50.012 Networks

Lecture 4: CDN and P2P

2021 Term 6

Assoc. Prof. CHEN Binbin

SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN

Learning objectives

Understand the design of content distribution networks (CDN), including the design objectives, key design considerations, and various approaches.

Understand the peer-to-peer architecture: benefits, challenges, basic operations and incentive mechanism of BitTorrent.

Read: textbook Section 2.4, 2.5, 2.6

Content distribution networks

- *Challenge:* how to stream content (selected from millions of videos) to hundreds of thousands of *simultaneous* users?
  - Recall: they're highly heterogeneous, including locations, connectivity, etc.
- *Option 1:* single, large "mega-server"
  - point of network congestion
  - long path to distant clients
  - multiple copies of video sent over outgoing link (wastage)
  - single point of failure

….quite simply: this solution *doesn't scale*

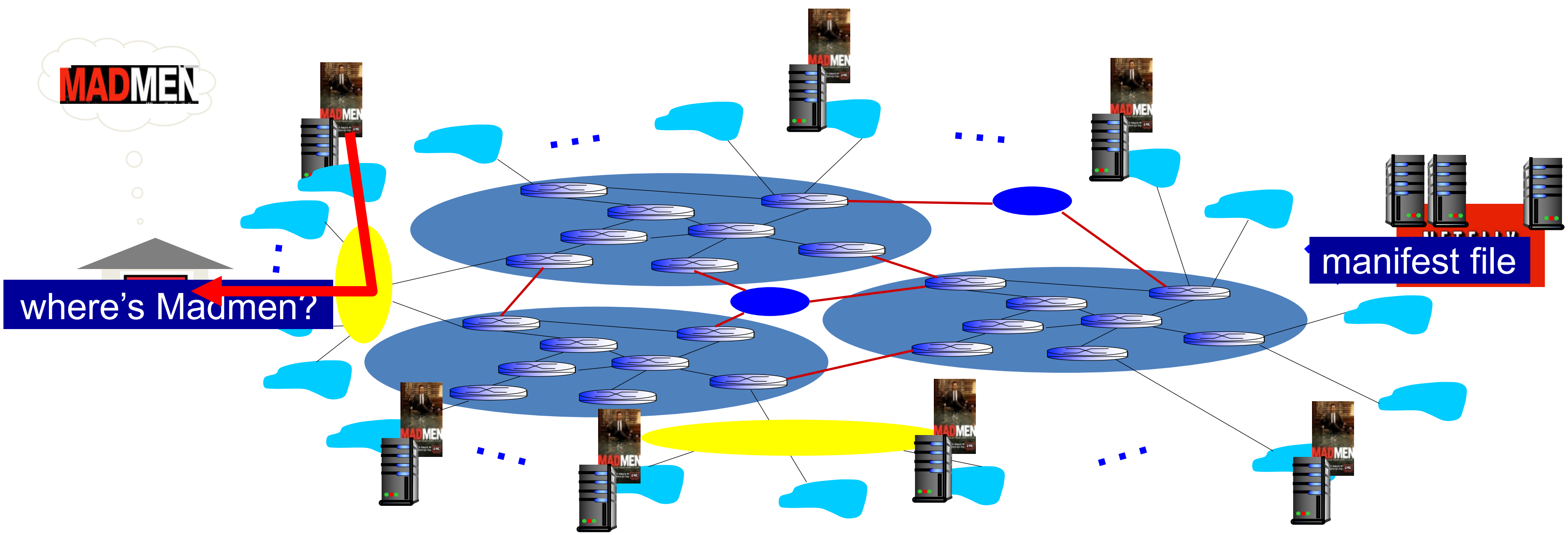
Content distribution networks

- *Challenge:* how to stream content (selected from millions of videos) to hundreds of thousands of simultaneous users?

- *Option 2:* store/serve multiple copies of videos at multiple ***geographically distributed*** sites *(CDN)*

Content distribution networks

- CDN: stores copies of content at CDN nodes
  - e.g., Netflix stores copies of MadMen
- subscriber requests content from Netflix
  - directed to nearby CDN copy (based on mapping IP address of request to geographical location), retrieves content

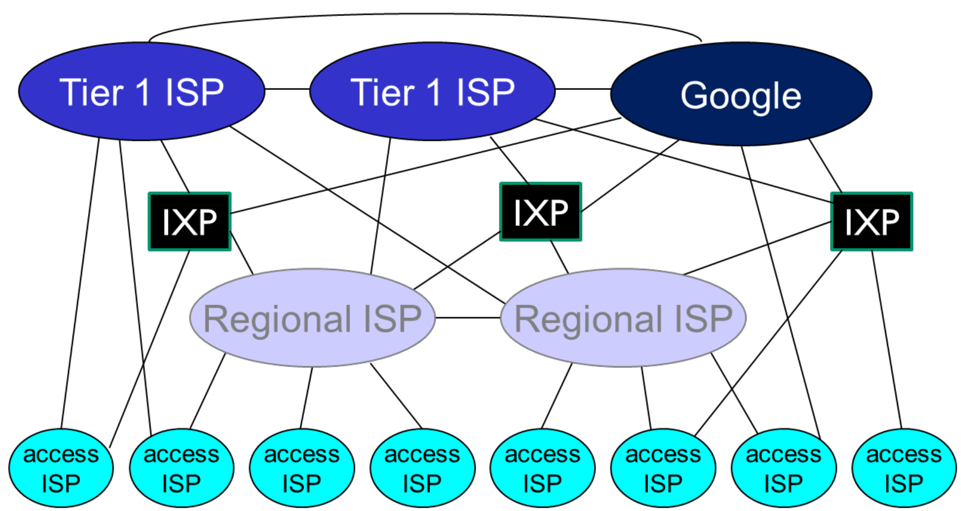


Questions for a CDN operator

- Where to put the servers?
- How to select the server?
- How to route the client to the server?
- How to replicate the content to the servers?

CDN Server Placement

- *Enter deep:* push CDN servers deep into many access ISPs
  - close to users
  - used by Akamai, 1700+ locations
- *Bring home:* smaller number (10's) of larger clusters in IXPs (Internet Exchange Points) near (but not within) access networks
  - used by Limelight



Types of CDN

- Commercial CDN
  - Akamai Technologies, Cloudflare, LimeLight
- Content provider’s own CDN
  - Google, netflix
- Telco CDN

Cluster (Server) Selection

- Geographically close
- Best performance: real-time measurement
- Lowest load: Load-balancing
- Cheapest: CDN may need to pay its provider ISP
- Any alive node: fault-tolerant

Content Access / Routing

- Naming (DNS)–based
- Application-driven
  - E.g., use HTTP redirect
  - Multiple connection setup, name lookups
- Routing (anycast)-based
  - Coarse-grained

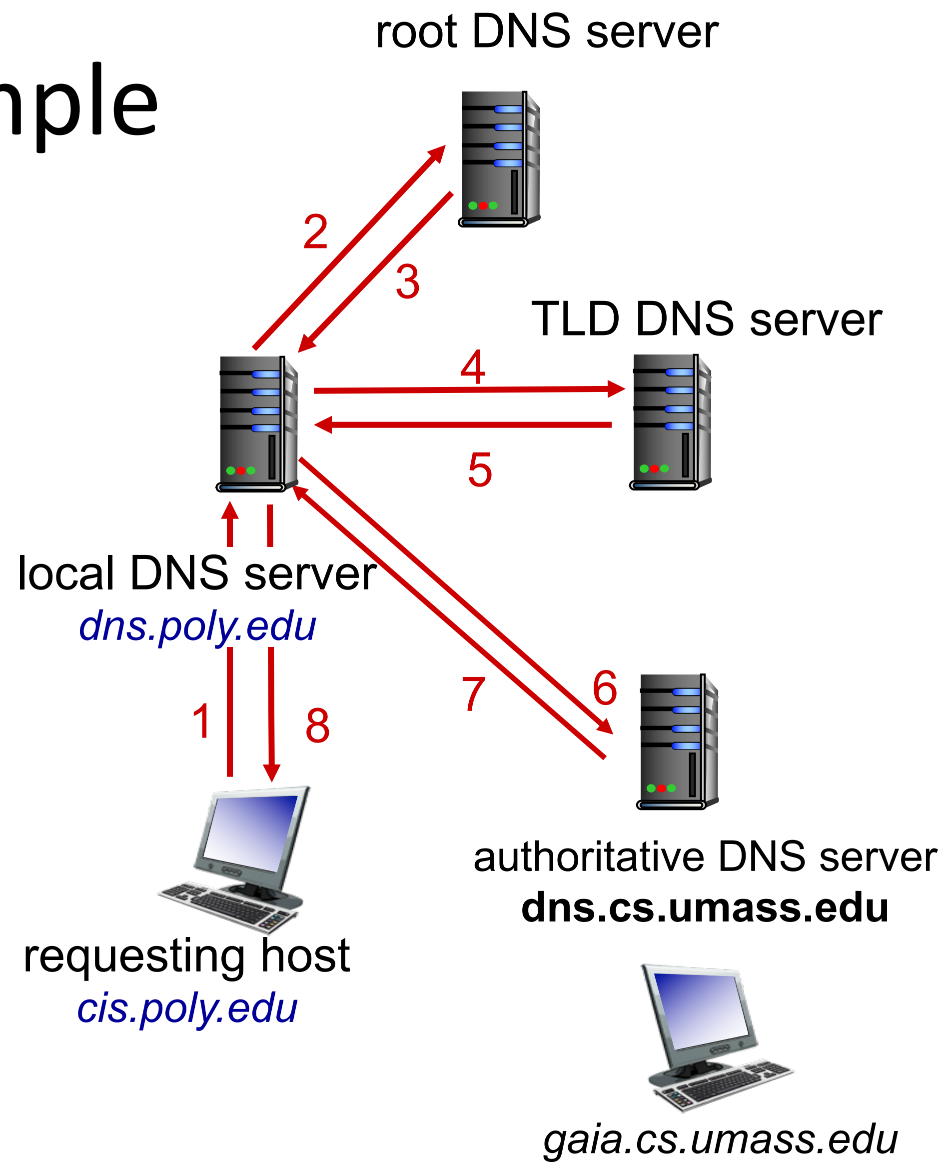
A quick recap of DNS
(Domain Name System)

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"



DNS records

DNS: distributed database 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

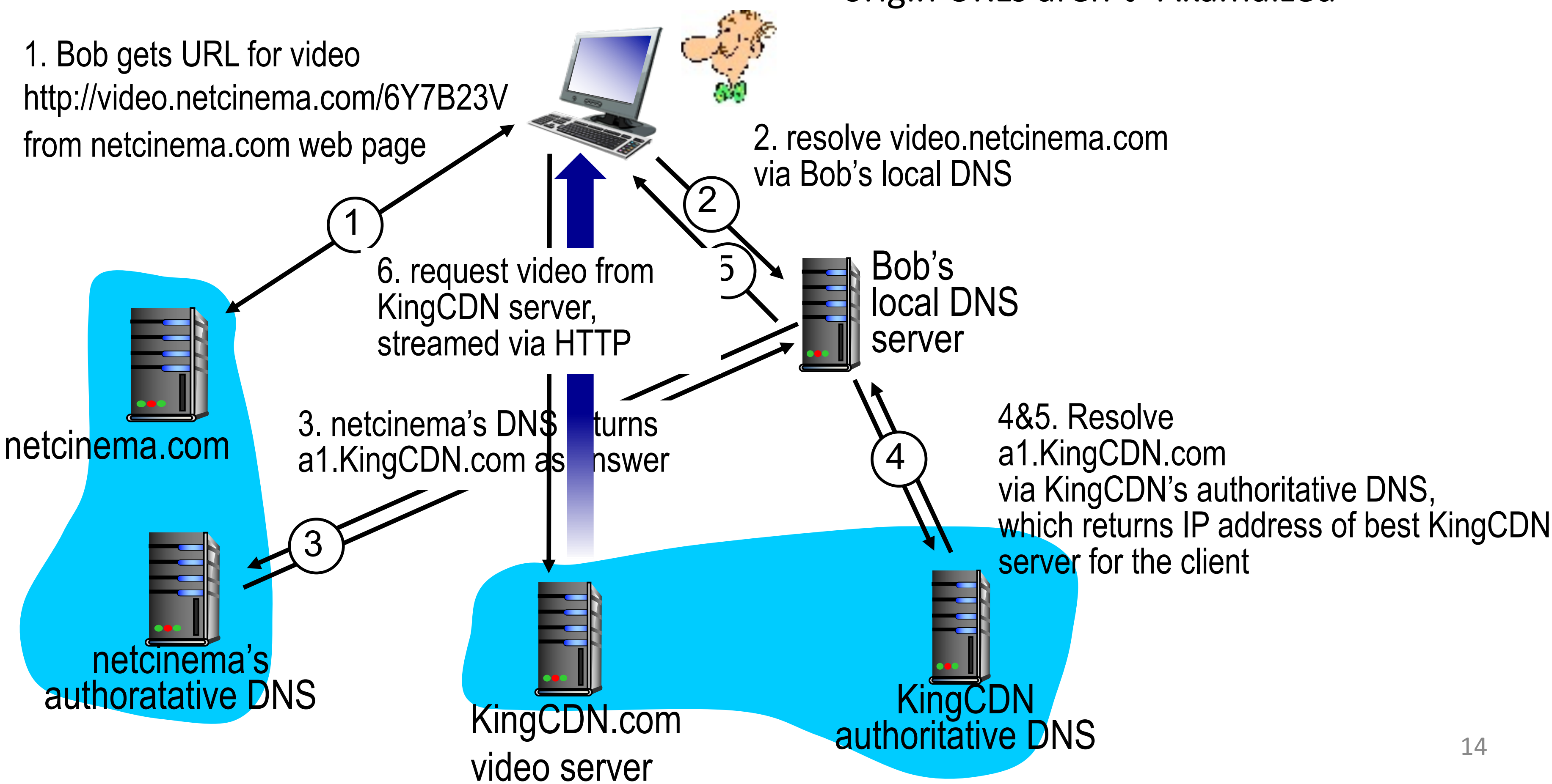
- **`value`** is name of mailserver associated with **`name`**

CDN content access by DNS redirect

Bob (client) requests video http://video.netcinema.com/6Y7B23V

- video stored in CDN run by KingCDN.com

Used by: third-party CDNs like Akamai, if origin URLs aren't "Akamaized"

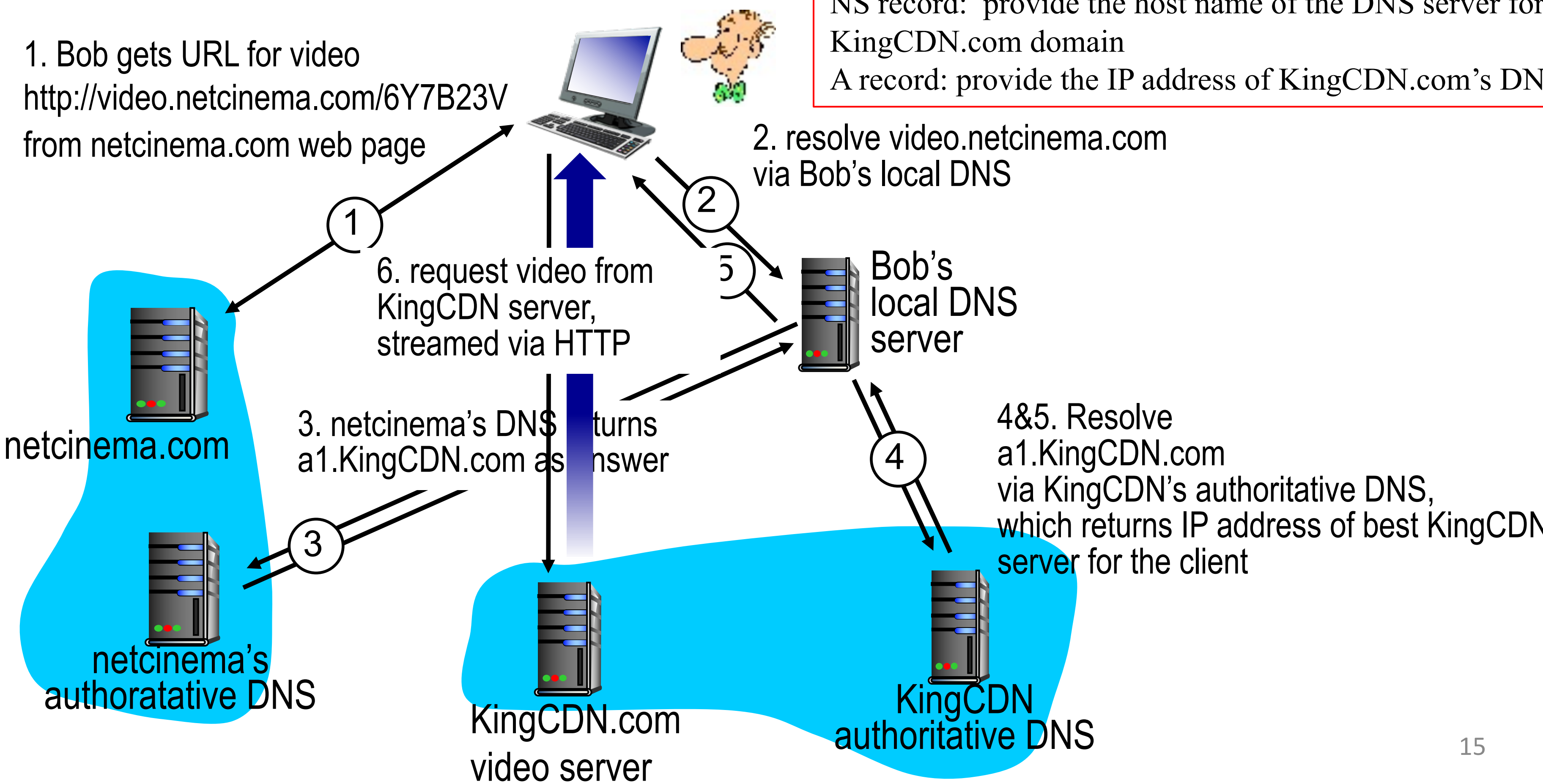


CDN content access by DNS redirect

Bob (client) requests video http://video.netcinema.com/6Y7B23V

- video stored in CDN run by KingCDN.com

The DNS records sent at step 3:
CNAME record: map video.netcinema.com to a1.KingCDN.com
NS record: provide the host name of the DNS server for the KingCDN.com domain
A record: provide the IP address of KingCDN.com's DNS s



Akamai Resource Locators (ARL)

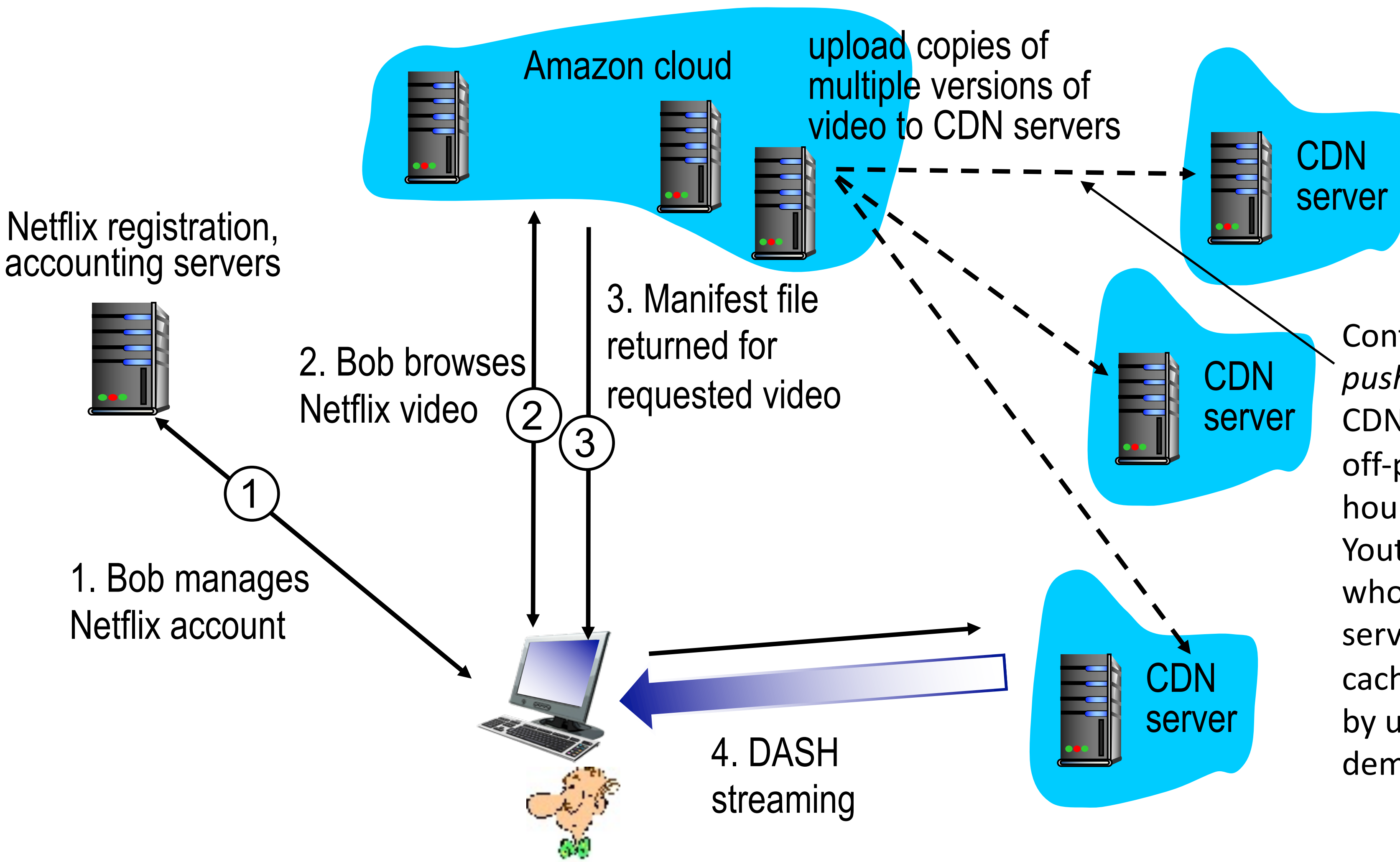
- Each customer has its dedicated domain name
  - E.g., a123.g.akamaitech.net
  - Akamai operates the authoritative DNS servers for these names
- The original server can "akamaize" the origin URL to ARL (Akamai Resource Locators)
  - E.g.,: <img src="http://sutd.edu.sg/img/welcome.gif"> becomes <img src ="http://a123.g.akamaitech.net/7/123/456/xobjdatax/sutd.edu.sg/img/welcome.gif">
- Client browser issues GET to CDN instead of origin server
- Read more:
  Akamai "Fast Internet Content Delivery with FreeFlow"

Case study: Netflix

- Netflix used to rely on third-party CDNs
  – Akamai, LimeLight, level-3
- It has built its private CDN for delivering videos
  – Still rely on Akamai for delivering web pages
  – Server placements: bring home (50+ IXP) + enter deep (free rack of servers for access ISPs)
- Netflix rack: 1 (access ISP) to 10+ (IXP) servers
  – Each server has multiple 10Gbps Ethernet ports, 100 Terabytes+ storage

Case study: Netflix

NB. CDN is owned (hence directly controlled) by Netflix: no need for previous DNS redirect



Content *pushed* to CDN during off-peak hours (cf. Youtube, whose CDN servers *pull* & cache content by user demand)

Content replication

	Mechanism	Pros.	Cons.
Netflix	Push		
Youtube	Pull		

Why the difference in their design?

Use of CDN

- DDoS Attack mitigation
- Handle flash crowds
- Web application firewalls (WAF)

Outline

Content distribution networks (CDN)

Peer-to-peer architecture

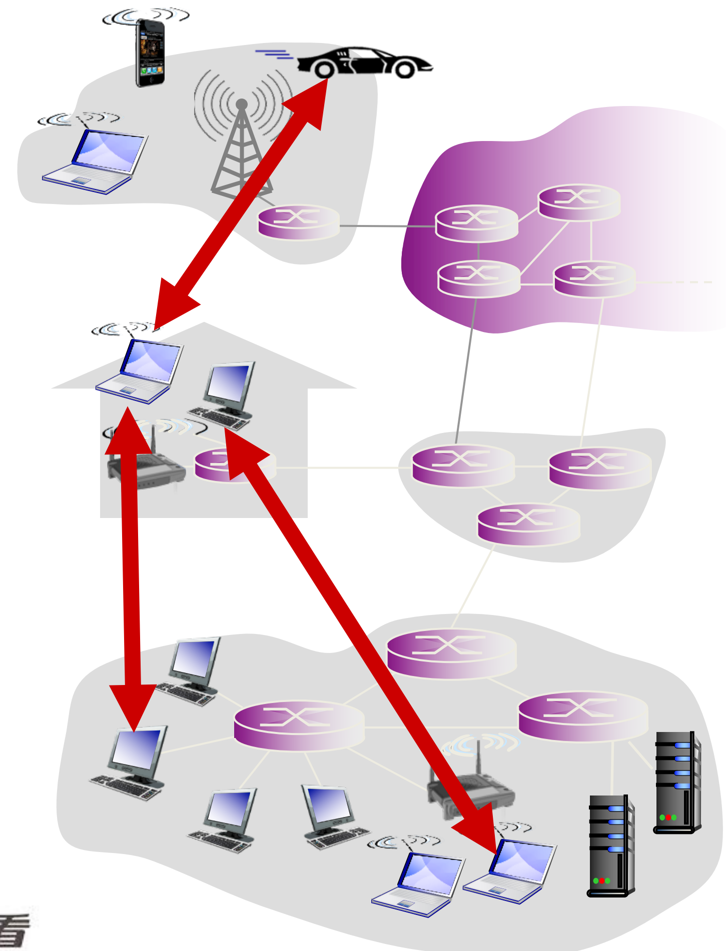
Pure P2P architecture

- *No* always-on server
- Arbitrary end systems (e.g., our laptops) directly communicate as peers
- Peers are intermittently connected and may change IP addresses

Examples:

- File (e.g., video) distribution (BitTorrent)
- Multimedia streaming (KanKan)
- VoIP (Skype)

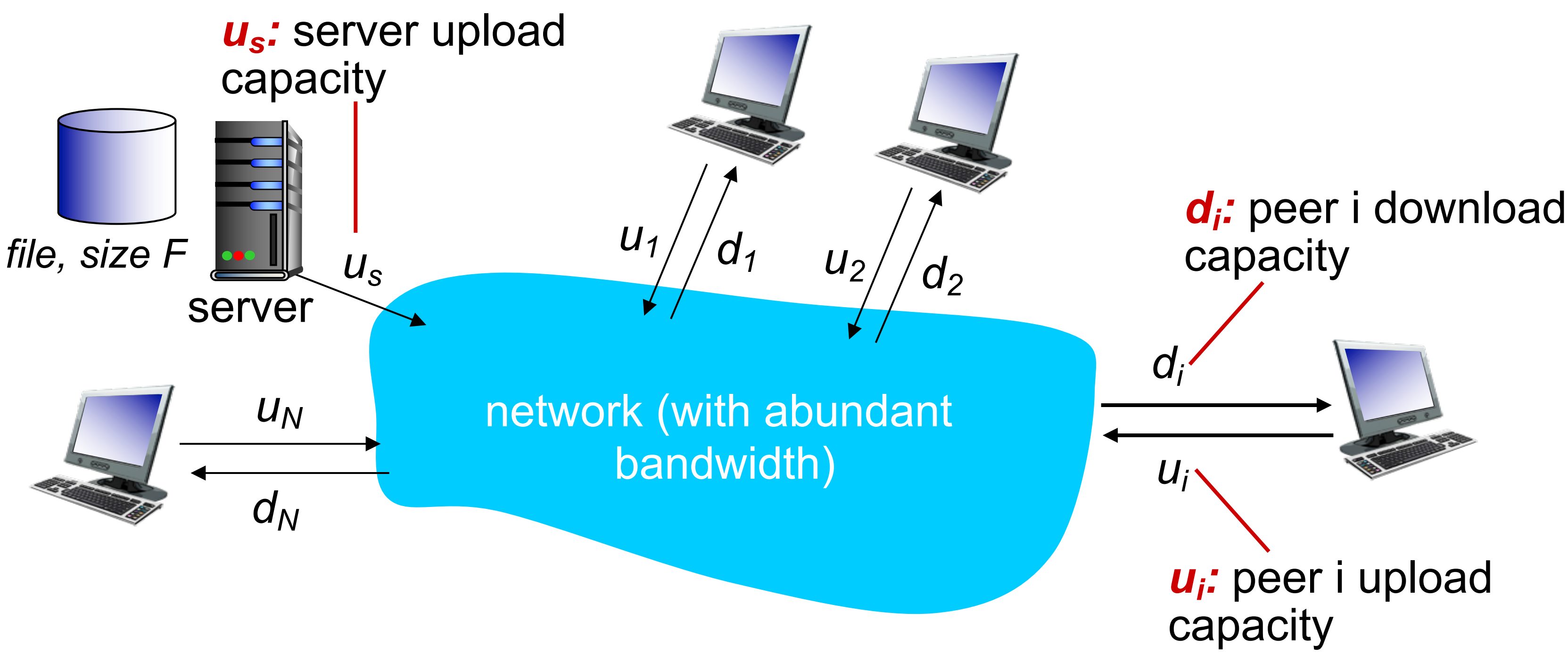
迅雷看看
www.kankan.com
网 络 高 清 影 院



File distribution: client-server vs P2P

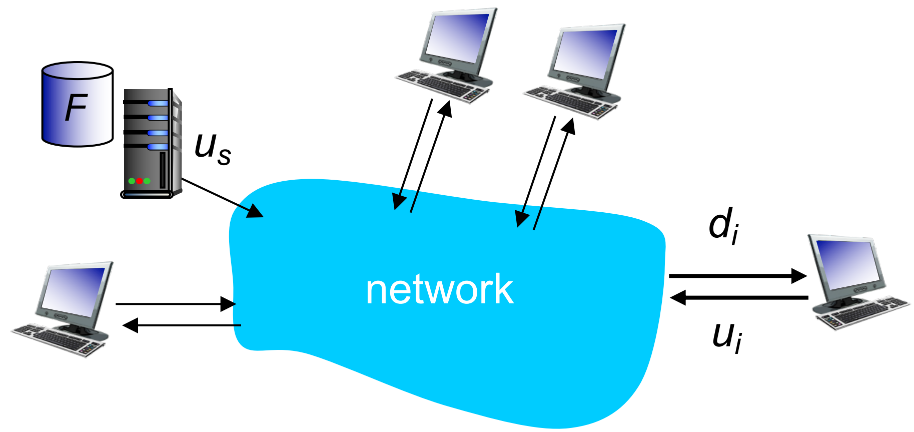
Question: How much time to distribute file (size *F*) from one server to *N* peers?

- peer upload/download capacity (access bandwidth) is limited resource, compared with core bandwidth



File distribution time (lower bound): client-server

- *server transmission:* must sequentially send (upload) N file copies:
  - time to send one copy: F/u_s
  - time to send N copies: NF/u_s
- *client:* each client must download file copy
  - d_{min} = min client download rate
  - min client download time: F/d_{min}



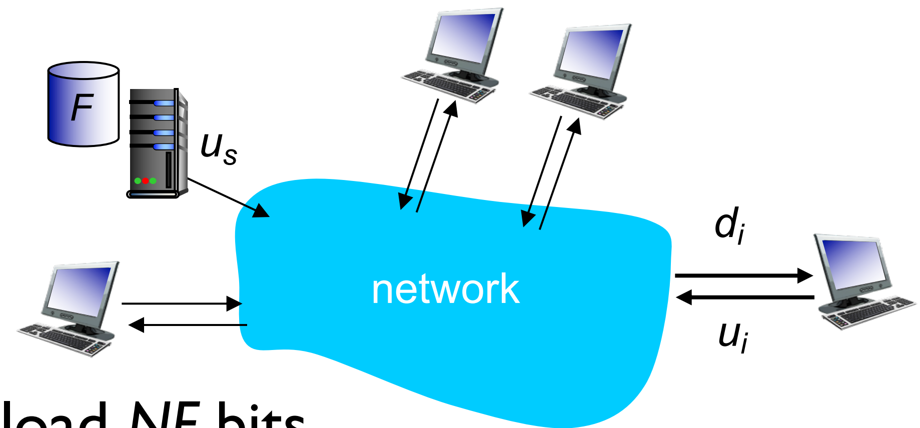
time to distribute F to N clients using client-server approach

$$D_{c\text{-}s} \geq max\{NF/u_s\,,F/d_{min}\}$$

increases linearly in N; susceptible to "flash crowds" particularly

File distribution time (lower bound): P2P

- *server transmission:* must upload at least one copy
  - time to send one copy: F/u_s



- *client:* each client must download a file copy
  - min client download time: F/d_{min}
- *clients:* as aggregate must download *NF* bits
  - implies upload of the same number of bits (*why?*)
  - max upload rate is $u_s + \Sigma u_i$ (if it's feasible to schedule the distribution to make all the uploads concurrently active)

time to distribute F to N clients using P2P approach

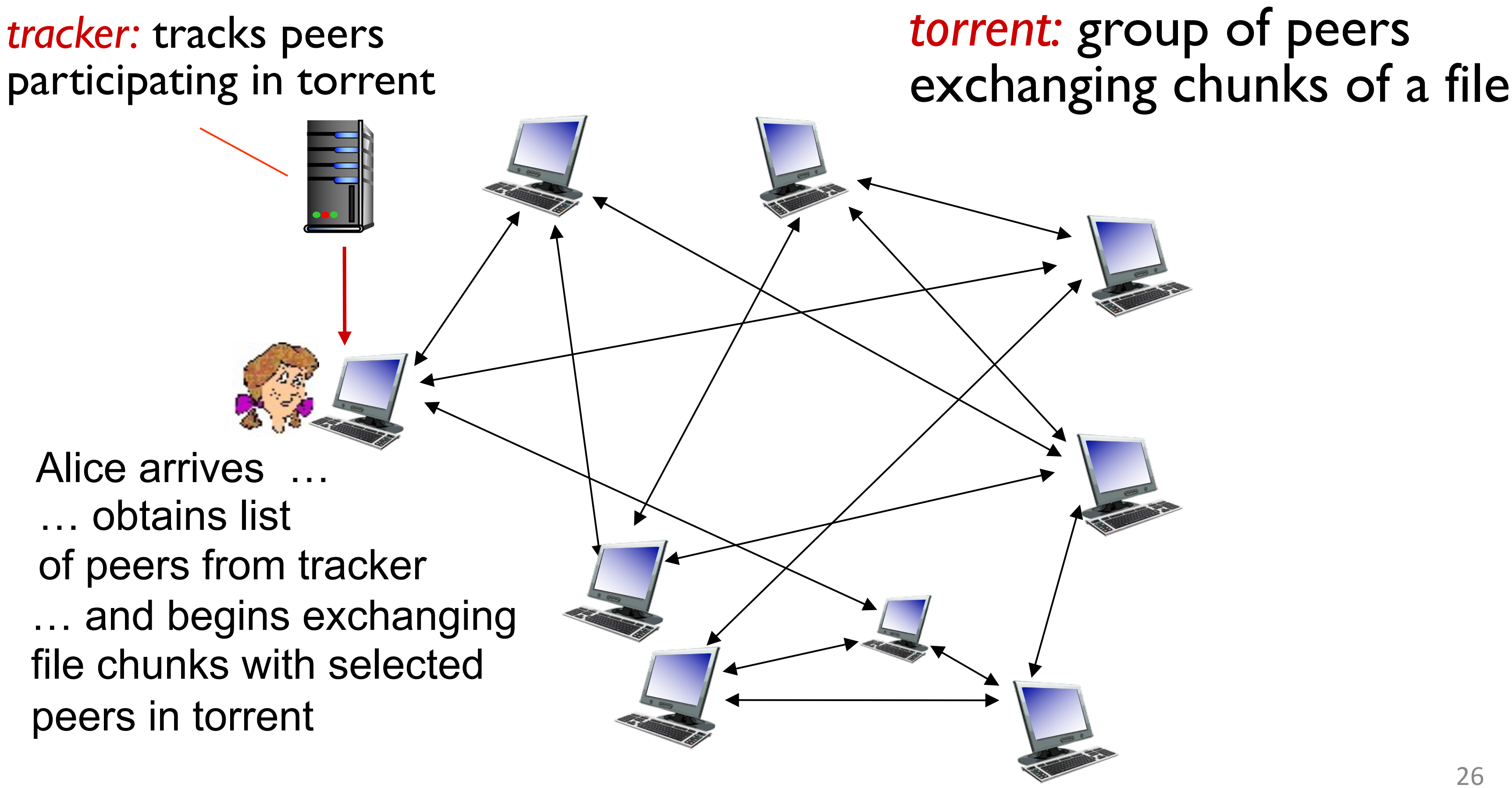
$$D_{P2P} \geq max\{F/u_s, F/d_{min}, NF/(u_s + \Sigma u_i)\}$$

increases linearly in *N* …

… but so does this, as each peer brings service capacity

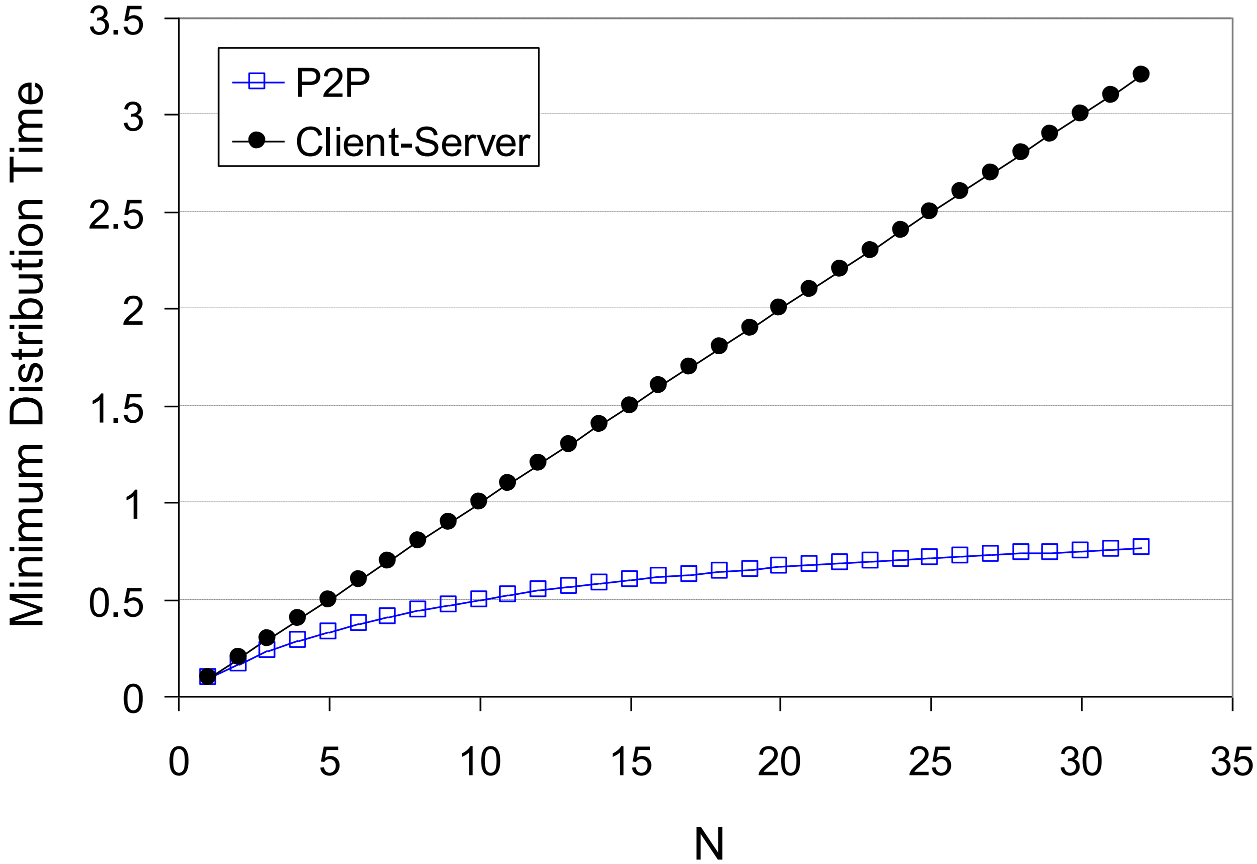
P2P file distribution: BitTorrent

- file divided into 256 kB chunks
- peers in torrent send/receive (trade) file chunks



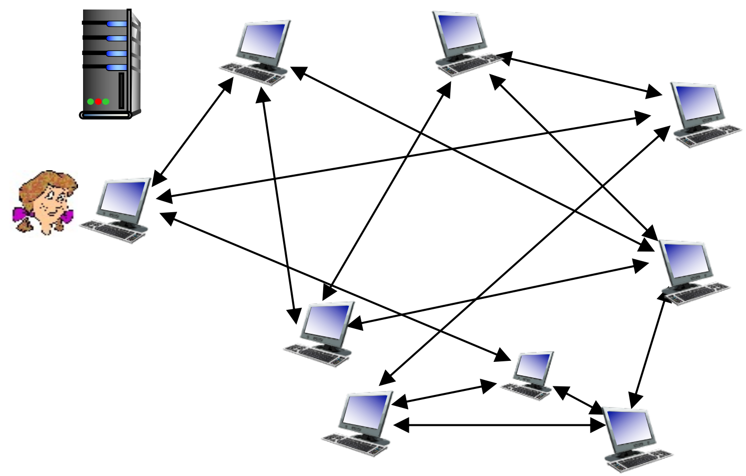
Client-server vs. P2P: example

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



P2P file distribution: BitTorrent

- Peer joining torrent:
  - initially has no chunks, but will accumulate them over time from other peers
  - registers with tracker to get list of peers, connects to subset of peers ("neighbors")



- While downloading, peer uploads chunks to other peers
- Peer may change peers with whom it exchanges chunks
- *Churn:* peers may come and go (highly dynamic & organic, *no a priori* set up reliable infrastructure)
- Once peer has entire file, it may (selfishly) leave or (altruistically) remain in torrent
  - peers ultimately interested in download
  - In general need *incentives* for them to provide upload (e.g., tit-for-tat)

BitTorrent: requesting, sending file chunks

Requesting chunks:

- at any given time, different peers have different subsets of file chunks
- periodically, Alice asks each peer for list of chunks that they have
- Alice requests missing chunks from peers, rarest first
  - Aim to make all the chunks equally available (they are equally important)

Sending chunks: tit-for-tat

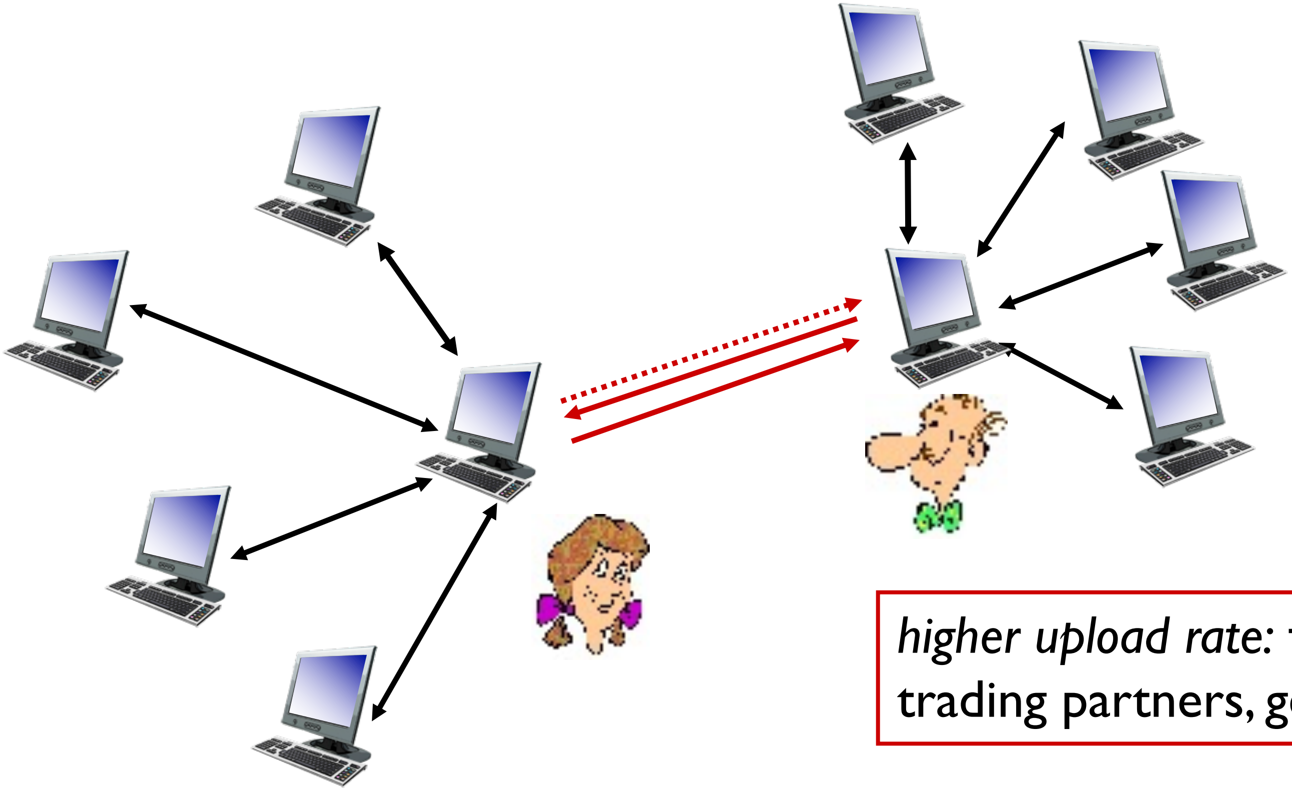
- Alice sends chunks to those four peers currently sending her chunks *at highest rate*
  - these 4 are "unchoked"
  - other peers are choked by Alice (do not receive chunks from her)
  - re-evaluate top 4 every10 secs
- every 30 secs: randomly select another peer, starts sending chunks
  - optimistically "unchoke" this peer: give chance of discovering an even better trading partner than the original top 4
  - newly chosen peer may join top 4

BitTorrent: tit-for-tat

(1) Alice "optimistically unchokes" Bob

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

(3) Bob in turn becomes one of Alice's top-four providers



higher upload rate: find better trading partners, get file faster!