



Application Layer: DNS

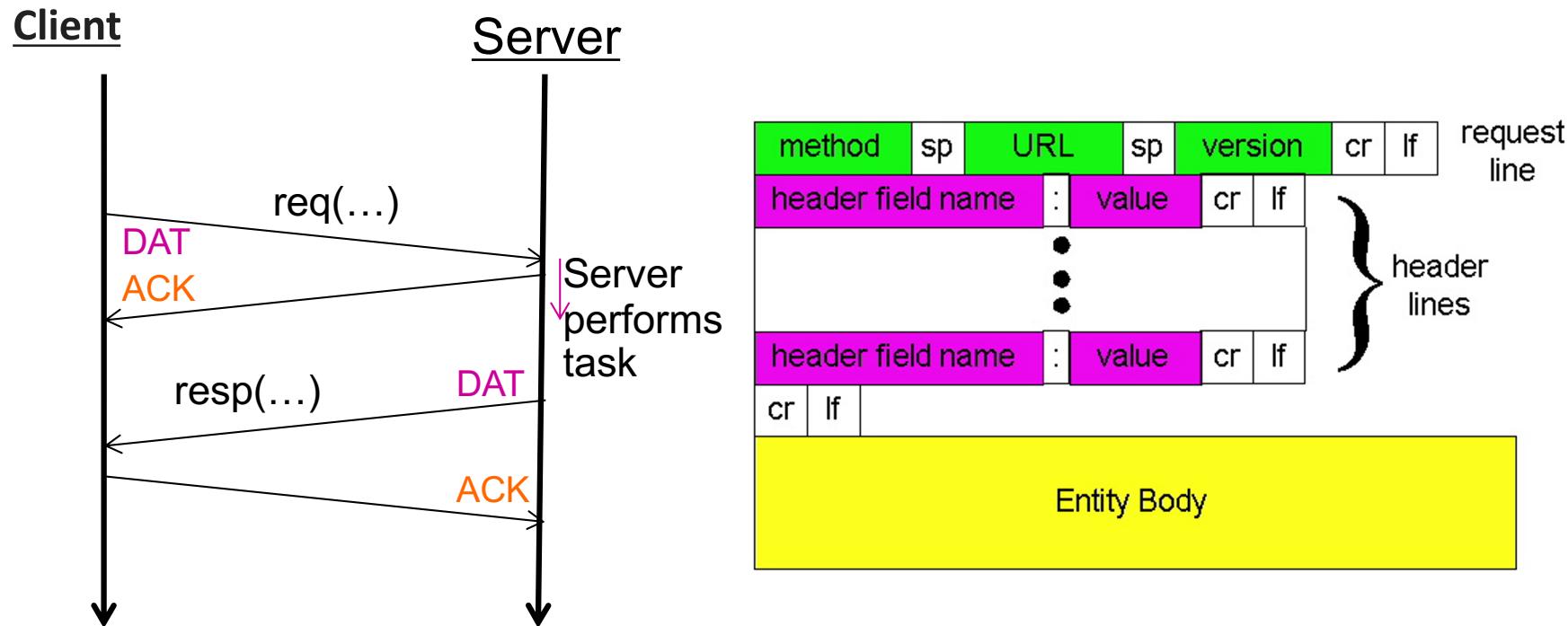
Michael Kirkedal Thomsen

Based on slides compiled by Marcos Vaz Salles, adaptions by Vivek Shah and Michael Kirkedal Thomsen

Recap: Protocol for A6

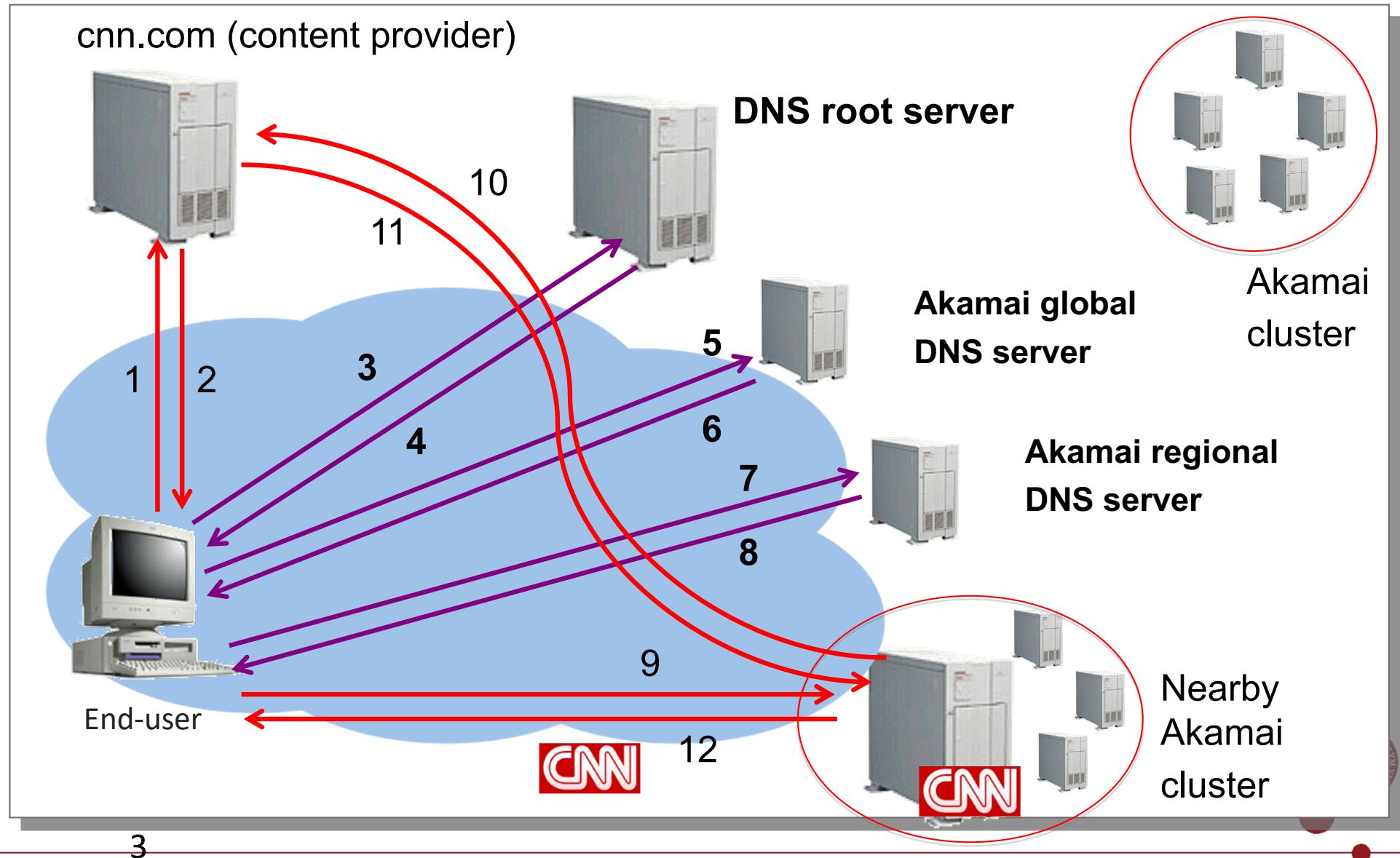
What defines a protocol?

- Syntax: Message format(s)
- Semantics: Set of rules defining message exchange



Recap: How Akamai Works

Source: Freedman (partial)



Recap: Hierarchical Names

- **Host name:** www.cs.princeton.edu
- **Domain:** registrar for each top-level domain (e.g., .edu)
- **Host name:** local administrator assigns to each host
- **IP addresses:** 128.112.7.156
- **Prefixes:** ICANN, regional Internet registries, and ISPs
- **Hosts:** static configuration, or dynamic using DHCP
(more on DHCP later in the course ☺)



Separating Names and IP Addresses

- Names are easier (for us!) to remember
 - www.cnn.com vs. 64.236.16.20
- IP addresses can change underneath
 - Move www.cnn.com to 173.15.201.39
 - 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

Source: Freedman



Outline: Domain Name System

- 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
- DNS software components
- DNS resolvers
- DNS servers
- DNS queries
- Iterative queries
- Recursive queries
- DNS caching based on time-to-live (TTL)

Source: Freedman



Strawman Solution: Central Server

- All you need is to map names!
 - Central server
 - One place where all mappings are stored
 - All queries go to the central server
-
- Is this a good solution?
 - What would be the potential drawbacks?



Source: Freedman (partial)

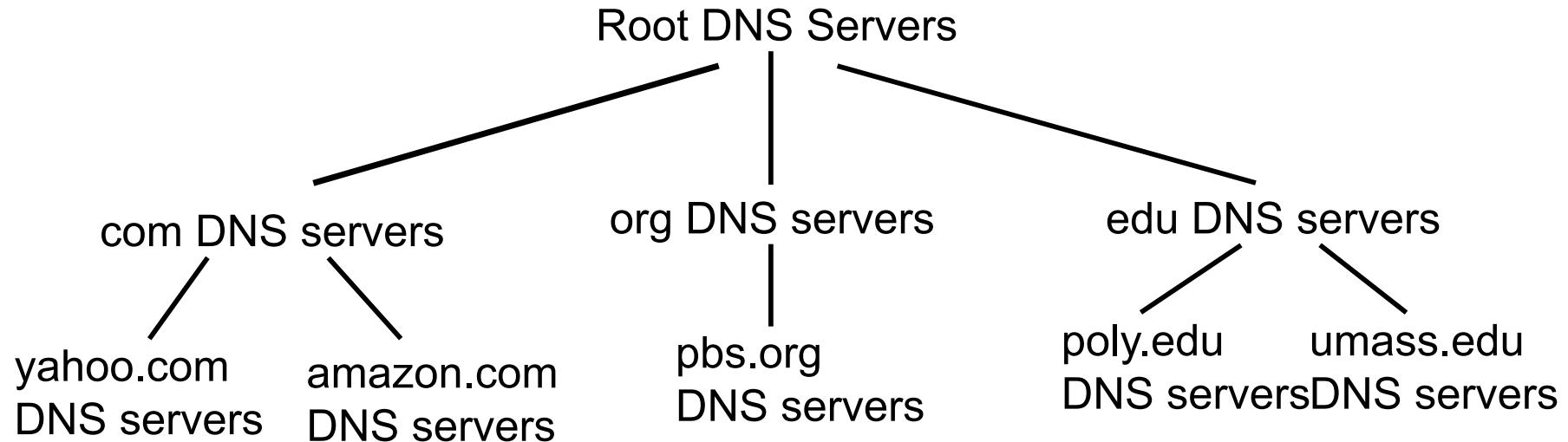
Domain Name System (DNS)

- 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

Source: Freedman



Distributed, Hierarchical Database



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

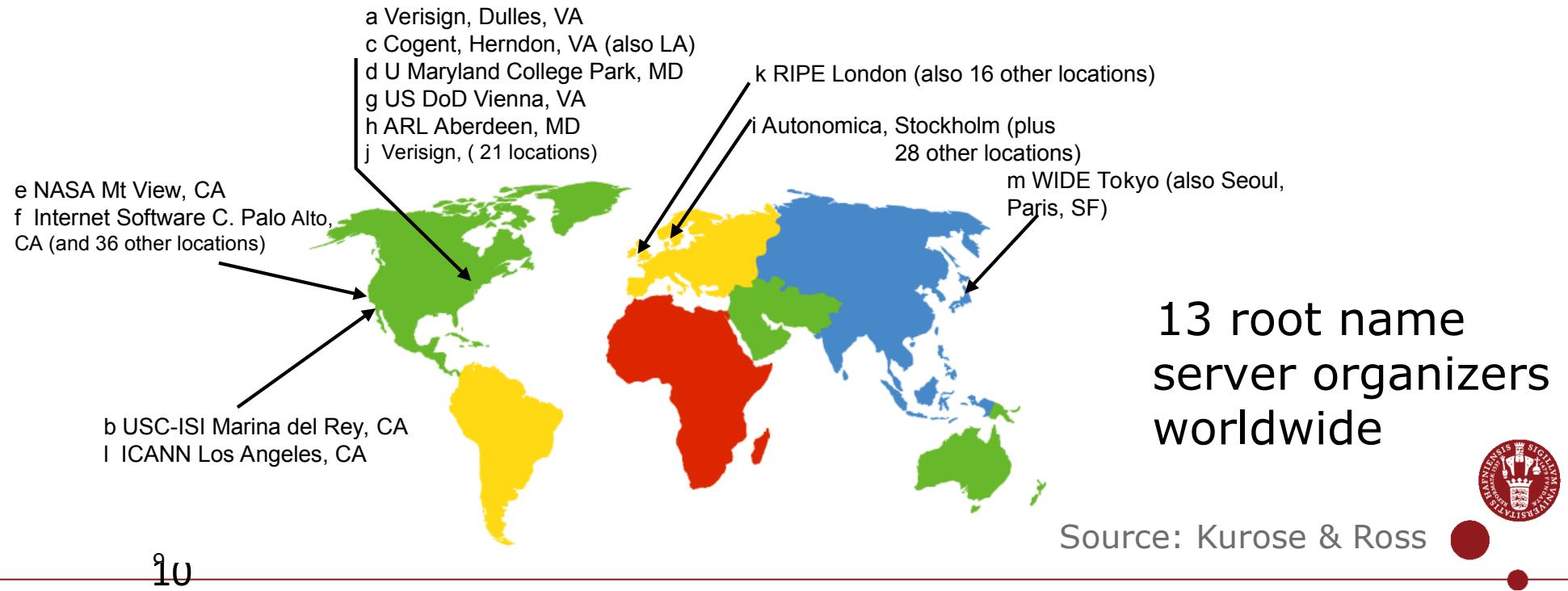
- client queries a root server to find com DNS server
- client queries com DNS server to get amazon.com DNS server
- client queries amazon.com DNS server to get IP address for www.amazon.com



Source: Kurose & Ross

DNS: Root name servers

- contacted by local name server that can not resolve name
- root name server

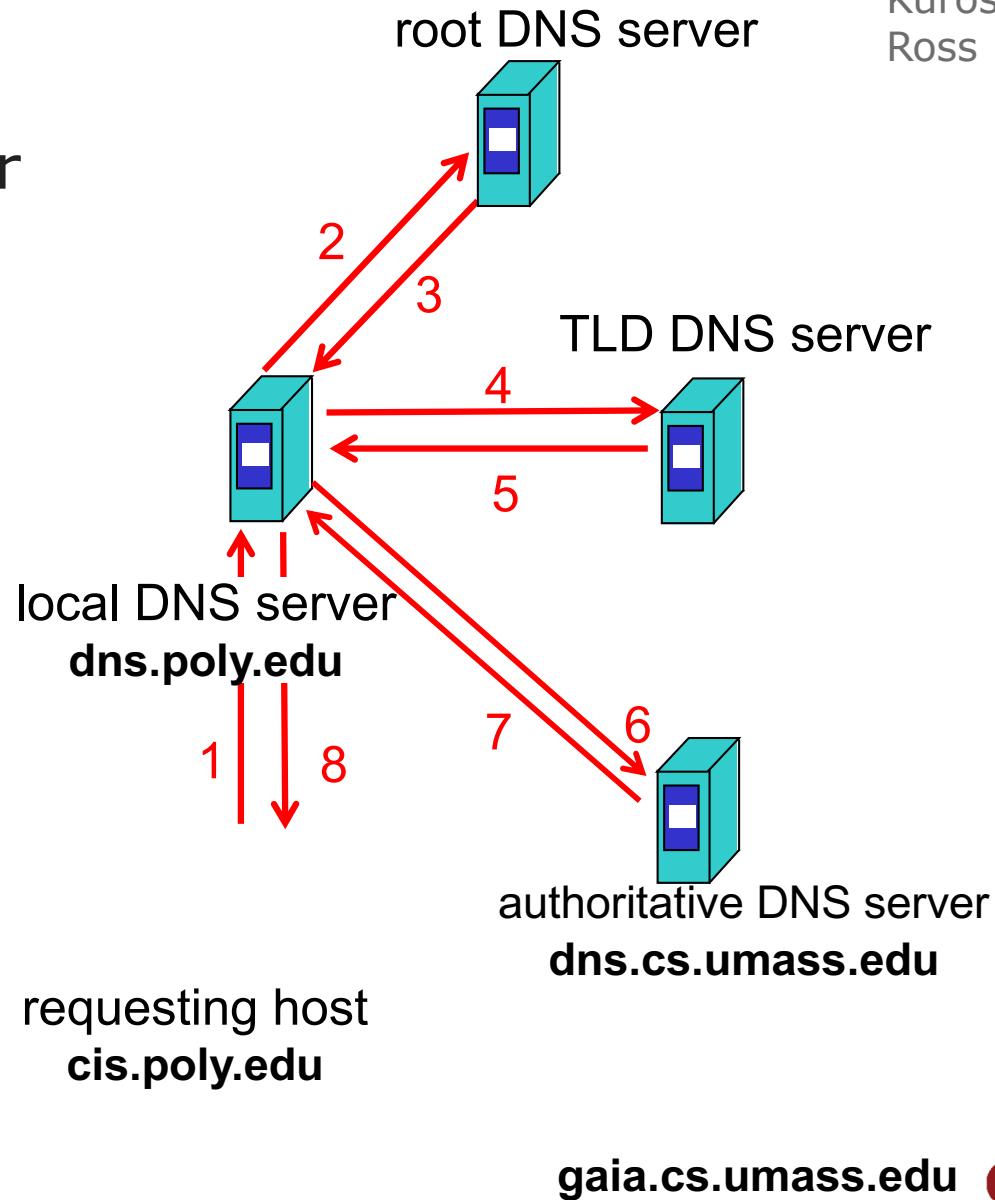


DNS Name Resolution Example

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

Iterated query

- contacted server replies with name of server to contact
- “I don’t know this name, but ask this server”



Source:
Kurose &
Ross

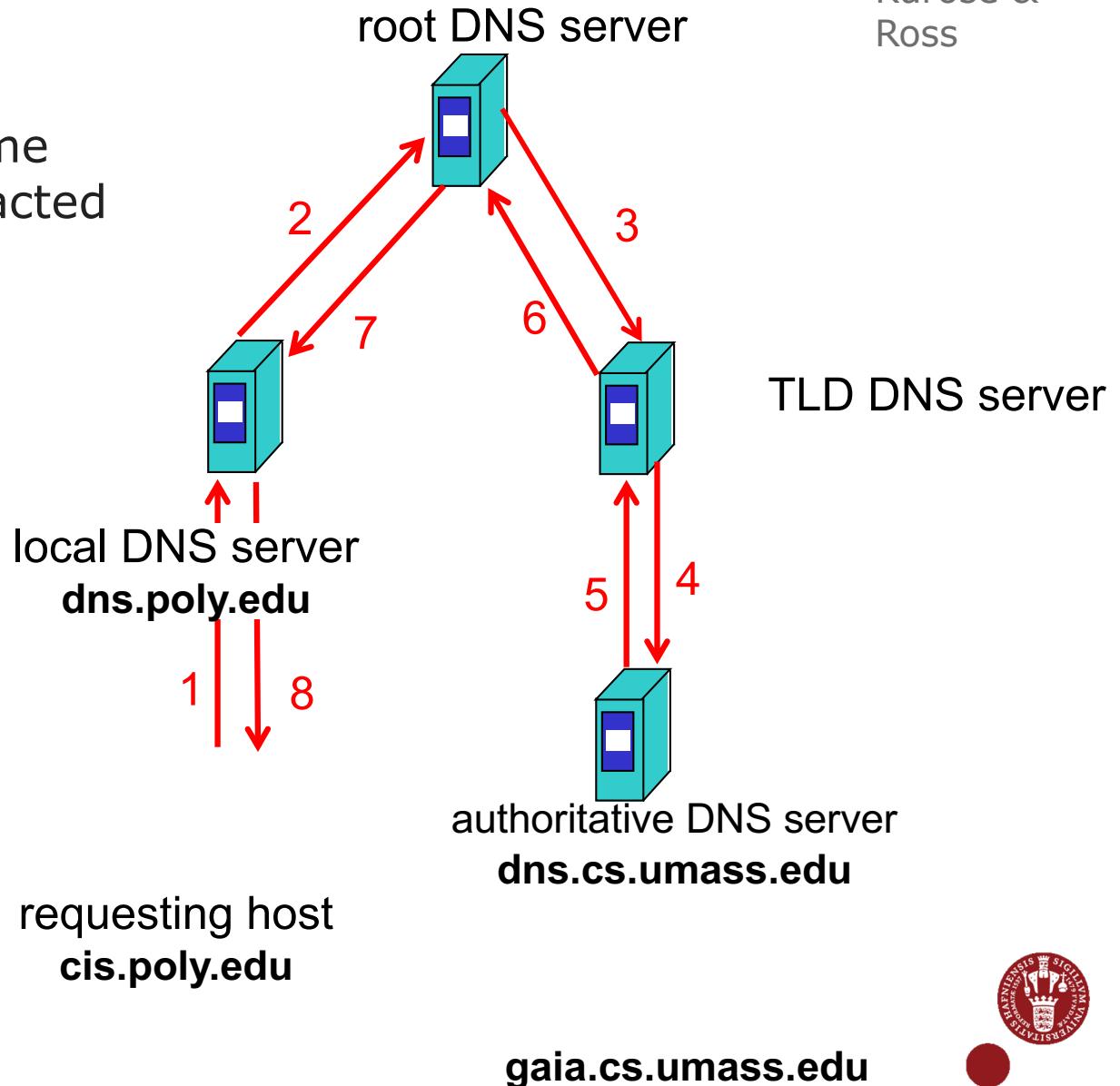


DNS Name Resolution Example

Source:
Kurose &
Ross

Recursive query

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



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) visited often
 - Local DNS server often has the information cached

Source: Freedman (partial)



Time to Live & Negative Caching

- 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

Source: Freedman (partial)



DNS 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=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

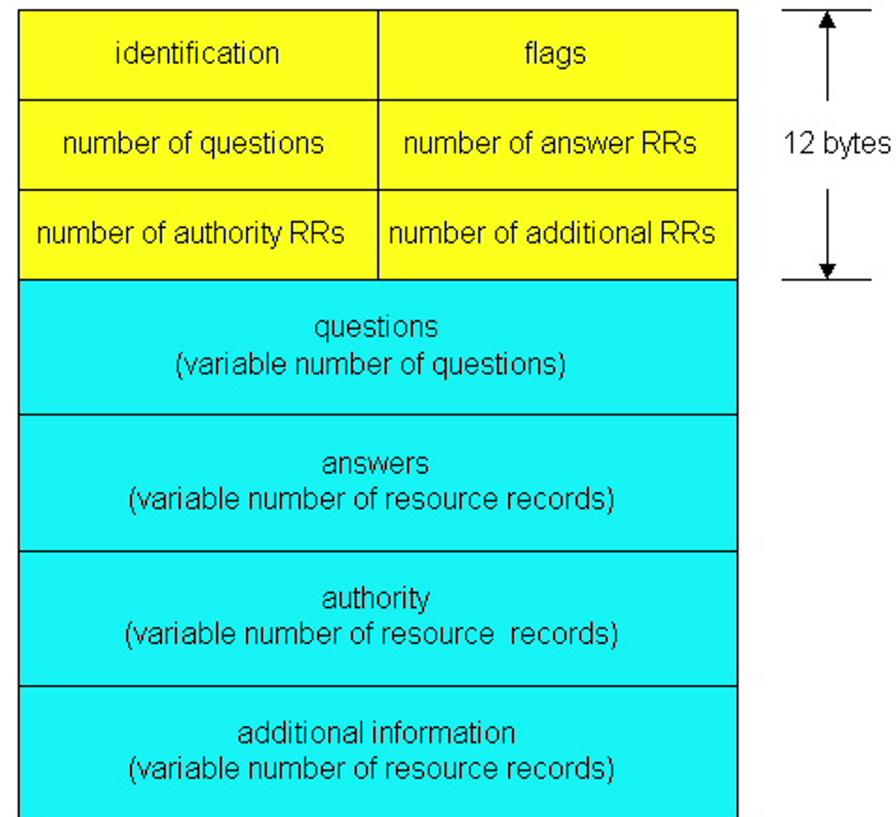
- **value** is name of mailserver associated with **name**



DNS protocol, 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



Inserting records into DNS

- example: new startup “Network Utopia”
- register name `networkuptopia.com` at *DNS registrar* (e.g., Network Solutions)
- provide names, IP addresses of authoritative name server (primary and secondary)
- registrar inserts two RRs into com TLD server:
 - (`networkutopia.com`, `dns1.networkutopia.com`, NS)
 - (`dns1.networkutopia.com`, `212.212.212.1`, A)
- create authoritative server Type A record for `www.networkuptopia.com`; Type MX record for `networkutopia.com`
- **How do people get IP address of your Web site?**



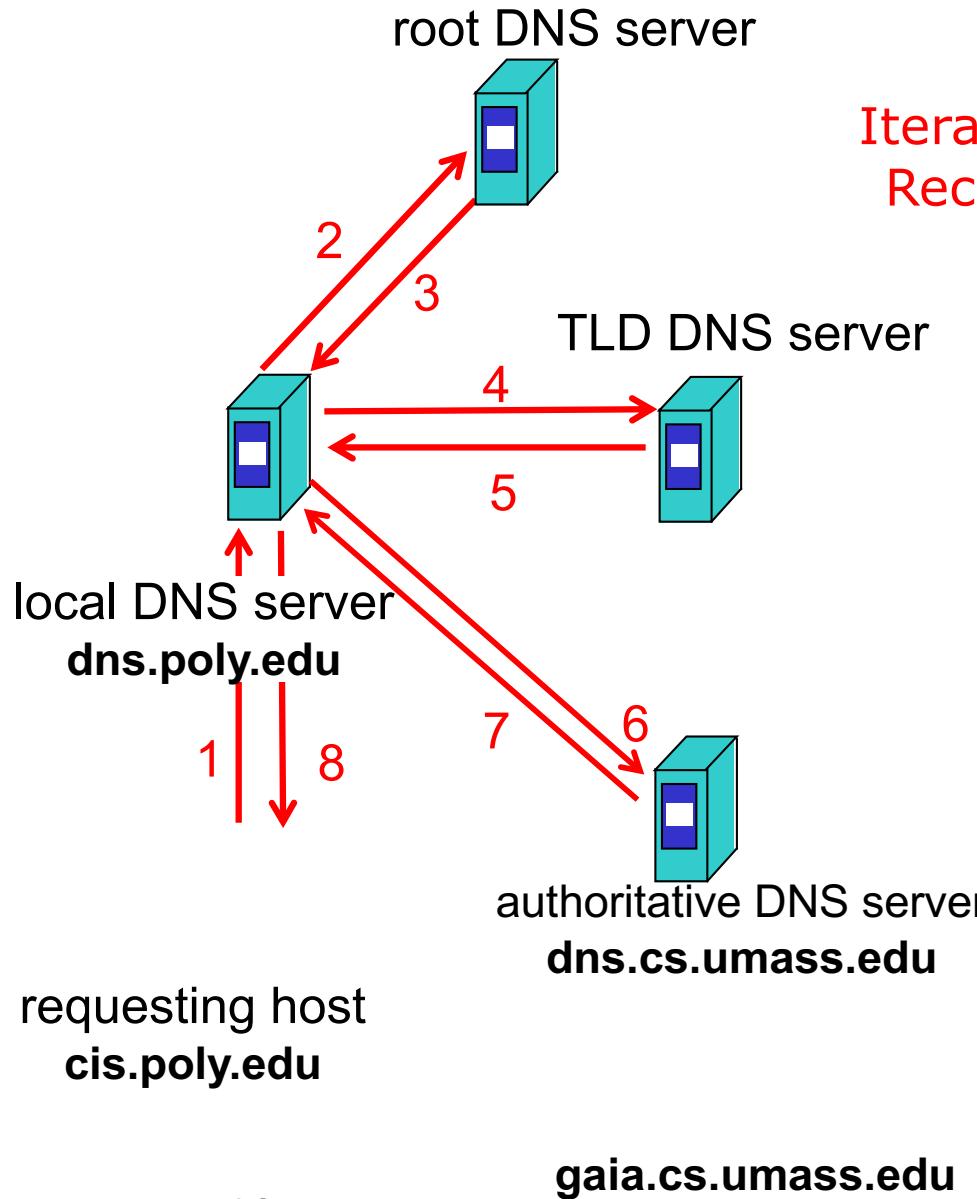
DNS security

- DNS cache poisoning
 - Ask for www.evil.com
 - Additional section for (www.cnn.com, 1.2.3.4, A)
 - Thanks! I won't bother check what I asked for
- DNS hijacking
 - Let's remember the domain. And the UDP ID (source port + transaction ID).
 - 16 bits: 65K possible transaction IDs
 - What rate to enumerate all in 1 sec? 64B/packet
 - $64 * 65536 * 8 / 1024 / 1024 = 32 \text{ Mbps}$
 - Prevention: Also randomize the DNS source port
 - E.g., Windows DNS alloc's 2500 DNS ports: ~164M possible IDs
 - Would require 80 Gbps
 - Kaminsky attack: this source port...wasn't random after all

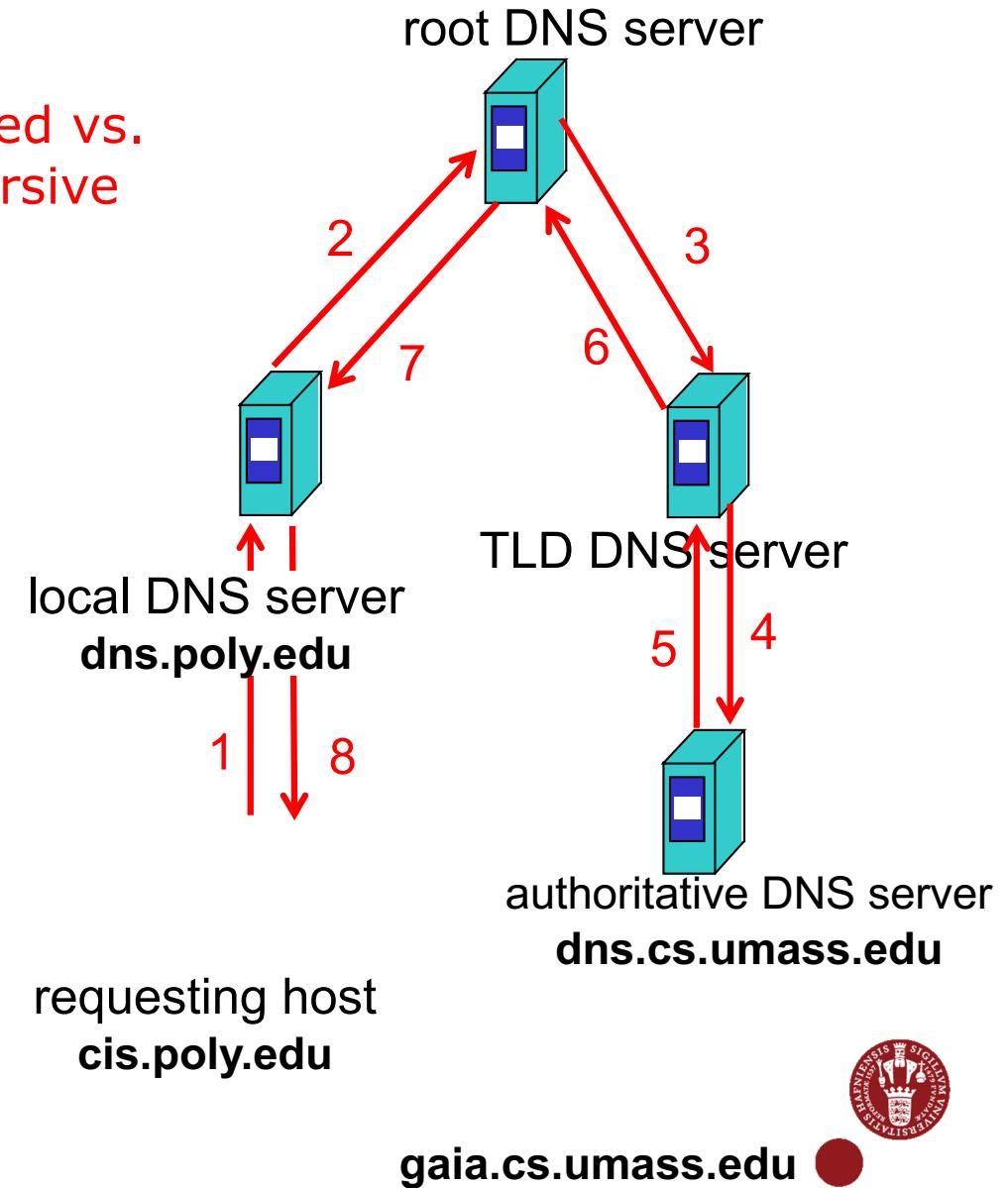
Source: Freedman (partial)



How does caching affect each variant? In which one do you expect caching to work best?



Iterated vs.
Recursive



Use dig and nslookup with what you learned today

```
bonii@Bigbang@14:21:50 ~ $nslookup www.diku.dk
```

```
Server: 192.38.118.220  
Address: 192.38.118.220#53
```

Non-authoritative answer:

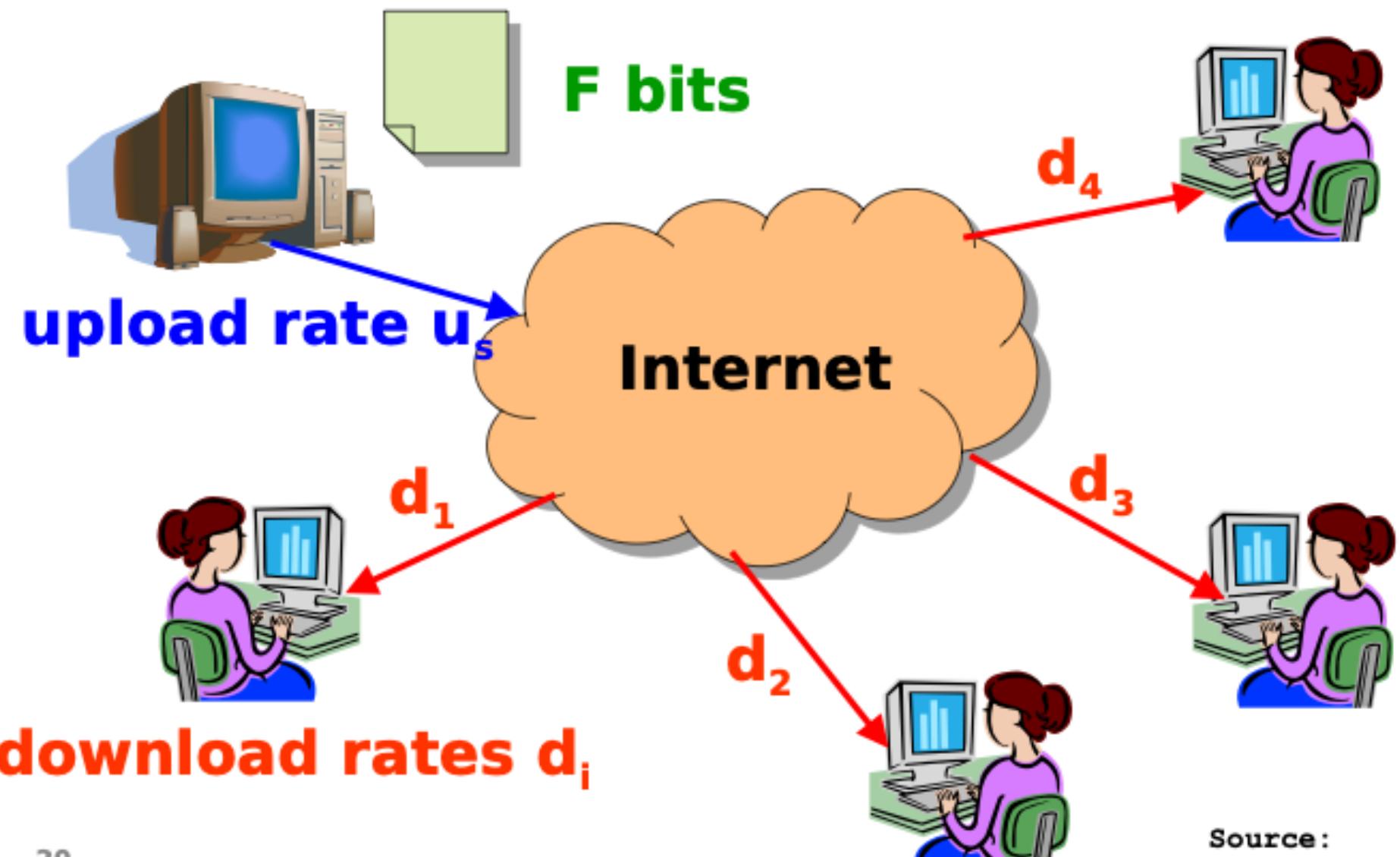
```
www.diku.dk canonical name = web-aggregator.diku.dk.  
Name: web-aggregator.diku.dk  
Address: 130.226.14.83
```

```
bonii@Bigbang@14:25:04 ~ $dig +trace www.diku.dk
```

```
; <>> DiG 9.10.4-P3 <>> +trace www.diku.dk  
;; global options: +cmd  
. 185772 IN NS g.root-servers.net.  
. 185772 IN NS e.root-servers.net.  
. 185772 IN NS c.root-servers.net.  
. 185772 IN NS k.root-servers.net.  
. 185772 IN NS d.root-servers.net.  
. 185772 IN NS l.root-servers.net.  
. 185772 IN NS m.root-servers.net.  
. 185772 IN NS a.root-servers.net.  
. 185772 IN NS i.root-servers.net.  
. 185772 IN NS h.root-servers.net.  
. 185772 IN NS f.root-servers.net.  
. 185772 IN NS j.root-servers.net.  
. 185772 IN NS b.root-servers.net.  
;; Received 824 bytes from 192.38.118.220#53(192.38.118.220) in 0 ms
```



Server Distributing a Large File



Server Distributing a Large File

- Sending an F -bit file to N receivers
 - Transmitting NF bits at rate u_s
 - ... takes at least NF/u_s time
- Receiving the data at the slowest receiver
 - Slowest receiver has download rate $d_{min} = \min_i\{d_i\}$
 - ... takes at least F/d_{min} time
- Download time: $\max\{NF/u_s, F/d_{min}\}$



Speeding Up the File Distribution

- Increase the server upload rate
 - Higher link bandwidth at the server
 - Multiple servers, each with their own link
- Alternative: have the receivers help
 - Receivers get a copy of the data
 - ... and redistribute to other receivers
 - To reduce the burden on the server

Source:
Freedman



Summary

- DNS
 - Hierarchical names
 - Recursive vs. iterative name resolution, caching
 - Addresses, aliases, resource records

