Chapter 2: Application layer

2.1 Principles of network applications

2.2 Web and HTTP

2.3 FTP

2.4 Electronic Mail

- SMTP, POP3, IMAP

2.5 DNS

2.6 P2P applications

2.7 Socket programming with TCP

2.8 Socket programming with UDP

DNS: Domain Name System

people: many identifiers:

- SSN, name, passport #

Internet hosts, routers:

- IP address used for addressing packets
  - 4 bytes or 32 bits
  - e.g., 129.23.4.51
  - used for routing
- "name",
  - e.g., www.yahoo.com
  - variable length
  - used by humans

Q: map between IP address and name, and vice versa ?

Domain Name System (DNS):

- *distributed database*
  - implemented in hierarchy of many *name servers*
  - *stores DNS records (RRs)*
- *application-layer protocol*
  - host, routers, name servers communicate to *resolve* names (address/name translation)
  - Runs on top of UDP, port 53
  - RFC 1035 by Mockapetris (UCI grad)

DNS Summary

- Core Internet functionality
- Implemented as a network application
  - On top of UDP, port 53
  - Defined in RFCs: 1034, 1035 (1987)
  - http://www.ietf.org/rfc/rfc1035.txt
  - Proposed by Mockapertis (UCI PhD 1982)
    - http://en.wikipedia.org/wiki/Paul_Mockapetris
  - Many extensions, e.g. DNSSEC

DNS services

- hostname to IP address translation

```
nslookup (or dig,whois) athina.calit2.uci.edu

Name:       athina.calit2.uci.edu
Address:    128.195.177.83
```

- host aliasing: canonical vs. alias names

```
nslookup  (or dig) www.cnn.com
www.cnn.com    canonical name = www.cnn.com.vgtf.net
www.cnn.com.vgtf.net canonical name = cnn-56m.gslb.vgtf.net.
Name:       cnn-56m.gslb.vgtf.net
Address:    157.166.249.11
Name:       cnn-56m.gslb.vgtf.net
Address:    157.166.248.10
```

- mail server aliasing

```
Nslookup –type=mx stanford.edu
Stanford.edu    mail exchanger = 40 mx1.stanford.edu.
stanford.edu    mail exchanger = 20 mx2.stanford.edu.
stanford.edu    mail exchanger = 20 mx3.stanford.edu.
```

- load distribution
  - replicated Web servers: set of IP addresses for one canonical name
  - rotating

```
nslookup (or dig) google.com

Name:       google.com
Address:    74.125.227.167
Name:       google.com
Address:    74.125.227.168
Name:       google.com
Address:    74.125.227.169
.....
```

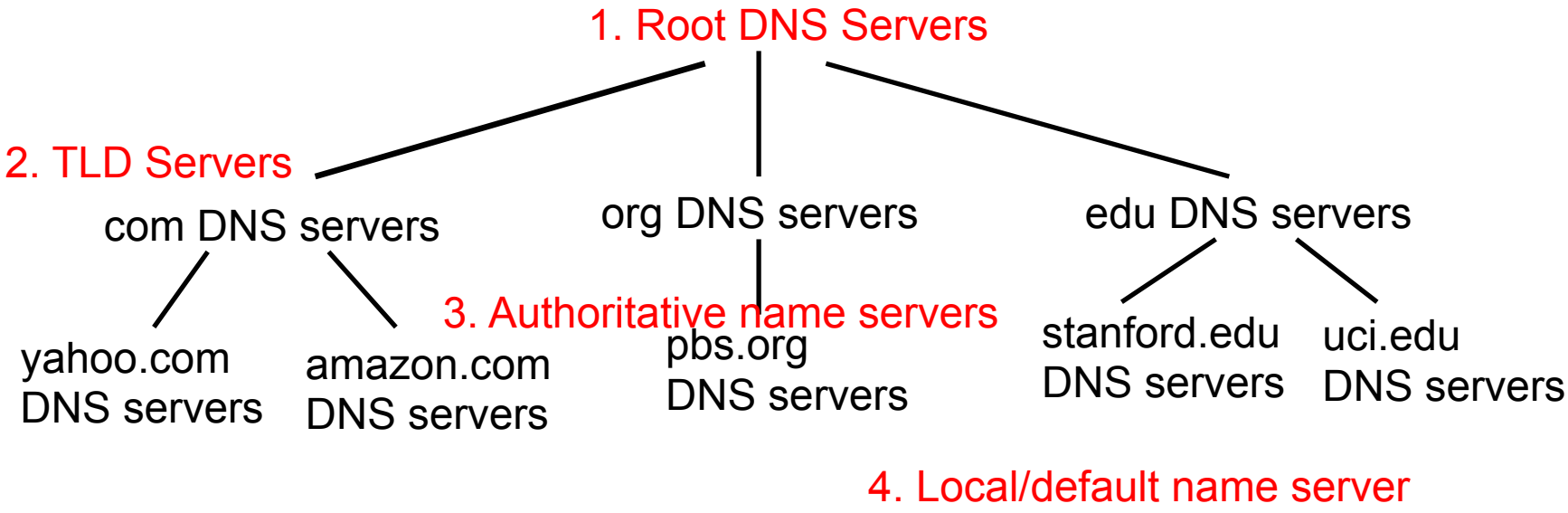
Why not a centralized DNS?

Why not centralized DNS?

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

doesn't *scale!*

Distributed, Hierarchical Database

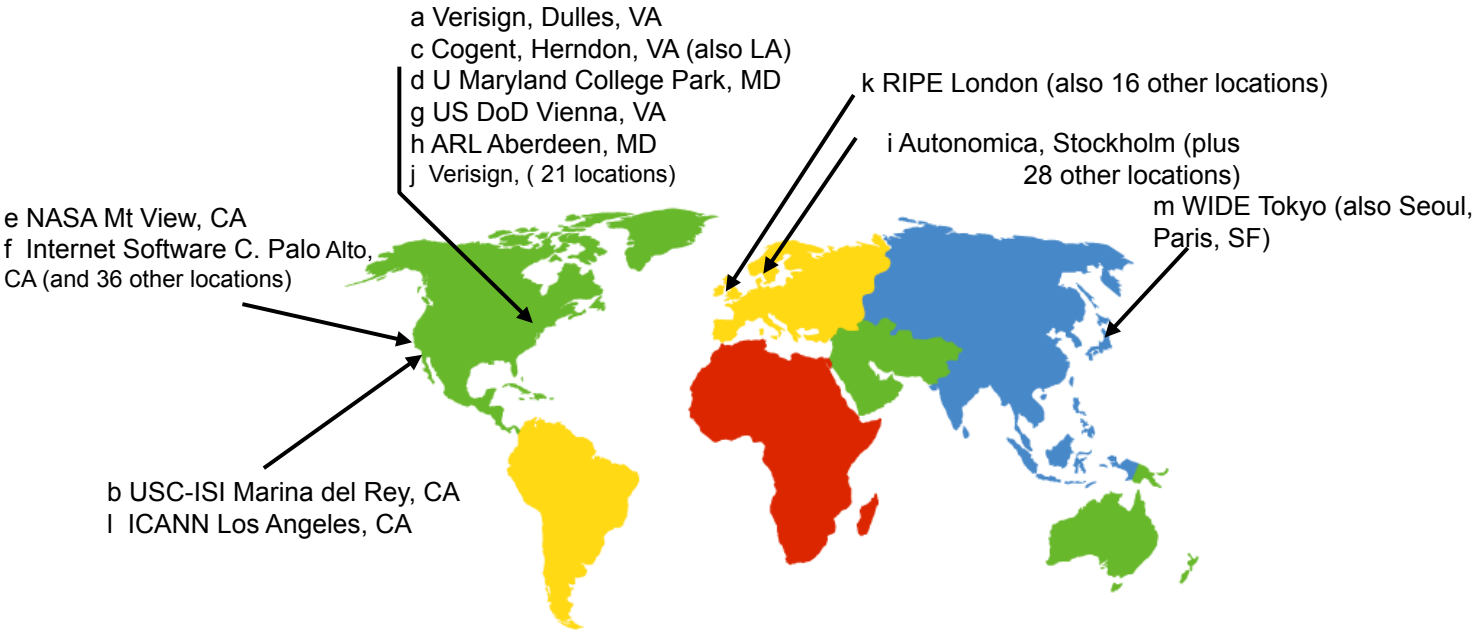


client wants IP for www.amazon.com; 1st approx:

- 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

1. Root name servers

- 13 root name servers worldwide: a, b…m
  - in fact replicated: 247 root servers as of 2011
- contacted by local name server that cannot resolve name
- root name server:
  - contacts authoritative name server if name mapping not known
  - gets mapping
  - returns mapping to local name server



https://www.iana.org/domains/root/servers

TLD and Authoritative Servers

2. Top-level domain (TLD) servers:

- responsible for com, org, net, edu, aero, jobs, museums, and all top-level country domains, e.g.: uk, fr, ca, jp
- Network Solutions maintains servers for .com TLD
- Educause for .edu TLD
- https://www.iana.org/domains/root/db
  - E.g. look up who is running TLDs for country codes

3. Authoritative DNS servers:

- organization's DNS servers, providing authoritative hostname to IP mappings for organization's servers (e.g., Web, mail).
- can be maintained by organization or service provider

4. Local Name Server

- does not strictly belong to the hierarchy
- each ISP (residential ISP, company, university) has one
- 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
    - caches records
- Example: /etc/resolv.conf

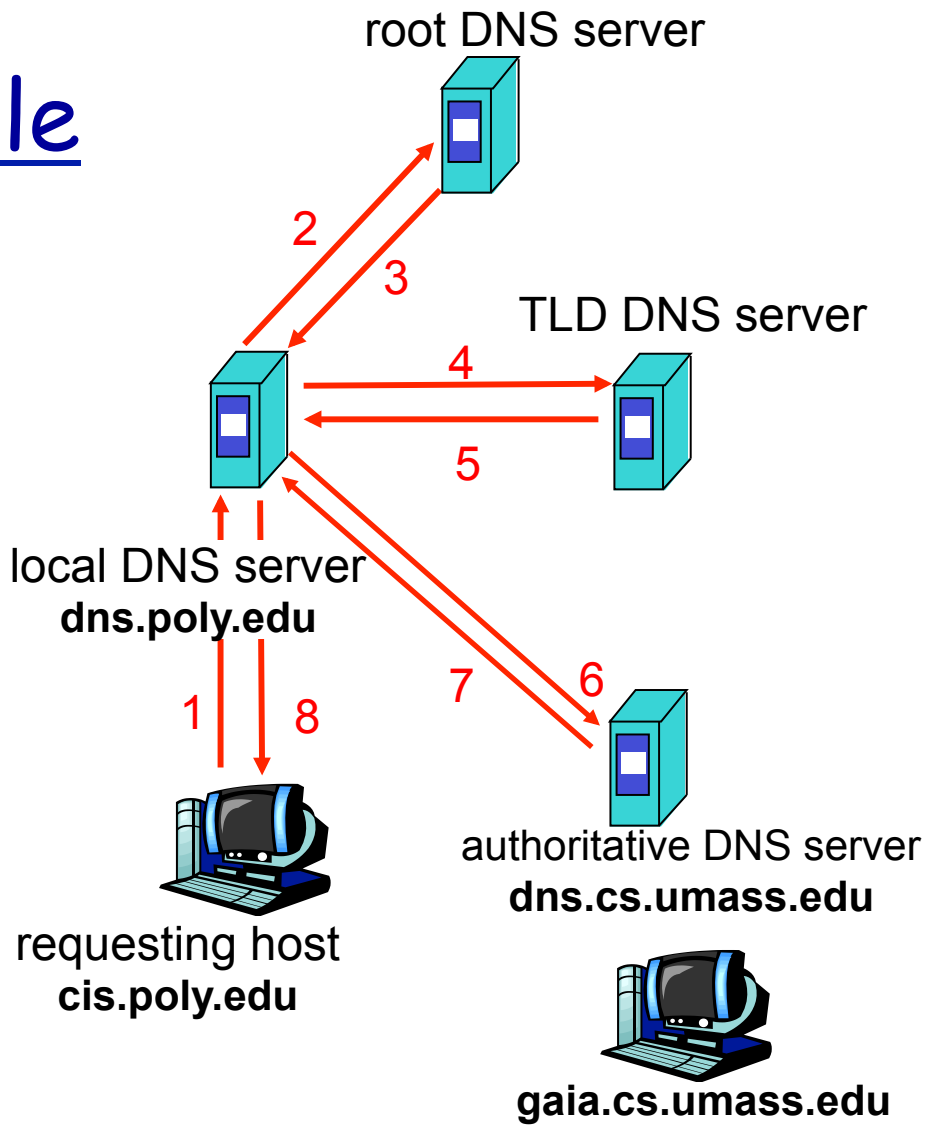
```
Athina-Markopoulous-MacBook-Air-4:2015 athina$ more /etc/resolv.conf
#
# Mac OS X Notice
#
# This file is not used by the host name and address resolution
# or the DNS query routing mechanisms used by most processes on
# this Mac OS X system.
#
# This file is automatically generated.
#
nameserver 68.105.28.11
nameserver 68.105.29.11
nameserver 68.105.28.12
Athina-Markopoulous-MacBook-Air-4:2015 athina$ nslookup 68.105.28.11
Server:         68.105.28.11
Address:        68.105.28.11#53

Non-authoritative answer:
11.28.105.68.in-addr.arpa    name = cdns1.cox.net.

[Repeat from a uci machine.....]
```

DNS name resolution example

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

iterative query:

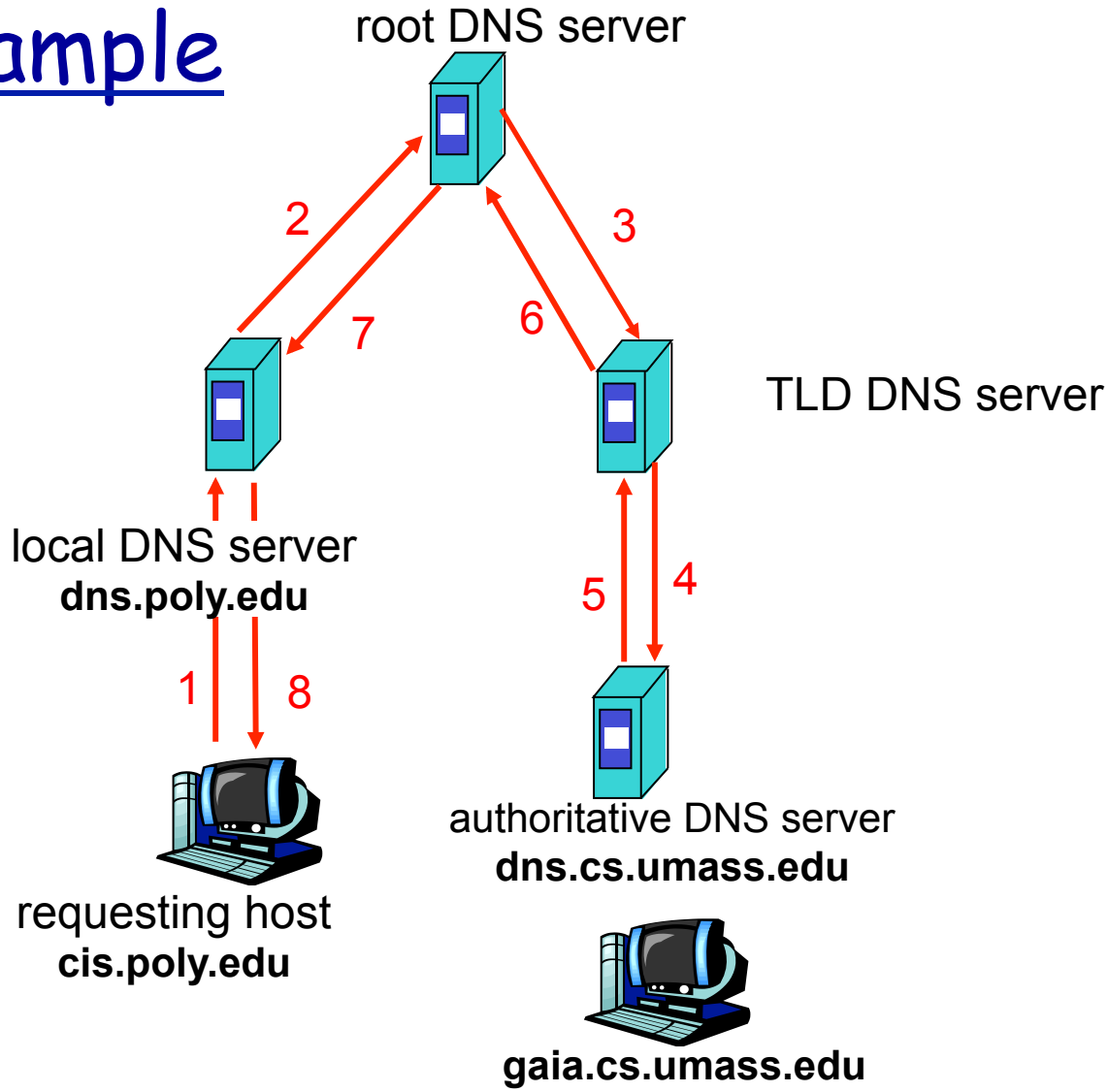
- contacted server replies with name of server to contact
- "I don't know this name, but ask this server"
- Here all but first query are iterative



DNS name resolution example

recursive query:

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



DNS: caching and updating records

- once (any) name server learns a mapping, it *caches* it
  - TLD servers typically cached in local name servers
    - thus root name servers not often visited
  - cache entries timeout (disappear) after some time
    - Time-to-live (TTL) by default is 2 days
    - Needed because records change often

- How to configure the records in the database
  - Statically
  - update/notify mechanisms RFC 2136

DNS records -Summary

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=CNAME

- **name** is alias name for some "canonical" (the real) name
- **value** is canonical name
- E.g. `www.ibm.com` is really `servereast.backup2.ibm.com`

Type=MX

- **value** is name of mailserver associated with **name**

DNS records

DNS: distributed db storing resource records (RR)

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

Type=A

- **name** is hostname
- **value** is IP address
- Stored at authoritative server of that domain

Example

- (**odysseas.calit2.uci.edu, 128.195.185.112,** A)
  - You can lookup this info (both directions)
    - by command line, e.g.: `nslookup or dig`
    - or on the web, e.g. `http://www.kloth.net/services/nslookup.php`

DNS records

DNS: distributed db storing resource records (RR)

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

Type=NS

- **name** is domain (e.g., foo.com)
- **value** is hostname of authoritative name server for this domain
- this record is used to route a request further

Example

- **(uci.edu, ns1.service.uci.edu, NS)**
  - type "`nslookup -ty=ns uci.edu`"

DNS records

DNS: distributed db storing resource records (RR)

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

Type=MX

- **value** is name of mailserver associated with **name**

Example

- **(uci.edu, mta.service.uci.edu, MX)**
  - type "`nslookup -ty=mx uci.edu`"
  - type "`nslookup -ty=mx stanford.edu`"
  - Can have multiple `NS` and `MS` records
  - several `MX` records, allow for load balancing

DNS records

DNS: distributed db storing resource records (RR)

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

Type=CNAME

- **name** is alias name for some "canonical" (the real) name
- **value** is canonical name

Example

- (www.networkedsystems.uci.edu, odysseas.calit2.uci.edu, CNAME)
  - "nslookup -type=cname networkedsystems.uci.edu"
  - alias, and potential for load balancing
  - a company can have the same alias for several servers running on the same IP address…

Example of CNAME

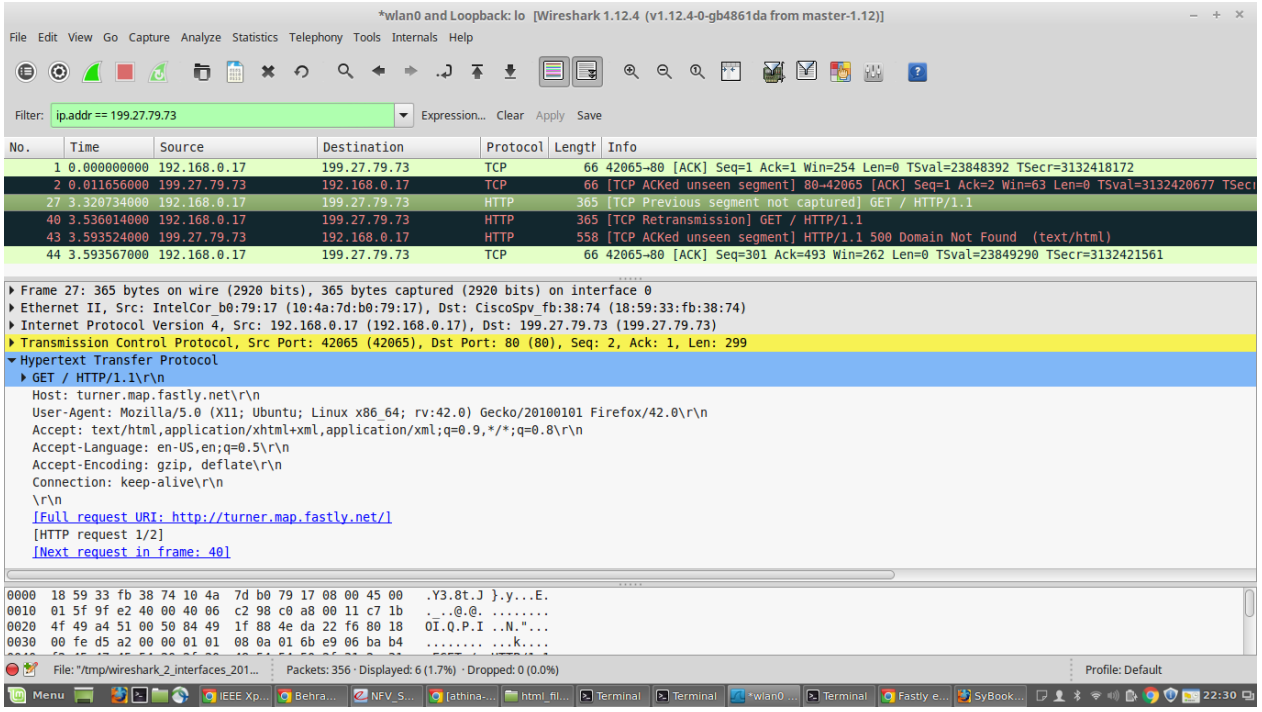
- Better example: networkedsystems.uci.edu
- Virtual hosting on odysseas.calit2.uci.edu
- Try it on the browser

```
Athina-Markopoulous-MacBook-Air-4:~ athina$ nslookup networkedsystems.uci.edu
Server:         68.105.28.11
Address:    68.105.28.11#53

Non-authoritative answer:
networkedsystems.uci.edu        canonical name = odysseas.calit2.uci.edu.
Name:   odysseas.calit2.uci.edu
Address: 128.195.185.112

Athina-Markopoulous-MacBook-Air-4:~ athina$ nslookup odysseas.calit2.uci.edu
Server:         68.105.28.11
Address:    68.105.28.11#53

Non-authoritative answer:
Name:   odysseas.calit2.uci.edu
Address: 128.195.185.112
```

Question from last time about CNAME

- cnn.com hosted on Fastly CDN
- DNS correctly resolves cname and finds IP address
- turner.map.fastly.net gets resolved and responds to:
  - ping, telnet at port 80, or HTTP requests from the browser
- It returns an error:
  - probably has to do with CDN virtual hosting

nslookup (or dig) www.cnn.com
www.cnn.com canonical name = turner.map.fastly.net



More on CNAME

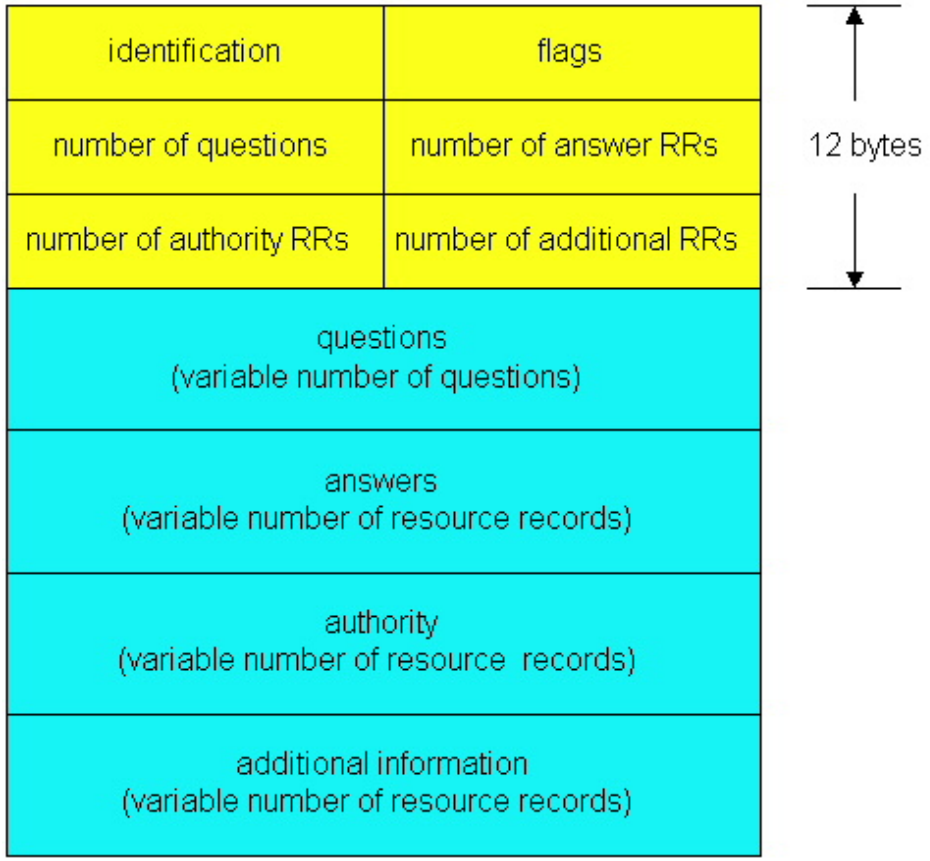
- https://en.wikipedia.org/wiki/CNAME_record
  - "This can prove convenient when running multiple services (like an FTP server *and* a webserver; each running on different ports) from a single IP address. This can prove convenient when running multiple services (like an FTP server *and* a webserver; each running on different ports) from a single IP address. One can, for example, point ftp.example.com and www.example.com to the DNS A record for example.com, which in turns points to the IP address. Then, if the IP address ever changes, one only has to record the change in one place within the network: in the DNS A record.
- See RFCs:
  - https://www.ietf.org/rfc/rfc1034.txt
  - https://tools.ietf.org/html/rfc2181

DNS protocol, messages

DNS protocol : *query* and *reply* messages, both with same *message format*

msg header

- identification: 16 bit # for query, reply to query uses same #
- flags:
  - query or reply
  - recursion desired
  - recursion available
  - reply is authoritative



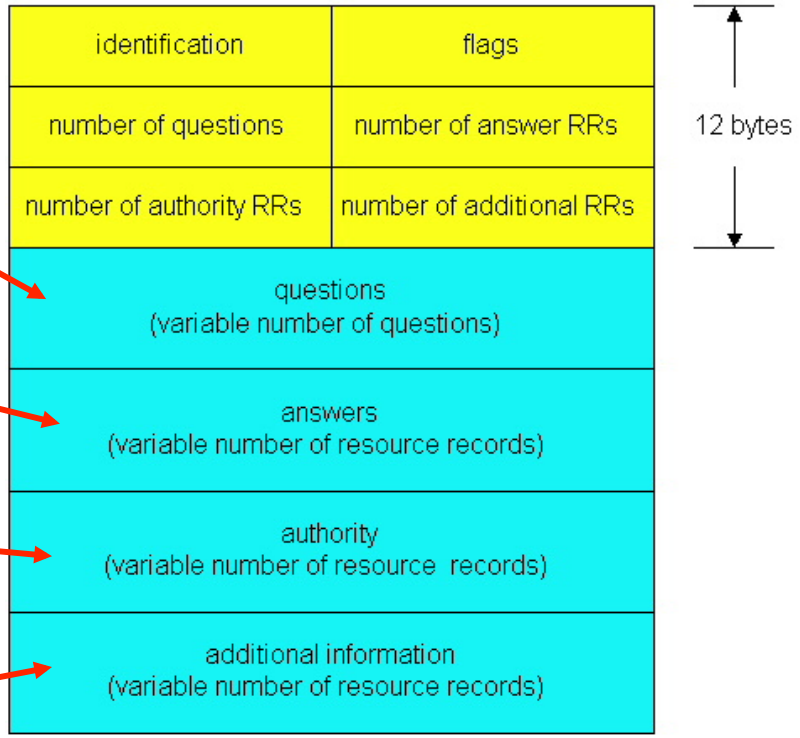
DNS protocol, messages

Name, type fields for a query

(>=1) RRs in response to query

(>=1) records Pointing to authoritative servers

additional "helpful" info that may be used



http://www.ietf.org/rfc/rfc1035.txt

Example: inserting records into DNS

- Example: new startup "Network Utopia"
- Register name networkutopia.com at *DNS registrar* (e.g., Network Solutions, see www.internic.net)
  - provide names, IP addresses of authoritative name server (primary and secondary), verifies uniqueness, puts into database for a small fee, acredited by ICANN
  - registrar inserts two RRs into .com TLD server:

```
(networkutopia.com, dns1.networkutopia.com, NS)
(dns1.networkutopia.com, 212.212.212.1, A)
(networkutopia.com, dns2.networkutopia.com, NS)
(dns2.networkutopia.com, 212.212.212.2, A)
```

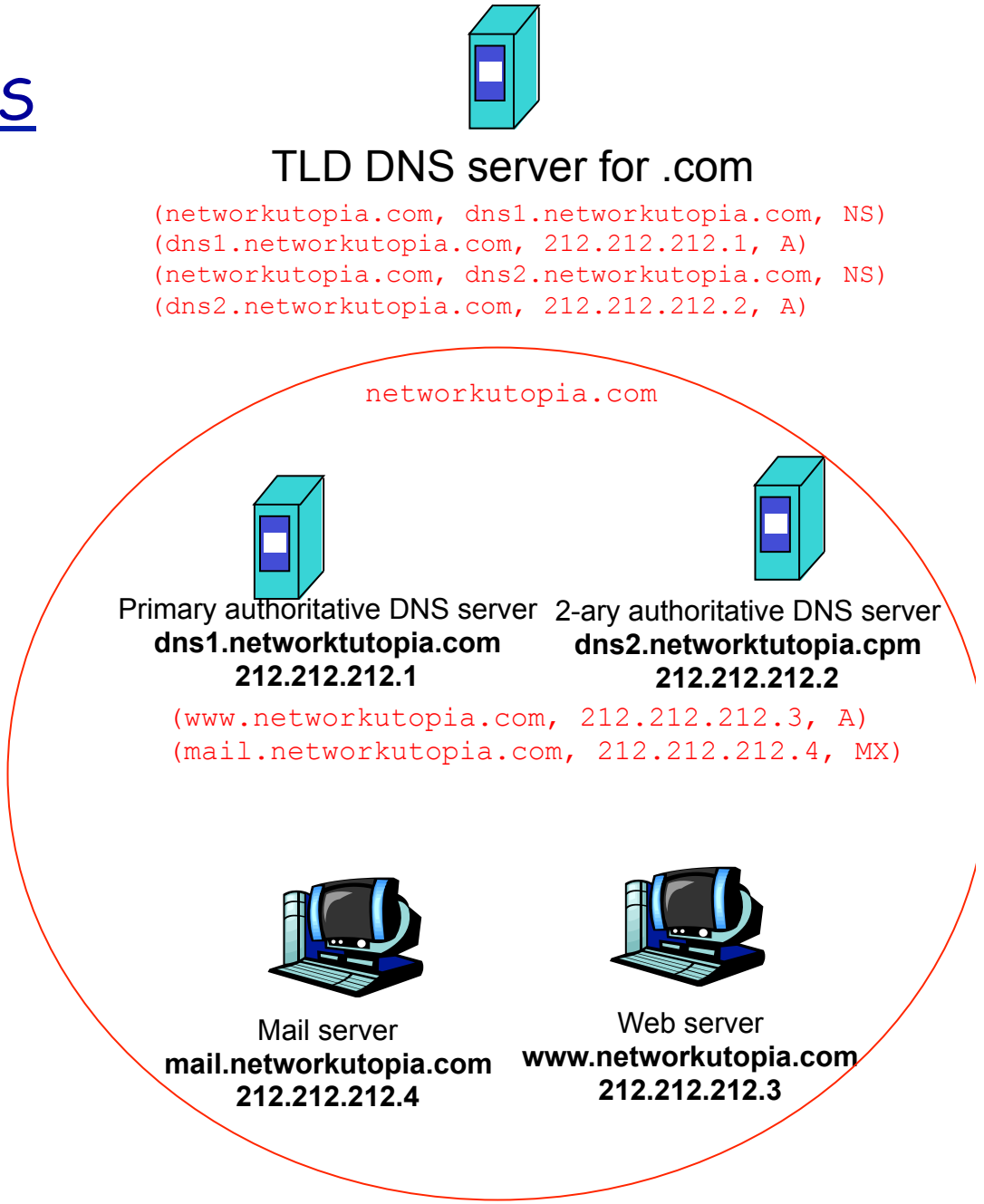
  - Create Type A record in your own authoritative server

```
(www.networkutopia.com, 212.212.212.3, A)
```

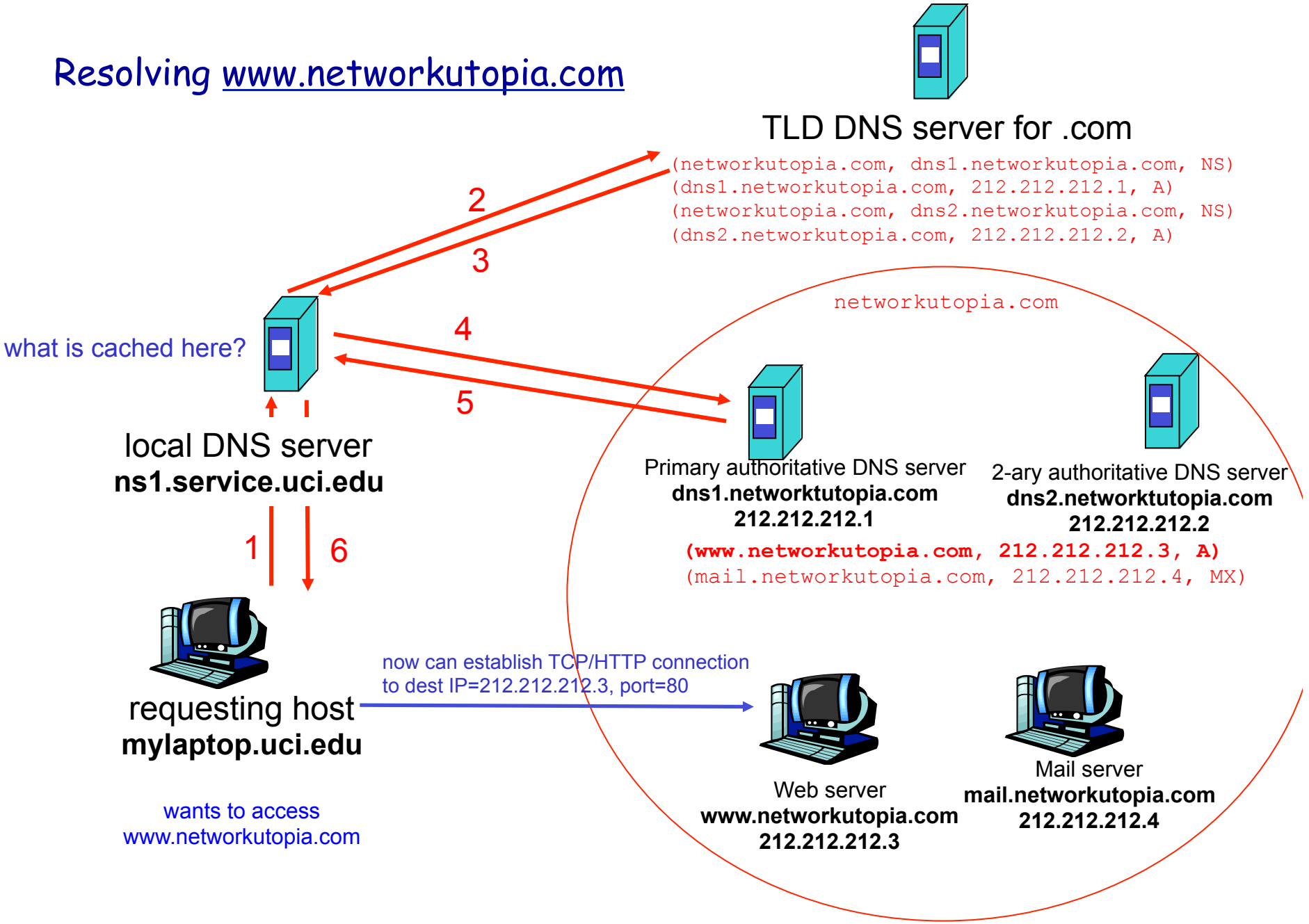
  - Create Type MX record in your own authoritative server

```
(mail.networkutopia.com, 212.212.212.4, MX)
```

Inserting records into DNS



Resolving www.networkutopia.com



Example cont'd: quering DNS records

- Q: How do people visit the website www.networkutopia.com?

- A:
  - Host: sends query to local DNS server
  - Local DNS server: asks TLD server [or root , if TLD not in cache]
  - TLD server: provides A and NS records for dns1.networkutopia.com

    `(networkutopia.com, dns1.networkutopia.com, NS)`
    `(dns1.networkutopia.com, 212.212.212.1, A)`

  - Local DNS server: sends query to authoritative server (212.212.212.1)

  - Authoritative name server: provides type A record

    `(www.networkutopia.com, 212.212.212.3, A)`

  - Local DNS server: returns this info to host

  - Host: establishes TCP/HTTP connection to (IP: 212.212.212.3, port 80)

Q: DNS Load balancing

- DNS may return many RR in same response
  - E.g. dig www.google.com
- Clients:
  - by default, they pick the first one
  - could also choose not to – this part is not standard
- Order of multiple records: Unspecified
  - most often: Round Robin
  - or static or preference to numerically "closer"networks
  - or taking into account load or RTT, or other metric computed by the client or by other (non DNS) servers
- Some references
  - http://en.wikipedia.org/wiki/Domain_Name_System
  - http://en.wikipedia.org/wiki/Round-robin_DNS
  - RFC1794: http://tools.ietf.org/html/rfc1794
  - http://technet.microsoft.com/en-us/library/cc787484(v=ws.10).aspx

DNS Security Vulnerabilities

- ICANN: http://www.icann.org/
- Attacks against root servers (2002, 2007):
  - DNS root servers proved robust (to pings or queries).
  - Thanks to: traffic filtering, caching, anycast load balancing
  - http://www.icann.org/en/announcements/factsheet-dns-attack-08mar07_v1.1.pdf
- DDoS attacks to TLD more dangerous
  - TLD 98% of DNS queries today are invalid
  - http://www.caida.org/publications/papers/2008/root_internet/root_internet.pdf
- Redirect/Man in the middle
  - Cache poisoning: send bogus replies to DNS servers that cache
    - Using DNS to redirect traffic
      - 2008: Kaminsky vulnerability: http://www.unixwiz.net/techtips/iguide-kaminsky-dns-vuln.html
      - 2009: Twitter Blackout: http://www.theregister.co.uk/2009/12/18/dns_twitter_hijack/
  - Intercept queries
    - E.g. to block access to Facebook in China
- Using DNS to launch DDoS attacks
  - Send requests with spoofed source address (target) – responses flood target
  - Requires amplification
- DNSSEC:
  - https://www.icann.org/resources/pages/dnssec-qaa-2014-01-29-en

DNS Summary

- Core Internet functionality
- Implemented as a network application
  - On top of UDP/TCP
  - Defined in RFCs: 1034, 1035 (1987)
  - http://www.ietf.org/rfc/rfc1035.txt
  - Proposed by Mockapertis (UCI PhD 1982)
    - http://en.wikipedia.org/wiki/Paul_Mockapetris
  - Many extensions, e.g. DNSSEC