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 file sharing
- 2.7 Socket programming with TCP
- 2.8 Socket programming with UDP
- 2.9 Building a Web server

DNS: Domain Name System

- Imagine a world without DNS

- You would have to remember the IP addresses of
  - Every website you want to visit
  - Your bookmarks will be a list of IP addresses

  - You will speak like
    "I went to 167.33.24.10, and there was an awesome link to 153.11.35.81… "

DNS: Domain Name System

People: many identifiers:

- SSN, name, passport #

Internet hosts, routers:

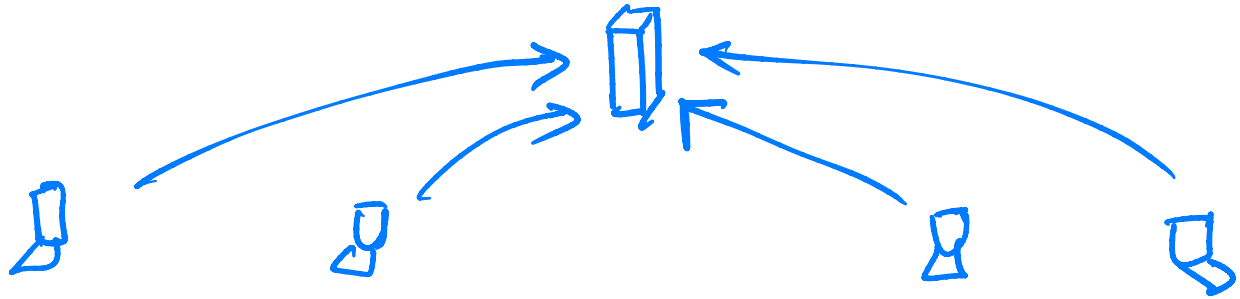
- IP address (32 bit) - used for addressing datagrams
- "name", e.g., ww.yahoo.com - used by humans

Q: map between IP addresses and name ?

Domain Name System:

- ***distributed database*** implemented in hierarchy of many *name servers*
- *application-layer protocol* host, routers, name servers to communicate to *resolve* names (address/name translation)
  - note: core Internet function, implemented as application-layer protocol
  - complexity at network's "edge"

DNS



DNS services

- Hostname to IP address translation
- Host aliasing
  - Canonical and alias names

tokyo.ibm.com ibm.com

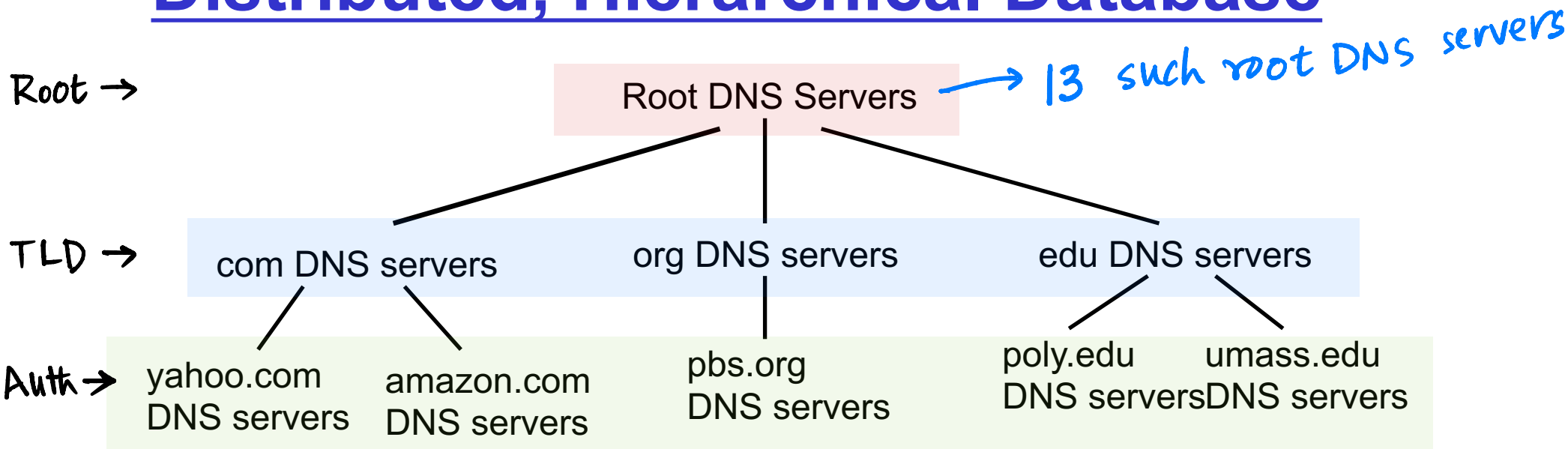
- Load distribution
  - Replicated Web servers: set of IP addresses for one canonical name

Why not centralize DNS?

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

doesn't *scale!*

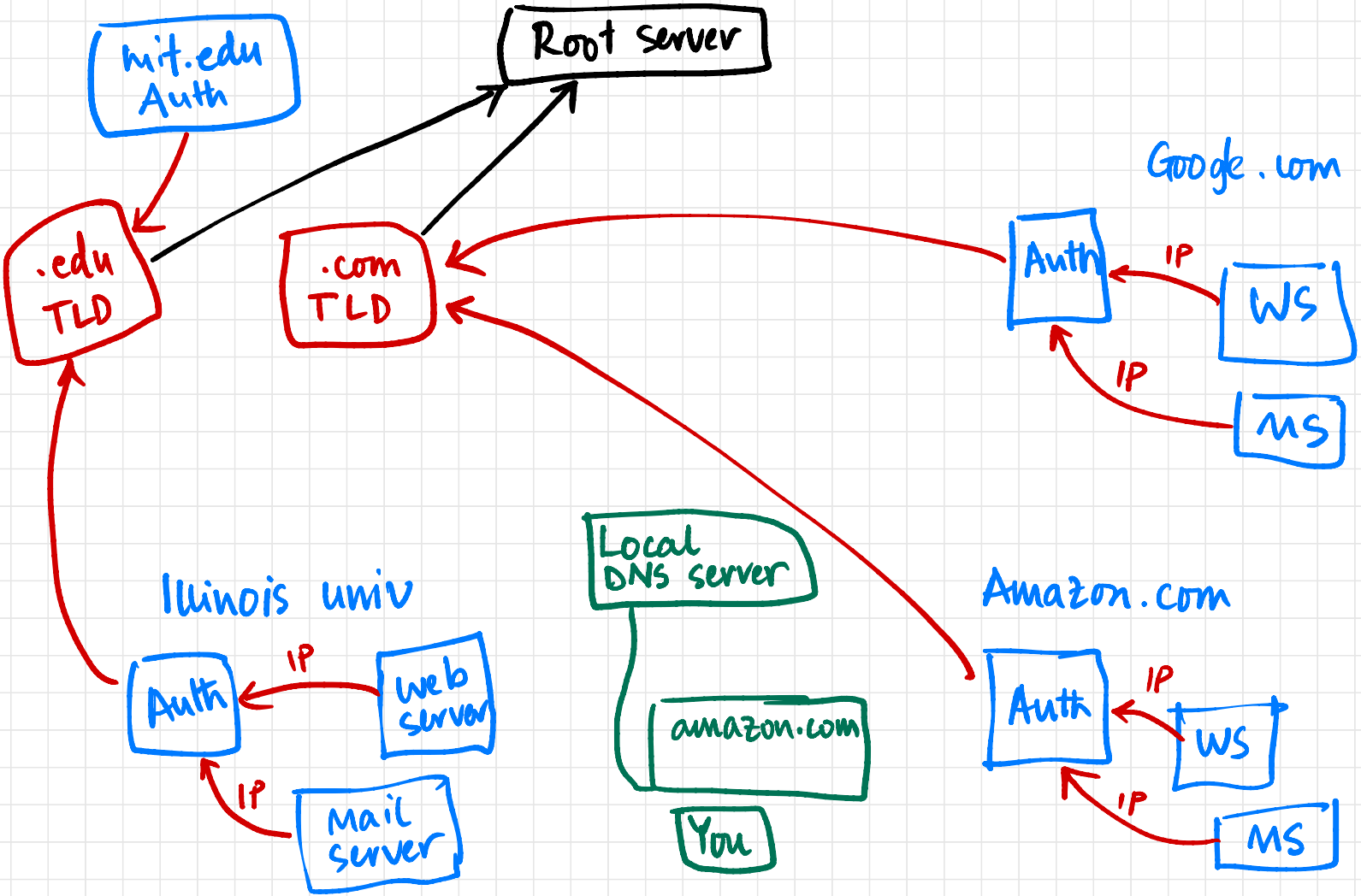
Distributed, Hierarchical Database



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 web server

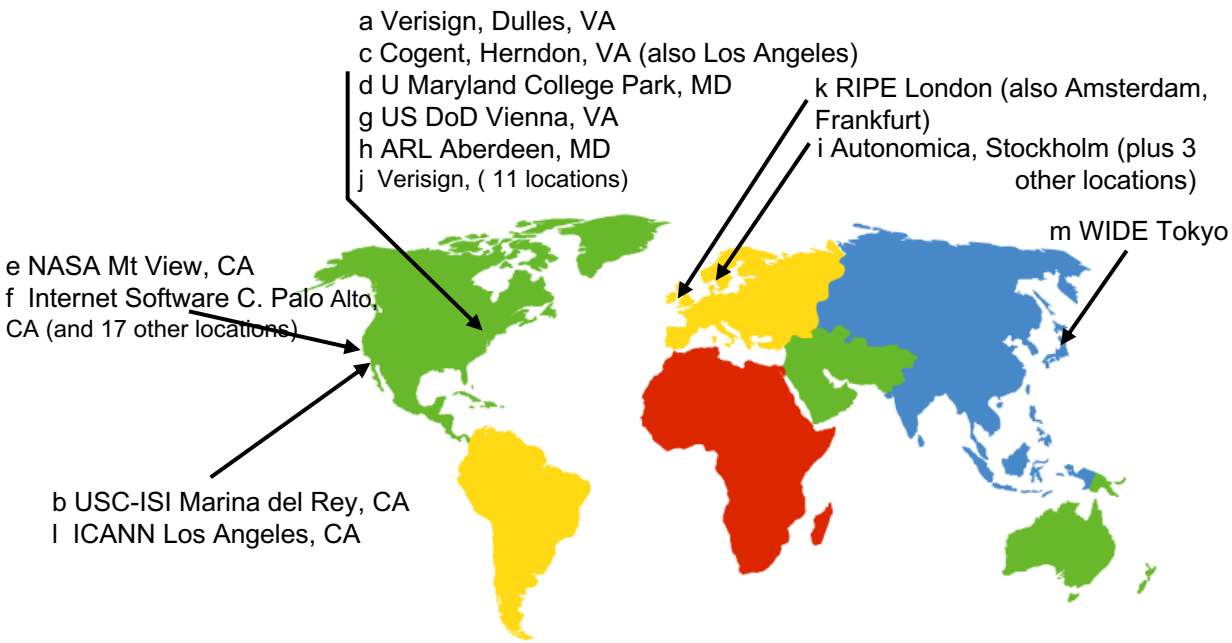
- Client queries same Amazon.com DNS server to get IP address of mail.amazon.com mail server.


Root server
mit.edu Auth
.edu TLD
.com TLD
Google.com
Auth
IP
WS
IP
MS
Local DNS server
amazon.com
You
Illinois univ
Auth
IP
web server
IP
Mail server
Amazon.com
Auth
IP
WS
IP
MS

DNS: Root name servers

- contacted by local name server that can not resolve name
- root name server:
  - contacts authoritative name server if name mapping not known
  - gets mapping
  - returns mapping to local name server

ask the boss: root of the tree



13 root name servers worldwide

TLD and Authoritative Servers

- Top-level domain (TLD) servers:
  - responsible for com, org, net, edu, etc.
  - all top-level country domains uk, fr, ca, jp.
  - Network solutions maintains servers for com TLD
  - Educause for edu TLD

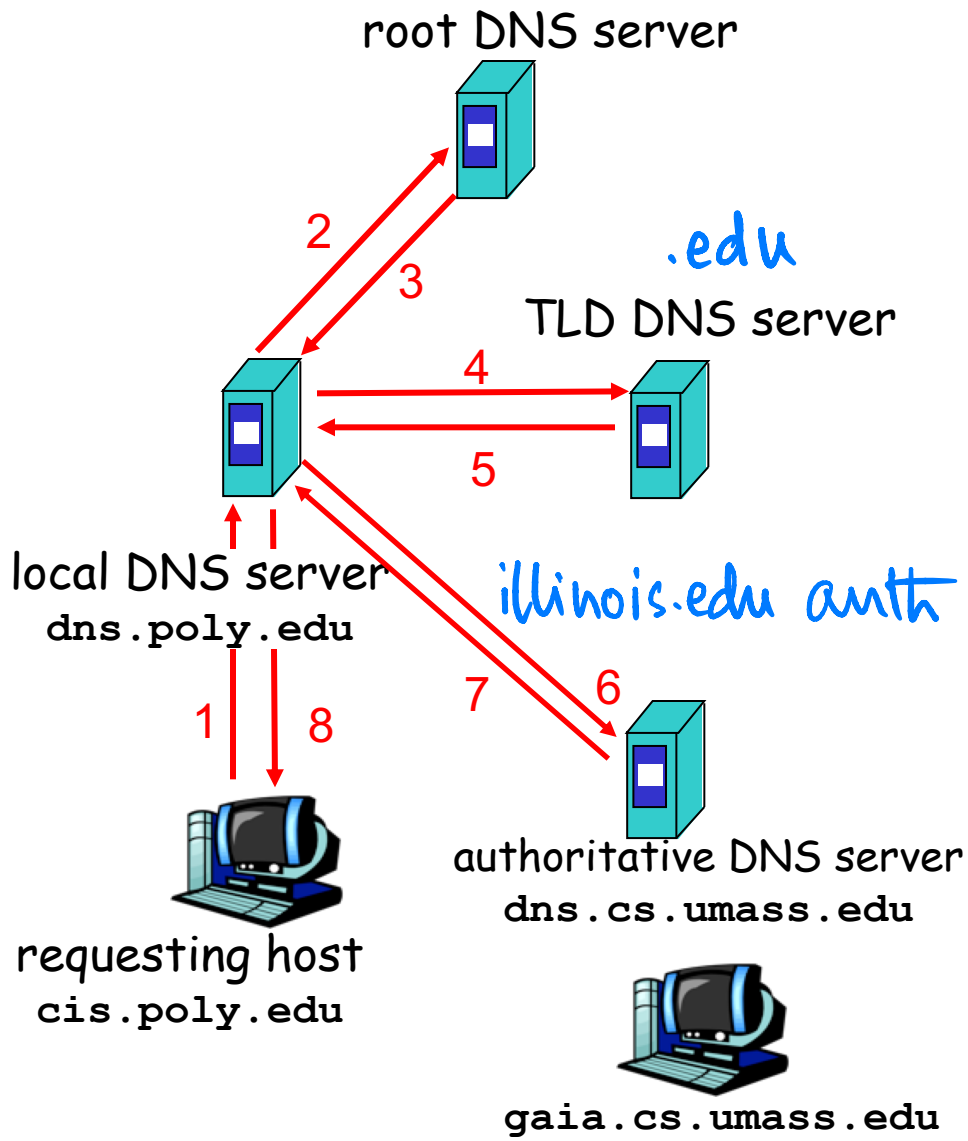
- Authoritative DNS servers:
  - An organization's DNS servers,
    - providing authoritative hostname to IP mappings for organization's servers (e.g., Web and mail).
  - Can be maintained by organization or service provider

Local Name Server

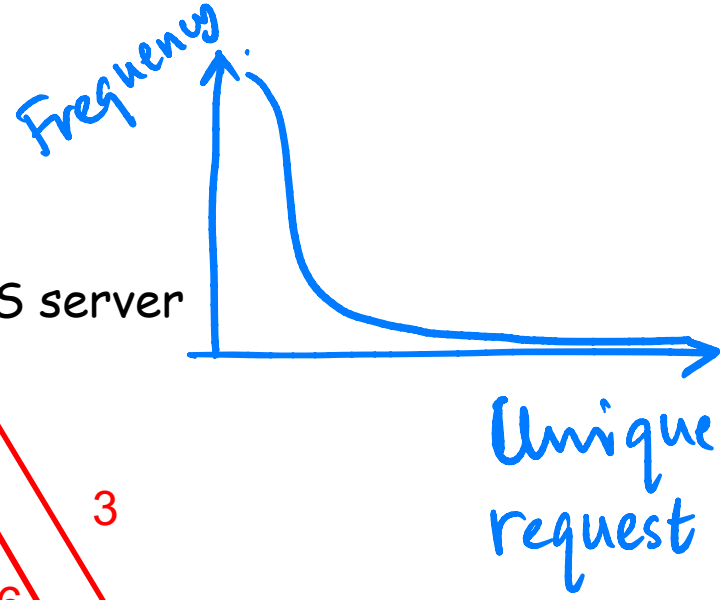
- Does not strictly belong to hierarchy
- Each ISP (residential, company, univ) has one.
  - Also called "default name server"
- When a host makes a DNS query
  - query is sent to its local DNS server
  - Acts as a proxy, forwards query into hierarchy.

Example

- **Iterative Querying**
  Host at cis.poly.edu wants IP address for gaia.cs.umass.edu



Recursive queries

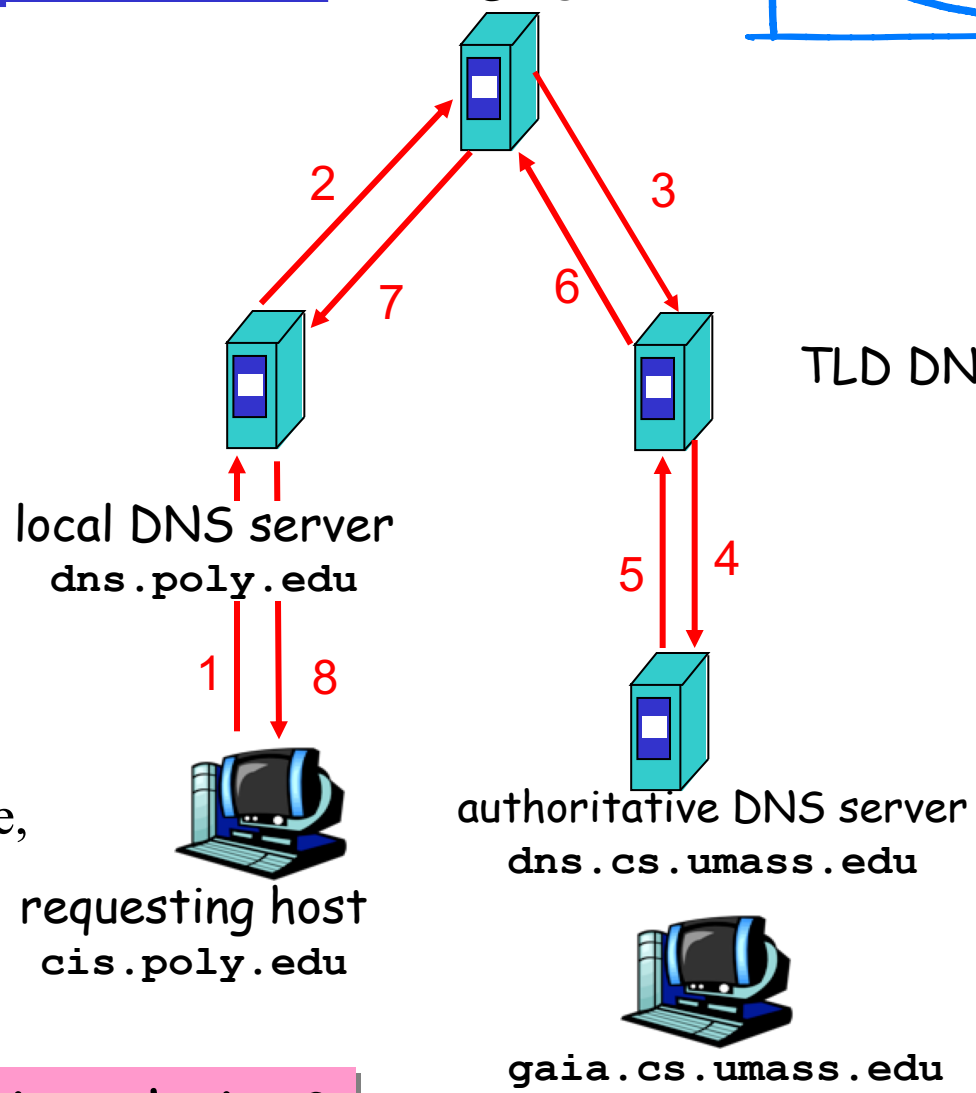


recursive query:

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

iterated query:

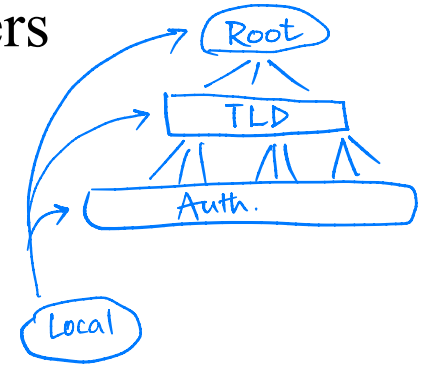
- contacted server replies with name of server to contact
- "I don't know this name, but ask this server"



Which is a better design choice?

DNS: caching and updating records

- 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



- Update/notify mechanisms under design by IETF
  - RFC 2136
  - http://www.ietf.org/html.charters/dnsind-charter.html

Auth → (cnn.com, 166.34.33.11, A, 10)

TLD → (cnn.com, 166.34.23.18, NS, 10)

DNS records

4 categories:
A
NS
CNAME
MX

DNS: distributed db storing resource records (RR)

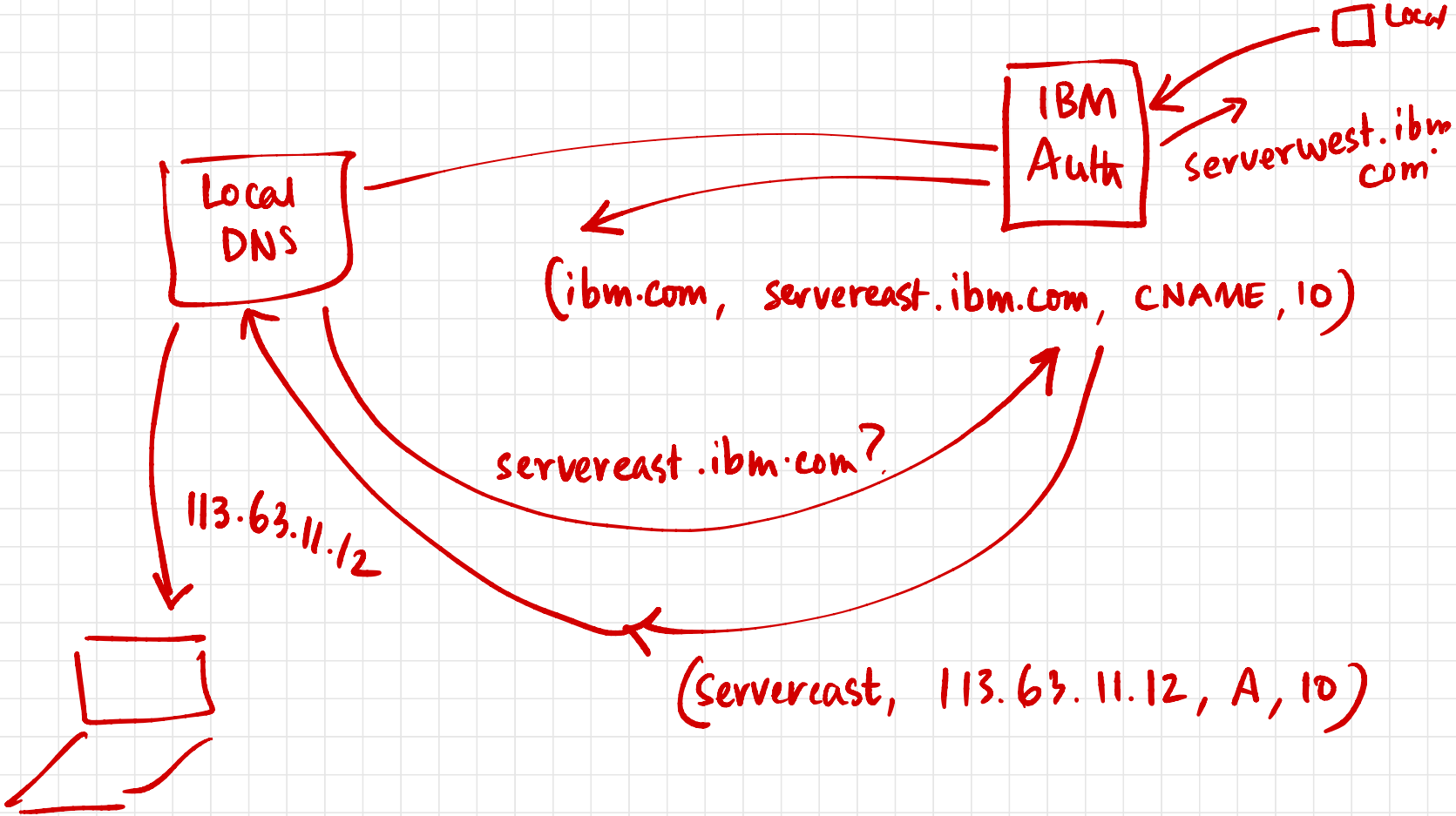
RR format: `(name, value, type, ttl)`

time to live

- 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**

alias

cannonical

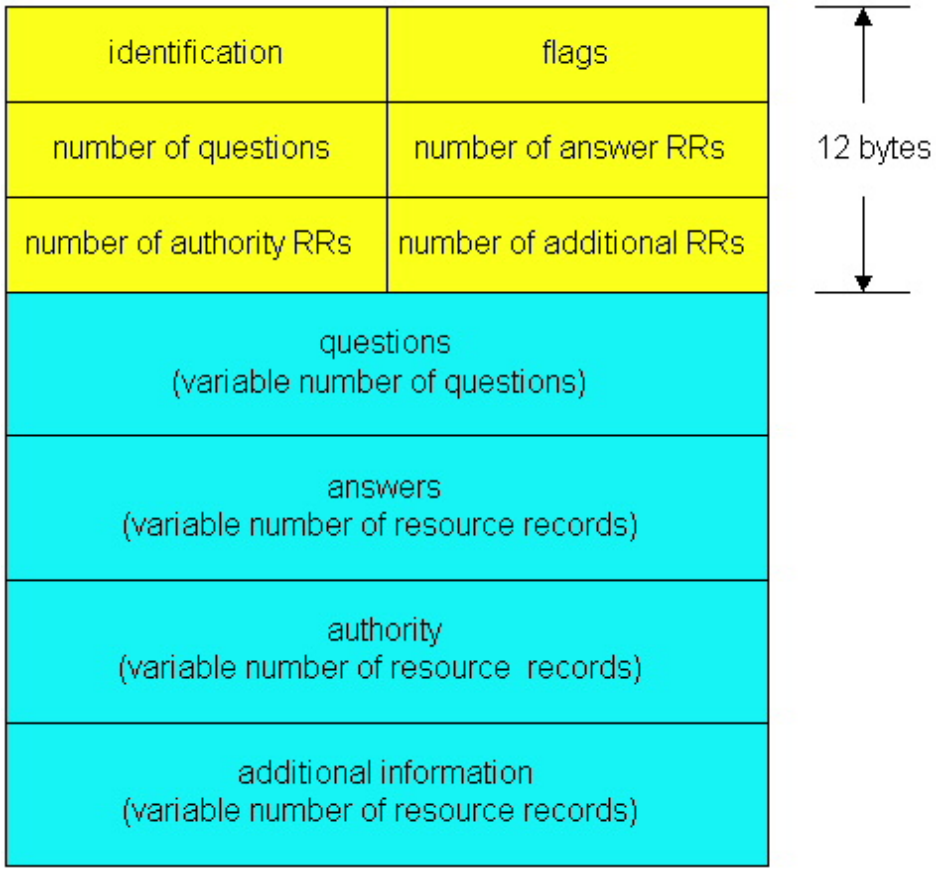

Local
IBM Auth
serverwest.ibm.com
Local DNS
(ibm.com, servereast.ibm.com, CNAME, 10)
servereast.ibm.com?
113.63.11.12
(servereast, 113.63.11.12, A, 10)

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



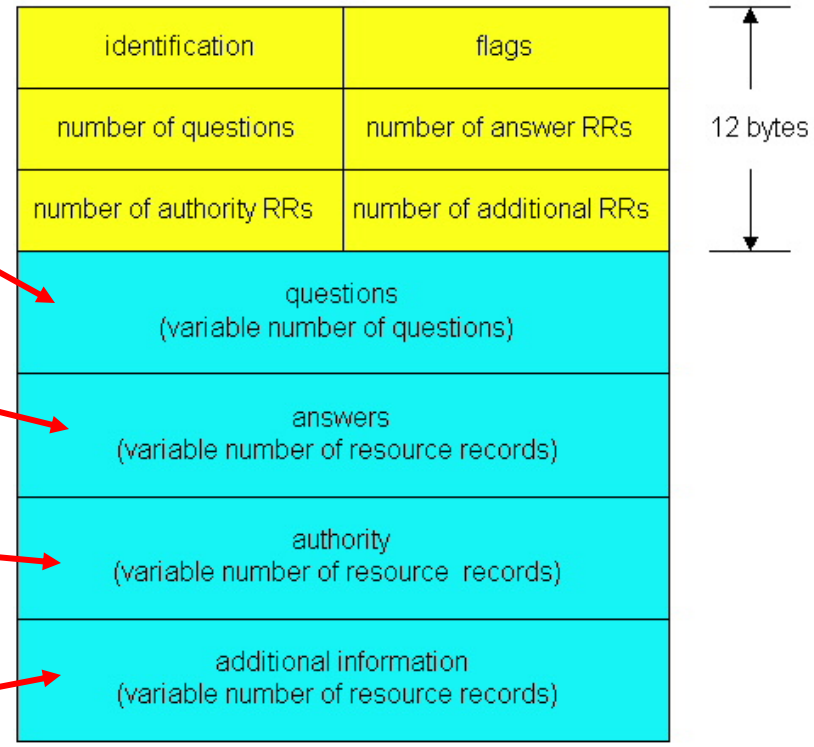
DNS protocol, messages

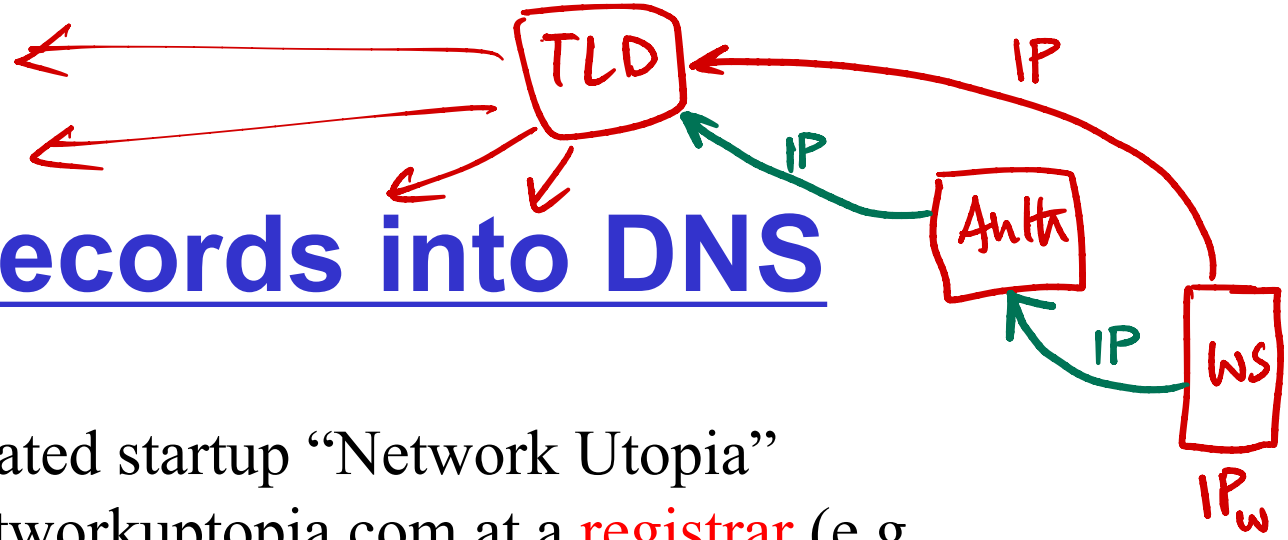
Name, type fields for a query

RRs in response to query

records for authoritative servers

additional "helpful" info that may be used





Inserting records into DNS

- Example: just created startup "Network Utopia"
- Register name networkuptopia.com at a registrar (e.g., Network Solutions)
  - Need to provide registrar with names and IP addresses of your authoritative name server (primary and secondary)
  - Registrar inserts two RRs into the com TLD server:

nameserver

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

(networkuto.com, 212.212.212.1, A)

- Also, in the startup's Auth server, put Type A record for www.networkuptopia.com and Type MX record for networkutopia.com
- How do people get the IP address of your Web site?

Questions ?

Chapter 2: Application layer

- 2.1 Principles of network applications
  - app architectures
  - app requirements
- 2.2 Web and HTTP
- 2.4 Electronic Mail
  - SMTP, POP3, IMAP
- 2.5 DNS
- 2.6 P2P file sharing [Napster, Gnutella, Kazaa]
- 2.7 Socket programming with TCP
- 2.8 Socket programming with UDP
- 2.9 Building a Web server

P2P file sharing

Example

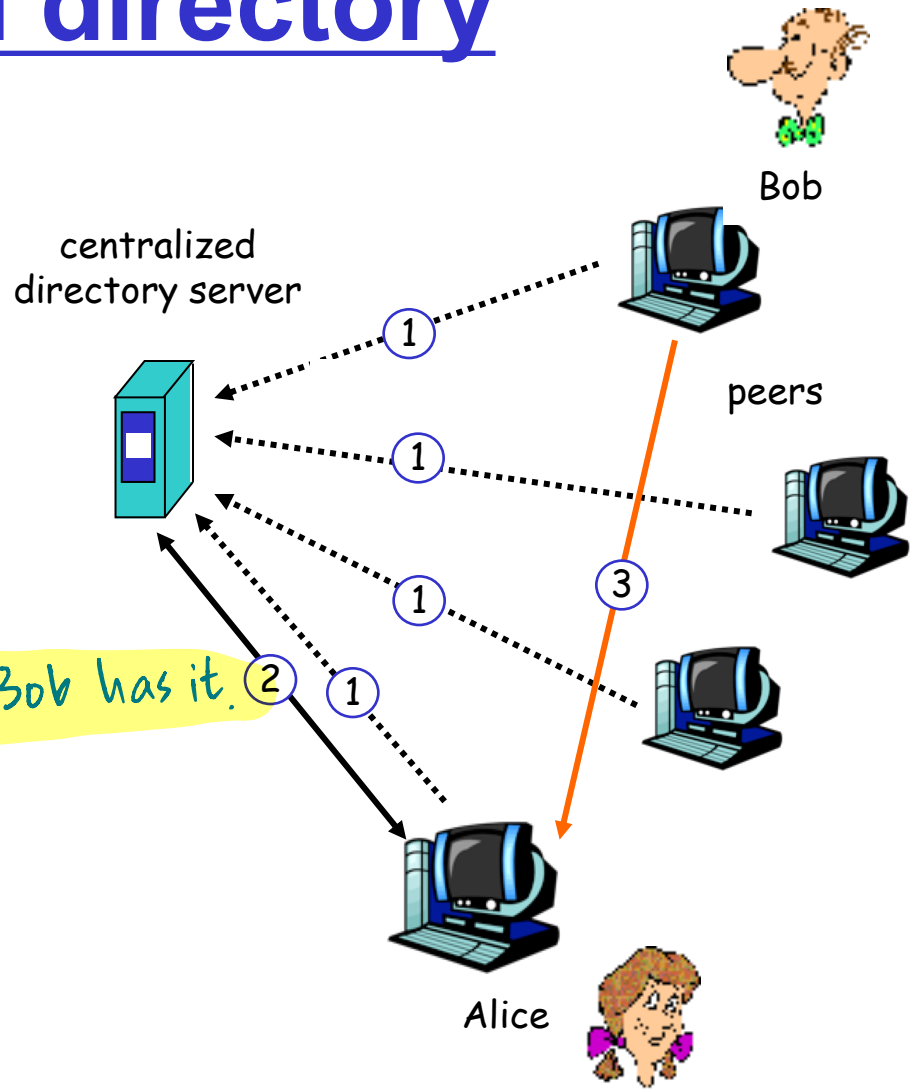
- Alice runs P2P client application on her notebook computer
- Intermittently connects to Internet; gets new IP address for each connection
- Asks for "Hey-Jude.mp3"
- Application displays other peers that have copy of Hey Jude.
- Alice chooses one of the peers, Bob.
- File is copied from Bob's PC to Alice's notebook: HTTP
- While Alice downloads, other users uploading from Alice.
- Alice's peer is both a Web client and a transient Web server.

All peers are servers = highly scalable!

P2P: centralized directory

original "Napster" design

1) when peer connects, it informs central server:
   - IP address
   - content
2) Alice queries for "Hey Jude" . Server informs that Bob has it.
3) Alice requests file from Bob



P2P: problems with centralized directory

- Single point of failure
- Performance bottleneck
- Copyright infringement

file transfer is decentralized, but locating content is highly centralized

Query flooding: Gnutella

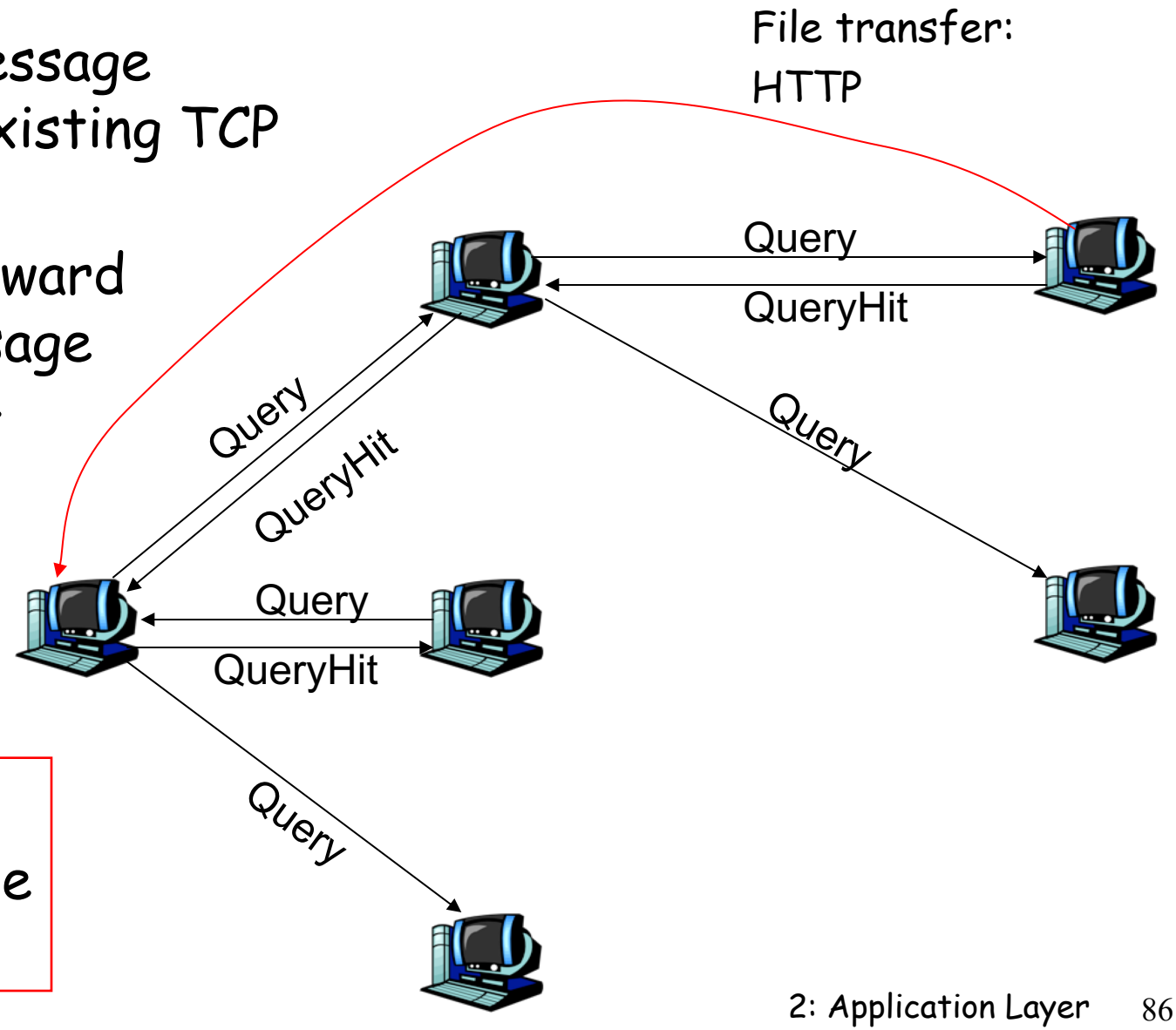
- fully distributed
  - no central server
- public domain protocol
- many Gnutella clients implementing protocol

overlay network: graph

- edge between peer X and Y if there's a TCP connection
- all active peers and edges is overlay net
- Edge is not a physical link
- Given peer will typically be connected with < 10 overlay neighbors

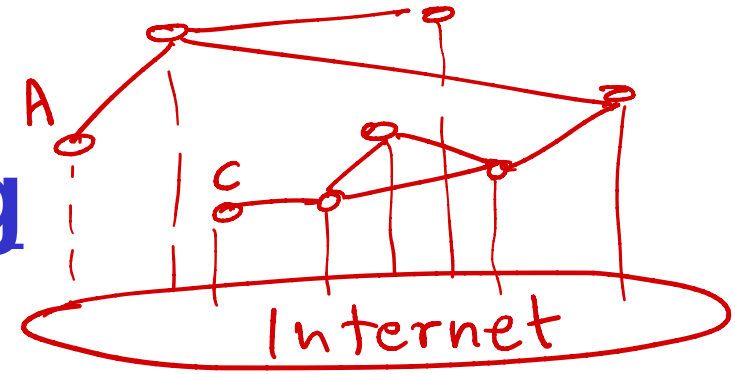
Gnutella: protocol

- Query message sent over existing TCP connections
- peers forward Query message
- QueryHit sent over reverse path

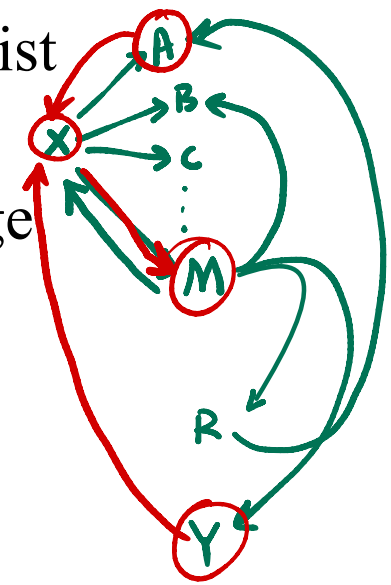


Scalability:
limited scope flooding

Gnutella: Peer joining



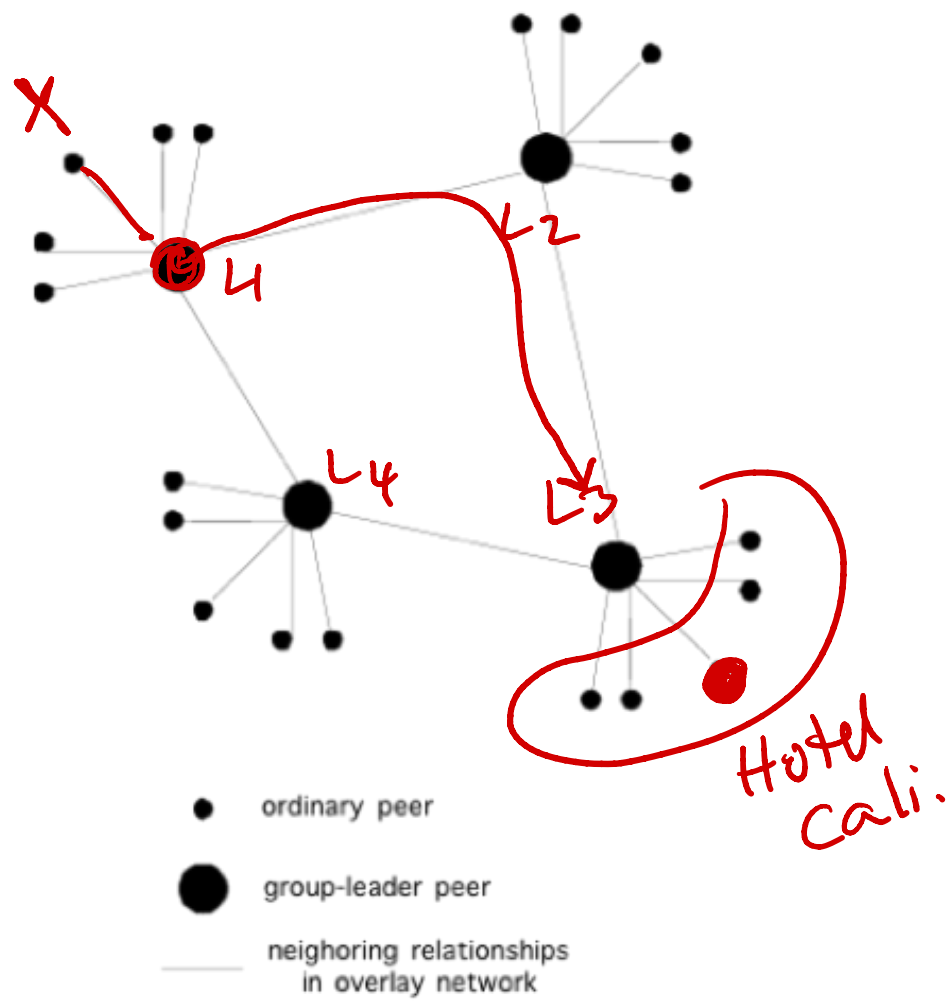
1. Joining peer X must find some other peer in Gnutella network: use list of candidate peers
2. X sequentially attempts to make TCP with peers on list until connection setup with Y
3. X sends Ping message to Y; Y forwards Ping message
4. All peers receiving Ping message respond with Pong message
5. X receives many Pong messages. It can then setup additional TCP connections



What happens when peer leaves: find out as an exercise!

Exploiting heterogeneity: KaZaA

- Each peer is either a group leader or assigned to a group leader.
  - TCP connection between peer and its group leader.
  - TCP connections between some pairs of group leaders.
- Group leader tracks the content in all its children.



KaZaA: Querying

- Each file has a hash and a descriptor
- Client sends keyword query to its group leader
- Group leader responds with matches:
  - For each match: metadata, hash, IP address
- If group leader forwards query to other group leaders, they respond with matches
- Client then selects files for downloading
  - HTTP requests using hash as identifier sent to peers holding desired file

KaZaA tricks

- Limitations on simultaneous uploads
- Request queuing
- Incentive priorities
- Parallel downloading

For more info:

- J. Liang, R. Kumar, K. Ross, "Understanding KaZaA," (available via cis.poly.edu/~ross)

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 file sharing
- 2.7 Socket programming with TCP
- 2.8 Socket programming with UDP
- 2.9 Building a Web server