CS 3700
Networks and Distributed Systems

Lecture 18: Peer-to-Peer Systems

Revised 3/23/13

Outline

- Peer-to-Peer Overview
- Example: Bittorrent
  - µTP: Micro Transport Protocol
  - Cheating on BitTorrent

Traditional Internet Services Model

- Client-server
  - Many clients, 1 (or more) server(s)
  - Web servers, DNS, file downloads, video streaming
- Problems
  - Scalability: how many users can a server support?
    - What happens when user traffic overload servers?
    - Limited resources (bandwidth, CPU, storage)
  - Reliability: if # of servers is small, what happens when they break, fail, get disconnected, are mismanaged by humans?
  - Efficiency: if your users are spread across the entire globe, how do you make sure you answer their requests quickly?

The Alternative: Peer-to-Peer

- A simple idea
  - Users bring their own resources to the table
  - A cooperative model: clients = peers = servers
- The benefits
  - Scalability: # of "servers" grows with users
    - BYOR: bring your own resources (storage, CPU, B/W)
  - Reliability: load spread across many peers
    - Probability of them all failing is **very** low…
  - Efficiency: peers are distributed
    - Peers can try and get service from nearby peers

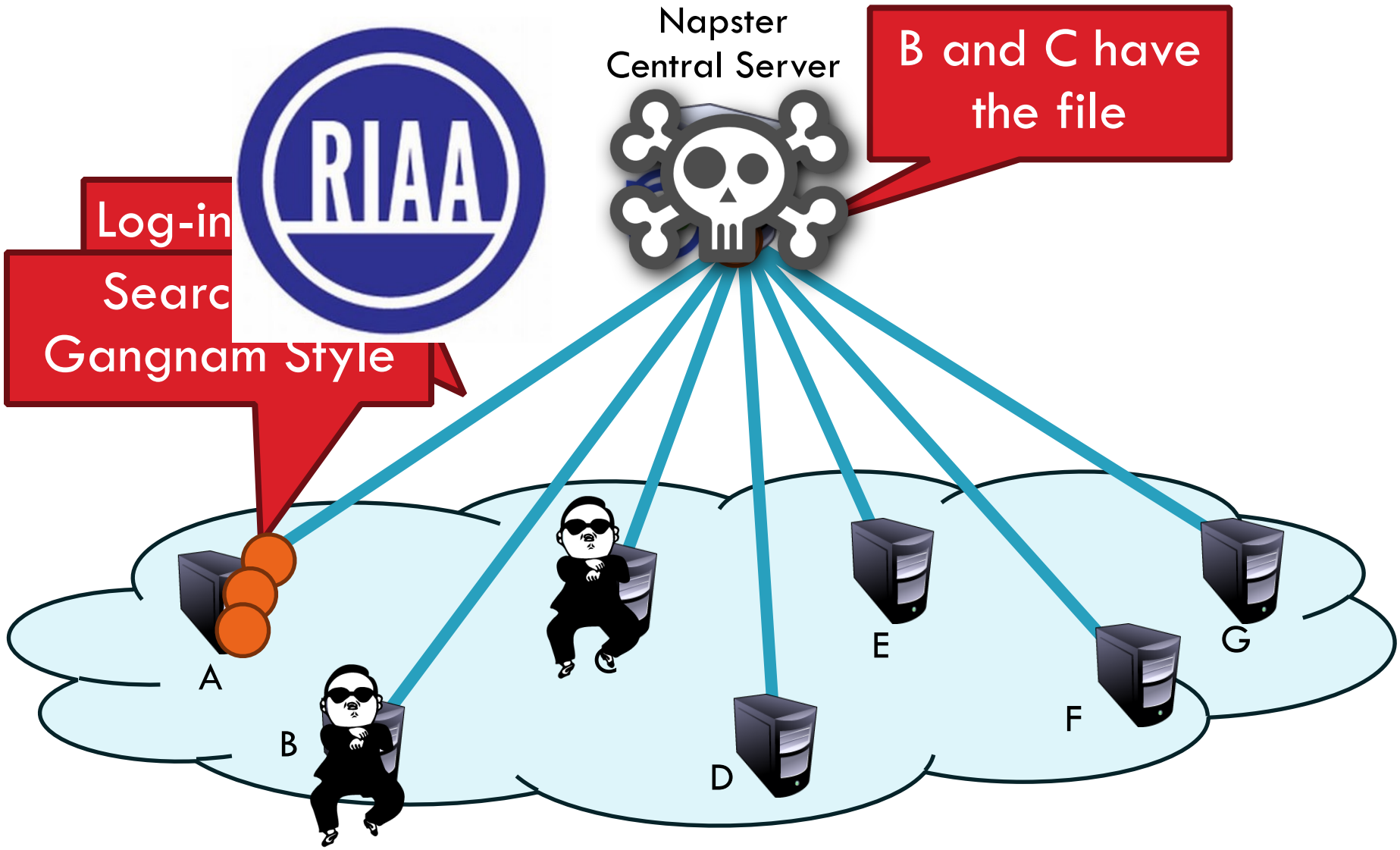
The Peer-to-Peer Challenge

- What are the key components for leveraging P2P?
  - Communication: how do peers talk to each other
  - Service/data location: how do peers know who to talk to
- New reliability challenges
  - Network reachability, i.e. dealing with NATs
  - Dealing with churn, i.e. short peer uptimes
- What about security?
  - Malicious peers and cheating
  - The Sybil attack

Centralized Approach

- The original: Napster
  - 1999-2001
  - Shawn Fanning, Sean Parker
  - Invented at NEU
  - Specialized in MP3s (but not for long)
- Centralized index server(s)
  - Supported all queries
- What caused its downfall?
  - Not scalable
  - Centralization of liability

Napster Architecture

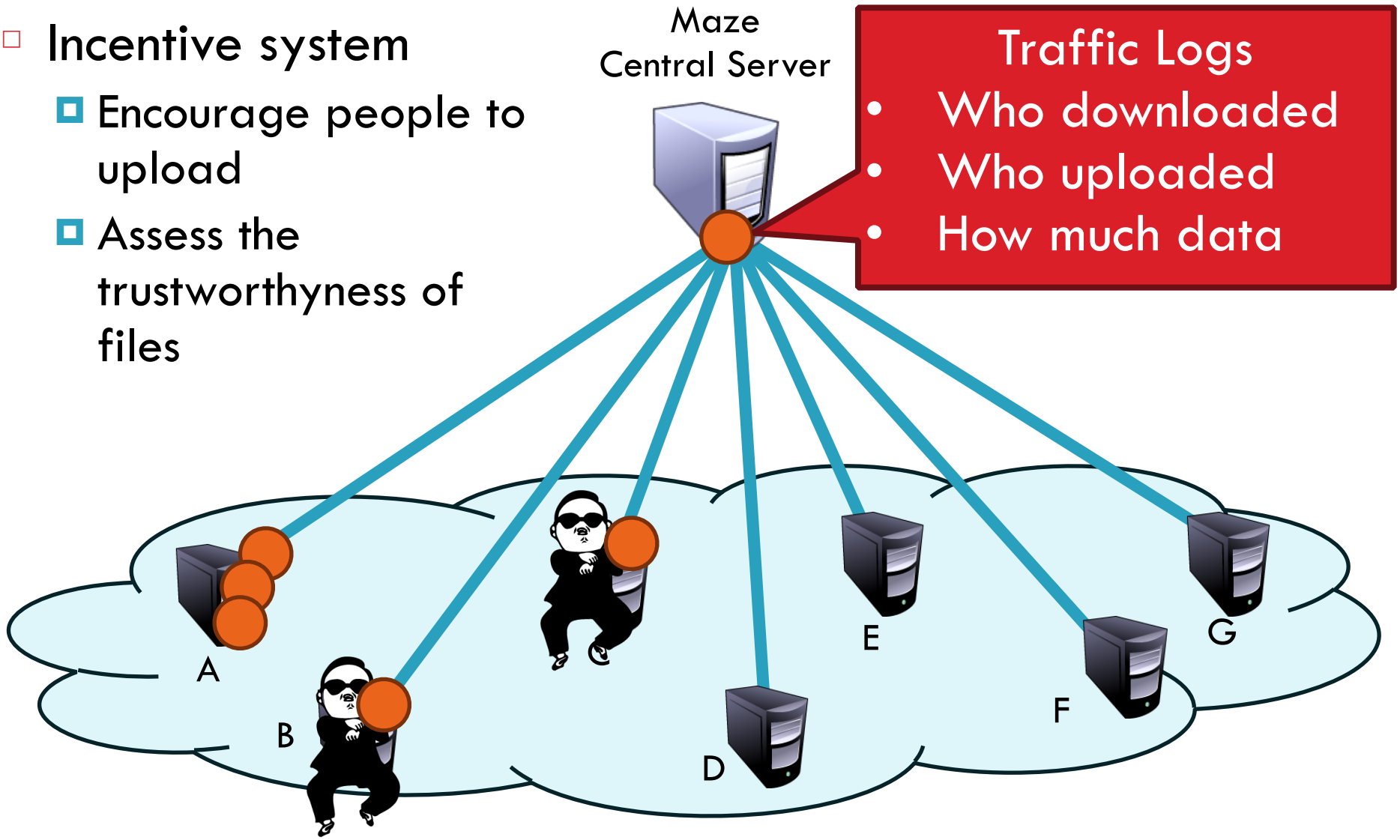


Centralized != Scalable?

- Another centralized protocol: Maze
  - Highly active network in China / Asia
  - Over 2 million users, more than 13 TB transferred/day
  - Central index servers run out of PKU
  - Survives because RIAA/MPAA doesn't exist in China
- Why is this interesting?
  - Shows centralized systems can work
    - Of course have to be smart about it…
  - Central servers "see" everything
    - Quite useful for research / measurement studies

Maze Architecture

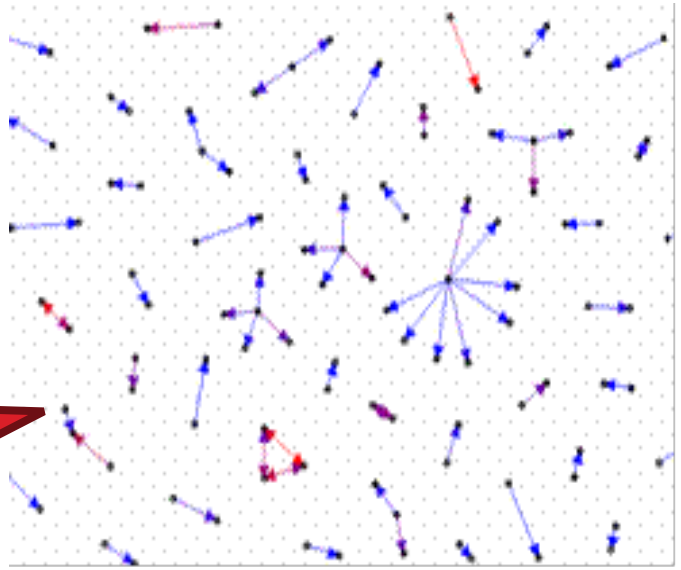
- Incentive system
  - Encourage people to upload
  - Assess the trustworthyness of files



Colluding Users

- Why and How of collusion
  - Collusion gets you points in Maze (incentive system)
  - Spawn fake users/identities for free
- Collusion detectors (ICDCS 2007)
  - Duplicate traffic across links
  - Pair-wise mutual upload behavior
  - Peer-to-IP ratio of clients
  - Traffic concentration

The Sybil Attack



Duplicate transfer graph: 100 links w/ highest duplicate transfer rates

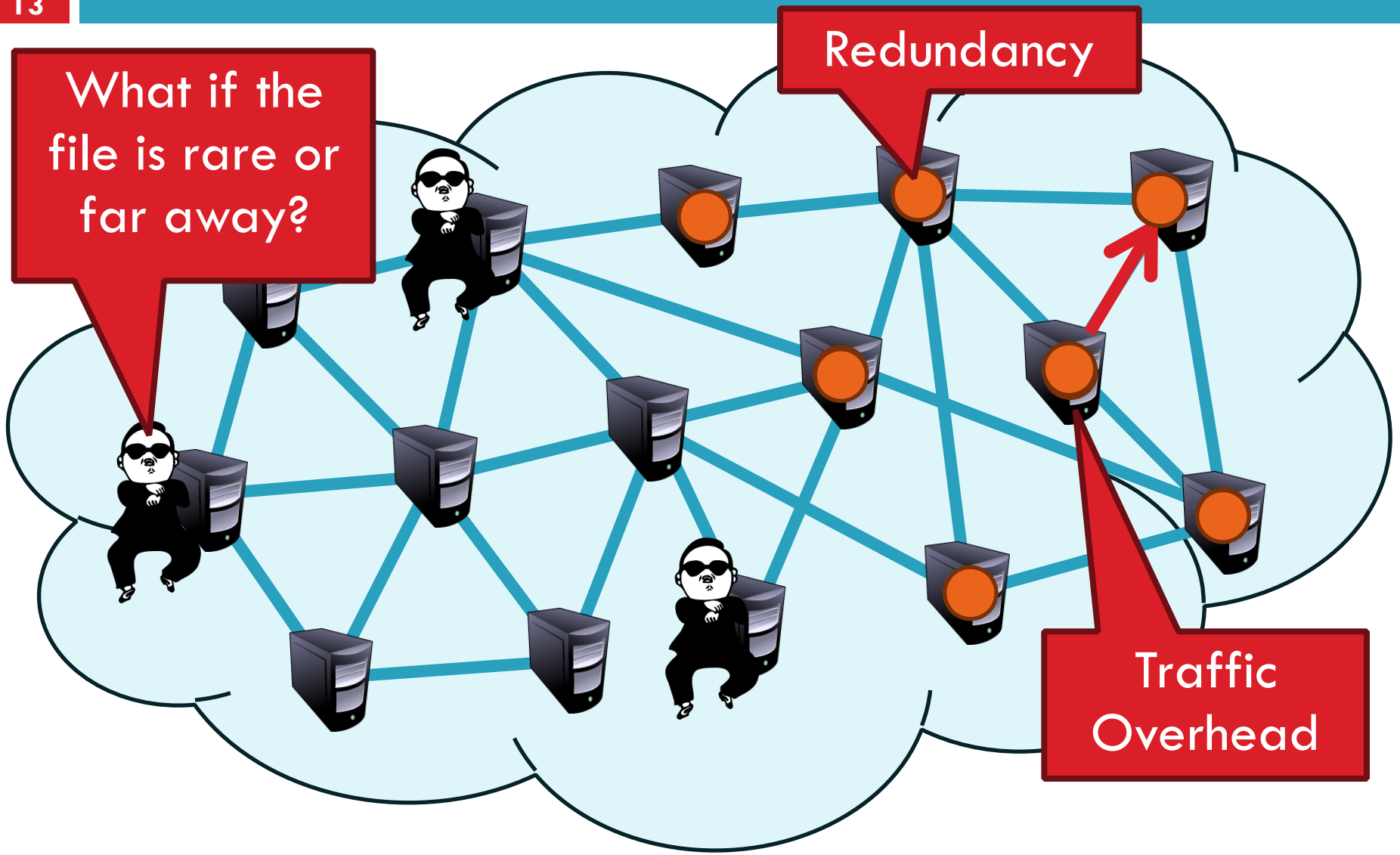
Unstructured P2P Applications

- Centralized systems have single points of failure
- Response: fully unstructured P2P
  - No central server, peers only connect to each other
  - Queries sent as controlled flood
  - Later systems are hierarchical for performance reasons
- Limitations
  - Bootstrapping: how to join without central knowledge?
  - Floods of traffic = high network overhead
  - Probabilistic: can only search a small portion of the system
  - Uncommon files are easily lost

Gnutella

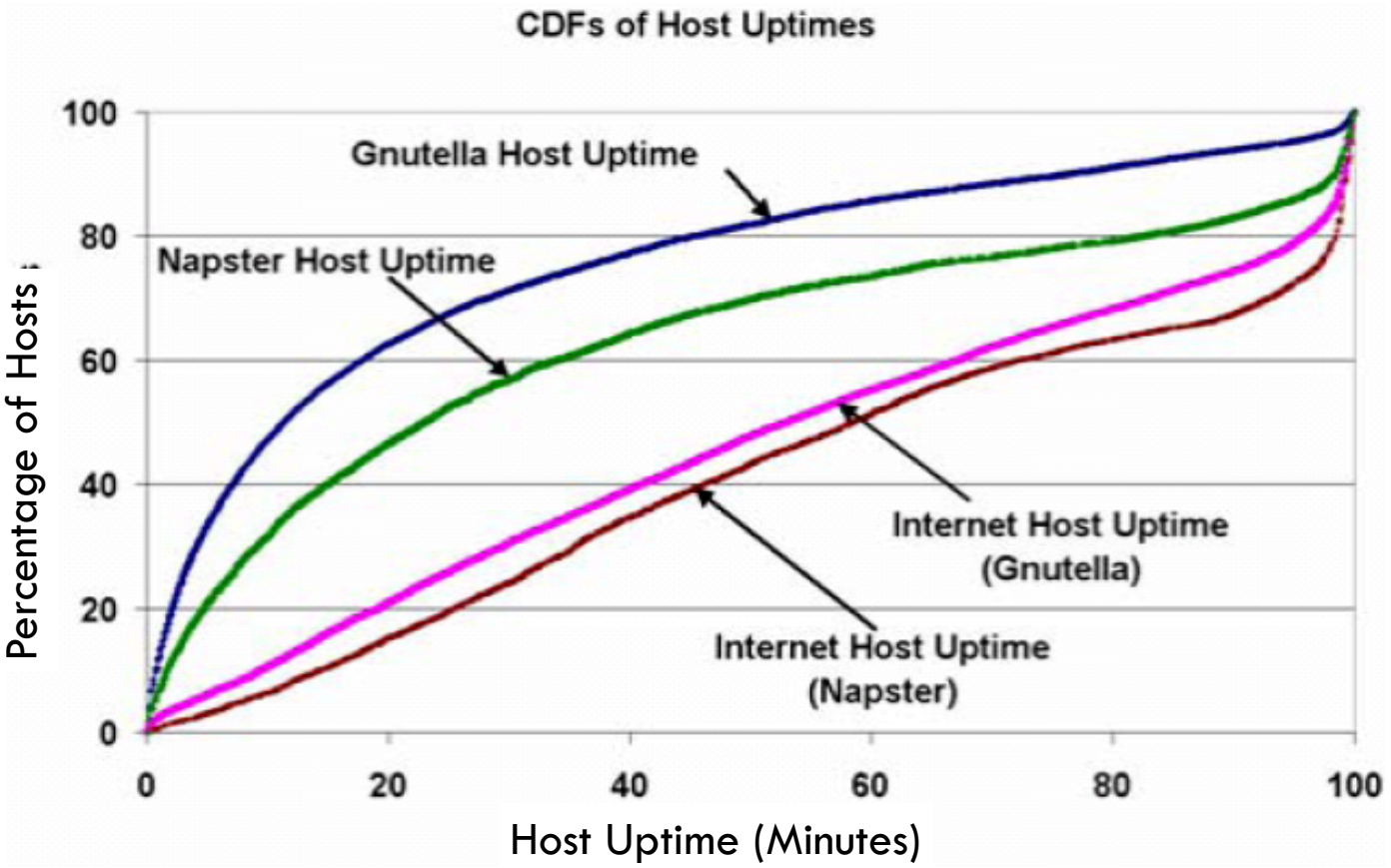
- First massively popular unstructured P2P application
  - Justin Frankel, Nullsoft, 2000
  - AOL was not happy at all
- Original design: flat network
  - Join via bootstrap node
  - Connect to random set of existing hosts
  - Resolve queries by localized flooding
    - Time to live fields limit hops
- Recent incarnations use hierarchical structure
- Problems
  - High bandwidth costs in control messages
  - Flood of queries took up all avail b/w for dialup users

File Search via Flooding in Gnutella



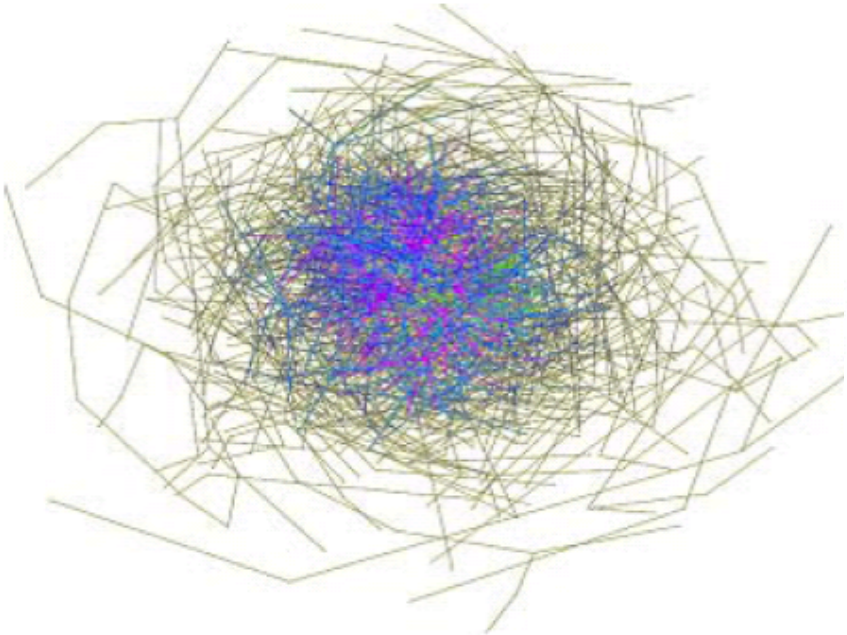
Peer Lifetimes

- Study of host uptime and application uptime (MMCN 2002)
  - 17,000+ Gnutella peers for 60 hours
  - 7,000 Napster peers for 25 hours

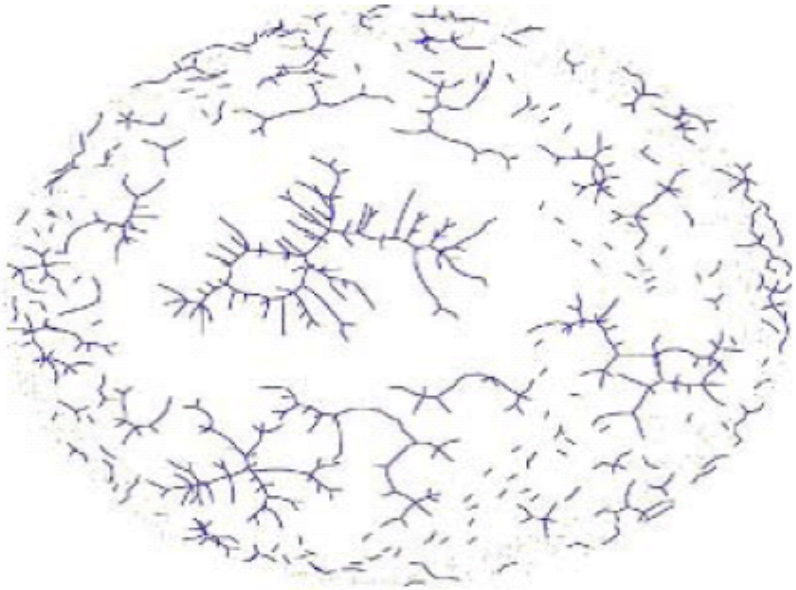


Resilience to Failures and Attacks

- 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



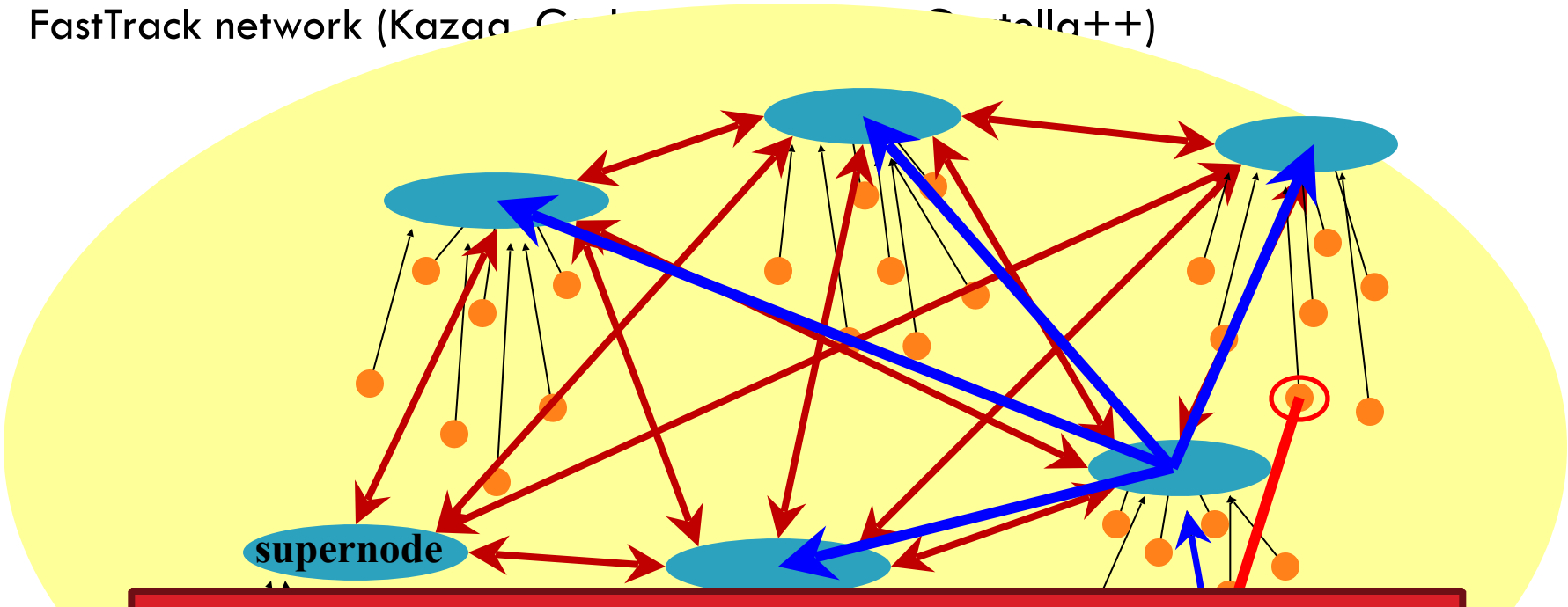
1771 Peers in Feb, 2001



After random 30% of peers removed

Hierarchical P2P Networks

- FastTrack network (Kazaa, [illegible]tella++)



- Improves scalability
- Limits flooding
- Still no guarantees of performance
- What if a supernode leaves the network?

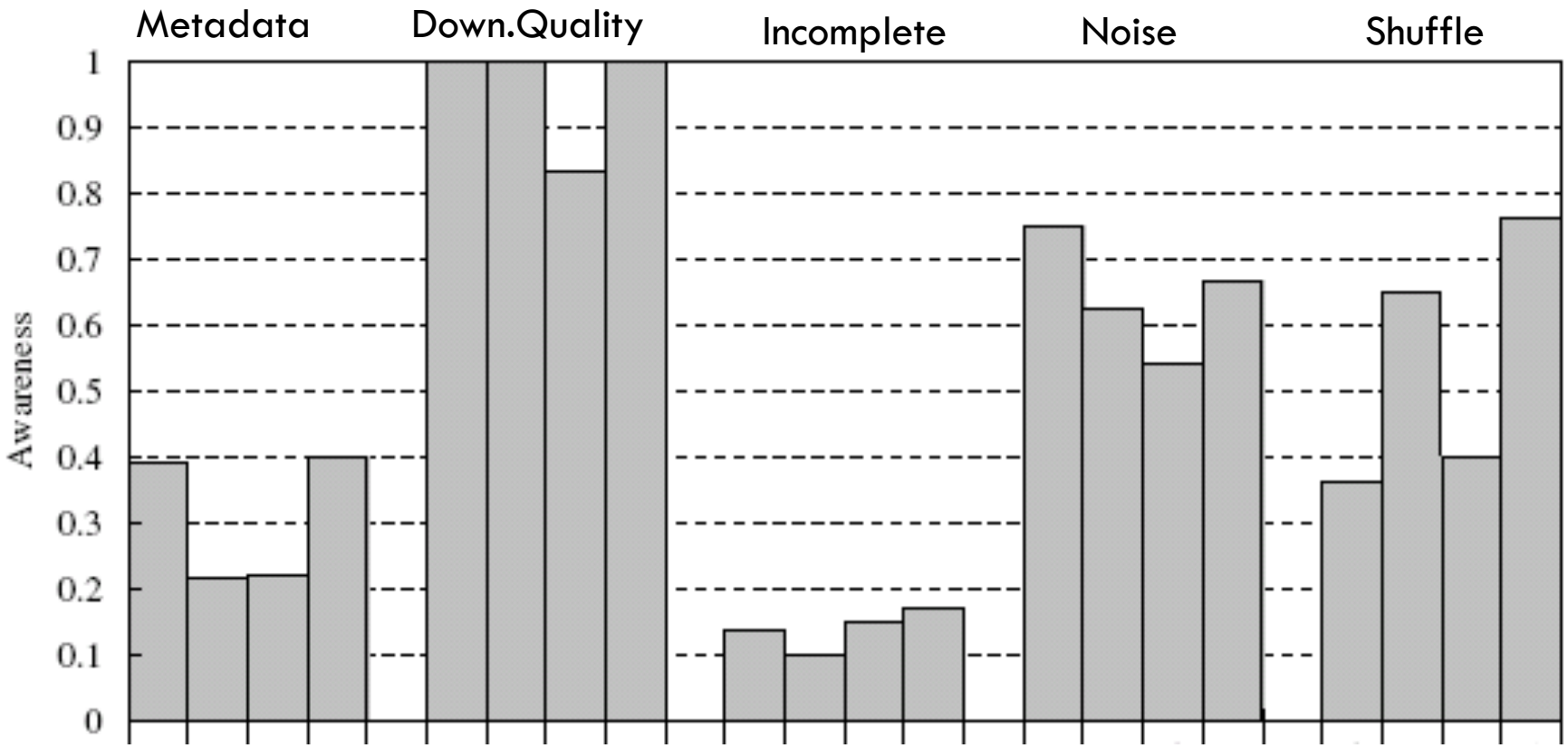
Kazaa

- Very popular from its inception
  - Hierarchical flooding helps improve scale
  - Large shift to broadband helped quite a bit as well
  - Based in Europe, more relaxed copyright laws

- New problem: poison attacks
  - Mainly used by RIAA-like organizations
  - Create many Sybils that distribute "popular content"
    - Files are corrupted, truncated, scrambled
    - In some cases, audio/video about copyright infringement
  - Quite effective in dissuading downloaders

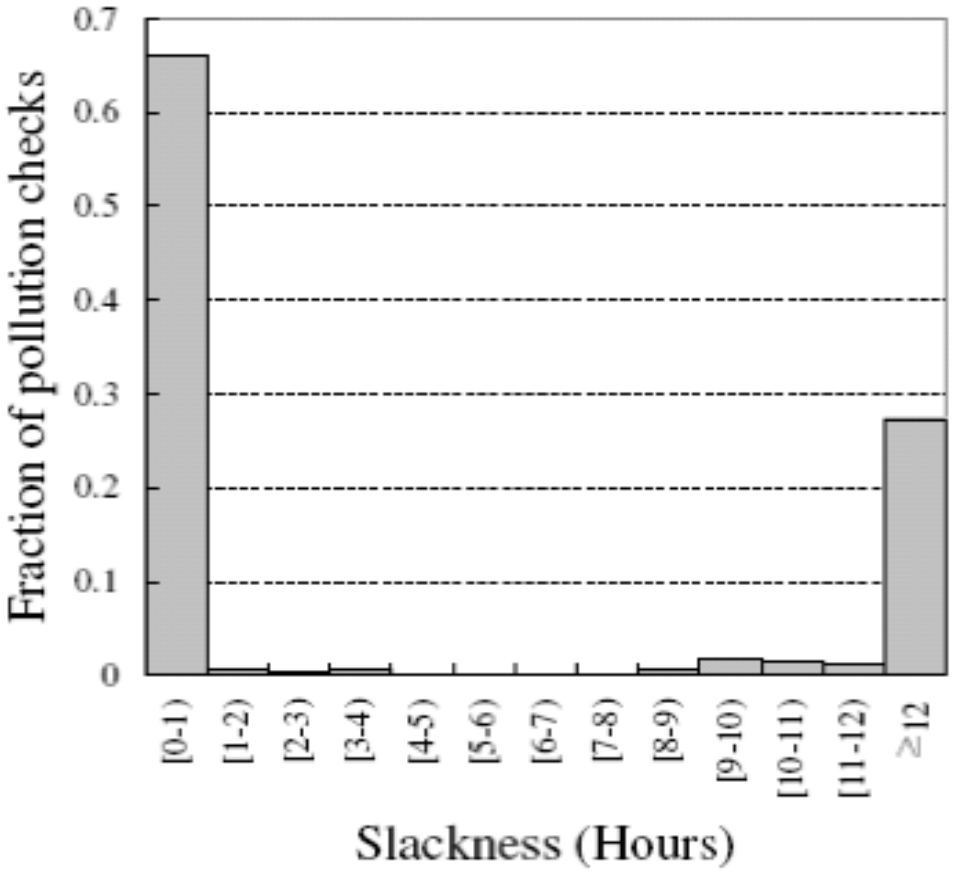
Data Poisoning on Kazaa

- Why is poisoning effective? (IPTPS 2006)
  - People don't check their songs!
  - Apparently not easy to detect file pollution!



Distribution of Poisoned Files

- Why are poisoned files so widely distributed?
  - "Slackness", even when users are "asked" to check files



Skype: P2P VoIP

- P2P client supporting VoIP, video, and text based conversation, buddy lists, etc.
  - Based on Kazaa network (FastTrack)
  - Overlay P2P network consisting of ordinary and Super Nodes (SN)
  - Ordinary node connects to network through a Super Node
- Each user registers with a central server
  - User information propagated in a decentralized fashion
- Uses a variant of STUN to identify the type of NAT and firewall

What's New About Skype

- MSN, Yahoo, GoogleTalk all provide similar functionality
  - But generally rely on centralized servers
- So why peer-to-peer for Skype?
  - One reason: cost
    - If redirect VoIP through peers, can leverage geographic distribution
    - i.e. traffic to a phone in Berlin goes to peer in Berlin, thus becomes a local call
  - Another reason: NAT traversal
    - Choose peers to do P2P rendezvous of NAT'ed clients
- Increasingly, MS is using infrastructure instead of P2P

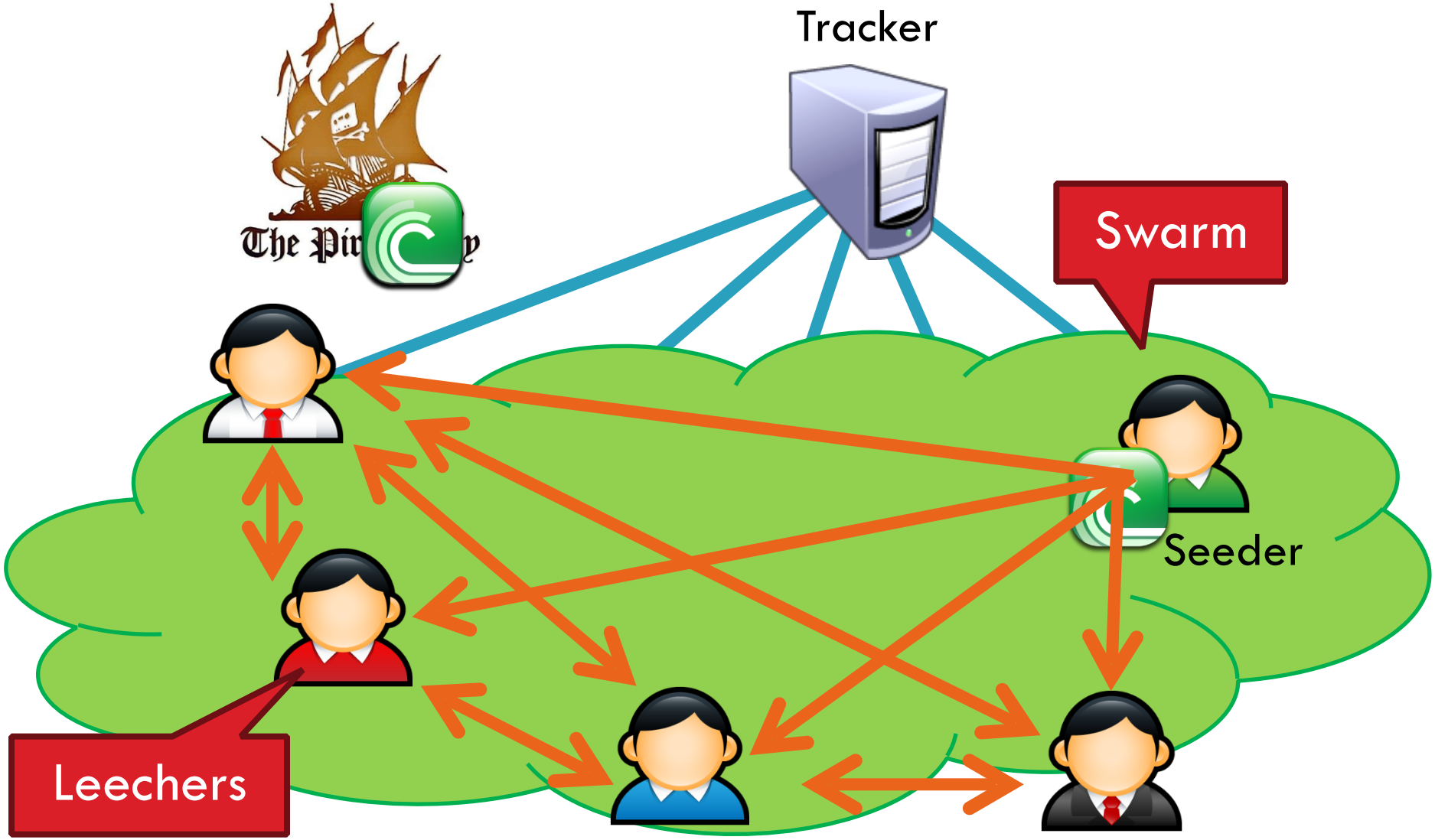
Outline

- Peer-to-Peer Overview
- Example: Bittorrent
  - µTP: Micro Transport Protocol
  - Cheating on BitTorrent

What is BitTorrent

- Designed for fast, efficient content distribution
  - Ideal for large files, e.g. movies, DVDs, ISOs, etc.
  - Uses P2P file swarming
- Not a full fledged P2P system
  - Does not support searching for files
  - File swarms must be located out-of-band
  - Trackers acts a centralized swarm coordinators
    - Fully P2P, trackerless torrents are now possible
- Insanely popular
  - 35-70% of all Internet traffic

BitTorrent Overview



.torrent File

- Contains all meta-data related to a torrent
  - File name(s), sizes
  - Torrent hash: hash of the whole file
  - URL of tracker(s)
- BitTorrent breaks files into pieces
  - 64 KB – 1 MB per piece
  - .torrent contains the size and SHA-1 hash of each piece
- Basically, a .torrent tells you
  - Everything about a given file
  - Where to go to start downloading

Torrent Sites

- Just standard web servers
  - Allow users to upload .torrent files
  - Search, ratings, comments, etc.
- Some also host trackers
- Many famous ones
  - Mostly because they host illegal content
- Legitimate .torrents
  - Linux distros
  - World of Warcraft patches

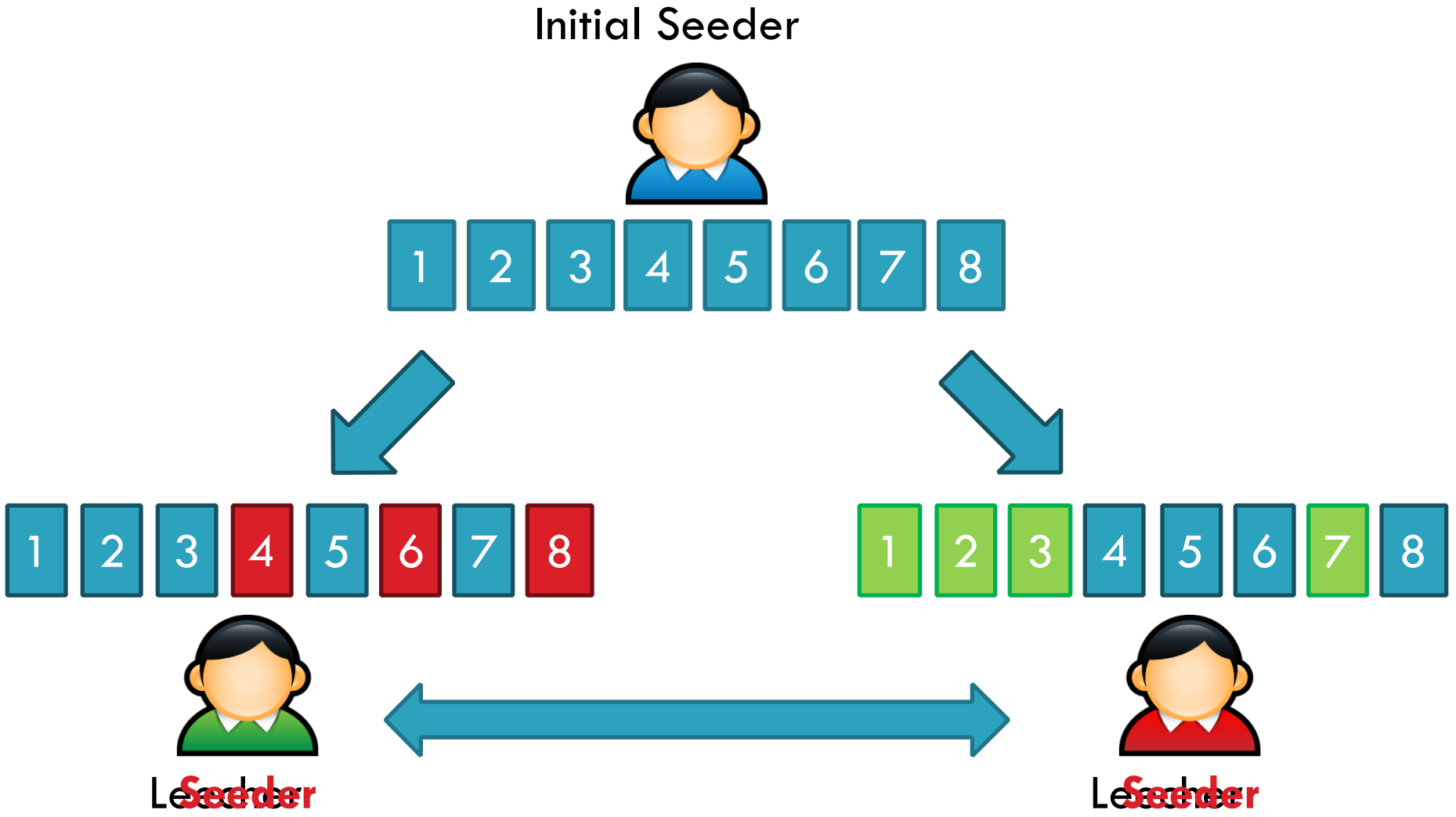
Torrent Trackers

- Really, just a highly specialized webserver
  - BitTorrent protocol is built on top of HTTP
- Keeps a database of swarms
  - Swarms identified by torrent hash
  - State of each peer in each swarm
    - IP address, port, peer ID, TTL
    - Status: leeching or seeding
    - Optional: upload/download stats (to track fairness)
  - Returns a random list of peers to new leechers

Peer Selection

- Tracker provides each client with a list of peers
  - Which peers are best?
    - Truthful (not cheating)
    - Fastest bandwidth
- Option 1: learn dynamically
  - Try downloading from many peers
  - Keep only the best peers
  - Strategy used by BitTorrent
- Option 2: use external information
  - E.g. Some torrent clients prefer peers in the same ISP

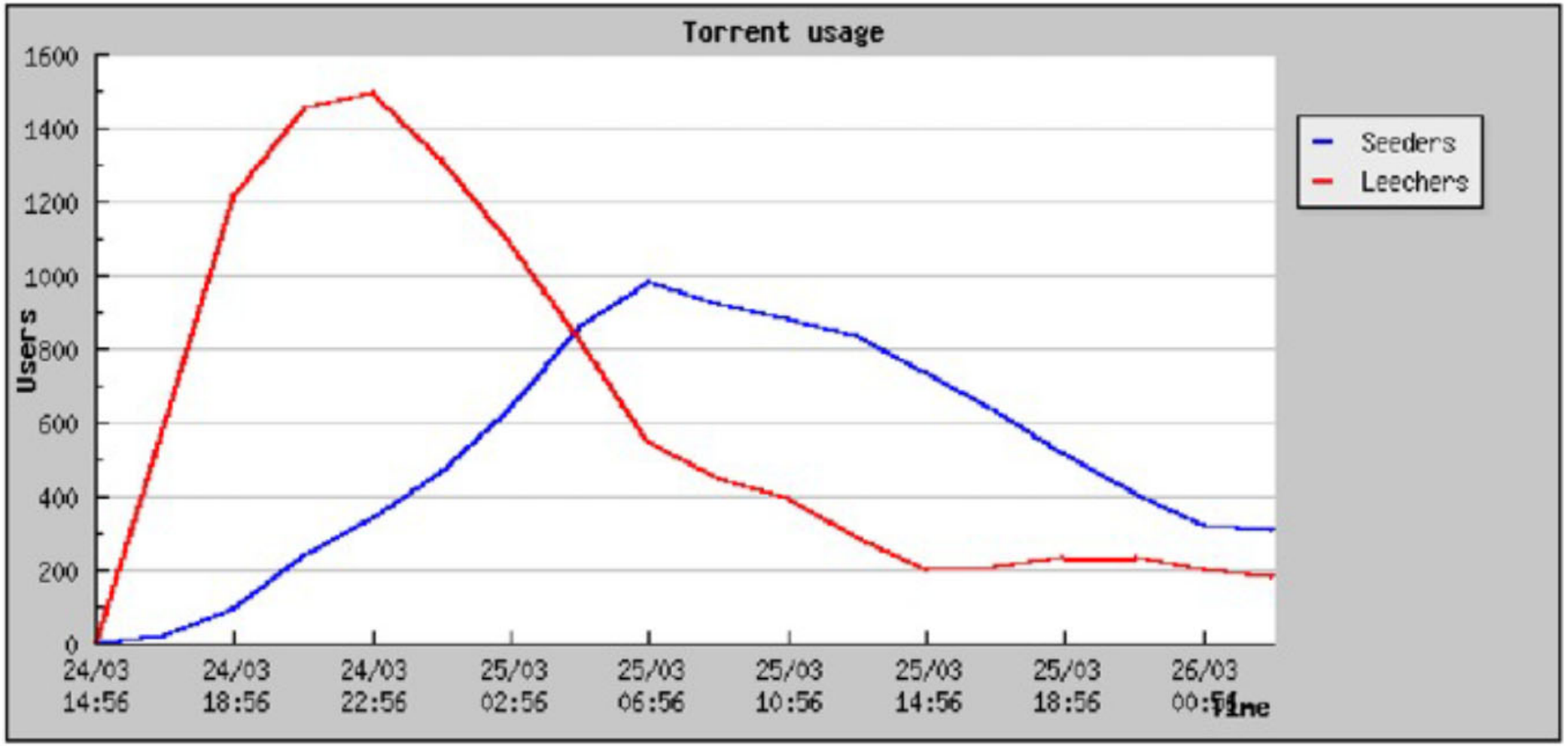
Sharing Pieces



The Beauty of BitTorrent

- More leechers = more replicas of pieces
- More replicas = faster downloads
  - Multiple, redundant sources for each piece
- Even while downloading, leechers take load off the seed(s)
  - Great for content distribution
  - Cost is shared among the swarm

Typical Swarm Behavior



Sub-Pieces and Pipelining

- Each piece is broken into sub-pieces
  - ~16 KB in size
- TCP Pipelining
  - For performance, you want long lived TCP connections (to get out of slow start)
  - Peers generally request 5 sub-pieces at a time
  - When one finished, immediately request another
  - Don't start a new piece until previous is complete
    - Prioritizes complete pieces
    - Only complete pieces can be shared with other peers

Piece Selection

- Piece download order is critical
  - Worst-case scenario: all leeches have identical pieces
    - Nobody can share anything :(
  - Worst-case scenario: the initial seed disappears
    - If a piece is missing from the swarm, the torrent is broken
- What is the best strategy for selecting pieces?
  - Trick question
  - It depends on how many pieces you already have

Download Phases

- 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

Upload and Download Control

- How does each peer decide who to trade with?
- Incentive mechanism
  - Based on tit-for-tat, game theory
  - "If you give a piece to me, I'll give a piece to you"
  - "If you screw me over, you get nothing"
  - Two mechanisms: choking and optimistic unchoke

A Bit of Game Theory

- Iterated prisoner's dilemma
- Very simple game, two players, multiple rounds
  - Both players agree: +2 points each
  - One player defects: +5 for defector, +0 to other
  - Both players defect: +0 for each
- Maps well to trading pieces in BitTorrent
  - Both peers trade, they both get useful data
  - If both peers do nothing, they both get nothing
  - If one peer defects, he gets a free piece, other peer gets nothing
- What is the best strategy for this game?

Tit-for-Tat

- Best general strategy for iterated prisoner's dilemma
- Meaning: "Equivalent Retaliation"

Rules

1. Initially: cooperate
2. If opponent cooperates, cooperate next round
3. If opponent defects, defect next round

Round			Points
1	*Cooperate*	*Cooperate*	*+2 / +2*
		Totals:	*+14 / +14*

Choking

- Choke is a temporary refusal to upload
  - Tit-for-tat: choke free riders
  - Cap the number of simultaneous uploads
    - Too many connections congests your network
  - Periodically unchoke to test the network connection
    - Choked peer might have better bandwidth

Optimistic Unchoke

- Each peer has one optimistic unchoke slot
  - Uploads to one random peer
  - Peer rotates every 30 seconds
- Reasons for optimistic unchoke
  - Help to bootstrap peers without pieces
  - Discover new peers with fast connections

BitTorrent Protocol Fundamentals

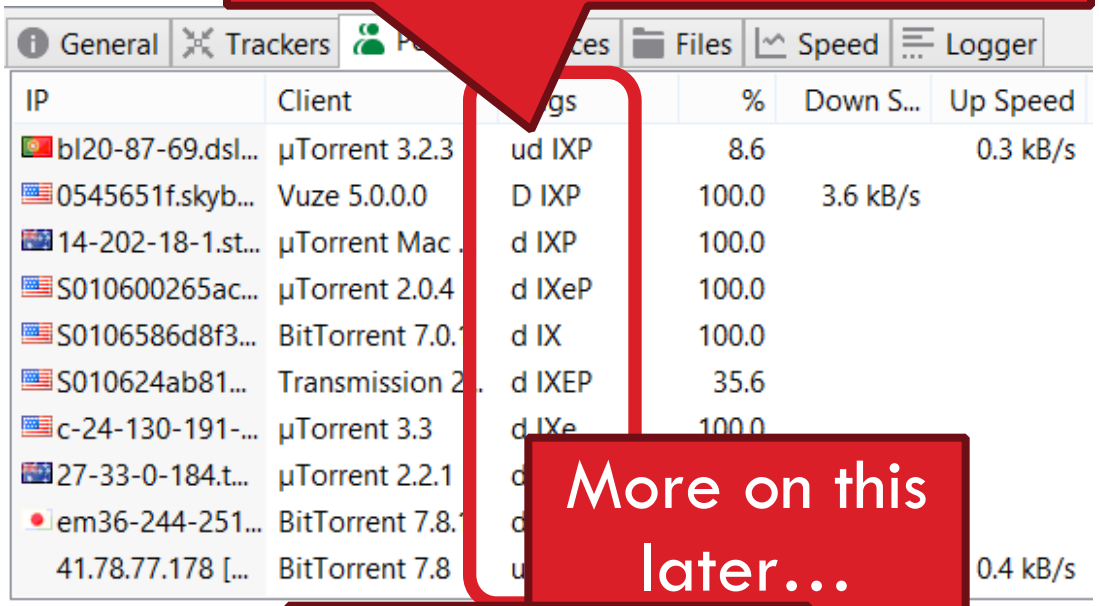
- BitTorrent divides time into rounds
  - Each round, decide who to upload to/download from
  - Rounds are typically 30 seconds
- Each connection to a peer is controlled by four states
  - Interested / uninterested – do I want a piece from you?
  - Choked / unchoked – am I currently downloading from you?
- Connections are bidirectional
  - You decide interest/choking on each peer
  - Each peer decides interest/chocking on you

Connection States

Error states. Connection should be closed.

Most peers are d or D. No need to connect with uninteresting peers.

- K – [illegible]nterested and unchoked
- S – snubbed (no data received in 60 seconds)
- F – piece(s) failed to hash
- Upload control
  - u – interested and choked
  - U – interested and unchoked
  - O – optimistic unchoke
  - ? – uninterested and unchoked
- Connection information
  - I – incoming connection
  - E/e – Using protocol encryption



More on this later…

More on this next week

- h – [illegible]
- P – [illegible]
- How was thi[illegible]er located?
  - H – DHT (distributed hash table)
  - L – local peer discovery (multicast)
  - X – peer exchange

Upload-Only Mode

- Once a peer completes a torrent, it becomes a seed
  - No downloads, no tit-for-tat
  - Who to upload to first?
- BitTorrent policy
  - Upload to the fastest known peer
  - Why?
  - Faster uploads = more available pieces
  - More available pieces helps the swarm

Trackerless Torrents

- New versions of BitTorrent have the ability to locate swarms without a tracker
  - Based on a P2P overlay
  - Distributed hash table (DHT)
- Recall: peers located via DHT are given “H” state
- More on this next week

Outline

- BitTorrent Basics
- µTP: Micro Transport Protocol
- Cheating on BitTorrent

BitTorrent and TCP

- BitTorrent accounts for 35-70% of all Internet traffic
- Thus, BitTorrent's behavior impacts everyone
- BitTorrent's use of TCP causes problems
  - Long lived, BitTorrent TCP flows are "elephants"
    - Ramp up past slow start, dominate router queues
  - Many applications are "mice," get trampled by elephants
    - Short lived flows (e.g. HTTP traffic)
    - Delay sensitive apps (i.e. VoIP, SSH, online games)
- Have you ever tried using SSH while using BitTorrent?

Making BitTorrent Play Nice

- Key issue: long-lived TCP flows are aggressive
  - TCP is constantly probing for more bandwidth
  - TCP induces queuing delay in the network
- Does BitTorrent really need to be so aggressive?
  - BitTorrent is not delay sensitive
    - Do you care if your download takes a few minutes longer?
  - BitTorrent is low-priority background traffic
    - You probably want to do other things on the Internet while BitTorrent is downloading
- Solution: use less a less aggressive transport protocol for BitTorrent

Micro Transport Protocol (µTP)

- Designed by BitTorrent, Inc.
- UDP-based transport protocol
- Uses LEDBAT principals
- Duplicates many TCP features
  - Window based sending, advertised windows
  - Sequence numbers (packet based, not byte based)
  - Reliable, in-order packet delivery
- Today: widely adopted by BitTorrent clients and open-sourced

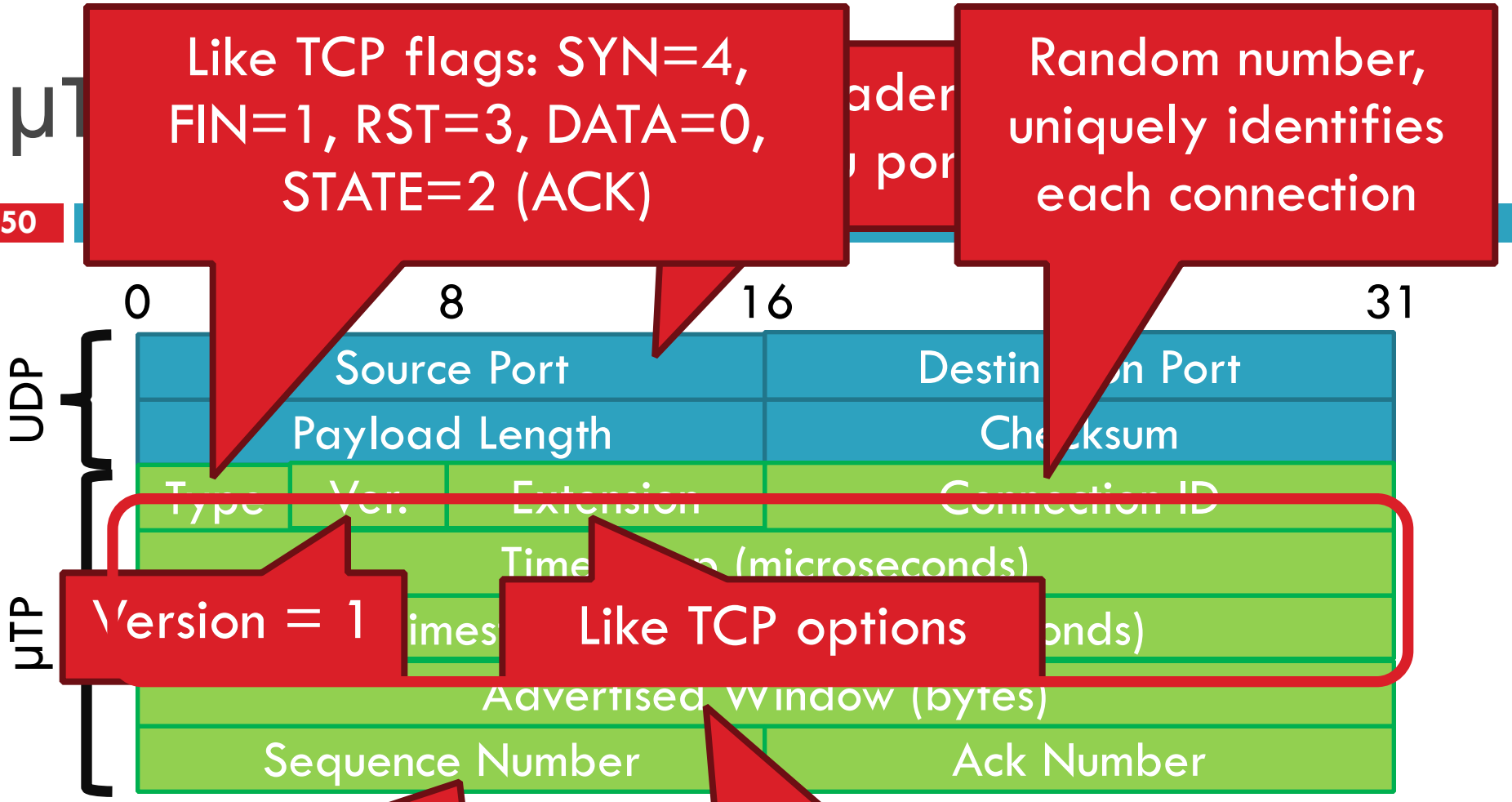
µTP and LEDBAT

- µTP is based on IETF LEDBAT standard (RFC 6817)
- Low Extra Delay Background Transport
  - Low delay congestion control algorithm
  - Seeks to use all available bandwidth…
  - … without increasing queuing delay on the path
- Goal: fast transfer of bulk data in the background
  - Use all available bandwidth (fast transfer speed)
  - … but, do not starve other applications
    - Background data transfer is not delay sensitive
    - Backoff gracefully and give bandwidth to delay sensitive applications

LEDBAT Details

- Delay-based congestion control protocol
  - Similar algorithm to TCP Vegas
  - Measure one-way delay, reduce rate when delay increases
- Constraint: be less aggressive than TCP
  - React early to congestion and slow down
  - Do not induce queuing delay in the network
- LEDBAT is a “scavenger” cc protocol
  - Scavenge unused bandwidth for file transfer
  - … but don’t take bandwidth from other flows

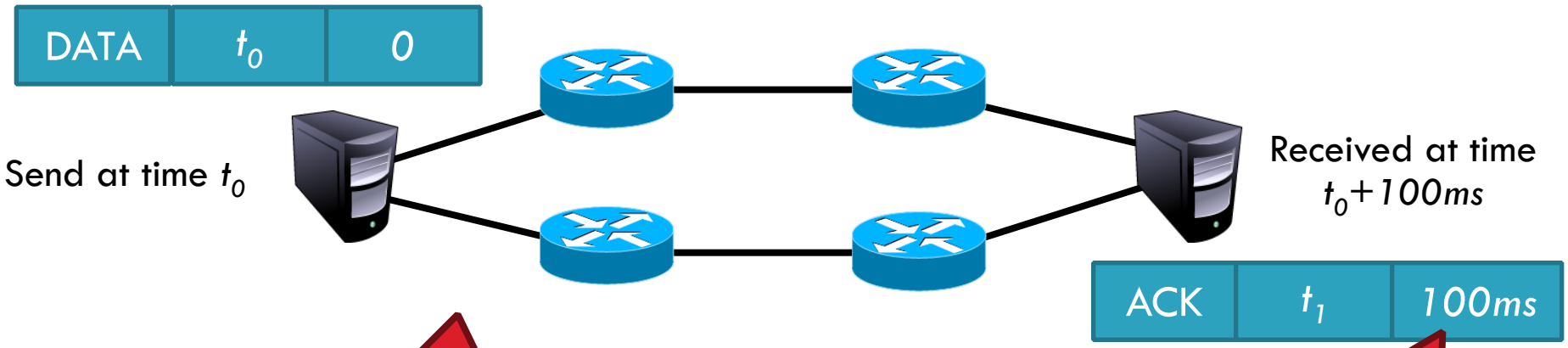
µT



- µTP header is similar to TCP
- Important new fields are the timestamps

Timestamps and Delay

- Timestamps used to measure one-way delay
  - Timestamp: time at which packet was sent
  - Timestamp Difference: sent time – received time



- e-way delay i
  - aths are not syn
  - path doesn't im

Sender knows one-way delay = 100ms

Time difference inserted into ACK

µTP Congest

µTP tries to keep one-way delay ~100ms

Estimate the baseline delay on the path

ba ast 2 minutes])

ou

Is delay below our target (positive value), or above our target (negative value)

off = CCONTROL_TARGET delay

Current delay on the path above the b

C

Time difference fro most recent ACK

Convert units from "time" to "packets"

scaled_gai or * window_factor

Finally, adjust the window size (may be + or – adjustment)

max_window = m _window + scaled_gain

More µTP Details

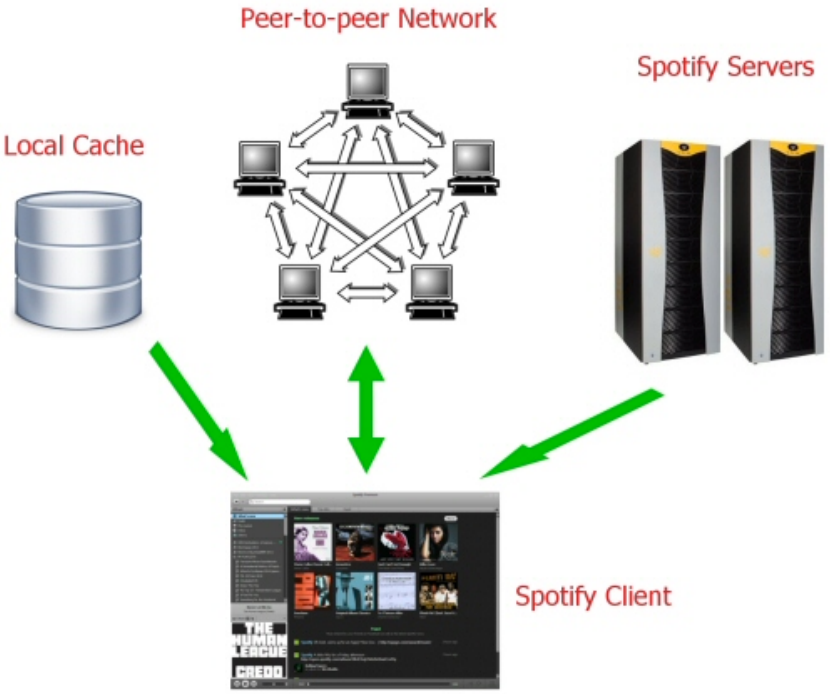
- Delay-based mechanism replaces slow start and additive increase
- What if a packet drops?
  - max_window = max_window * 0.5 (just like TCP)
- What if off_target is a large negative number?
  - max_window = 1 packet (don't starve the connection)
- Error handling in µTP :
  - Uses RTO like Tahoe to retransmit lost packets
  - Uses fast retransmit like TCP Reno

Discussion

- In this case, developing a new transport protocol was (arguably) the right decision
  - BitTorrent generates huge amounts of traffic
  - Whole Internet benefits if BitTorrent is more friendly
- However, inventing new protocols is hard
  - µTP reimplements most of TCP
    - RTO estimation, Nagle's algorithm, etc.
  - Early version of µTP performed much worse than TCP
    - Lots of bugs related to packet pacing and sizing
- Takeaway: develop new transport protocols only if absolutely necessary

Spotify

- Uses BT as basic protocol
  - Uses server for first 15s
  - Tries to find peers and download from them
  - Only 8.8% of bytes come from servers
- When 30s left
  - Starts searching for next track
  - Uses sever with 10s to go if no peers found



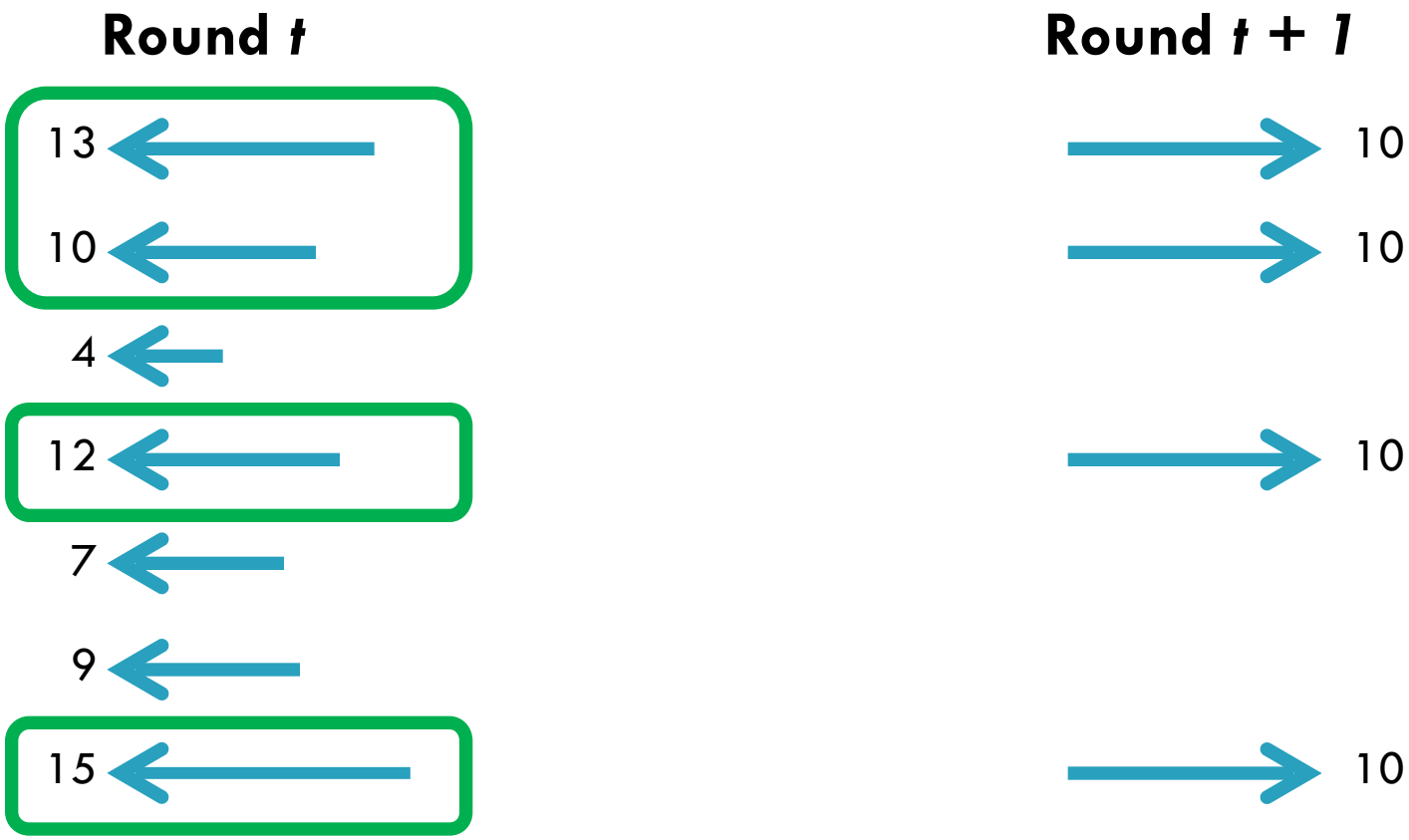
Outline

- BitTorrent Basics
- µTP: Micro Transport Protocol
- Cheating on BitTorrent

Incentives to Upload

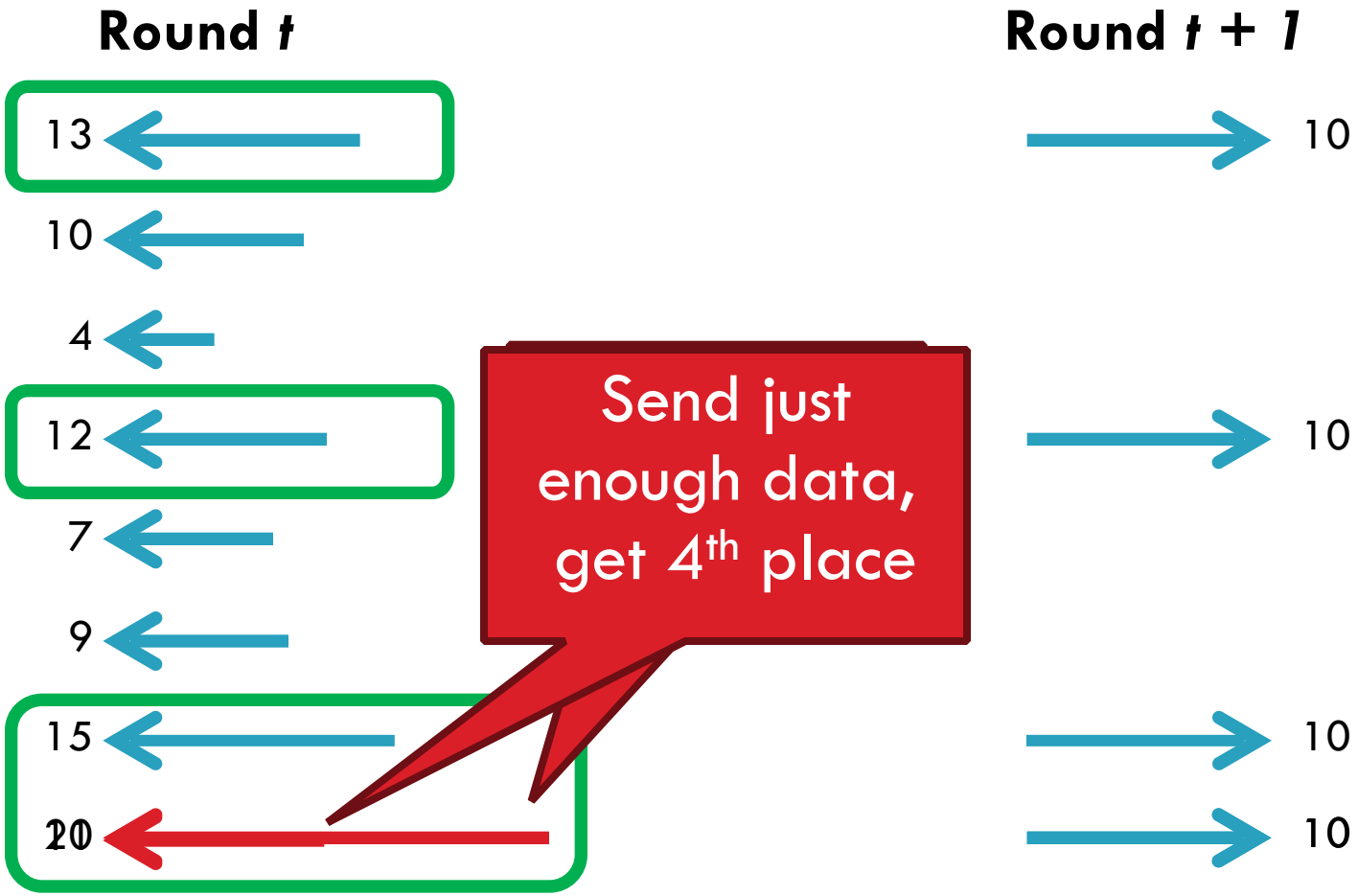
- Every round, a BitTorrent client calculates the number of pieces received from each peer
  - The peers who gave the most will receive pieces in the next round
  - These decisions are made by the unchoker
- Assumption
  - Peers will give as many pieces as possible each round
  - Based on bandwidth constraints,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

Unchoker Example

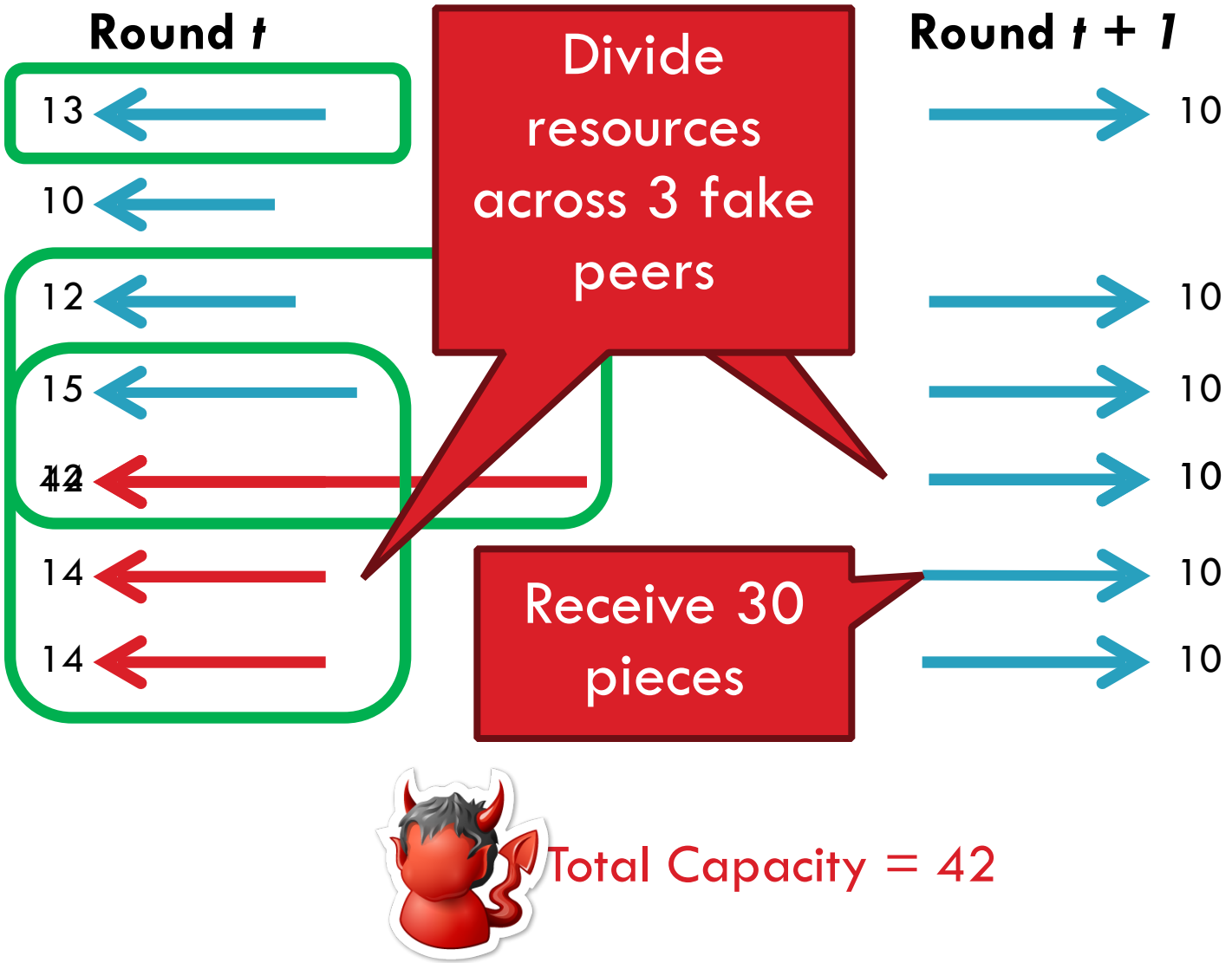


Abusing the Unchocker

- What if you really want to download from someone?



Sybil Attack



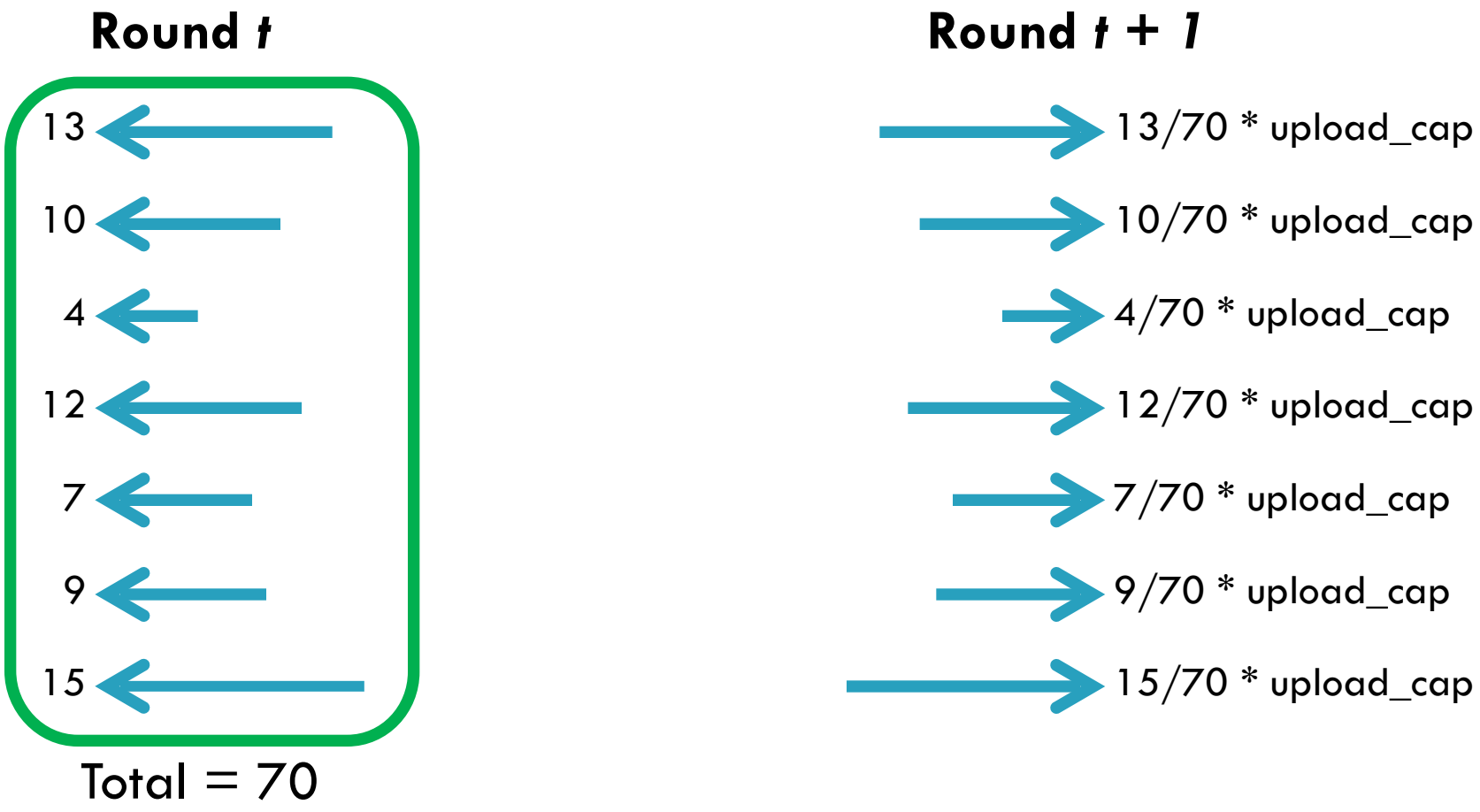
BitTyrant

- Piatek et al. 2007
  - Implements the “come in last strategy”
  - Essentially, an unfair unchoker
  - Faster than stock BitTorrent
    - For the Tyrant user
- Problem with BitTyrant
  - Tragedy of the commons
  - BitTyrant performs well if most peers are honest
  - As more peers use BitTyrant, performace suffers
  - If all users used BitTyrant, torrents wouldn’t work at all

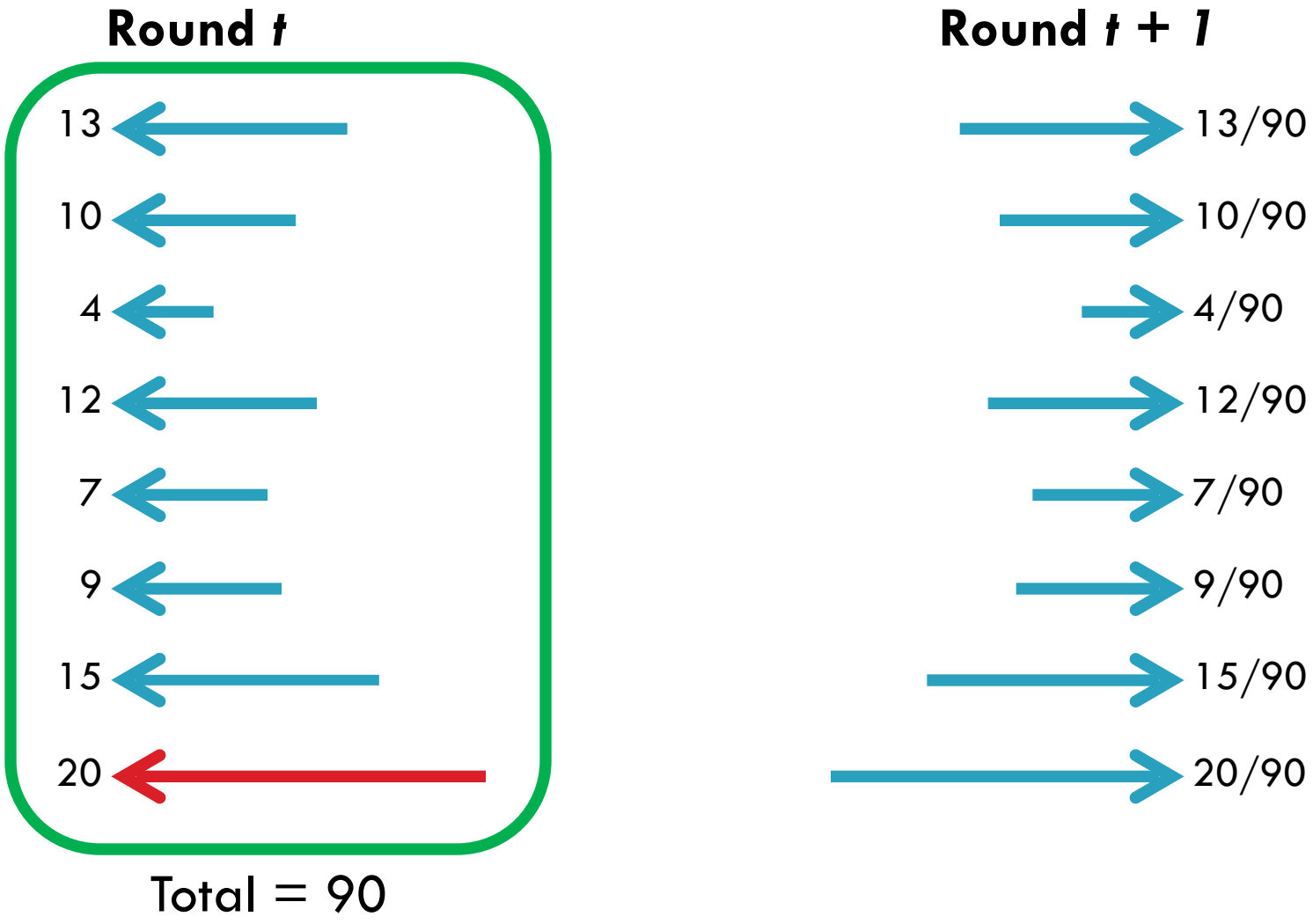
PropShare Unchoker

- Goal: modify BitTorrents incentive mechanisms to mitigate "come in last" and Sybil attacks
- Levin et al. 2008
  - Propose PropShare unchoker
  - PropShare clients allocate upload bandwidth proportionally across all peers
  - There is no longer a "top four"
- Can you cheat vs. PropShare?

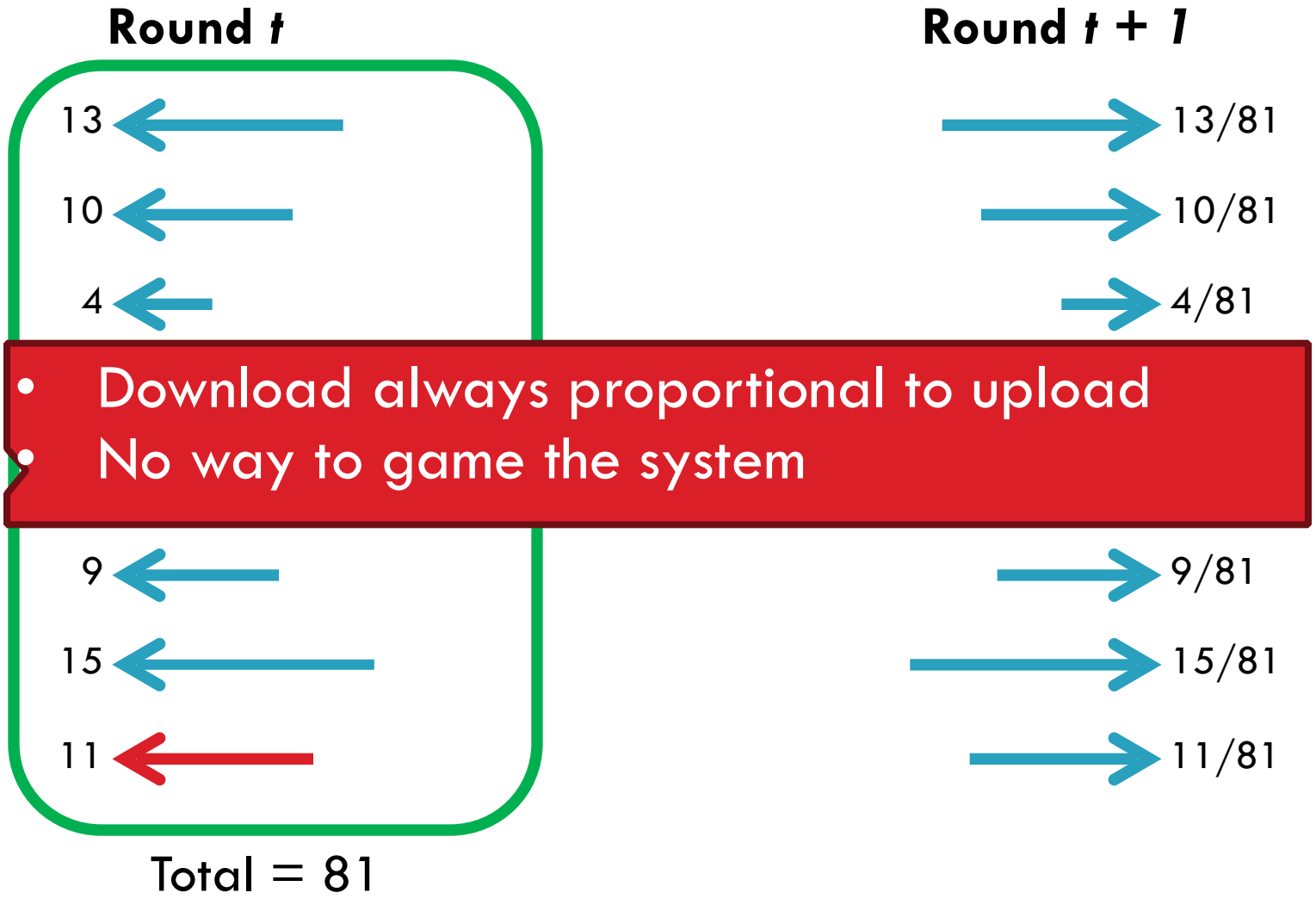
PropShare Unchoker



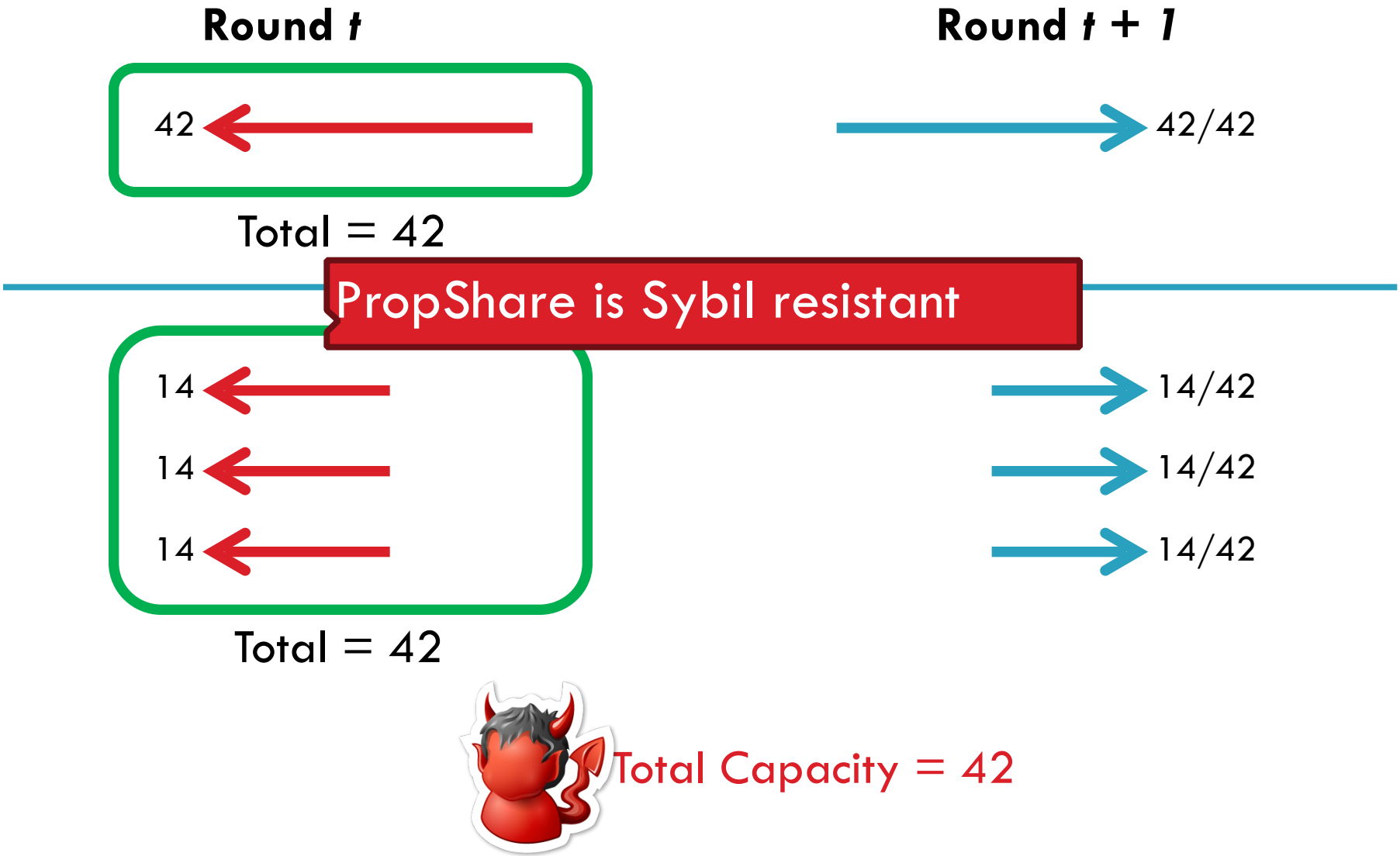
PropShare Resiliency to BitTyrant



PropShare Resiliency to BitTyrant



PropShare Resiliency to Sybils



Unchoker Summary

- BitTyrant and PropShare are both faster than stock BitTorrent
  - But for different reasons
- PropShare performs comparably to BitTyrant
- PropShare does not suffer from a tragedy of the commons
  - i.e. it's safe for all peers to use PropShare
  - Not true for BitTyrant

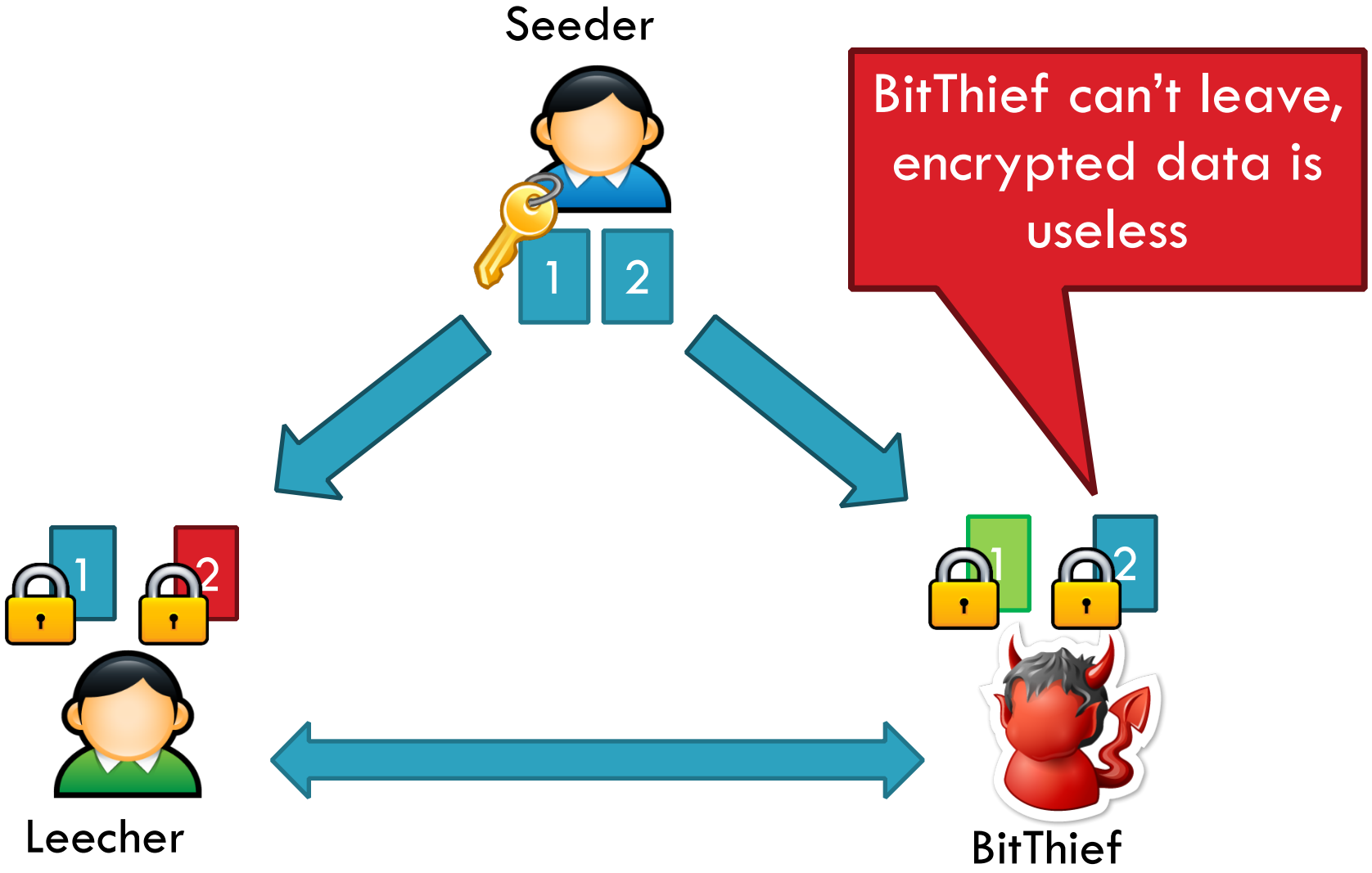
Abusing Optimistic Unchoking

- So far, assumed peers all have pieces to trade
  - Thus, all peers are interesting
- What about peers that have nothing?
  - The bootstrap mechanism is supposed to help them
  - Optimistic unchoke: reserve some bandwidth to give free pieces away (presumably to new peers)
- BitThief (Locher et al. 2006)
  - Abuses optimistic unchoke, uploads nothing
  - Swarm collapses if all peers use BitThief

BitThief Details

- Large-view exploit
  - The swarm is (potentially) huge
  - BitThief client tries to get optimistic unchoke from many, many peers
  - Will only receive one free piece from each
    - Since there is no reciprocal upload
  - But in aggregate, this is enough to finish download
- How to deal with this?
  - Enlist the help of peers
  - Have them verify that a given client uploads

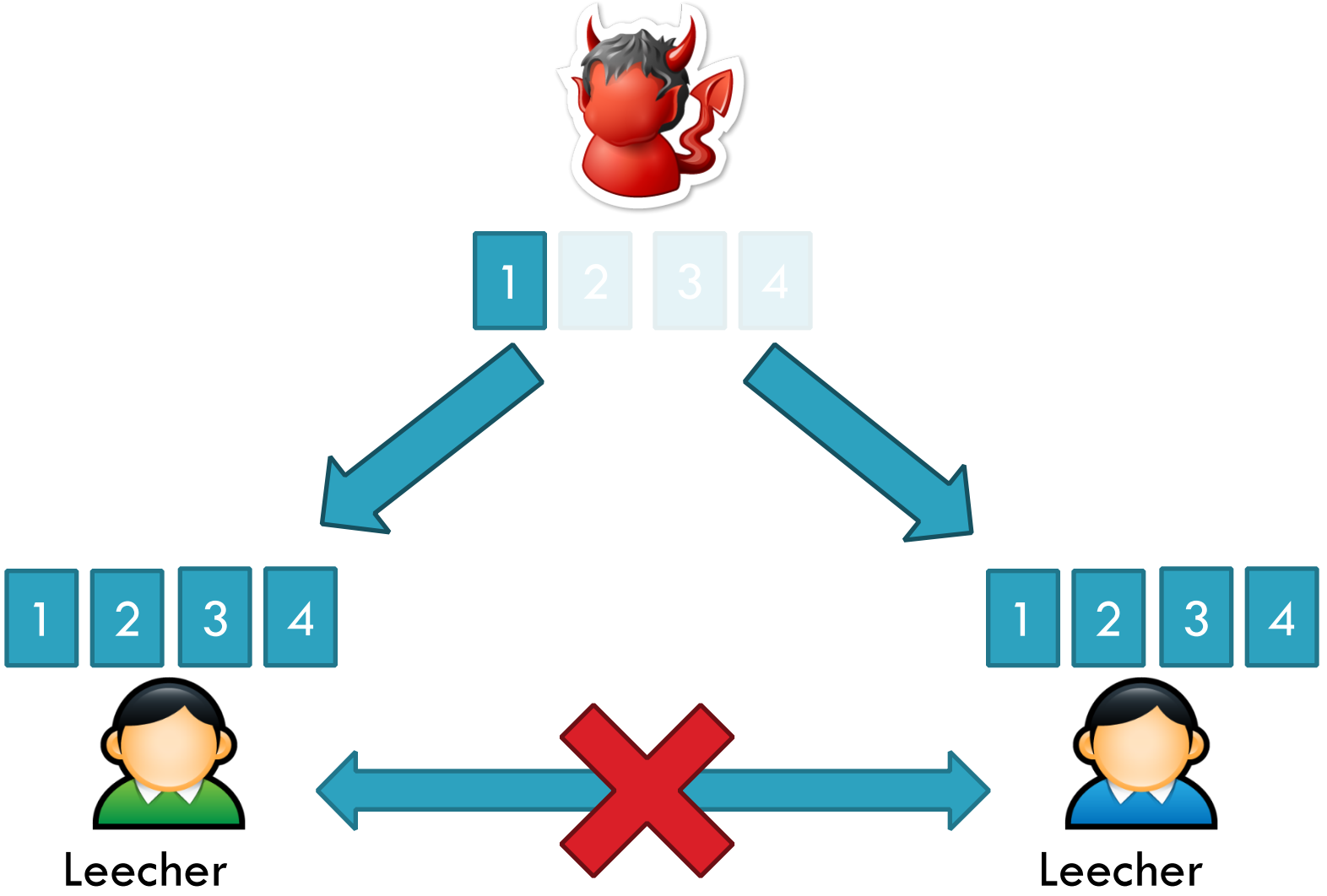
Encrypted Pieces



Abusing the Endgame

- Rare pieces are valuable
  - Make you popular, many people want to trade with you
  - More trading partners = faster downloads
- Selective piece revelation
  - You can't advertise pieces you don't have
    - Peers could detect this
  - But you can hide information about the pieces you have
- Why is this useful?
  - Pieces sent at time *t* impact your popularity at time *t+1*
  - Sending common pieces first, monopolize rare pieces

Strategic Piece Revelation



Conclusions

- BitTorrent is an extremely efficient tool for content distribution
  - Strong incentive system based on game theory
  - Most popular file sharing client since 2001
  - More active users than YouTube and Facebook combined
- However, BitTorrent is a large system with many different mechanisms
  - Ample room to modify the client, alter behavior
  - Cheating can happen, not all strategies are fair