



# DNS

# [ Host Names vs. IP addresses ]

## ■ Host names

- Mnemonic name appreciated by **humans**
- Variable length, full alphabet of characters
- Provide little (if any) information about physical location
- Examples: **www . cnn . com** and **bbc . co . uk**

## ■ IP addresses

- Numerical address appreciated by **routers**
- Fixed length, binary number
- Hierarchical, related to host location
- Examples: **64 . 236 . 16 . 20** and **212 . 58 . 224 . 131**



# Separating Naming and Addressing

- Names are easier to remember
  - **cnn.com** vs. **64.236.16.20** (but not shortened urls)
- Addresses can change underneath
  - Move **www.cnn.com** to **4.125.91.21**
  - e.g., renumbering when changing providers



# Separating Naming and Addressing

- Name could map to multiple IP addresses
  - [www.cnn.com](http://www.cnn.com) may refer to multiple (8) replicas of the Web site
  - Enables
    - Load-balancing
    - Reducing latency by picking nearby servers
    - Tailoring content based on requester's location/identity
- Multiple names for the same address
  - e.g., aliases like [www.cnn.com](http://www.cnn.com) and [cnn.com](http://cnn.com)



# Scalable (Name ↔ Address) Mappings

- Originally: per-host file
  - Flat namespace
  - `/etc/hosts`
  - SRI (Menlo Park) kept master copy
  - Downloaded regularly



# Scalable (Name ↔ Address) Mappings

- Why not centralize DNS?
  - Single point of failure
  - Traffic volume
  - Distant centralized database
  - Maintenance
- Doesn't scale!
- Root name server
  - Contacted by local name server that can not resolve name
  - Contacts authoritative name server if mapping not known
  - Gets mapping and returns it to local name server



# Domain Name Service (DNS)

- Large scale dynamic, distributed application
  - Replaced Network Information Center (NIC)
- RFC 1034 and 1035
- Name space
  - Set of possible names
- Bindings
  - Maps internet domain names into IP addresses
- Name server
  - Resolution mechanism



# [ Applications' use of DNS ]

- Local DNS server (“default name server”)
  - Usually near the endhosts that use it
  - Local hosts configured with local server (e.g., `/etc/resolv.conf`) or learn server via DHCP
- Client application
  - Extract server name (e.g., from the URL)
  - Do `getaddrinfo()` to trigger resolver code, sending message to server
- Server application
  - Extract client IP address from socket
  - Optional `getnameinfo()` to translate into name

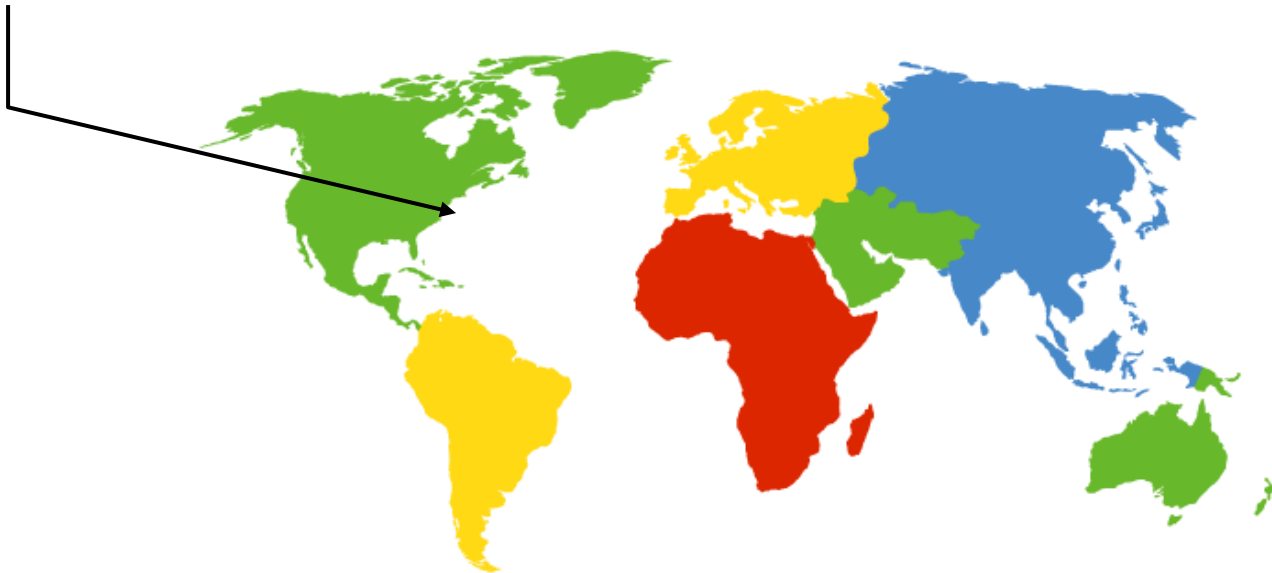




# [ DNS Root ]

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

Verisign, Dulles, VA



# DNS Root Servers

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



# [ TLD and Authoritative Servers ]

- Top-level domain (TLD) servers
  - Responsible for **com**, **org**, **net**, **edu**, etc, and all top-level country domains **uk**, **fr**, **ca**, **jp**.
    - Network Solutions maintains servers for **com** TLD
    - Educause for **edu** TLD
- Authoritative DNS servers
  - Organization's DNS servers
  - Provide authoritative hostname to IP mappings for organization's servers (e.g., Web, mail).
  - Can be maintained by organization or service provider

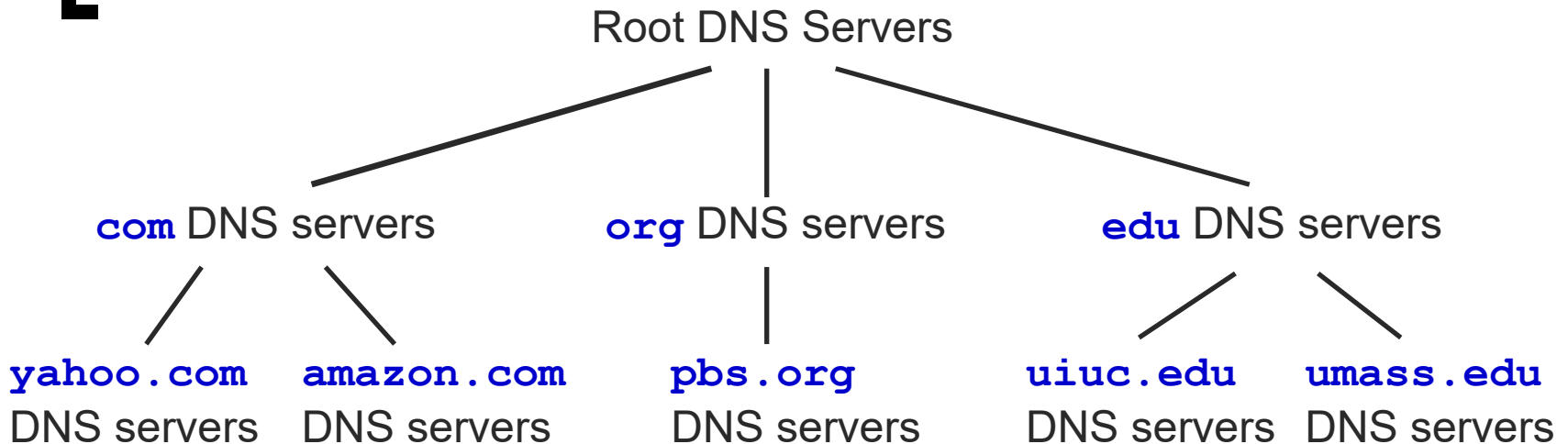


# [ Local Name Server ]

- One per ISP (residential ISP, company, university)
  - Also called “default name server”
- When host makes DNS query, query is sent to its local DNS server
  - Acts as proxy, forwards query into hierarchy
  - Reduces lookup latency for commonly searched hostnames



# Distributed, Hierarchical Database

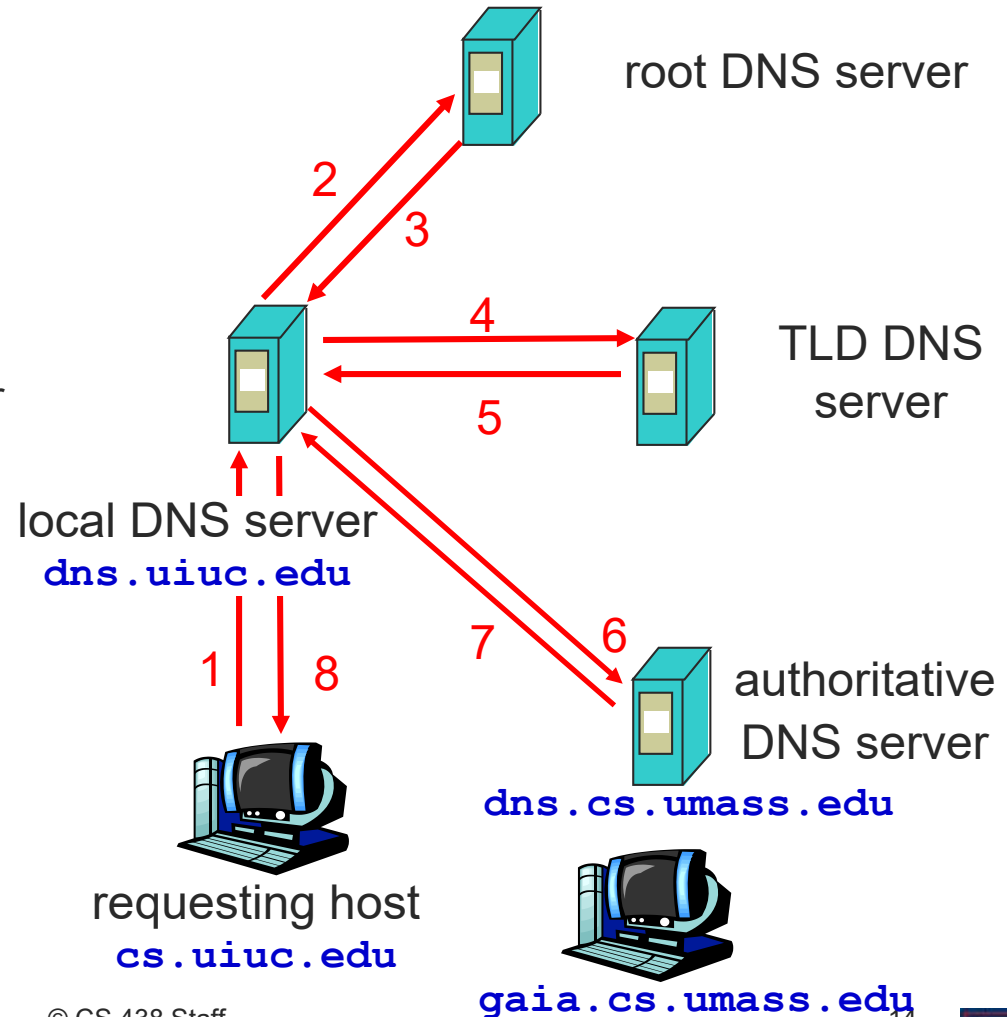


- Client wants IP for **www.amazon.com**
  - Client queries a root server to find **com** DNS server
  - Client queries **com** DNS server to get **amazon.com** DNS server
  - Client queries **amazon.com** DNS server to get IP address for **www.amazon.com**



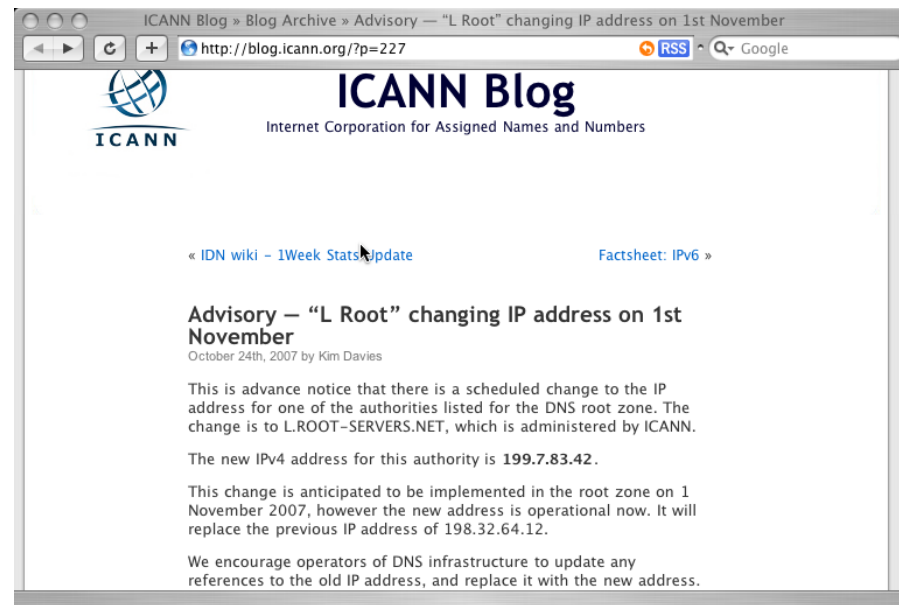
# DNS – Name Server

- Host at `cs.uiuc.edu`
  - Wants IP address for `gaia.cs.umass.edu`
- Recursive query
  - Ask server to get answer for you
  - e.g., request 1 and response 8
- Iterated query
  - Contacted server replies with name of server to contact
  - “I don’t know this name, but ask this server”



# But how did it know the root server IP?

- Hard-coded
- What if it changes?



# [ 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](http://www.cnn.com)) visited often
  - Local DNS server often has the information cached





# [ DNS: Caching ]

- How DNS caching works
  - DNS servers cache responses to queries
  - Responses include a “time to live” (TTL) field
- Once (any) name server learns mapping, it caches mapping
  - Cache entries timeout (disappear) after some time
  - TLD servers typically cached in local name servers
    - Thus root name servers not often visited



# DNS Resource Records

DNS: distributed DB storing resource records (RR)

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

- 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=PTR
  - name is reversed IP quads
    - e.g. `78.56.34.12.in-addr.arpa`
  - value is corresponding hostname
- Type=CNAME
  - name is alias name for some “canonical” name
  - e.g., `www.cs.mit.edu` is really `eeecsweb.mit.edu`
  - value is canonical name
- Type=MX
  - value is name of mailserver associated with name
  - Also includes a 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

16 bits	16 bits
Identification	Flags
# Questions	# Answer RRs
# Authority RRs	# Additional RRs
Questions (variable # of resource records)	
Answers (variable # of resource records)	
Authority (variable # of resource records)	
Additional information (variable # of resource records)	



# [ Reliability ]

- DNS servers are replicated
  - Name service available if at least one replica is up
  - Queries can be load-balanced between replicas
- Usually, UDP used for queries
  - Need reliability: must implement this on top of UDP
  - Spec supports TCP too, but not always implemented
- 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”
- Get a block of address space from ISP
  - Say **212.44.9.128/25**
- Register foobar.com at Network Solutions (say)
  - Provide registrar with names and IP addresses of your authoritative name server (primary and secondary)
  - Registrar inserts RR pairs into the **com** TLD server:
    - **(foobar.com, dns1.foobar.com, NS)**
    - **(dns1.foobar.com, 212.44.9.129, A)**
- Put in your (authoritative) server **dns1.foobar.com**:
  - Type A record for **www.foobar.com**
  - Type MX record for **foobar.com**



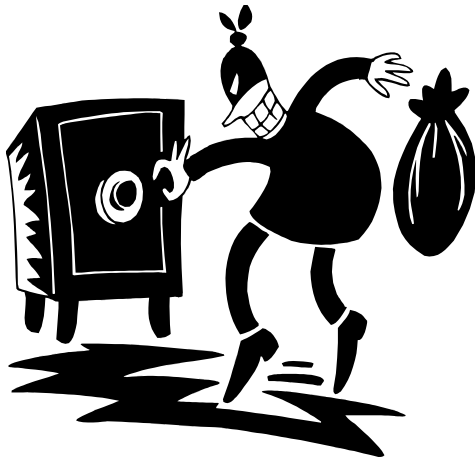
# Setting up foobar.com

- In addition, need to provide reverse PTR bindings
  - e.g., `212.44.9.129` → `dns1.foobar.com`
- Normally, these go in `9.44.212.in-addr.arpa`
- Problem
  - You can't run the name server for that domain. Why not?
  - Because your block is `212.44.9.128/25`, not `212.44.9.0/24`
  - Whoever has `212.44.9.0/25` won't be happy with you owning their PTR records
- Solution: ISP runs it for you
  - Now it's more of a headache to keep it up-to-date :-)



# Security Analysis of DNS

- What security issues does the design & operation of the Domain Name System raise?



16 bits	16 bits
Identification	Flags
# Questions	# Answer RRs
# Authority RRs	# Additional RRs
Questions (variable # of resource records)	
Answers (variable # of resource records)	
Authority (variable # of resource records)	
Additional information (variable # of resource records)	



# Security Problem #1: Starbucks (and China...)

- As you sip your latte and surf the Web, how does your laptop find [google.com](https://www.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 in **either** direction
- How can you know you’re getting correct data?
  - Today, you can’ t. (Though if site is HTTPS, that helps)
  - One day soon: **DNSSEC** extensions to DNS





# Security Problem #2: 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:

```
;; QUESTION SECTION:
;www.foobar.com.          IN      A
                           Evidence of the attack
                           disappears 5 seconds later!

;; ANSWER SECTION:
www.foobar.com.          300     IN      A      212.44.9.144

;; AUTHORITY SECTION:
foobar.com.              600     IN      NS     dns1.foobar.com.
foobar.com.              600     IN      NS     google.com.

;; ADDITIONAL SECTION:
google.com.              5       IN      A      212.44.9.155
```



# [ 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



# Cache Poisoning

- Suppose Bad Guy is at Starbucks 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 Starbucks customer is served the bogus answer out of the local server's cache
    - In this case, the reply uses a **large** TTL



# [ Summary ]

- Domain Name System (DNS)
  - Distributed, hierarchical database
  - Distributed collection of servers
  - Caching to improve performance
- DNS currently lacks authentication
  - Can't tell if reply comes from the correct source
  - Can't tell if correct source tells the truth
  - Malicious source can insert extra (mis)information
  - Malicious bystander can spoof (mis)information
  - Playing with caching lifetimes adds extra power to attacks

