Chapter 2: Application layer

- 2.1 Principles of network applications
  應用層原理
- 2.2 Web and HTTP
- 2.3 FTP
- 2.4 Electronic Mail
  - SMTP, POP3, IMAP
- 2.5 DNS
  網域名稱系統
- 2.6 P2P Applications
  點對點應用
- 2.7 Socket programming with TCP
- 2.8 Socket programming with UDP

DNS: Domain Name System

People: many identifiers:

- SSN, name, passport #
  多個識別碼

Internet hosts, routers:

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

Q: map between IP addresses and name ?
IP與名稱的對應

Domain Name System:

- *distributed database*
  分散式資料庫
  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: Domain Name System

DNS services

- hostname to IP address translation
  名稱轉譯
- host aliasing主機別名
  - Canonical, alias names
- mail server aliasing
  郵件伺服器別名
- load distribution
  負載分配
  - replicated Web servers: set of IP addresses for one canonical name

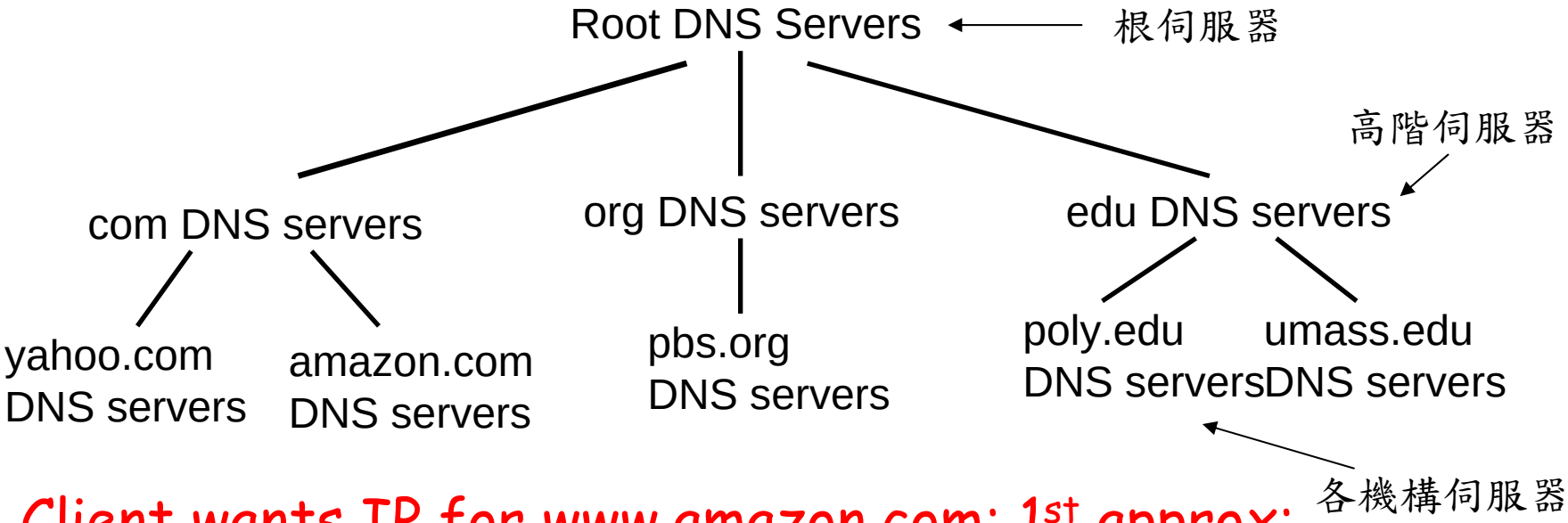
Why not centralize DNS?

- single point of failure
  一個壞全部壞
- traffic volume
  大網路流量
- distant centralized database
  遠距離集中式資料庫（delay）
- maintenance
  維護問題

doesn't *scale!* *無法擴充*

Distributed, Hierarchical Database

分散式、階層式資料庫

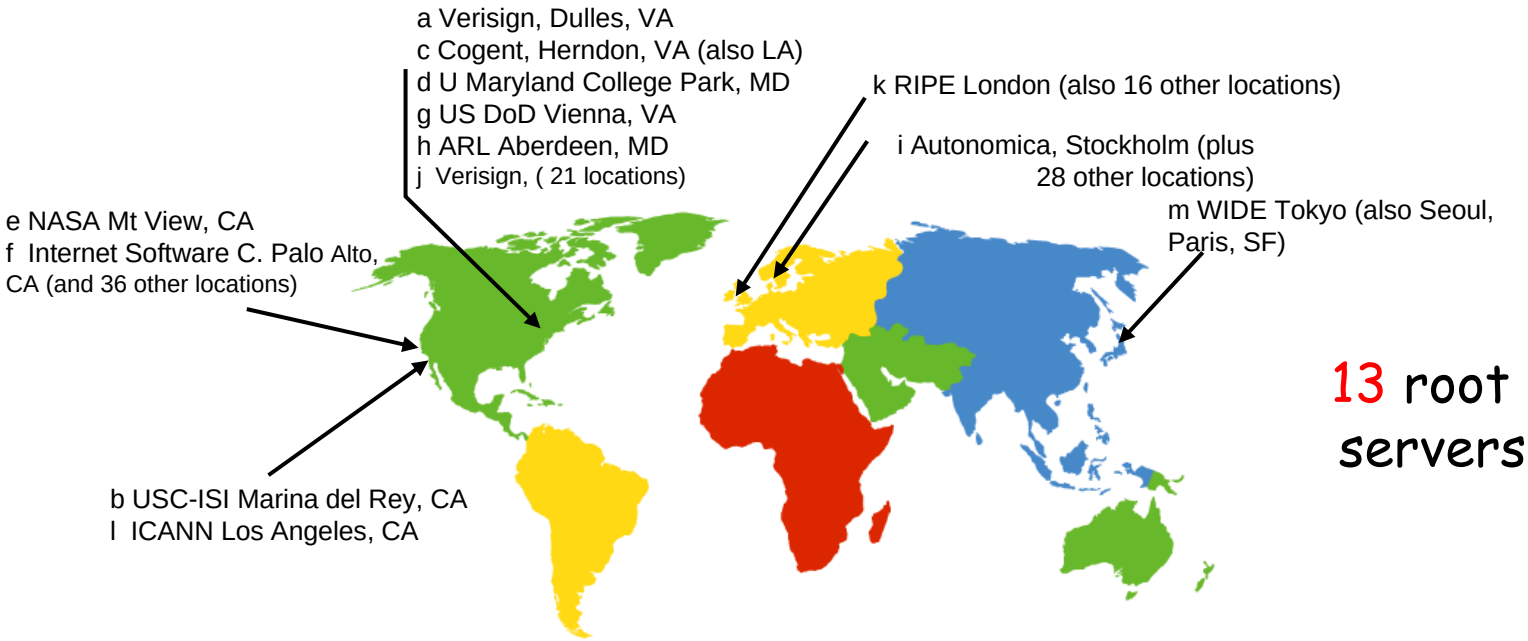


Client wants IP for www.amazon.com; 1st approx:

- 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

DNS: Root name servers 根伺服器

- 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



13 root name servers worldwide

TLD and Authoritative Servers

- Top-level domain (TLD) servers: 高階網域伺服器
  - responsible for com, org, net, edu, etc, and all top-level country domains uk, fr, ca, jp.
  - Network Solutions maintains servers for com TLD
  - Educause for edu TLD
- Authoritative DNS servers: 各機構之官方伺服器
  - 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 Server 區域伺服器

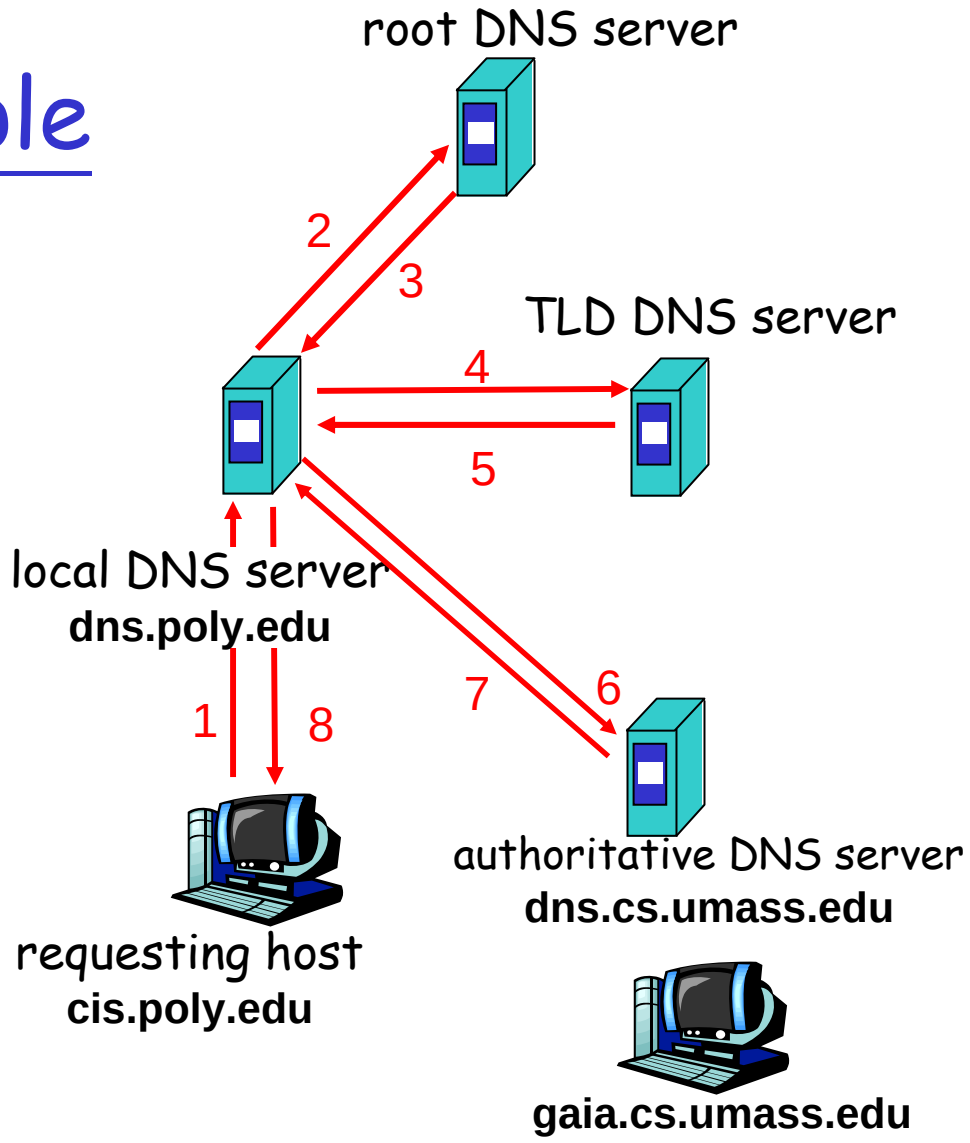
- does not strictly belong to hierarchy
  不一定是階層式架構
- each ISP (residential ISP, company, university) has one.
  - also called "default name server"
- when host makes DNS query, query is sent to its local DNS server
  - acts as proxy, forwards query into hierarchy
    和proxy作用相同，將查詢傳至上層

DNS name resolution example

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

iterated query:
循環式查詢

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

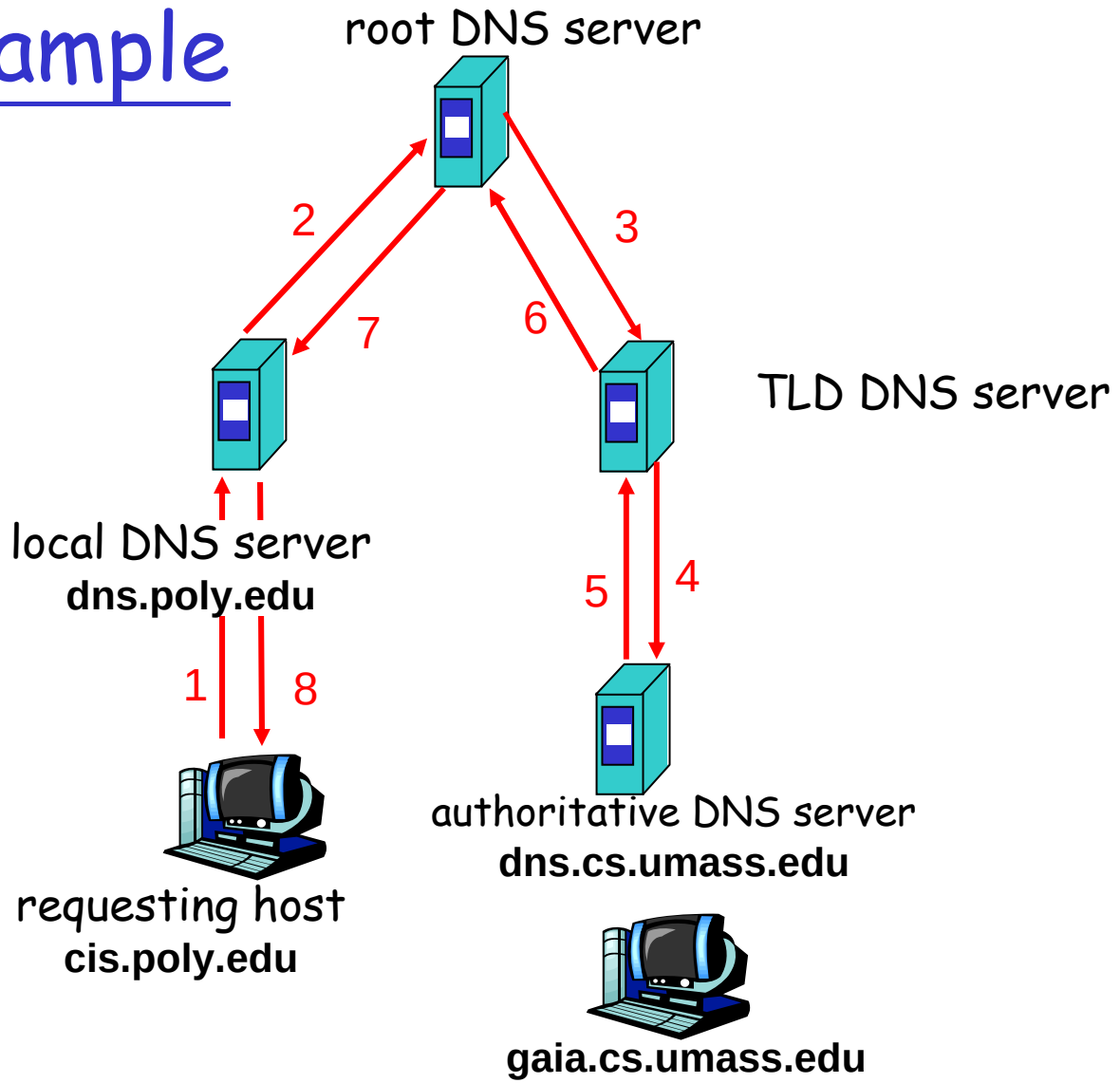


DNS name resolution example

recursive query:

遞迴式查詢

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



DNS: caching and updating records
DNS快取及更新

- once (any) name server learns mapping, it *caches* mapping
  - cache entries timeout (disappear) after some time 每隔一段時間就丟掉快取的資訊
  - TLD servers typically cached in local name servers 快取TLD伺服器的位址
    - 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

DNS records DNS記錄

DNS: distributed db storing resource records (RR)

RR format: **(name, value, type, ttl)**記錄格式

- 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

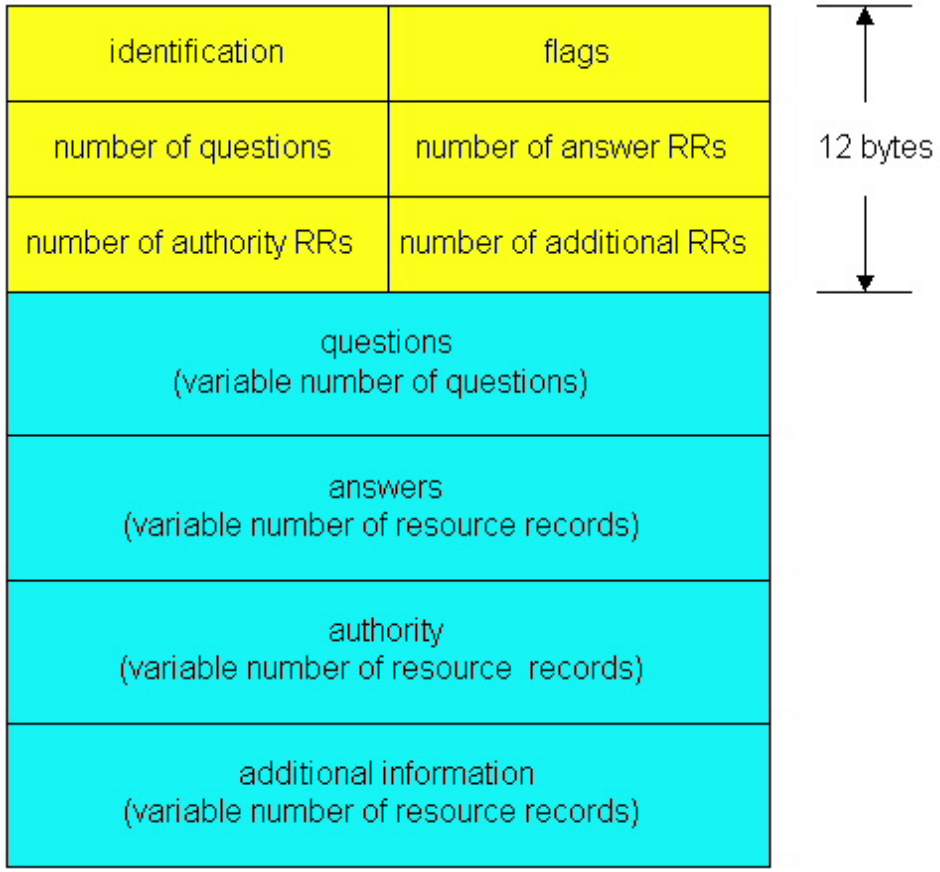
    `www.ibm.com` is really `servereast.backup2.ibm.com`
  - **value** is canonical name
- Type=MX
  - **value** is name of mailserver associated with **name**

DNS protocol, messages 協定與訊息

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

msg header

- identification: 16 bit # for query, reply to query uses same #
  在查詢及回應的訊息中，用相同的id
- flags:
  - query or reply
  - recursion desired
  - recursion available
  - reply is authoritative



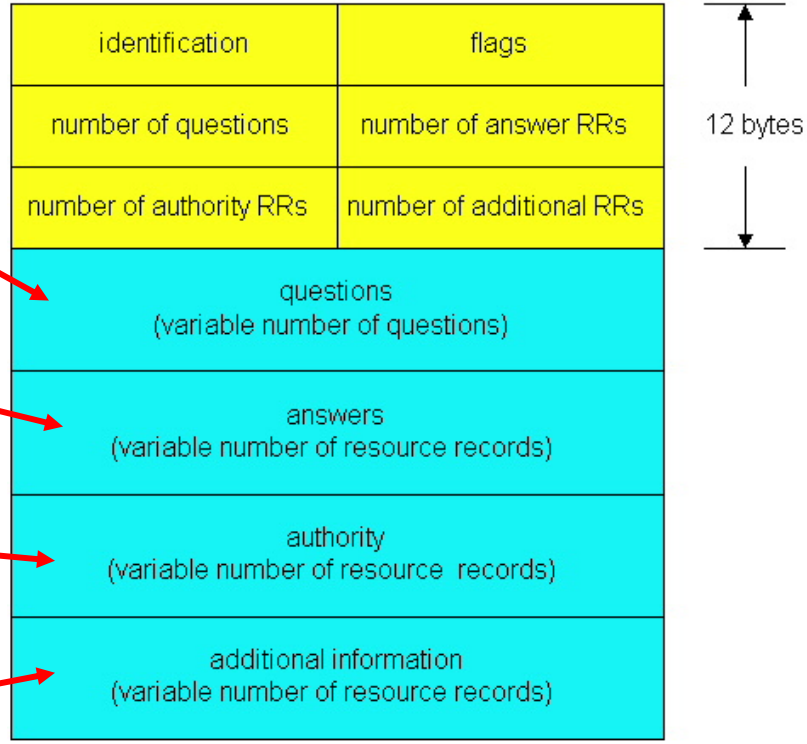
DNS protocol, messages

Name, type fields for a query

RRs in response to query
可以有多個回覆

records for authoritative servers

additional "helpful" info that may be used



Inserting records into DNS

如何增加一筆資料進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?

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
  點對點應用
- 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"
- 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!

所有電腦都是server

P2P: centralized directory

集中式目錄

original "Napster" design

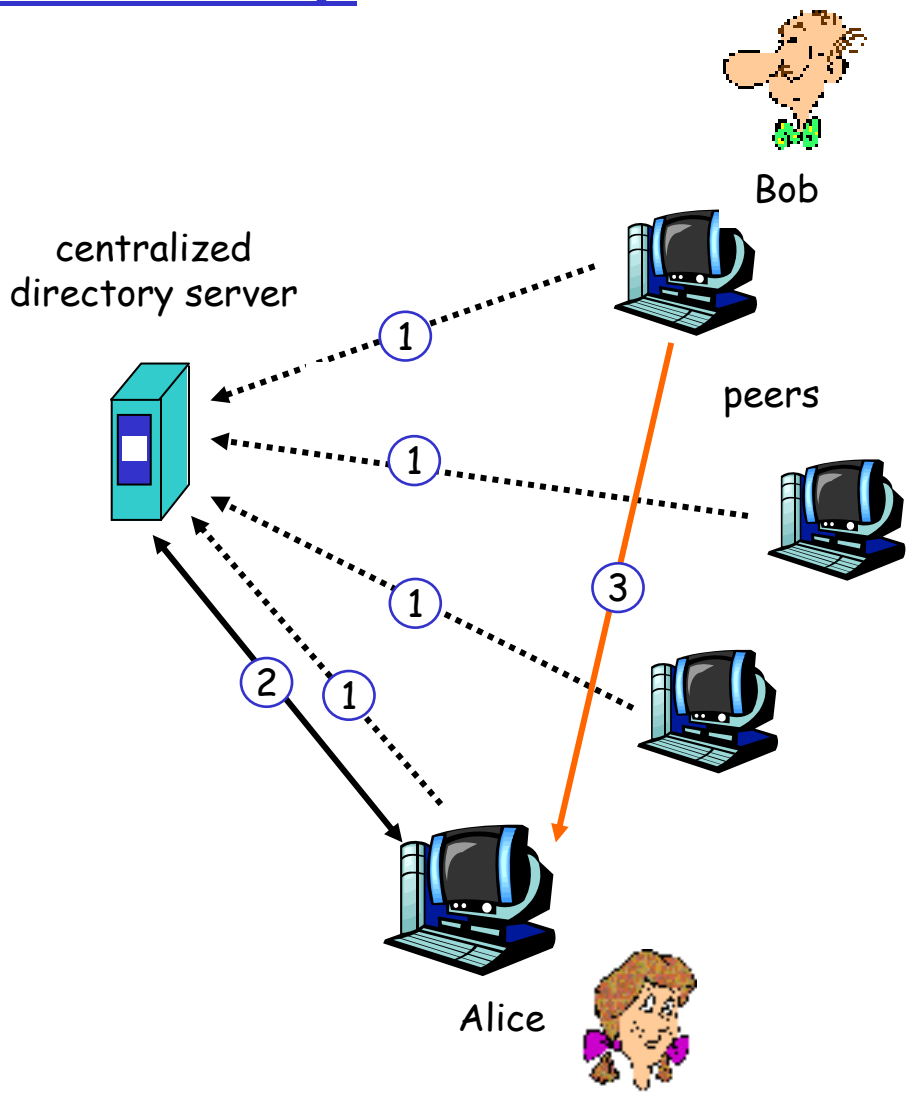
1) when peer connects, it informs central server:
   - IP address
   - Content

   向主機註冊

2) Alice queries for "Hey Jude"

   向主機查詢，主機回覆有檔案的電腦

3) Alice requests file from Bob



P2P: problems with centralized directory

- single point of failure
  單點失效
- performance bottleneck
  效能瓶頸
- copyright infringement: "target" of lawsuit is obvious
  版權問題

file transfer is decentralized, but locating content is highly centralized

Query flooding: Gnutella
查詢漫出：以Gnutella為例

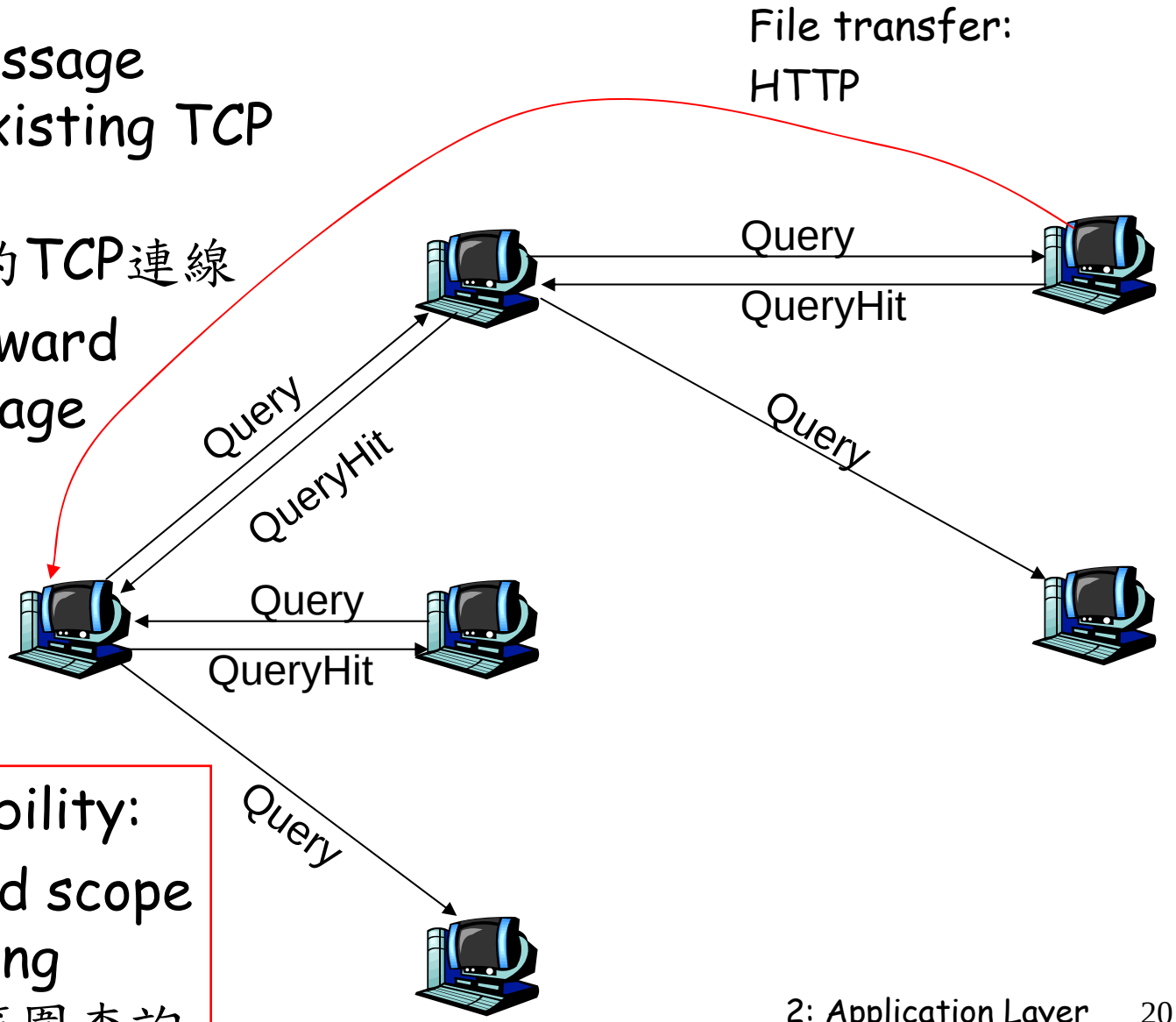
- fully distributed
  - no central server
    完全分散式運作
- public domain protocol
  公開的協定
- many Gnutella clients implementing protocol

overlay network: graph

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

Gnutella: protocol

- Query message sent over existing TCP connections
  透過已存在的TCP連線
- peers forward Query message
  各點幫忙
- QueryHit sent over reverse path



Scalability:
limited scope flooding
有限範圍查詢

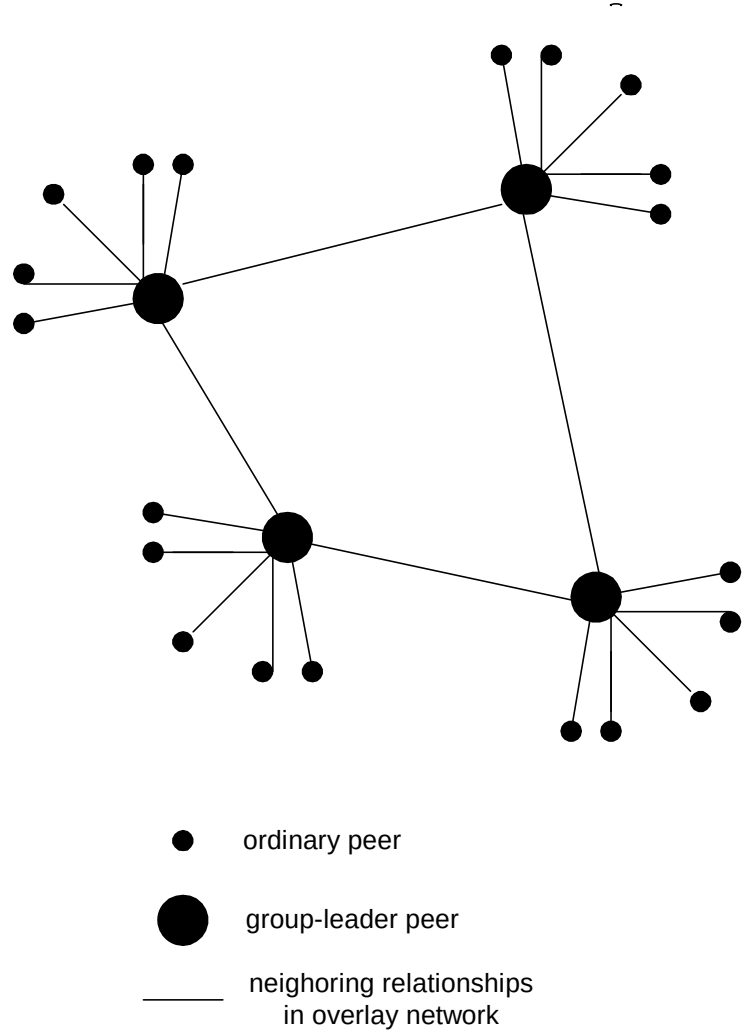
Gnutella: Peer joining 對等點加入

1. joining peer Alice must find another peer in Gnutella network: use list of candidate peers
2. Alice sequentially attempts TCP connections with candidate peers until connection setup with Bob
3. *Flooding:* Alice sends Ping message to Bob; Bob forwards Ping message to his overlay neighbors (who then forward to their neighbors….)
   - peers receiving Ping message respond to Alice with Pong message
4. Alice receives many Pong messages, and can then setup additional TCP connections

Peer leaving: see homework problem!

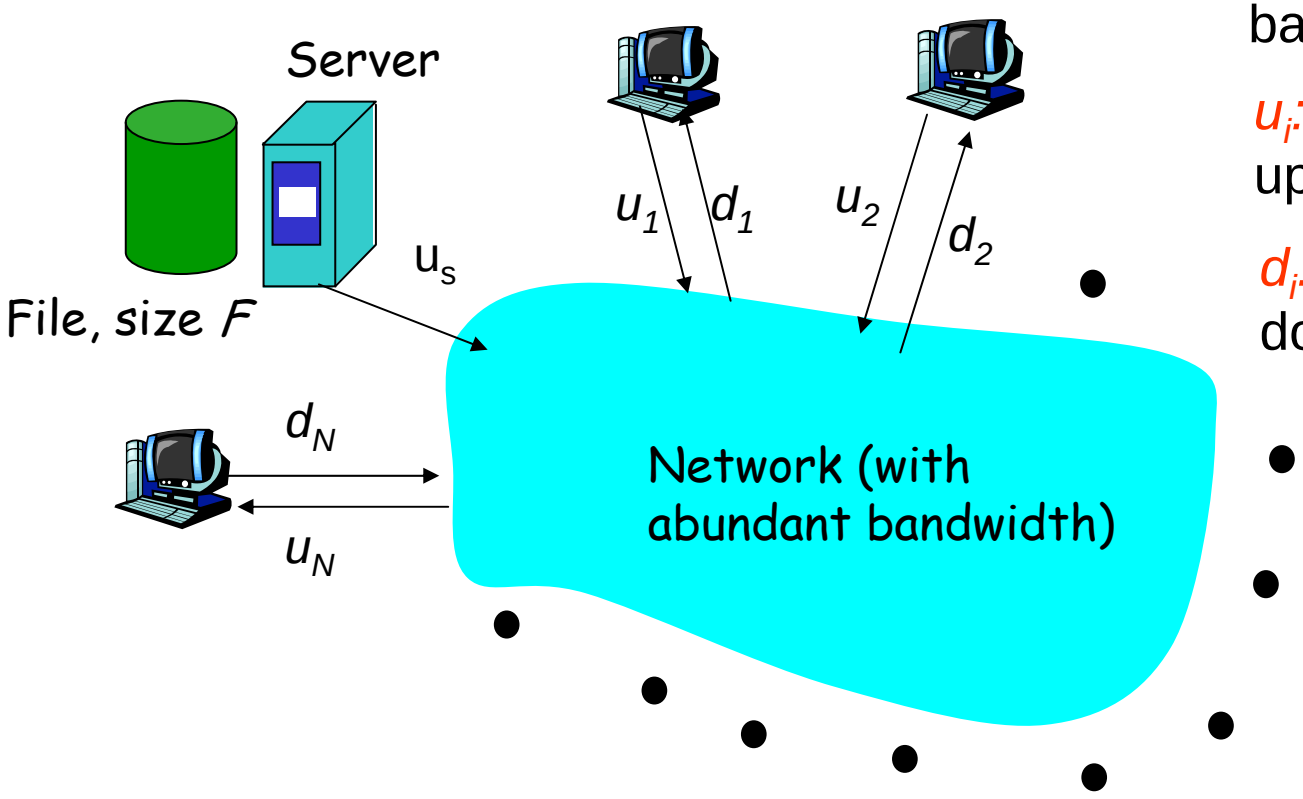
Hierarchical Overlay 階層式重疊

- between centralized index, query flooding approaches
- 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 content in its children
  group leader知道他下游對等點的內容



Comparing Client-server, P2P architectures

Question: How much time distribute file initially at one server to N other computers?
要花多少時間將檔案傳送至N台電腦?



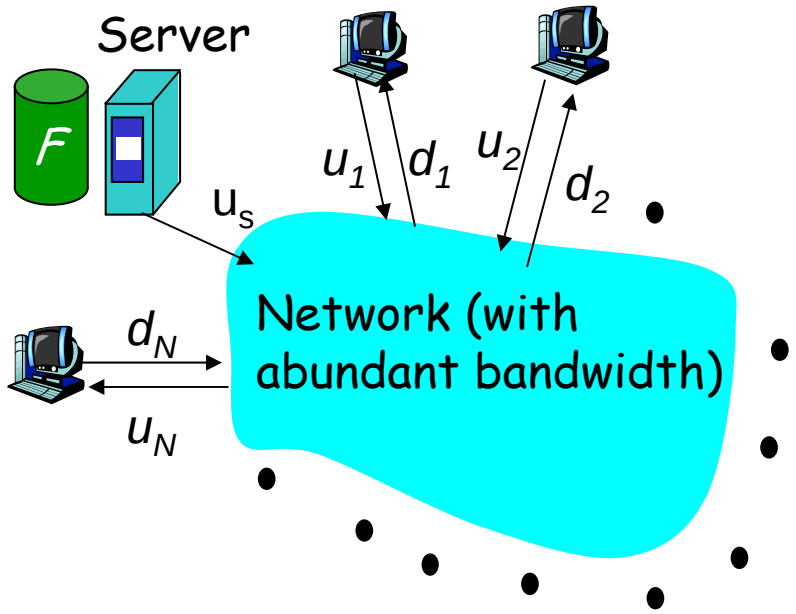
u_s: server upload bandwidth

u_i: client/peer i upload bandwidth

d_i: client/peer i download bandwidth

Client-server: file distribution time

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

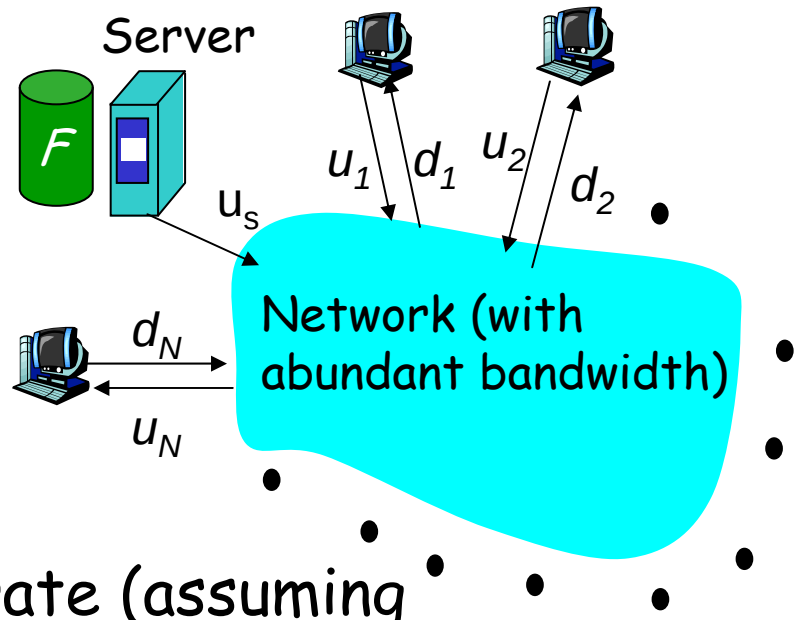


Time to distribute F to N clients using client/server approach $= d_{cs} = \max\{ NF/u_s, F/\min_i(d_i) \}$

increases linearly in N (for large N)

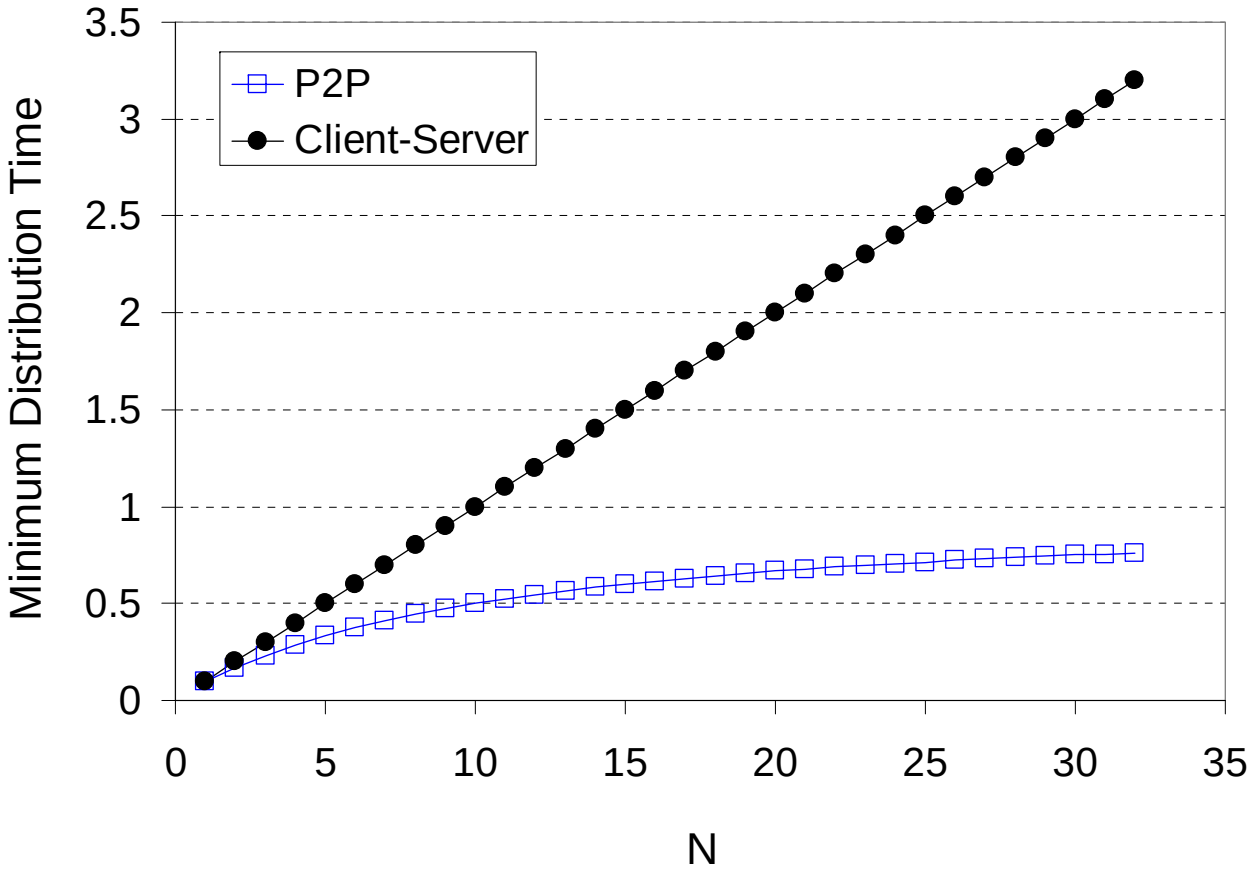
P2P: file distribution time

- 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 (assuming all nodes sending file chunks to same peer): $u_s + \sum u_i$ (i=1,N)



$$d_{P2P} = \max \{ F/u_s, F/\min(d_i)_i, NF/(u_s + \sum u_i)_{i=1,N} \}$$

Comparing Client-server, P2P architectures

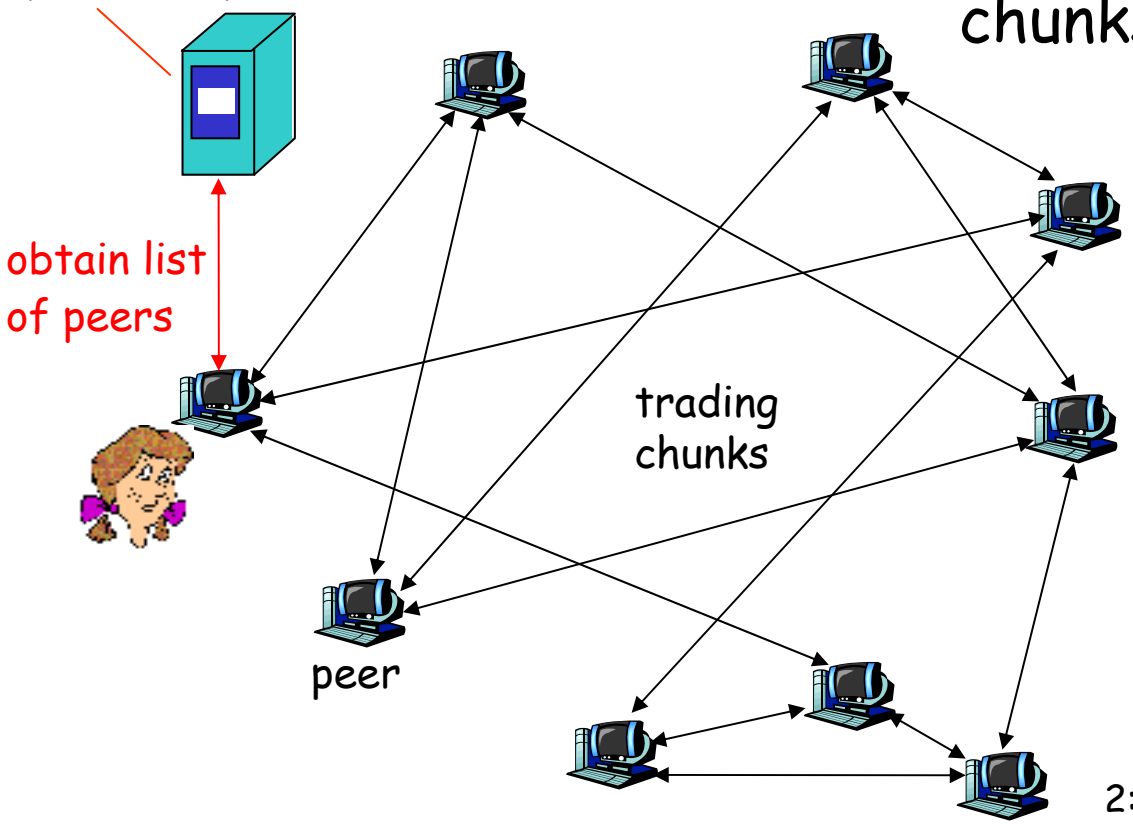


P2P Case Study: BitTorrent

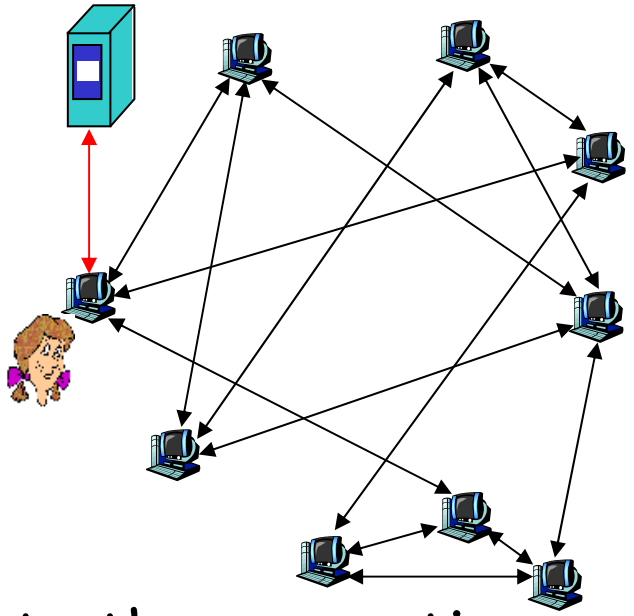
- P2P file distribution

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

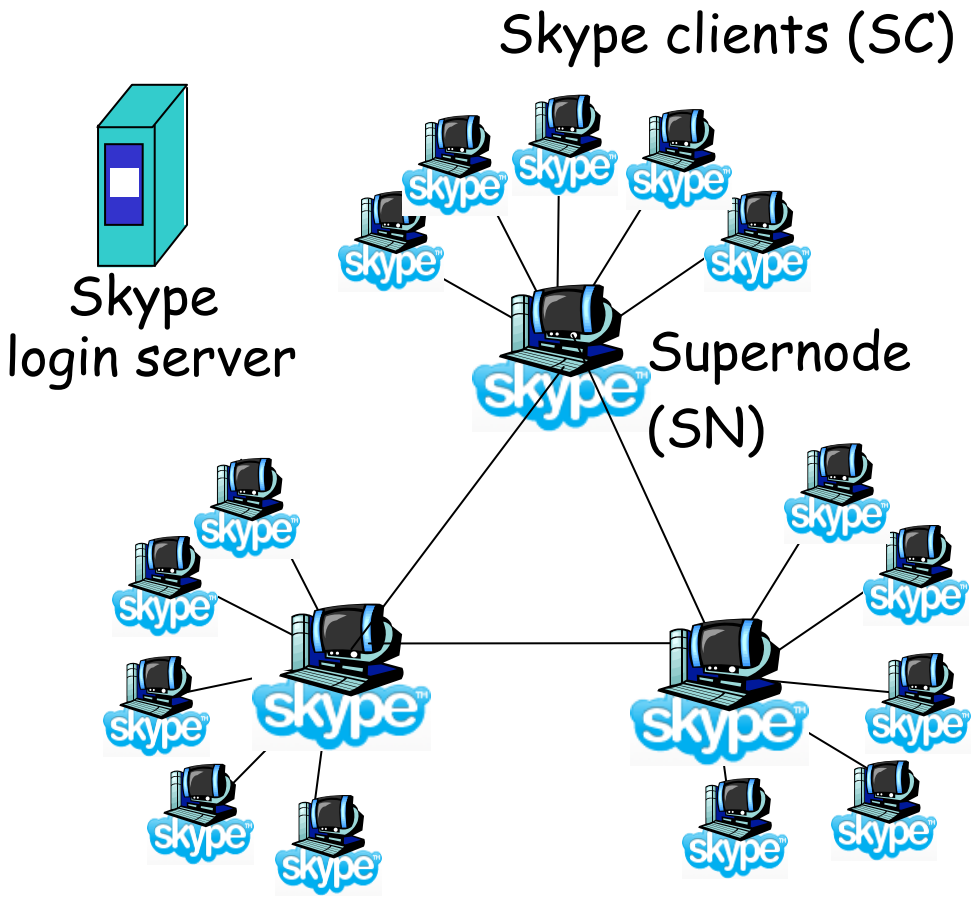
- at any given time, different peers have different subsets of file chunks
- periodically, a peer (Alice) asks each neighbor for list of chunks that they have.
- Alice issues requests for her missing chunks
  - rarest first

Sending Chunks: tit-for-tat

- Alice sends chunks to four neighbors currently sending her chunks *at the highest rate*
  - re-evaluate top 4 every 10 secs
- every 30 secs: randomly select another peer, starts sending chunks
  - newly chosen peer may join top 4

P2P Case study: Skype

- P2P (pc-to-pc, pc-to-phone, phone-to-pc) Voice-Over-IP (VoIP) application
  - also IM
- proprietary application-layer protocol (inferred via reverse engineering)
- hierarchical overlay



Skype: making a call
Skype打電話

- User starts Skype
- SC registers with SN
  - list of bootstrap SNs
- SC logs in (authenticate)
- Call: SC contacts SN will callee ID
  - SN contacts other SNs (unknown protocol, maybe flooding) to find addr of callee; returns addr to SC
- SC directly contacts callee, overTCP