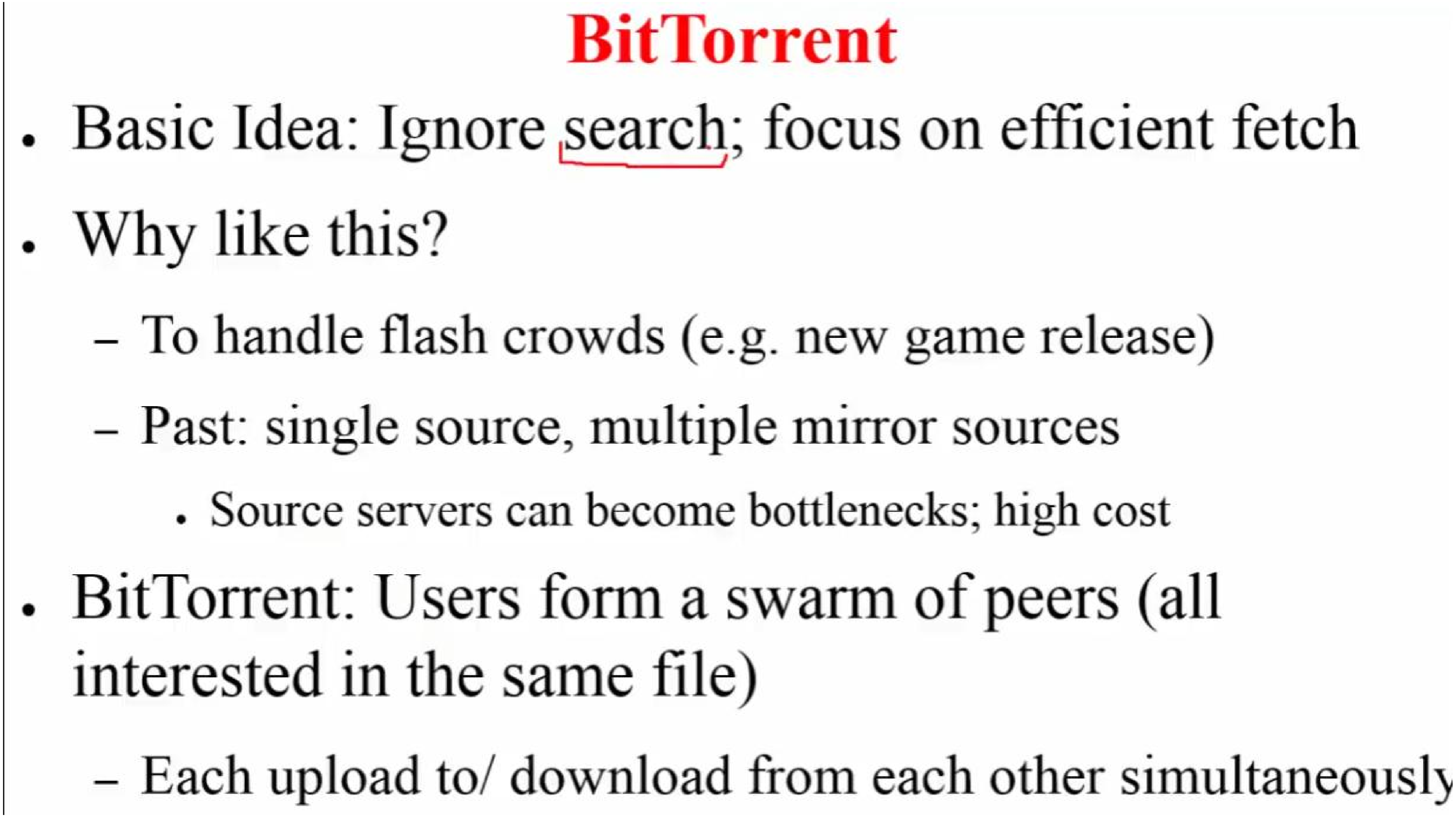
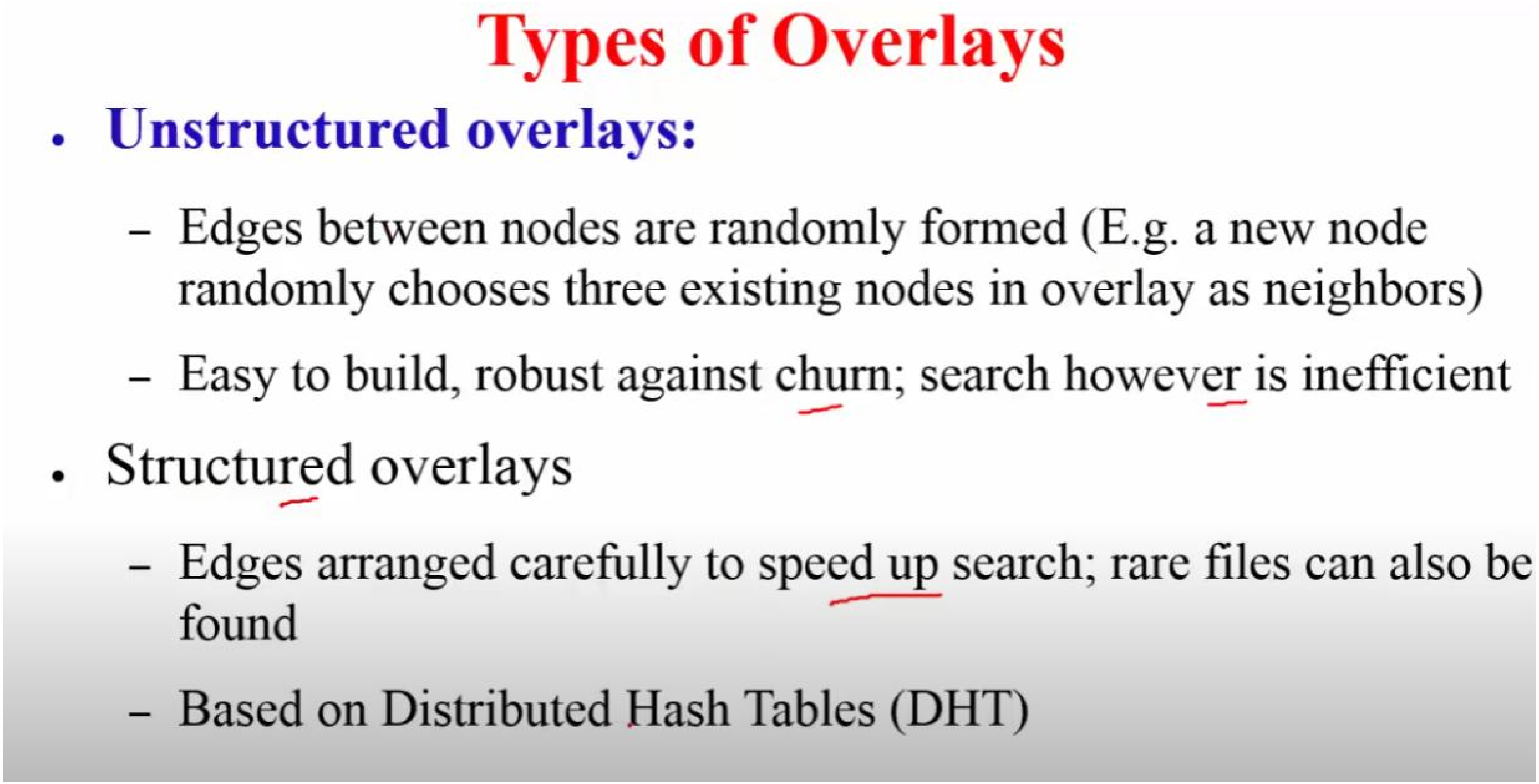


BitTorrent

- Basic Idea: Ignore search; focus on efficient fetch
- Why like this?
  - To handle flash crowds (e.g. new game release)
  - Past: single source, multiple mirror sources
    - Source servers can become bottlenecks; high cost
- BitTorrent: Users form a swarm of peers (all interested in the same file)
  - Each upload to/ download from each other simultaneousl

Discussion

- Pros
  - Encourages peers to share resources, discourages freeloaders
  - Can resume partially downloaded files
- Cons
  - Works well for “hot” content; not so much for obscure content
    - Performance deteriorates if swarm cools off
    - Search may be difficult
  - Single point of failure (tracker)

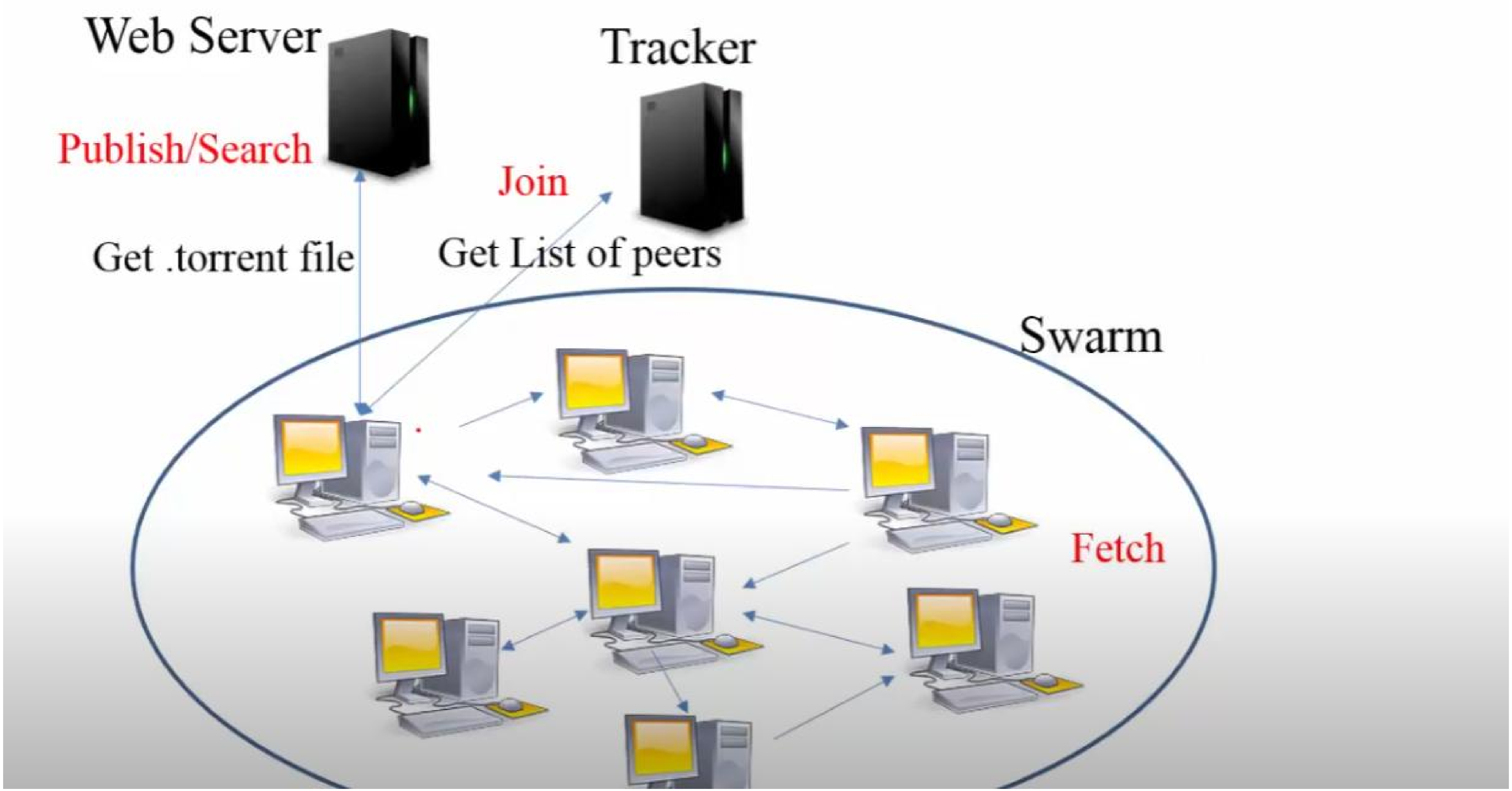


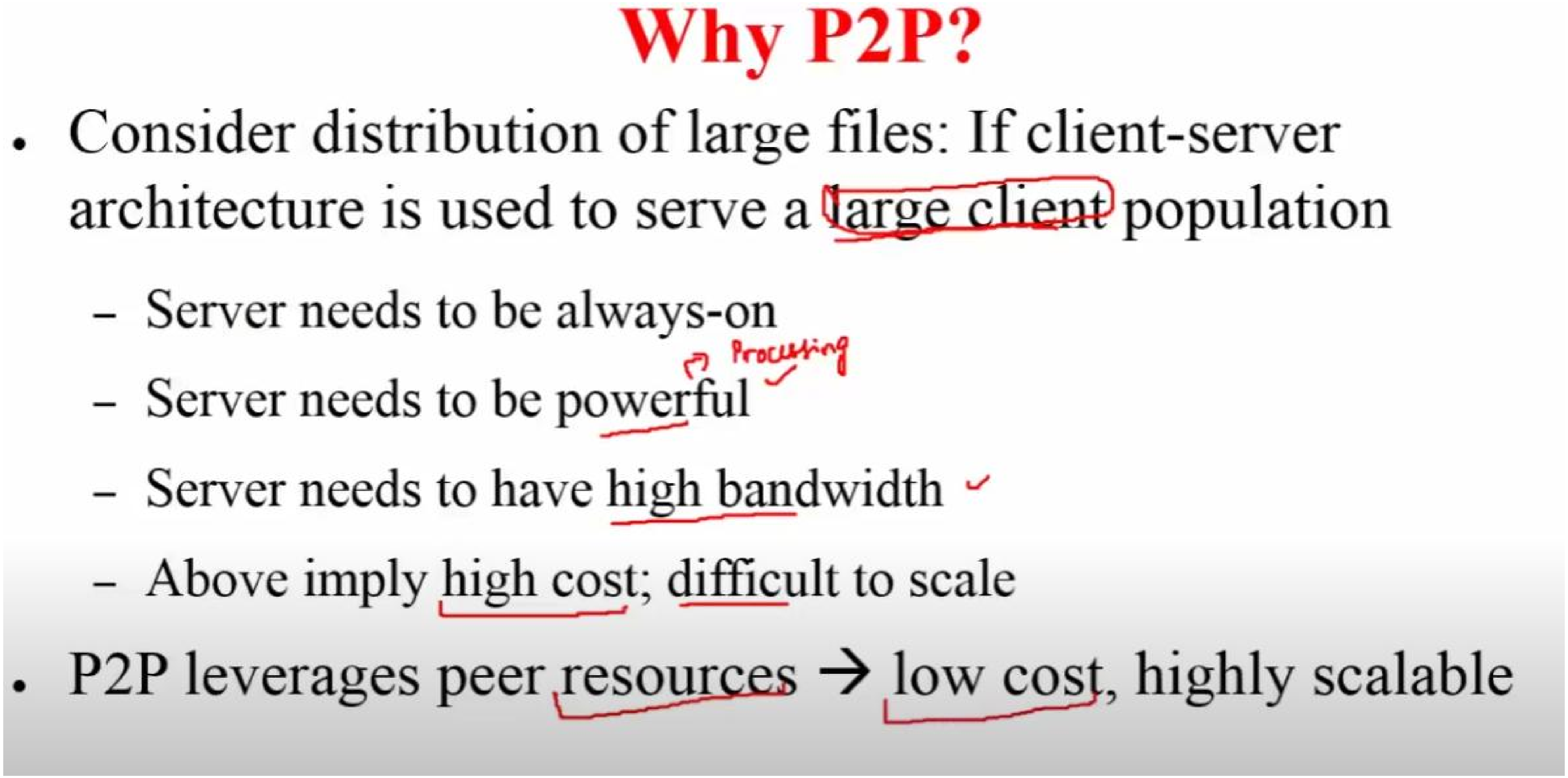
Types of Overlays

- **Unstructured overlays:**
  - Edges between nodes are randomly formed (E.g. a new node randomly chooses three existing nodes in overlay as neighbors)
  - Easy to build, robust against churn; search however is inefficient
- Structured overlays
  - Edges arranged carefully to speed up search; rare files can also be found
  - Based on Distributed Hash Tables (DHT)

Application Functionality

- Join : how does a user join the peer-to-peer overlay network?
- Publish : how does a user advertise files willing to share?
- Search : how does a user find a file?
- Fetch : how does a user download a file?
- Challenge: Peers come and go (churn); Peer's IP address can change over time


Web Server
Tracker
Publish/Search
Join
Get .torrent file
Get List of peers
Swarm
Fetch



Why P2P?

- Consider distribution of large files: If client-server architecture is used to serve a large client population
  - Server needs to be always-on
  - Server needs to be powerful Processing ✓
  - Server needs to have high bandwidth ✓
  - Above imply high cost; difficult to scale
- P2P leverages peer resources → low cost, highly scalable

Publish

- Create a torrent file (.torrent). Includes meta data about the file you wish to share and url of a tracker
- Meta Data:
  - Name of the file
  - SHA-1 hash of each piece (a file is divided into pieces)
  - Piece size (usually 256KB)
  - Length of the file

Publish

- Tracker: A server that coordinates file transfer; keeps track of peers in the swarm
  - Both public (any one can use) and private trackers available (only by invitation)
- Publish the torrent file on a web server
- Ensure a machine that has the entire file joins the swarm (initial seed)
  - Seed: A node with a complete file

Search/Join

- Search: Out of band (use google or on popular websites that host torrents) to find the torrent file
- Join: Contact "tracker" listed in the torrent file
  - Provide your IP-addr, Port info, amount uploaded/downloaded etc (do this periodically)
  - Tracker provides a list of peers who are downloading same file

Terminology

- Seed: A node with the complete file
- Leech: A node that is still downloading the file
- Peer: Runs BitTorrent client. Can be a seed or a leech
- Sub-piece: File is divided into pieces (typically 256KB). Each piece further sub-divided into sub-pieces (typically 16 no.).
  - Unit of request is a subpiece; 5 requests pipelined at once
  - A peer can upload only after receiving a complete piece

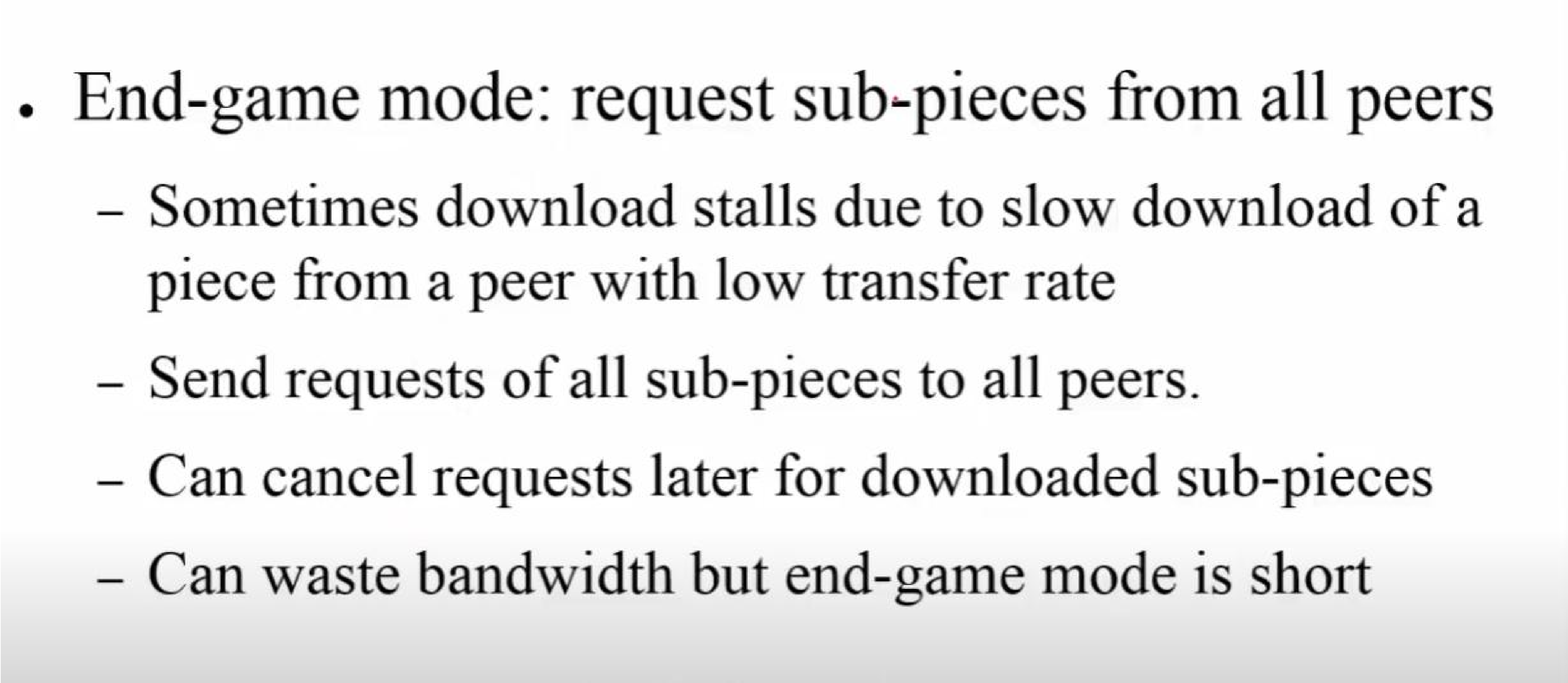
Fetch

- As part of Join, a leecher gets a list of peers who are downloading same file
- Which piece (sub-piece) to request?
  - Piece Selection Algorithm
- From whom to request? And whose request to accept or deny

Piece Selection

- Goal: Enable fast download of the entire file
- Challenge: Peers come and go; initial seeder may be taken down
  - Do not want a situation where none of the peers have the missing pieces → Need to ensure small overlap of pieces across peers
- Solution:
  - General Rule: Rarest First
  - At beginning: Random First Piece
  - At end: Endgame Mode

- Rarest first: request piece that is owned by least number of peers
  - Initial Seed can get more information out
  - Replicates rarest pieces as quickly as possible
- Random first piece: Do it at beginning
  - Helps assemble first piece fast so that upload can begin
  - Switch to rarest first after this



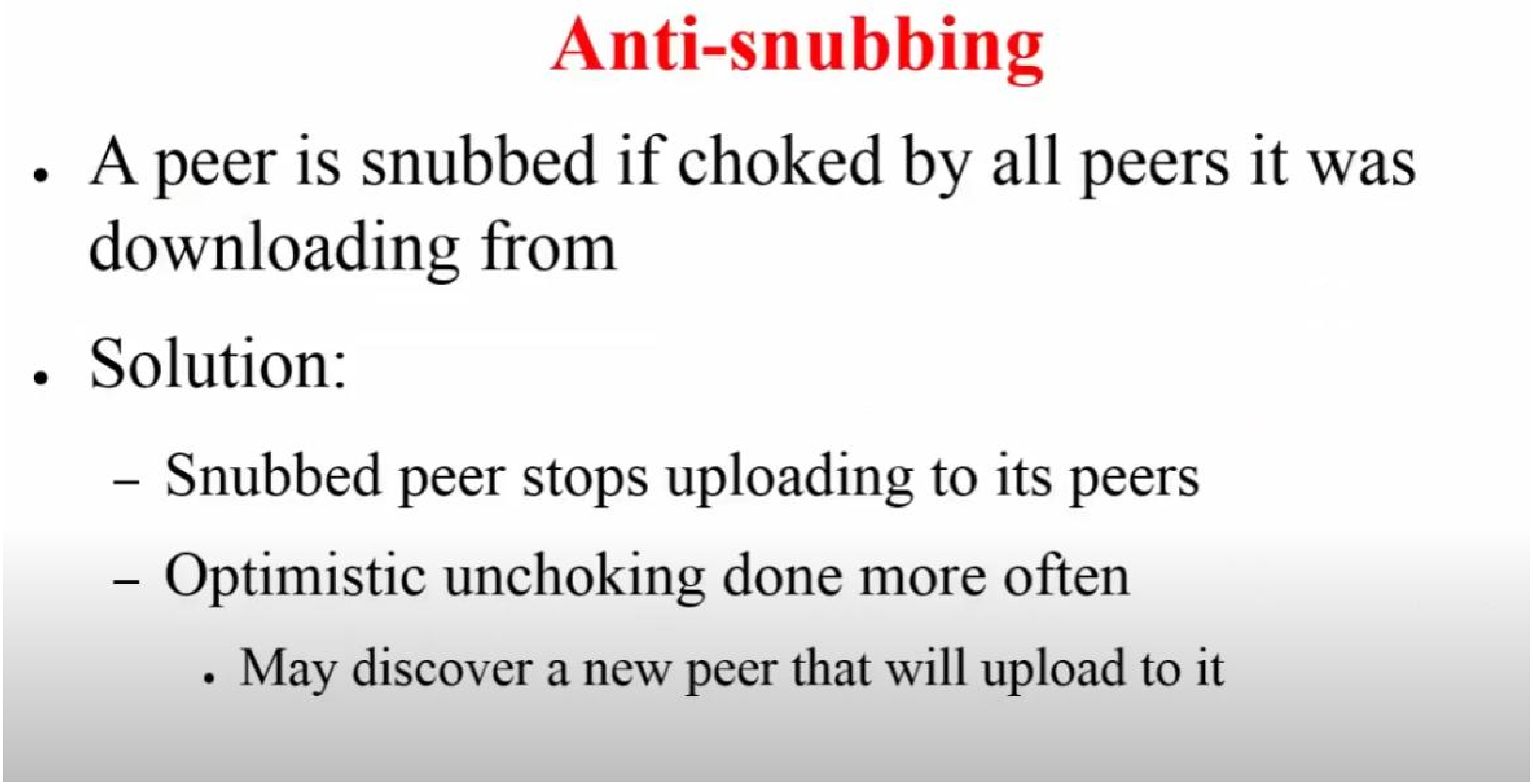
- End-game mode: request sub-pieces from all peers
  - Sometimes download stalls due to slow download of a piece from a peer with low transfer rate
  - Send requests of all sub-pieces to all peers.
  - Can cancel requests later for downloaded sub-pieces
  - Can waste bandwidth but end-game mode is short

Choking Algorithm

- Goal: Utilize all available resources
- Challenge: Freeloaders (peers who download but not upload)
- Solution: Tit-for-tat
  - Download from whoever than can but upload to via tit-for-tat strategy (i.e upload to peers which upload to them; choke others)
  - Results in connections actively transferring in both directions
  - Probe new peers for better transfer rates

Choking

- Temporary refusal to upload to a peer
- Peers always unchoke (i.e. upload to) a fixed number of peers (default is 4)
- Who to unchoke?
  - Upload to 3 peers that provide the best download rate; Revised periodically (say every 10 sec)
  - Optimistic unchoke: probe a random peer for better choices (rotate peer every 30 sec)



Anti-snubbing

- A peer is snubbed if choked by all peers it was downloading from
- Solution:
  - Snubbed peer stops uploading to its peers
  - Optimistic unchoking done more often
    - May discover a new peer that will upload to it