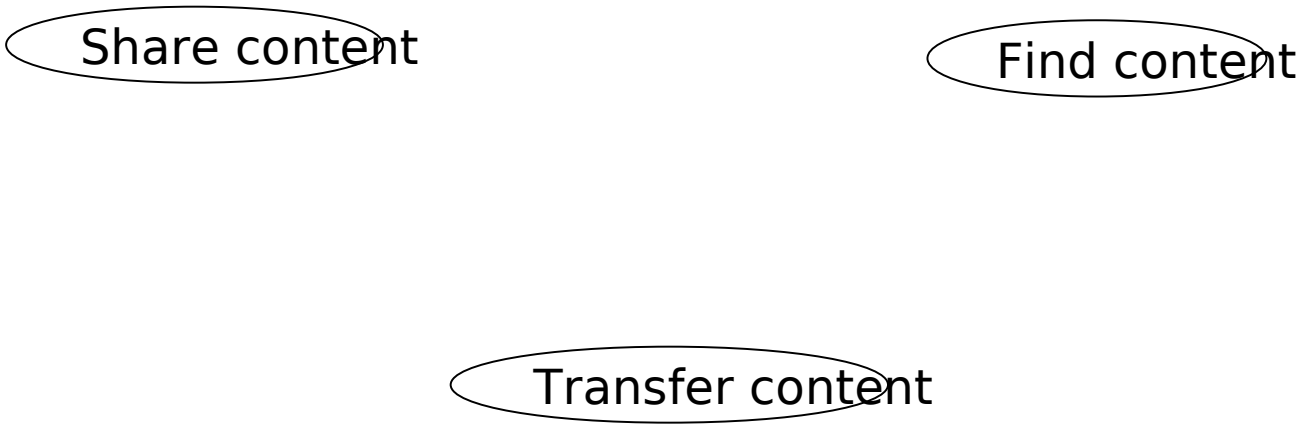
P2P content distribution

T-110.7100 Applications and Services in Internet, Fall 2009

Jukka K. Nurminen

Steps of content sharing



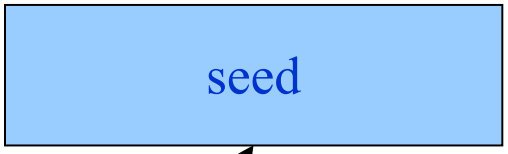
BitTorrent – content downloading

- Efficient content distribution
- Bram Cohen, 2001
- Key idea: you can receive faster than what your peer is able to send
  - Peer serving multiple users
  - Asynchronous connections
  - E2E speed of Internet
- File divided into pieces, recipient receives pieces from multiple peers
- Each recipient supplies pieces of the data to newer recipients

BitTorrent – components

Maintaining information about which peers have the content available

Normal website – hosting of metadata files (torrent-files)

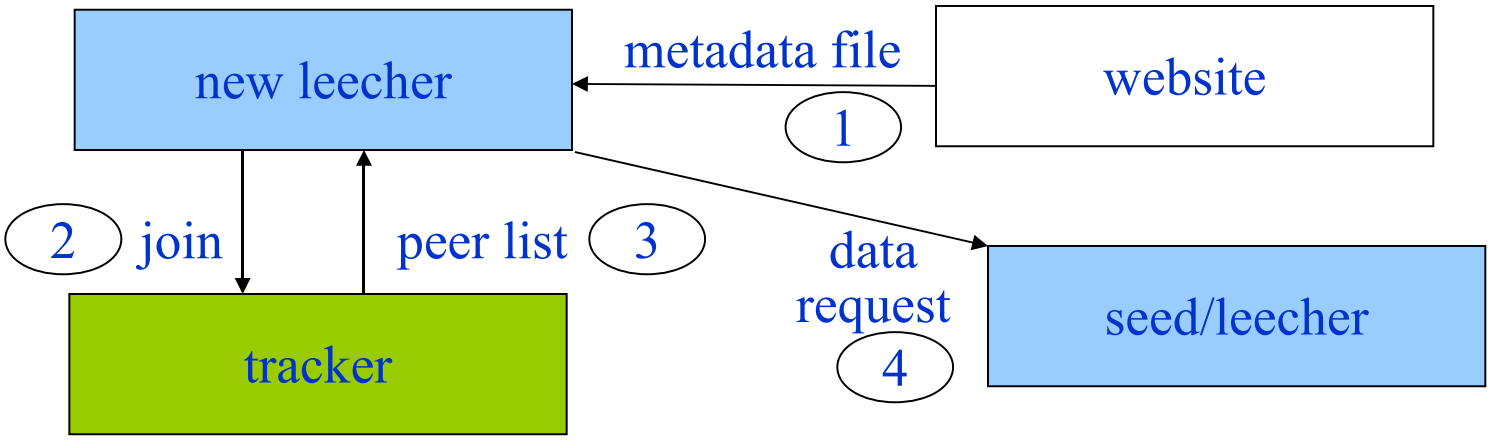


Peer with entire file

Peer that is still downloading (has only parts of the file)

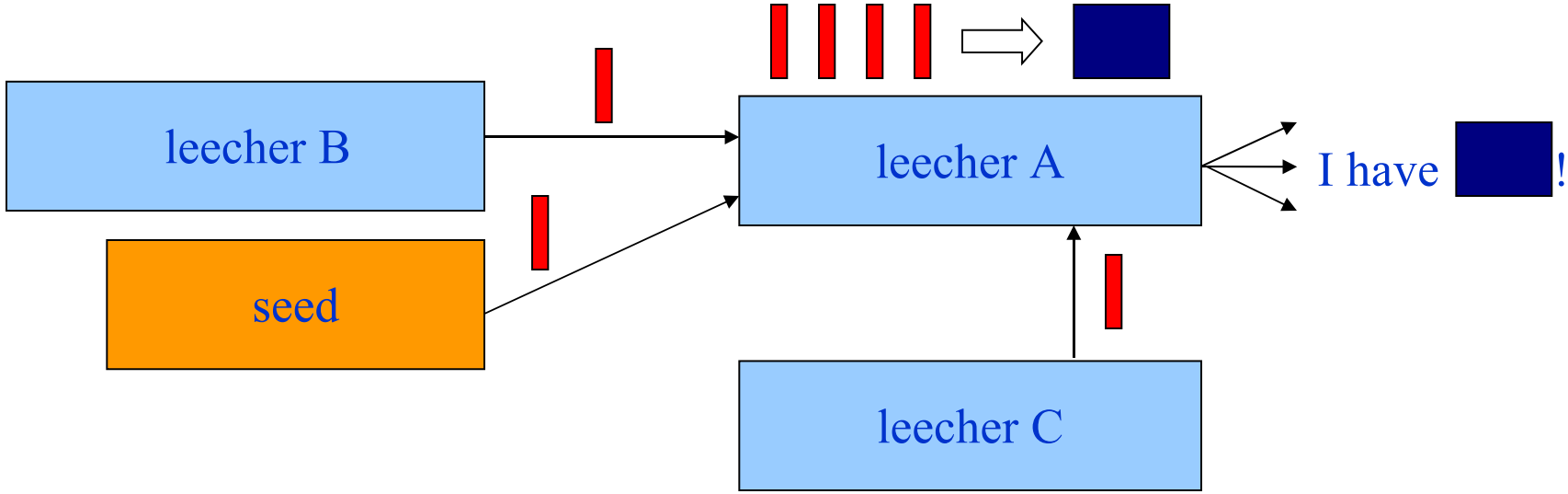
BitTorrent – joining a torrent

Adapted from Nikitas Liogkas, Robert Nelson, Eddie Kohler, Lixia Zhang, "Exploiting BitTorrent For Fun," University of California, Los Angeles



1. obtain the *metadata file (.torrent -file)*
2. contact the *tracker*
3. obtain a *peer list* (contains seeds & leechers)
4. contact peers from that list for data

BitTorrent – exchanging data



- Download sub-pieces *in parallel*
- Verify *pieces* using hashes
- *Advertise* received pieces to the entire peer list
- Look for the *rarest* pieces

BitTorrent Summary

- Benefits
  - reduced cost and burden on any given individual source
  - much higher redundancy
  - greater resistance to abuse or "flash crowds“
  - less dependence on the original distributor
- Disadvantages
  - Slow start and finish
    - downloads take time to rise to full speed because peer connections take time to establish
    - Special end game algorithms
  - Full content has to be downloaded before playing can start (in most cases)
  - Central tracker can be a bottleneck
    - Distributed trackers based on DHT
- Applications
  - Legal video distribution (e.g. BitTorrent, Vuze)
  - Illegal video distribution (e.g. PirateBay)
  - Distribution of patches (e.g. Wow, Linux distros)

P2P streaming

Traditional stream delivery models

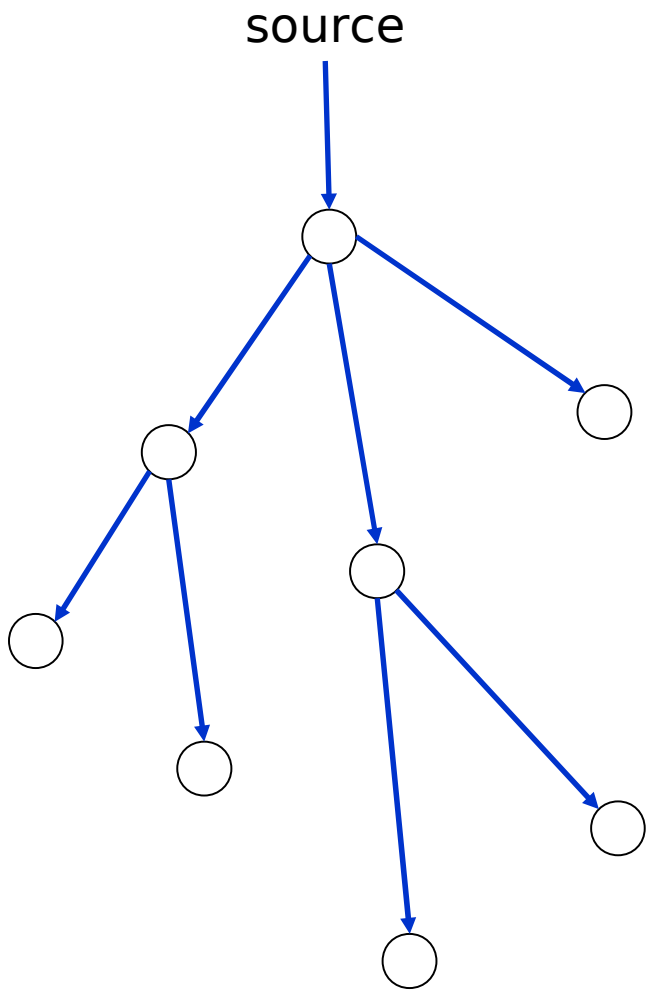
- Server
  - Widely used, simple and easy
  - Free Internet radios, YouTube, Liveleak.com, Google video, …
  - Allows using standard clients (browser)
  - Limited server output capacity / stream quality; expensive to scale
- Server grid
  - Content delivery network
  - Expensive to scale
- IP multicast / LAN multicast
  - The “ideal” model proposed for 20+ years
  - Not available in large scale Internet
    - Technical + non-technical constraints
  - Perhaps possible in local environments

P2P streaming (“peercasting”)

- Each receiver of the stream forwards it to other receivers
- Promises
  - No servers required
  - “Infinite” scalability
- Challenges
  - Churn: peers constantly join and leave the network
  - Limited peer capabilities: asymmetric data connections
  - Limited peer visibility: NAT, firewall
  - Optimal use of network resources

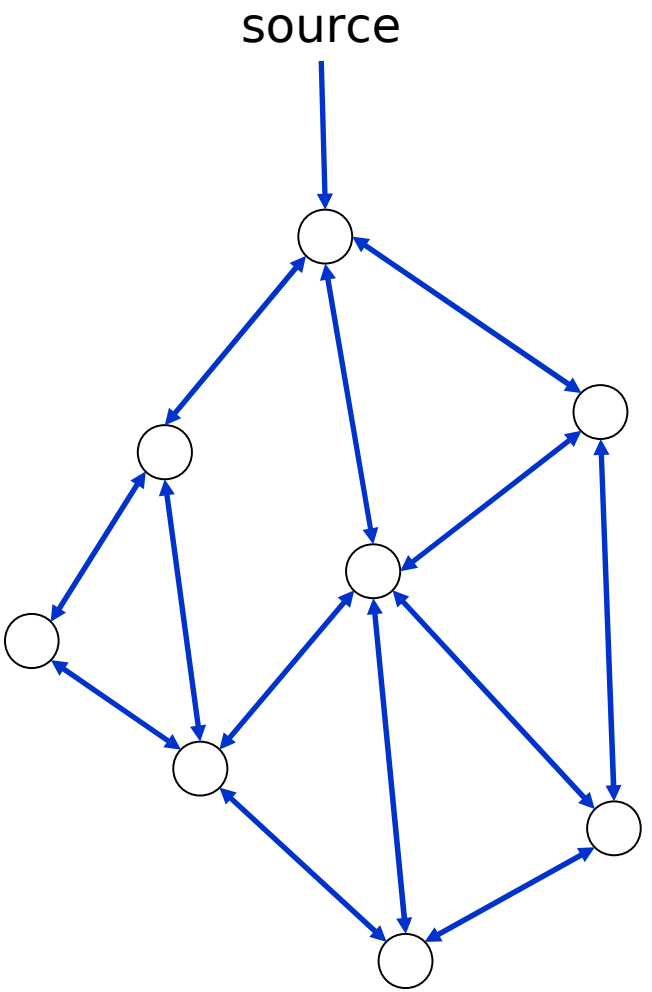
Multicast tree (ca. 2002)

- First practical approach
  - End-System Multicast II
  - Open source solutions (peercast, freecast)
  - Over 20 well-known variants
- Peers form a tree topology
  - Own tree for each data stream
  - Forward stream down the tree
- Works in practice
  - Scales 10…100…1000? users
- Problems
  - Large output bandwidth required
  - Tree optimization
  - Tree repair due to churn
  - Less than half of peers can contribute



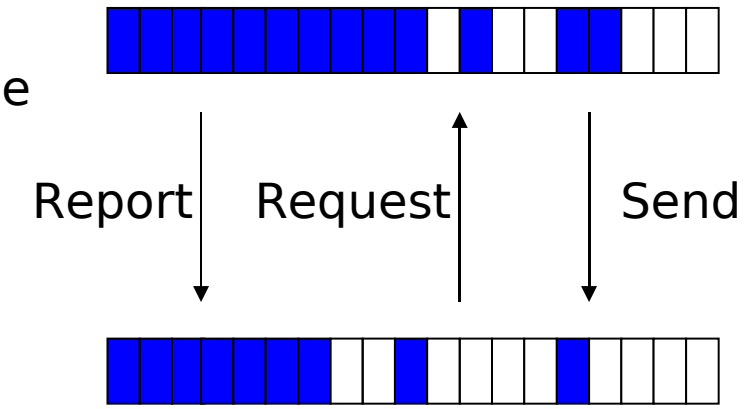
Data-driven overlay (ca. 2004)

- The mainstream practical approach
  - Active area for current research
  - Coolstreaming (2004),
    Chainsaw (2005),
    GridMedia (2006),
    PRIME (2006),
    HotStreaming (2007)
- BitTorrent for streams
  - Chunk stream in small pieces
  - Distribute pieces in a swarm
- Works well in practice
  - Most large-scale solutions
  - Coolstreaming, PPLive, Roxbeam, Sopcast ...
  - Scales to 10k ... 100k ... 1M?



Basic data-driven overlay approach

- Coolstreaming/DONet (2004), Chainsaw (2005)
- Topology creation: gossiping protocol (SCAMP)
  - Peers maintain random partial view of the network
  - Peers select random *partners*
  - No centralized tracker
- Swarming: sliding buffer of pieces
  - Reports pieces it has to its partners
  - Partners request for pieces they don’t have
- Design problems
  - Whom to select as partner?
  - When and from whom to request a piece?
  - Overhead vs. latency?



Main challenges of data-driven approach

- Open research questions
  - Based on real-life experiences with Coolstreaming and 80k users
  - Affect negatively to end-user experience
- Dealing with flash crowd
  - How to cope if number of users increases from 1k to 100k in 10 minutes?
  - We don’t have infrastructure to support new users
  - Joining takes a long time
  - > 25% of new users must re-try joining
- Dealing with 50% of users that don’t contribute
  - Due to asymmetric connection, firewall, NAT, …
  - Where to get the missing output capacity?

Hybrid technology

- The best known technology for commercial large-scale streaming
  - Streaming to 100k … 1M users
  - Proposed practical solution to problems of data-driven overlay
  - Joost, future Coolstreaming
- A combination of P2P and server grid
  - Use P2P distribution in stable conditions
  - Use powerful servers to fill in missing output capacity
  - Servers support newcomers
  - Servers support users behind asymmetric connections
- For example
  - Joost is 1/3 P2P, 2/3 client-server