# CSCI 466: Networks

FTP, Peer 2 Peer Networks (P2P), Content Distribution Networks (CDN)

Reese Pearsall

Fall 2024



*All images are stolen from the internet

# Announcements

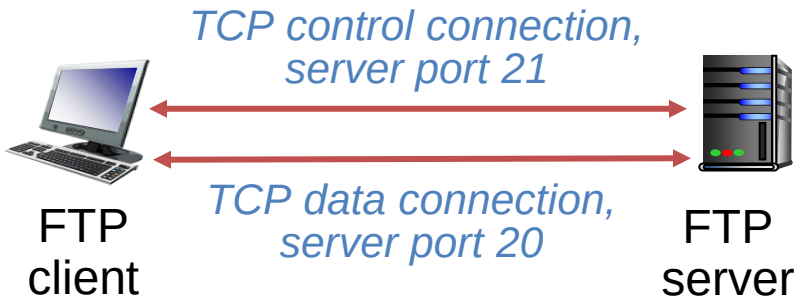
Wireshark Lab 1 due **tonight** at 11:59 PM

- Everyone needs to submit something to D2L, even if you work with partners

i hope you have a good weekend

# FTP

**File Transfer Protocol (FTP)-** protocol used for transferring files from server to client




- FTP communicates over two connections
  - Port 21 for control information
  - Port 20 for data
- Differences from HTTP
  - Control communication “out-of-band”
  - Server maintains per client state: authentication, current directory

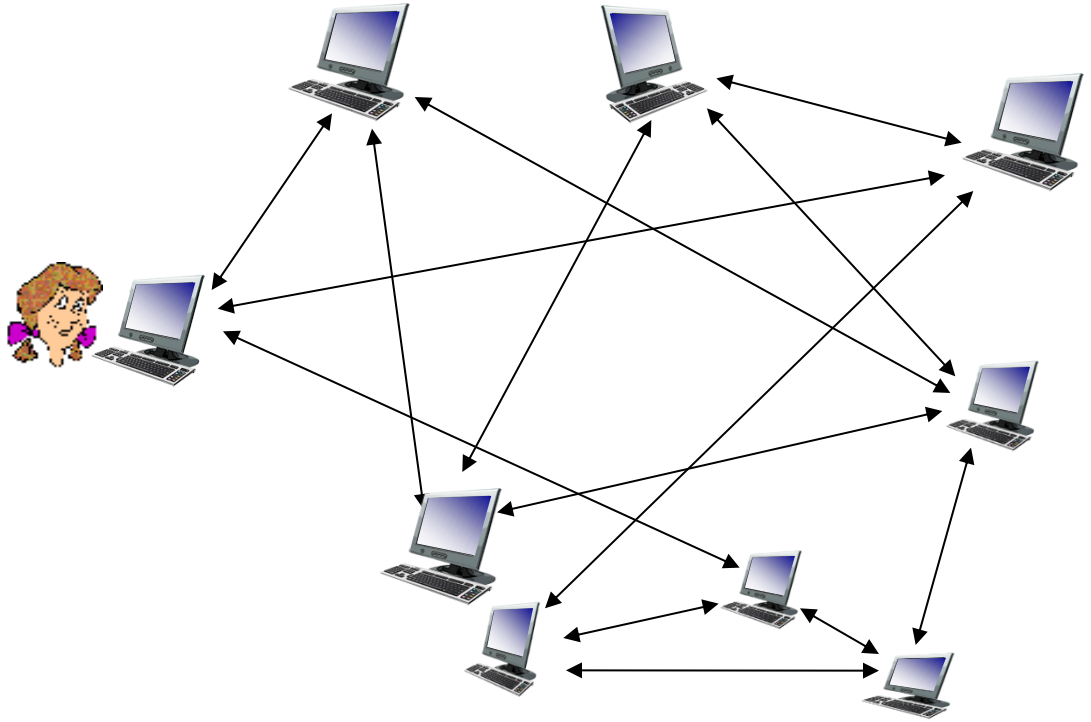
- **FTP procedure:**
  1. FTP client contacts FTP server at port 21, using TCP
  2. Client authorized over control connection
  3. Client browses remote directory, sends commands over control connection
  4. When server receives file transfer command, server opens 2nd TCP data connection (for file) to client
  5. After transferring one file, server closes data connection

Why use a separate control connection?

# WinSCP

# P2P Networks

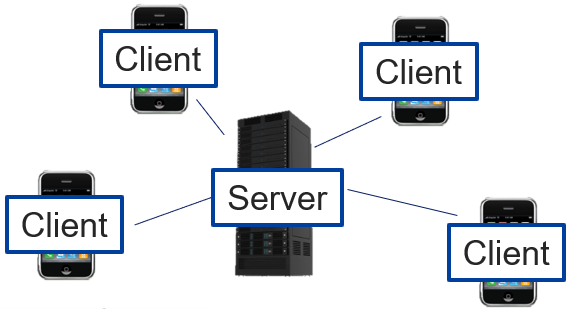
- No always-on server
- Arbitrary end systems directly communicate
- Peers are intermittently connected and change IP addresses



Time to distribute a file of size F to N clients

$$D_{P2P} = \max\left\{\frac{F}{u_s}, \frac{F}{d_{\min}}, \frac{NF}{u_s + \sum_{i=1}^{N} u_i}\right\}$$

## Client-Server Architecture



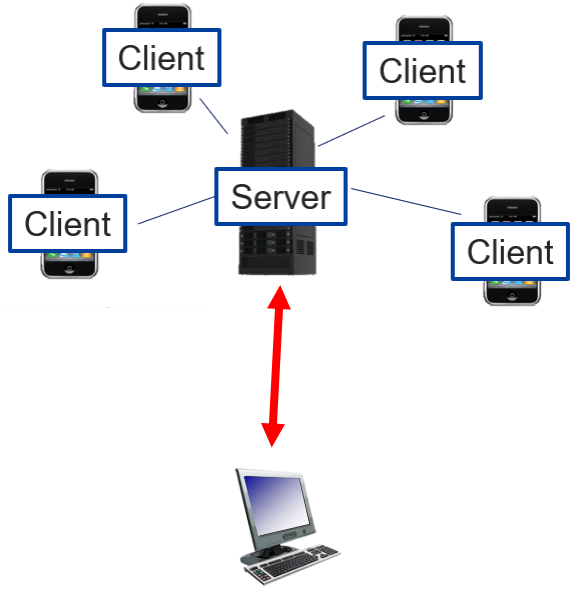

- Server can upload data at rate $u_s$
- Clients download data at rates $d_1, d_2, \ldots, d_N$

$$D_{CS} = \max\left\{\frac{NF}{u_s}, \frac{F}{d_{\min}}\right\}$$

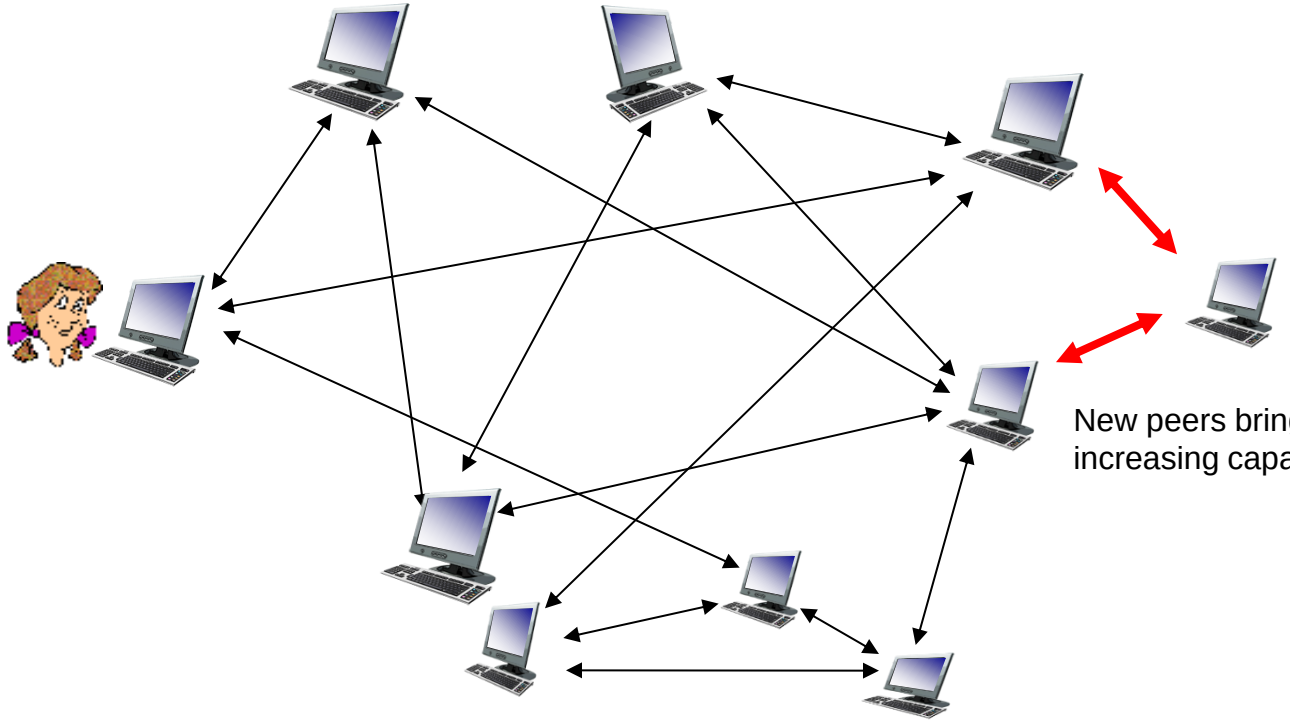
Time to distribute a file of size F to N clients

# P2P Networks

**Client-Server Architecture**



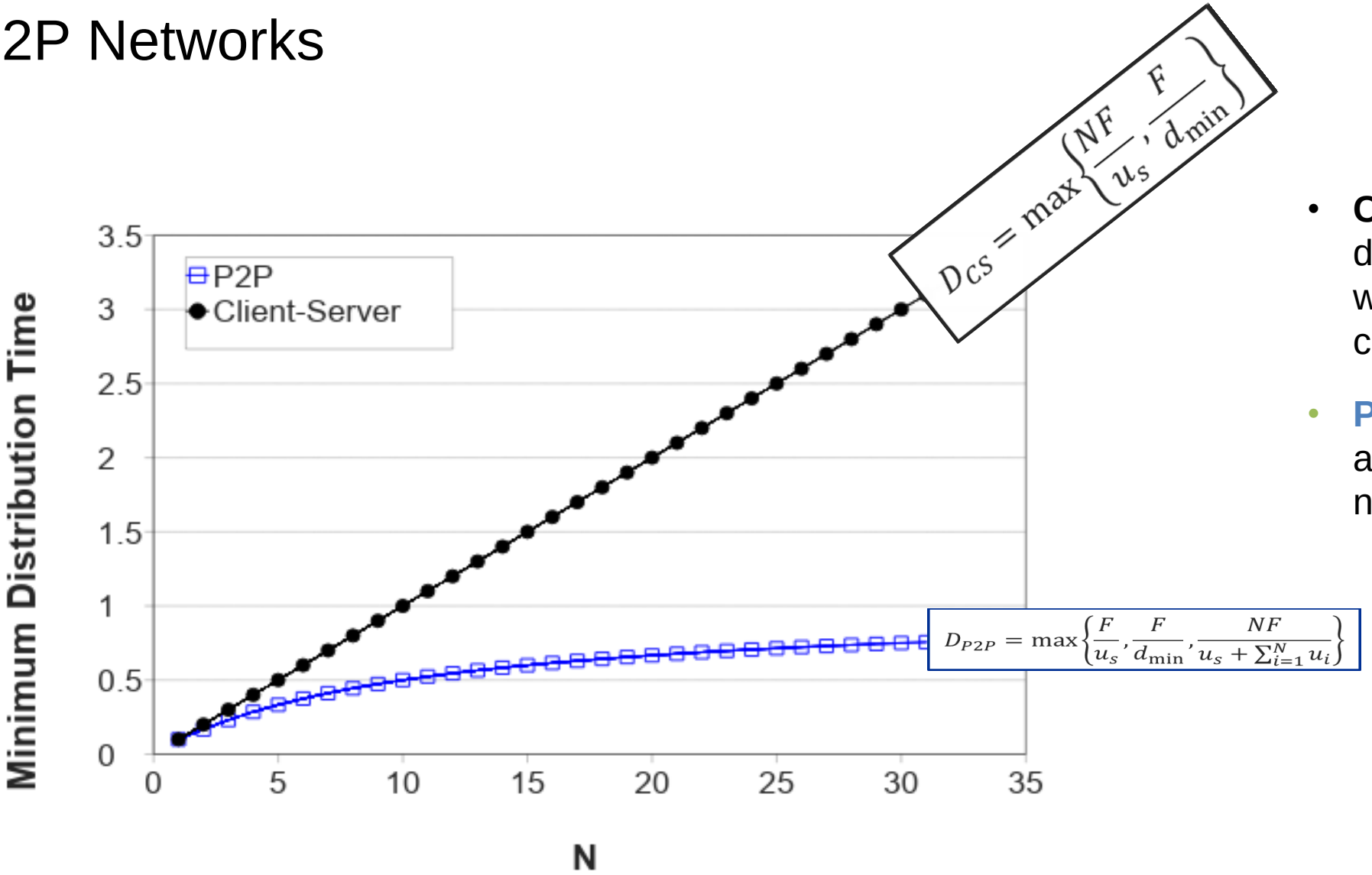

Existing clients have to share resources with new users




New peers are both a client and a server. There will not be a negative impact on current peers

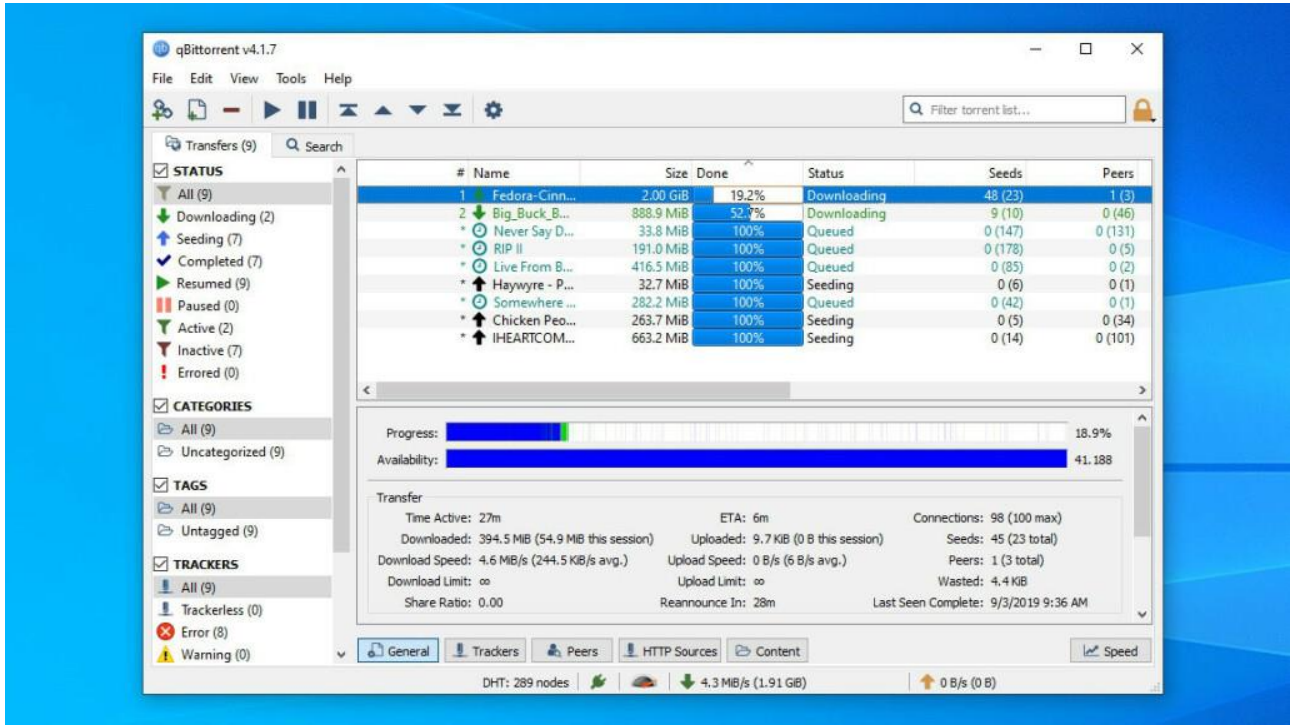
P2P architectures are *self-scaling*

# P2P Networks




- **Client server** distribution time grows with the number of clients
- **P2P** distribution time approaches 1 hour as number of clients grows

# P2P file distribution: BitTorrent




Service for sharing files over the internet in a decentralized fashion

(BitTorrent, uTorrent, qBittorrent are all popular torrent clients)

User 1

User 1

10.1.2.3

A **Tracker** is made, typically located at the person who created the torrent

Tracker

Chunk1: 10.1.2.3
Chunk2: 10.1.2.3
Chunk3: 10.1.2.3
Chunk4: 10.1.2.3
…
ChunkN: 10.1.2.3

User 1

10.1.2.3

User 2

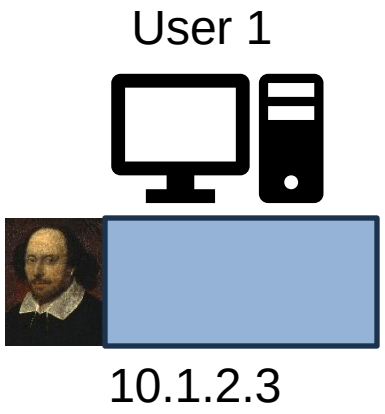
10.1.2.4

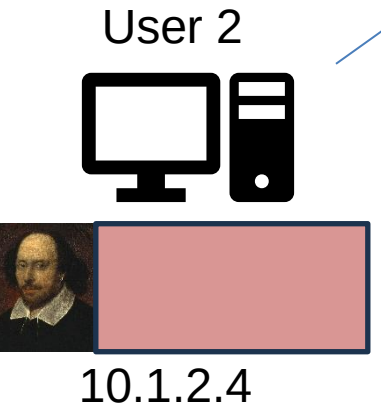
User 2 wants the file, so they download a **TorrentFile** which allows them to contact the Tracker and join the P2P

A **Tracker** is made, typically located at the person who created the torrent

Tracker

Chunk1: 10.1.2.3
Chunk2: 10.1.2.3
Chunk3: 10.1.2.3
Chunk4: 10.1.2.3
…
ChunkN: 10.1.2.3

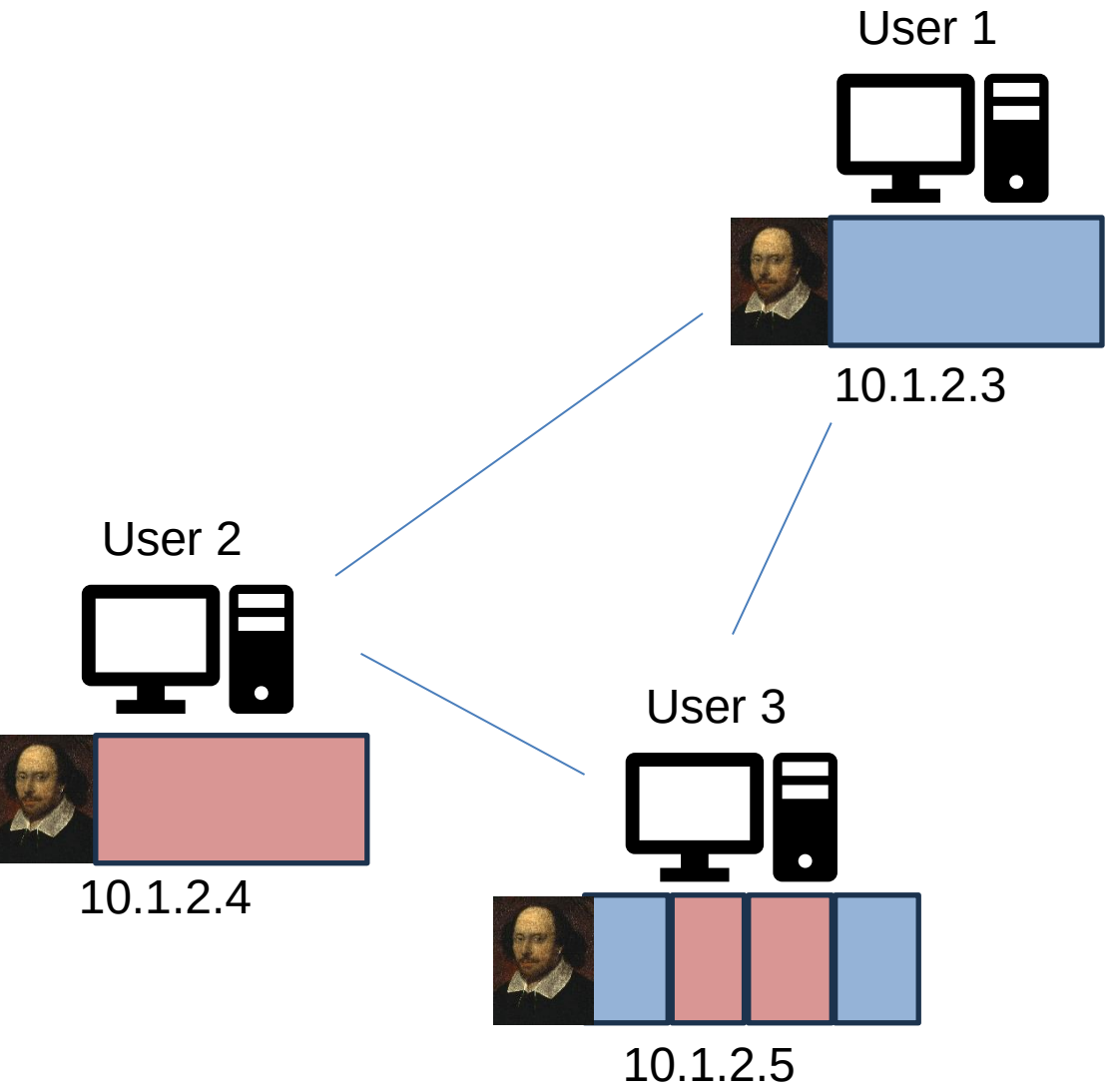



A **Tracker** is made, typically located at the person who created the torrent

## Tracker

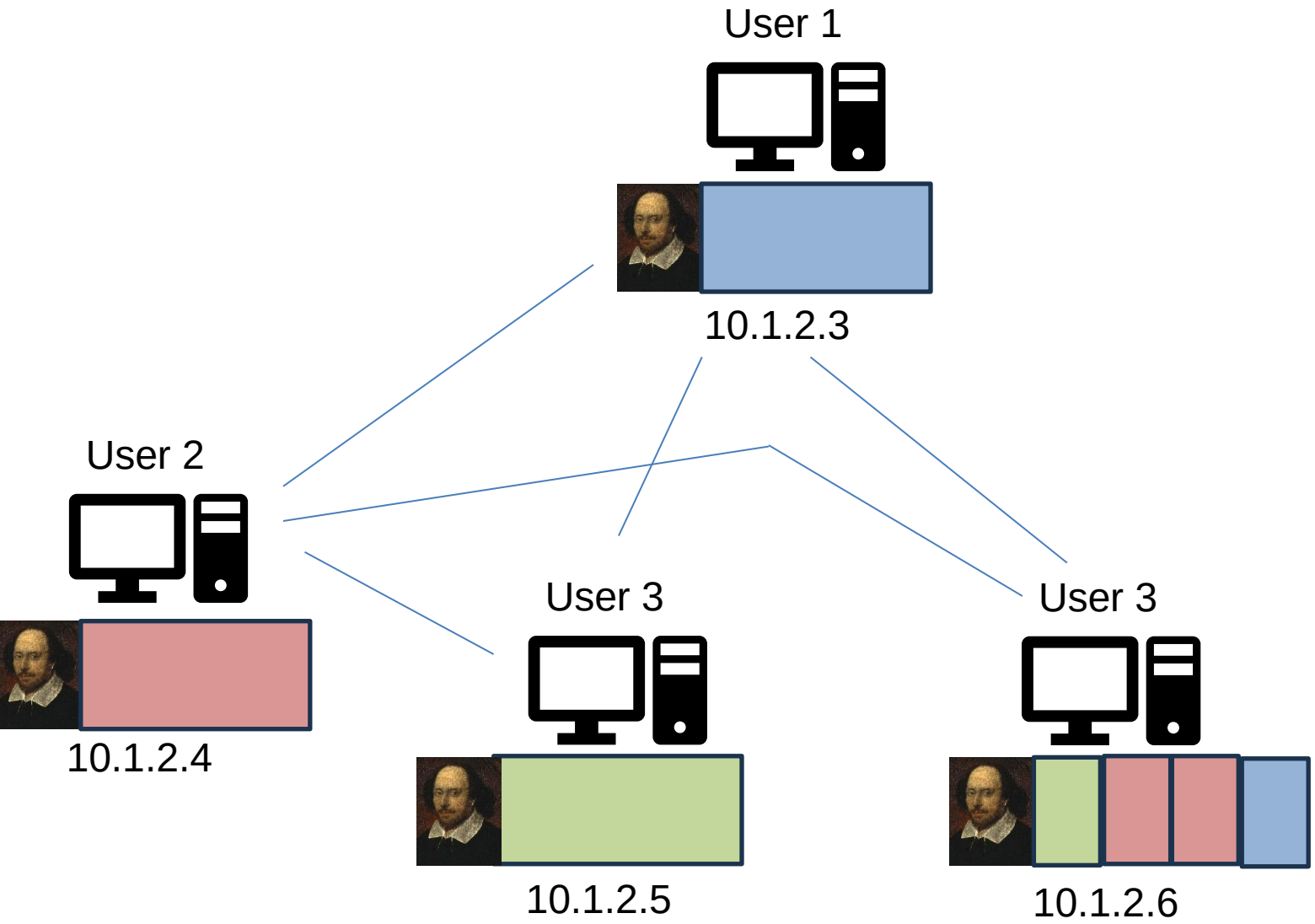
Chunk1: 10.1.2.3
Chunk2: 10.1.2.4
Chunk3: 10.1.2.4
Chunk4: 10.1.2.3
…
ChunkN: 10.1.2.3




A **Tracker** is made, typically located at the person who created the torrent

## Tracker

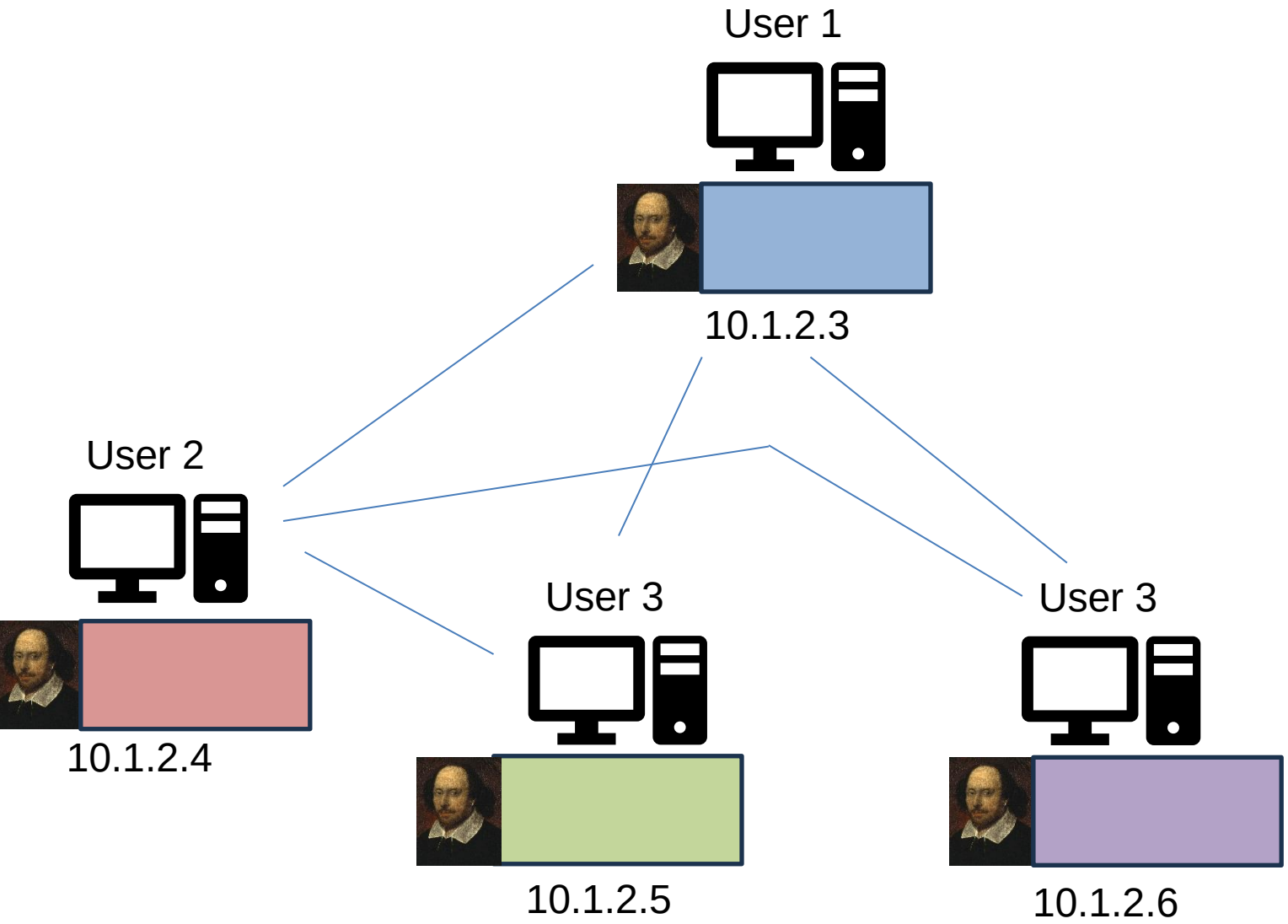
Chunk1: 10.1.2.3
Chunk2: 10.1.2.4
Chunk3: 10.1.2.4
Chunk4: 10.1.2.3
…
ChunkN: 10.1.2.3




A **Tracker** is made, typically located at the person who created the torrent

## Tracker

Chunk1: 10.1.2.5
Chunk2: 10.1.2.4
Chunk3: 10.1.2.4
Chunk4: 10.1.2.3
…
ChunkN: 10.1.2.3




A **Tracker** is made, typically located at the person who created the torrent

## Tracker

Chunk1: 10.1.2.5
Chunk2: 10.1.2.4
Chunk3: 10.1.2.4
Chunk4: 10.1.2.3
…
ChunkN: 10.1.2.3

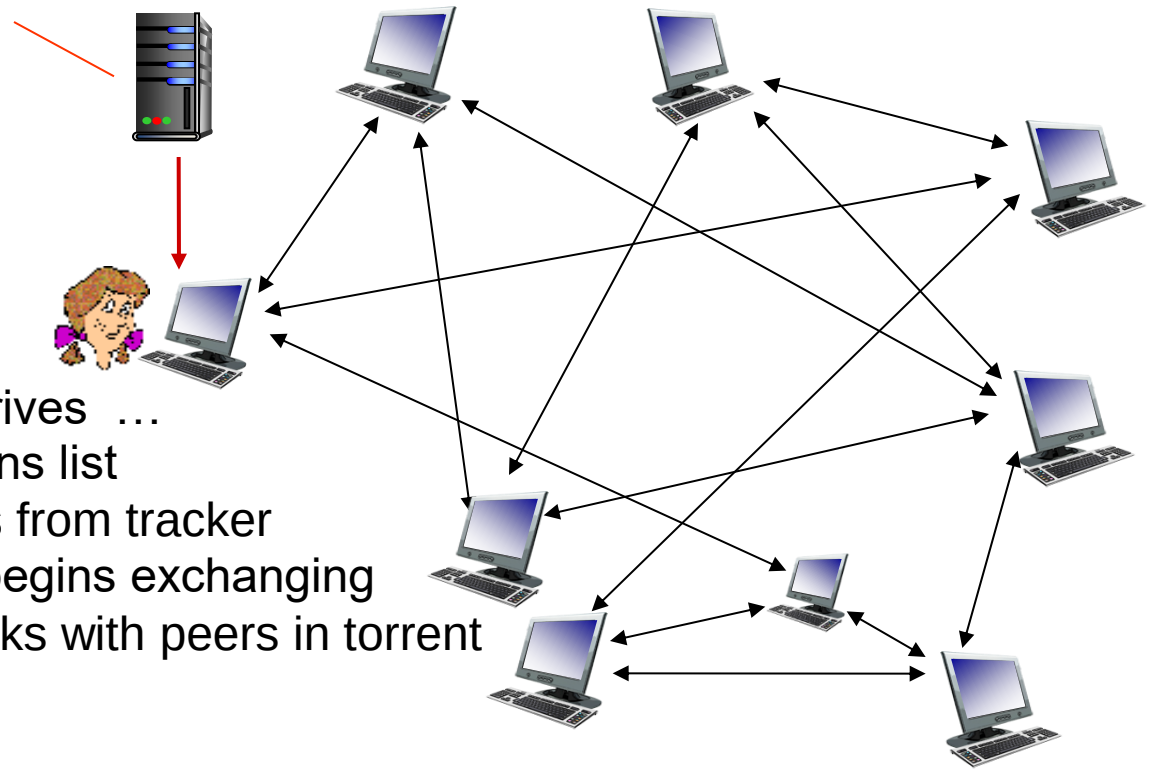
As more users join the network, it grows the capability of the networks and chunks are more scattered

# P2P file distribution: BitTorrent

- Files are divided into chunks
- Peers in torrent send/receive file chunks

*tracker:* tracks peers participating in torrent

*torrent:* group of peers exchanging chunks of a file

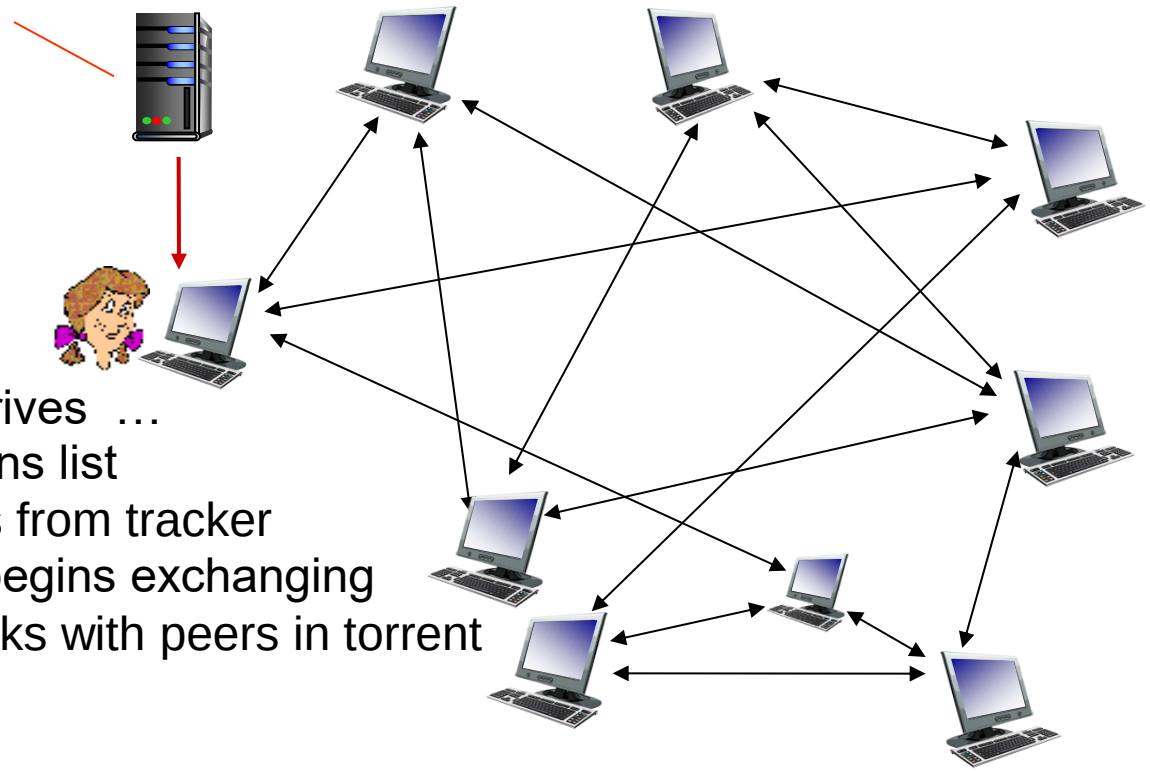


Alice arrives …
… obtains list
of peers from tracker
… and begins exchanging
file chunks with peers in torrent

# P2P file distribution: BitTorrent

- Files are divided into chunks
- Peers in torrent send/receive file chunks

*tracker:* tracks peers participating in torrent



Alice arrives …
… obtains list
of peers from tracker
… and begins exchanging
file chunks with peers in torrent

*torrent:* group of peers exchanging chunks of a file

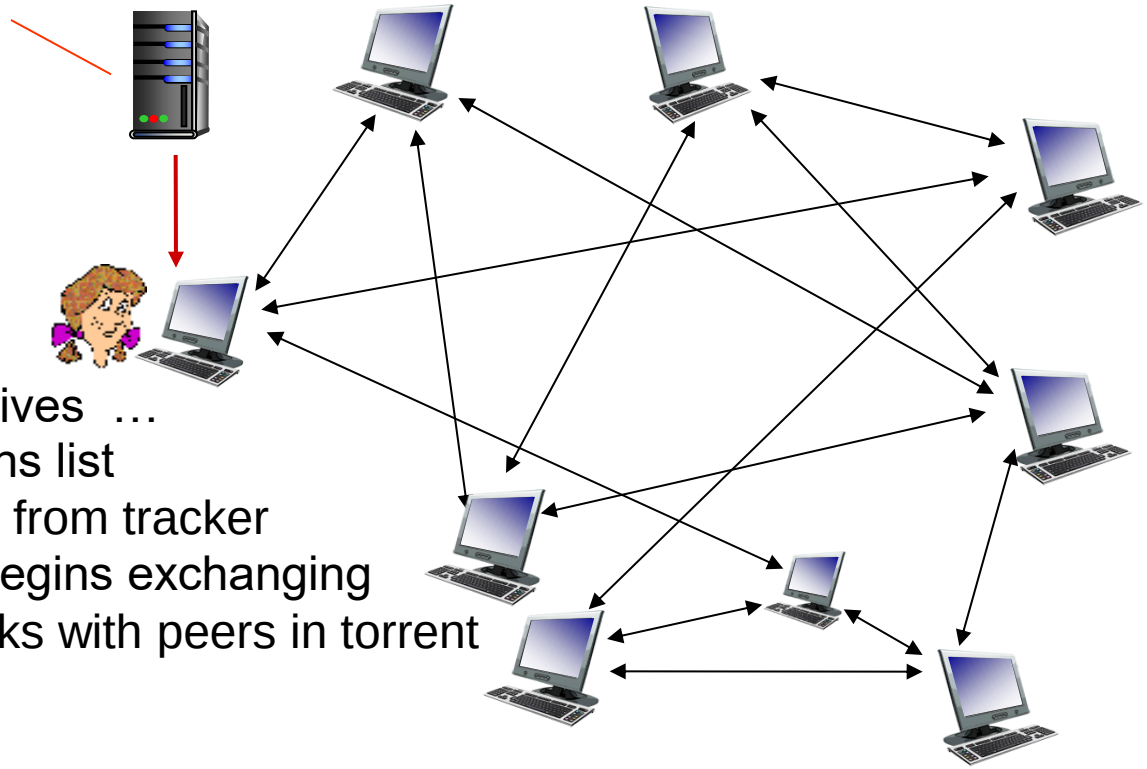
**peer joining torrent:**

- 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")

# P2P file distribution: BitTorrent

- Files are divided into chunks
- Peers in torrent send/receive file chunks

*tracker:* tracks peers participating in torrent

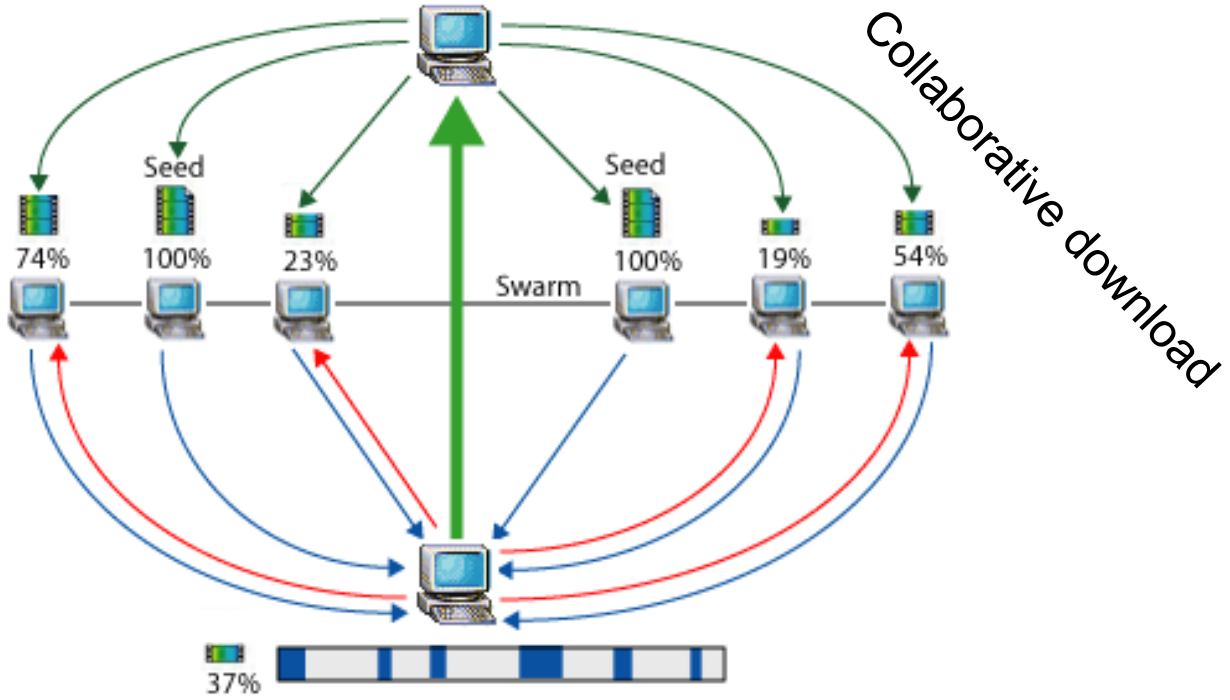


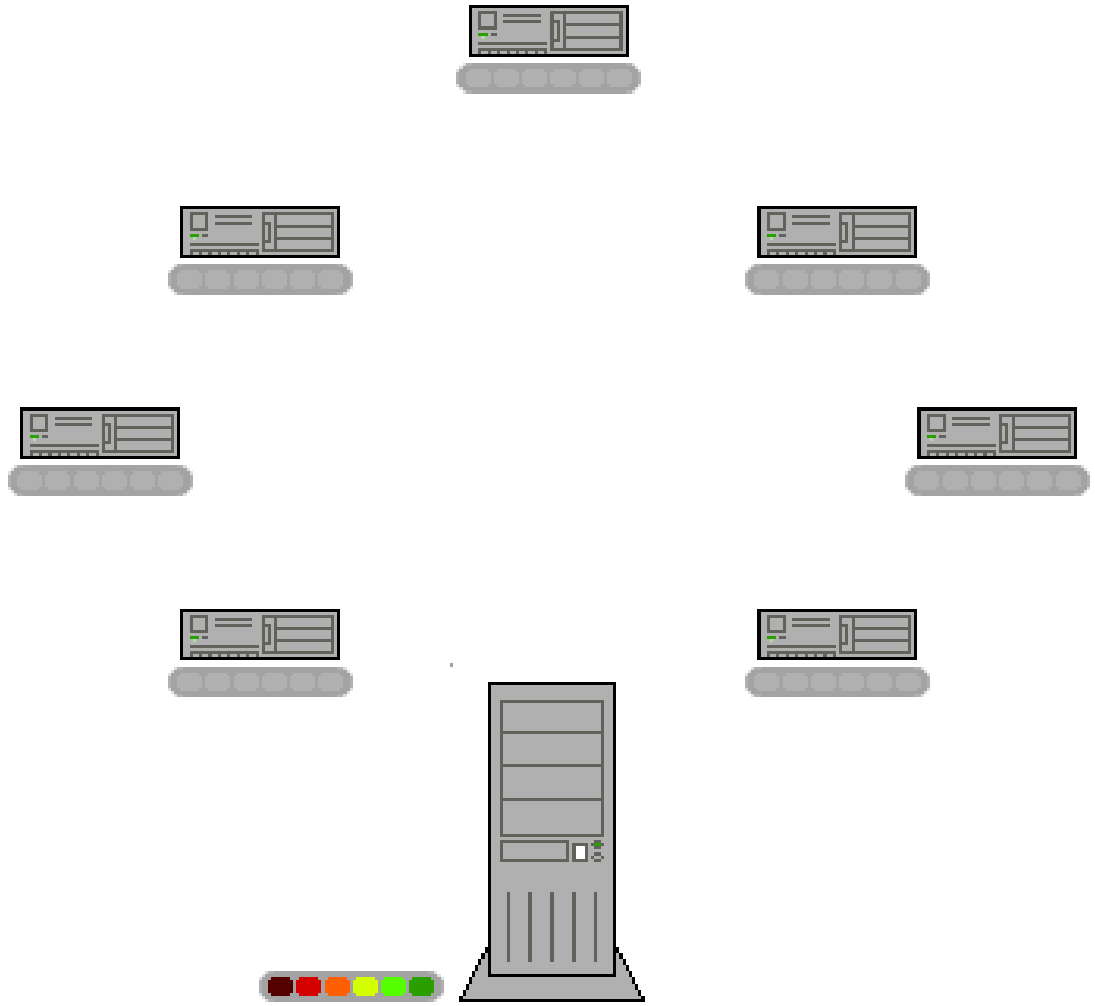
Alice arrives …
… obtains list of peers from tracker
… and begins exchanging file chunks with peers in torrent

*torrent:* group of peers exchanging chunks of a file

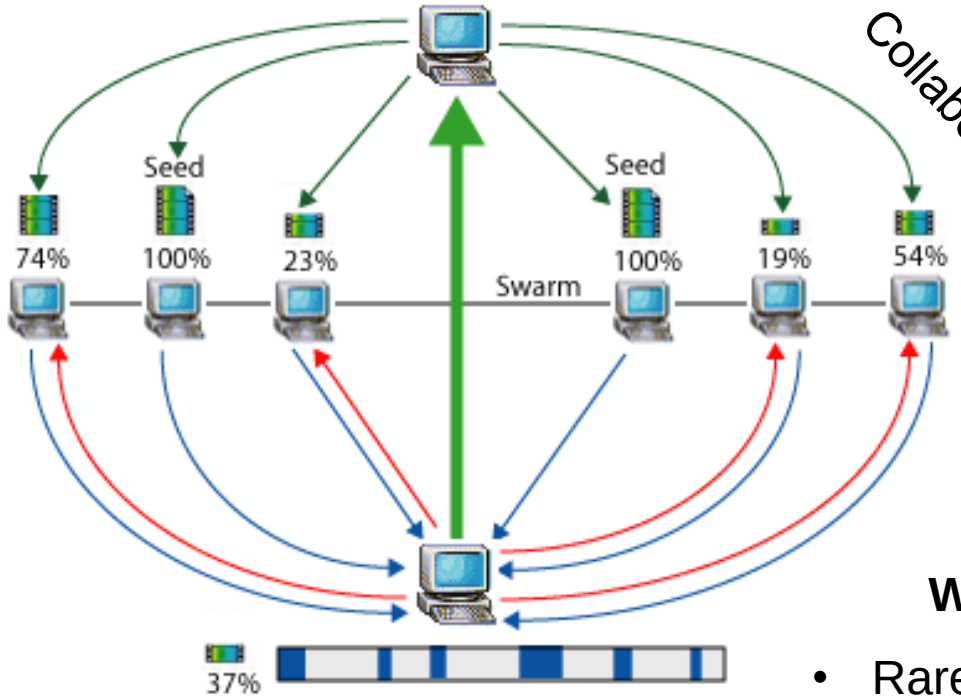
- while downloading, peer uploads chunks to other peers
- peer may change peers with whom it exchanges chunks
- *churn:* peers may come and go
- once peer has entire file, it may (selfishly) leave or (altruistically) remain in torrent

# P2P file distribution: BitTorrent

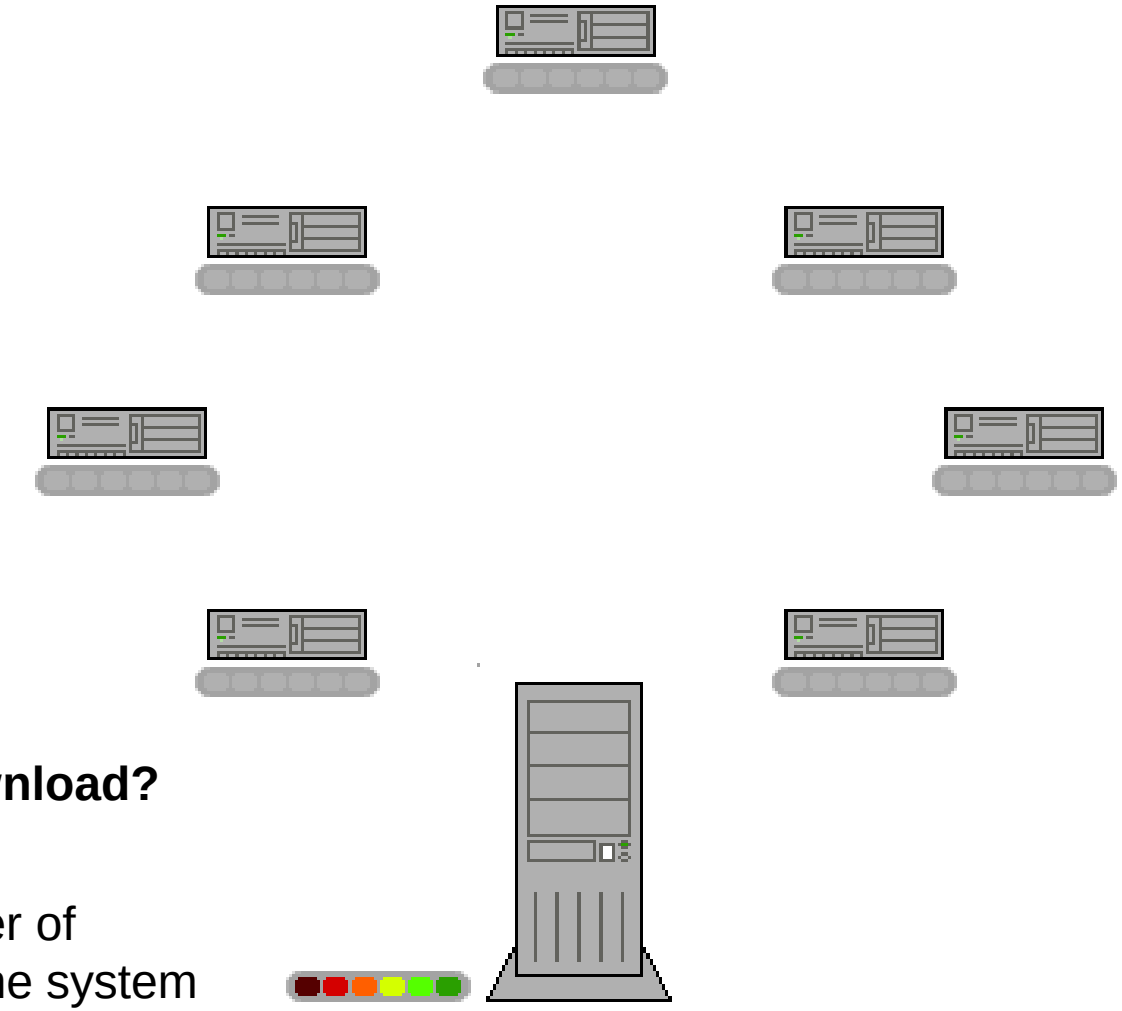


# P2P file distribution: BitTorrent



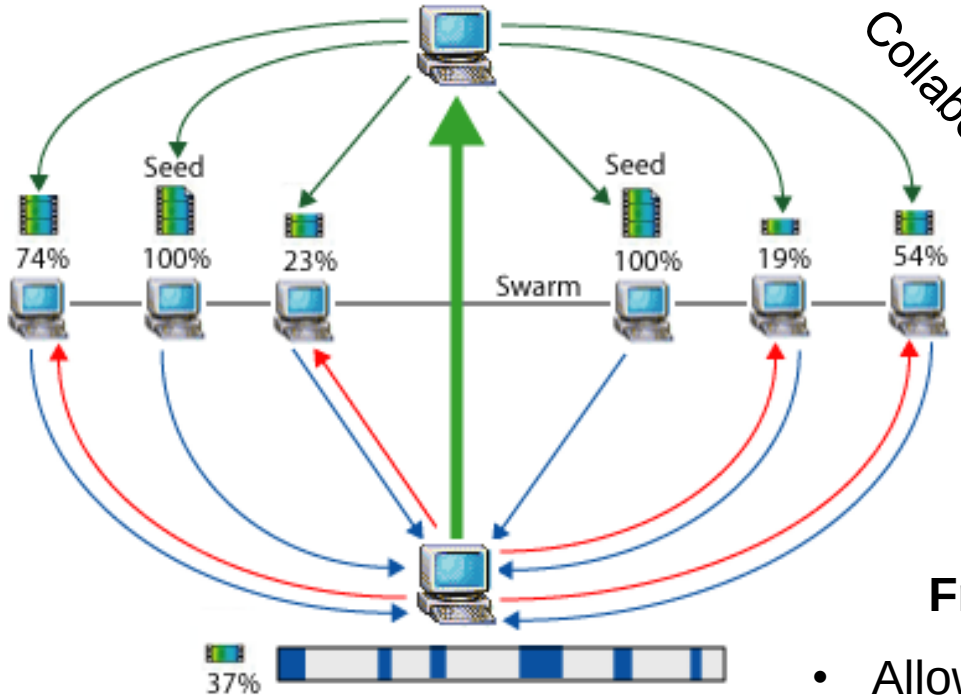

Collaborative download



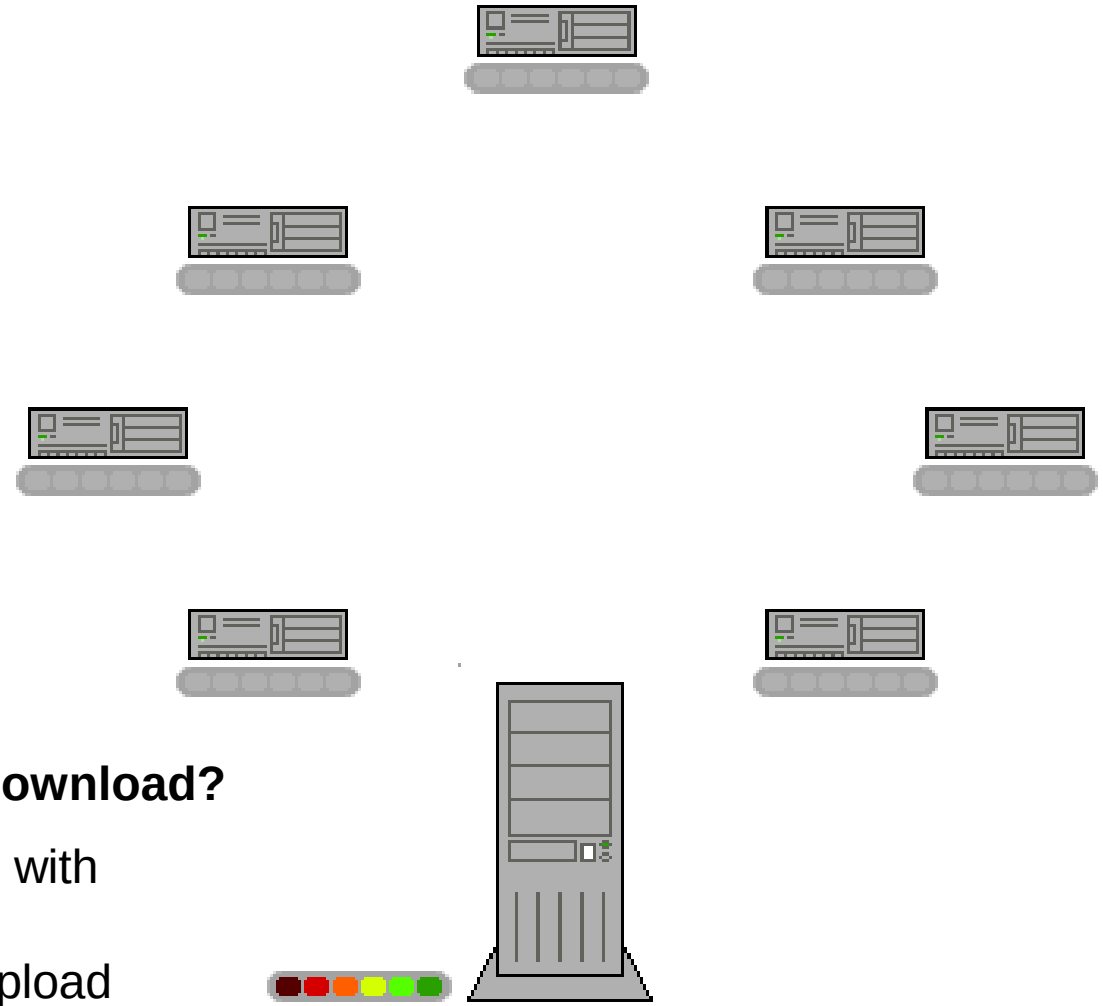
**Which chunks to download?**

- Rarest first
- Try to equalize the number of copies of each chunk in the system

# P2P file distribution: BitTorrent




Collaborative download



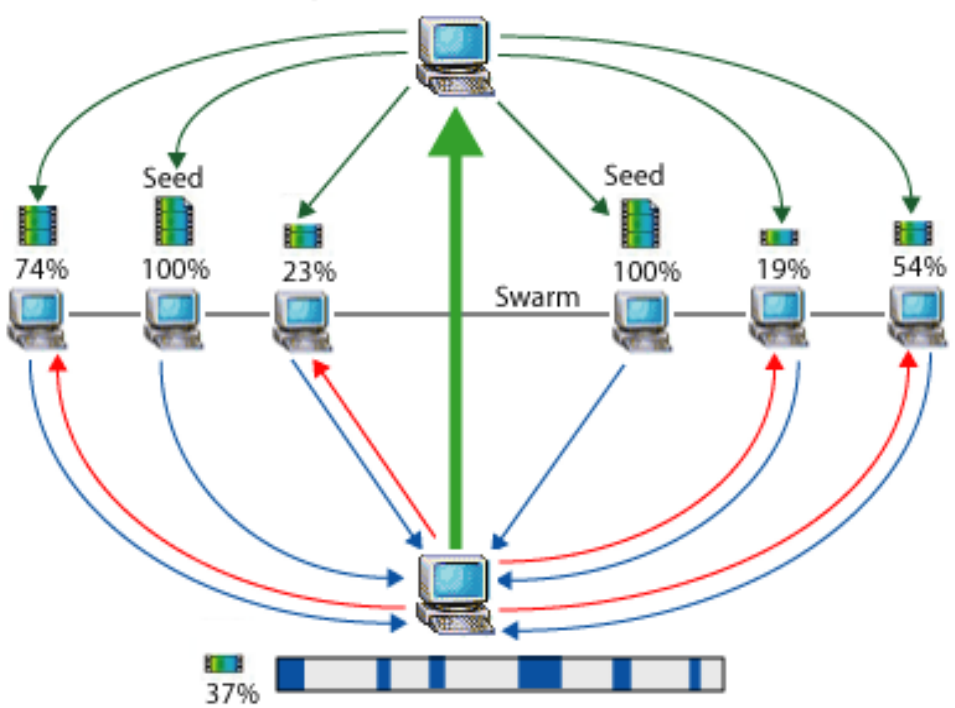
**From which peers to download?**

- Allow download to N peers with highest upload rates
- Allows peers with similar upload rates to find each other

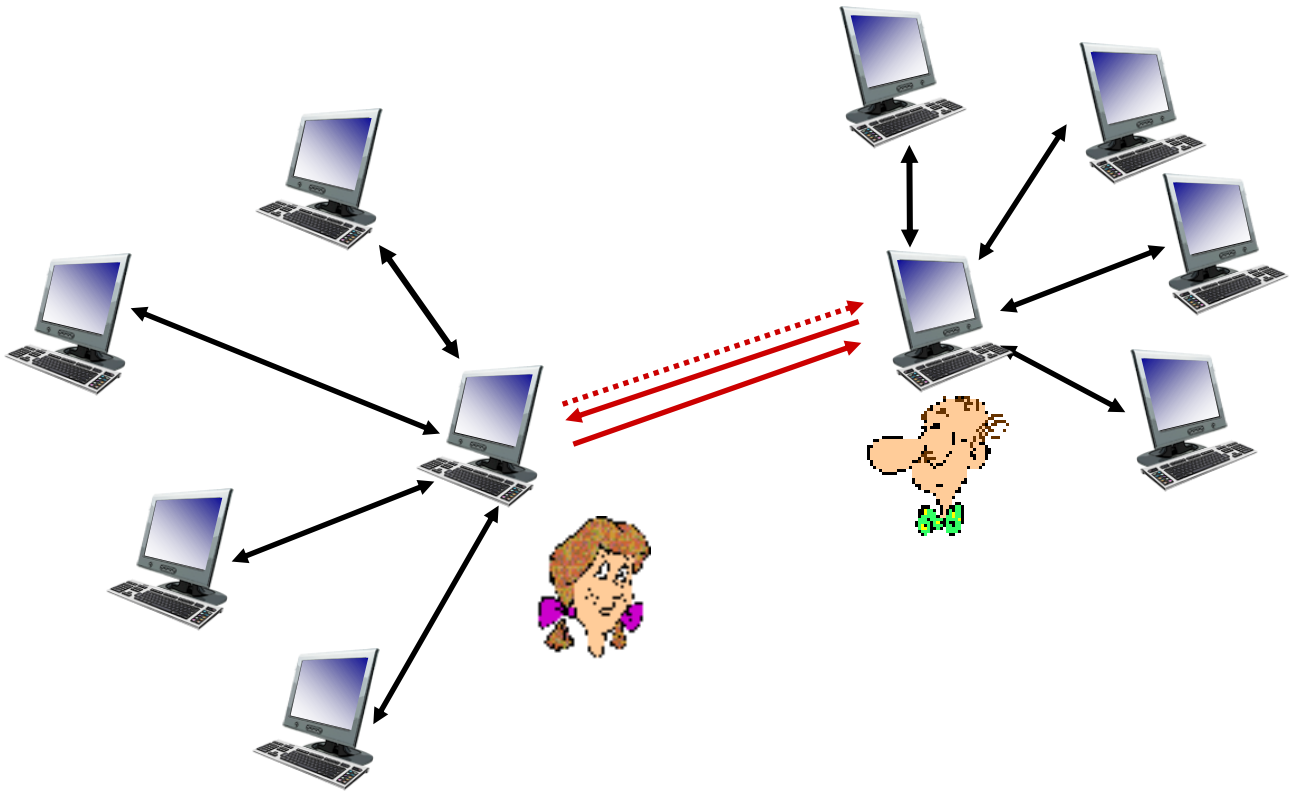
# P2P file distribution: BitTorrent

*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



# P2P file distribution: BitTorrent



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

## *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

BitTorrent is referred to as an **unstructured** P2P

Security Concerns?
- IP Address
- Malware

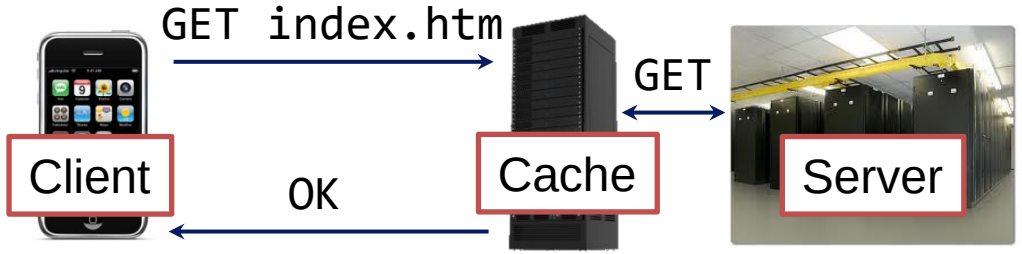
# CDNs

- video traffic: major consumer of Internet bandwidth
  - Netflix, YouTube: 37%, 16% of downstream residential ISP traffic
  - ~1B YouTube users, ~75M Netflix users
- challenge:  scale - how to reach ~1B users?
  - single mega-video server won't work (why?)
- challenge: heterogeneity
  - different users have different capabilities (e.g., wired versus mobile; bandwidth rich versus bandwidth poor)
- *solution:* distributed, application-level infrastructure




# CDNs

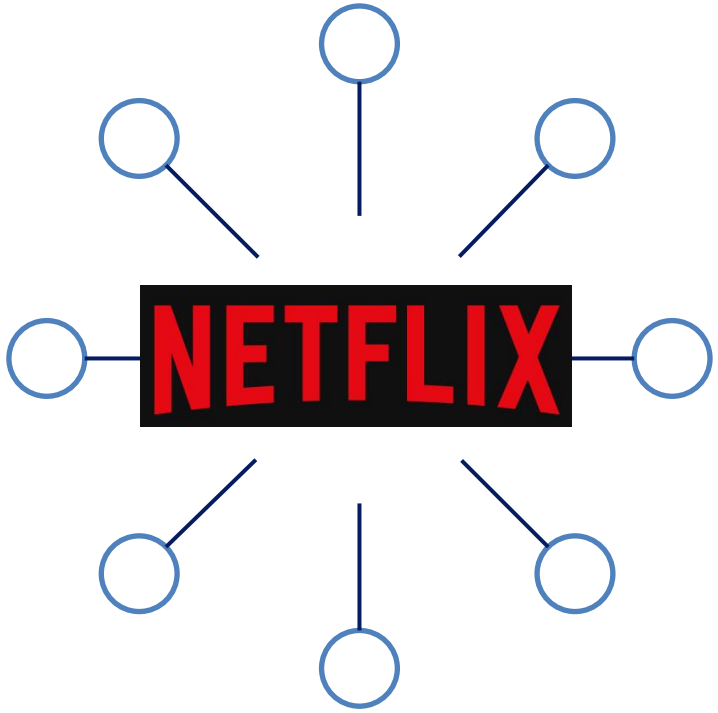
- Caching
  - Save previously delivered data
  - Subsequent requests served from cache on the browser, or in the access network
- Applications
  - Reduce response time for client request
  - Reduce ISP traffic costs
- Content distribution networks
  - Distributed caches
  - Web objects addressed to CDN server
  - CDN server fetches from content provider on first access




- Conditional GET
  - Cache: specify date of cached copy in HTTP request
    `If-modified-since: <date>`
  - Server: response contains no object if cached copy is up-to-date:
    `HTTP/1.0 304 Not Modified`

# CDNs

- Challenge:
  - How to stream content to millions of users?
- Option 1:
  - Single mega-datacenter
  - Pros: Simple
  - Cons:
    - Single point of failure
    - Point of network congestion
    - Long path to distant clients
    - Multiple copies of video sent over outgoing link



# CDNs

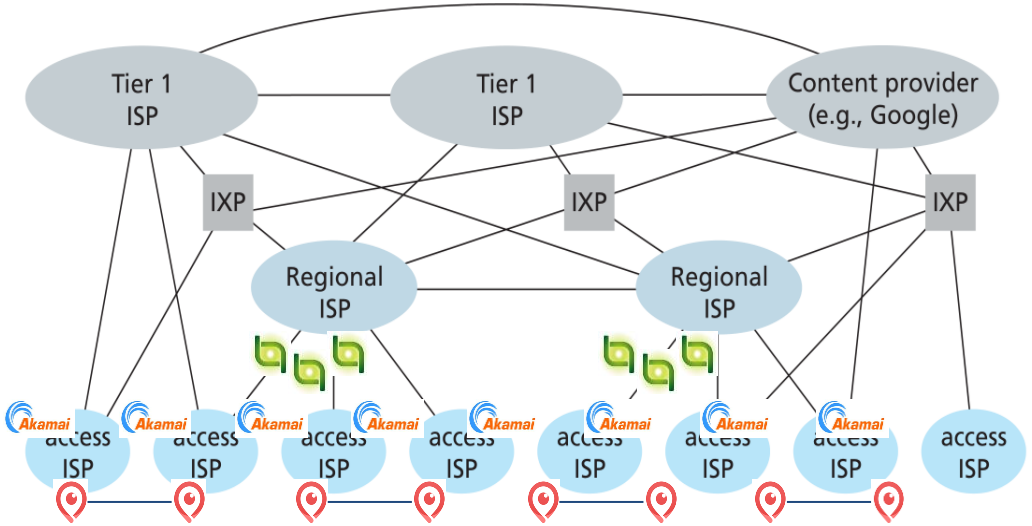
- Option 2:
  - Store/serve multiple copies of videos at multiple geographically distributed sites

A **content delivery network (CDN)** refers to a geographically distributed group of cache servers which work together to provide fast delivery of Internet content. **(not a web host)**


34 DNS lookups

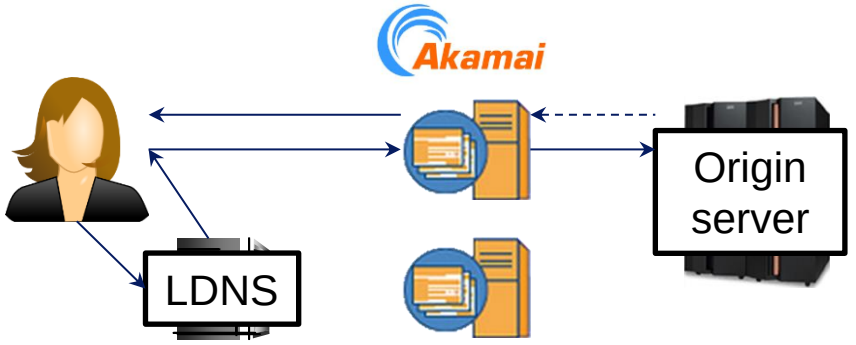
204 HTTP requests

520 KB of data downloaded




amazon web services™ Akamai Level(3) COMMUNICATIONS Google™

56% of domains resolve to a CDN

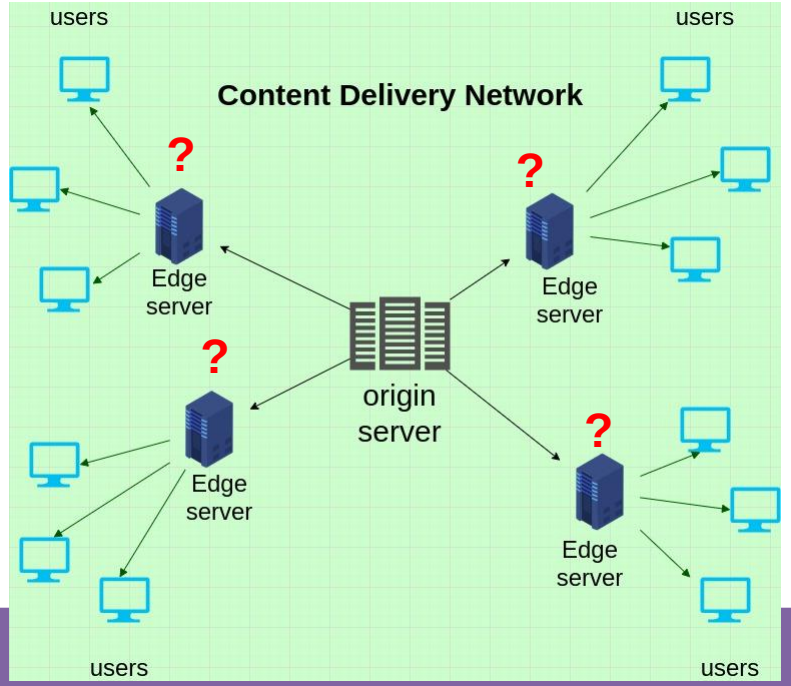


# CDNs

https://**cdn**.discordapp.com/attachments/101763035862114311 0/1017849417216294972/unknown.png

# CDNs

- Challenge: how does CDN DNS select “good” CDN node to stream to client
  - Pick CDN node **geographically** closest to client’s local DNS
  - Pick CDN node with **shortest delay** (or min # hops) to client (CDN nodes periodically ping access ISPs, reporting results to CDN DNS)



# CDNs

In HTTP steaming, video is stored at an HTTP server and retrieved with a GET request

Frames are sent to a client buffered and played back after a certain threshold

All clients receiving the same encoding of the video despite widely different bandwidth

# CDNs

In HTTP steaming, video is stored at an HTTP server and retrieved with a GET request

Frames are sent to a client buffered and played back after a certain threshold

All clients receiving the same encoding of the video despite widely different bandwidth

**Dynamic Adaptive Streaming over HTTP (DASH)**- video is encoded into several different versions

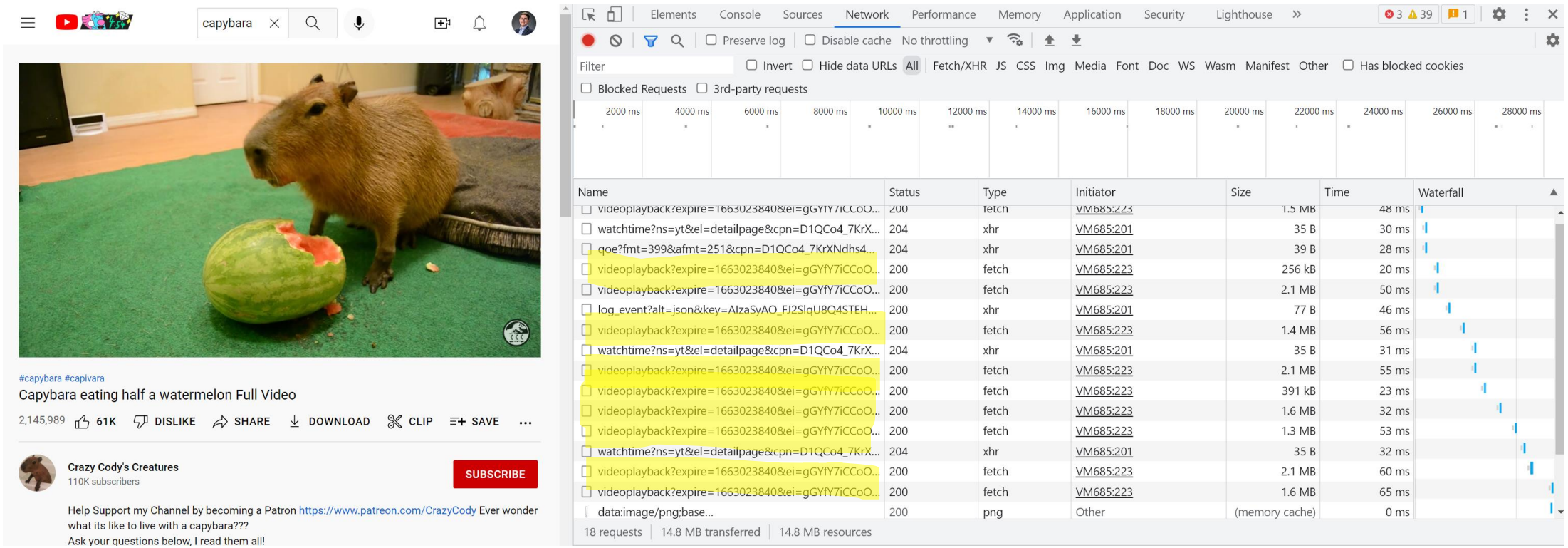
- Different encoding rate, different quality, etc

Clients dynamically request chunks of video segments every few seconds via GET requests

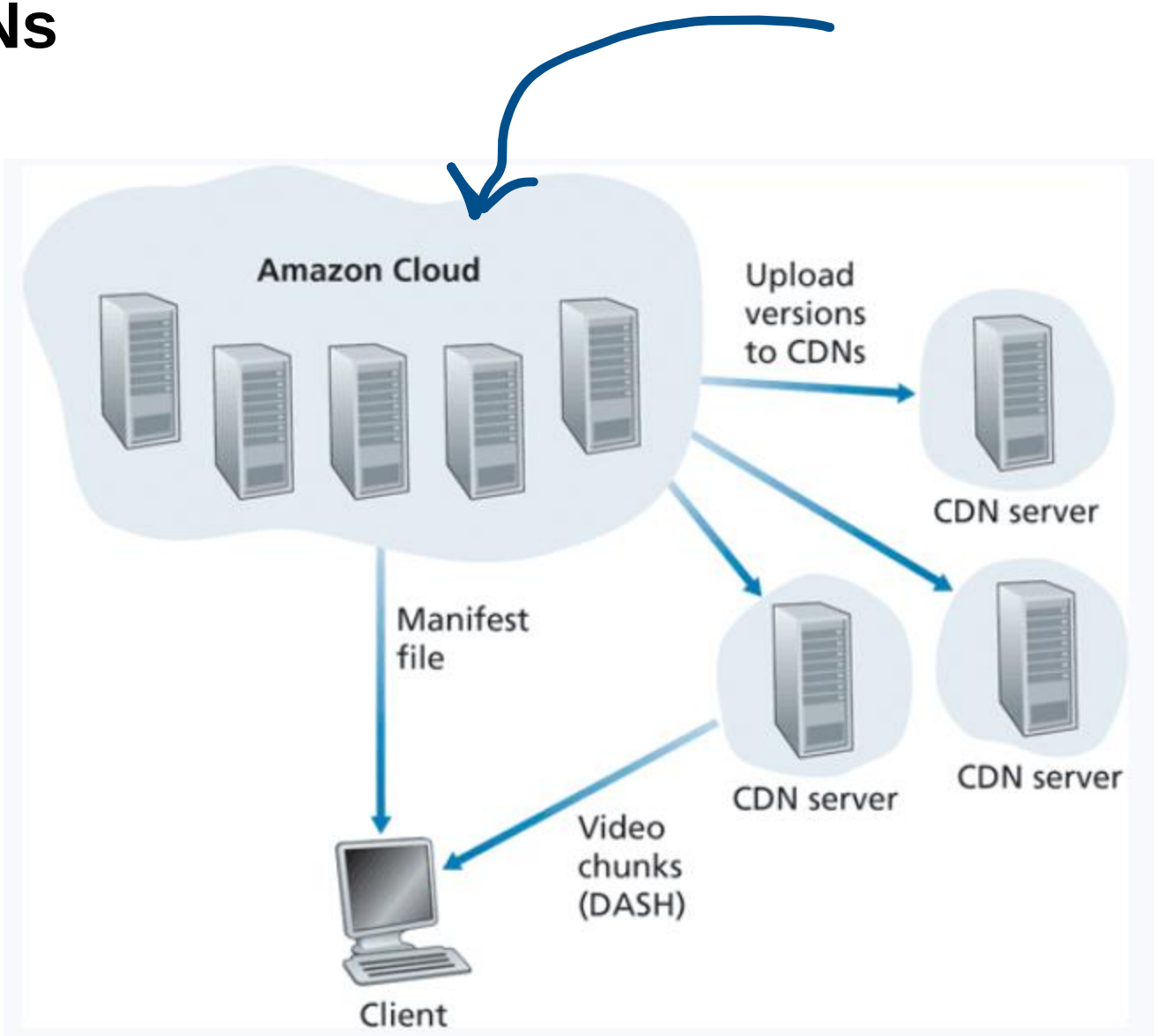
- Is the current bandwidth good? → Retrieve good quality
- Is the current bandwidth bad? → Retrieve ok quality

# CDNs

When streaming, you are consistently issuing GET requests

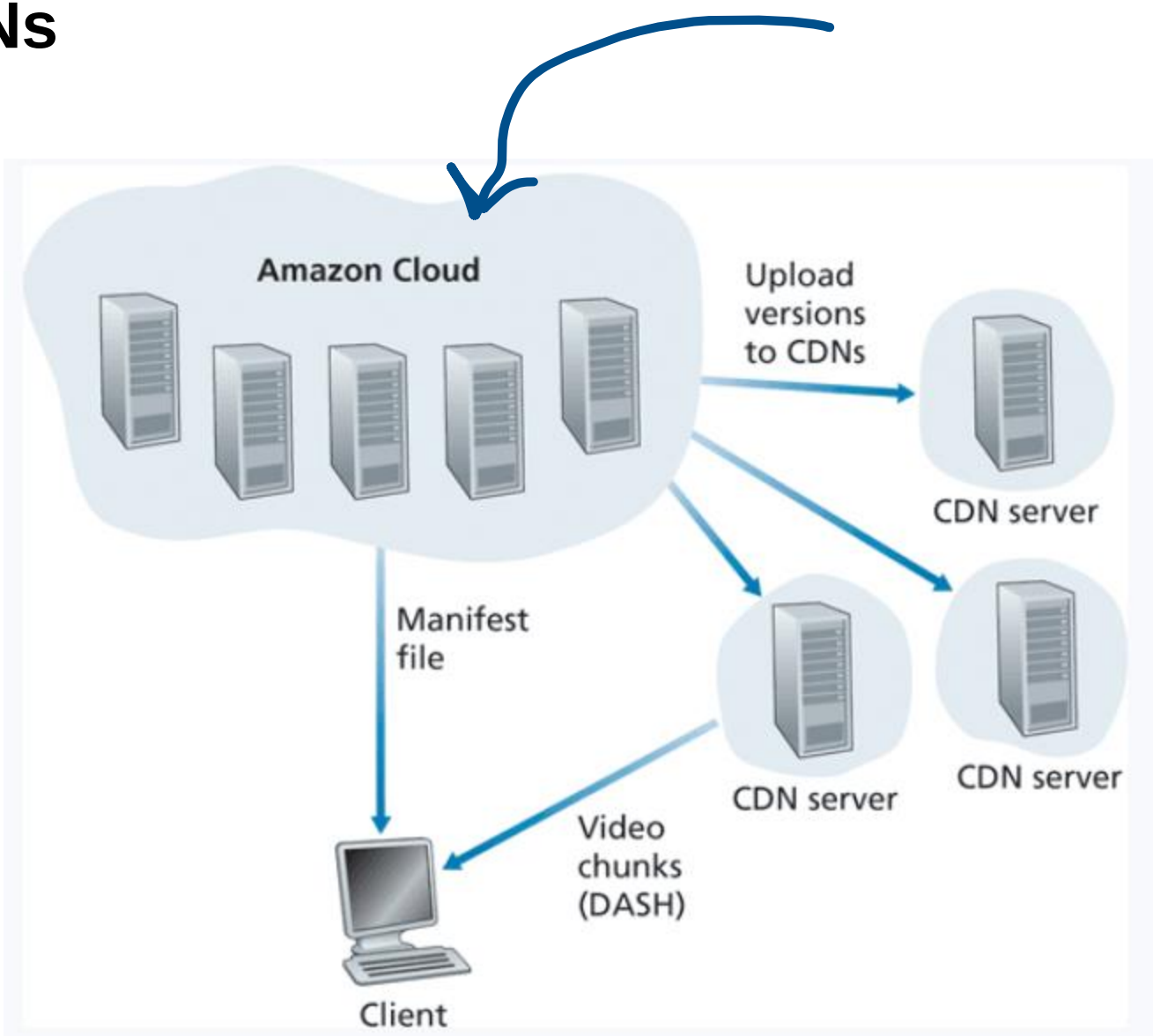


# CDNs




Studio maser versions are uploaded to a private Amazon cloud

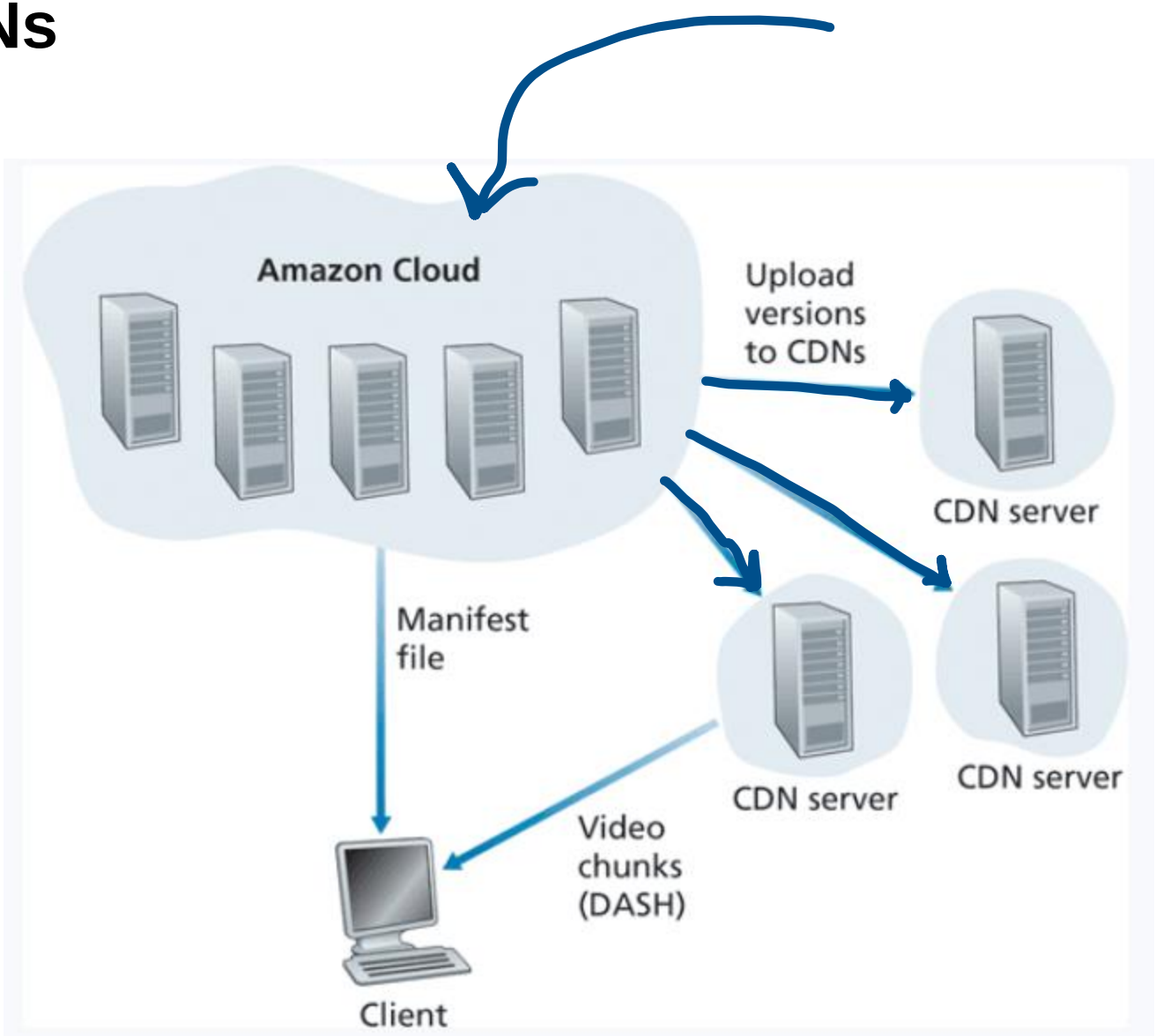
# CDNs




Studio maser versions are uploaded to a private Amazon cloud

Videos are processed into many different formats, allowing for DASH

# CDNs




Studio maser versions are uploaded to a private Amazon cloud

Videos are processed into many different formats, allowing for DASH

Versions are uploaded to Netflix's CDNs