DNS: Domain Name System

- Internet communication requires IP addresses
- Humans prefer to use names
- Automated system available to translate names to addresses
- Known as Domain Name System (DNS)

Hierarchical Names

- Accommodate a large, rapidly expanding set of names without requiring a central site
  - Decentralize the naming mechanism by delegating authority for parts of the namespace
  - Distribute responsibility for the mapping between names and addresses
- Partitioning of a namespace
  - Support efficient name mapping
  - Guarantee autonomous control of name assignment

Hierarchical Names (Cont.)

- Authority for names
  - Namespace is partitioned at the top level
  - Authority for names in subdivisions is passed to designated agents
  - For example

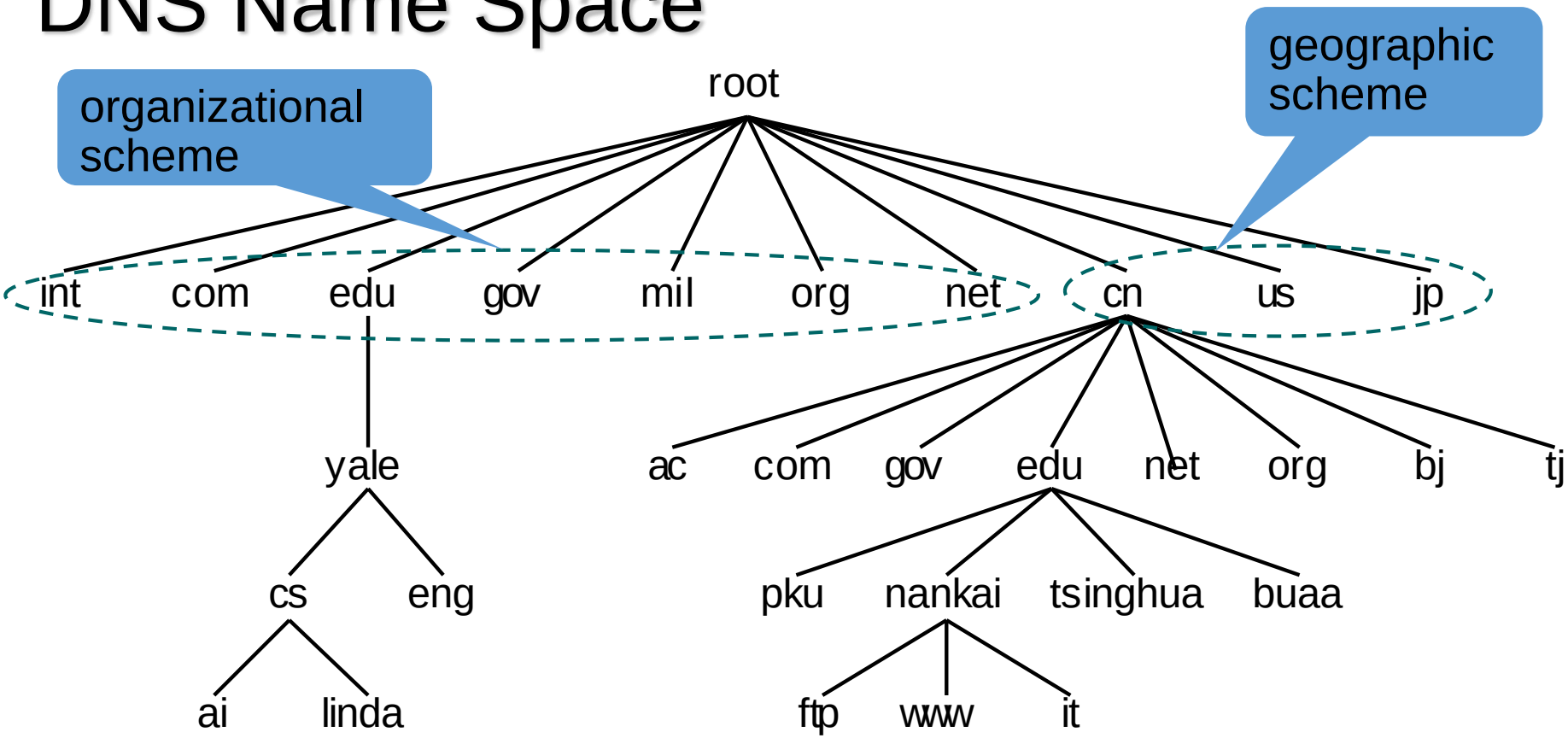
    local.site
- Subset Authority
  - Authority may be further subdivided at each level
  - For example

    local.group.site

Internet Domain Name System

- Hierarchical naming scheme
- Abstract:
  - name syntax
  - rules for delegating authority
- Concrete:
  - implementation of a distributed computing system that efficiently maps names to addresses
- RFC 1034, 1035
  - RFC 1034: domain names - concepts and facilities
  - RFC 1035: Domain names - implementation and specification

DNS Name Space



Domain Name Partition

- Organizational scheme
- Geographic scheme

DNS Name Space (Cont.）

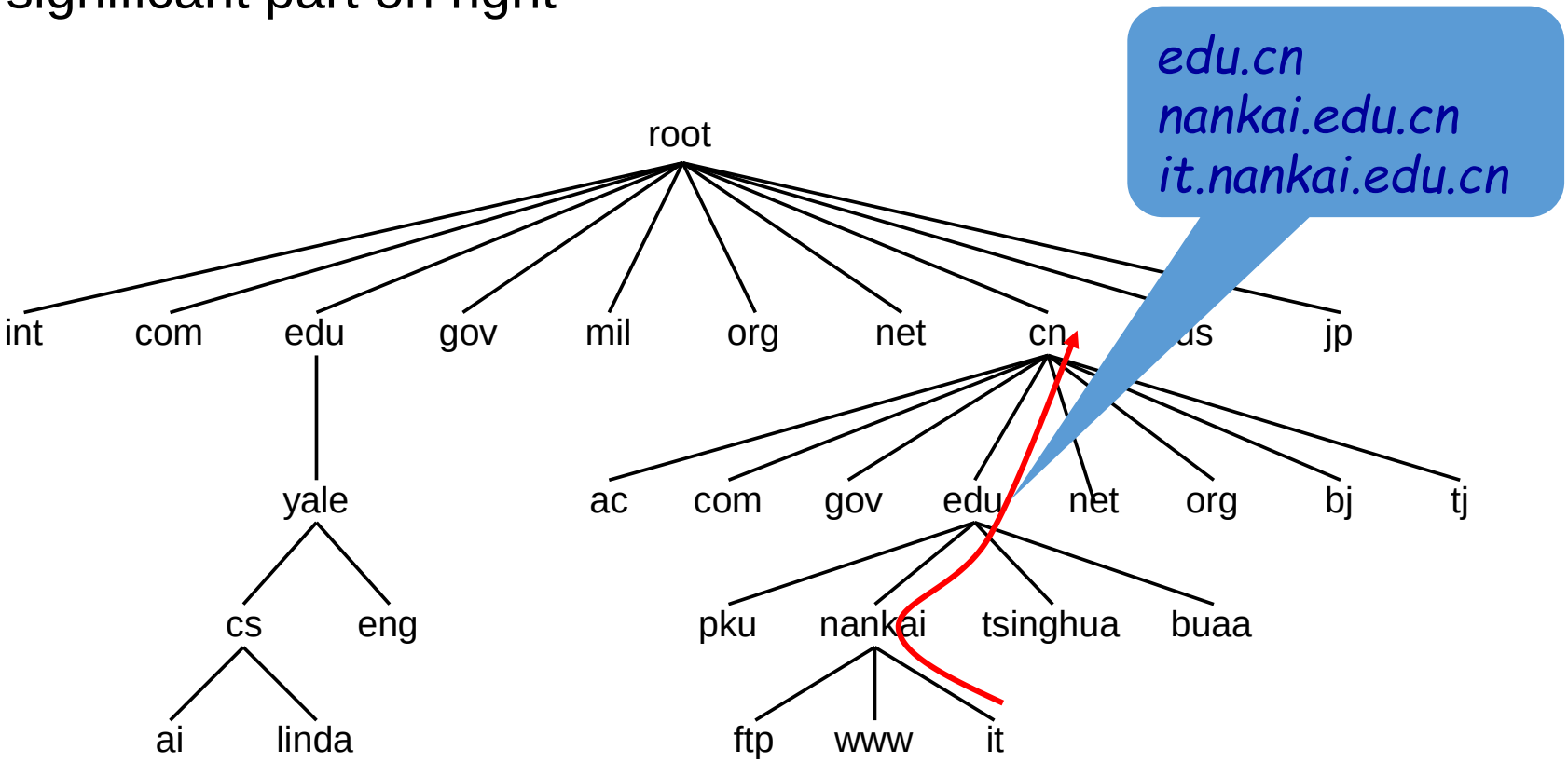
Top-level Domain Name	*Assigned to*
com	*Commercial organizations*
edu	*Educational institutions*
gov	*Government institutions*
mil	*Military groups*
net	*Major network support centers*
org	*Organizations other than those above*
arpa	*Temporary ARPANET domain*
int	*International organizations*
country code	*An country*

DNS Name Space (Cont.）

划分模式	我国二级域名	分配给
类别域名	ac	科研机构
	com	工、商、金融等企业
	edu	教育机构
	gov	政府部门
	net	互联网络、接入网络信息中心和运行中心
	org	各种非盈利性的组织
行政区域名	bj	北京市
	sh	上海市
	tj	天津市
	cq	重庆市
	……	…….

Domain Name Syntax

- Alphanumeric label (segment) separated by dots, most significant part on right



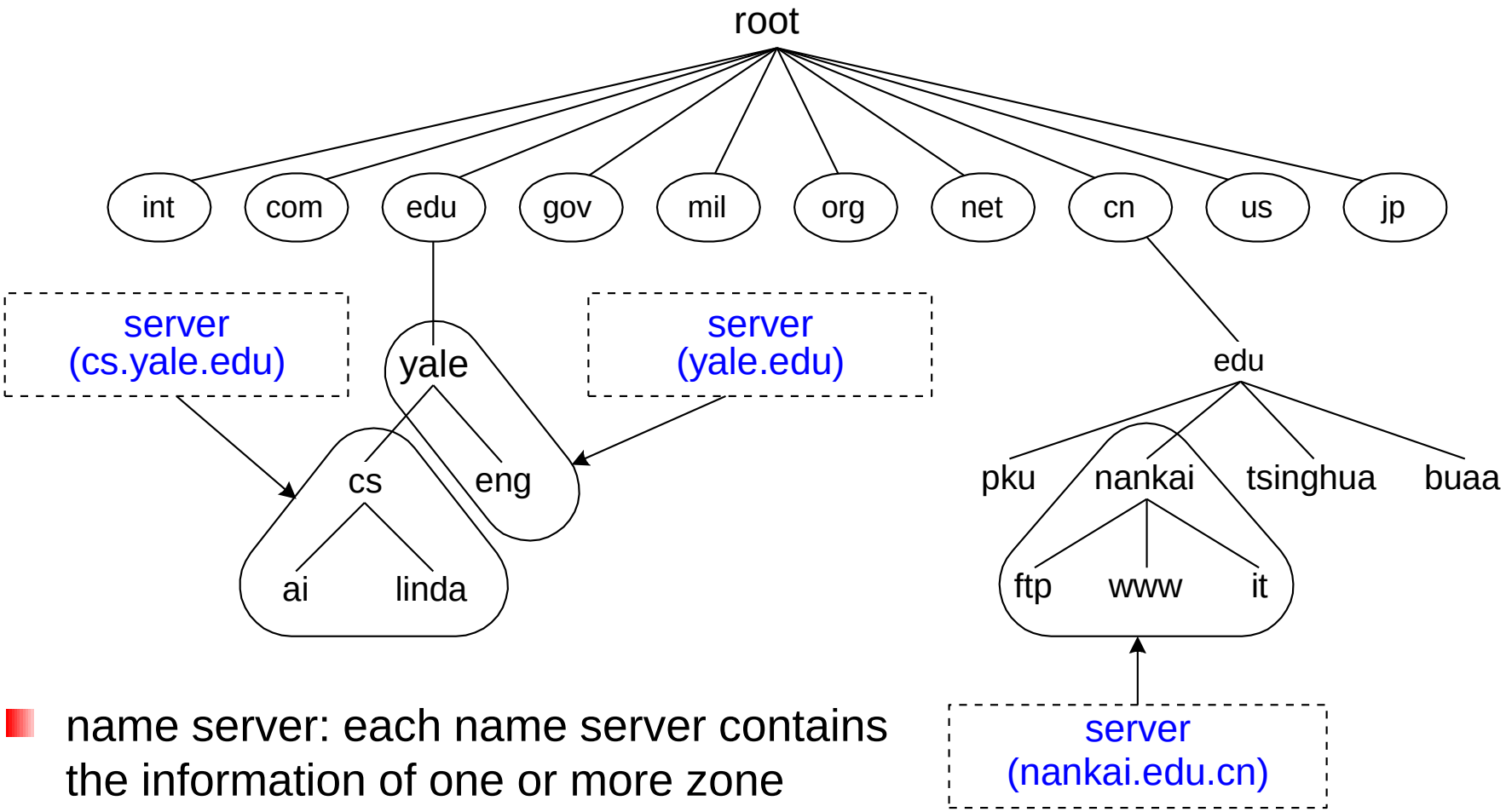
Name Resolution

- Mapping names to addresses
  - distributed
    - a set of servers operating at multiple sites cooperatively solve the mapping problem
  - efficient
    - most name can be mapped locally, only a few require internet traffic
  - reliable
    - no single machine failure will prevent the system from operating correctly
  - general purpose
    - not restricted to machine names

Name Resolution (Cont.)

- Name server
  - A server program that supplies name-to-address translation, mapping from domain names to IP address
- Name resolver
  - A client software that request name resolving
  - Use one or more name servers when mapping name to address

Name Server Hierarchy (Cont.)

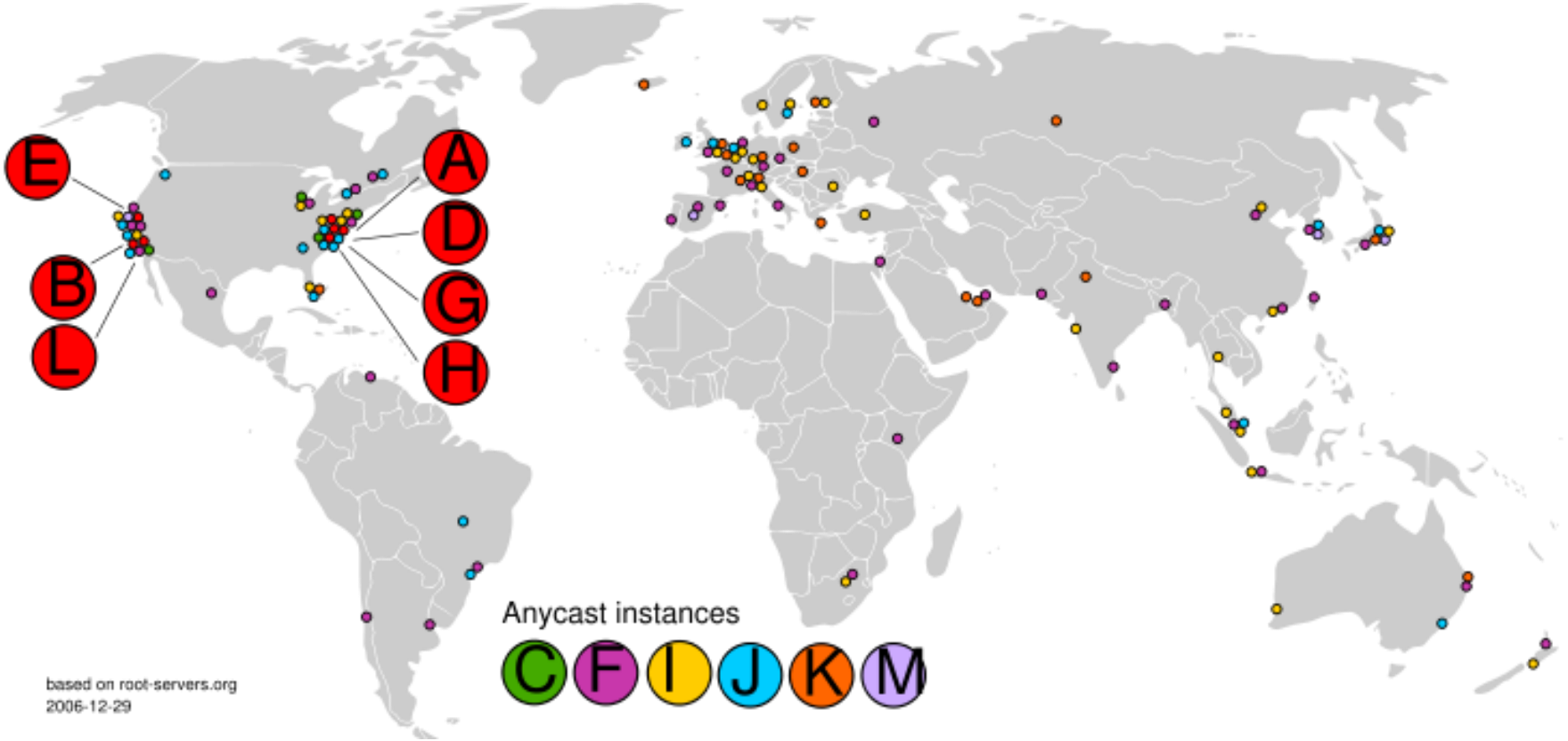


- name server: each name server contains the information of one or more zone
- parent server knows child server

Name servers

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

Root name servers



13 root name servers worldwide, *letter.root-servers.net*

Name servers

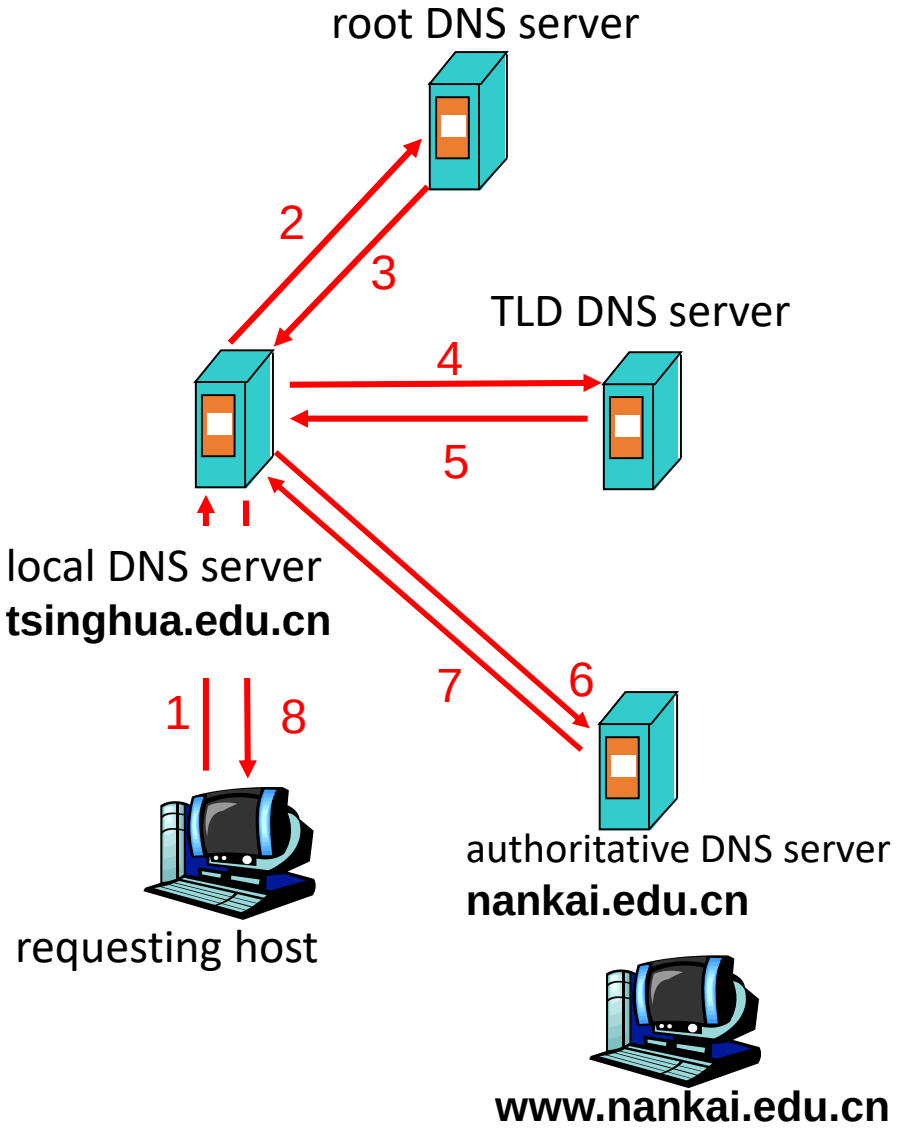
- top-level domain (TLD) servers:
  - responsible for com, org, net, edu, etc, and all top-level country domains uk, fr, ca, jp.
- authoritative name server:
  - organization's DNS servers, providing authoritative hostname to IP mappings for organization's servers (e.g., Web, mail).
  - can be maintained by organization or service provider
- local name servers:
  - 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

DNS name resolution example

- Host at tsinghua.edu.cn wants IP address for www.nankai.edu.cn

iterated query:

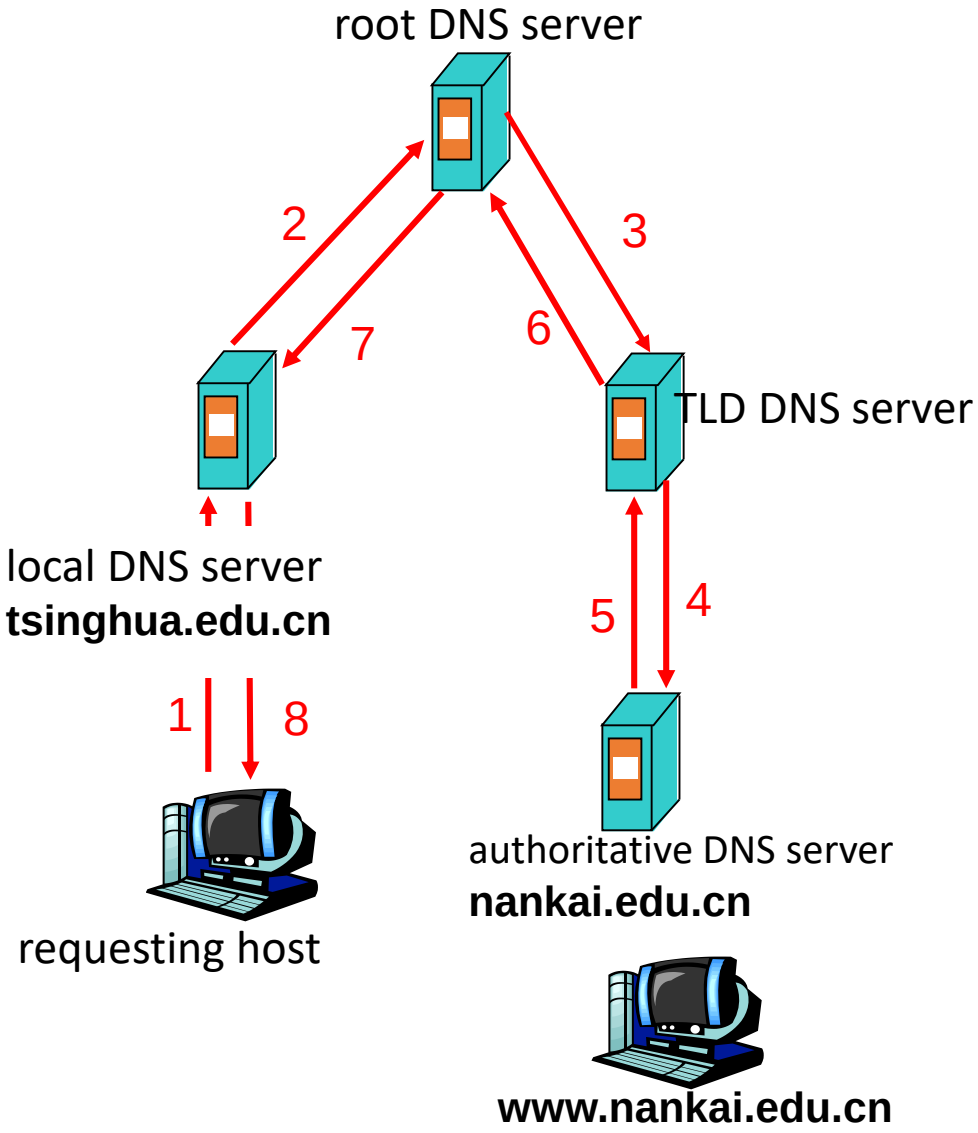
- contacted server replies with name of server to contact
- "I don't know this name, but ask this server"



DNS name resolution example

recursive query:

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



Name server caching

- Lower the cost of lookup for nonlocal names
- Name to address binding change infrenquently
- Server maintain a cache
  - once name server learns mapping, it caches mapping
  - record of where the mapping information for that name obtained
  - set timeout of entries according to authority TTL, cache entries timeout after some time
  - TLD servers typically cached in local name servers, thus root name servers not often visited

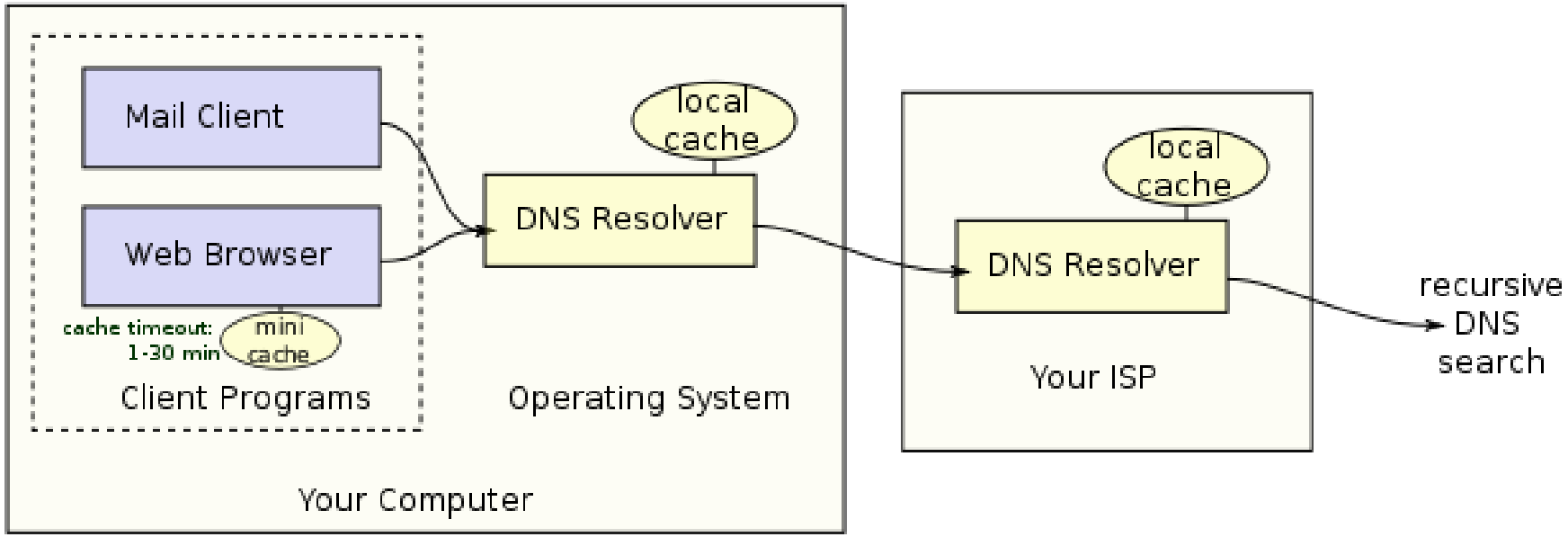
Name server caching (Cont.)

- Server report cache information to client
  - mark *nonauthoritative* binding
  - give the domain name and IP address of the server obtained binding
- Client receive answers quickly, but information may be out-of-date
  - If efficiency is important, the client will accept the nonauthoritative answer
  - If accuracy is important, the client will contact the authority and verify the binding is still valid

Host caching

- Method
  - download the database of names and addresses from local name server at startup
  - periodically obtain new mapping
  - cache recently used names
- Advantages
  - without any network activity, name resolution extremely fast
  - failure of local server don’t influence name resolution
  - reduce the computational load on the name server

Caching


Mail Client
Web Browser
cache timeout:
1-30 min
mini
cache
Client Programs
local
cache
DNS Resolver
Operating System
Your Computer
local
cache
DNS Resolver
Your ISP
recursive
DNS
search

Resource Record Structure

DNS: distributed database storing Resource Records (RR)

- Record structure
  - Name
  - Time to live: usually set to 86400s for authority resource record
  - Type
  - Class: Internet, I.e., IN
  - Value
- Alone database for each zone

Resource Record Type

Type	Meaning	Contents
A	Host Address	32-bit IP address
CNAME	Canonical Name	Canonical domain name for an alias
HINFO	CPU & OS	Name of CPU and operating system
MINFO	Mailbox info	Information about a mailbox or mail list
MX	Mail Exchanger	16-bit preference and name of host that acts as mail exchanger for the domain
NS	Name Server	Name of authoritative server for domain
PTR	Pointer	Domain name (like a symbolic link)
SOA	Start of Authority	Multiple fields that specify which parts of the naming hierarchy a server implements
TXT	Arbitrary text	Uninterpreted string of ASCII text

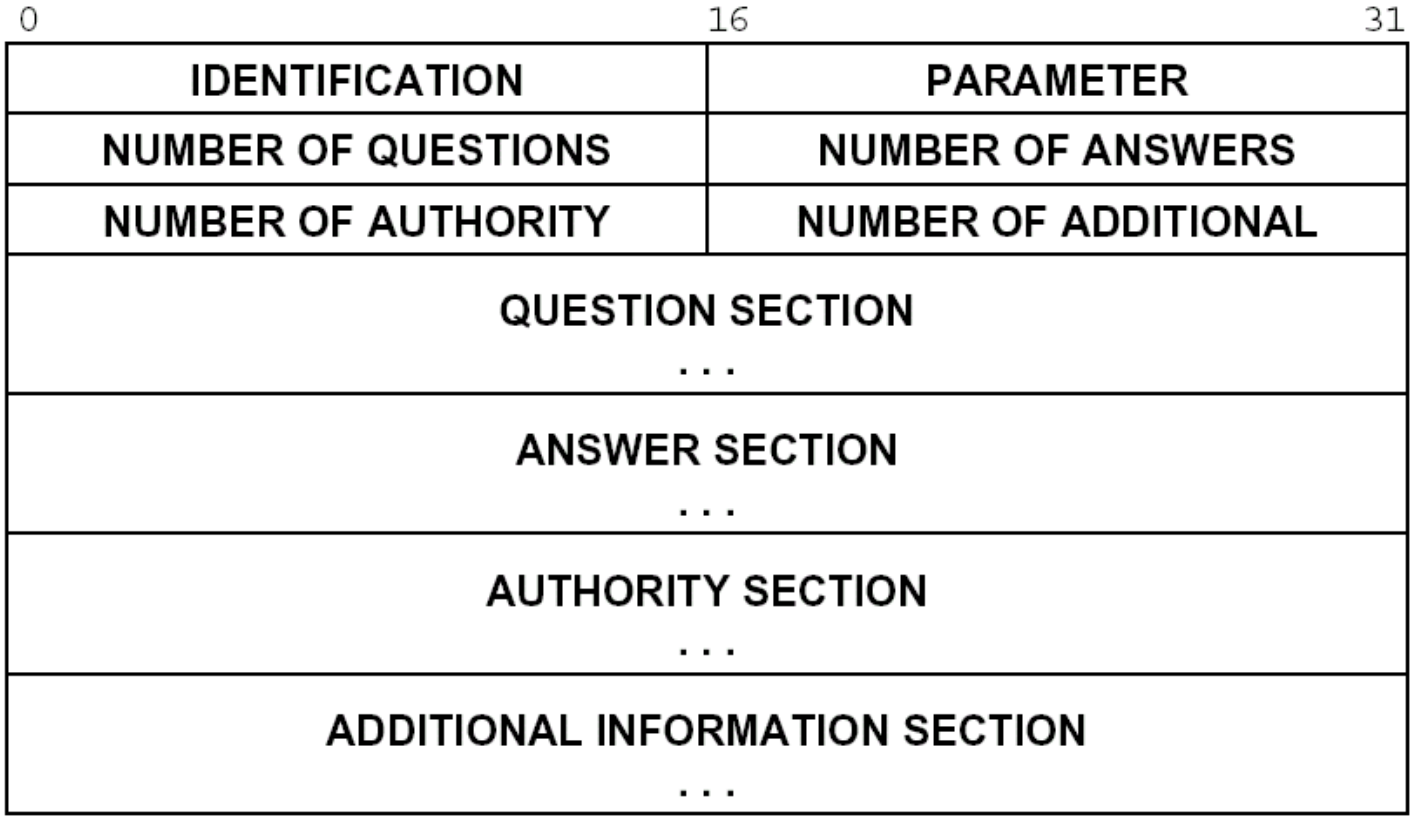
Type of IPv6 address: AAAA

Example of DNS Database

cs.vu.nl.	86400	IN	SOA	star boss (serial,refresh,retry,expire,ttl)
cs.vu.nl.	86400	IN	TXT	“A University”
cs.vu.nl.	86400	IN	MX	1 zephyer.cs.vu.nl.
cs.vu.nl.	86400	IN	MX	2 top.cs.vu.nl.
flits.cs.vu.nl.	86400	IN	HINFO	Sun Unix
flits.cs.vu.nl.	86400	IN	A	130.37.16.112
flits.cs.vu.nl.	86400	IN	MX	1 flits.cs.vu.nl.
flits.cs.vu.nl.	86400	IN	MX	2 zephyer.cs.vu.nl.
flits.cs.vu.nl.	86400	IN	MX	3 top.cs.vu.nl.
www.cs.vu.nl.	86400	IN	CNAME	top.cs.vu.nl.
ftp.cs.vu.nl.	86400	IN	CNAME	zephyer.cs.vu.nl.
zephyer	86400	IN	A	130.37.56.201
	IN	HINFO		Sun Unix

DNS Message Format

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



Message Format (Cont.)

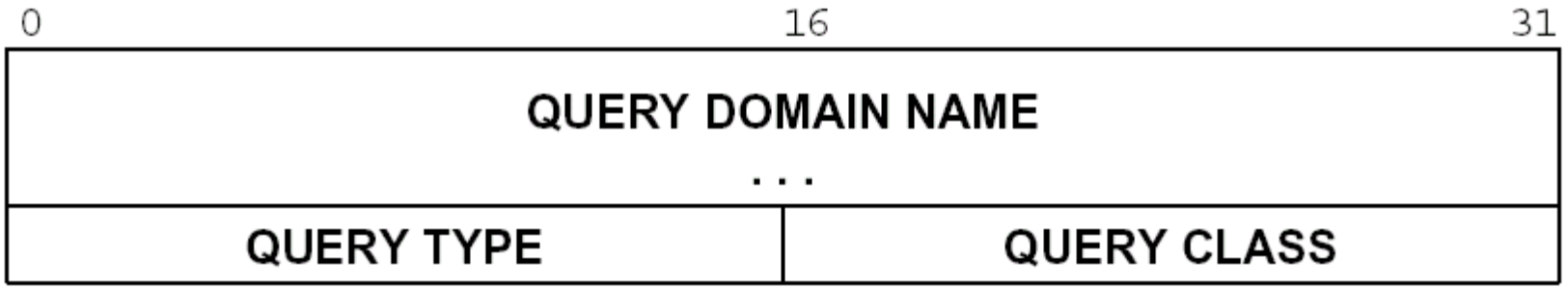
Bit of PARAMETER field	Meaning
0	Operation: 0 Query 1 Response
1-4	Query Type: 0 Standard 1 Inverse 2 Completion 1 (now obsolete) 3 Completion 2 (now obsolete)
5	Set if answer authoritative
6	Set if message truncated
7	Set if recursion desired
8	Set if recursion available
9-11	Reserved
12-15	Response Type: 0 No error 1 Format error in query 2 Server failure 3 Name does not exist

Message Format (Cont.)

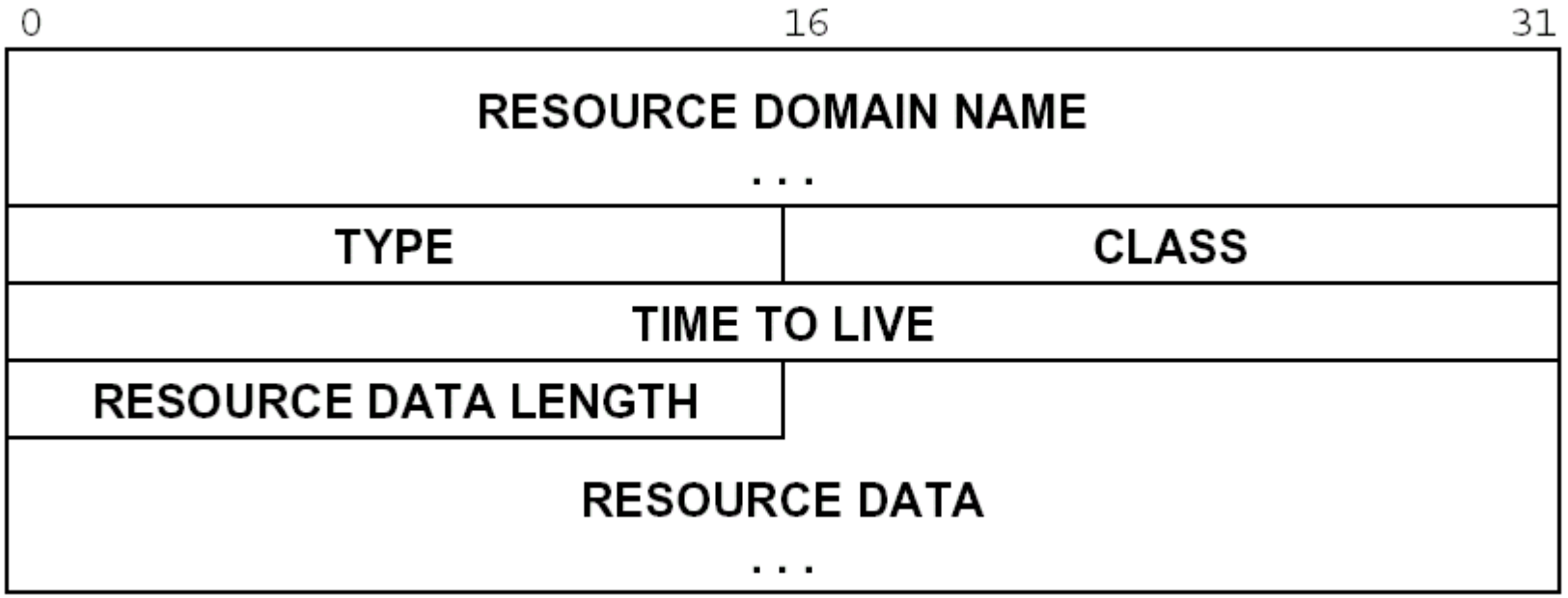
- Question: Carries the query name and other query parameters.
- Answer: Carries RRs which directly answer the query.
- Authority: Carries RRs which describe other authoritative servers. May optionally carry the SOA RR for the authoritative data in the answer section.
- Additional: Carries RRs which may be helpful in using the RRs in the other sections.

Message Format (Cont.)

question



answer
authority
additional

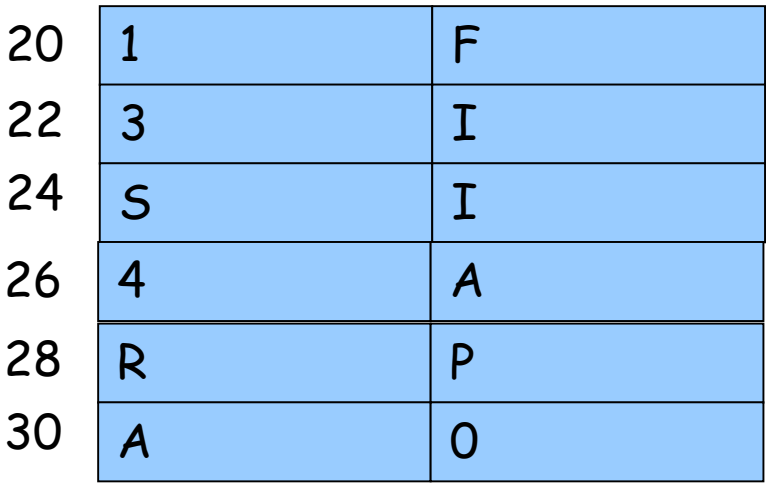


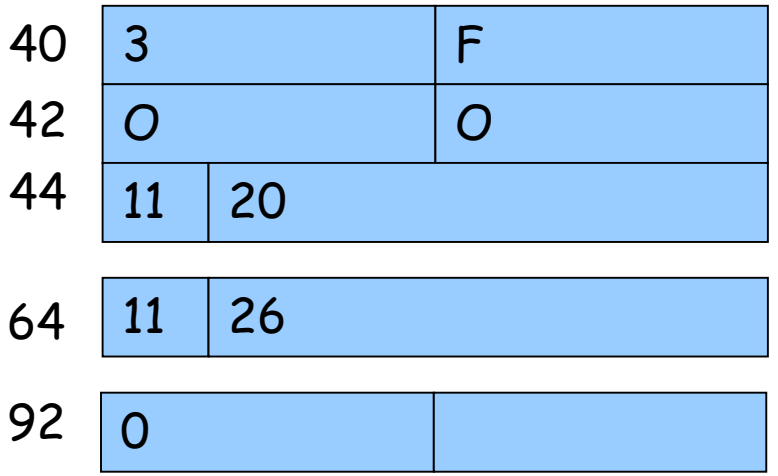
Compressed Name Format

- Domain name representation in message
  - each segment: first octet specifies length (*n*), following *n* octets (00xxxxxx)
  - Length octet containing zero marks the end of name
- Compress format
  - Suffixes of domain usually overlap in multiple
  - Name server store only one copy of each domain name
  - Pointer
    - top two bits of segment count field is 1s, next 14 bits is pointer (11xxxxxxxxxxxxxx)

Compressed Name Format: example

- A datagram might need to use the domain names F.ISI.ARPA, FOO.F.ISI.ARPA, ARPA, and the root. Ignoring the other fields of the message, these domain names might be represented as:





Message Format: example

- A mailer tying to send mail to Mockapetris@ISI.EDU might ask the resolver for mail information about ISI.EDU
- Query: question section

  QNAME=ISI.EDU, QTYPE=MX, QCLASS=IN

- Reply: answer section

  ISI.EDU. MX 10 VENERA.ISI.EDU.

  MX 10 VAXA.ISI.EDU.

- Reply: additional section

  VAXA.ISI.EDU. A 10.2.0.27

  A 128.9.0.33

  VENERA.ISI.EDU. A 10.1.0.52

  A 128.9.0.32

Message Format: example

- If a query (QNAME=BRL.MIL, QTYPE=A) is sent to C.ISI.EDU, the reply would be:
- answer section

  <empty>
- authority section

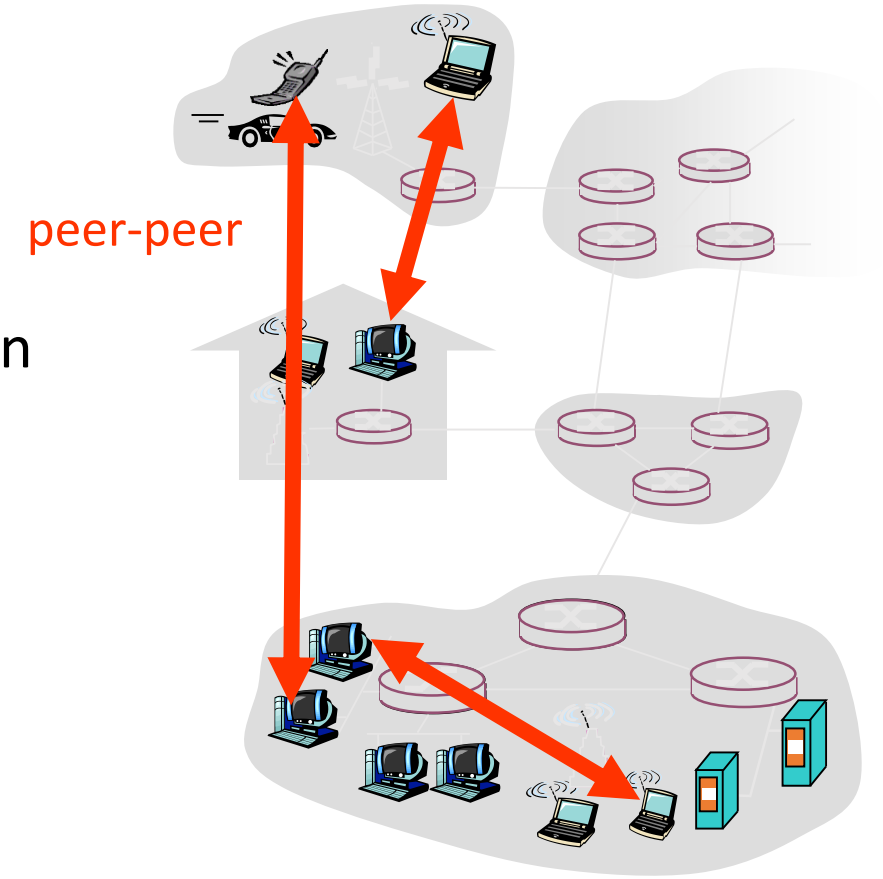
MIL.	86400	IN	NS	SRI-NIC.ARPA.
	86400		NS	A.ISI.EDU.

- additional section

A.ISI.EDU.	A	26.3.0.103
SRI-NIC.ARPA.	A	26.0.0.73
	A	10.0.0.51

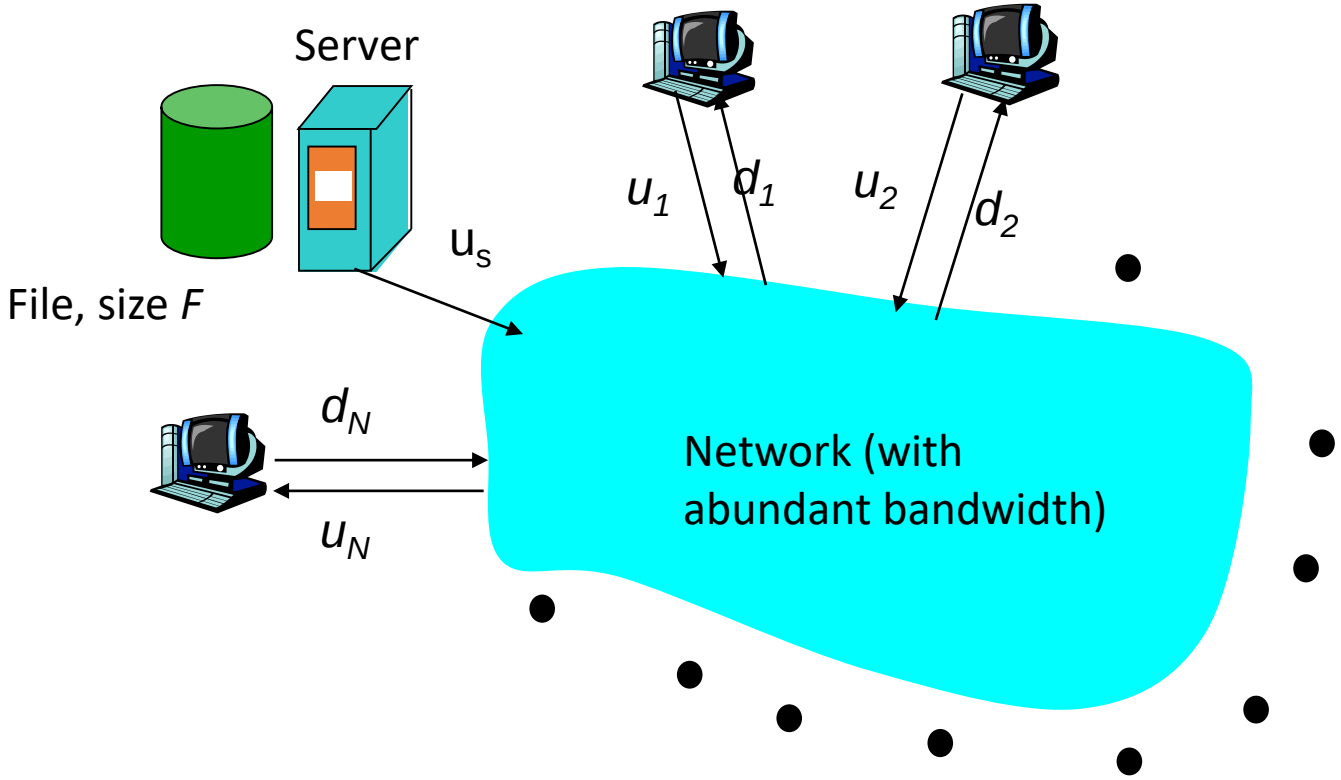
P2P architecture

- Three topics:
  - File distribution
  - Searching for information
  - Case Study: Skype



File Distribution: Client-Server vs P2P

Question : How much time to distribute file from one server to *N peers*?



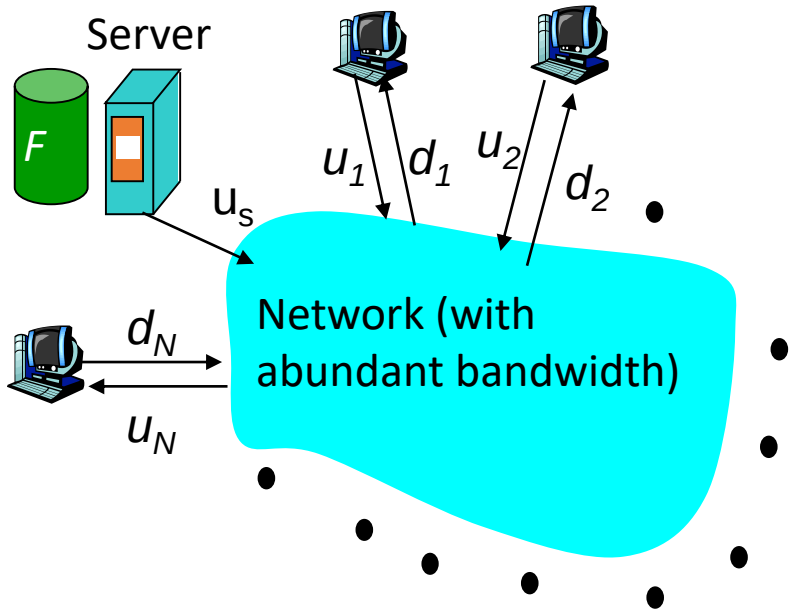
u_s: server upload bandwidth

u_i: peer i upload bandwidth

d_i: peer i download bandwidth

File distribution time: client-server

- server sequentially sends N copies:
  - NF/u_s time
- client i takes F/d_i time to download



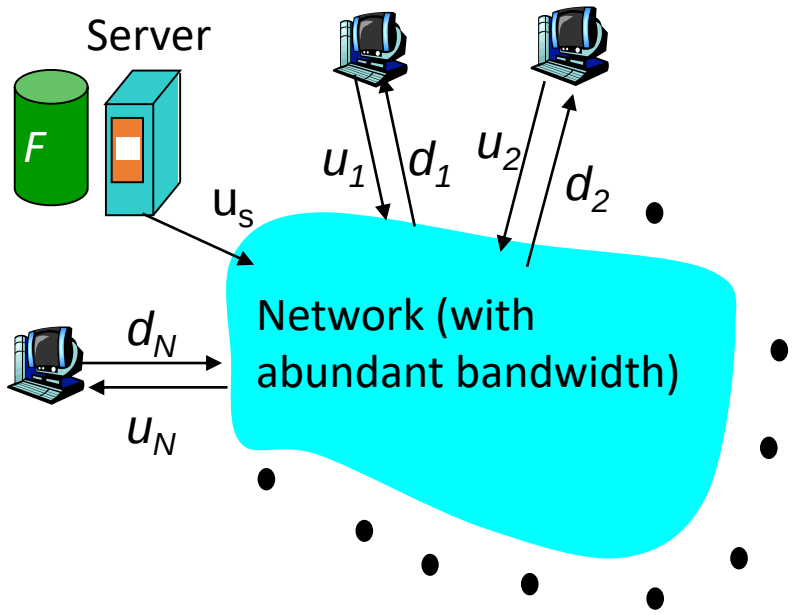
Time to distribute F to N clients using= $d_{cs} = \max\{ NF/u_s, F/\min(d_i) \}$ client/server approach

i

increases linearly in N(for large N)

File distribution time: P2P

- server must send one copy: F/u_s time
- client i takes F/d_i time to download
- NF bits must be downloaded (aggregate)

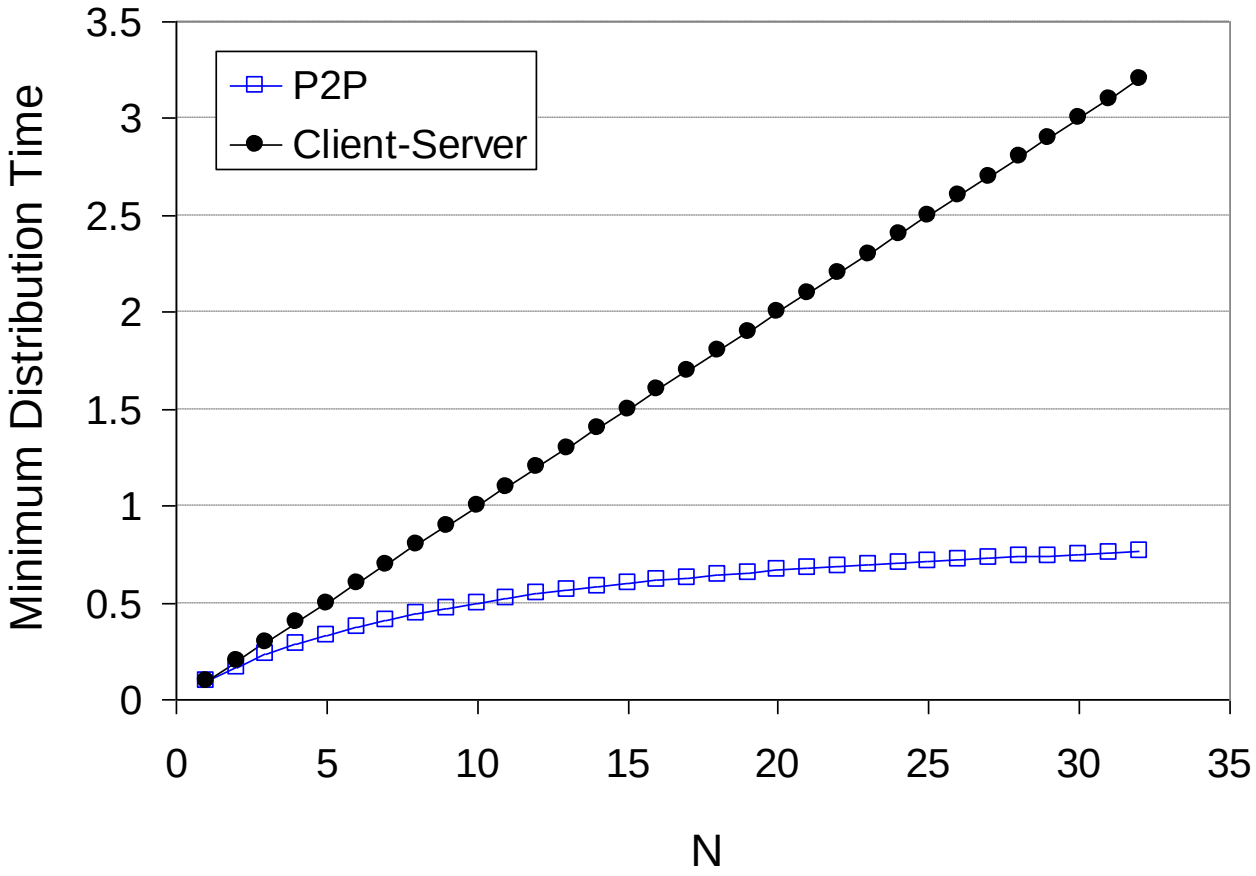


- fastest possible upload rate: $u_s + \Sigma u_i$

$$d_{P2P} = \max \{ F/u_s, F/\min(d_i) , NF/(u_s + \sum_i u_i) \}$$

Server-client vs. P2P: example

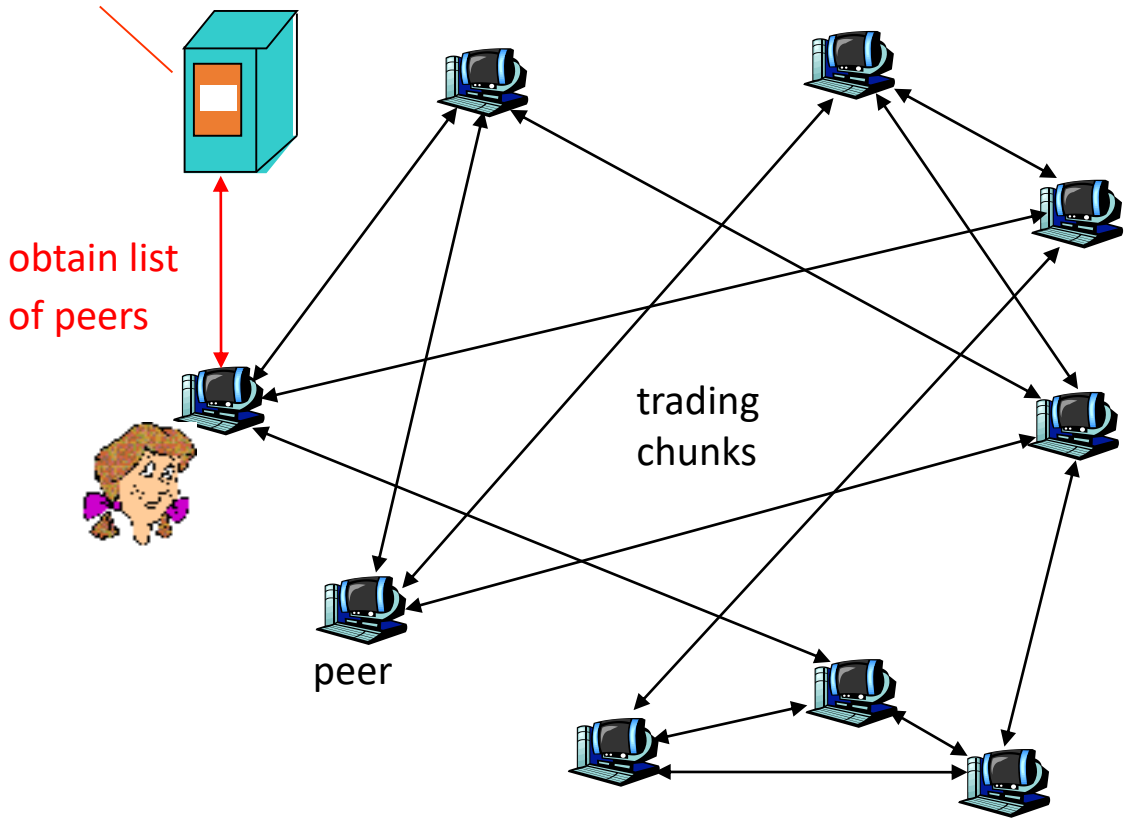
Client upload rate = u, F/u = 1 hour, u_s = 10u, $d_{min} \geq u_s$



File distribution: BitTorrent

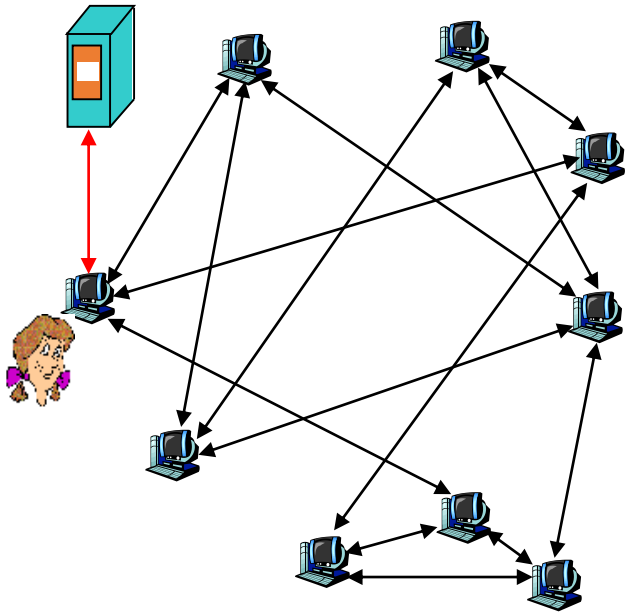
tracker: tracks peers participating in torrent

torrent: group of peers exchanging chunks of a file



BitTorrent (1)

- file divided into 256KB *chunks*.
- peer joining torrent:
  - has no chunks, but will accumulate them over time
  - registers with tracker to get list of peers, connects to subset of peers ("neighbors")
- while downloading, peer uploads chunks to other peers.
- peers may come and go
- once peer has entire file, it may (selfishly) leave or (altruistically) remain



BitTorrent (2)

Pulling Chunks

- at any given time, different peers have different subsets of file chunks
- periodically, a peer (Alice) asks each neighbor for list of chunks that they have.
- Alice sends requests for her missing chunks
  - rarest first

Sending Chunks: tit-for-tat

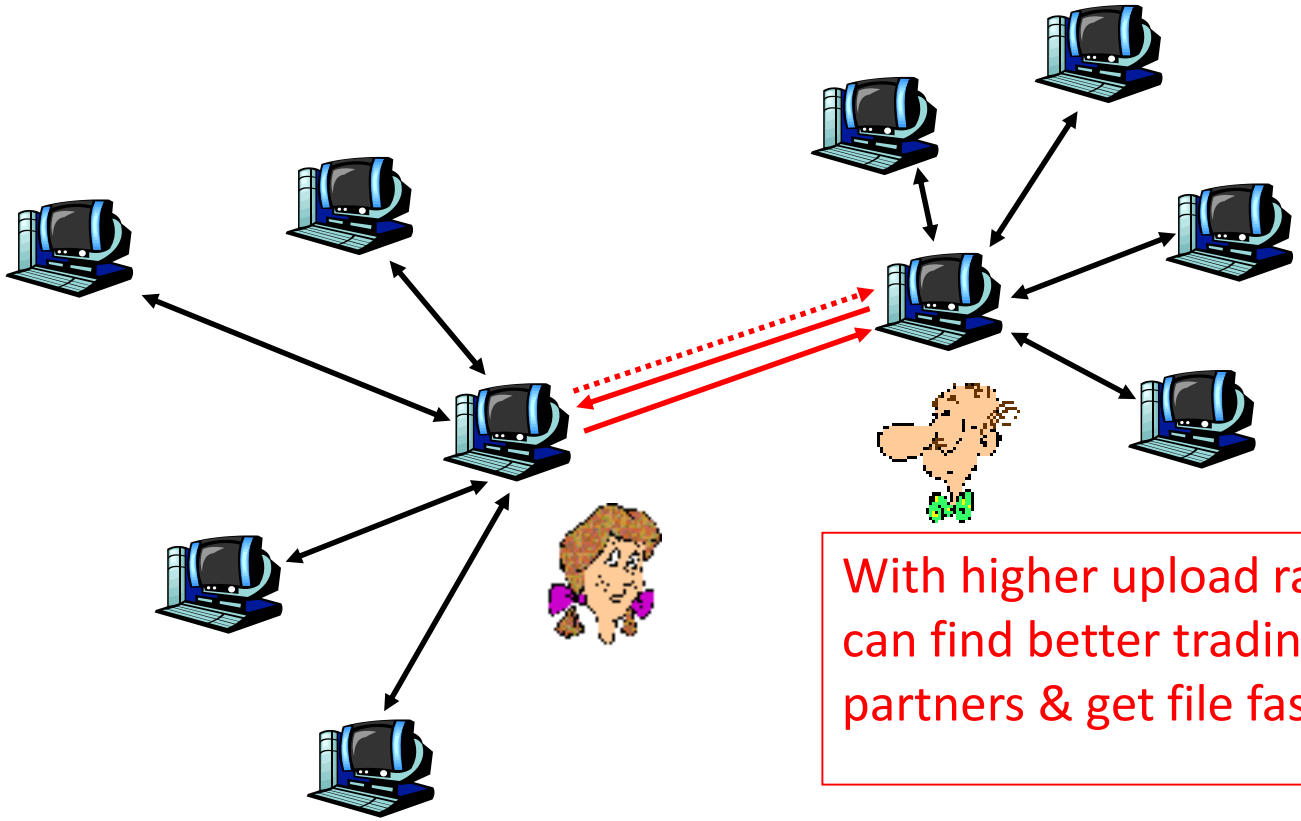
- Alice sends chunks to four neighbors currently sending her chunks *at the highest rate* periodically, a peer (Alice) asks each neighbor for list of chunks that they have.
  - reevaluate top 4 every 10 secs
- 30 secs: randomly select another peer, starts sending chunks
  - newly chosen peer may join top 4
  - “optimistically unchoke”

BitTorrent: Tit-for-tat

(1) Alice “optimistically unchokes” Bob

(2) Alice becomes one of Bob’s top-four providers; Bob reciprocates

(3) Bob becomes one of Alice’s top-four providers



With higher upload rate, can find better trading partners & get file faster!

P2P: searching for information

Index in P2P system: maps information to peer location (location = IP address & port number)

File sharing (eg. e-mule)

- Index dynamically tracks the locations of files that peers share.
- Peers need to tell index what they have.
- Peers search index to determine where files can be found

Instant messaging

- Index maps user names to locations.
- When user starts IM application, it needs to inform index of its location
- Peers search index to determine IP address of user.

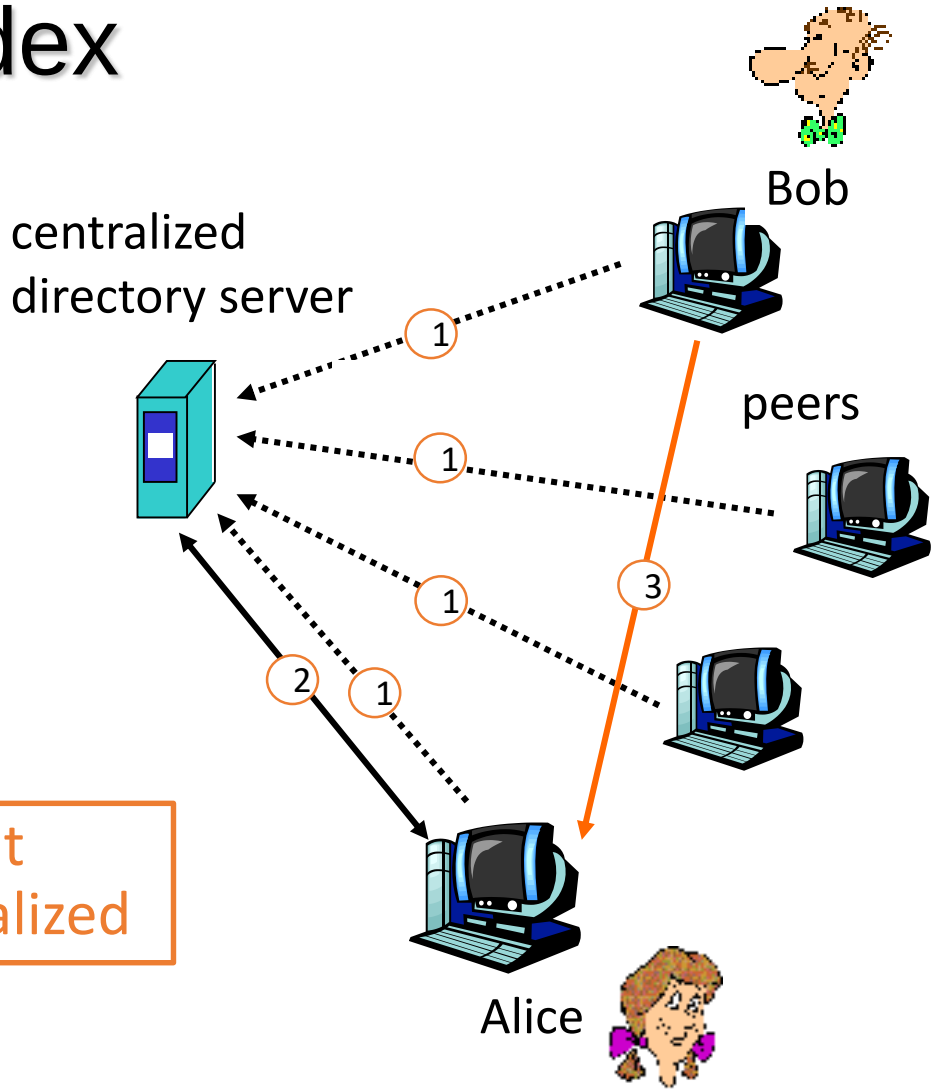
P2P: centralized index

Original “Napster” design

1) when peer connects, it informs central server:
   - IP address
   - content

2) Alice queries for “Hey Jude”

3) Alice requests file from Bob



file transfer is decentralized, but locating content is highly centralized

- single point of failure
- performance bottleneck

Query flooding

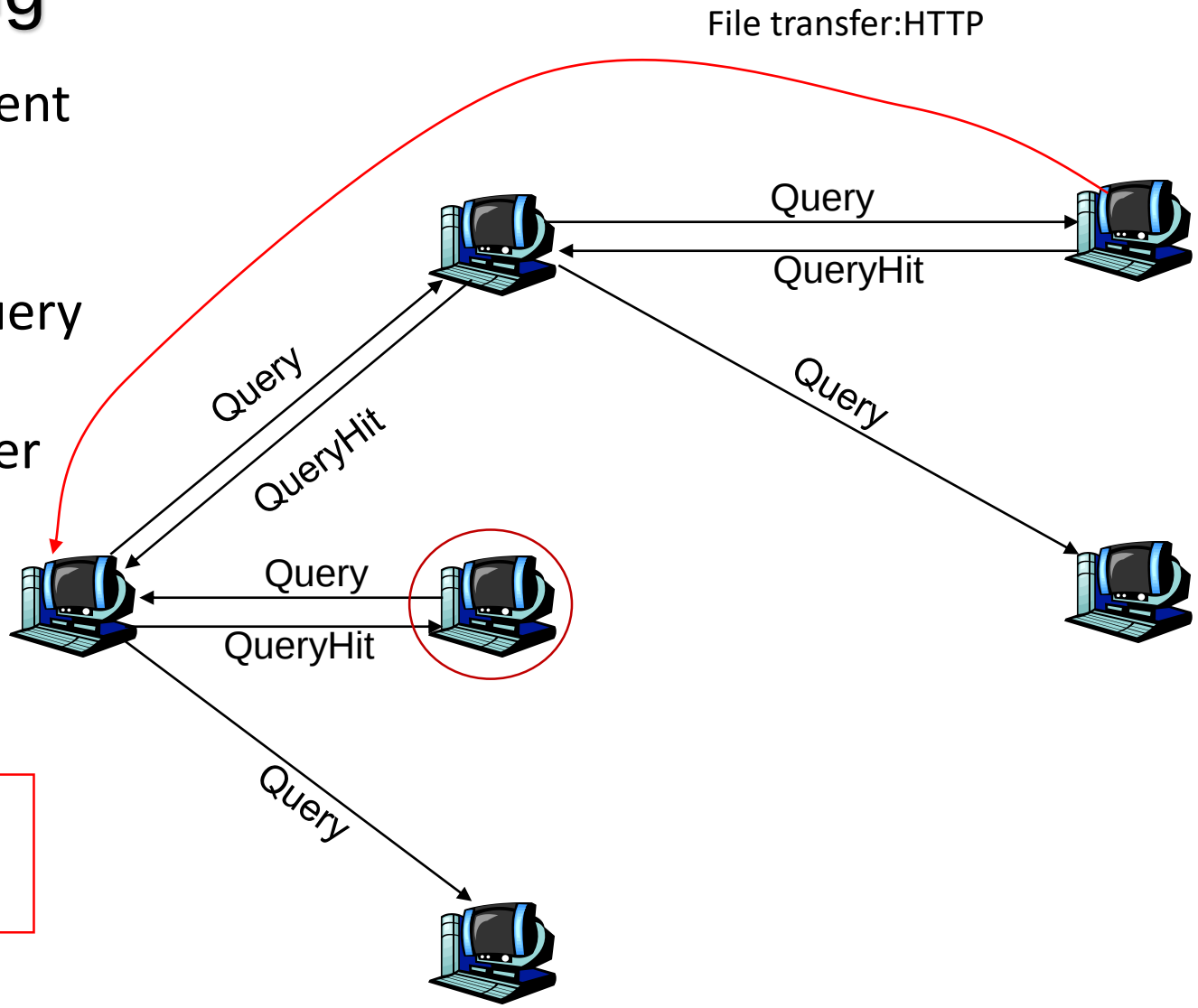
- fully distributed
  - no central server
- used by Gnutella
- Each peer indexes the files, it makes available for sharing

overlay network: graph

- edge between peer X and Y if there's a TCP connection
- all active peers and edges form overlay net
- edge: virtual (*not* physical) link
- given peer typically connected with < 10 overlay neighbors

Query flooding

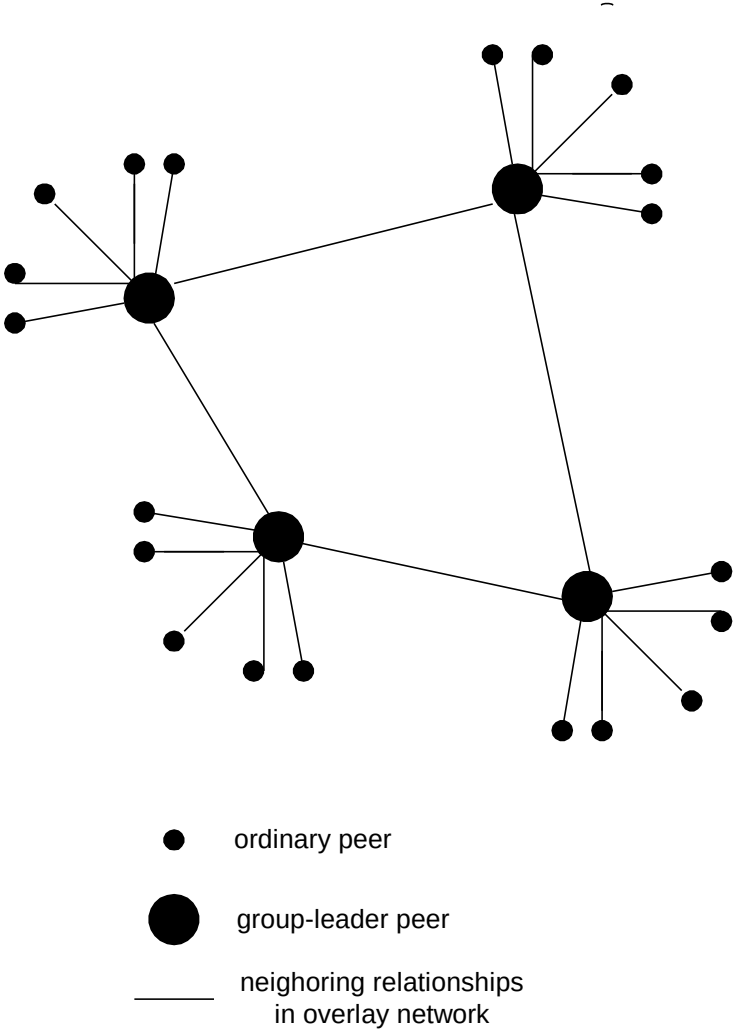
- Query message sent over existing TCP connections
- peers forward Query message
- QueryHit sent over reverse path



Scalability: limited scope flooding

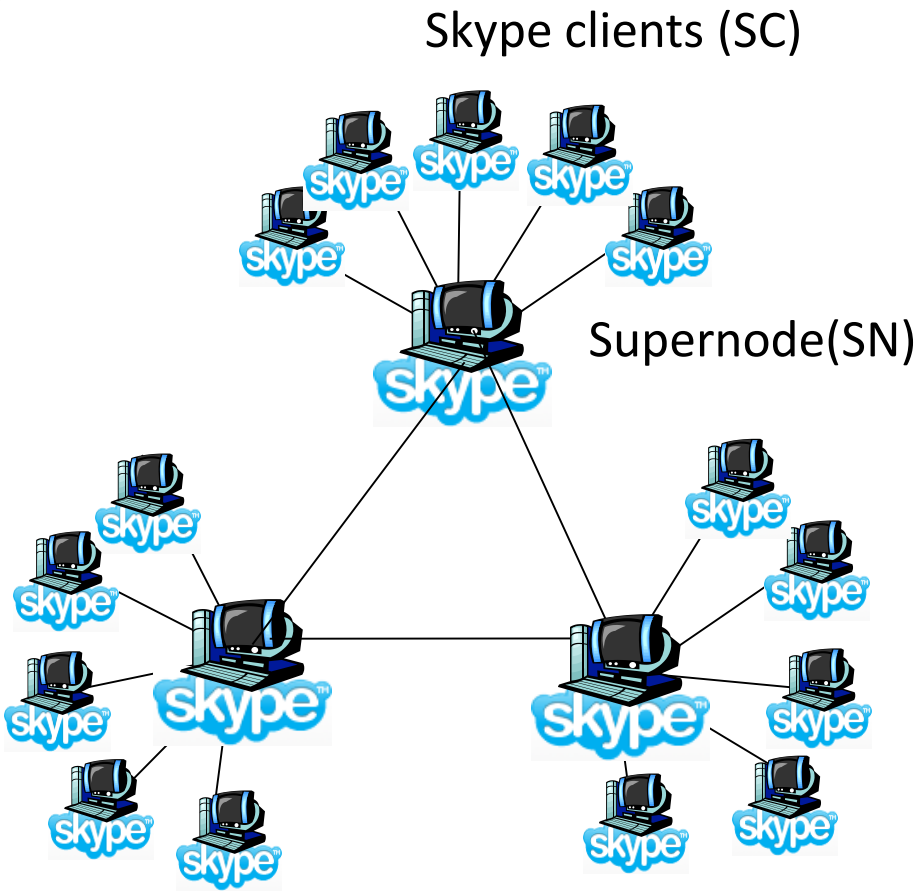
Hierarchical Overlay

- between centralized index, query flooding approaches
- each peer is either a *super node* or assigned to a super node
  - TCP connection between peer and its super node.
  - TCP connections between some pairs of super nodes.
- Super node tracks content in its children



P2P Case study: Skype

- inherently P2P: pairs of users communicate.
- proprietary application-layer protocol (inferred via reverse engineering)
- hierarchical overlay with SNs
- Index maps usernames to IP addresses; distributed over SNs



Summary

- application service requirements:
  - reliability, bandwidth, delay
- Internet transport service model
  - connection-oriented, reliable: TCP
  - unreliable, datagrams: UDP
- client-server and p2p

- specific protocols:
  - ftp
  - http
  - smtp, pop3
  - dns
  - p2p