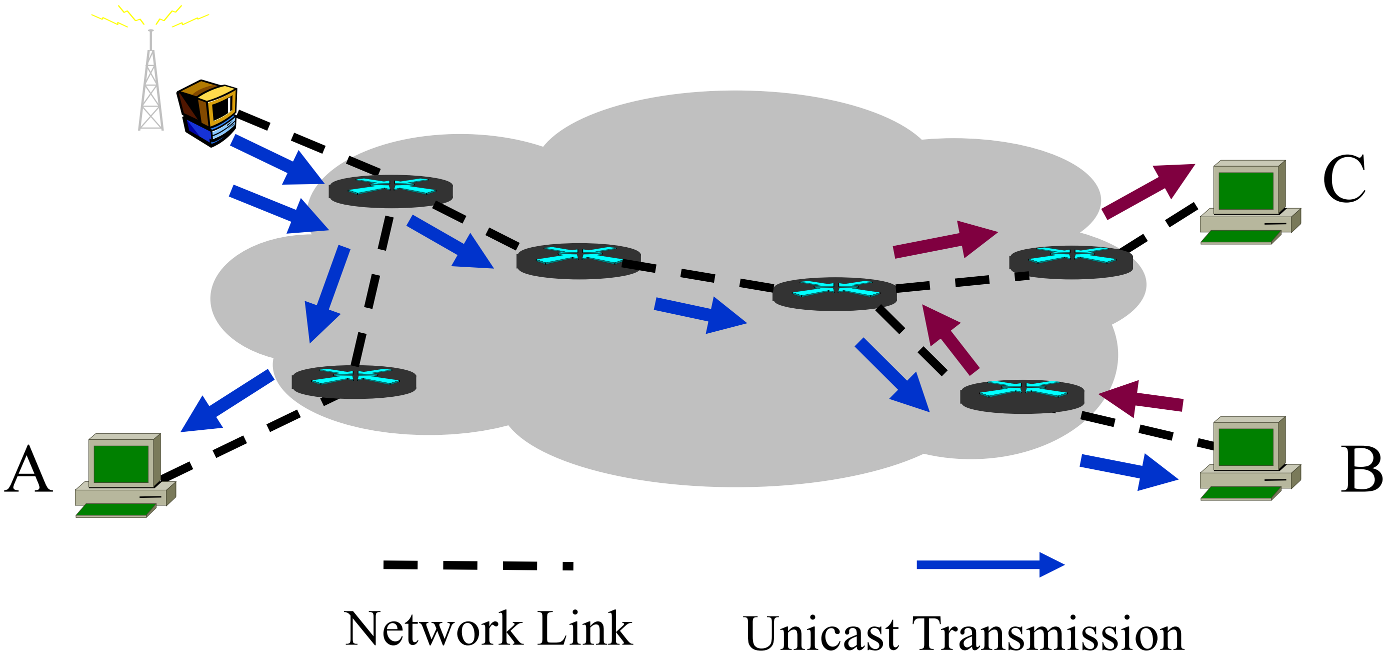
CSC 525:
Computer Networks

The Problem

- Download big files, possibly by many people.
  - Movies, software, scientific data
- Solutions
  - A single server
  - Many servers (CDN)
  - Multicast (IP or application layer)
  - Bittorrent (peer-to-peer)
- Bittorrent aims at the file transfer problem, not the search problem (e.g., DHT).
  - One study in 2004 said Bittorrent accounted for 1/3 of Internet traffic. It has been in decline in recent years.
  - Today most large file transfer/streaming use CDN.

Application Layer Multicast

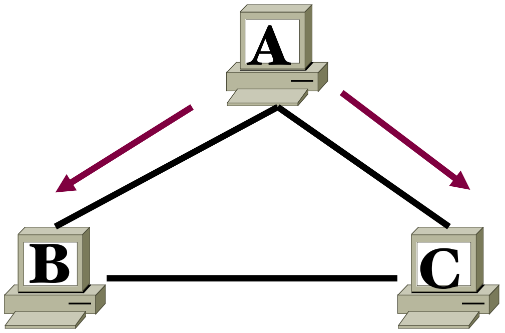
- Receivers share the load by sending data to other peers.
- However, leaf peers are not contributing, the file transfer is synchronous, and it doesn't work well with asymmetric links (upload << download).



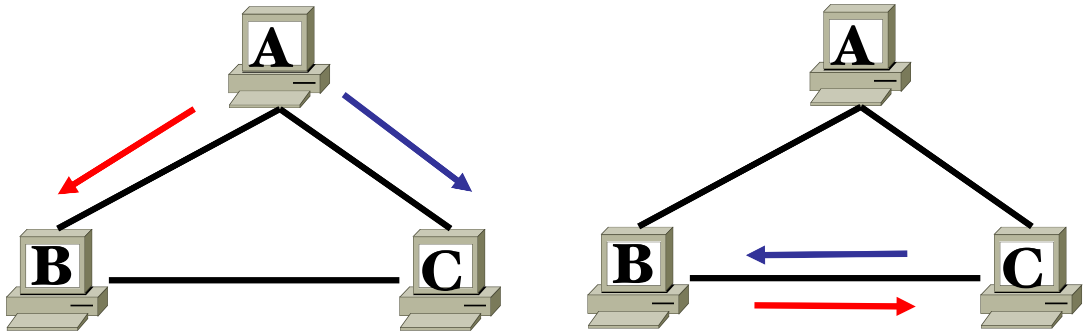
Multiple Dissemination Trees

- Multiple dissemination trees as a solution.
  - Assuming underlying path diversity.
- But how to build and manage these trees?

Example: File size 1MB, link bandwidth 0.5MB/s



Transfer the file in its entirety:
It takes A 2 seconds.



In the first 1s, A transfer the 1st half of file to B, and 2nd half of file to C.

In the next 1s, B and C exchange their parts. A is free to serve other users.

Two trees: the red one and the blue one.

Bittorrent

- Writtent by Bram Cohen (in Python) in 2001.
- The *swarming* approach
  - The lesson from the previous example is to enable concurrent downloading/uploading of different pieces of a file from/to different peers, and that is exactly what Bittorrent does.

  - In Bittorrent, a file is split into many pieces and uploaded to different peers. Peers are randomly connected and they download from each other the pieces they need.

  - whether to *explicitly* build and maintain dissemination trees is not critical.

Preparing a torrent

- Split the file into many *pieces*, usually 256KB each. Calculate the SHA-1 hash of each piece as its checksum.
- Pieces are further broken down to smaller *blocks*, which is the unit for requesting and transmitting.
- Set up a server called *tracker*, which is the entry point of the torrent.
  - Keep track of existing peers and some statistics.
  - Not involved in actual data transfer.
- Put all this information into a .torrent file and distribute it by web, email, etc.

Peers

- Seed: who has downloaded the entire file and remains in the swarm to serve others.
- Initial seed: the original content source.
- Leecher: who has not downloaded the entire file.

File Sharing

- A peer obtains the .torrent file out of band.
  - Usually from a web site.
- Contact the tracker
  - Get a list of randomly selected existing peers, usually around 50, depending on the current swarm size.
  - Will learn additional peers from the tracker later, default every 30 minutes.
- Contact peers to download missing pieces
  - Which piece? and which peer ?
  - Verify the hash after downloading an entire piece.

Choosing Pieces

- In the initial design: randomly chosen pieces.
- Later, *rarest-first*
  - Peers exchange with each other of which pieces they have.
  - Each peer, based on its own view, build a rarest-pieces set, and request the rarest pieces first.
- To guard against the case that original seeds are all gone before some peers finish downloading.

Choosing Peers

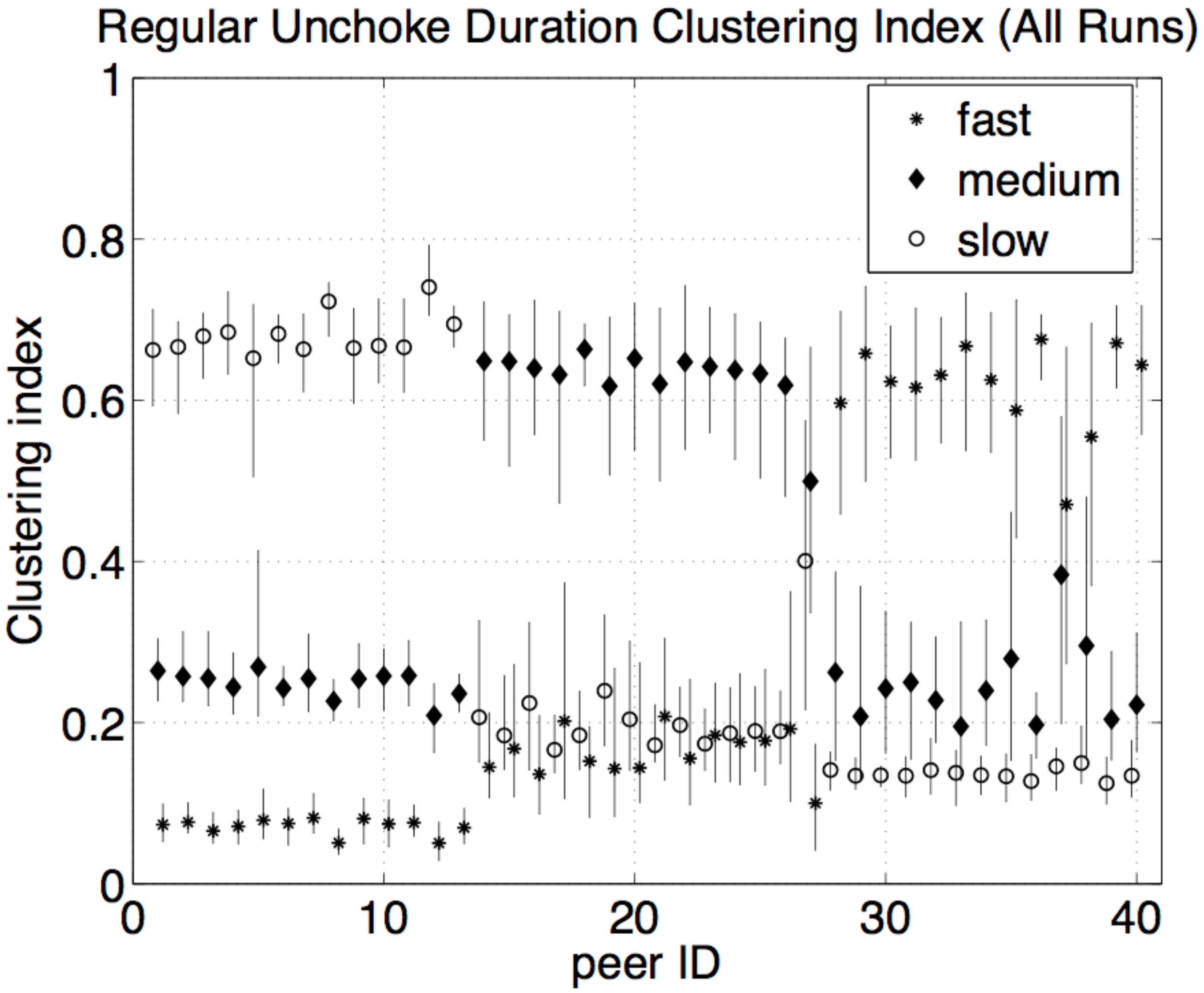
- Peer A is said to *choke* peer B if A decides not to upload to B. Otherwise it is called *unchoke*.
- Use the choke/unchoke algorithm to encourage uploading by peers.
  - Regular unchoke: a few peers with the largest upload rate to A.
  - Optimistic unchoke: a randomly selected peer
  - Done periodically, default every 10s.

The Dynamics of Bittorrents

- Run Bittorrent on Planet lab with controlled parameters
  - Around 40 peers.

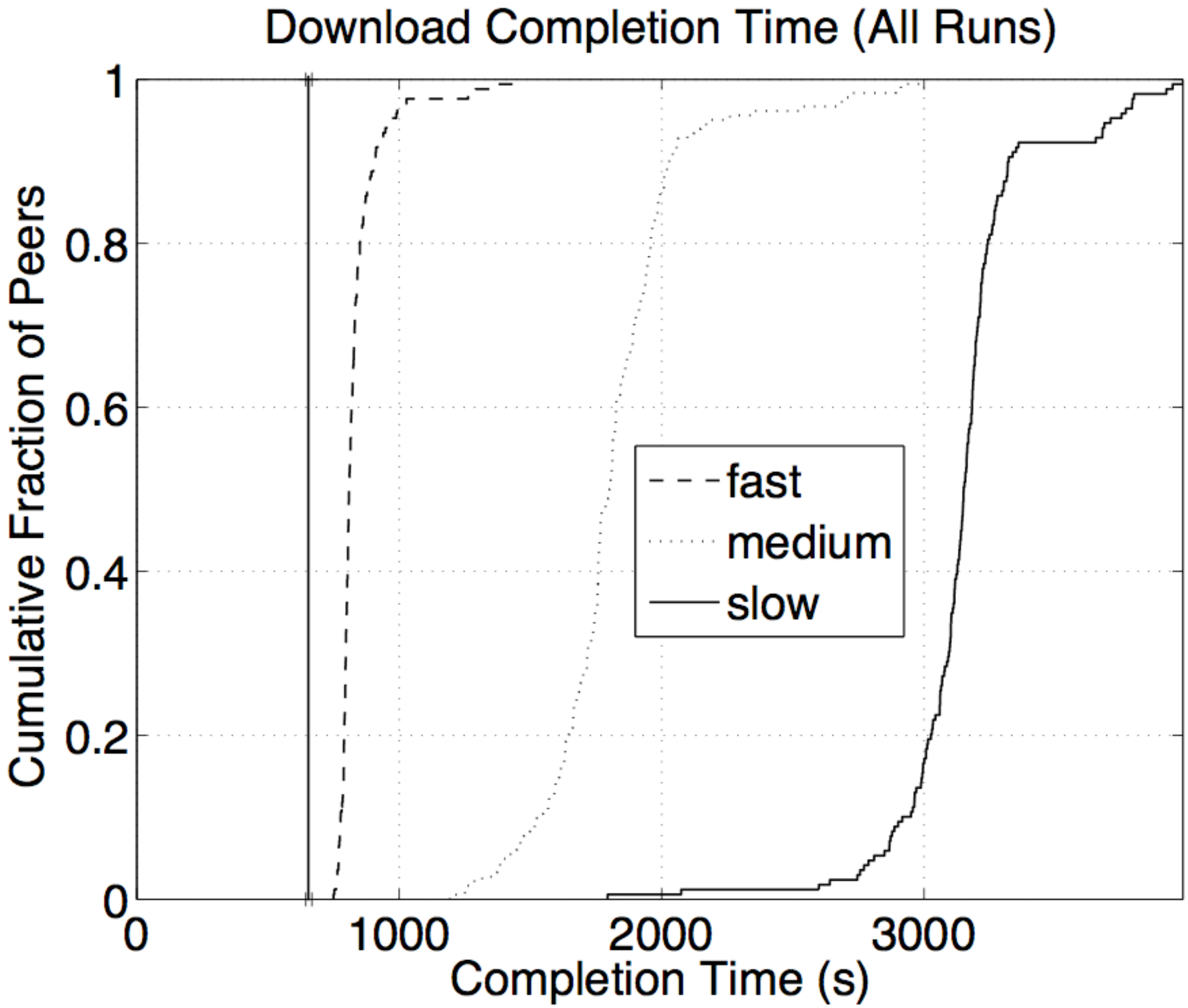
- Clustering
- Sharing incentives
- Upload utilization

Clustering



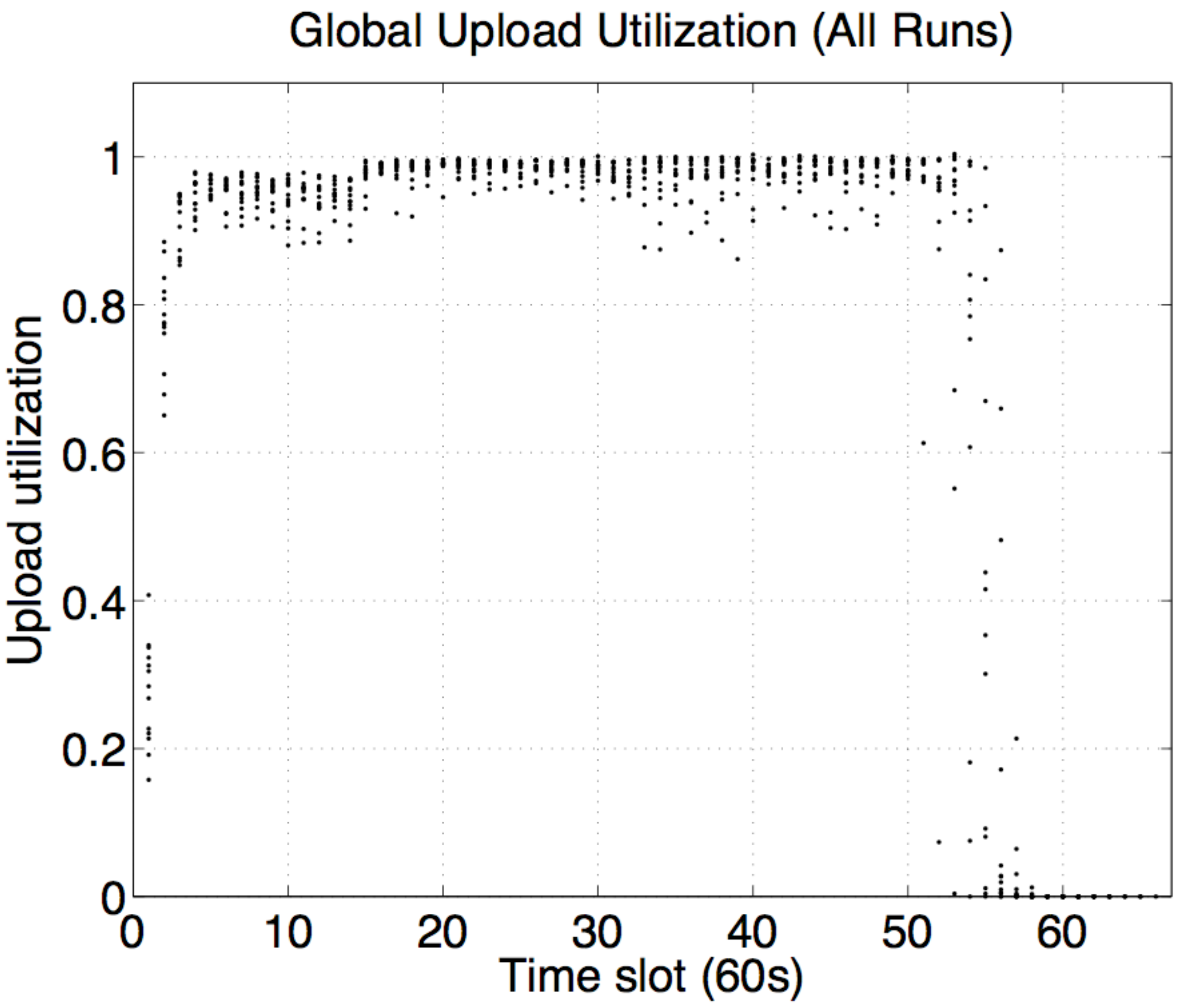
- One improvement is for tracker to match new leechers with peers of similar upload bandwidth.

Sharing Incentives



- Fast peers finish early, but also upload a lot more.

Upload Utilization



- Most of the time high utilization; at the beginning the utilization can be low.

Seed's unchoking strategy

- Old strategy: prefer peers with high download speed.
  - To quickly upload the entire file to the torrent.
  - But can be exploited by free-riders.
- New strategy: prefer peers with short unchoke times
  - To spread the upload to all peers for uniform service.
- An under-provisioned seed will reduce the clustering, sharing incentives, and upload utilization.

Cheating

- It is possible to cheat in bittorrent, i.e., downloading without uploading
  - Get a lot more peers from the tracker, so that the cheater can get more optimistic unchoke from other peers.
  - Download from seeds.
  - Other techniques ...
- The impact on the system performance is not well understood.
- The leech problem: a cold torrent.

Tracker

- The only centralized piece of the system
  - Single point of failure
- Multi-tracker design
- Trackerless design
  - Use DHT to locate peers of the interested torrent.

Use Bittorrent for streaming

- The original bittorrent was not for streaming.
- Need changes to make it work better:
  - Which piece to request? Must consider playback timing.
  - How to prepare the torrent? Must consider when the content will be available, probably need to contact the tracker periodically for information about the new pieces.