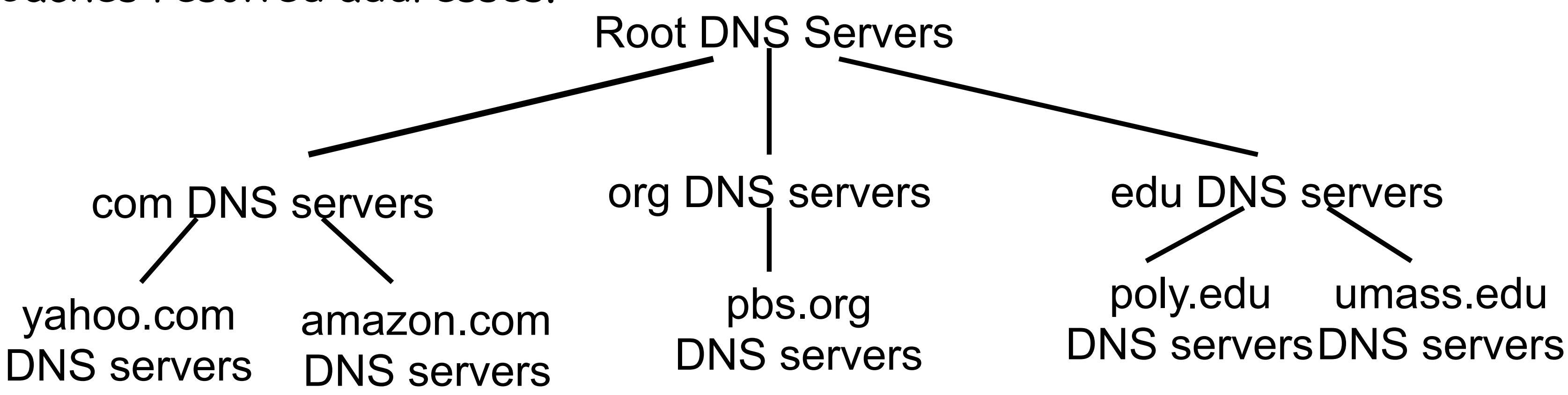
Domain Name System (DNS)

RFC 1034/1035

- The Internet identifies hosts by their IP address (e.g. 129.127.12.6)
- People tend to prefer names (e.g. www.cs.adelaide.edu.au)
- There are ***lots*** of Internet hosts. A centralised database wouldn't scale. The DNS is a distributed database.
  - organisations can change host names and IP addresses within their domain without informing a central authority
  - an organisation will typically have a name server
- A DNS server provides *name resolution* = conversion from a domain name to an IP address
  - a name server is a process listening on UDP/TCP port 53 for requests
  - when detected, the name is resolved, and a reply is sent
- Lots of applications are DNS clients, including web browsers.

DNS hierarchical database

- Root name server
  - (13 worldwide)
  - able to resolve all queries or identify another intermediate name server
- Top-level domain (TLD) servers:
  - 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:
  - 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 - handles local DNS requests. Must know at least one root server. Caches resolved addresses.



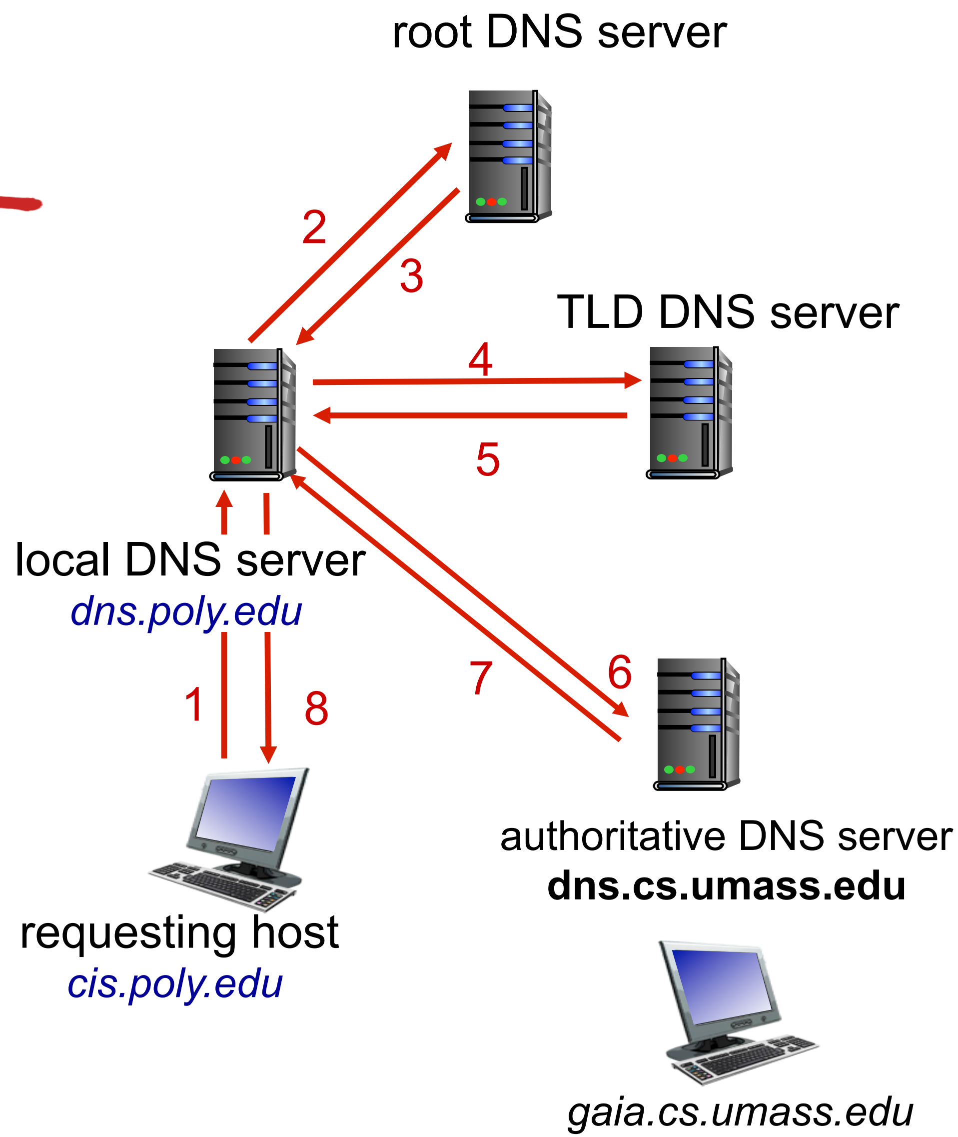
DNS name resolution example

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

iterated query:

❖ contacted server replies with name of server to contact

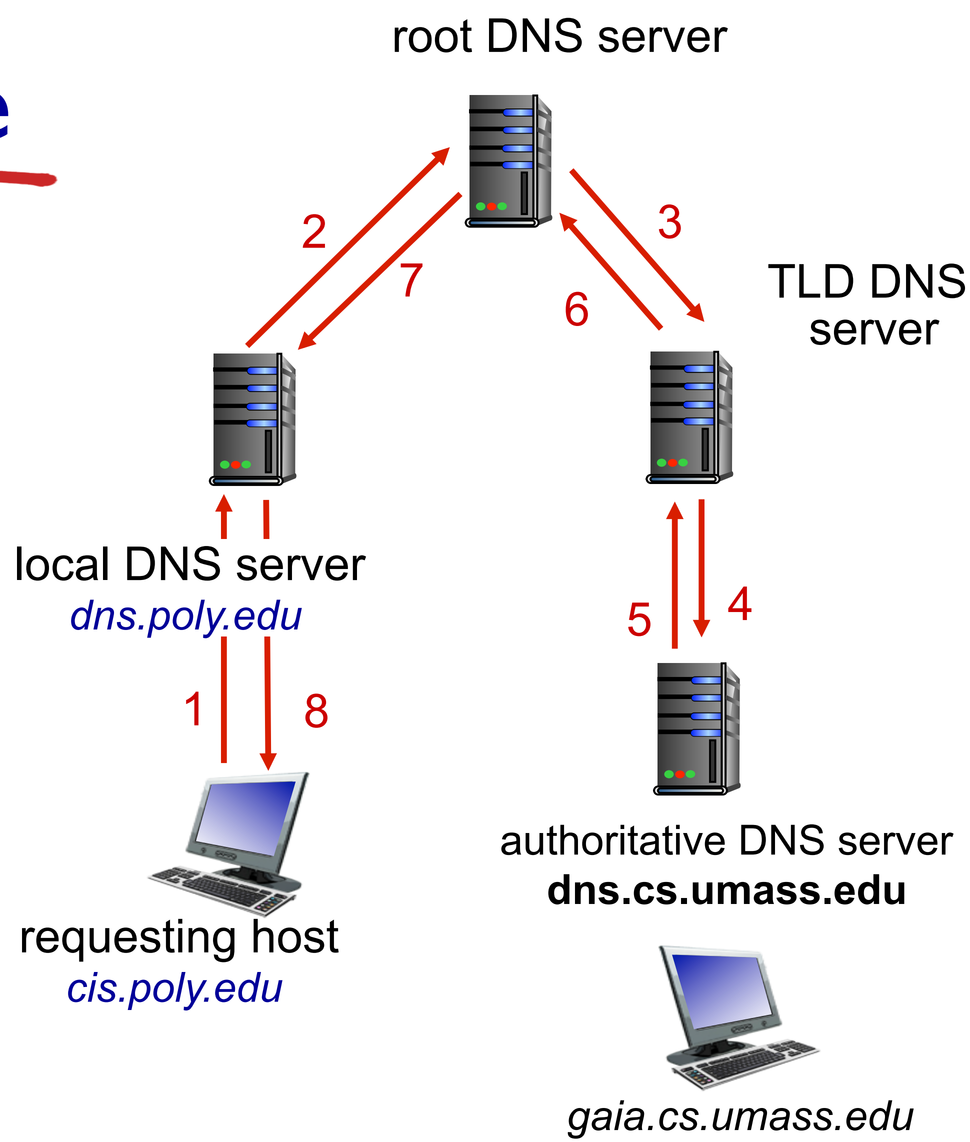
❖ "I don't know this name, but ask this server"



DNS name resolution example

recursive query:

- puts burden of name resolution on contacted name server
- heavy load at upper levels of hierarchy?



DNS records

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

DNS: distributed db storing resource records (RR)

- 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**

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:

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

- Put in authoritative server 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?

Attacking DNS

DDoS attacks

- Bombard root servers with traffic
  - Not successful to date
  - Traffic Filtering
  - Local DNS servers cache IPs of TLD servers, allowing root server bypass
- Bombard TLD servers
  - Potentially more dangerous

Redirect attacks

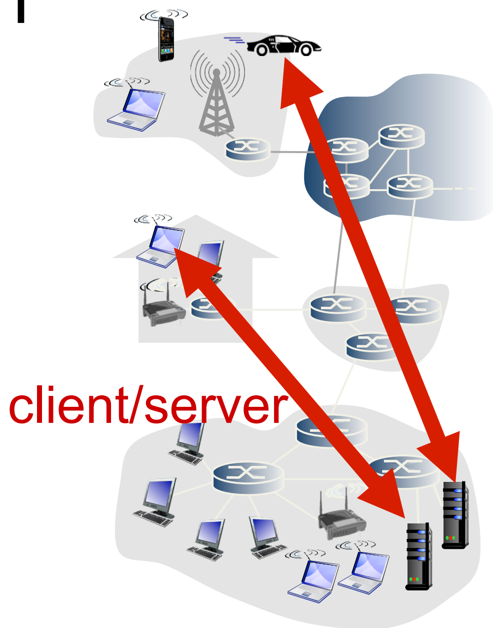
- Man-in-middle
  - Intercept queries
- DNS poisoning
  - Send bogus relies to DNS server, which caches

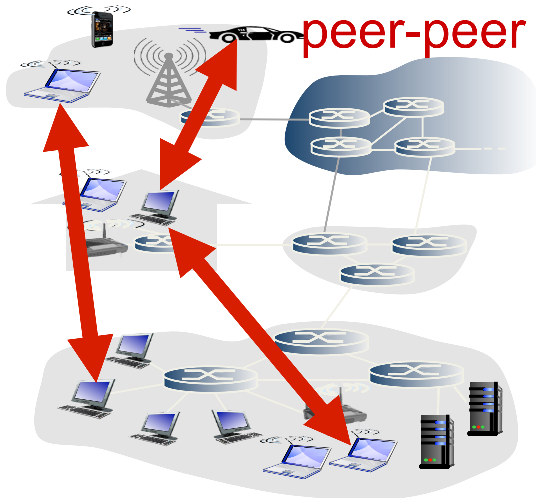
Exploit DNS for DDoS

- Send queries with spoofed source address: target IP
- Requires amplification

Models of Interaction

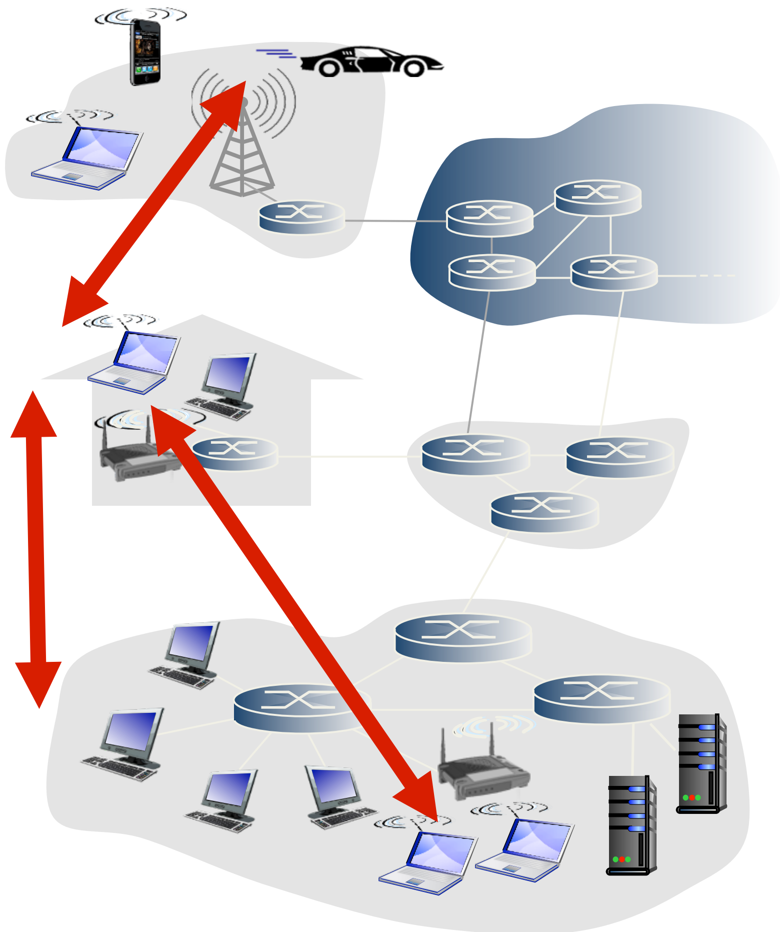
- Client - Server
  - central storage of information in always on server
  - distinction between client which receives service and server which provides service
  - note that it is possible for a host to act as both a client and as a server in different interactions.
  - Web, e-mail, FTP
- Peer to Peer
  - distributed storage of information
  - no clear distinction between clients and servers. Hosts share typically equal control of processing and data
  - Peers dynamically join and leave
  - Bit Torrent





Pure P2P architecture

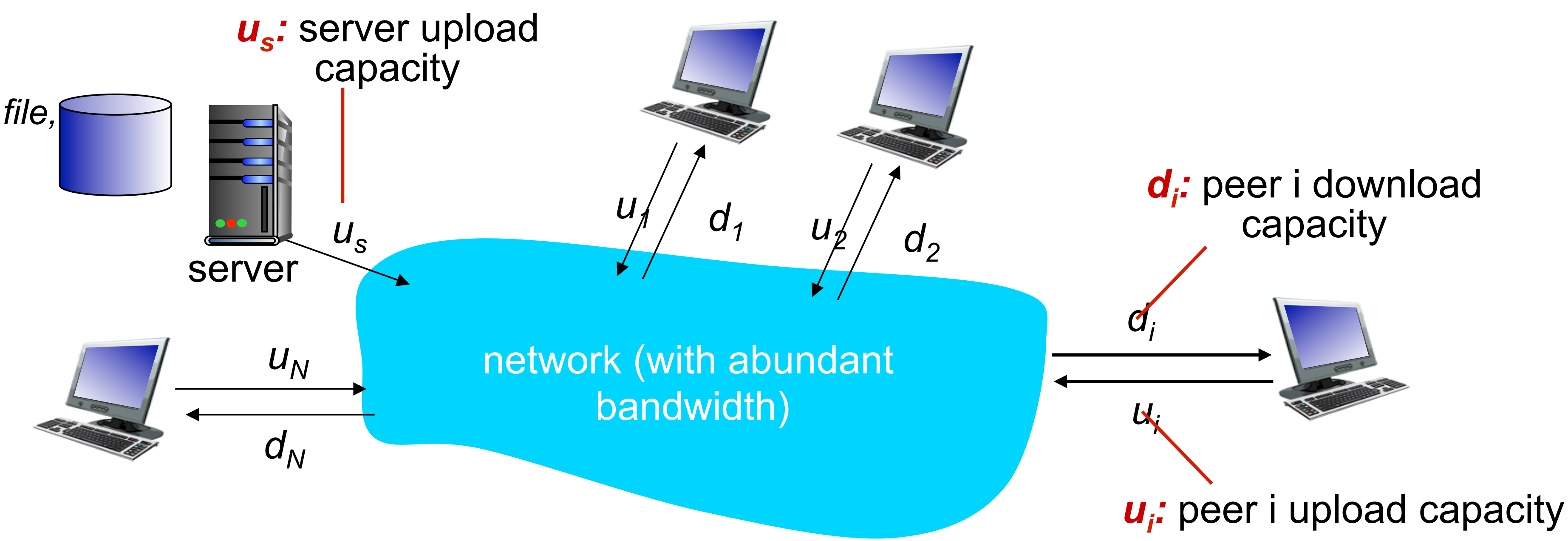
- *no* always-on server
- arbitrary end systems directly communicate
- peers are intermittently connected and change IP addresses
- Advantages
  – Distributes load of serving files.
- Challenges
  – How to find resources
  – Fairness
- *examples:*
  – file distribution (BitTorrent), Streaming (KanKan), VoIP (Skype)



File distribution: client-server vs P2P

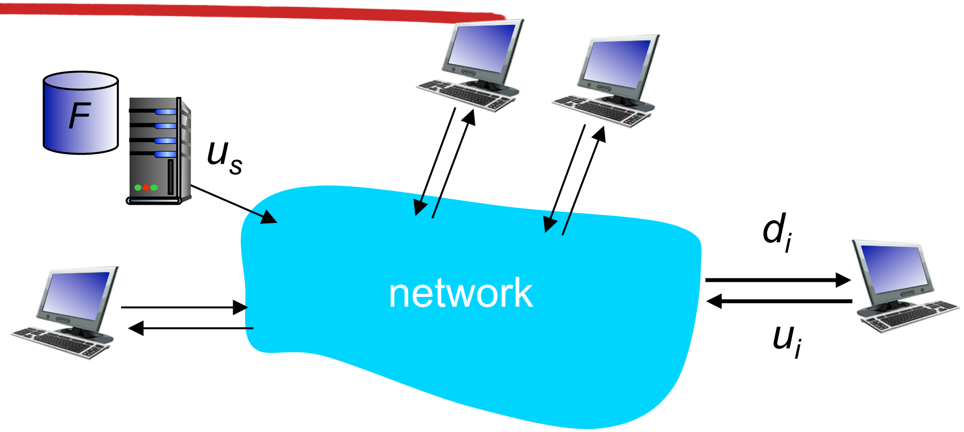
Question: how much time to distribute file (size *F*) from one server to *N* *peers*?

– peer upload/download capacity is limited resource



File distribution time: 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}

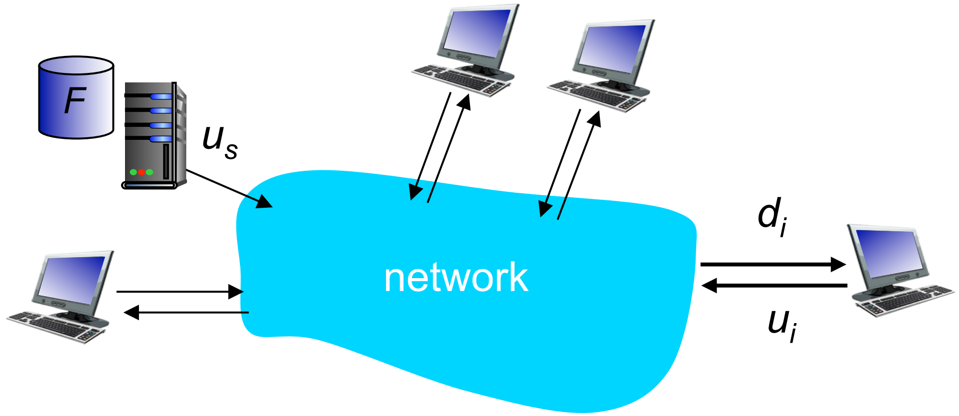
time to distribute F to N clients using client-server approach

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

increases linearly in N

File distribution time: P2P

- *server transmission:* must upload at least one copy
  - time to send one copy: F/u_s
- *client:* each client must download file copy
  - min client download time: F/d_{min}



- *clients:* as aggregate must download *NF* bits
  - max upload rate (limting max download rate) is $u_s + \Sigma u_i$

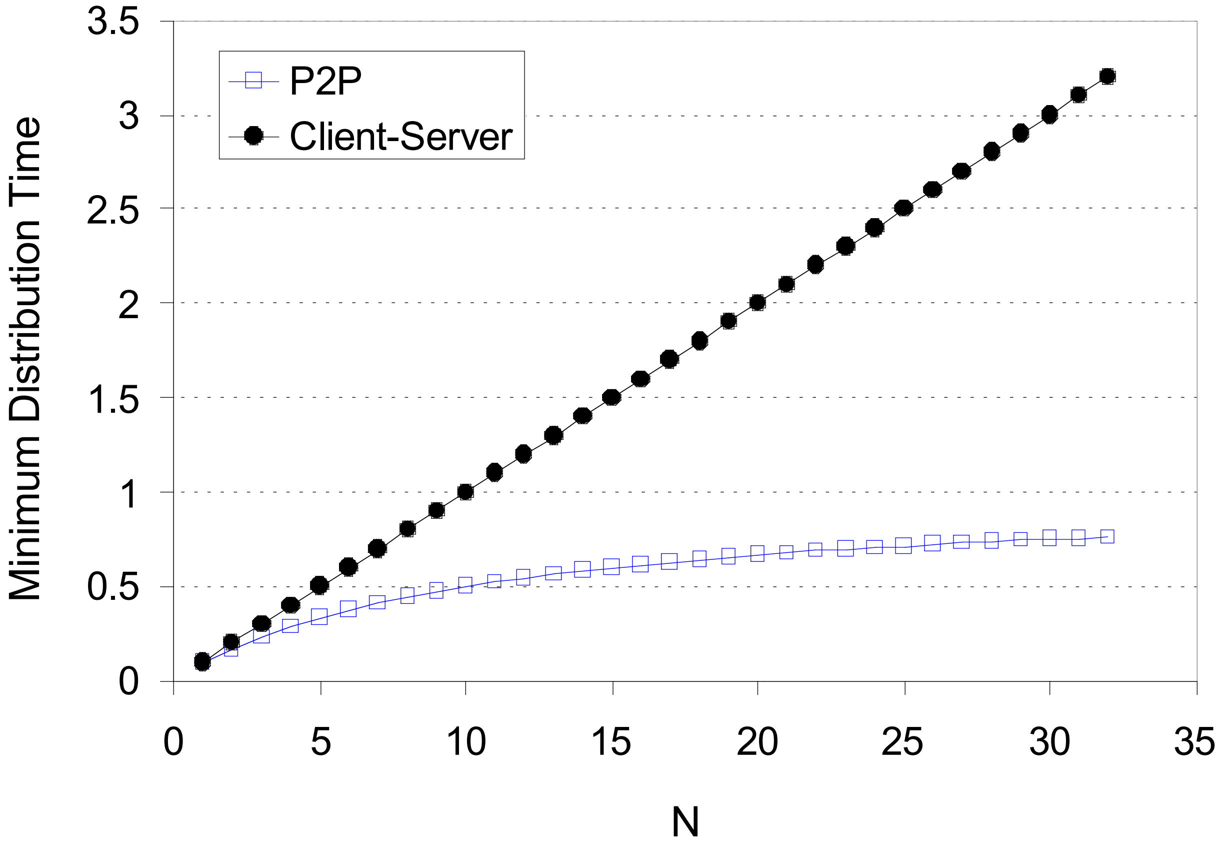
time to distribute F to N clients using P2P approach

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

increases linearly in *N* …

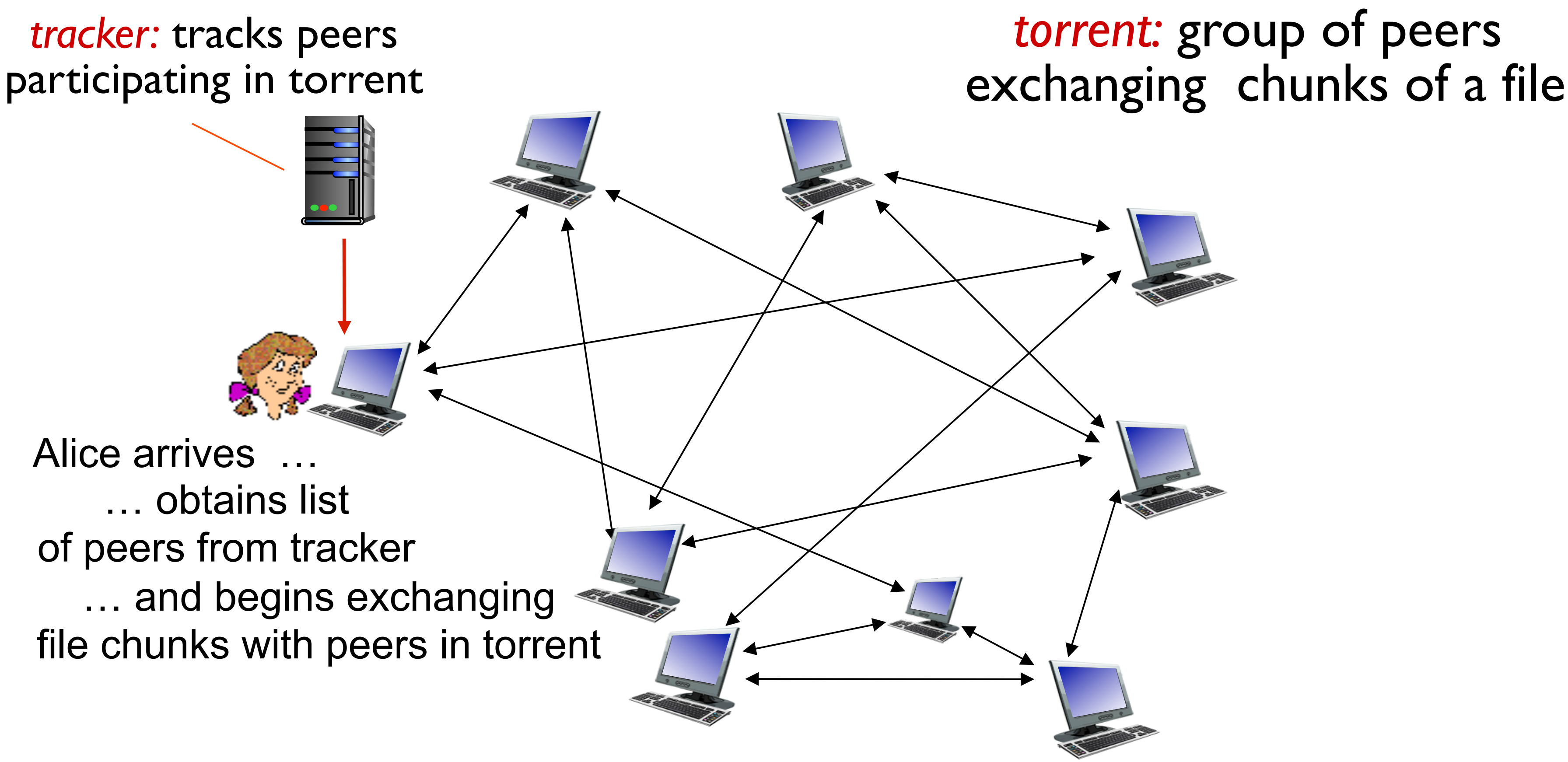
… but so does this, as each peer brings service capacity

Comparing Client-server, P2P architectures



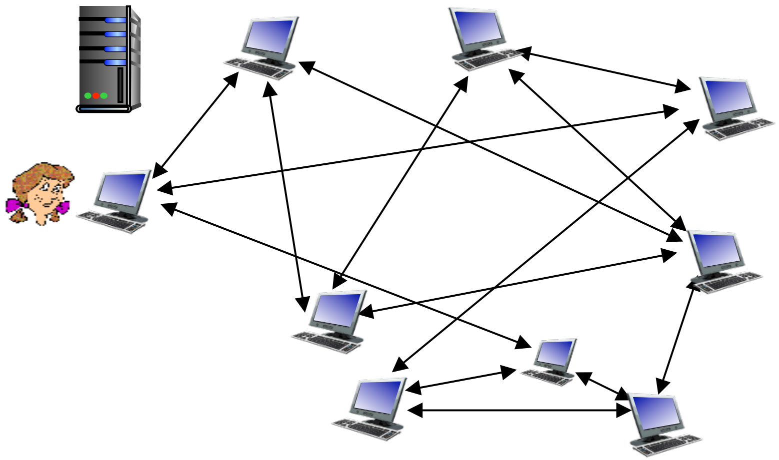
P2P file distribution: BitTorrent

- file divided into 256Kb chunks
- peers in torrent send/receive file chunks



P2P file distribution: BitTorrent

- peer joining torrent:
  - 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
- once peer has entire file, it may (selfishly) leave or (altruistically) remain in torrent

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

sending chunks: tit-for-tat

- Alice sends chunks to those four peers currently sending her chunks *at highest rate*
  - 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
  - newly chosen peer may join top 4

Food for thought…

- While P2P applications scale well to begin with, there are some concerns about the effect of mass-usage!
- On what assumptions is a packet-switching network based?
- On what assumptions is BitTorrent based?