CS 43: Computer Networks

09: Bit-Torrent and P2P

October 1, 2019

SWARTHMORE COLLEGE

Reading Quiz

Last class

- Email!
  - Application Layer Protocol

If SMTP only allows 7-bit ASCII, how do we send pictures/videos/files via email?

A. We encode these objects as 7-bit ASCII

B. We use a different protocol instead of SMTP

C. We're really sending links to the objects, rather than the objects themselves

Base 64

- Designed to be an efficient way to send binary data as a string

- Uses A-Z, a-z,0-9, “+” and “/” as digits

- A number with digits $d_n d_{n-1} \ldots d_1 d_0$ = $64^n * d_n + 64^{n-1} * d_{n-1} + \ldots + 64^* d_1 + d_0$

- Recall from CS 31: Other non-base-10 number systems (binary, octal, hex).

Multipurpose Internet Mail Extensions (MIME)

- Special formatting instructions

- Indicated in the header portion of message (not SMTP)
  - SMTP does *not* care, just looks like message data

- Supports
  - Text in character sets other than ASCII
  - Non-text attachments
  - Message bodies with multiple parts
  - Header information in non-ASCII character sets

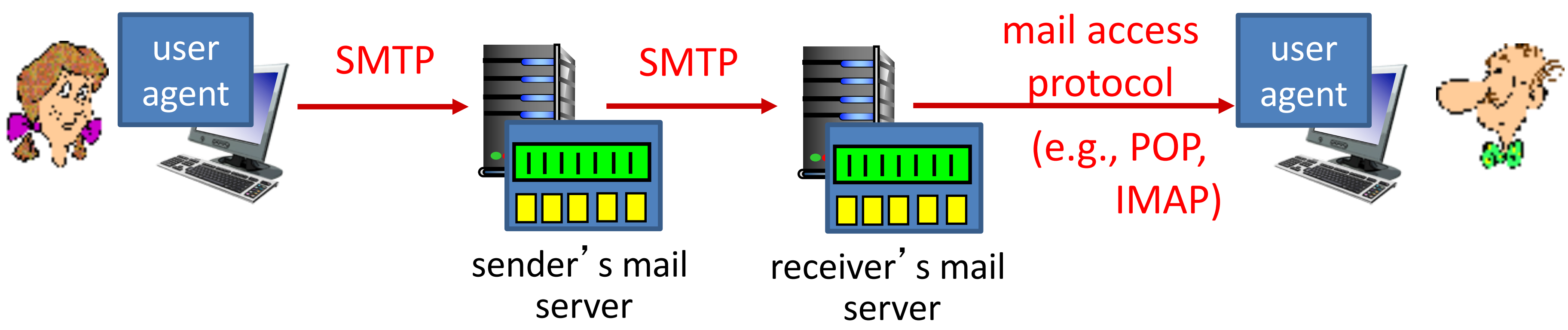
MIME

- Adds optional headers
  - Designed to be compatible with non-MIME email clients
  - Both clients must understand it to make sense of it

- Specifies content type, other necessary information

- Designates a boundary between email text and attachments

Mail access protocols



- SMTP: delivery/storage to receiver's server
- mail access protocol: retrieval from server
  - POP: Post Office Protocol: authorization, download
  - IMAP: Internet Mail Access Protocol: more features, including manipulation of stored messages on server
  - HTTP: gmail, Hotmail, Yahoo! Mail, etc.

POP3 protocol

authorization phase

- client commands:
  - **user:** declare username
  - **pass:** password
- server responses
  - **+OK**
  - **-ERR**

transaction phase, client:

- **list:** list message numbers
- **retr:** retrieve message by number
- **dele:** delete
- **quit**

```
S: +OK POP3 server ready
C: user bob
S: +OK
C: pass hungry
S: +OK user successfully logged on
```

```
C: list
S: 1 498
S: 2 912
S: .
C: retr 1
S: <message 1 contents>
S: .
C: dele 1
C: retr 2
S: <message 1 contents>
S: .
C: dele 2
C: quit
S: +OK POP3 server signing off
```

More about POP3

- Previous example uses "download and delete" mode
  - Bob cannot re-read e-mail if he changes client
- POP3 "download-and-keep": copies of messages on different clients
- POP3 is stateless across sessions
- Limitations:
  - Can't retrieve just the headers
  - Can't impose structure on messages

IMAP

- Keeps all messages in one place: at server
- Allows user to organize messages in folders
- Keeps user state across sessions:
  - names of folders and mappings between message IDs and folder name
- Can request pieces of a message (e.g., text parts without large attachments)

Webmail

- Uses a web browser
- Sends emails using HTTP rather than POP3 or IMAP
- Mail is stored on the 3rd party webmail company's servers

Summary

- Three main parts to email:
  - Mail User Agent (mail client): read / write for humans
  - Mail Transfer Agent: server that accepts / sends messages
  - SMTP protocol used to negotiate transfers
- No SMTP support for fraud detection
- Extensions (MIME) and encodings (Base64) for sending non-text data

Today

- P2P applications
- BitTorrent
  - Cooperative file transfers
- Briefly: Distributed Hash Tables
  - Finding things without central authority

Where we are

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

Transport: end-to-end connections, reliability

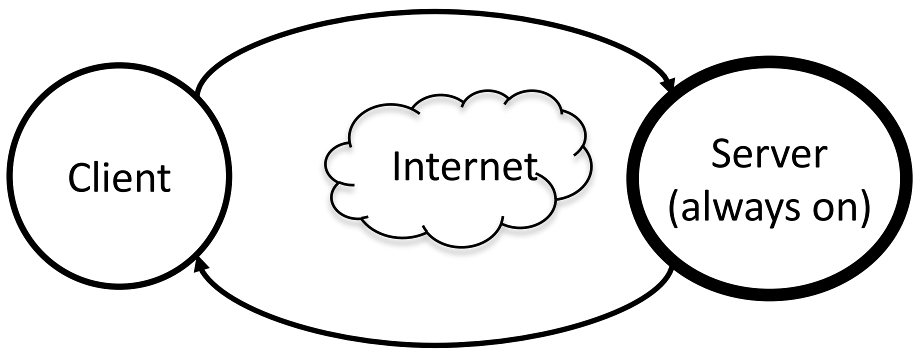
Network: routing

Link (data-link): framing, error detection

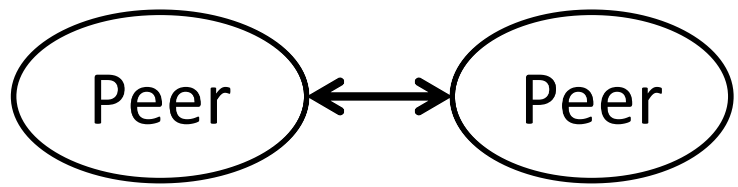
Physical: 1's and 0's/bits across a medium
(copper, the air, fiber)

Designating roles to an endpoint

Client-server architecture



Peer-to-peer architecture



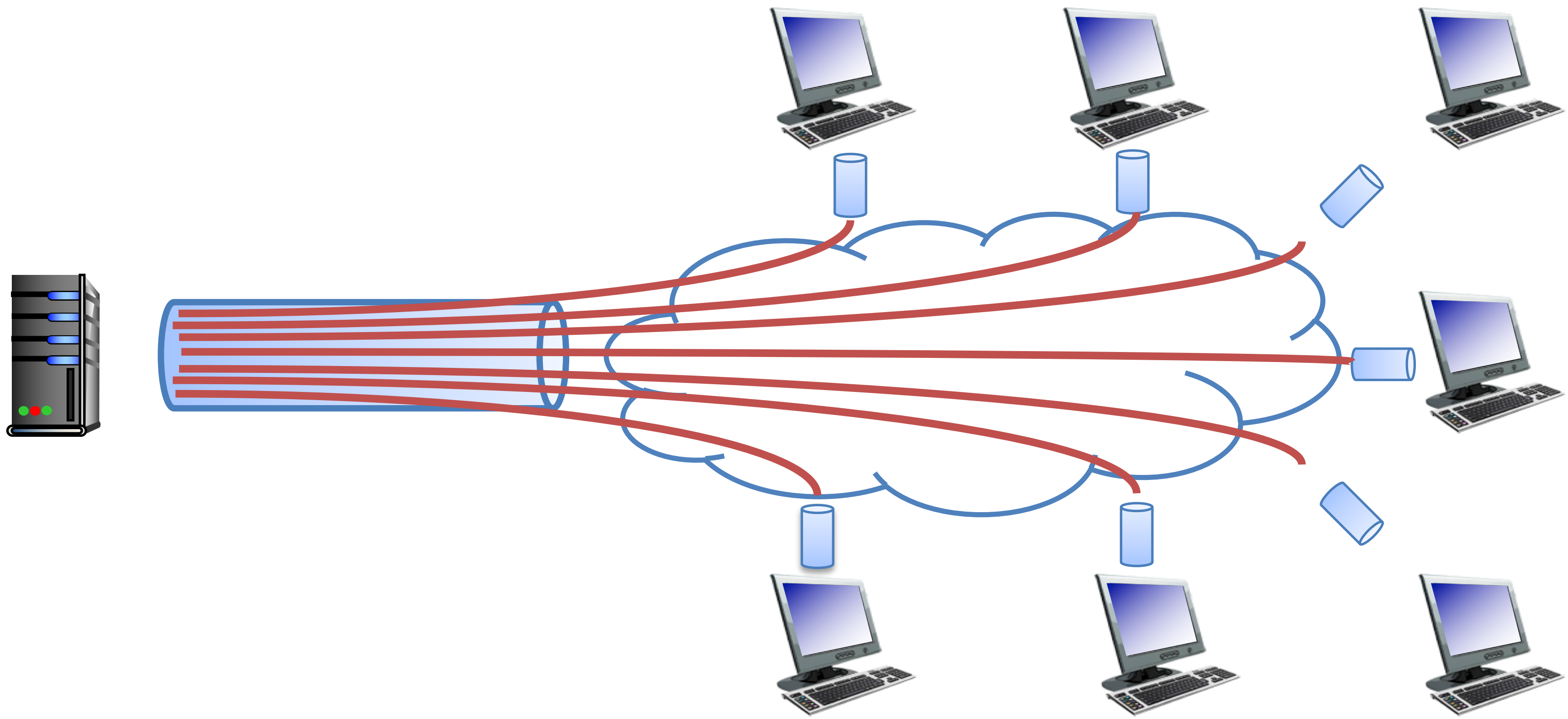
File Transfer Problem

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

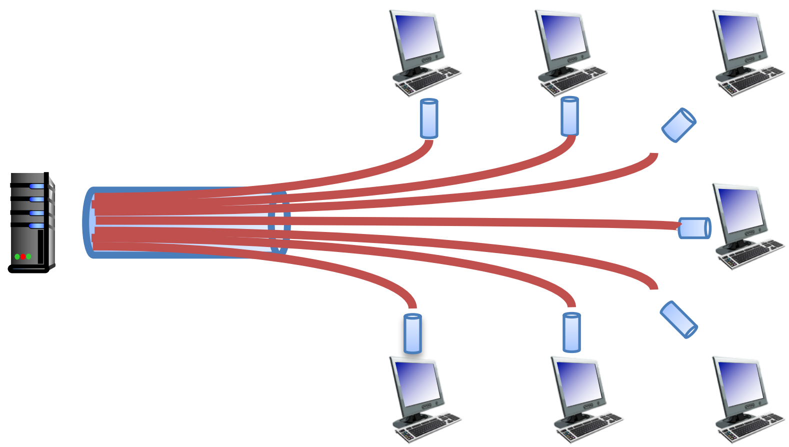
Traditional Client/Server

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

Traditional Client/Server

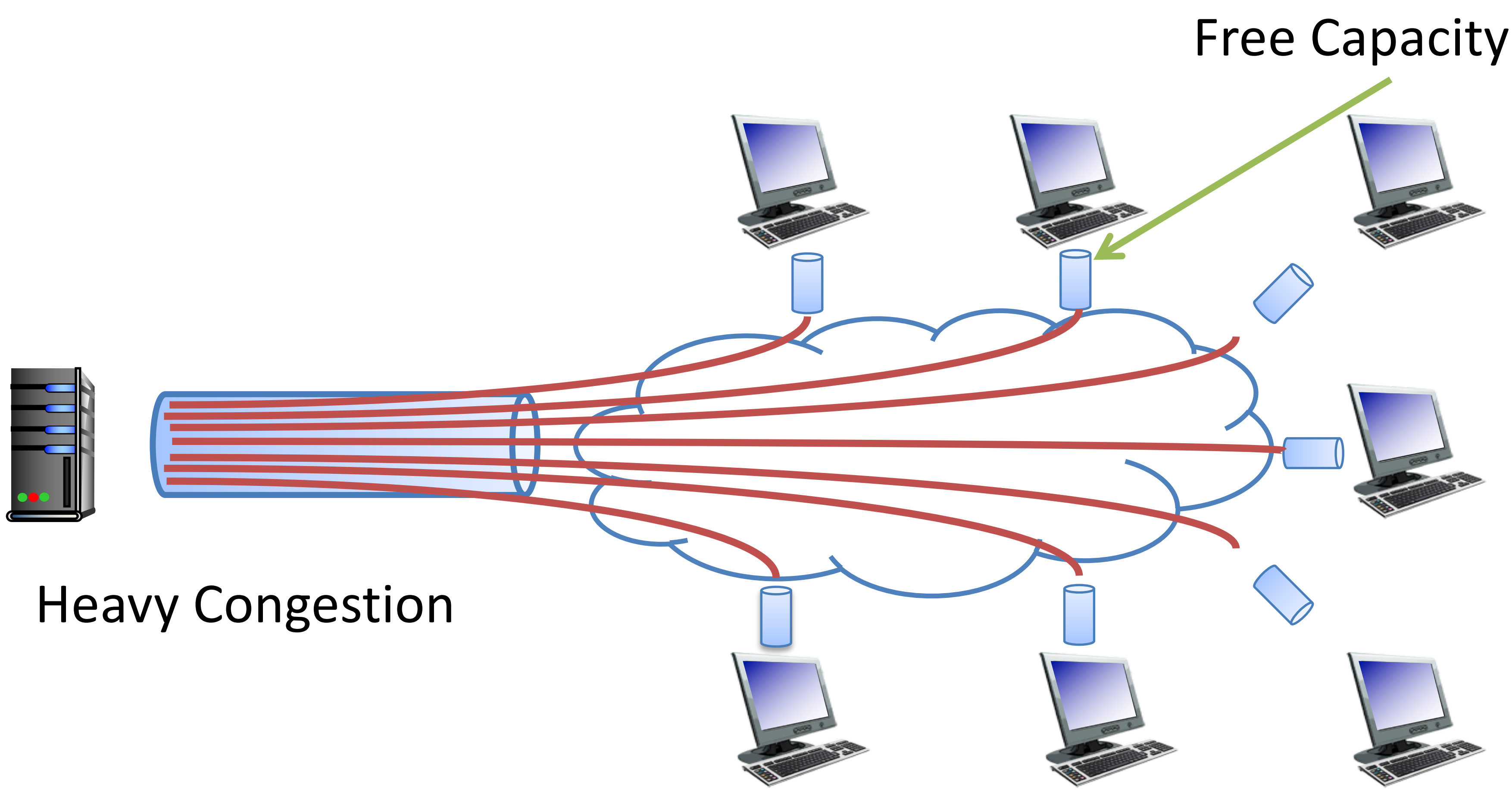


What is the biggest problem you run into with the traditional C/S model?

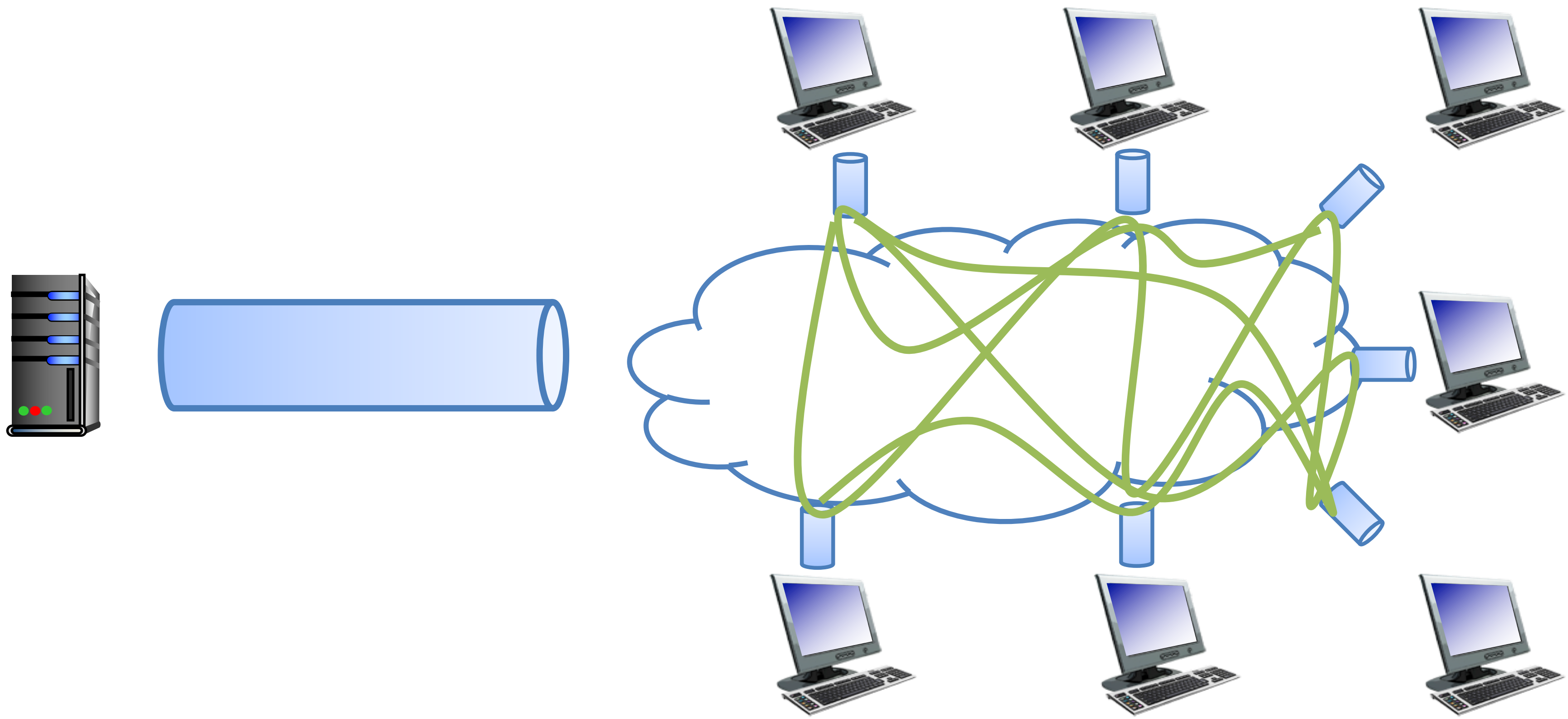


A. Scalability (how many end-hosts can you support?)
B. Reliability (what happens on failure?)
C. Efficiency (fast response time)

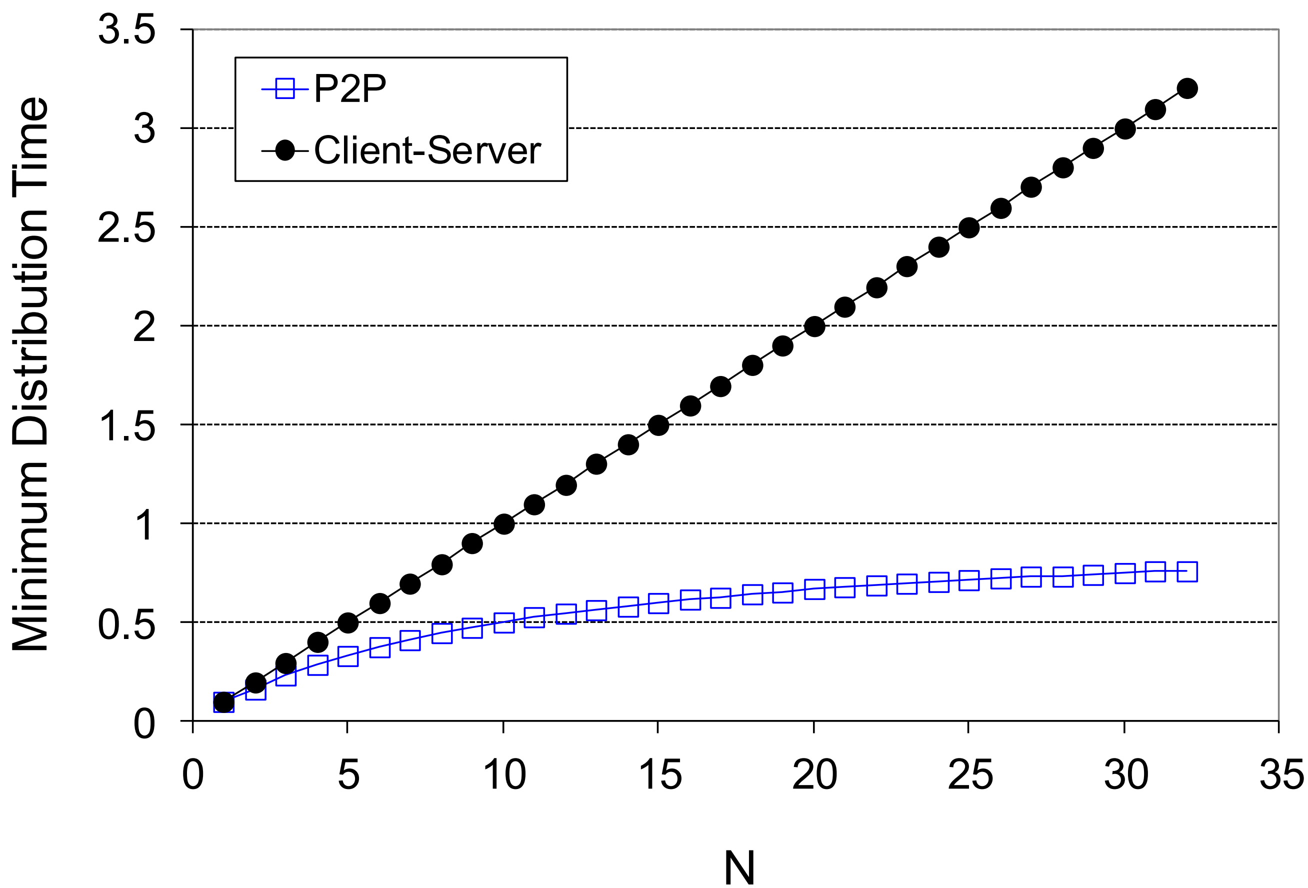
Traditional Client/Server



P2P Solution



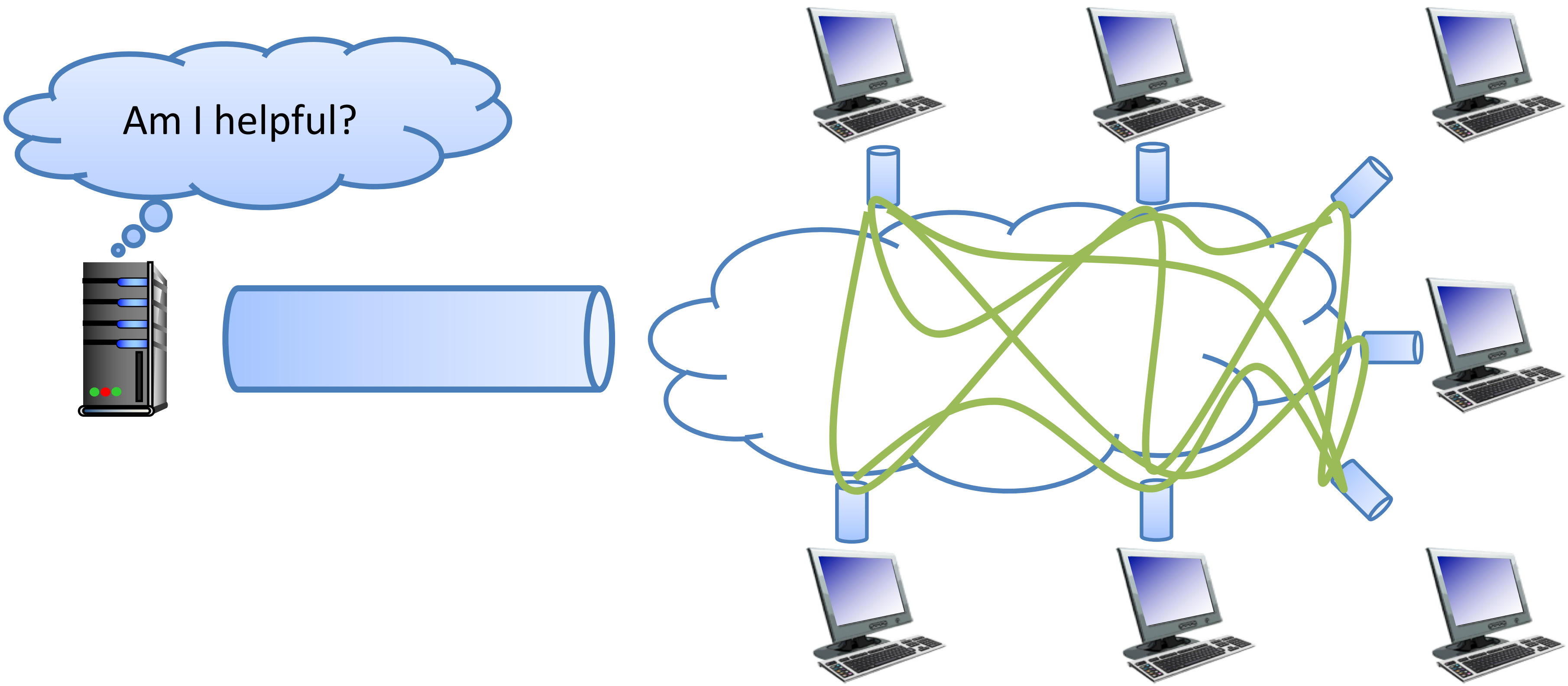
Client-server vs. P2P: example



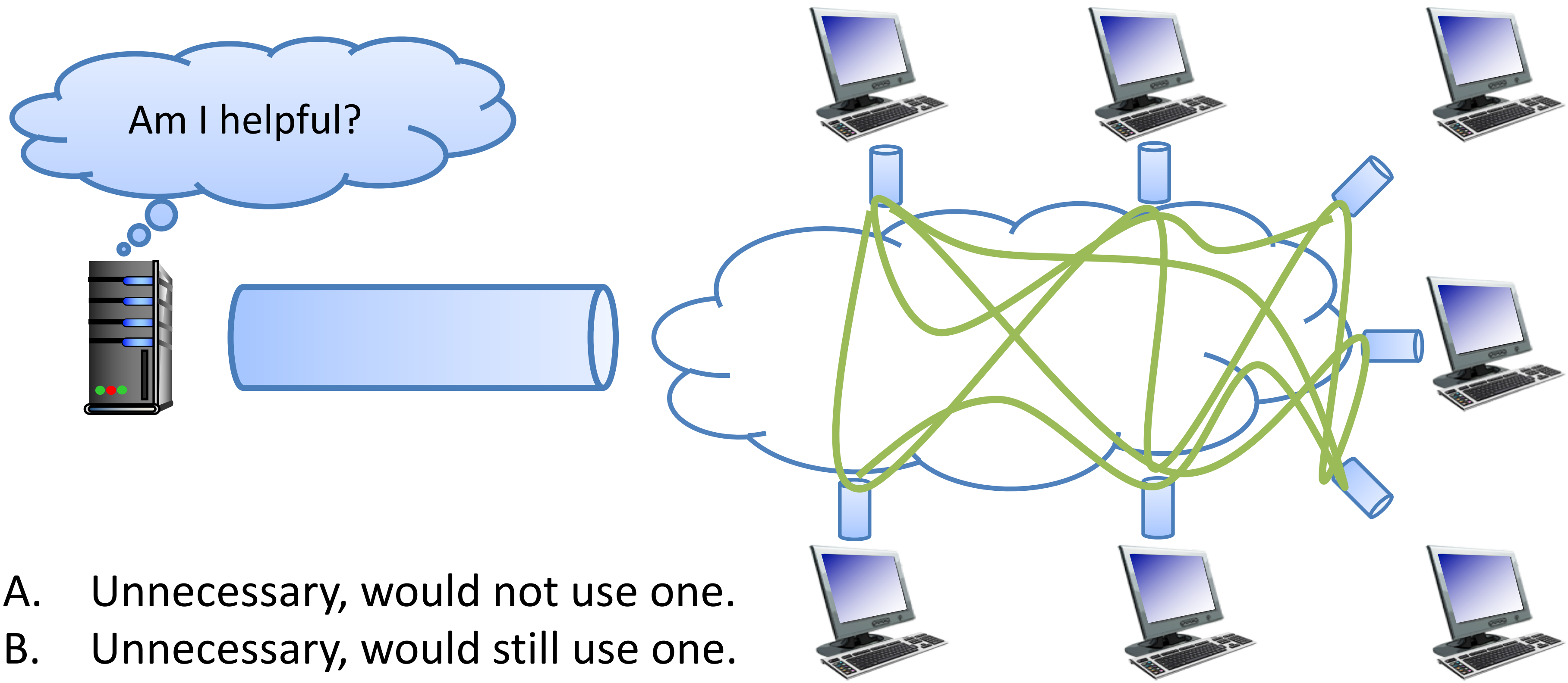
Let F = file size, client UL rate = u, server rate = u_s, d = client DL rate

Assumptions: F/u = 1 hour, $u_s = 10u$, $d_{min} \geq u_s$

P2P Solution

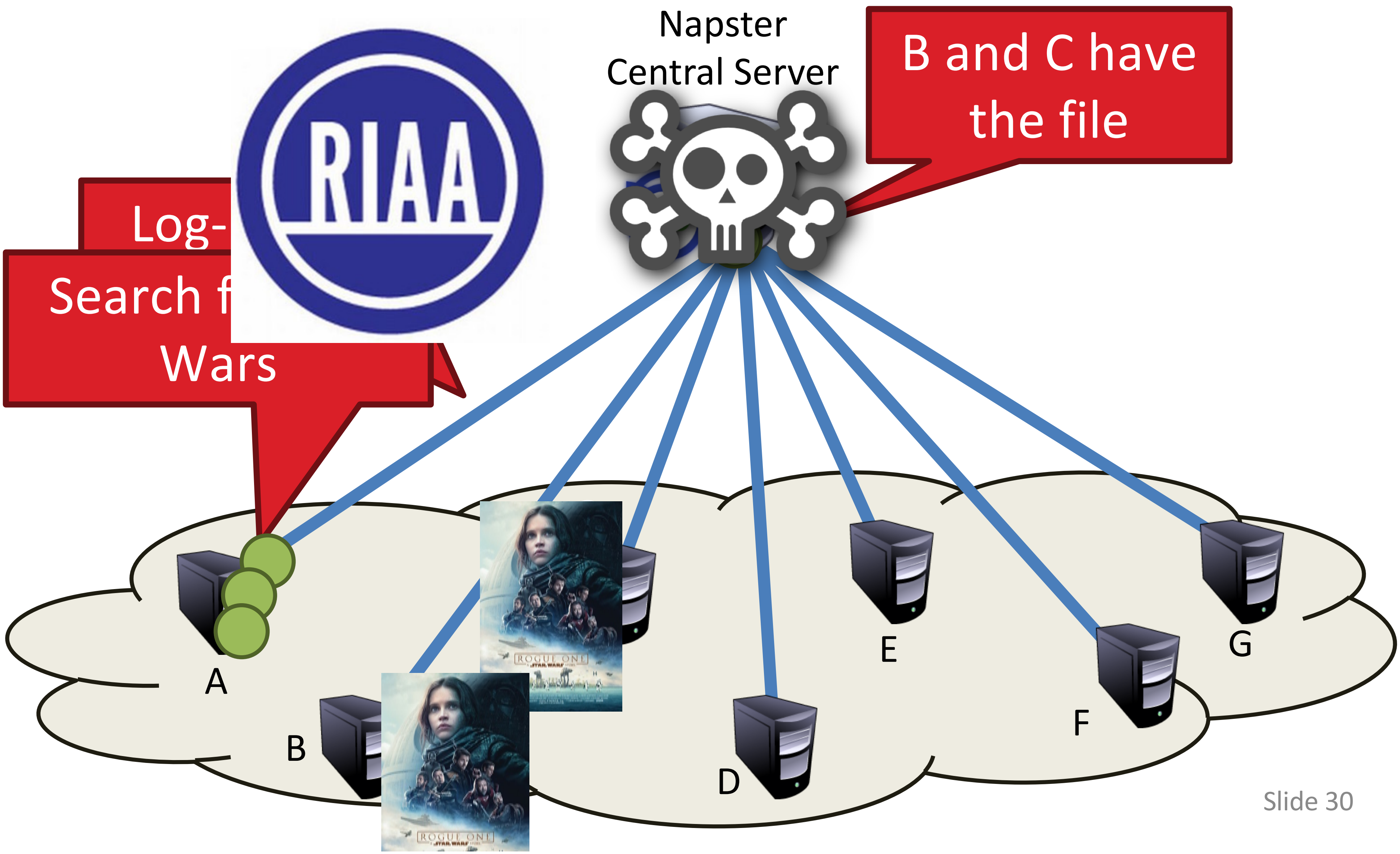


Do we need a centralized server at all? Would you use one for something?

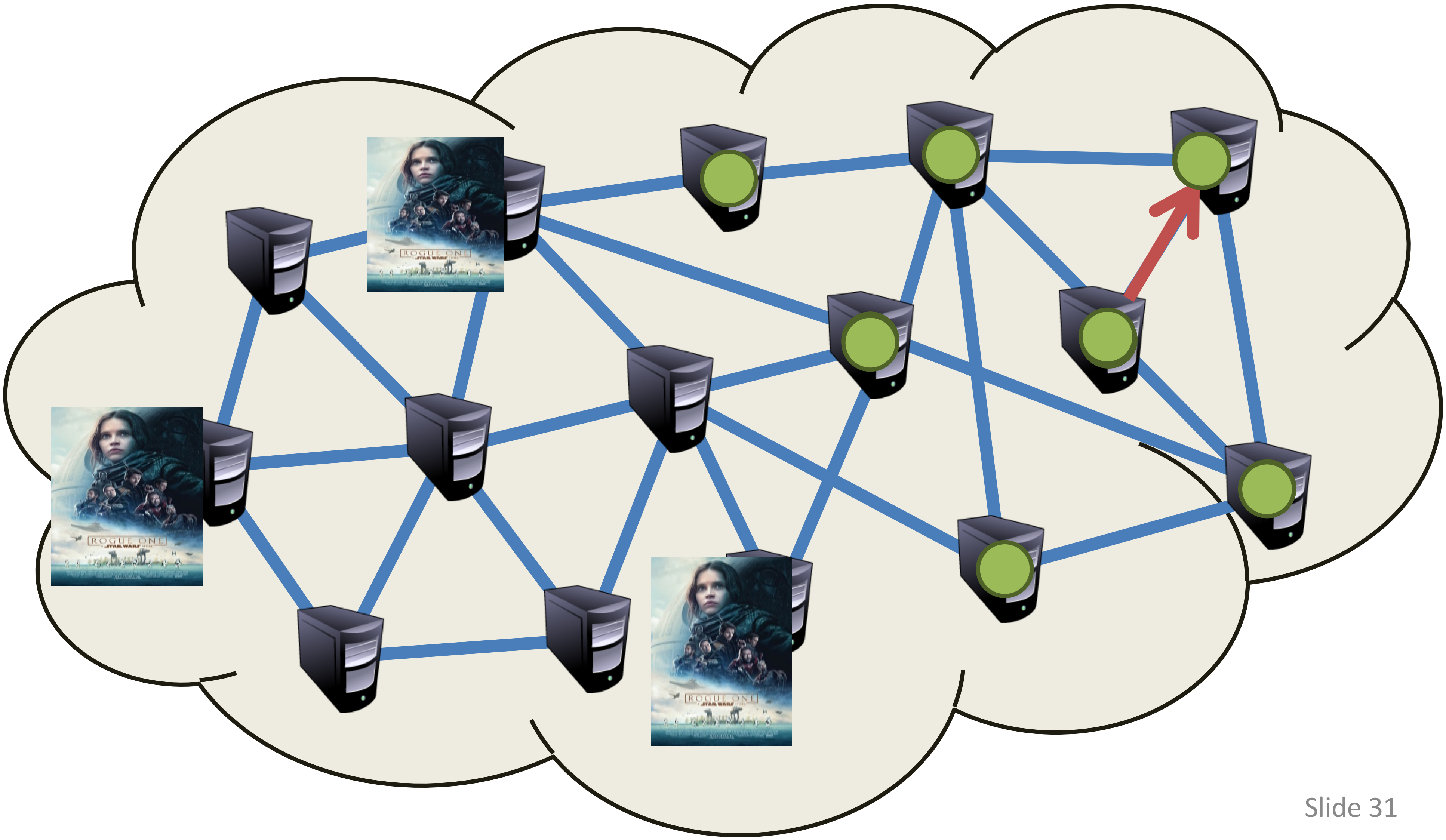


A. Unnecessary, would not use one.
B. Unnecessary, would still use one.
C. Necessary, would have to use it.
D. Something else.

Napster Architecture



File Search via Flooding in Gnutella

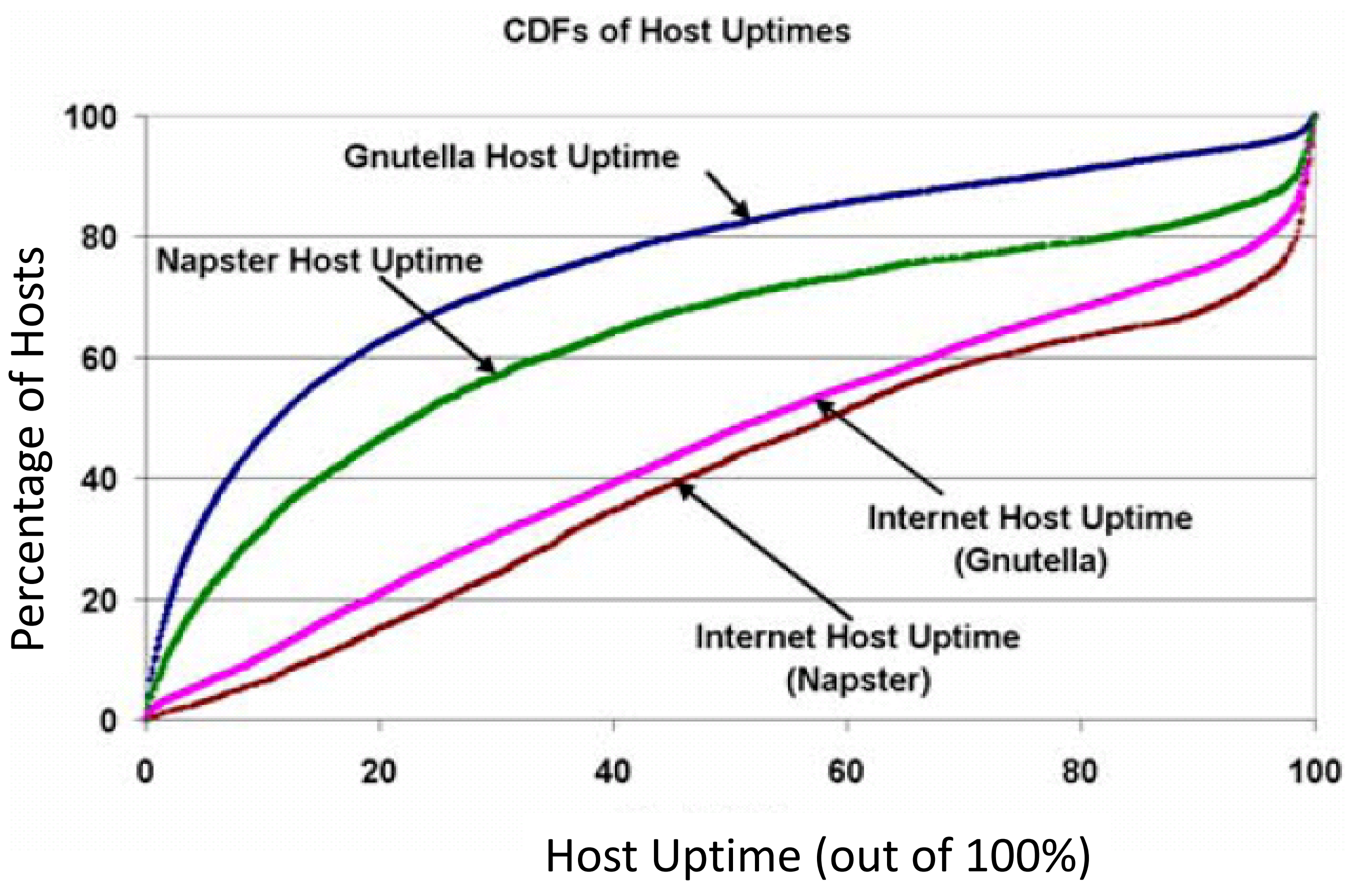


What are the cons of the Gnutella approach?

A. Access to rare files
B. Traffic overhead
C. Redundancy
D. Something else.

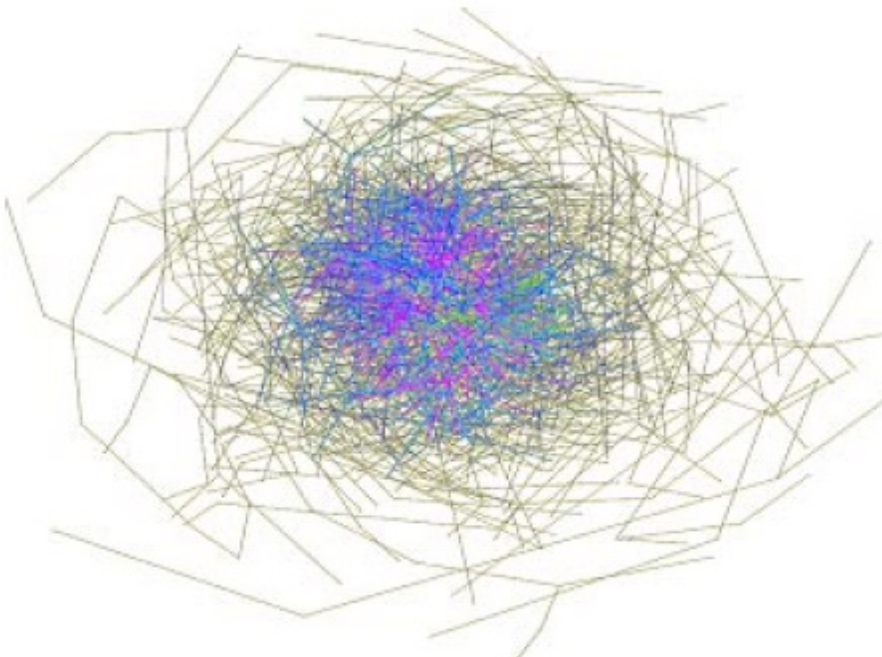
Peer Lifetimes: Highly available?

- Study of host uptime and application uptime (MMCN 2002)
  - Sessions are short (median is 60 minutes)
  - Hosts are frequently offline

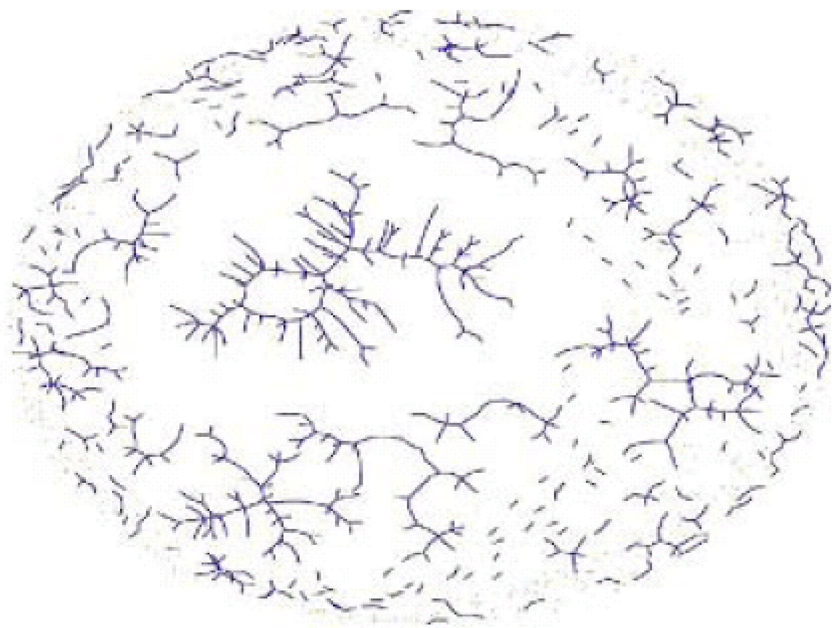


Resilience to Failures and Attacks

- 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



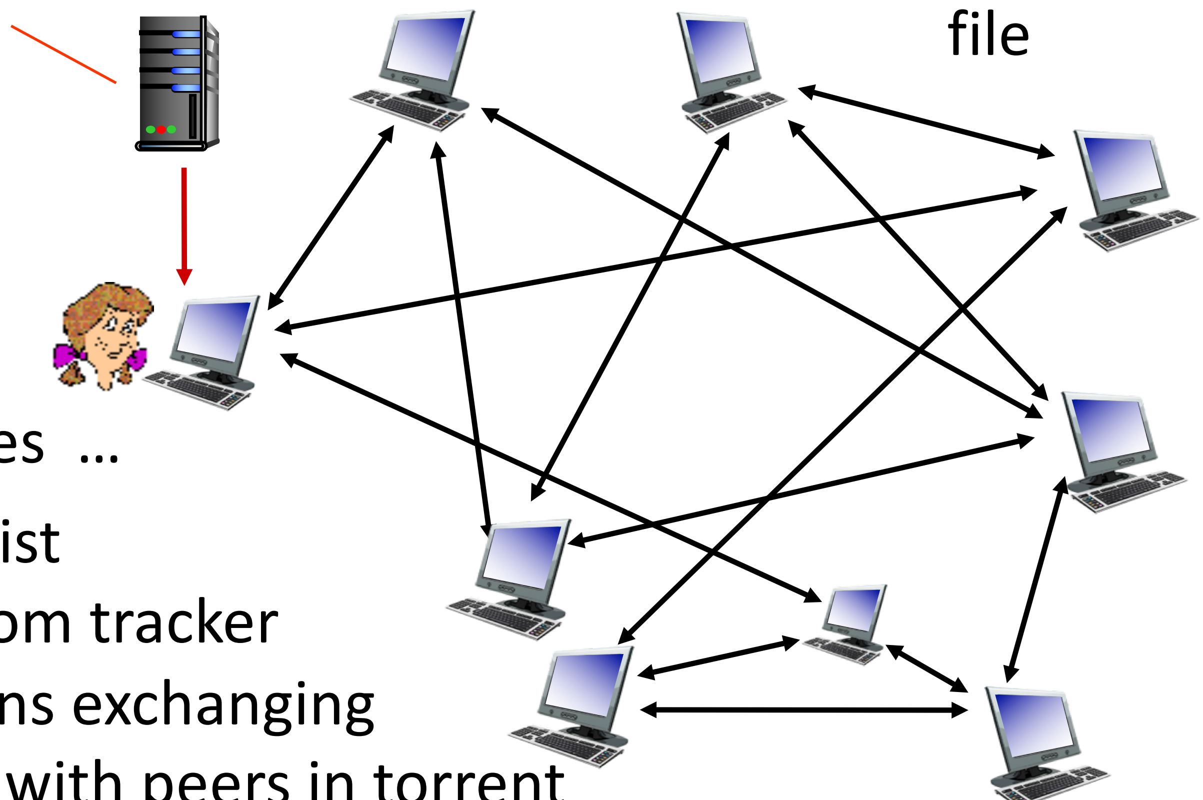
After top 4% of peers are removed
After random 30% of peers removed

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

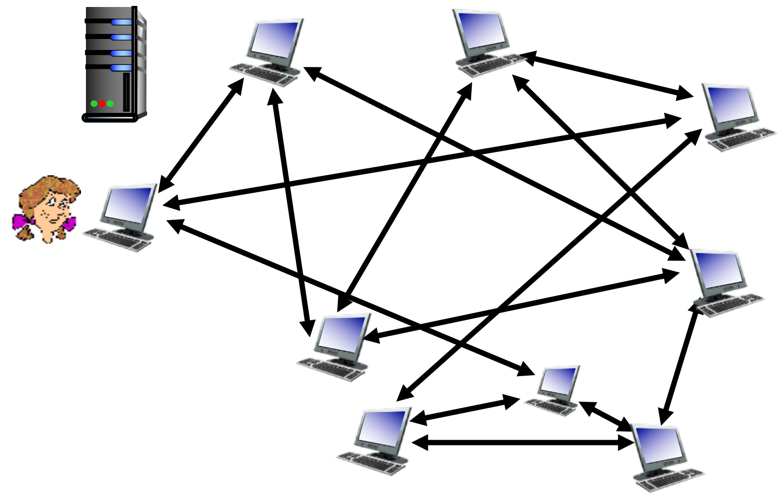
… and begins exchanging file chunks with peers in torrent

.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

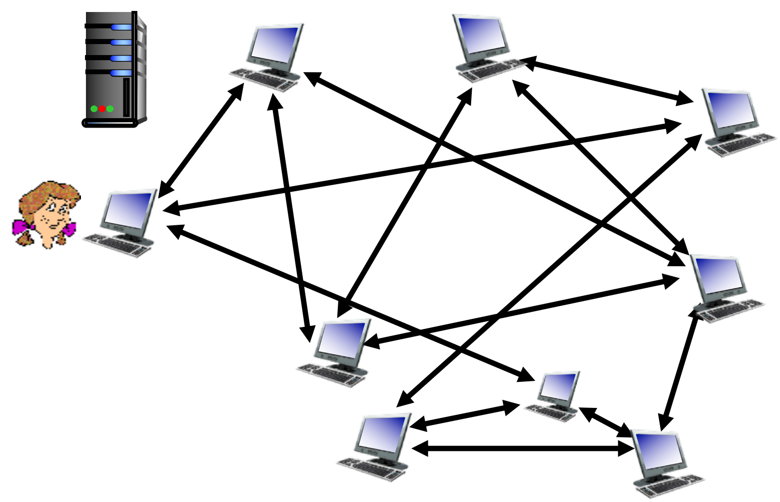
- has no chunks, but will accumulate them over time from other peers
- registers with tracker to get list of peers, connects to subset of peers ("neighbors")



P2P file distribution: BitTorrent

- While downloading, peer uploads chunks to other peers

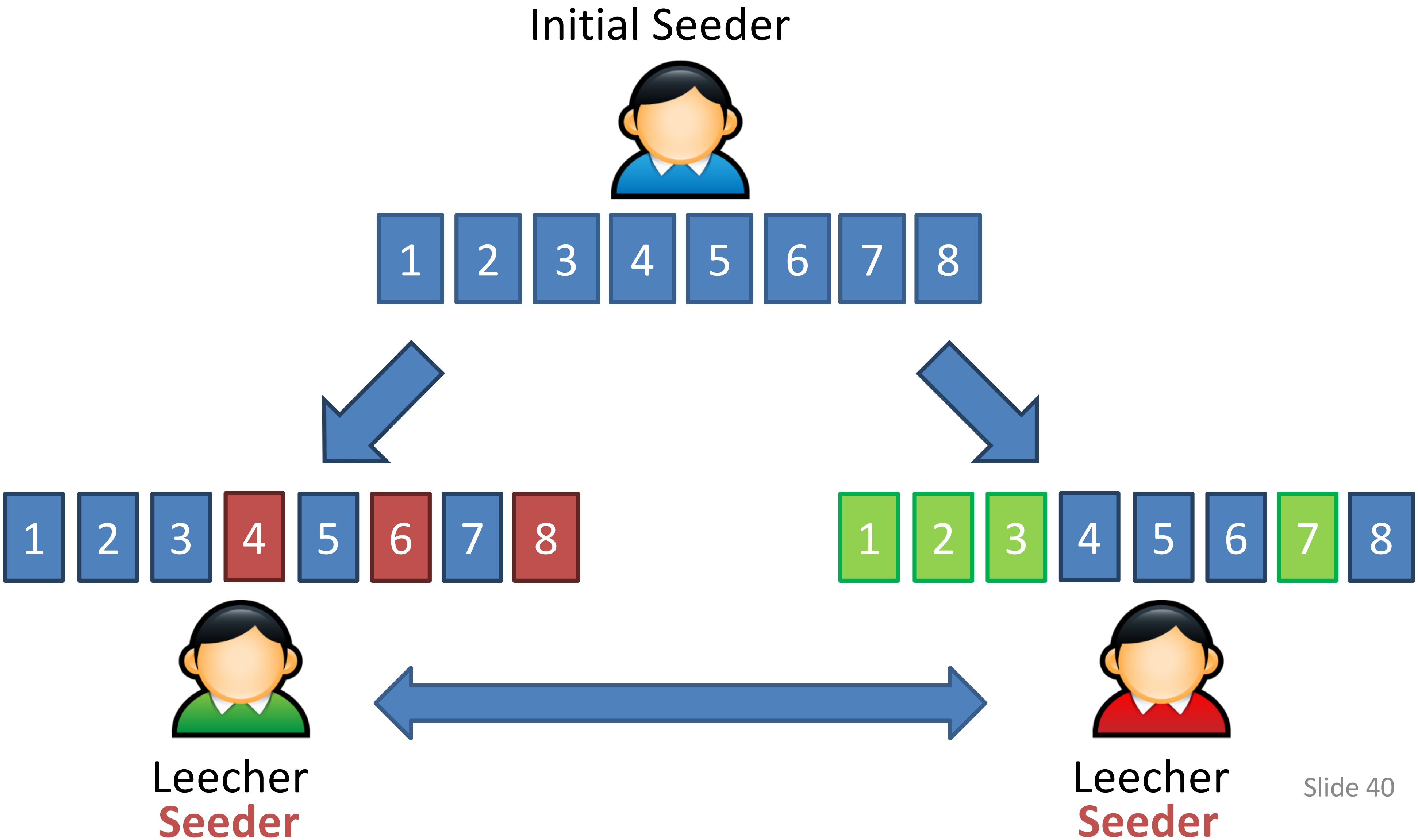
- *Churn:* peers may come and go
  - Peer may change peers with whom it exchanges chunks



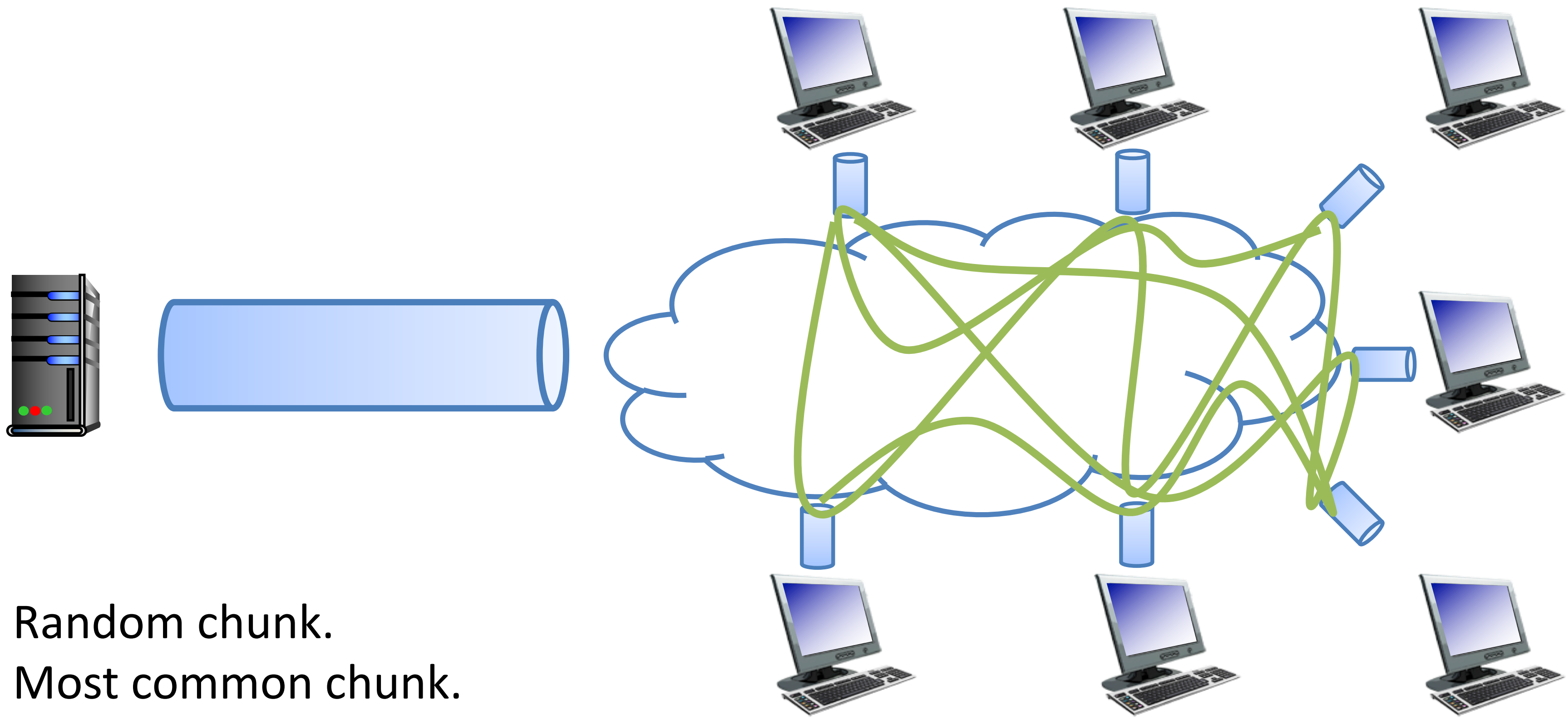
Requesting Chunks

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

Sharing Pieces

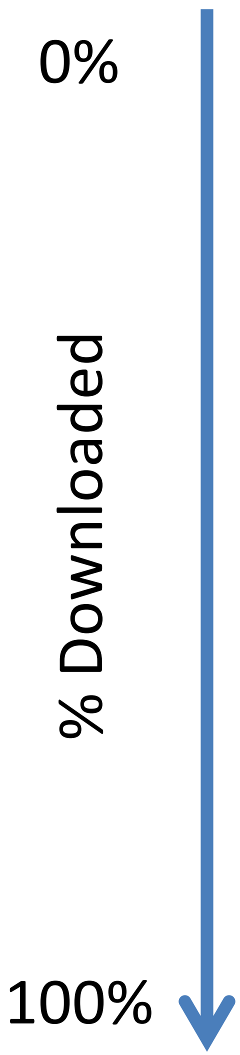


If you're trying to receive a file, which chunk should you request next?



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

Requesting Chunks



- 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

- 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

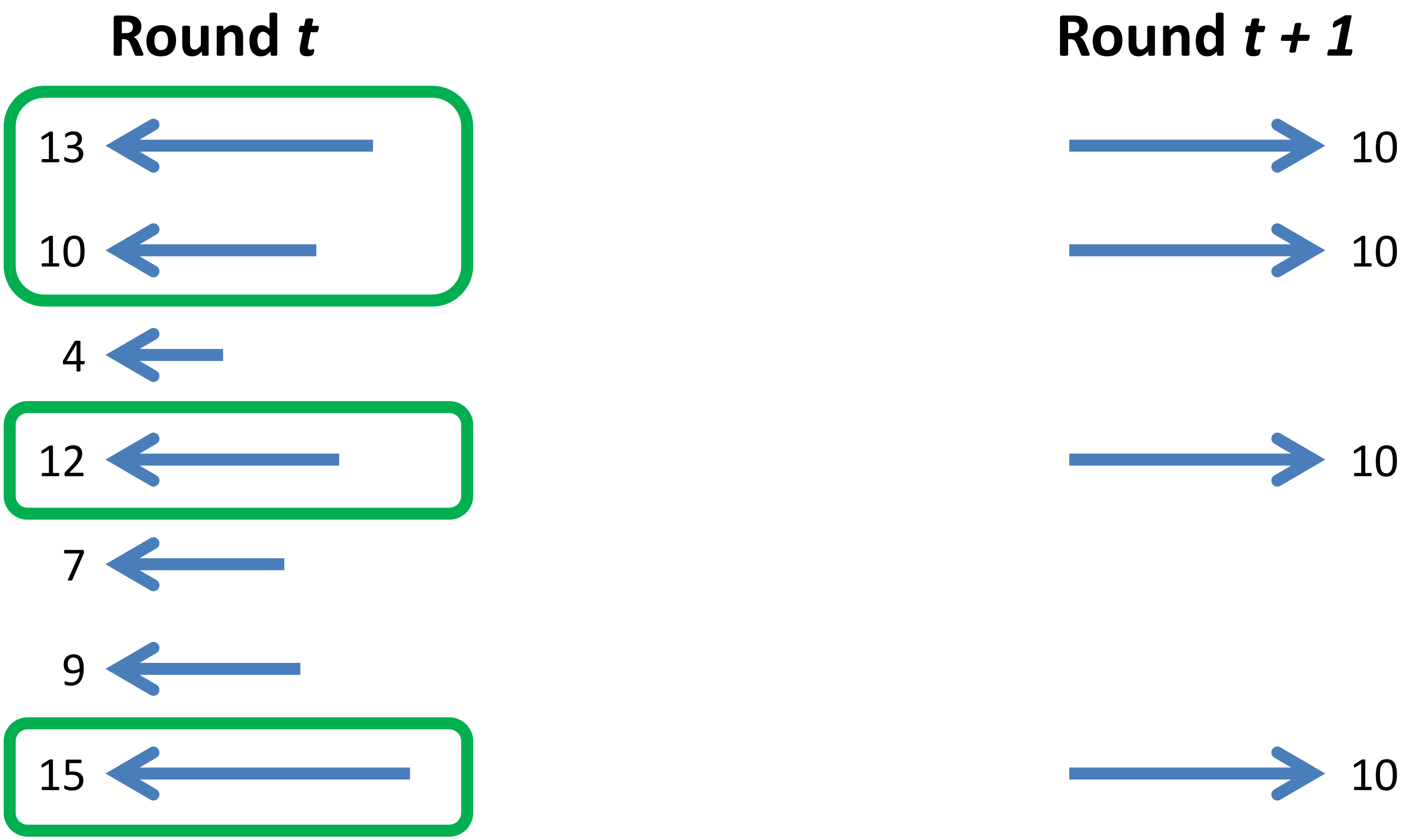
- BitTorrent was enormously successful
  - Large user base
  - Lots of aggregate traffic
  - Invented relatively recently

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

Incentives to Upload

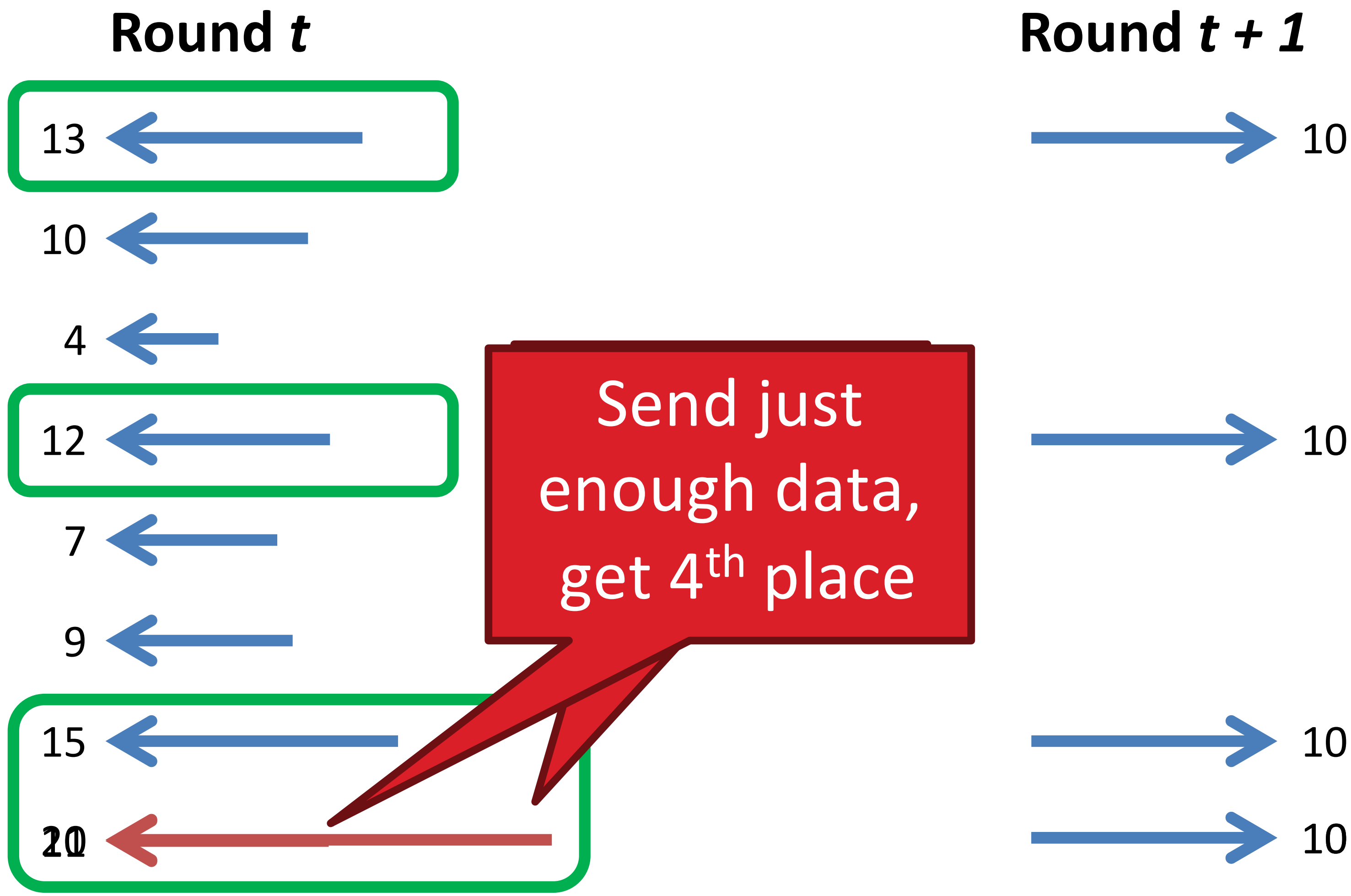
- Every round, a BitTorrent client calculates the number of pieces received from each peer
  - The peers who gave the most will receive pieces in the next round
  - These decisions are made by the unchoker
- Assumption
  - Peers will give as many pieces as possible each round
  - Based on bandwidth constraints,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

Unchoker Example

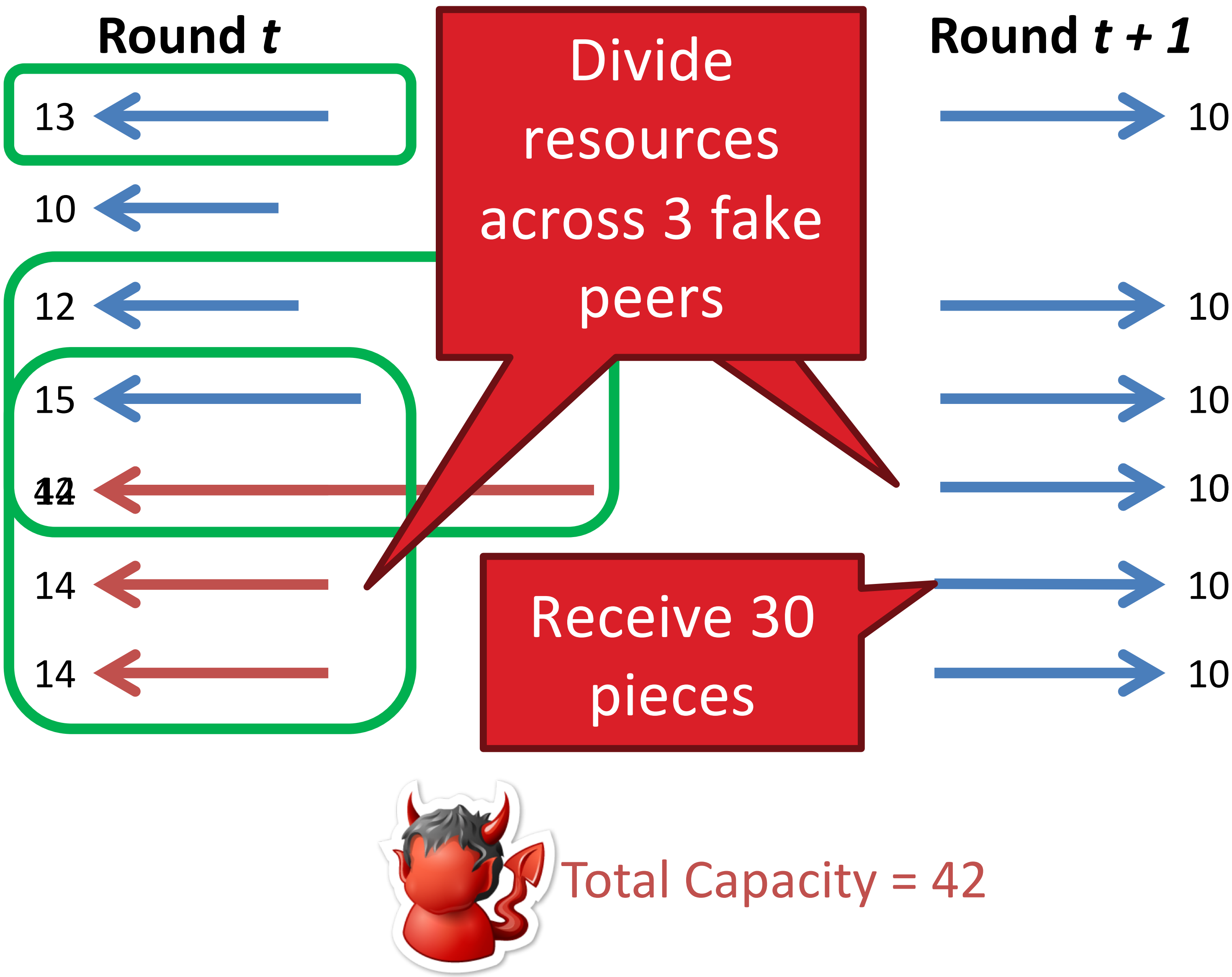


Abusing the Unchocker

- What if you really want to download from someone?



Sybil Attack

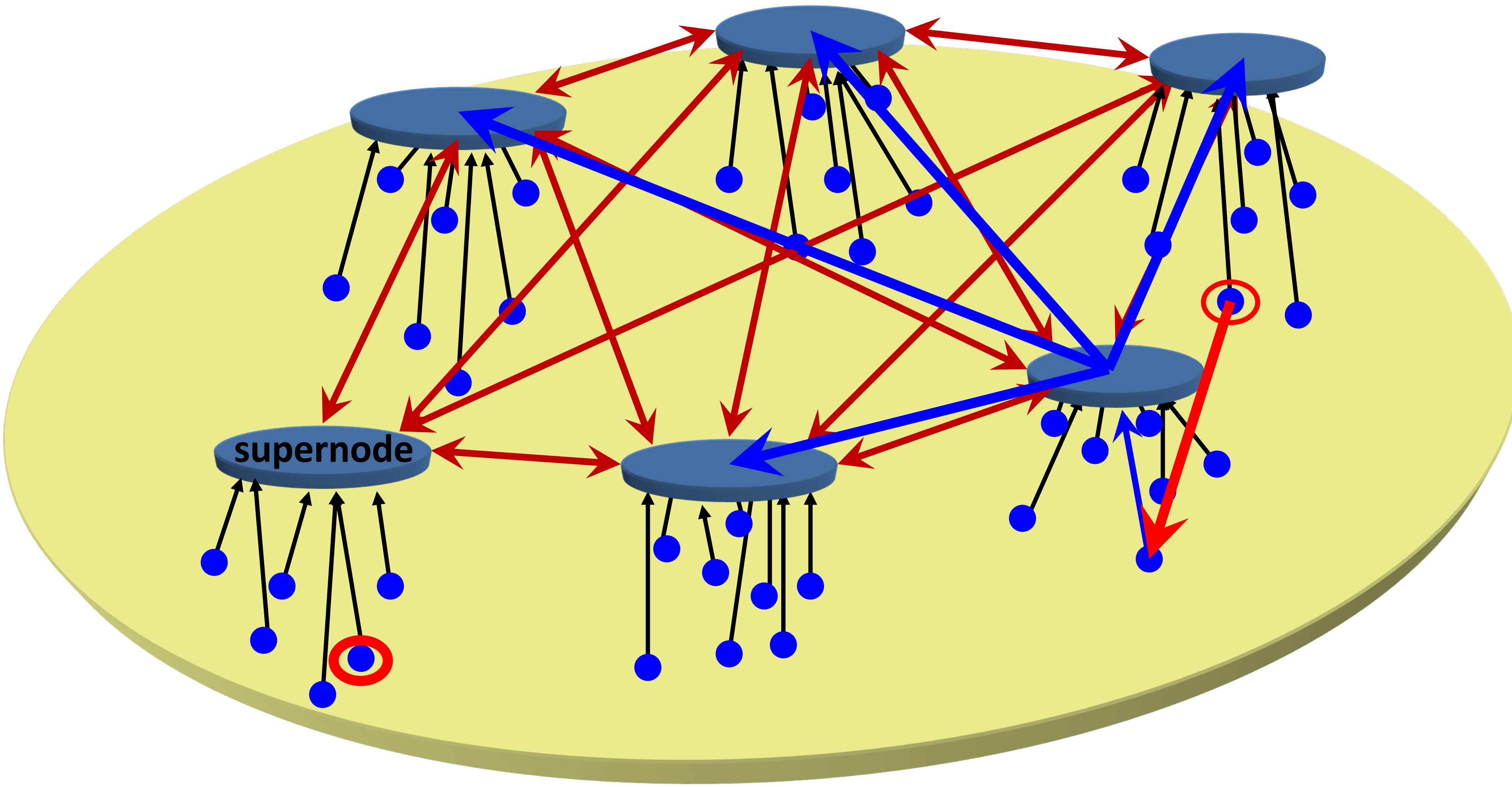


BitTyrant

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

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

Do the benefits of hierarchical P2P networks out-weight the cons?

A. Pros: Scalability
B. Pros: Limits flooding
C. Cons: No guarantees of performance
D. Cons: Failure?