CSC465 – Computer Networks
Spring 2004

Dr. J. Harrison

These slides are based on the text
"TCP/IP Protocol Suite (2nd Edition)"

Chapter 18

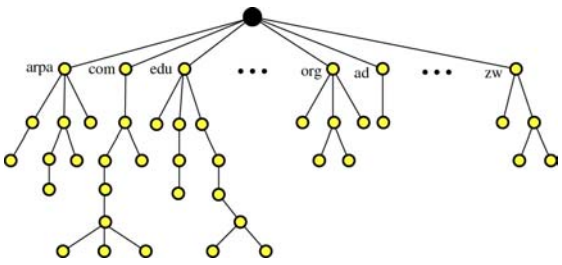
Domain Name System (DNS)

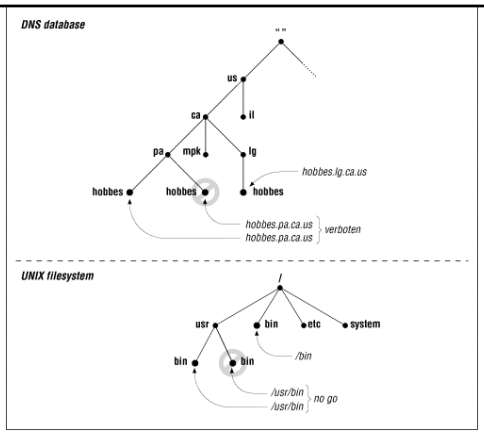
CONTENTS

Domain Name Space

- Hierarchical name space
- Names are designed with inverted-tree structure (root at top)
- Max 128 levels
- Each level of tree defines a hierarchical level in the name space
- Each node in tree has a label with max 63 chars
- Root label is null string
- To ensure uniqueness, DNS requires that children of node have different labels

Domain name space



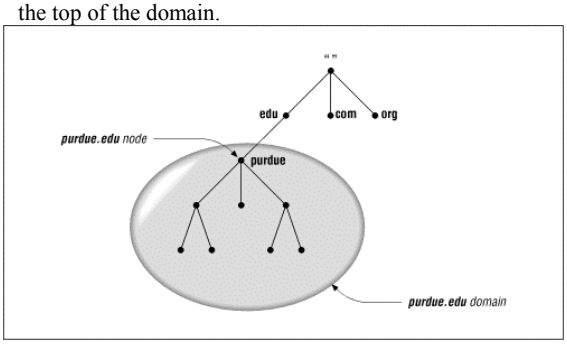


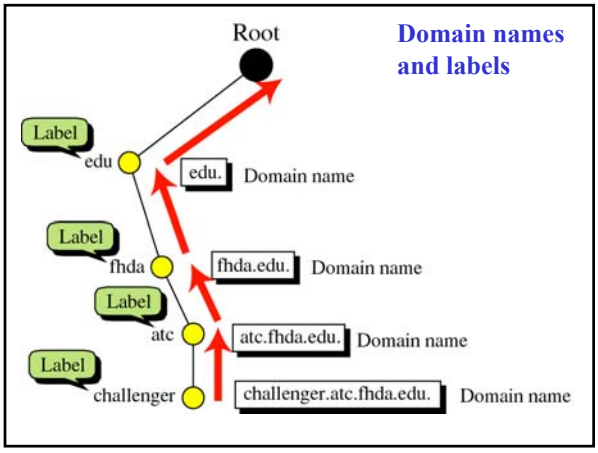
Domain Name Space

- Each node in the tree has a domain name
- Full domain name is a sequence of labels separated by dots
- Domain names read from node to root
- Last label is label of root (null)
- All full domain names end with a dot (“.”)

Domain is a subtree of the domain name space.

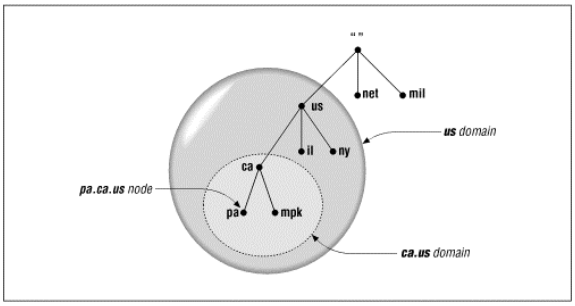
The name of a domain is the domain name of the node at the top of the domain.





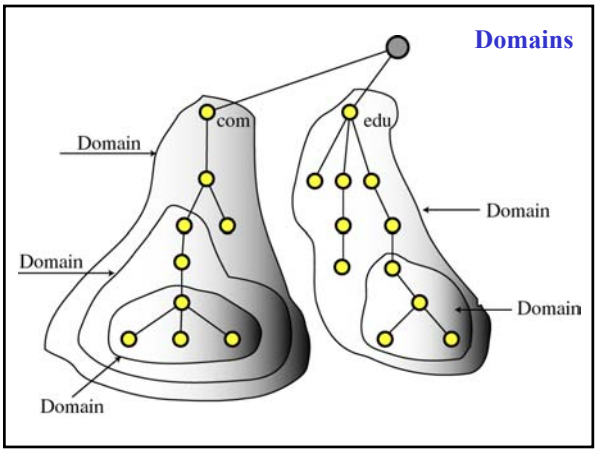
A node can be in multiple domains

pa.ca.us. is in both *ca.us.* domain and *us.* domains



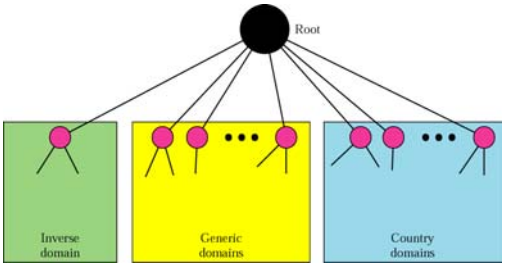
Examples of FQDN and PQDN

FQDN	PQDN
challenger.atc.fhda.edu.	challenger.atc.fhda.edu
cs.hmme.com.	cs.hmme
www.funny.int.	www

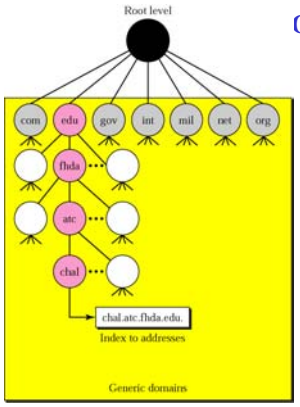


DISTRIBUTION OF NAME SPACE

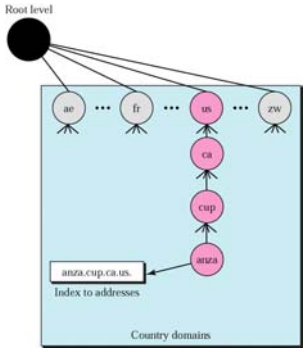
DNS in the Internet



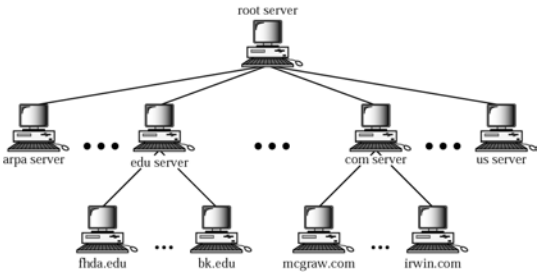
Generic domains



Country domains



Hierarchy of name servers



Centralization of DNS?

- single point of failure
- traffic volume
- distant centralized database
- maintenance

doesn't *scale!*

Delegation

- key goals of the design of the Domain Name System was to decentralize administration (achieved via *delegation)*
- an organization administering a domain can divide it into subdomains
- Each subdomain can be *delegated* to another organization
- Each organization becomes responsible for maintaining all the data in that subdomain
  - can freely change the data and even divide its subdomain up into more subdomains and delegate those
  - The parent domain contains only pointers to sources of the subdomain's data

Name Server

- programs that store information about the domain name space are called *name servers*
- Name servers generally have complete information about some part of the domain name space, called a *zone*
- A name servers can load DNS data from a file or from another name server
  - it then has *authority* for that zone
- Name servers can be authoritative for multiple zones

Name Server

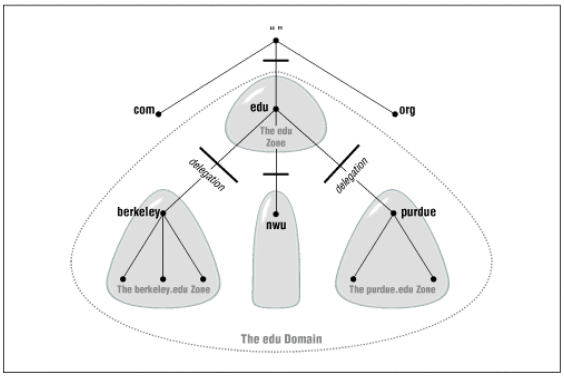
- no server has all name-to-IP address mappings

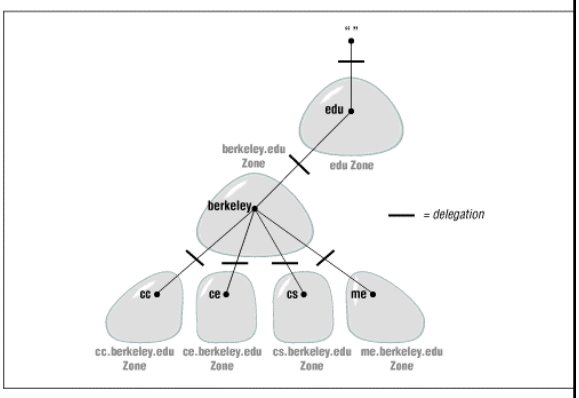
local name servers:

  - each ISP, company has *local (default) name server*
  - host DNS query first goes to local name server

authoritative name server:

  - for a host: stores that host's IP address, name
  - can perform name/address translation for that host's name

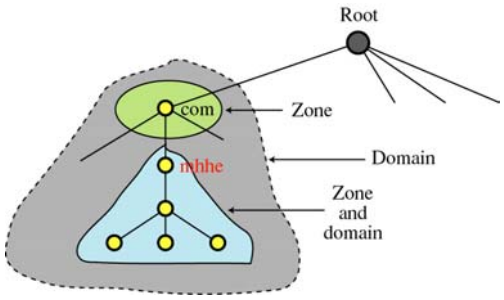




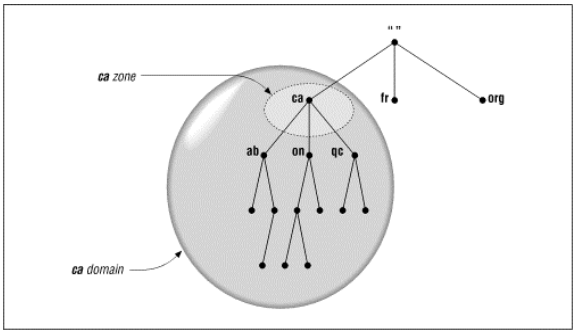
Zones vs. Domains

- All top-level domains, and many domains at the second level and lower, like *unlv.edu* and *sun.com*, are broken into smaller, more manageable units by delegation (called ***zones***)
- otherwise, the domain manager would have to manage the subdomains themselves
- makes much more sense to delegate
  - distribute work
  - subdomain management become autonomous

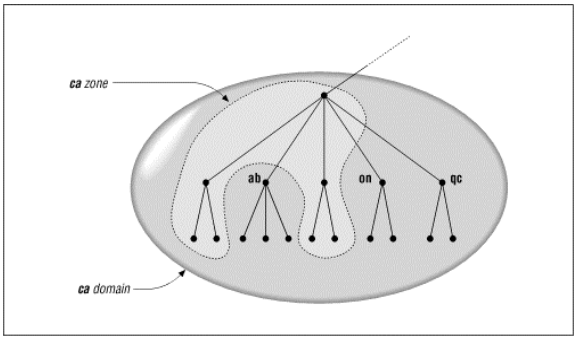
Zones and domains



Example: All Canadian states have responsibility for their own subdomains; complete delegation of subdomains



Example: Two Canadian states managed within the ***ca*** zone; other 3 have their subdomains delegated to them



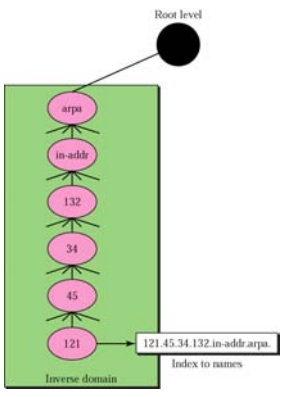
Name Server Types

- DNS specs define two types of name servers: *primary* and *secondary*
- A *primary* name server for a zone reads the data for the zone from a file on its host
- A *secondary* name server for a zone gets the zone data from another name server that is authoritative for the zone (called its *master* server)
  - a *secondary* can load zone data from another *secondary,* which would be termed its *master*
- When a secondary starts up, it may pulls the zone data over from its master server ***(zone transfer)***

Name Server Types

- Both the primary master and slave name servers for a zone are authoritative for that zone
- Slave name servers are important because it's a good idea to set up more than one name server for any given zone
- a name server can be authoritative for >1 zone
- a name server can be a primary master for one zone and a slave for another

Inverse domain



Resolvers

- clients that access name servers
- Programs running on a host that need information from the domain name space use the resolver
- The resolver handles:
  - Querying a name server
  - Interpreting responses (which may be *resource records* or an error)
  - Returning the information to the programs that requested it

Resolution

- Name servers are adept at retrieving data from the domain name space
- they provide data about zones for which they're authoritative
- they can also search through the domain name space to find data for which they're not authoritative
- process is called ***name resolution***

Root Name Servers

- know where there are authoritative name servers for each of the top-level domains
  - most are authoritative for the generic top-level domains
- top-level name servers can provide the list of name servers that are authoritative for the second-level domain
- Each name server queried gives the querier info about how to get "closer" to the answer it's seeking, or it provides the answer itself.

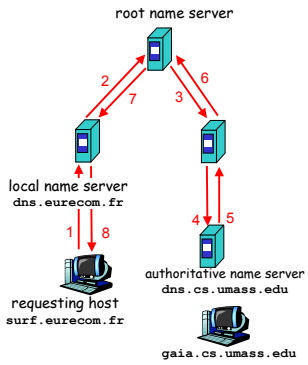
Root Name Servers

- if no caching, resolution would have to start at the root name servers
- root name servers crucial to the operation of DNS
- if all the Internet root name servers were unreachable for an extended period, all resolution on the Internet would fail
- ∴∃ 13 for redundancy

"The"13 Root Name Servers

Root name server:

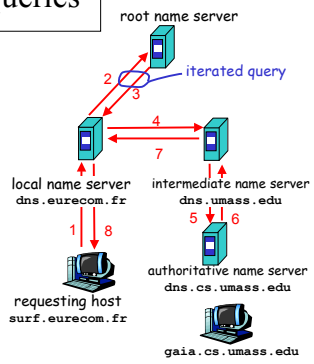
- may not know authoritative name server
- may know *intermediate name server:* who to contact to find authoritative name server



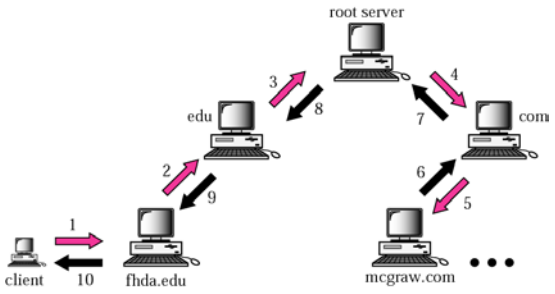
DNS: Iterated Queries

recursive query:
- puts burden of name resolution on contacted name server
- heavy load?

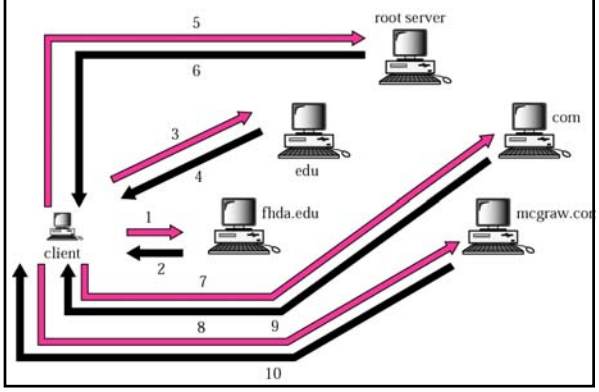
iterated query:
- contacted server replies with name of server to contact
- "I don't know this name, but ask this server"



Recursive resolution



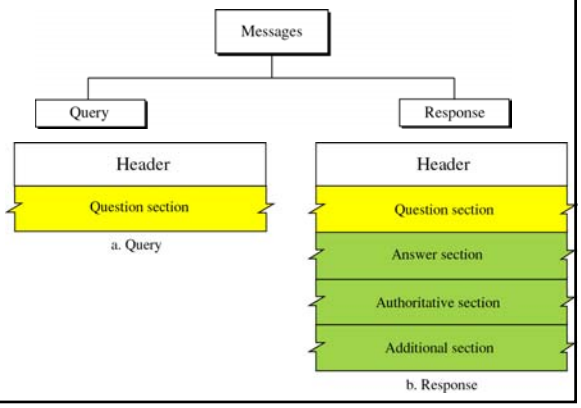
Iterative resolution



DNS: caching and updating records

- once (any) name server learns mapping, it *caches* mapping
  - cache entries timeout (disappear) after some time (TTL field)
- update/notify mechanisms under design by IETF
  - RFC 2136
  - http://www.ietf.org/html.charters/dnsind-charter.html

DNS Query and Response Messages



Header format

Identification	Flags
Number of question records	Number of answer records (All 0s in query message)
Number of authoritative records (All 0s in query message)	Number of additional records (All 0s in query message)

16-bit field ***Identification*** used by the client to match the response with the query

Client uses a different ID# each time its sends a query

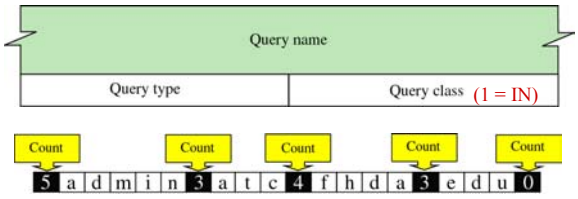
Server duplicates this number in the corresponding response

Flags fields

QR: Query/Response
OpCode: 0 standard, 1 inverse, 2 server status
AA: Authoritative (was responding server?)
TC: Truncated (too large response for UDP)
RD: Recursion Desired (both query & response)
RA: Recursion Available (only response)
rCode: Status of the error

TYPES OF RECORDS

Question record format

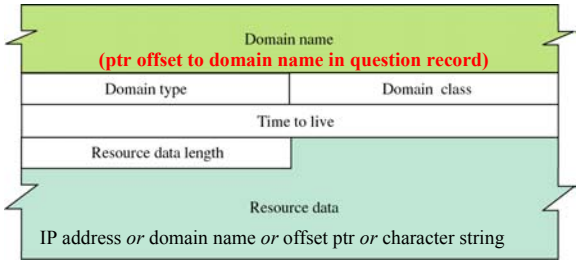


admin.atc.fhda.edu.

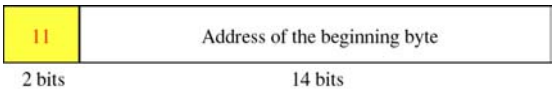
Some Query Types

- **Type=A;** host name to host IP
  - Ex: ***(relay1.bar.foo.com, 145.37.93.126, A)***
- **Type=NS**; domain to an authoritative name server for domain
  - Ex: ***(foo.com, dns.foo.com, NS)***
- **Type=CNAME;** provides canonical hostname for alias hostname
  - Ex: ***(foo.com, relay1.bar.foo.com,CNAME)***
- **Type=MX;** canonical name of a mailserver
  - Ex: ***(foo.com, mail.bar.foo.com, MX)***

Resource record format



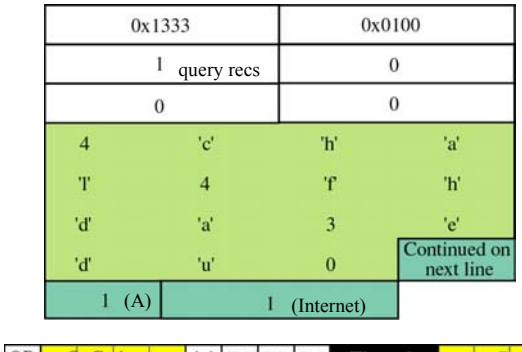
Format of an offset pointer



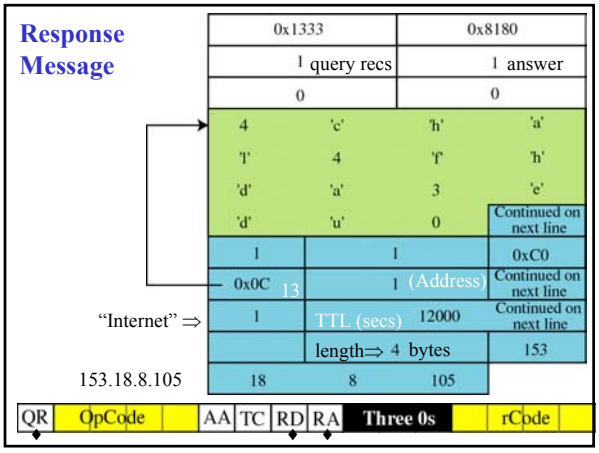
Example 1

A resolver sends a query message to a local server to find the IP address for the host "*chal.fhda.edu*.". We discuss the query and response messages separately.

Example of a query message



Response Message



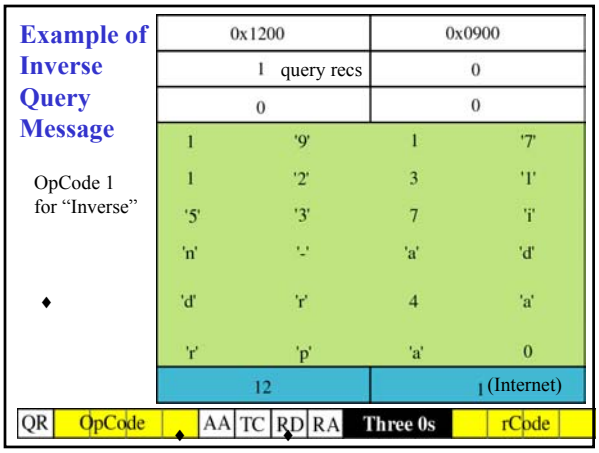
Example 2

An FTP server has received a packet from an FTP client with IP address 153.2.7.9. The FTP server wants to verify that the FTP client is an authorized client.

The list of authorized clients are specified as domain names. The FTP server asks the resolver (DNS client) to send an inverse query to a DNS server to obtain domain name.

Example of Inverse Query Message

OpCode 1 for "Inverse"



Example of inverse response message

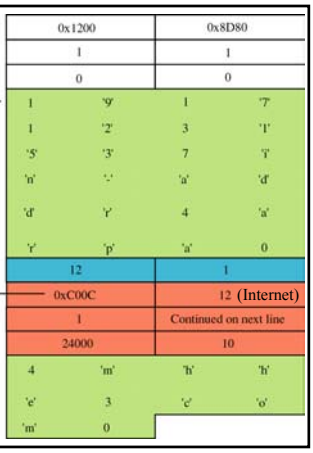
"in-addr.arpa" is added to domain space

153.2.7.9 ⇒

"9.7.2.153.in-addr.arpa"

12 ⇒ query type is PTR

12 ⇒ domain type is PTR



DNS can use the services of UDP or TCP using the well-known port 53.

END OF PRESENTATION