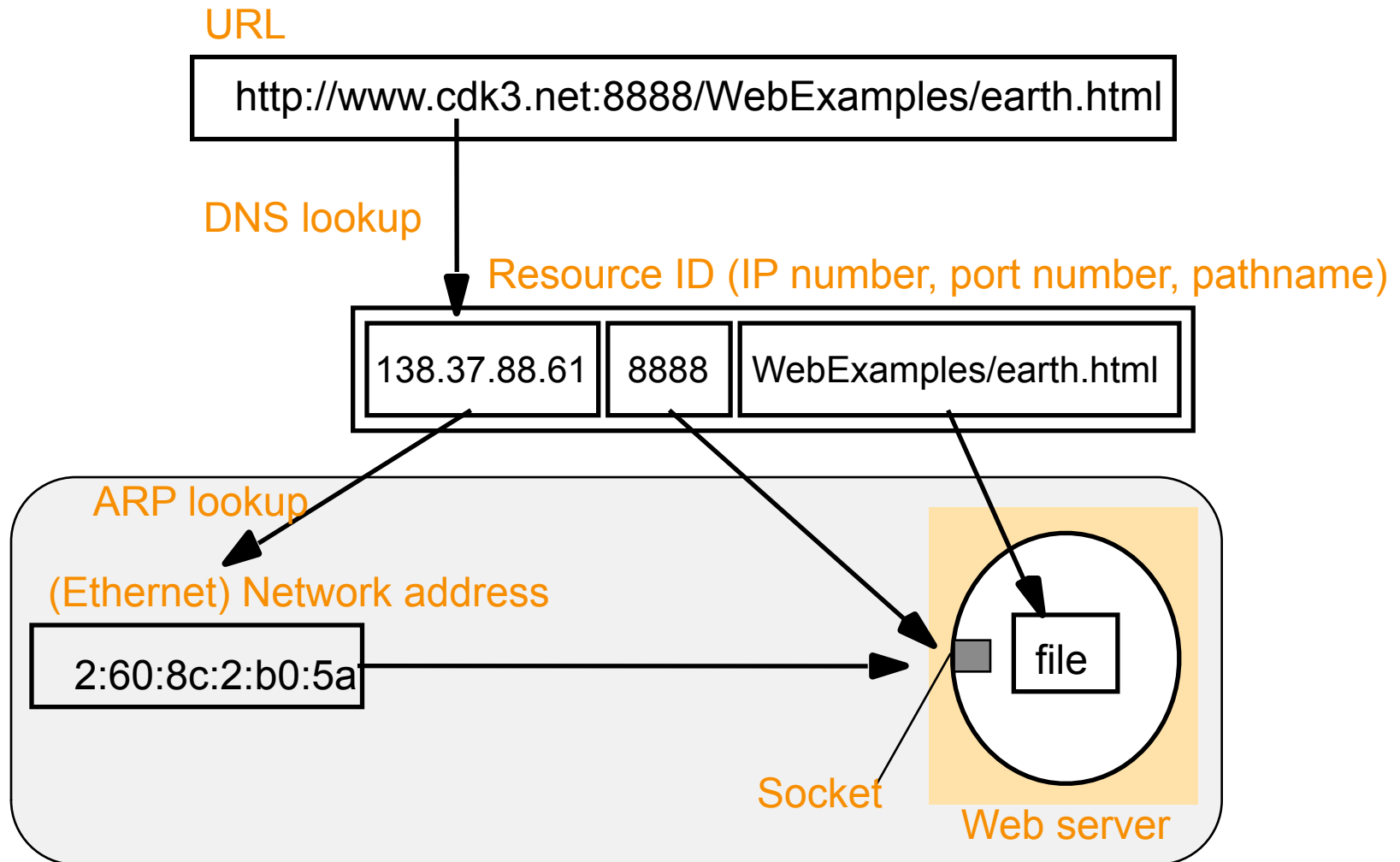
- Αποθηκεύουν συλλογή ενός ή περισσότερων *naming contexts* – σύνολα από προσδέσεις (*bindings*) ανάμεσα σε ονόματα και κατηγορήματα αντικειμένων-object attributes (χρήστες, Η/Υ, υπηρεσίες, απομακρυσμένα αντικείμενα).
- Βασική λειτουργία τους – η *επίλυση (resolution)* ενός ονόματος, δηλαδή η εύρεση των κατηγορημάτων του.
- Απαιτήσεις από υπηρεσίες ονομάτων:
  - Υποστήριξη μη πεπερασμένου αριθμού ονομάτων για πολλές διαχειριστικές αρχές.
  - Μεγάλος χρόνος ζωής.
  - Υψηλή διαθεσιμότητα (*high availability*) – η λειτουργία πολλών συστημάτων βασίζεται σε *name services*.
  - Απομόνωση βλαβών – ώστε τοπικές βλάβες να μην επηρεάζουν ολόκληρη την υπηρεσία.
  - Ανοχή στην καχυποψία (*tolerance of mistrust*) – σε ένα ανοικτό σύστημα δεν είναι δυνατόν όπως όλα τα στοιχεία του είναι έμπιστα μεταξύ τους.



# Name spaces – χώροι ονομάτων

- Χώρος ονομάτων – όλα τα έγκυρα ονόματα που αναγνωρίζει μια συγκεκριμένη υπηρεσία.
- Τα ονόματα πρέπει να:
  - Έχουν εσωτερική δομή, η οποία αντιπροσωπεύει τη θέση τους σε ένα ιεραρχικό χώρο ονομάτων, ώστε:
    - Να επιτρέπεται η χρήση παρόμοιων υπο-ονομάτων χωρίς συγκρούσεις.
    - Να είναι δυνατή η ομαδοποίηση αλληλοσυσχετιζόμενων ονομάτων.
  - Διευκολύνουν την διαχείριση της εμπιστοσύνης.
  - Διευκολύνουν την αναδόμηση δένδρων ονομάτων.
- Πεδία ονομάτων (naming domains): χώροι ονομάτων υπό μια κοινή διαχειριστική αρχή. Η αρχή αυτή έχει απόλυτο έλεγχο για το ποιά ονόματα θα προσδεθούν (bind) σε ποιούς πόρους του πεδίου.

# Composed naming domains used to access a resource from a URL



# Επίλυση Ονομάτων (name resolution)

- Επαναληπτική διαδικασία κατά την οποία ένα όνομα παρουσιάζεται επαναληπτικά σε διαδοχικά πλαίσια ονομάτων (naming contexts).
- Καθένα από αυτά τα πλαίσια είτε επιλύει (αντιστοιχίζει) το όνομα σε ένα σύνολο από πρωτογενή κατηγορήματα (primitive attributes) είτε απεικονίζει μια μετασχηματισμένη μορφή του σε ένα άλλο πλαίσιο ονομάτων για περαιτέρω επίλυση.
- Στην περίπτωση χρήσης ψευδωνύμων (aliases), πρώτα γίνεται μετασχηματισμός του ψευδωνύμου στο κανονικό όνομα και μετά επίλυση (πρόβλημα κύκλων).
- Το άτοπο του κεντρικού εξυπηρετητή ονομάτων για το Διαδίκτυο (DNS) – bottleneck, critical point of failure, scalability. Προσέγγιση:
  - Αναπαραγωγή (replication)
  - Κατανομή (partitioning)
- *Πλοήγηση (navigation)*: η διαδικασία εξεύρεσης δεδομένων ονοματολογίας από έναν ή περισσότερους εξυπηρετητές ονομάτων.



# Πλοήγηση σε Ονοματοχώρους

- Επαναληπτική πλοήγηση (iterative navigation):
  - Το σύστημα πελάτη παρουσιάζει το προς επίλυση όνομα σε τοπικό εξυπηρετητή ονομάτων.
  - Αν γίνει επίλυση τοπικά, επιστρέφεται στον πελάτη το κατάλληλο κατηγορήμα.
  - Διαφορετικά, επιστροφή άλλου εξυπηρετητή ονομάτων.
  - Υιοθέτηση στο *DNS* και *NFS*.
- Πλοήγηση multicasting:
  - Εκπομπή τού προς επίλυση ονόματος από τον πελάτη σε διάφορους εξυπηρετητές.
  - Μόνο ο εξυπηρετητής που διαθέτει το όνομα, επιστρέφει την απάντηση στον πελάτη.
  - Προβλήματα: τι θα συμβεί αν κανείς από αυτούς δεν διαθέτει την απάντηση;





# 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
- 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



# Strawman Solution: 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



## Strawman Solution #2: 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**



# 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
- Performing the translations
  - Local DNS servers
  - Resolver software



# DNS - The Internet Domain Name System

- A distributed naming database
- Name structure reflects administrative structure of the Internet
- Rapidly resolves domain names to IP addresses
  - exploits caching heavily
  - typical query time ~100 milliseconds
- Scales to millions of computers
  - partitioned database
  - caching
- Resilient to failure of a server
  - replication

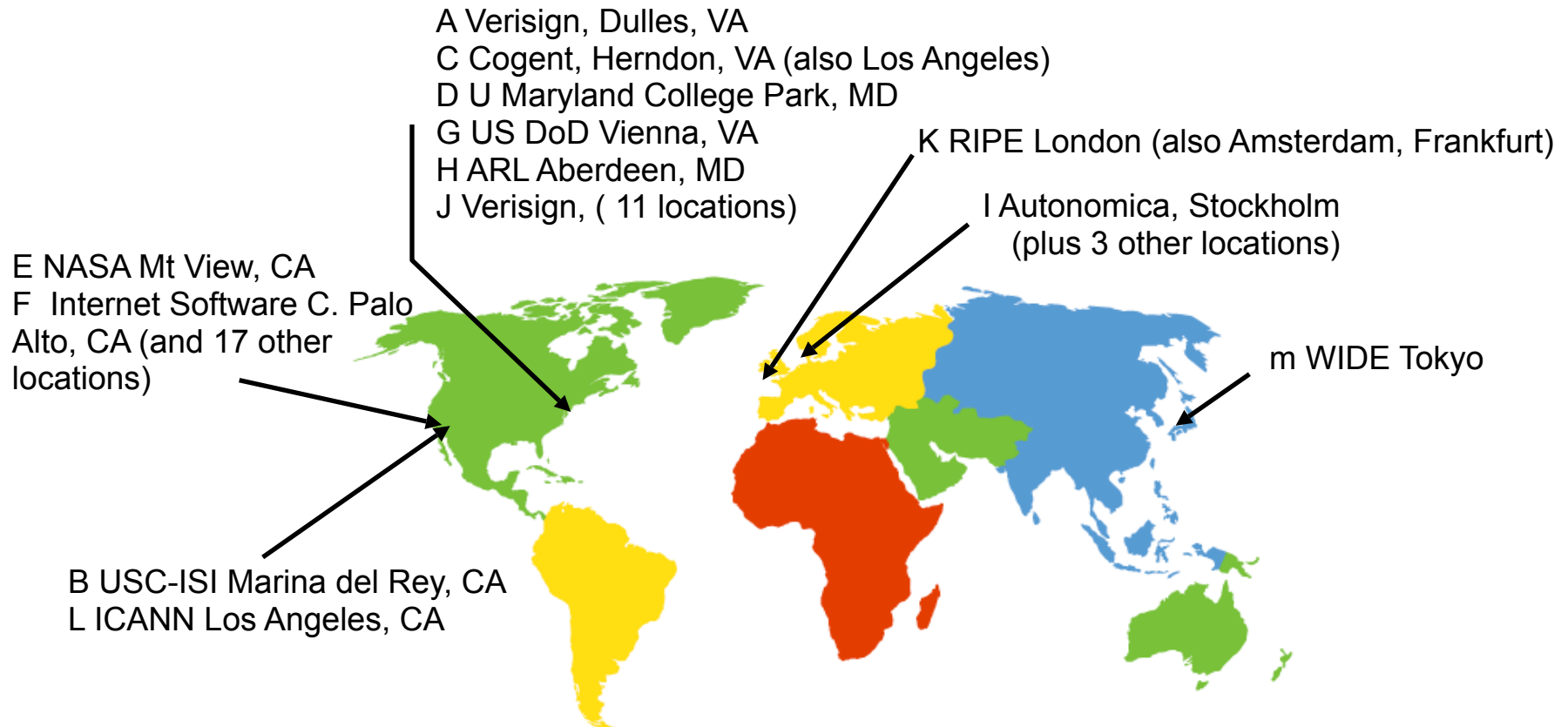
# Αρχιτεκτονική DNS

- Ζώνες (zones): μη επικαλυπτόμενα τμήματα ενός χώρου ονομάτων, το καθένα εκ των οποίων υποστηρίζεται από διαφορετικό εξυπηρετητή
  - υποδένδρα της ιεραρχίας του DNS, τα οποία ανήκουν σε διαφορετική διοικητική αρχή.
  - Κάθε ζώνη έχει συνήθως έναν πρωταρχικό εξυπηρετητή και περισσότερους δευτερεύοντες εξυπηρετητές, οι οποίοι μπορούν να υποκαταστήσουν τον πρωταρχικό σε περίπτωση βλάβης.
  - Μεγάλοι οργανισμοί μπορούν να οργανώνουν τα πεδία τους σε περισσότερες της μιας ζώνες.
- Η αποδοτική απεικόνιση διευθύνσεων IP σε ονόματα κόμβων (hostnames) προϋποθέτει μια διαφορετική ιεραρχία, βασιζόμενη σε διευθύνσεις IP.
  - Η ανάθεση των διευθύνσεων IP υποστηρίζεται από τρία αρχεία απογραφής (registries): APNIC (Ασία), ARIN (Β. Αμερική), RIPE NCC (Ευρώπη), LACNIC (Latin America & Caribbean), AfriNIC.
  - Η ανάθεση διευθύνσεων IP στα τρία αρχεία απογραφής γίνεται από τον οργανισμό ICANN.
  - Με την δέσμευση ενός συνόλου διευθύνσεων IP από έναν οργανισμό, ο οργανισμός αυτός γίνεται υπεύθυνος για ένα τμήμα του ονοματοχώρου *in-addr.arpa*. Πρόκειται για μια ιεραρχία που βασίζεται στις οκτάδες των 32-μπιτ των διευθύνσεων IP.



# DNS Root Servers

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





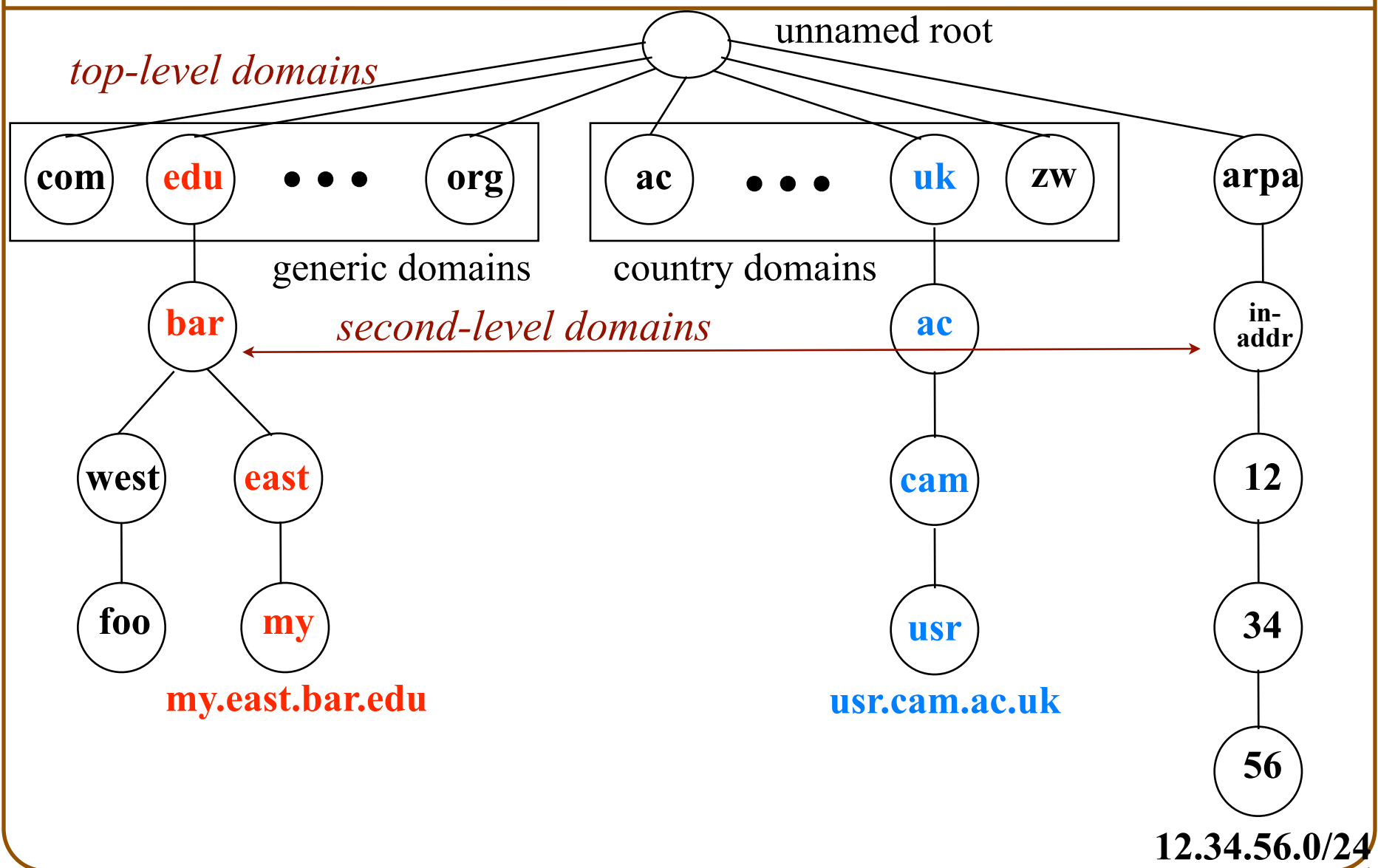


# TLD and Authoritative DNS Servers

- Top-level domain (TLD) servers
  - Generic domains (e.g., com, org, edu)
  - 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





# Reverse Lookup

- A reverse lookup is a query of the DNS for domain names when the IP address is known.
  - The DNS stores IP addresses in the form of domain names as a specially formatted names in pointer (PTR) records within the infrastructure top-level domain arpa.
  - For IPv4, the domain is in-addr.arpa. For IPv6, the reverse lookup domain is ip6.arpa.
  - The IP address is represented as a name in reverse-ordered octet representation for IPv4, and reverse-ordered nibble representation for IPv6.
- When performing a reverse lookup, the DNS client converts the address into these formats, and then queries the name for a PTR record following the delegation chain as for any DNS query.
  - For example, the IPv4 address 208.80.152.2 is represented as a DNS name as 2.152.80.208.in-addr.arpa. The DNS resolver begins by querying the root servers, which point to ARIN's servers for the 208.in-addr.arpa zone.



# 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

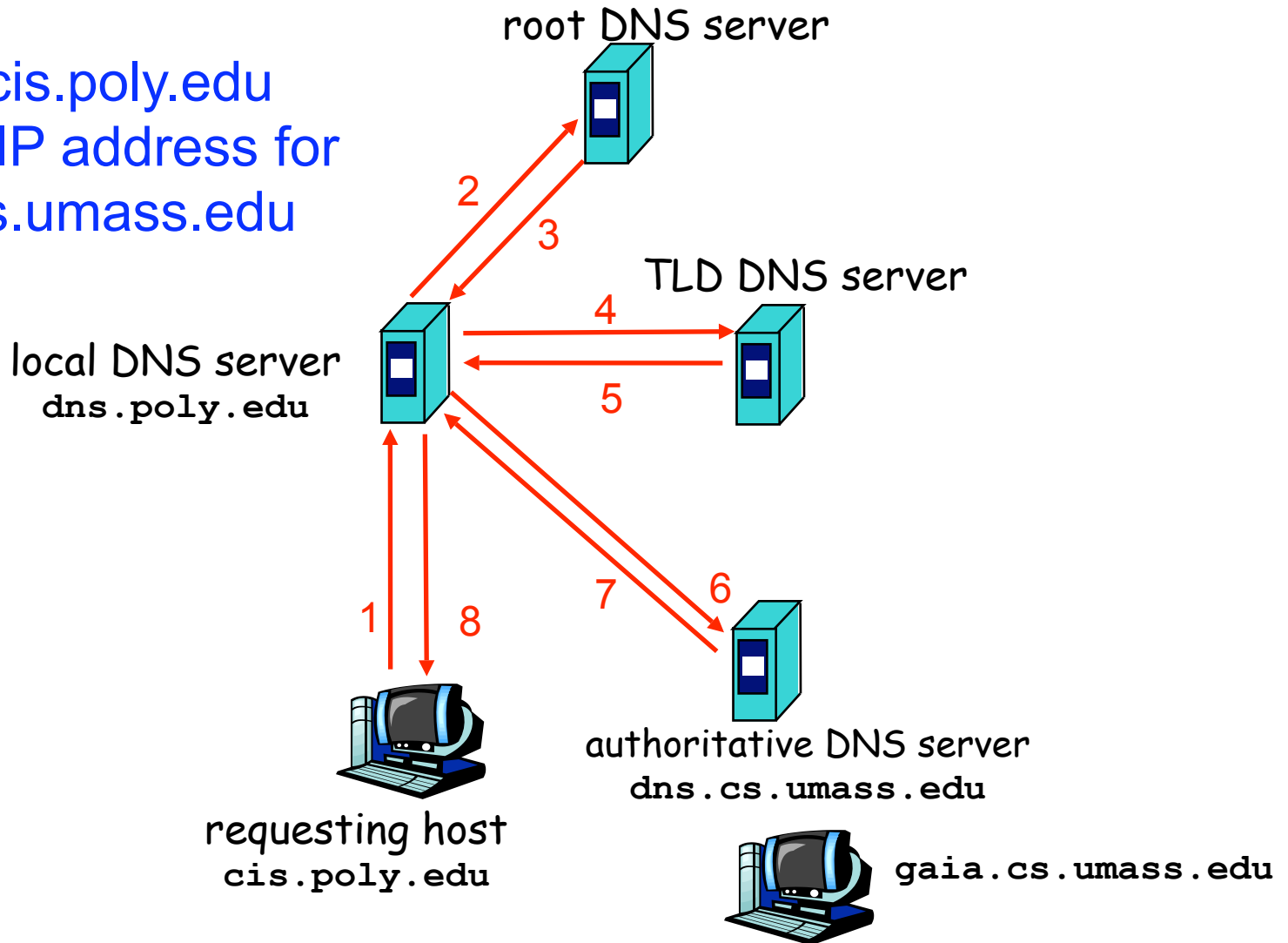
# DNS Resolver (επιλυτής DNS)

- Η μετάφραση ονομάτων κόμβων σε διευθύνσεις IP και αντιστρόφως συντονίζεται από το Domain Name System (DNS).
- Εφαρμογές του Διαδικτύου, όπως οι πλοηγοί (browsers), επικοινωνούν με το DNS μέσω ενός αναλυτή (*resolver*), δηλ. μιας βιβλιοθήκης η οποία έχει συνδεθεί (linked) με την εφαρμογή.
- Λειτουργίες του αναλυτή:
  - *gethostbyname()* : *hostname* → *IP address*
  - *gethostbyaddr()* : *IP address* → *hostname*
- Τυπικά, κάθε αναλυτής πρέπει να γνωρίζει τουλάχιστον έναν εξυπηρετητή DNS στον οποίο να απευθύνει τα ερωτήματά του (συνήθως: λίστα εξυπηρετητών, με τους κοντινότερους πρώτους στη λίστα).



# Example

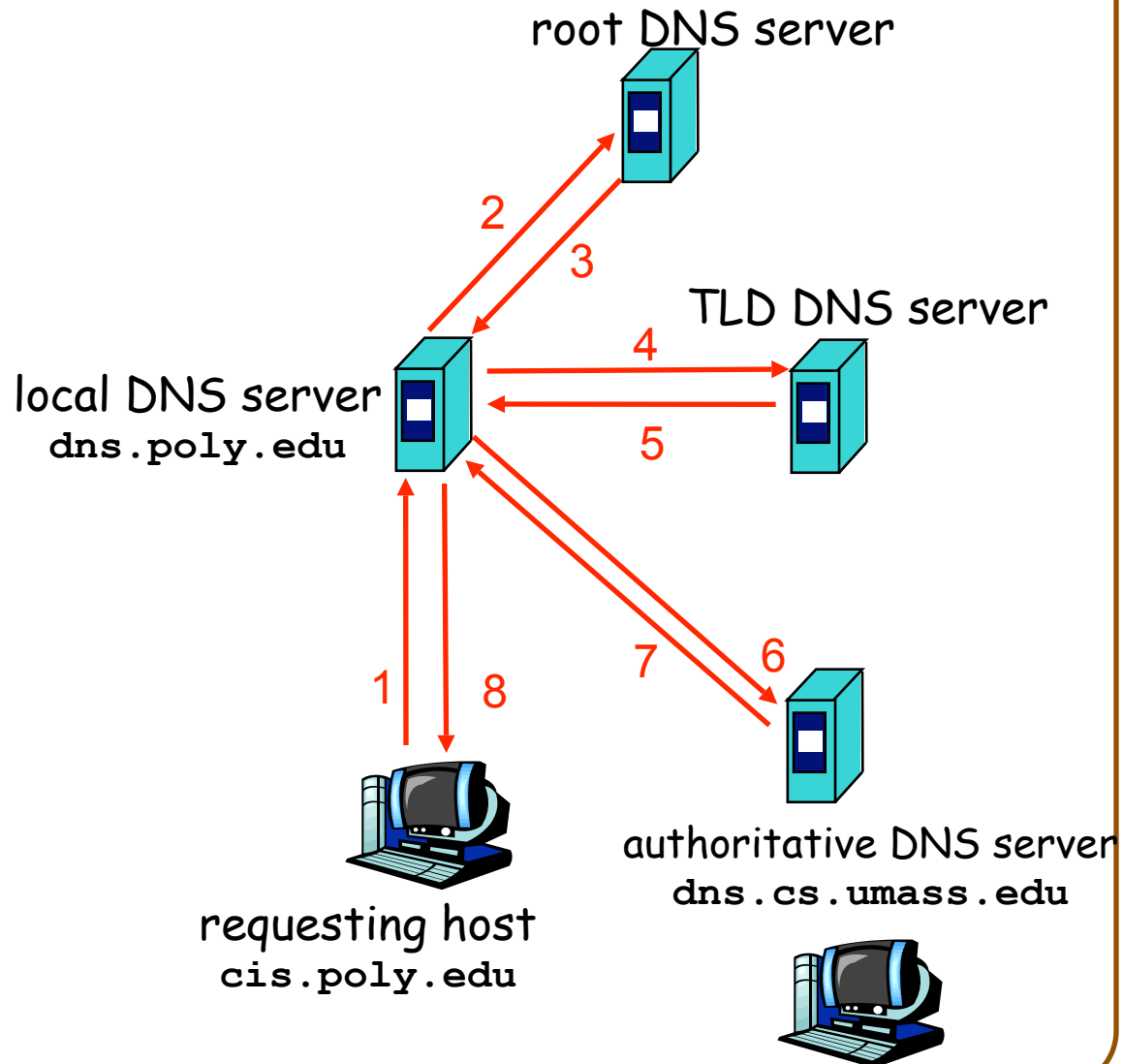
Host at cis.poly.edu  
wants IP address for  
gaia.cs.umass.edu





# 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 take 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](http://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 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

# Το πρωτόκολλο DNS

- Καθορίζει τον τρόπο επικοινωνίας ανάμεσα στους πελάτες και τους εξυπηρετητές DNS.
- Ένας πελάτης DNS στέλνει μια επερώτηση (query) για πληροφορίες σε κάποιο εξυπηρετητή DNS.
- Ένας εξυπηρετητής DNS απαντάει στις επερωτήσεις των πελατών-επιλυτών DNS και αποστέλλει επερωτήσεις σε άλλους εξυπηρετητές DNS.
- Οι επερωτήσεις DNS εκτελούνται:
  - *Αναδρομικά*: ο παραλήπτης-εξυπηρετητής αναλαμβάνει να επιλύσει ολόκληρη την επερώτηση του πελάτη.
  - *Επαναληπτικά*: ο παραλήπτης-εξυπηρετητής απαντά κατευθείαν στον πελάτη είτε με τη ζητούμενη διεύθυνση IP είτε με τη διεύθυνση του επόμενου εξυπηρετητή στην ιεραρχία DNS.
- Οι εξυπηρετητές ρίζας (root servers) του DNS χειρίζονται μόνο επαναληπτικές επερωτήσεις.



# DNS Protocol (RFC 1035)

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

|   |                          |
|---|--------------------------|
| identification  | flags                    |
| number of questions   | number of answer RRs     |
| number of authority RRs   | number of additional RRs |
| questions<br>(variable number of questions)                     |                          |
| answers<br>(variable number of resource records)                |                          |
| authority<br>(variable number of resource records)              |                          |
| additional information<br>(variable number of resource records) |                          |

↑  
12 bytes  
↓



# DNS Resource Records

DNS: distributed db storing resource records (RR)

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

- Type=A (*address record*)
  - **name** is hostname
  - **value** is IP address
- Type=NS (*name-server record*)
  - **name** is domain (e.g. foo.com)
  - **value** is hostname of authoritative name server for this domain
- Type=CNAME (*canonical name*)
  - **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 resource records

| <i>Record type</i> | <i>Meaning</i>                        | <i>Main contents</i>                              |
|--------------------|---------------------------------------|---|
| A                  | A computer address                    | IP number   |
| NS                 | An authoritative name server          | Domain name for server                            |
| CNAME              | The canonical name for an alias       | Domain name for alias                             |
| SOA                | Marks the start of data for a zone    | Parameters governing the zone                     |
| WKS                | A well-known service description      | List of service names and protocols               |
| PTR                | Domain name pointer (reverse lookups) | Domain name                                       |
| HINFO              | Host information                      | Machine architecture and operating system         |
| MX                 | Mail exchange                         | List of < <i>preference</i> , <i>host</i> > pairs |
| TXT                | Text string                           | Arbitrary text                                    |



# 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 backoff 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](http://www.myimages.com)



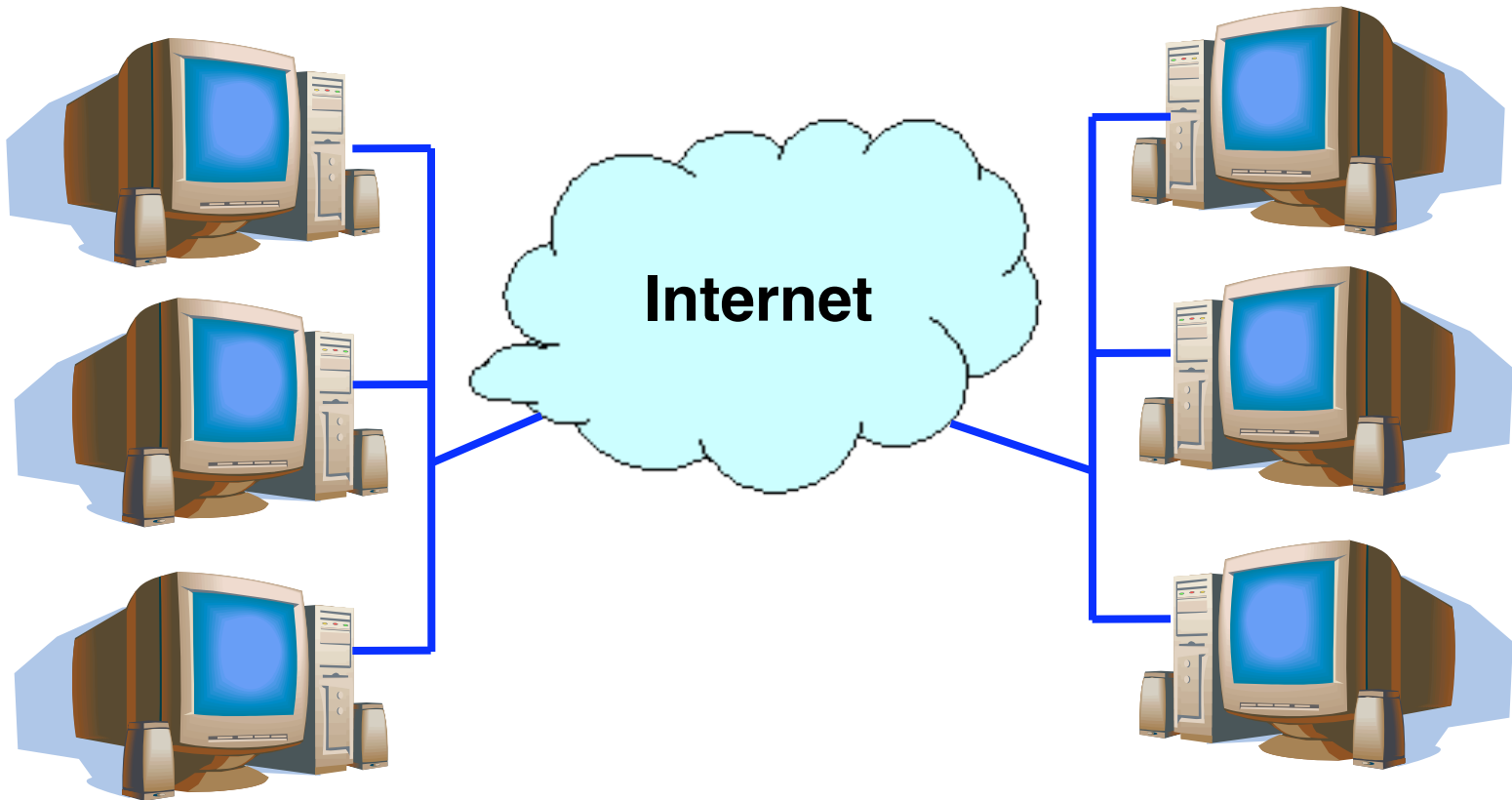
# When are DNS Queries Unnecessary?

- 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 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