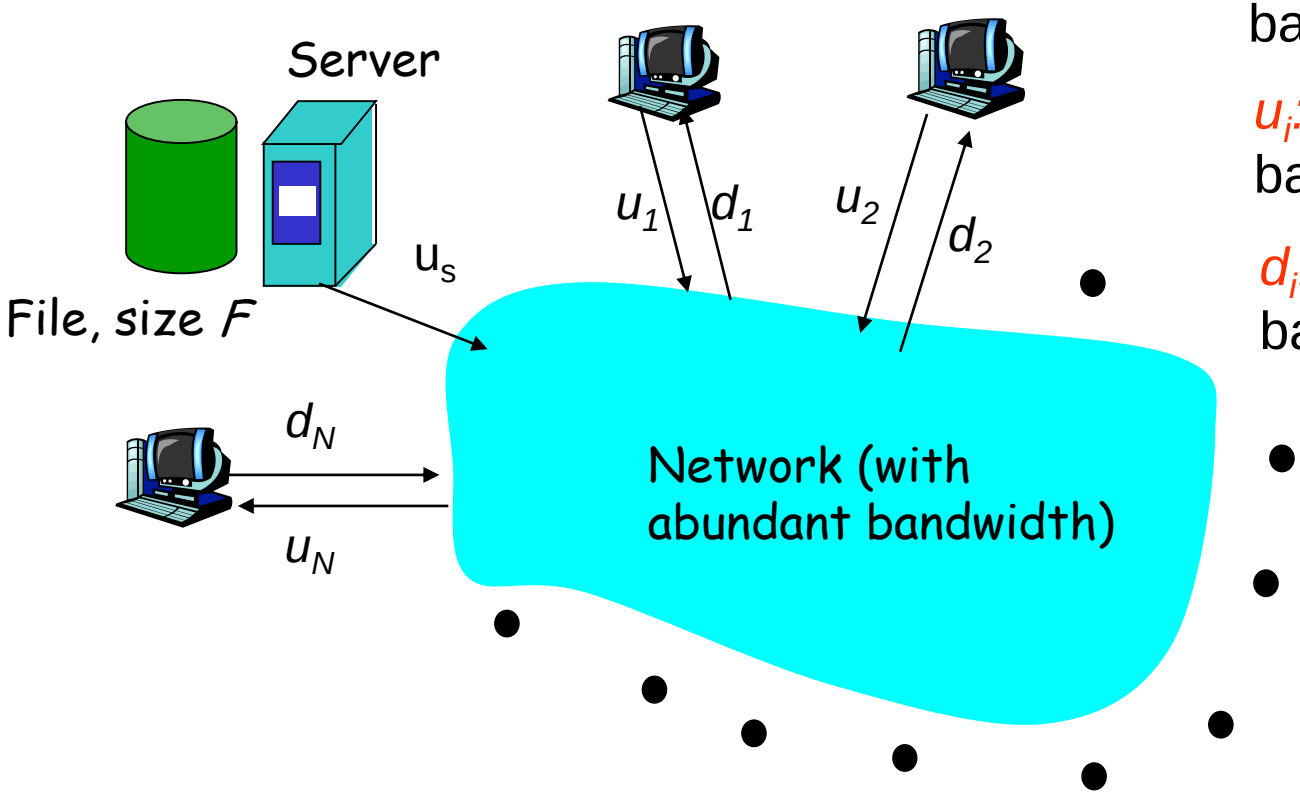
File Distribution: Server-Client vs P2P

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



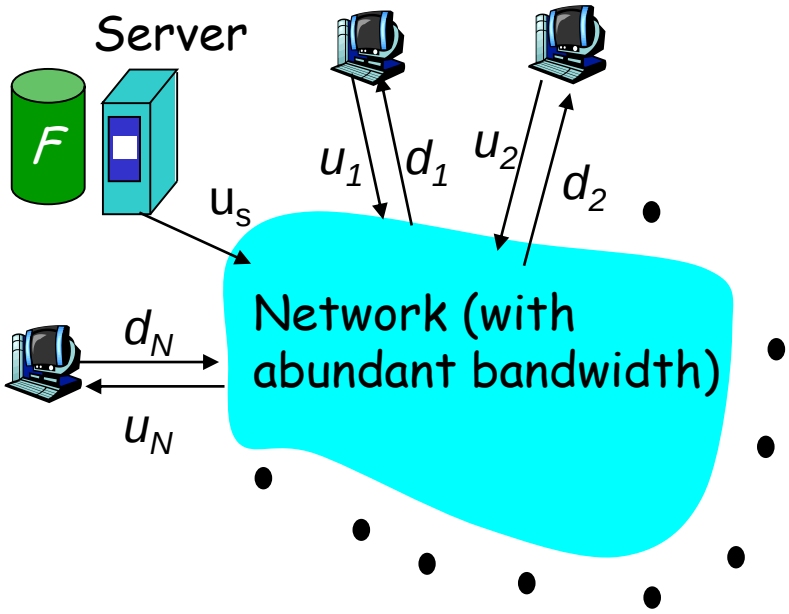
u_s: server upload bandwidth

u_i: peer i upload bandwidth

d_i: peer i download bandwidth

File distribution time: server-client

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

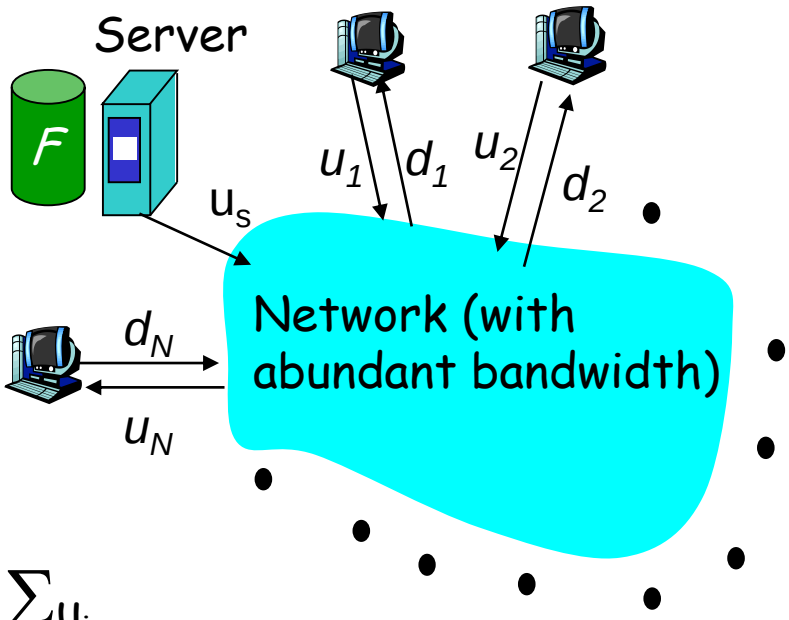


Time to distribute F to N clients using client/server approach $= d_{cs} = \max_i \{ NF/u_s, F/\min(d_i) \}$

increases linearly in N (for large N)

File distribution time: P2P

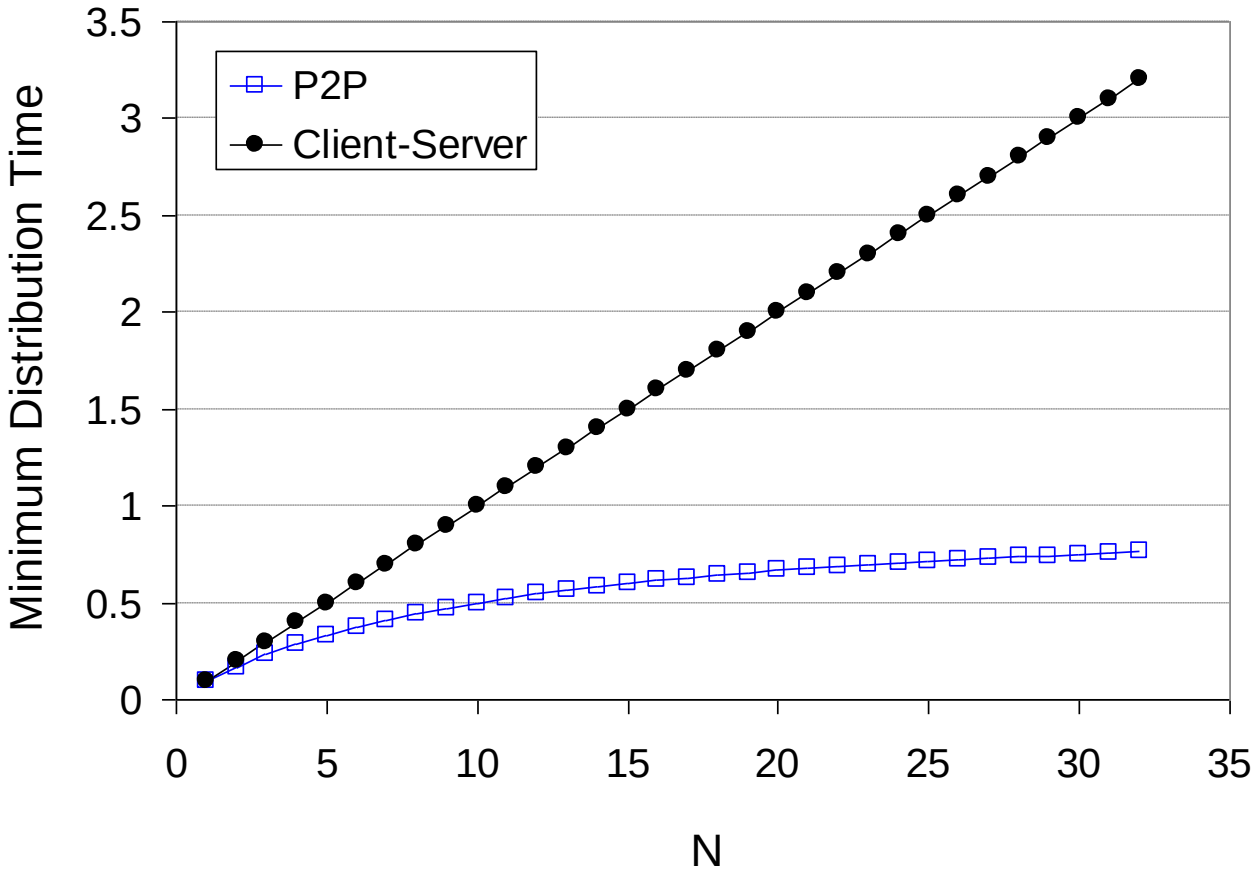
- server must send one copy: F/u_s time
- client i takes F/d_i time to download
- NF bits must be downloaded (aggregate)
  - fastest possible upload rate: $u_s + \sum u_i$



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

Server-client vs. P2P: example

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



BitTorrent

- Arguably biggest source of p2p traffic
  - P2P 54%-70% of ISP traffic
  - BT 20%-57% of ISP traffic

Ipoque Internet study 2008/09

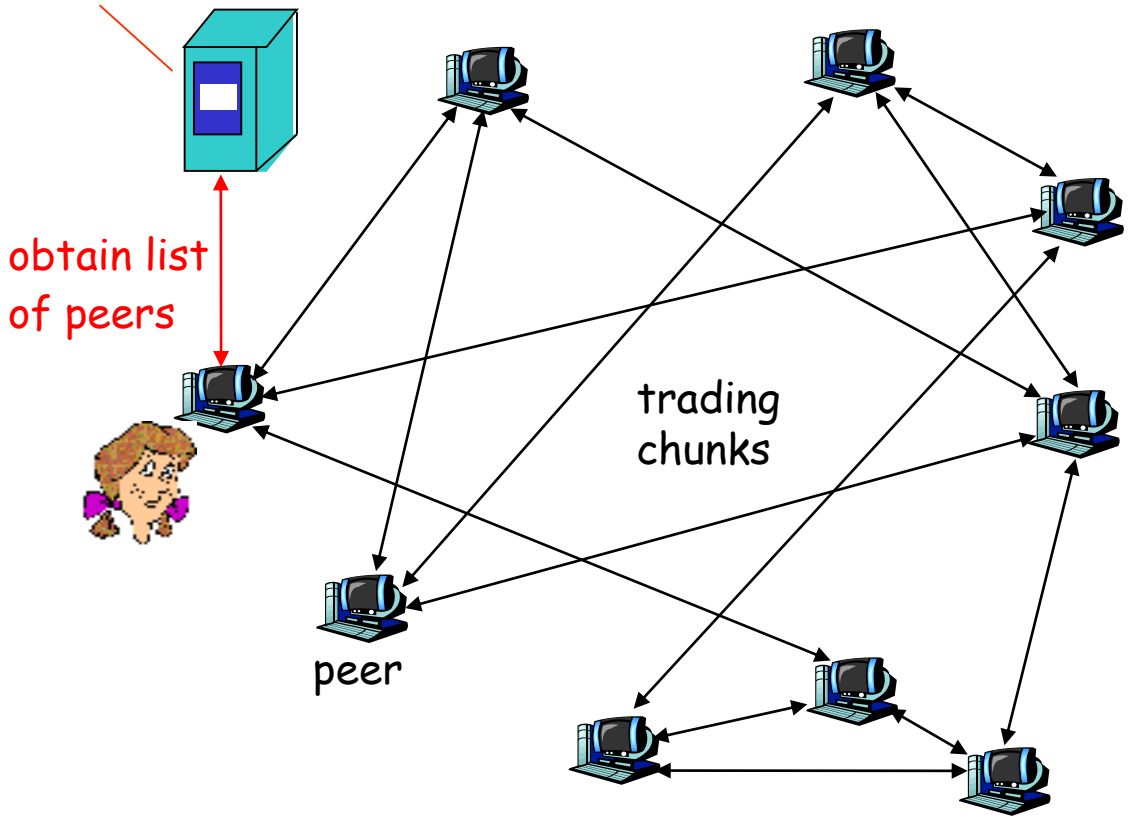
- Second generation file-sharing protocol
  - Contents split into many small pieces
    - Pieces are downloaded from both leechers and seeds
  - Distribution paths are dynamically determined
    - Based on data availability
  - One overlay per content

File distribution: BitTorrent

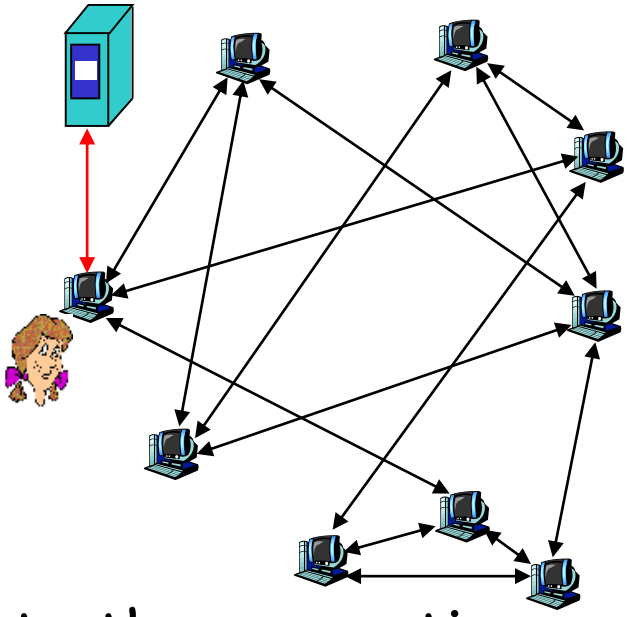
- P2P file distribution

tracker: tracks peers participating in torrent

torrent: group of peers exchanging chunks of a file



BitTorrent (1)

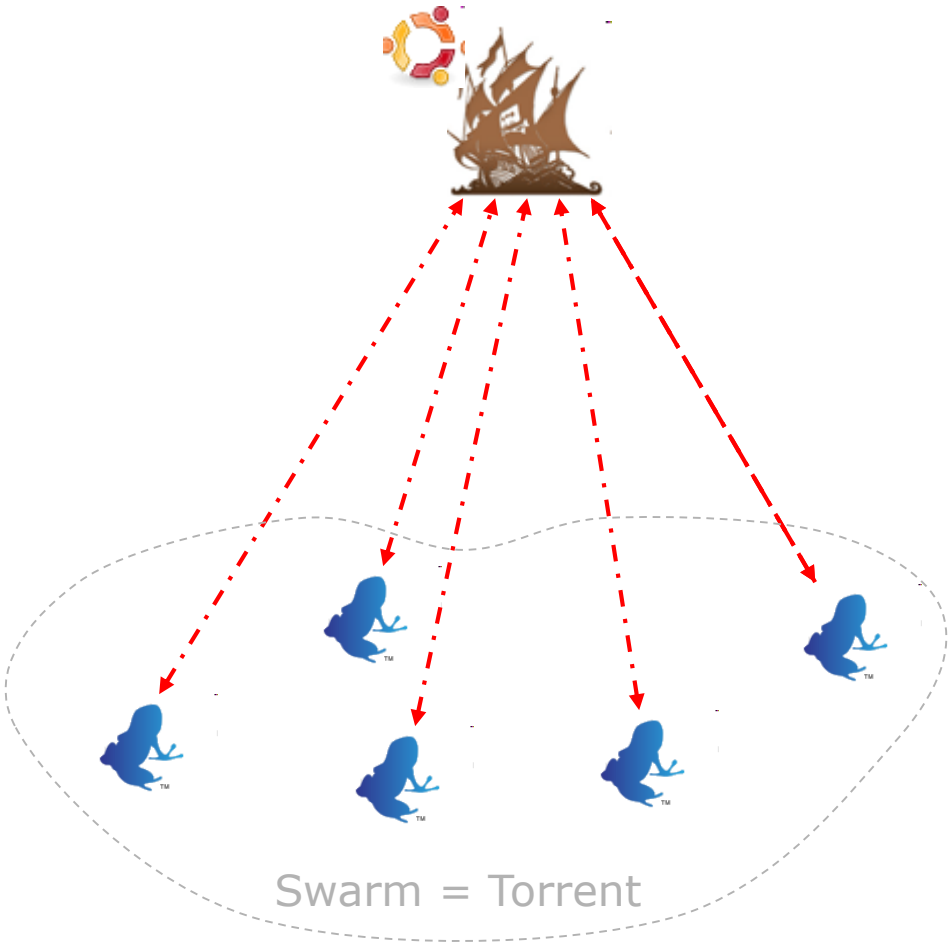


- file divided into 256KB *chunks*.
- peer joining torrent:
  - has no chunks, but will accumulate them over time
  - registers with tracker to get list of peers, connects to subset of peers ("neighbors")
- while downloading, peer uploads chunks to other peers.
- peers may come and go
- once peer has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

Background

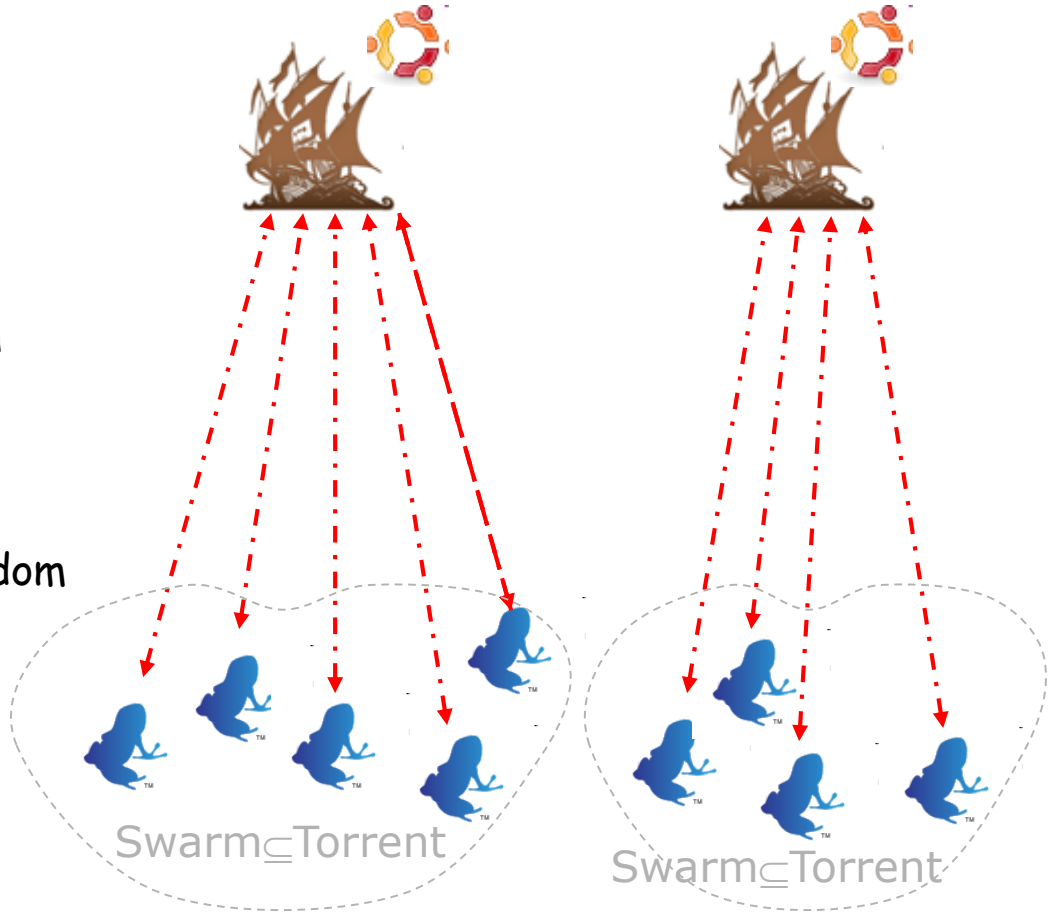
Peer discovery in BitTorrent

- Torrent file
  - Tracker address
- Trackers
  - Register torrent file
  - Maintain state information
  - Swarm – torrent
- Peers
  - Obtain torrent file
  - Announce
  - Report status
  - Peer exchange (PEX)
- Issues
  - Central point of failure
  - Tracker load



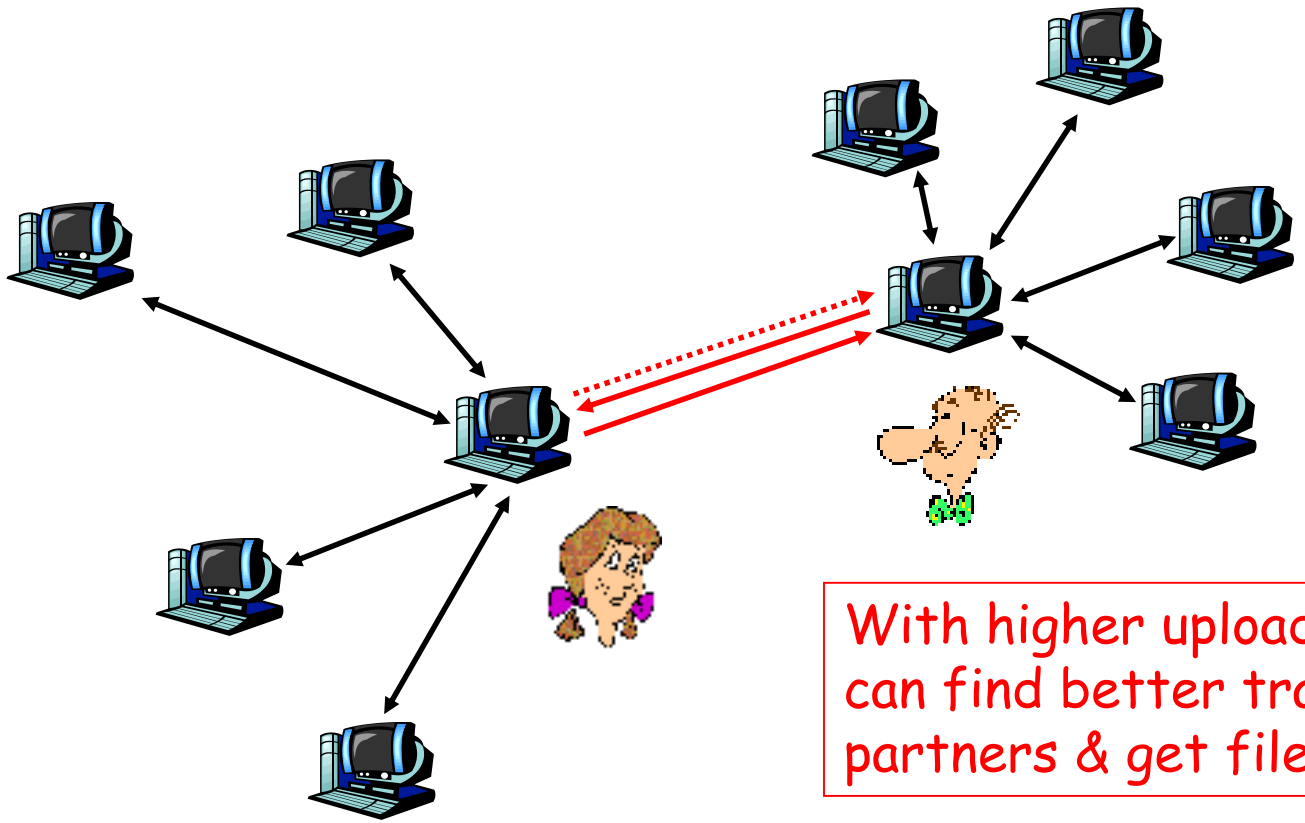
Background
Multi-tracked torrents

- Torrent file
  - Trackers' addresses
- Trackers
  - Register torrent file
  - Maintain state information
  - Swarm – torrent
- Peers
  - Obtain torrent file
  - Choose one tracker at random
  - Announce
  - Report status
  - Peer exchange (PEX)
- Issue
  - Multiple smaller swarms



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

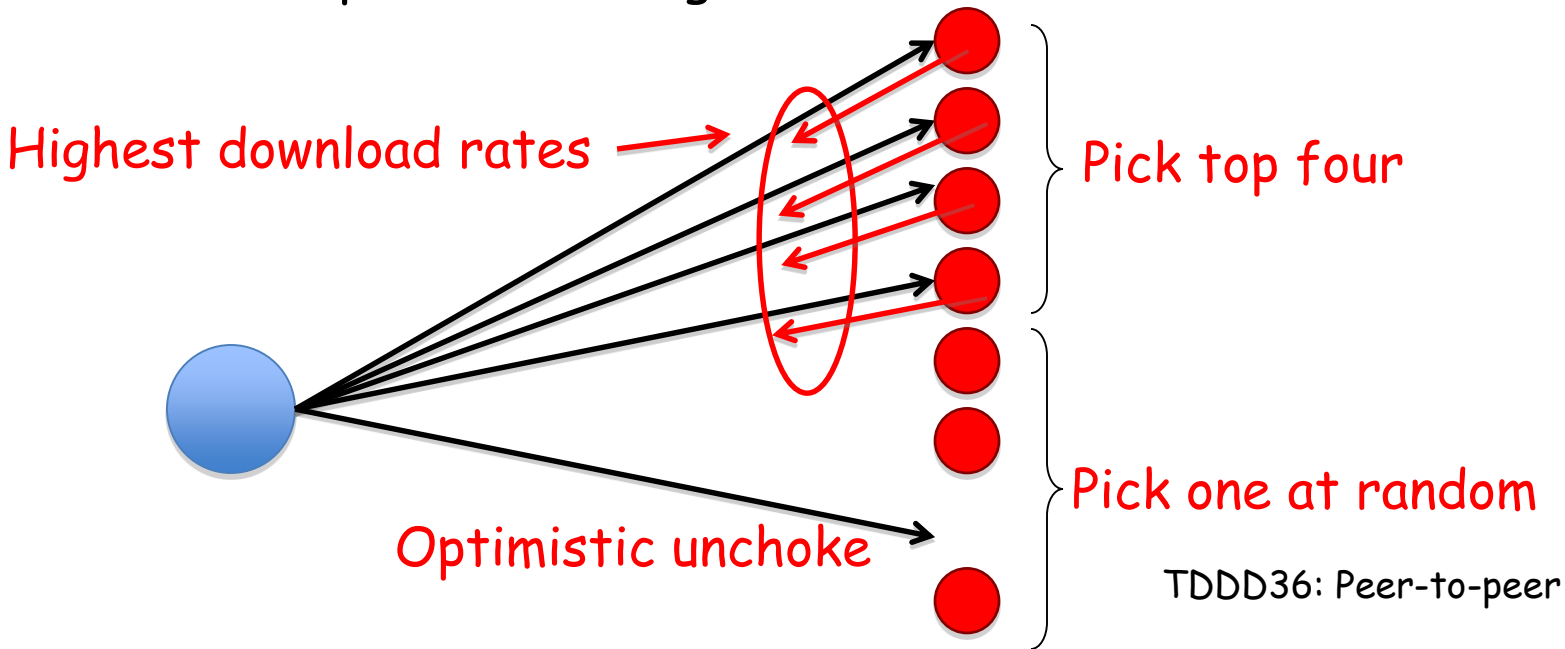


With higher upload rate, can find better trading partners & get file faster!

Download using BitTorrent

Background: Incentive mechanism

- Establish connections to large set of peers
  - At each time, only upload to a small (changing) set of peers
- Rate-based tit-for-tat policy
  - Downloaders give upload preference to the downloaders that provide the highest download rates



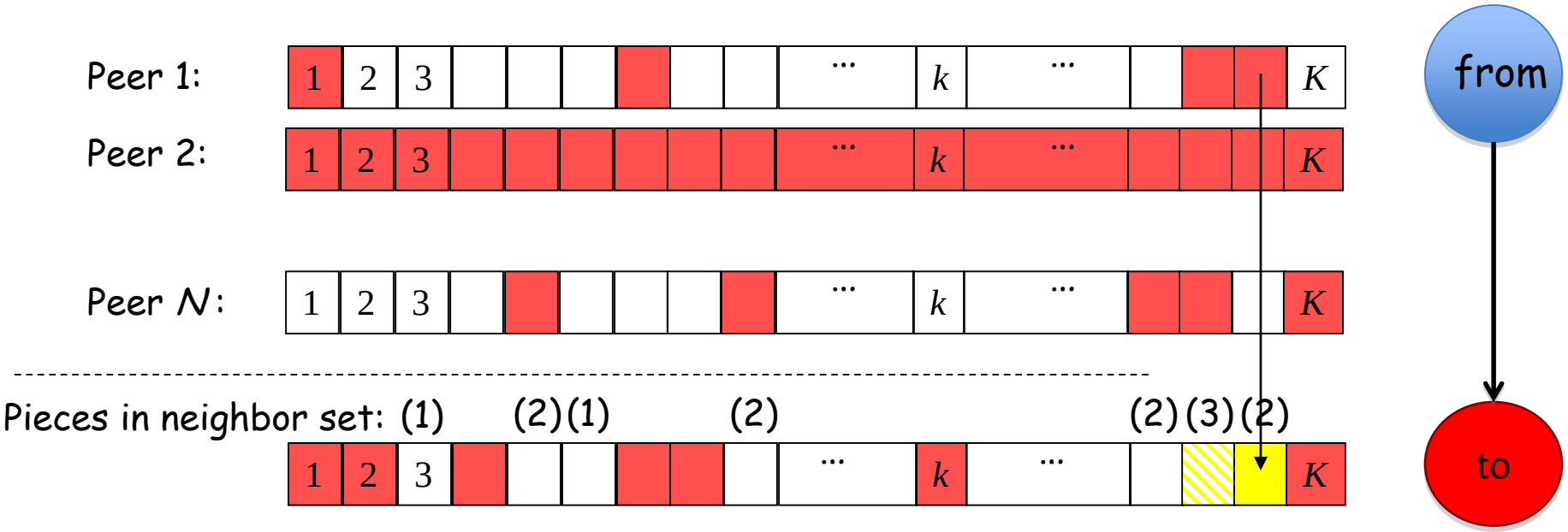
BitTorrent: Rarest first

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

Download using BitTorrent

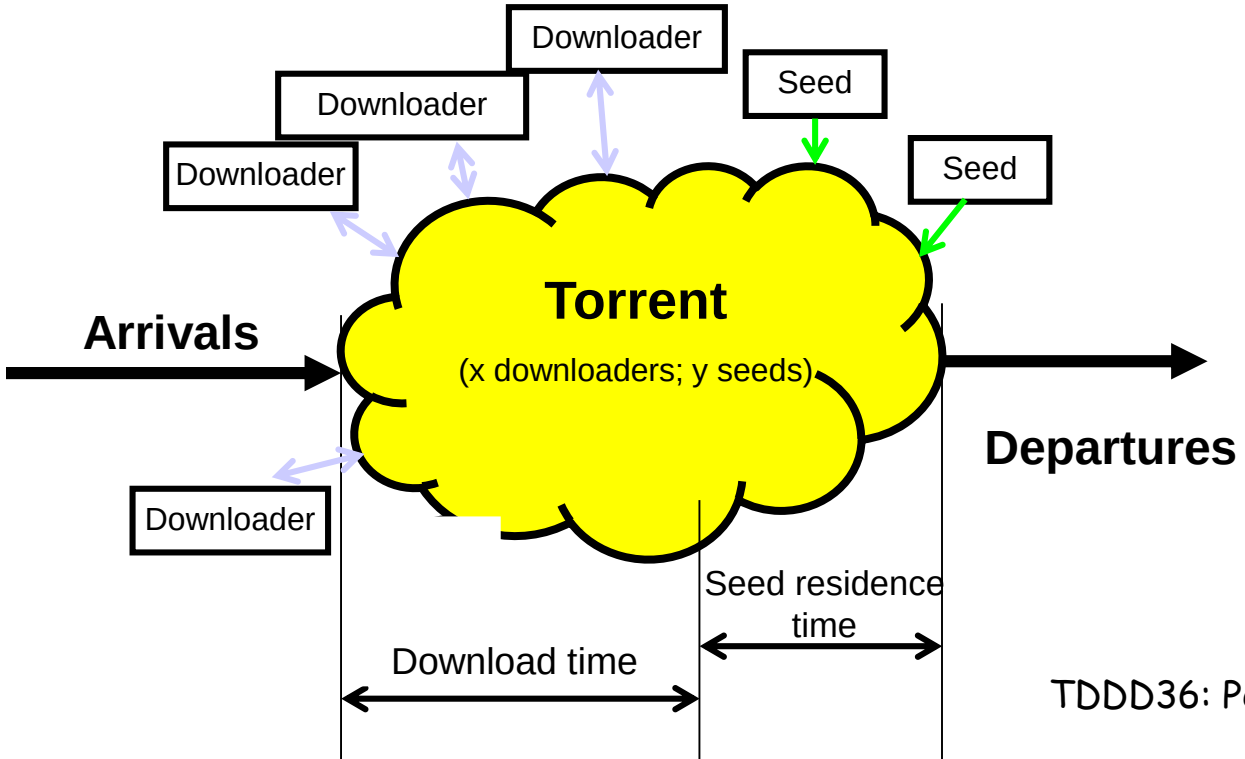
Background: Piece selection



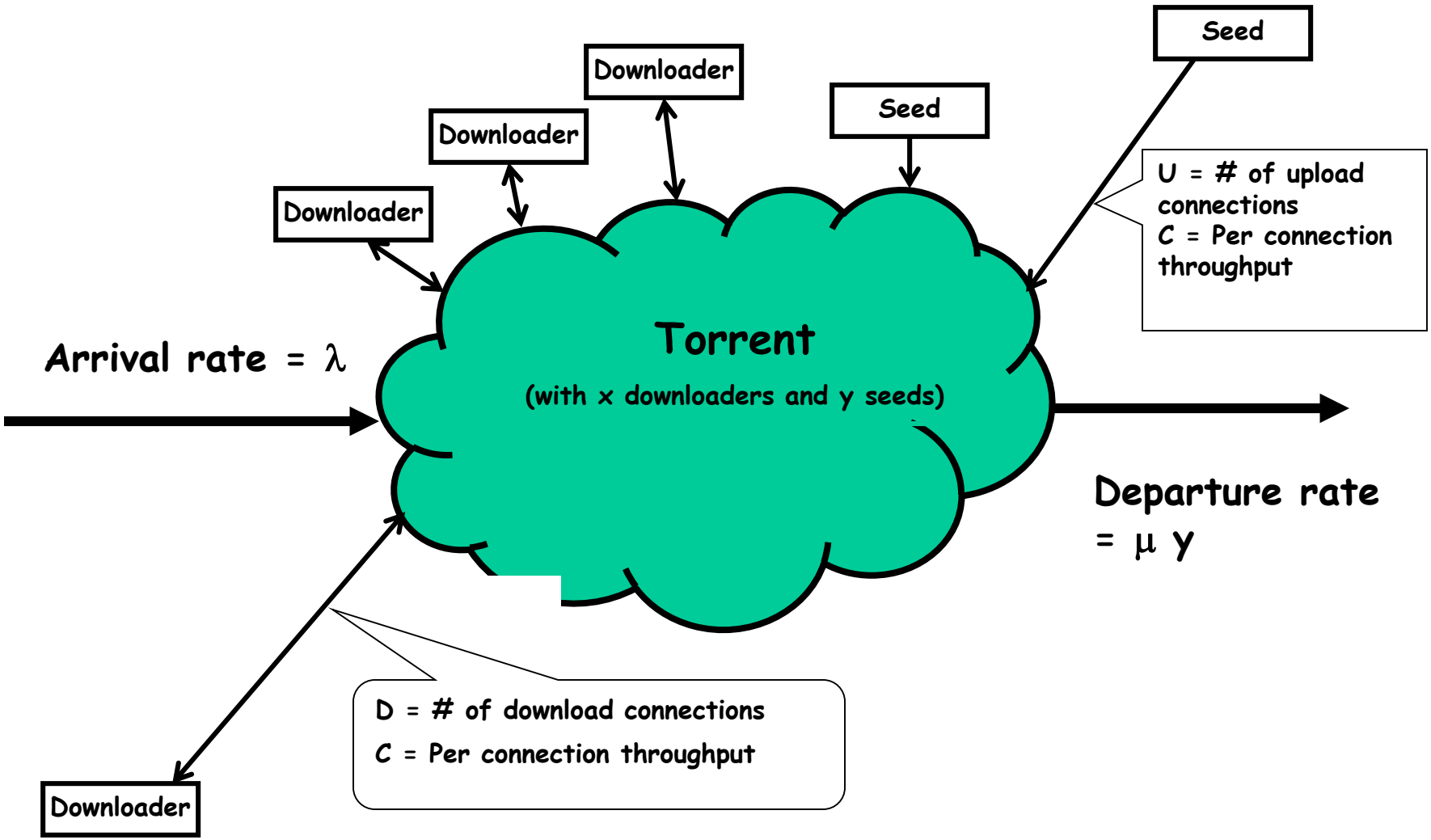
- Rarest first piece selection policy
  - Achieves high piece diversity
- Request pieces that
  - the uploader has;
  - the downloader is interested (wants); and
  - is the rarest among this set of pieces

BitTorrent-like systems

- File split into many smaller pieces
- Pieces are downloaded from both seeds and downloaders
- Distribution paths are dynamically determined
  - Based on data availability



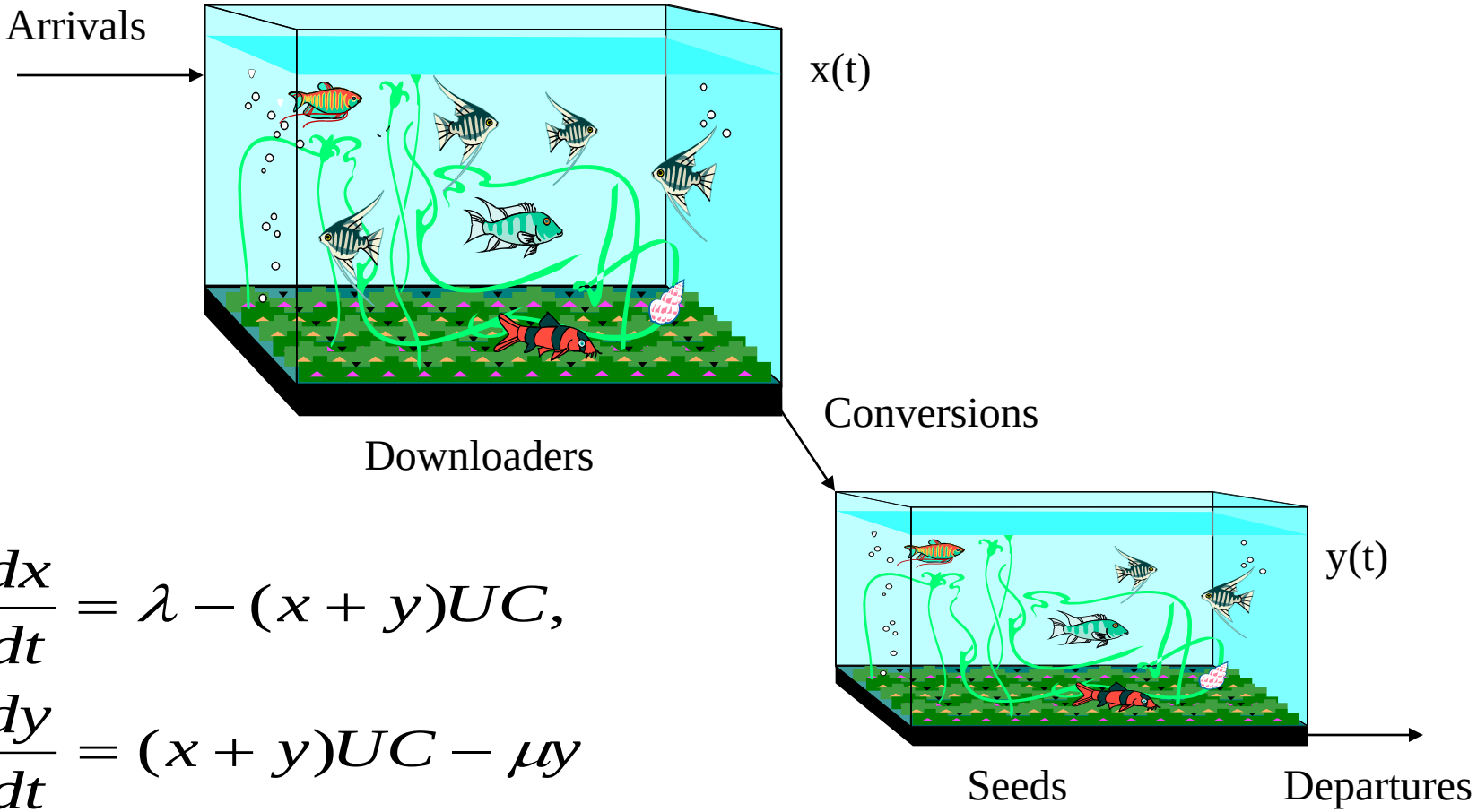
BitTorrent Model



Assumptions and Parameters

- Single swarm; homogeneous peers
- x downloaders and y seeds at time t
- D download conns > U upload conns
- System is demand-driven: xD > (x+y)U
- Download latency = T
- Number of pieces in the file = M
- Startup delay = τ
- Media Playback Rate = r

Fluid Model Overview



$$\frac{dx}{dt} = \lambda - (x + y)UC,$$

$$\frac{dy}{dt} = (x + y)UC - \mu y$$

Model: Rarest-First

- Conversion of downloaders to seeds at rate (x+y)UC.

- Therefore the change of swarm population:

$$\frac{dx}{dt} = \lambda - (x+y)UC,$$

$$\frac{dy}{dt} = (x+y)UC - \mu y$$

Model: Rarest-First

- Download latency: $T = \frac{1}{UC} - \frac{1}{\mu}$
- Sequential progress: $k = M + 1 - \sqrt{(M+1)/r}$
- Startup delay: $\tau = 1 - \frac{2\sqrt{(M+1)} - 2}{M}$