CS 43: Computer Networks

09: P2P Systems & BitTorrent

October 6, 2020

SWARTHMORE COLLEGE

Slides Courtesy: Kurose & Ross, K. Webb, D. Choffnes

Last class

- Application Layer: SMTP & Email

Today

- P2P vs Client-Server applications

- P2P examples
  - Napster

- BitTorrent
  - Cooperative file transfers

Where we are

Application: the application (So far: HTTP, Email, DNS)
Today: BitTorrent, Skype, P2P systems

Transport: end-to-end connections, reliability

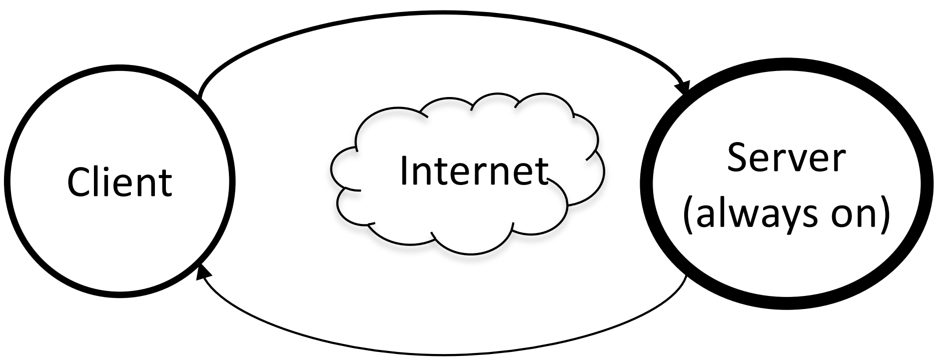
Network: routing

Link (data-link): framing, error detection

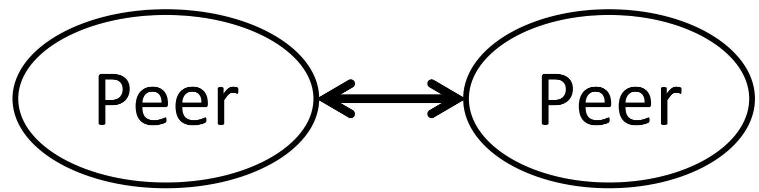
Physical: 1's and 0's/bits across a medium
(copper, the air, fiber)

Designating roles to an endpoint

Client-server architecture



Peer-to-peer architecture



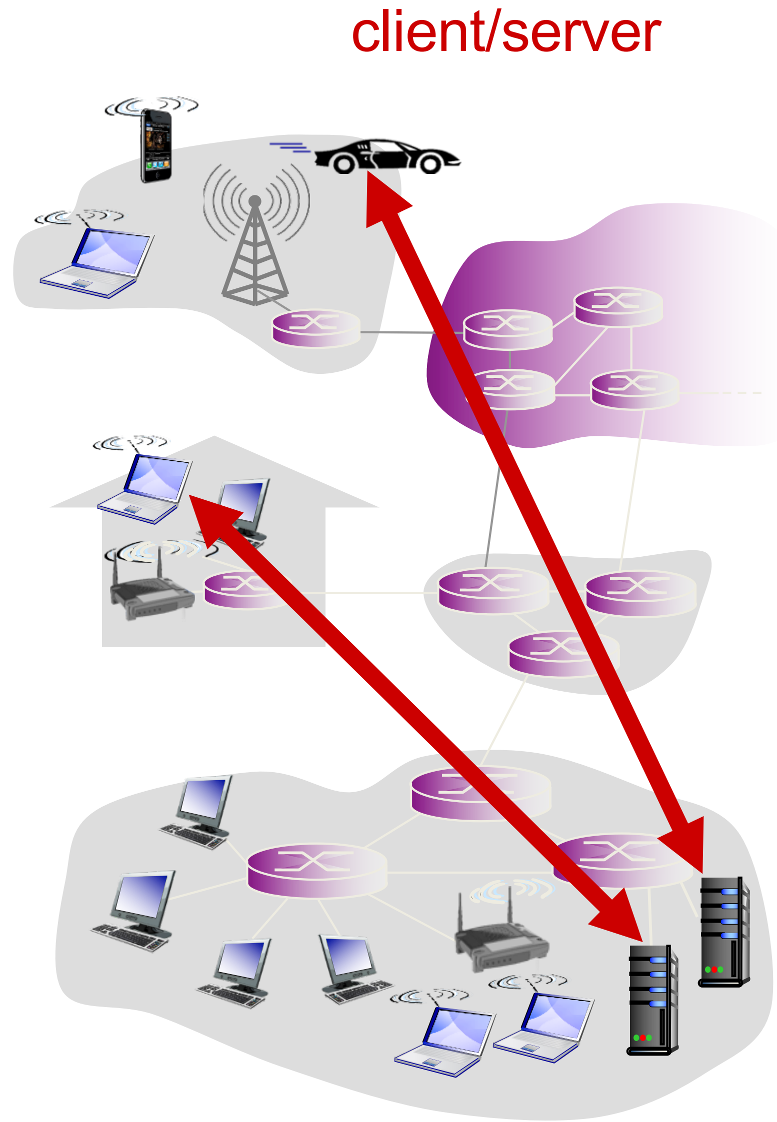
Client-Server Architecture

server:

- always-on host
- permanent IP address
- data centers for scaling

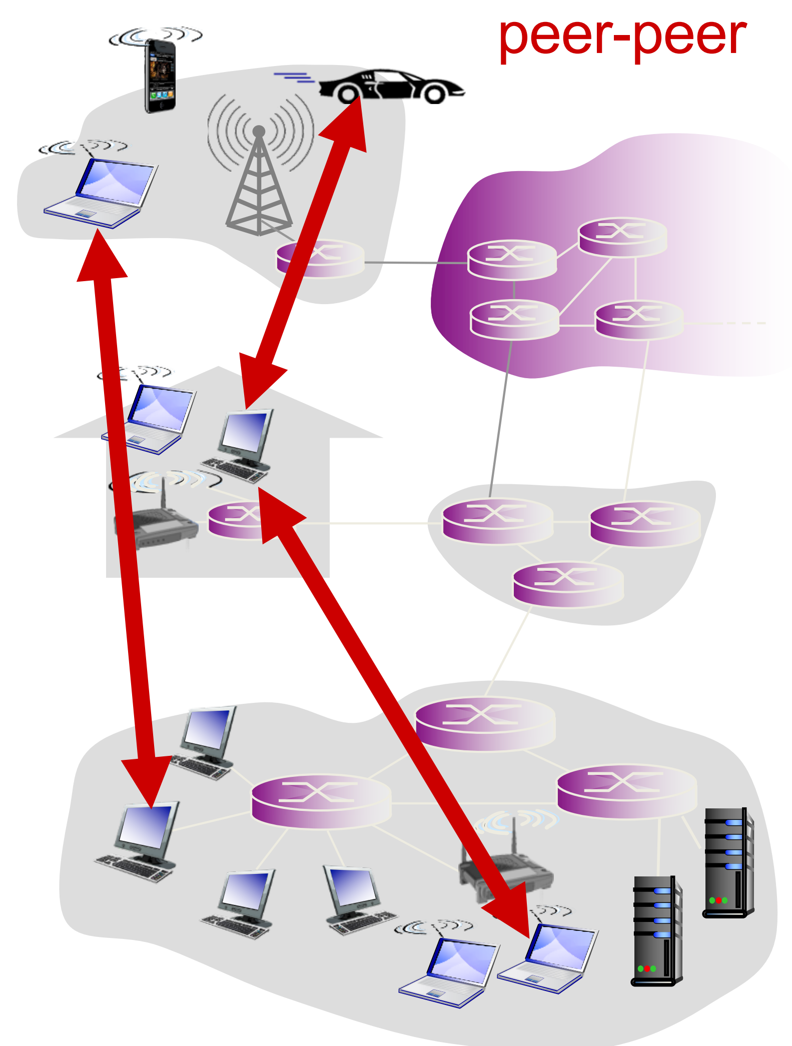
clients:

- communicate with server
- may be intermittently connected
- may have dynamic IP addresses
- do not communicate directly with each other



Peer-to-Peer Architecture

- no always-on server
- A peer talks directly with another peer
  - Symmetric responsibility (unlike client/server)
- peers request service from other peers, provide service in return to other peers
  - self scalability – new peers bring new service capacity, as well as new service demands
- peers are intermittently connected and change IP addresses
  - complex management



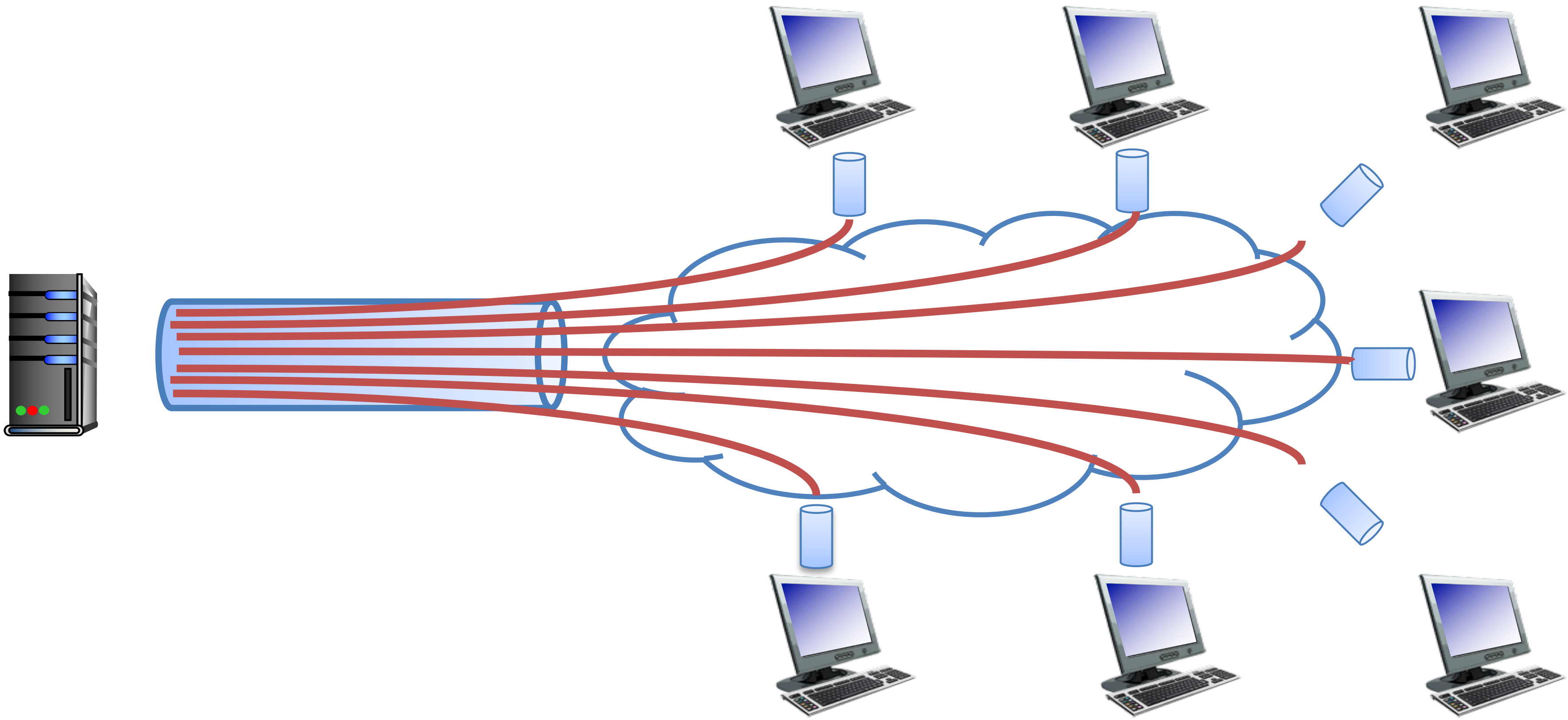
File Transfer Problem

- You want to distribute a file to a large number of people as quickly as possible.

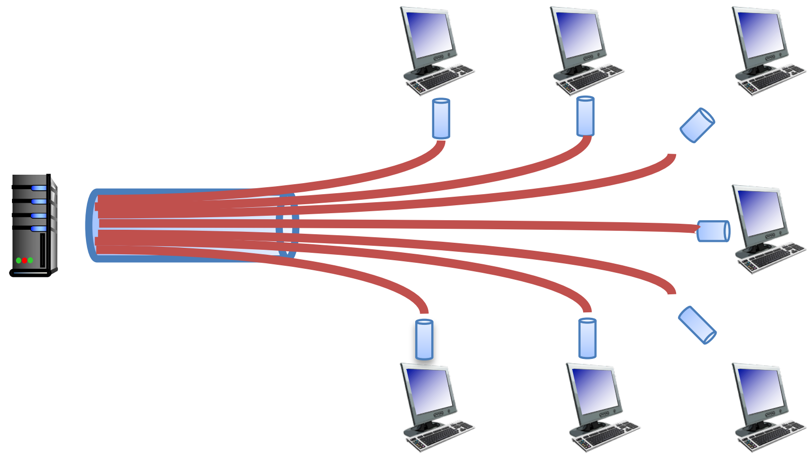
Traditional Client/Server

- Many clients, 1 (or more) server(s)
- Web servers, DNS, file downloads, video streaming

Traditional Client/Server

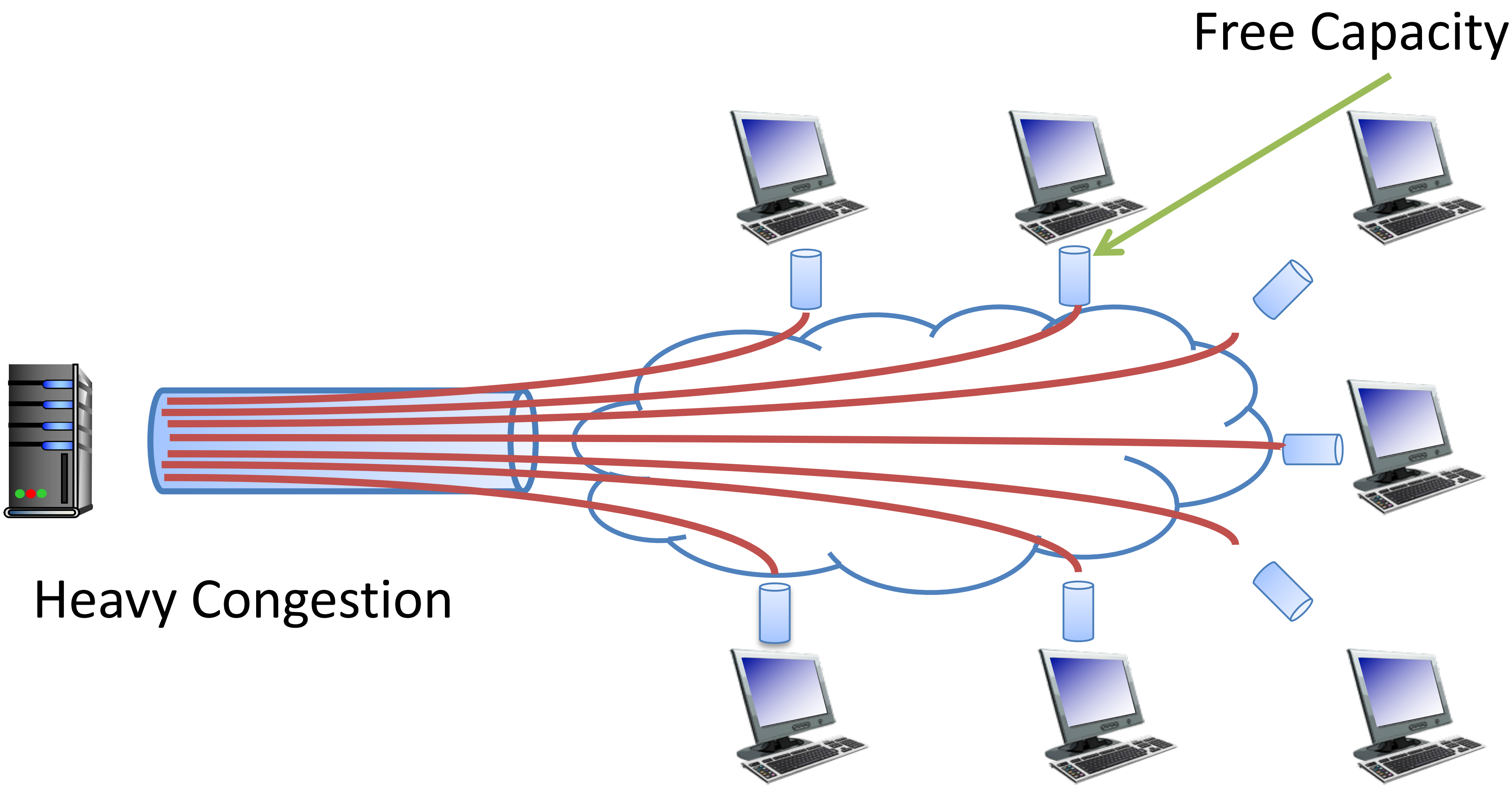


What is the biggest problem you run into with the traditional C/S model?

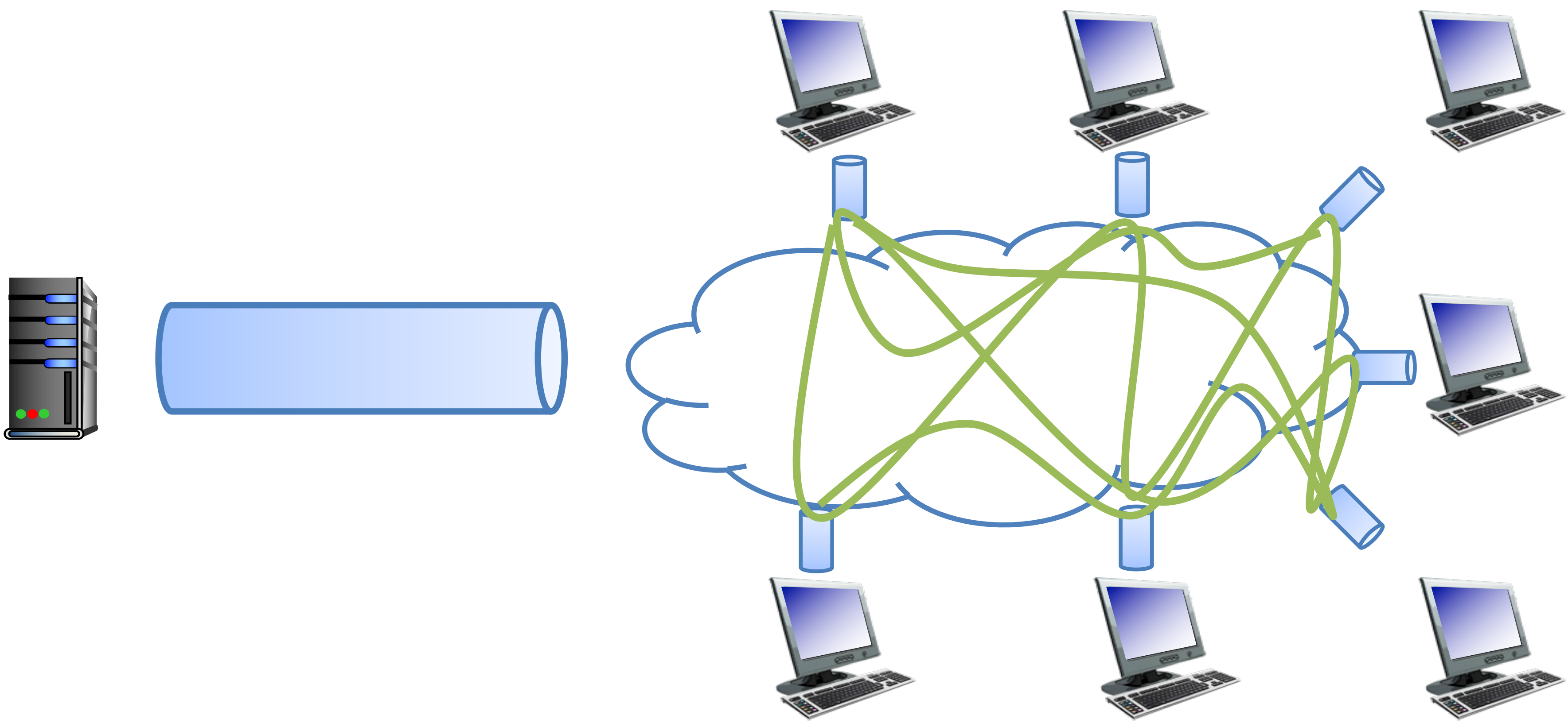


A. Scalability (how many end-hosts can you support?)
B. Reliability (what happens on failure?)
C. Efficiency (fast response time)

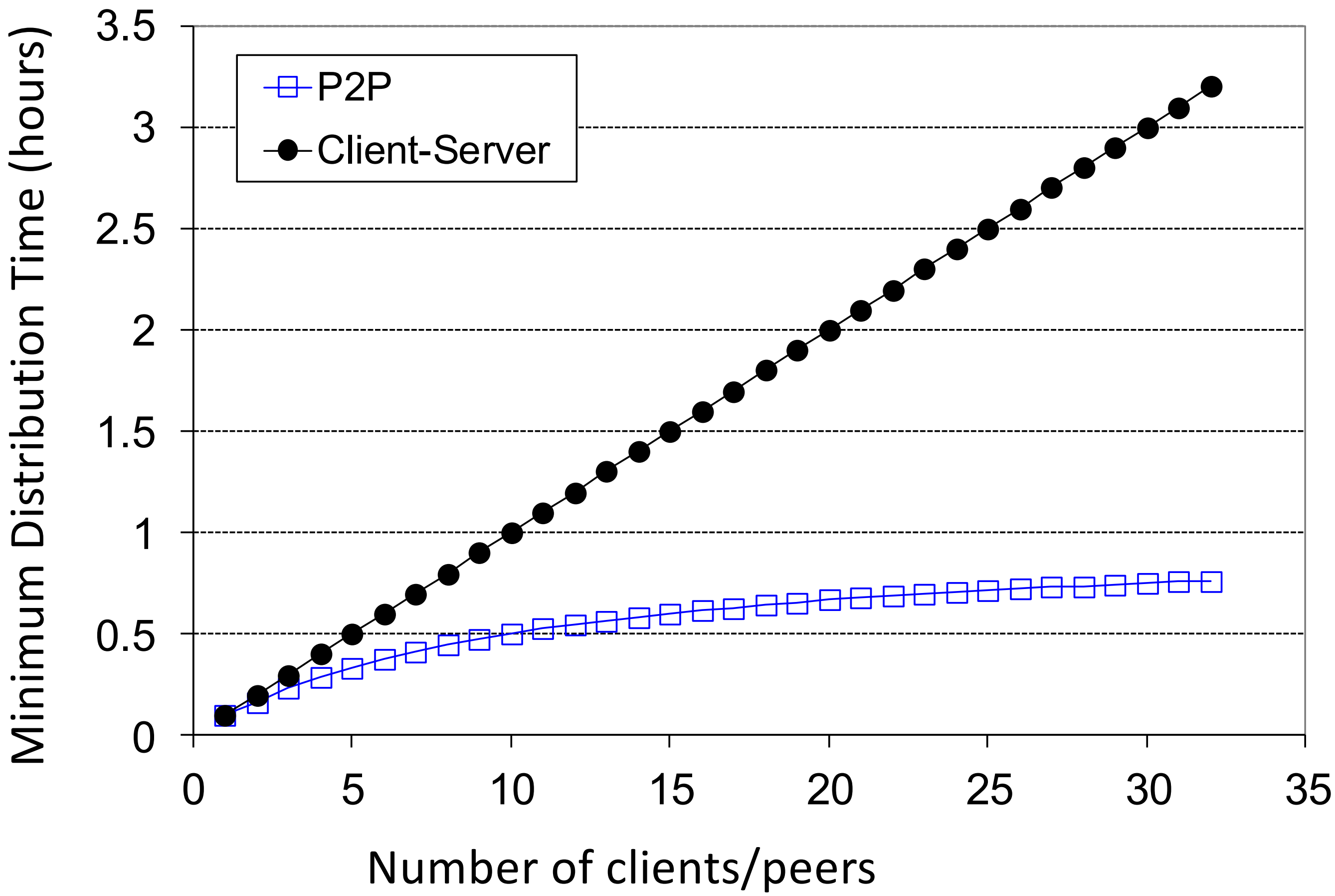
Traditional Client/Server



P2P Solution



Client-server vs. P2P: example



In a peer-to-peer architecture, are there clients and servers?

A. Yes

B. No

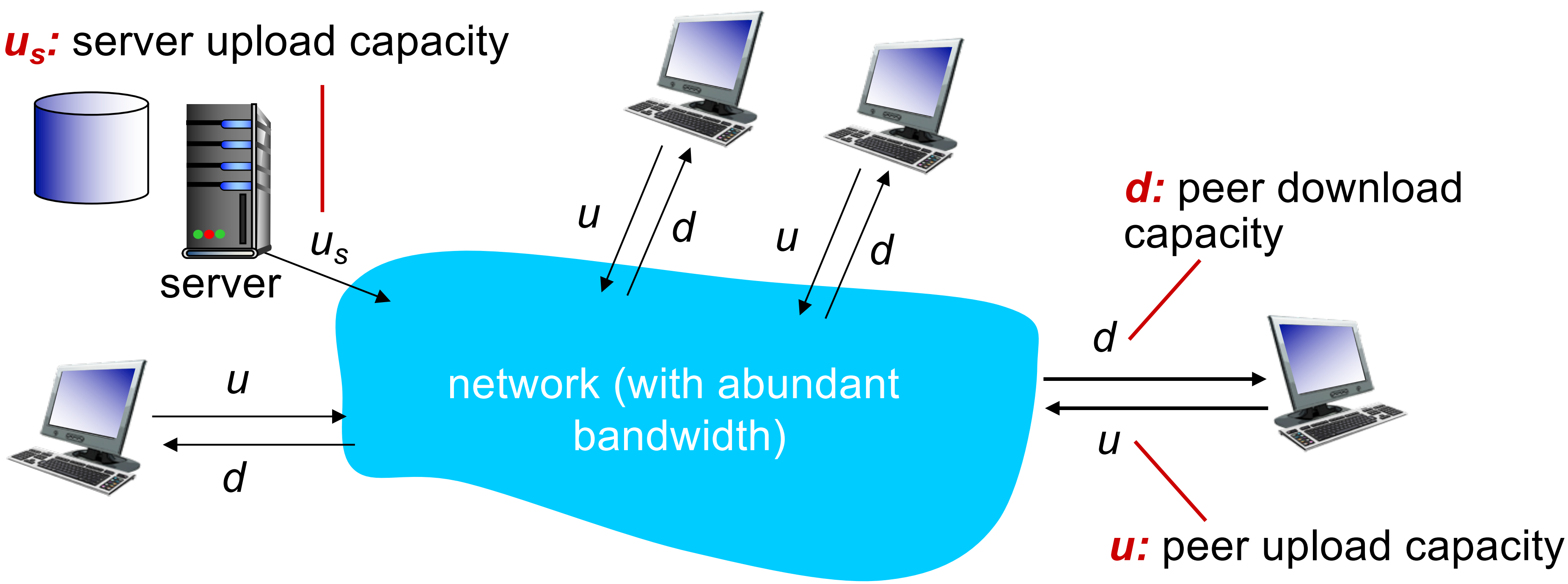
In a peer-to-peer architecture, are there clients and servers?

A. Yes (peers can both send and receive data and act as the TCP server calling bind and listen or the TCP client)

B. No

File size = 6 Gbits = 6000 Mb (megabits)
Number of peers = 10
Server upload rate of u_s = 100 Mbps (megabits per second)
Peer upload rate of u = 20Mbps
Peer download rate of d = 50Mbps

Worksheet Question



C/S Model

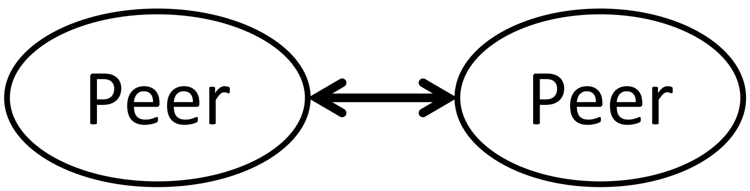
- Minimum time to distribute the file = max(time to upload the file, time to download the file)
- Time to upload the file = NF/u_s = 6000*10/100 = 600s
- Time to download the file = 6000/50 = 120s
- Min time = 600s.

P2P Model

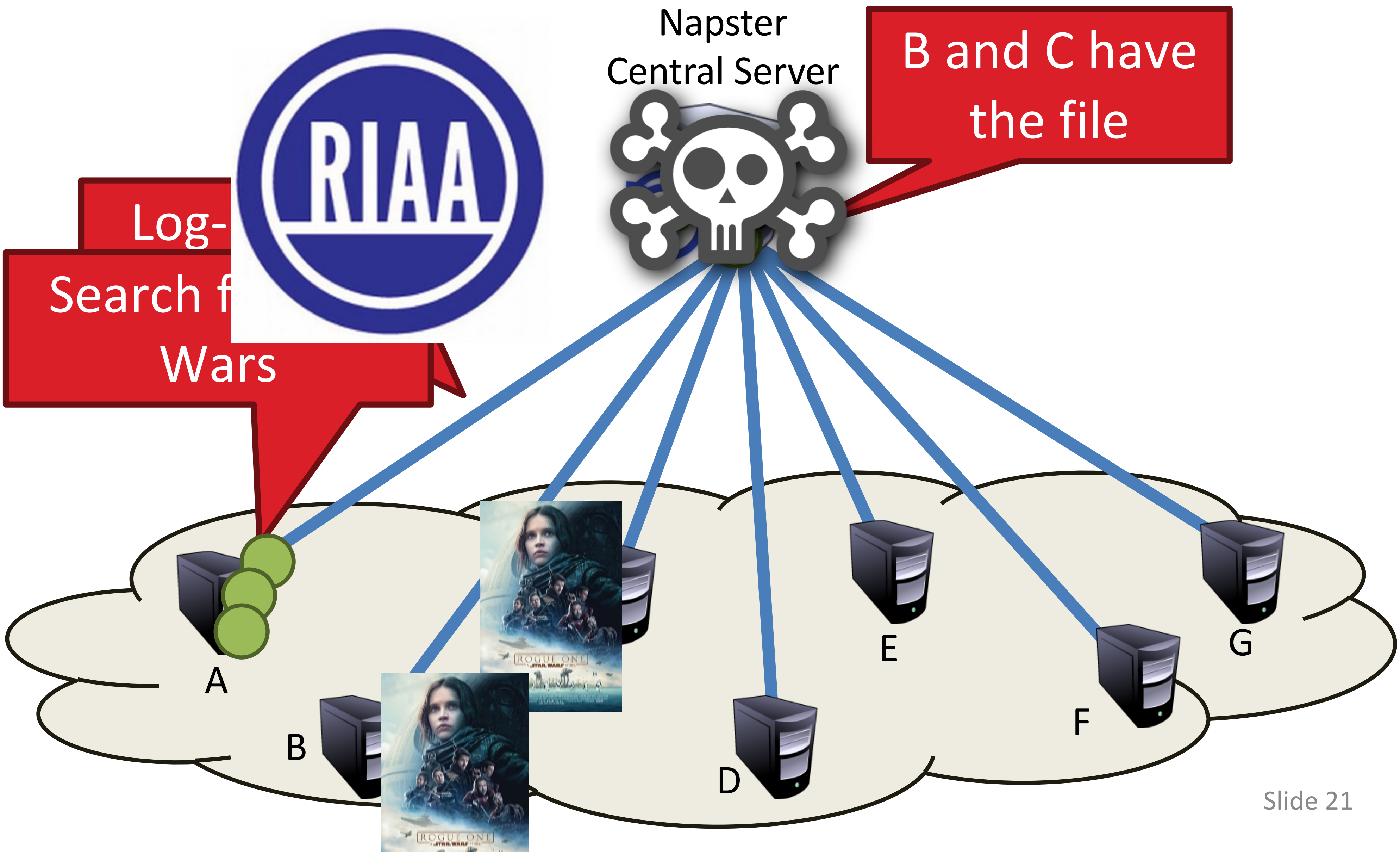
- Minimum time to distribute the file = max(time to upload the file, time to download the file)
- Time to upload the file from the server = F/u_s = 6000/100 = 60s
- Time to upload from peers to every other peer 6000*10/(100+20*10) = 200s
- Time to download the file = 6000/50 = 120s
- Min time = 200s

Designating roles to an endpoint

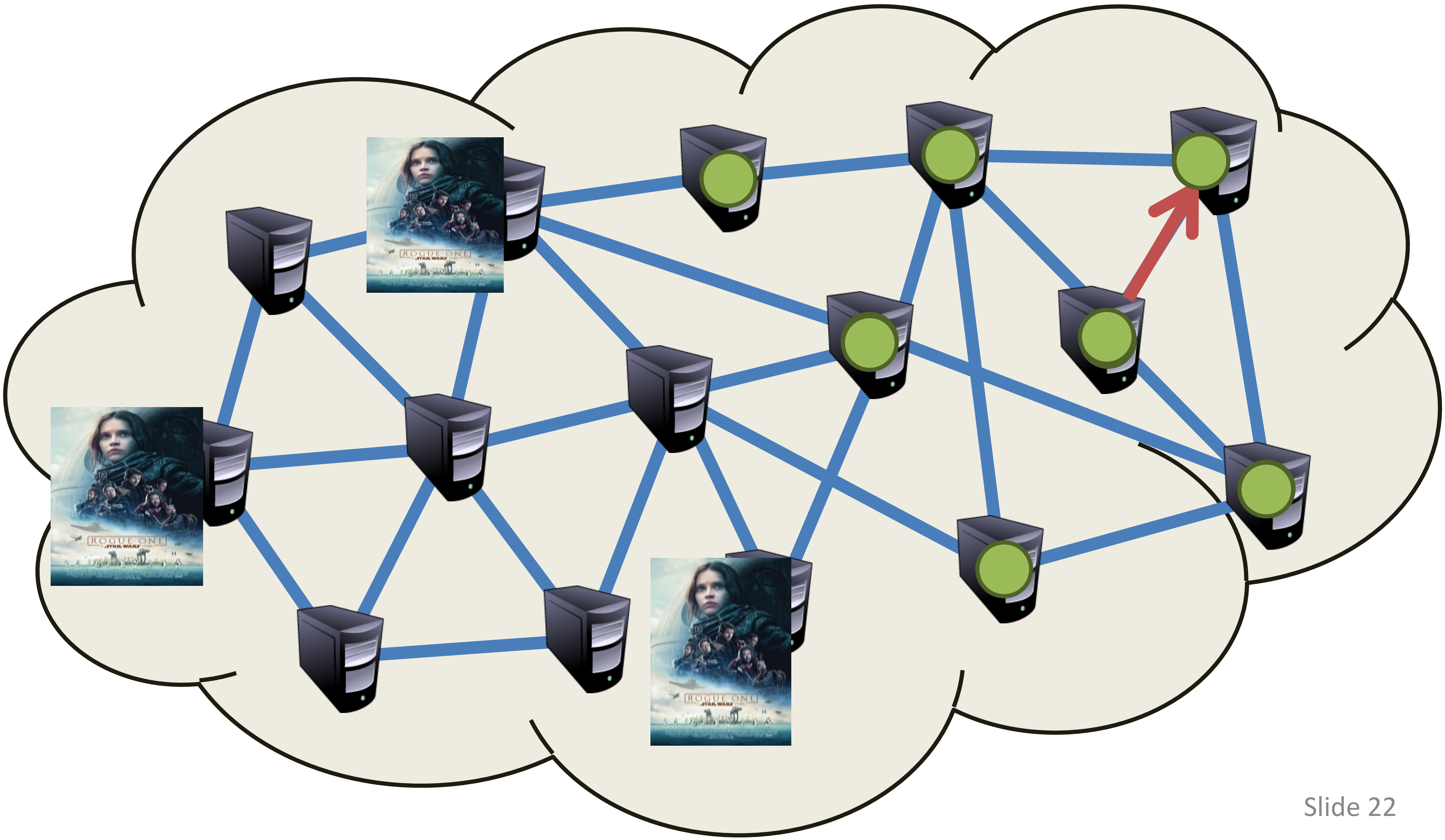
Peer-to-peer architecture



Napster Architecture

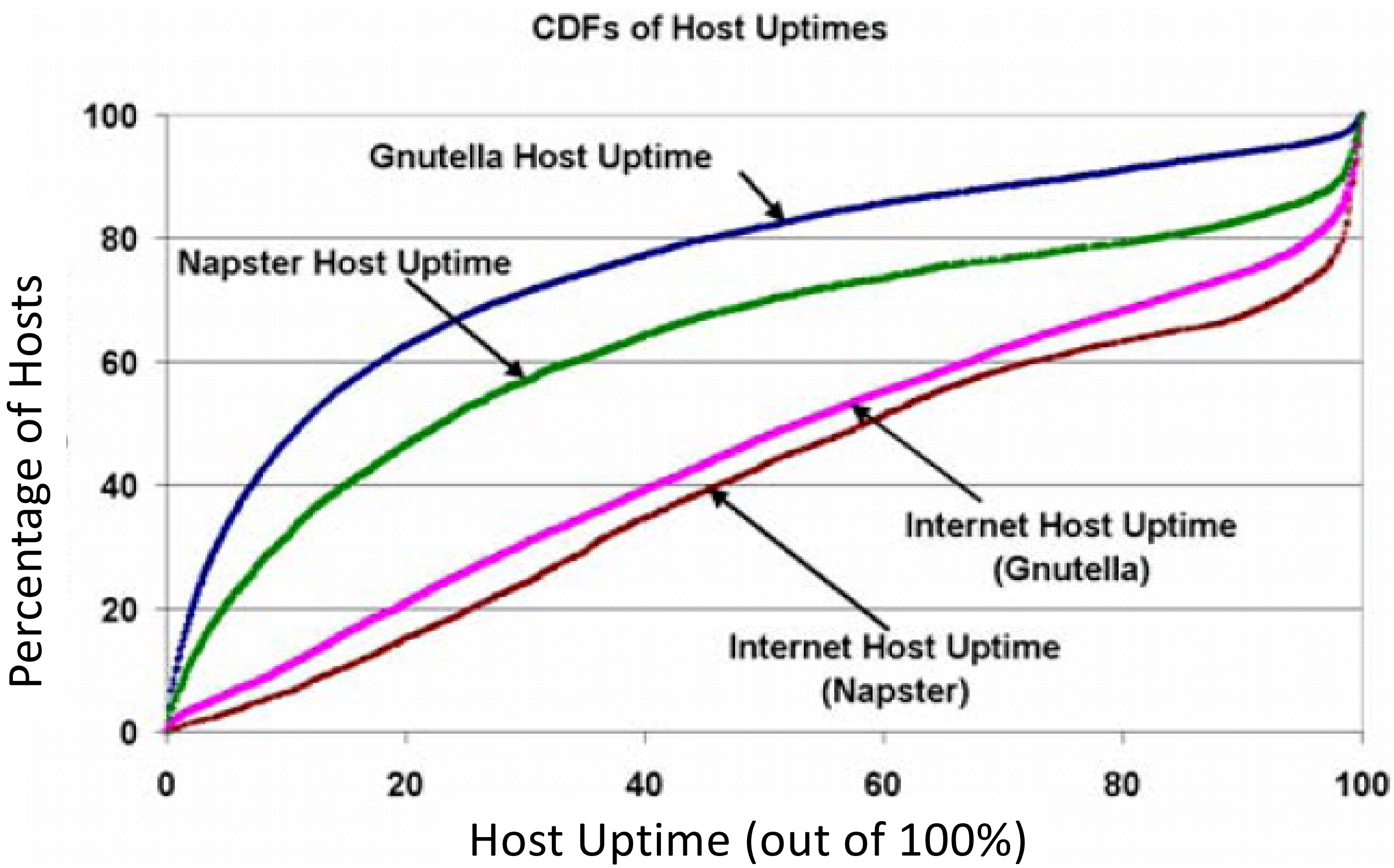


File Search via Flooding in Gnutella



Peer Lifetimes: Highly available?

“only 20% of the peers in each system had an IP-level uptime of 93% or more.”



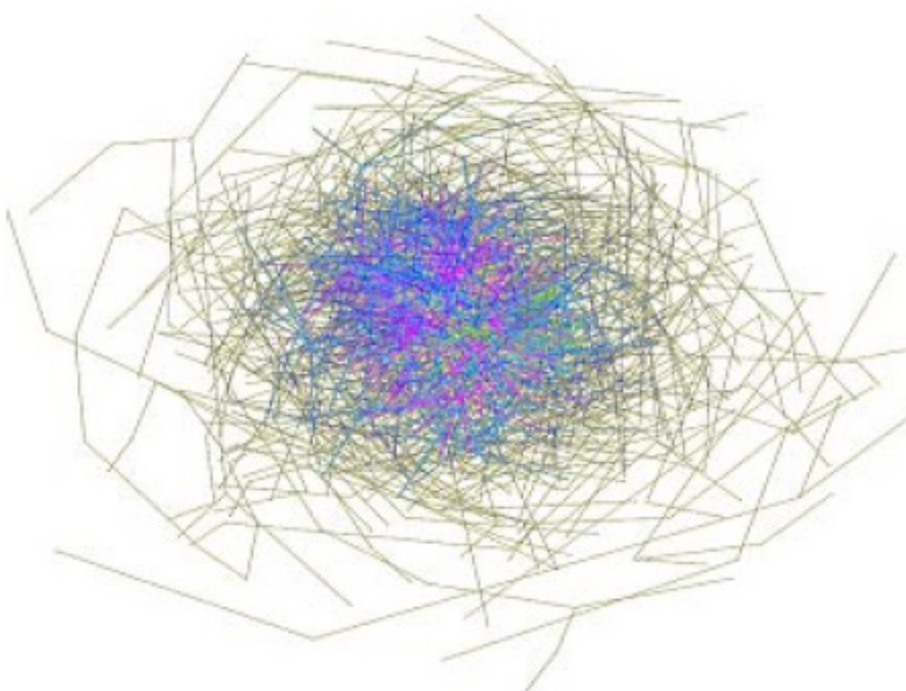
Sessions are short ~60 minutes

Hosts are frequently offline

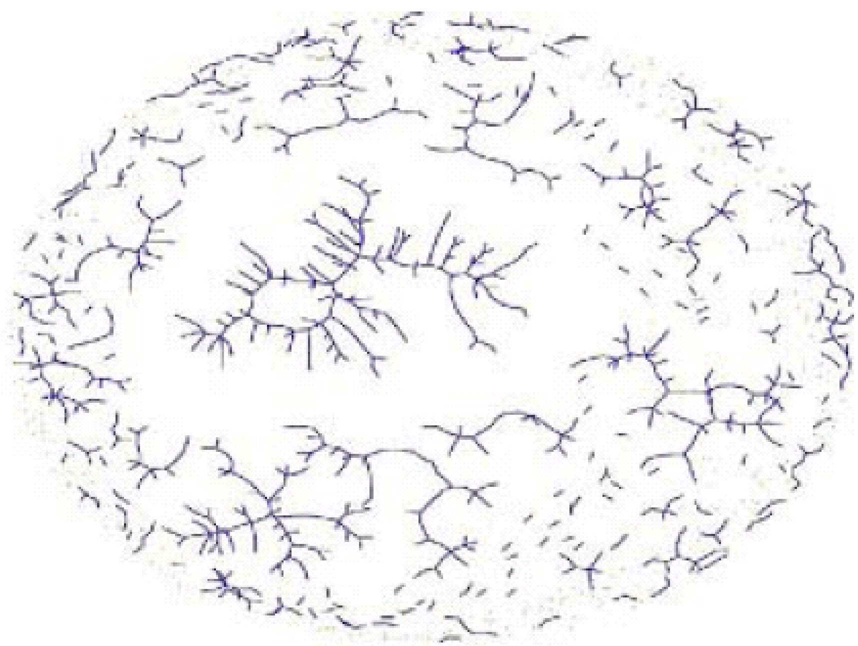
Study of host uptime and application uptime (MMCN 2002)

Resilience to Failures and Attacks

- Previous studies (Barabasi) show interesting dichotomy of resilience for "scale-free networks"
  - Resilient to random failures, but not attacks
- Here's what it looks like for Gnutella



1771 Peers in Feb, 2001

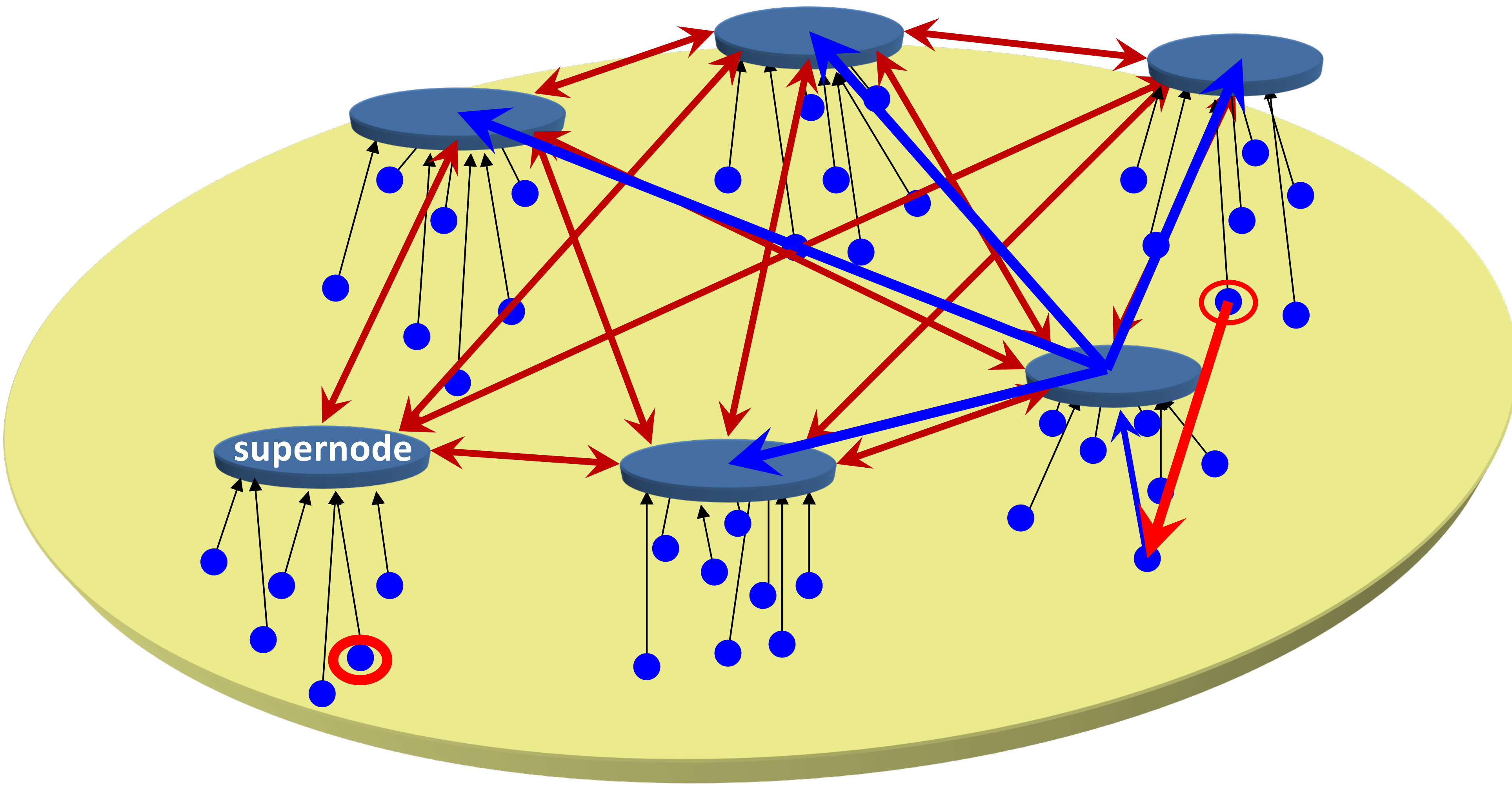


After top 4% of peers are removed

After random 30% of peers removed

Hierarchical P2P Networks

- FastTrack network (Kazaa, Grokster, Morpheus, Gnutella++)



Skype: P2P VoIP

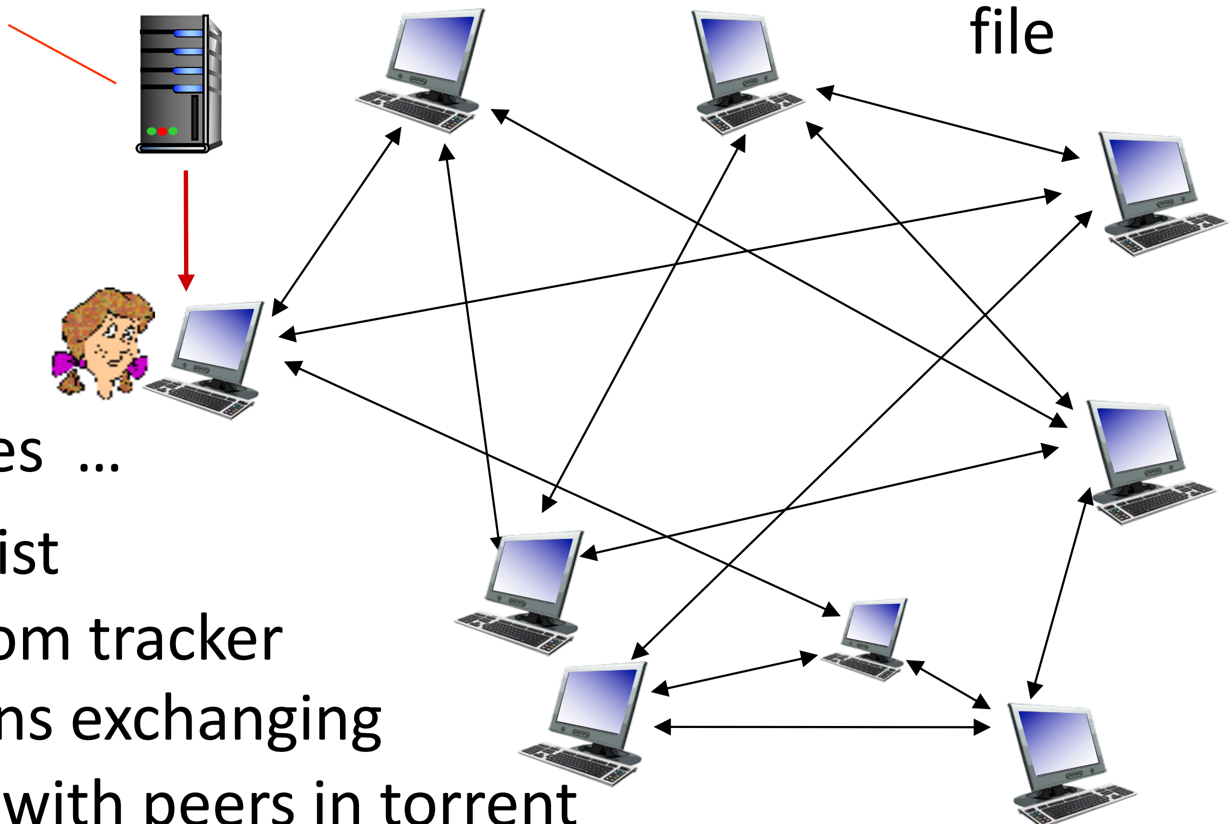
- P2P client supporting VoIP, video, and text based conversation, buddy lists, etc.
  - Overlay P2P network consisting of ordinary and Super Nodes (SN)
- Each user registers with a central server
  - User information propagated in a decentralized fashion

P2P file distribution: BitTorrent

- File divided into chunks (commonly 256 KB)
- 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

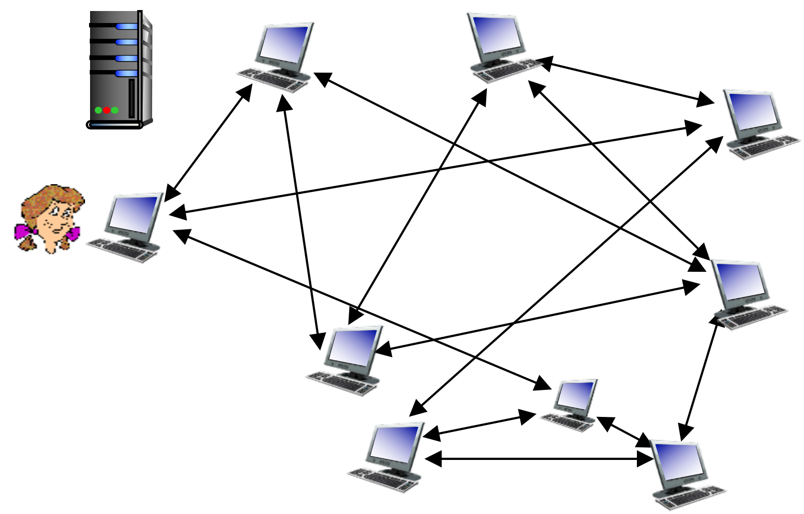
.torrent files

- Contains address of tracker for the file
  - Where can I find other peers?

- Contain a list of file chunks and their cryptographic hashes
  - This ensures pieces are not modified

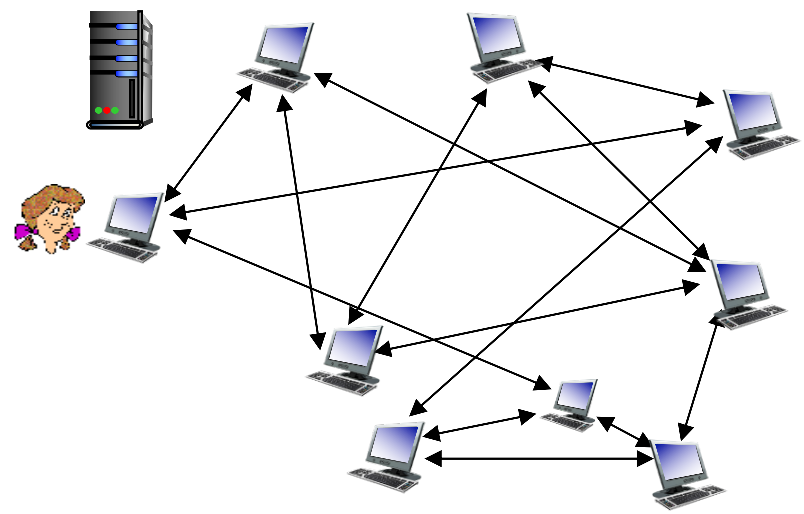
BitTorrent : Peer Joining

- 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

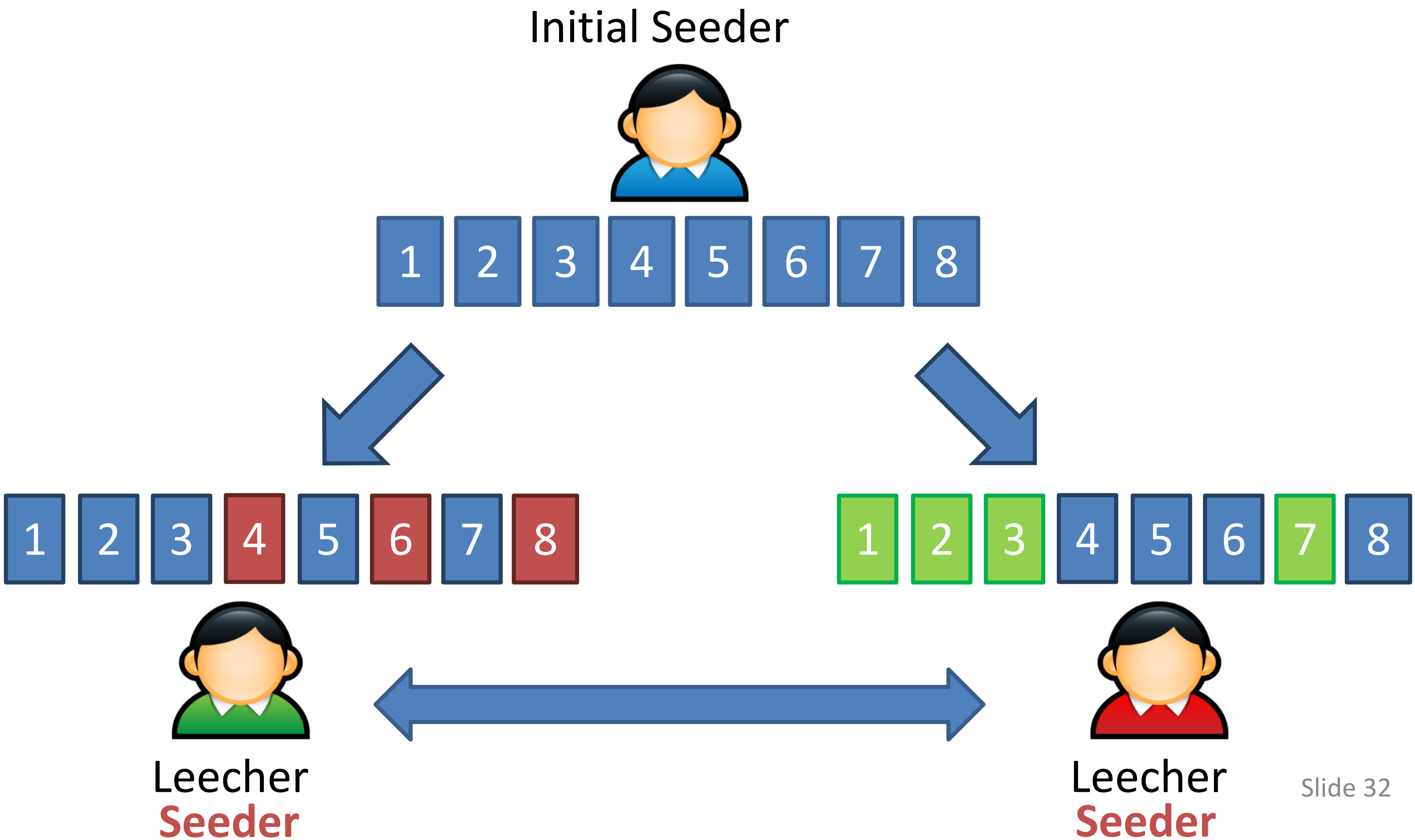
- While downloading, peer uploads chunks to other peers
- *Churn:* peers may come and go
  - Peer may change peers with whom it exchanges chunks



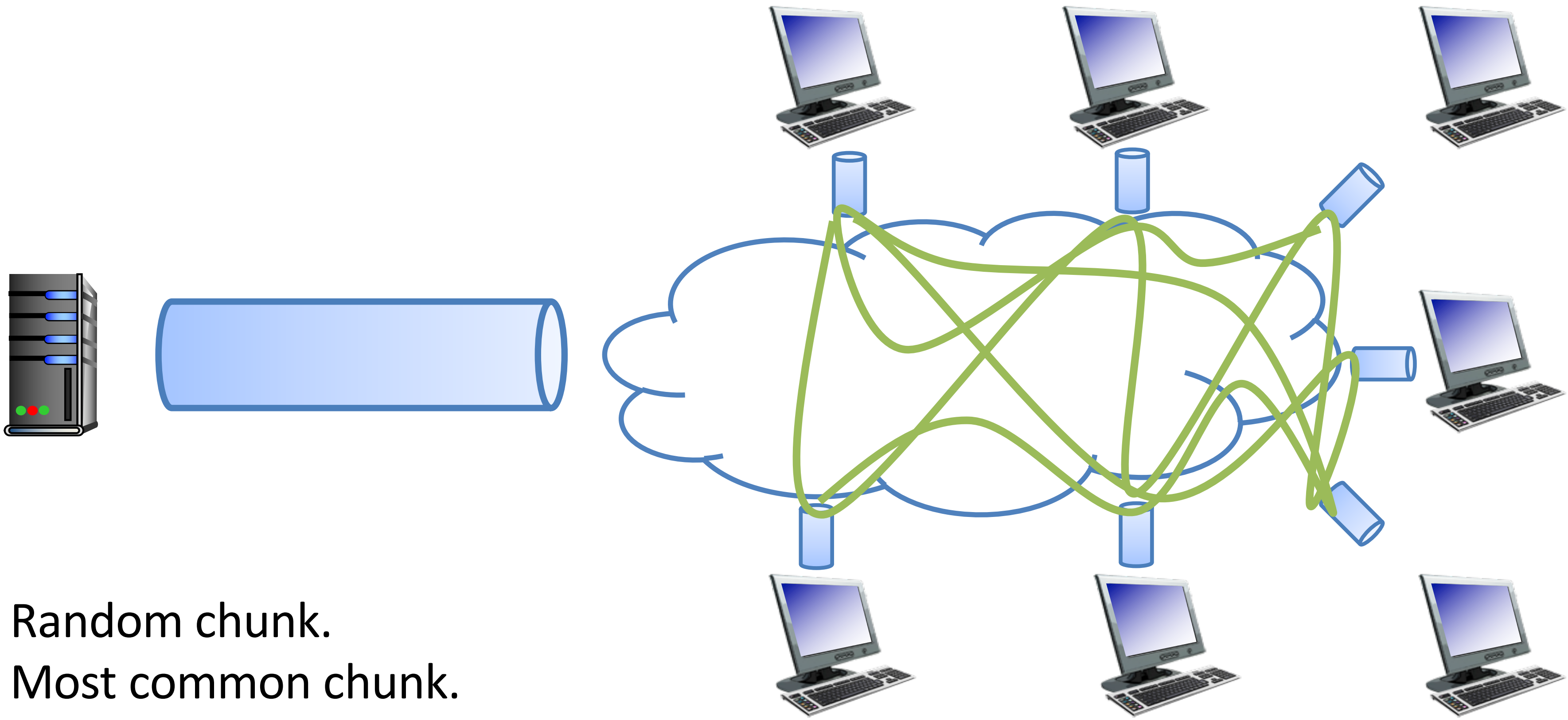
Requesting Chunks

- At any given time, peers have different subsets of file chunks.
- Periodically, ask peers for list of chunks that they have.
- Once peer has entire file, it may (selfishly) leave or (altruistically) remain in torrent

Sharing Pieces



If you're trying to receive a file, which chunk should you request next?



A. Random chunk.
B. Most common chunk.
C. Least common chunk.
D. Some other chunk.
E. It doesn't matter.

Requesting Chunks

0%

% Downloaded

100%

- Bootstrap: random selection
  - Initially, you have no pieces to trade
  - Essentially, beg for free pieces at random
- Steady-state: rarest piece first
  - Ensures that common pieces are saved for last
- Endgame
  - Simultaneously request final pieces from multiple peers
  - Cancel connections to slow peers
  - Ensures that final pieces arrive quickly

Sending Chunks: tit-for-tat

- A node sends chunks to those four peers currently sending it chunks at highest rate
  - other peers are choked (do not receive chunks)
  - re-evaluate top 4 every ~10 secs

- Every 30 seconds: randomly select another peer, start sending chunks
  - "optimistically unchoke" this peer
  - newly chosen peer may join top 4

Academic Interest in BitTorrent

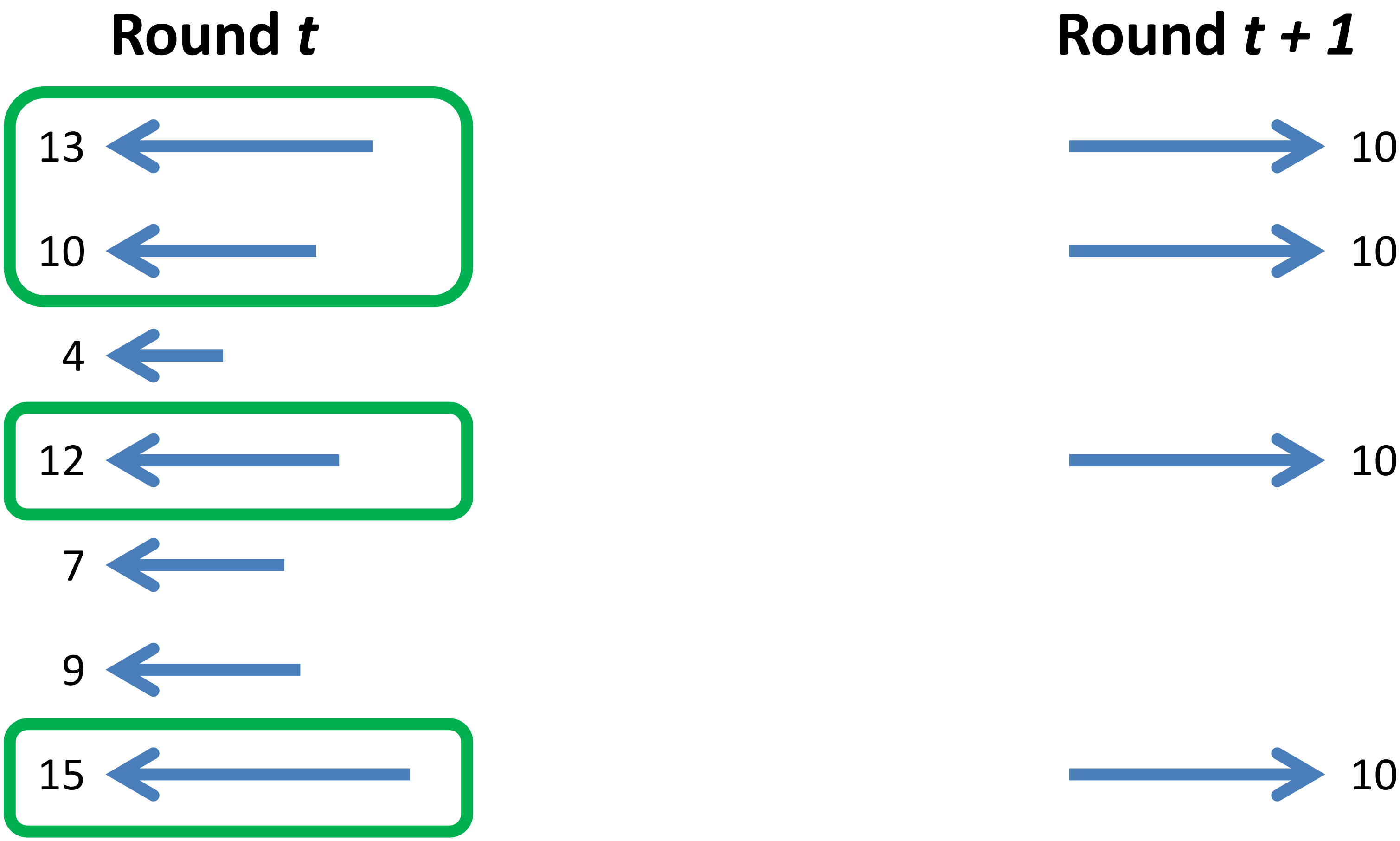
- BitTorrent was enormously successful
  - Large user base
  - Lots of aggregate traffic
  - Invented relatively recently

- Research
  - Modifications to improve performance
  - Modeling peer communications (auctions)
  - Gaming the system (BitTyrant)

Incentives to Upload

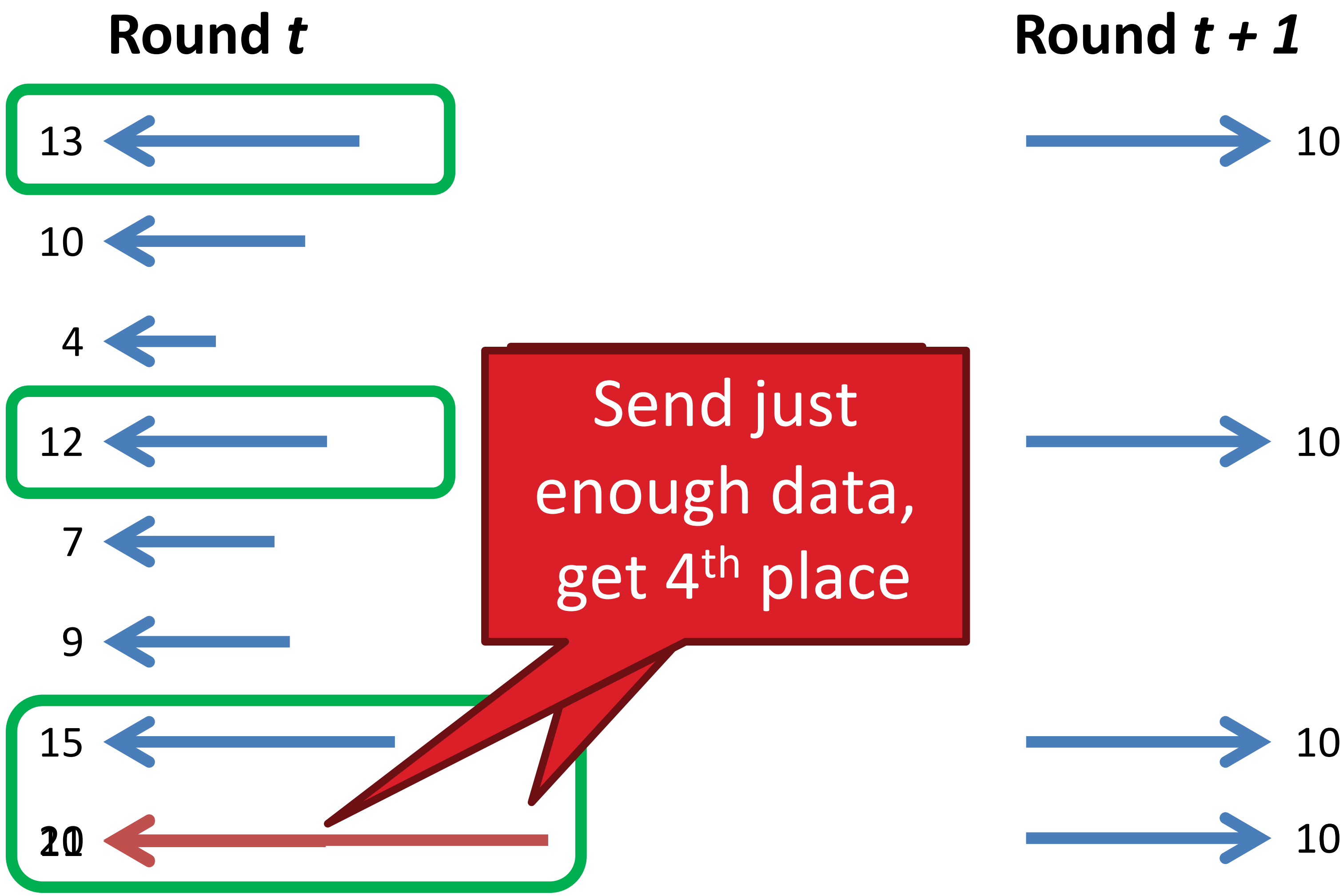
- Every round, a BitTorrent client calculates the number of pieces received from each peer
  - The peers who gave the most will receive pieces in the next round
  - These decisions are made by the unchoker
- Assumption
  - Peers will give as many pieces as possible each round
  - Based on bandwidth constraints, etc.
- Can an attacker abuse this assumption?

Unchoker Example



Abusing the Unchocker

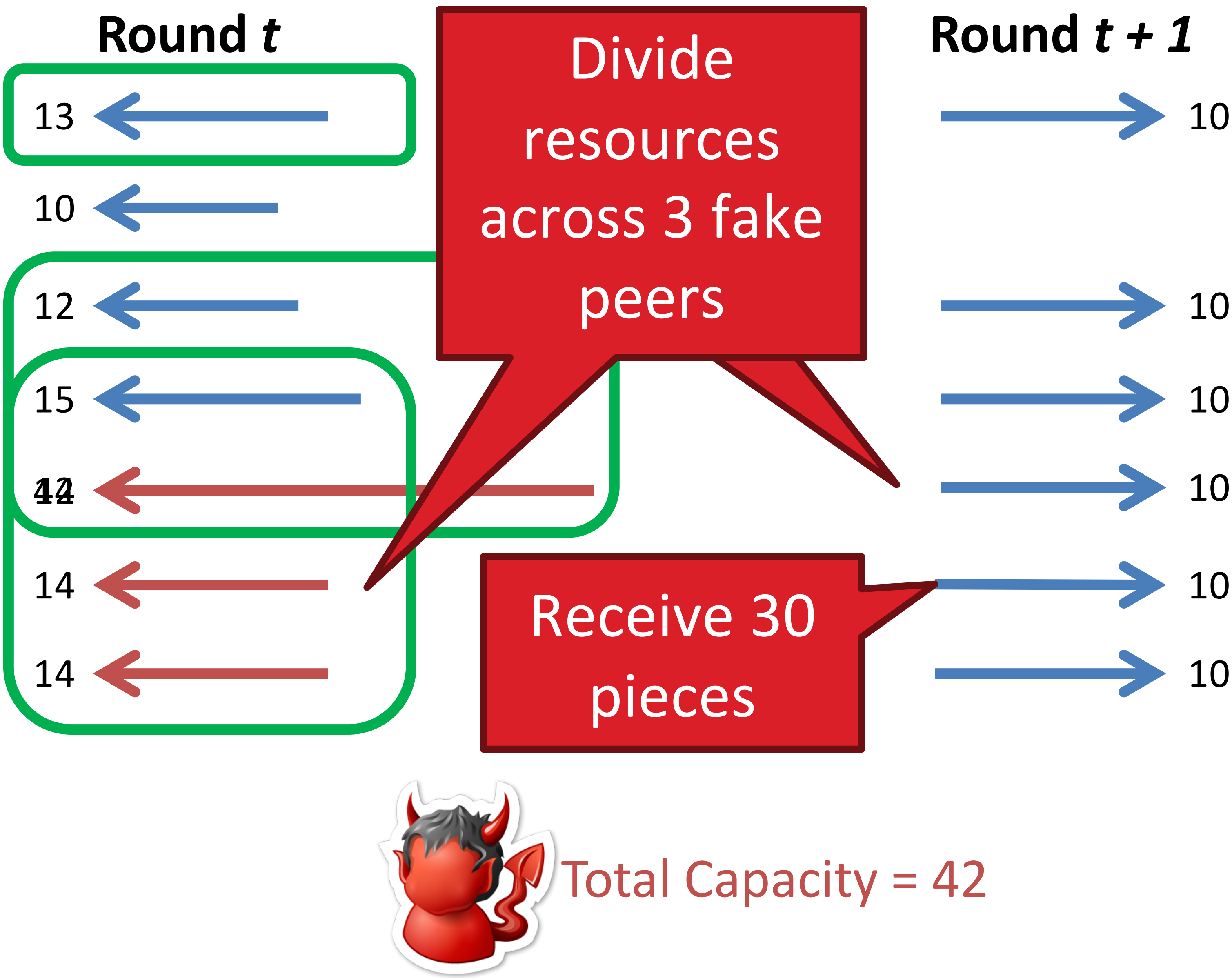
- What if you really want to download from someone?



BitTyrant

- Piatek et al. 2007
  - Implements the “come in last strategy”
  - Essentially, an unfair unchoker
  - Faster than stock BitTorrent (For the Tyrant user!)

Sybil Attack



Summary

- Application Layer: P2P
  - Symmetric responsibility
  - Self-scalability
  - No central authority
- Different flavors:
  - hybrid, hierarchical, completely decentralized
- Incentivize peers using game theory
  - choice of chunk to download
  - tit-for-tat model
  - other optimizations possible