TDDD25
Distributed Systems

Peer-to-Peer Systems

Christoph Kessler

IDA
Linköping University
Sweden

LINKÖPING UNIVERSITY

PEER-TO-PEER SYSTEMS

1. Characteristics of Peer-to-Peer Systems
2. The Napster File System
3. BitTorrent

P2P Basics

Main characteristics of peer-to-peer systems:

- Each user contributes resources to the system.
- All the nodes have the same functional capabilities and responsibilities (although they may differ in the resources they contribute).
- Correct operation does not depend on the existence of any centrally administered system.

Key issues:

- Choice of strategy for
  - the placement of data and their replica across many hosts;
  - the access to data

  such that
  - workload of nodes and communication lines is balanced;
  - availability of data is provided.

Anonymity of providers and users is offered (at least to a certain degree).

Why Do We Need Peer-to-Peer?

- **Example**: music or video content sharing by end users
- If only particular servers, which are centrally managed, can provide services/data, then scalability is limited:
  - server capacity
  - network bandwidth provided to a server
- To avoid the scaling problem
  - Peer-to-peer systems use the data and computing resources available in the personal computers and workstations present on the Internet and other networks.
  - Instead of separately managed servers, services are provided by all these resources together.

Important!

- Availability of processes/computers in peer-to-peer systems is a problem!

  → Services cannot rely on guaranteed access to a host.

Peer-to-Peer Systems - History

Volunteer computing

- Offering idle CPU cycles for HPC applications
- Early pioneer: SETI@home (1999) https://setiathome.berkeley.edu/
  - Downloading and processing radioastronomy data packets as a useful screensaver application (at a time when electricity was cheap...)
- Later (early 2000s): ***Grid computing*** middleware, such as GLOBUS toolkit
  - More stable scenarios and resources; organizations as contributors
  - A predecessor of modern cloud computing

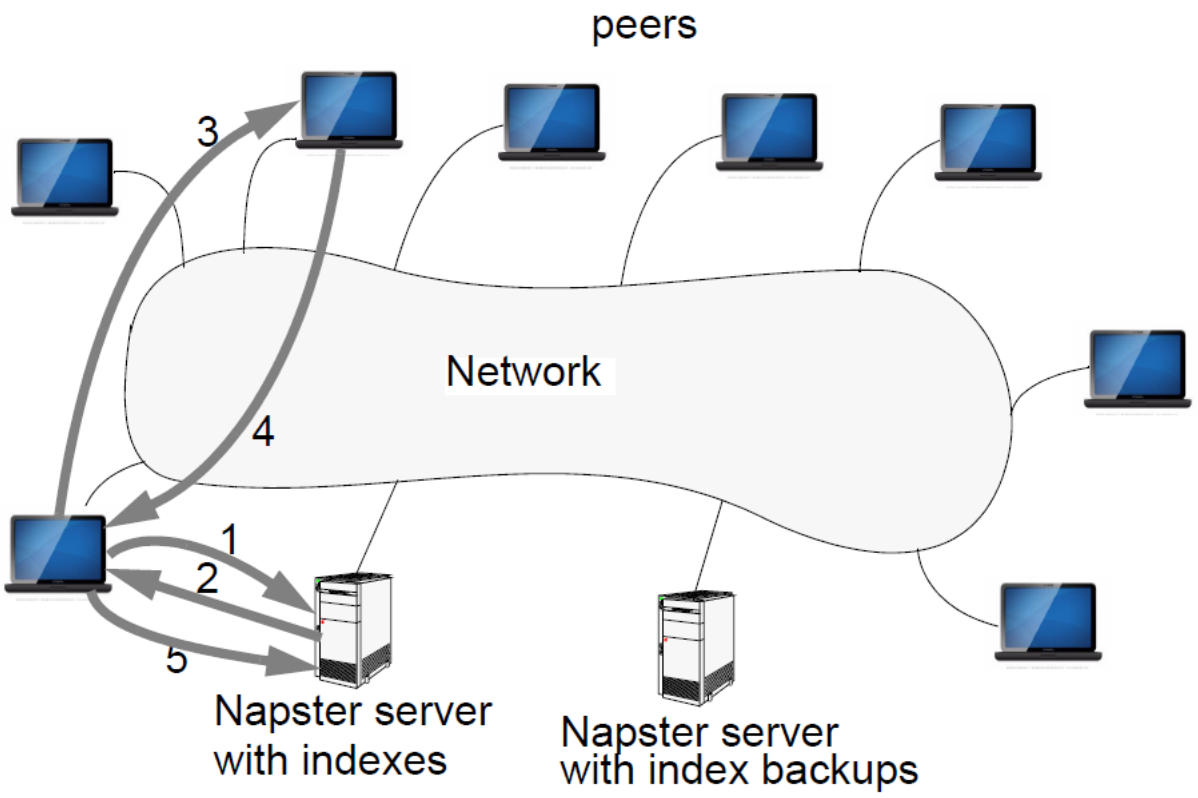
File/data sharing

- First Pioneer: Napster (1999)
  - The index is centralized!
- Later Systems: Freenet, Gnutella, Kazaa, BitTorrent
  - Only semi-centralized or completely distributed
  - Better anonymity, scalability, fault tolerance
- **Blockchain**: Secure, decentralised database
  - Growing list of ordered records e.g. about financial transactions
  - Bitcoin is implemented on top of Blockchain technology.

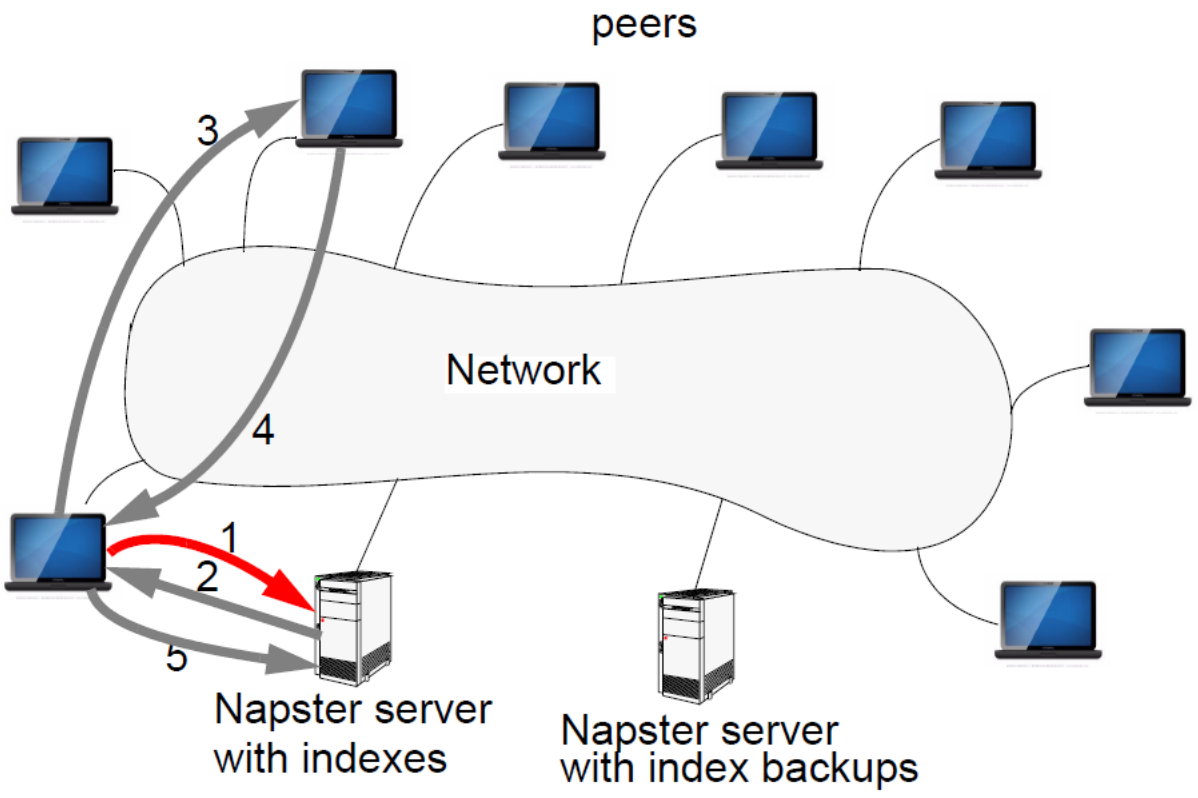
The Napster File Sharing System

- Napster provided a globally-scalable information storage and retrieval service for digital music (.mp3) files.
- Napster was the first to demonstrate the feasibility of a peer-to-peer solution on large scale.
- Napster, as an open service, was shut down July 2001, as result of lawsuits on copyright issues.

The Napster File Sharing System

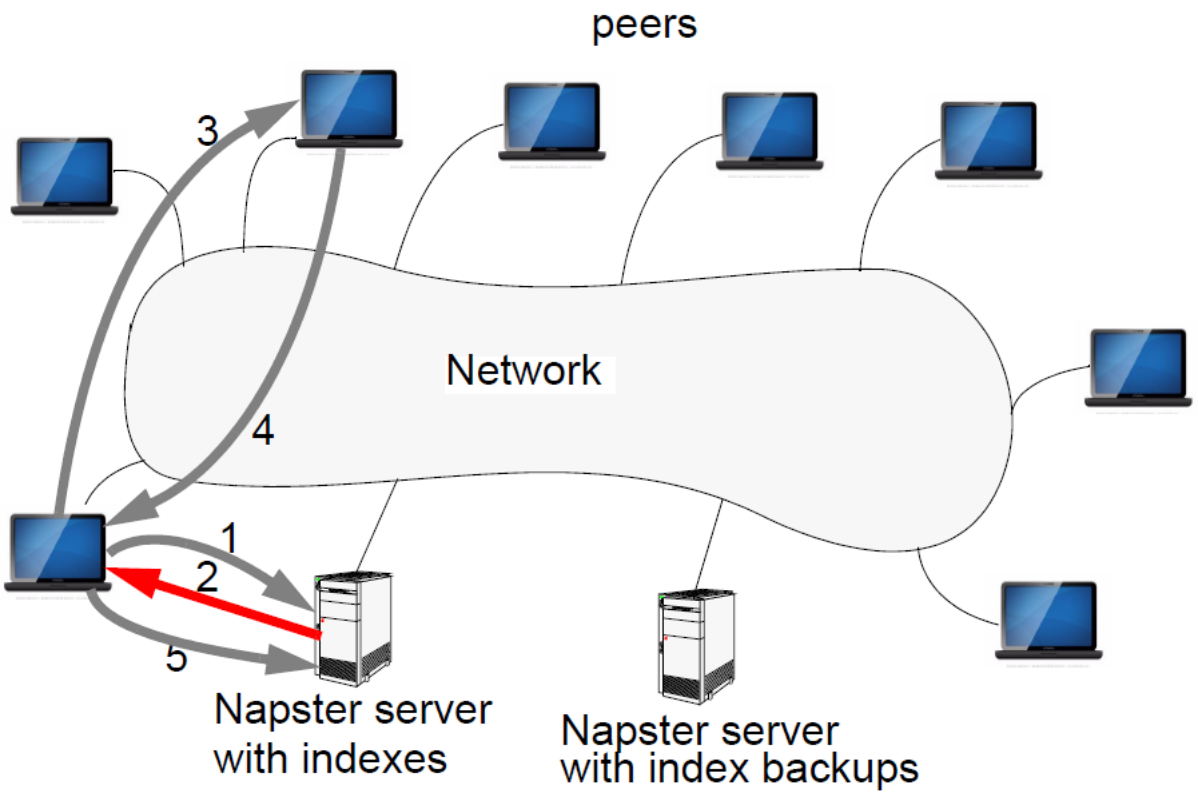


The Napster File Sharing System



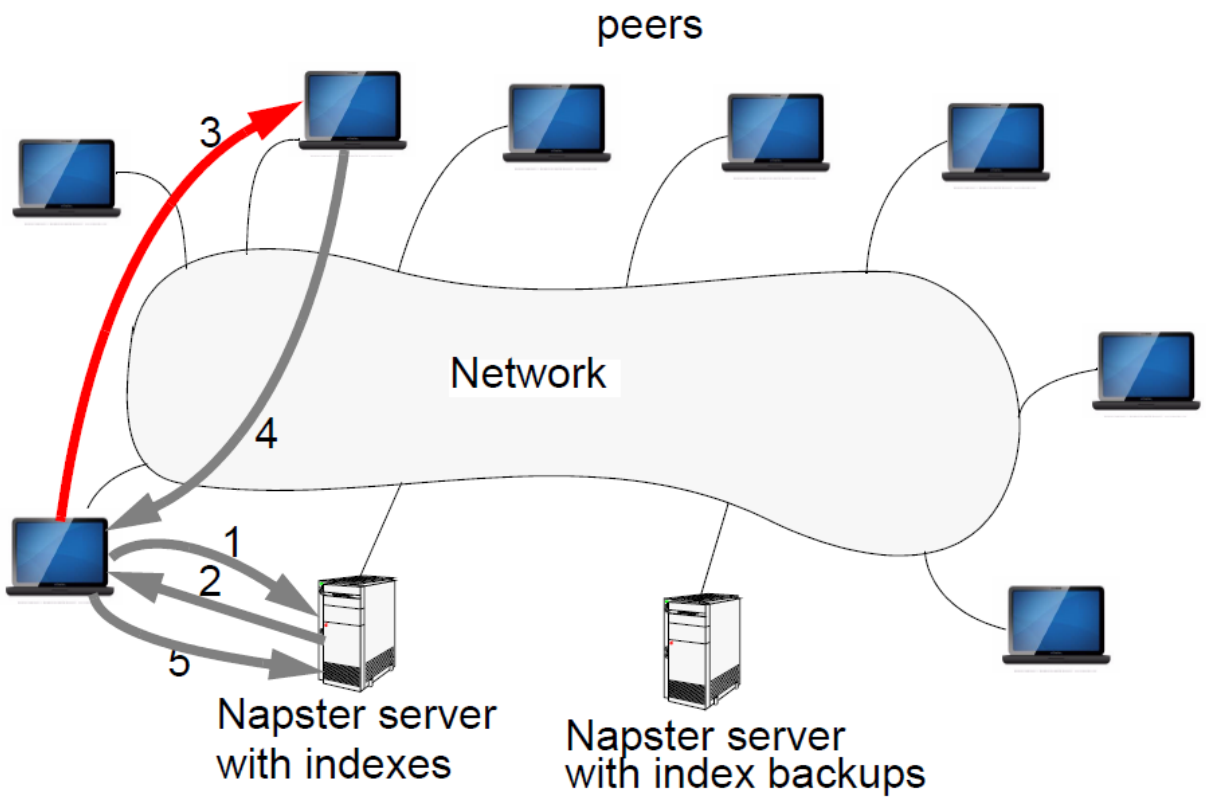
Step 1: File location request;

The Napster File Sharing System



Step 1: File location request;
Step 2: List of peers offering the files;

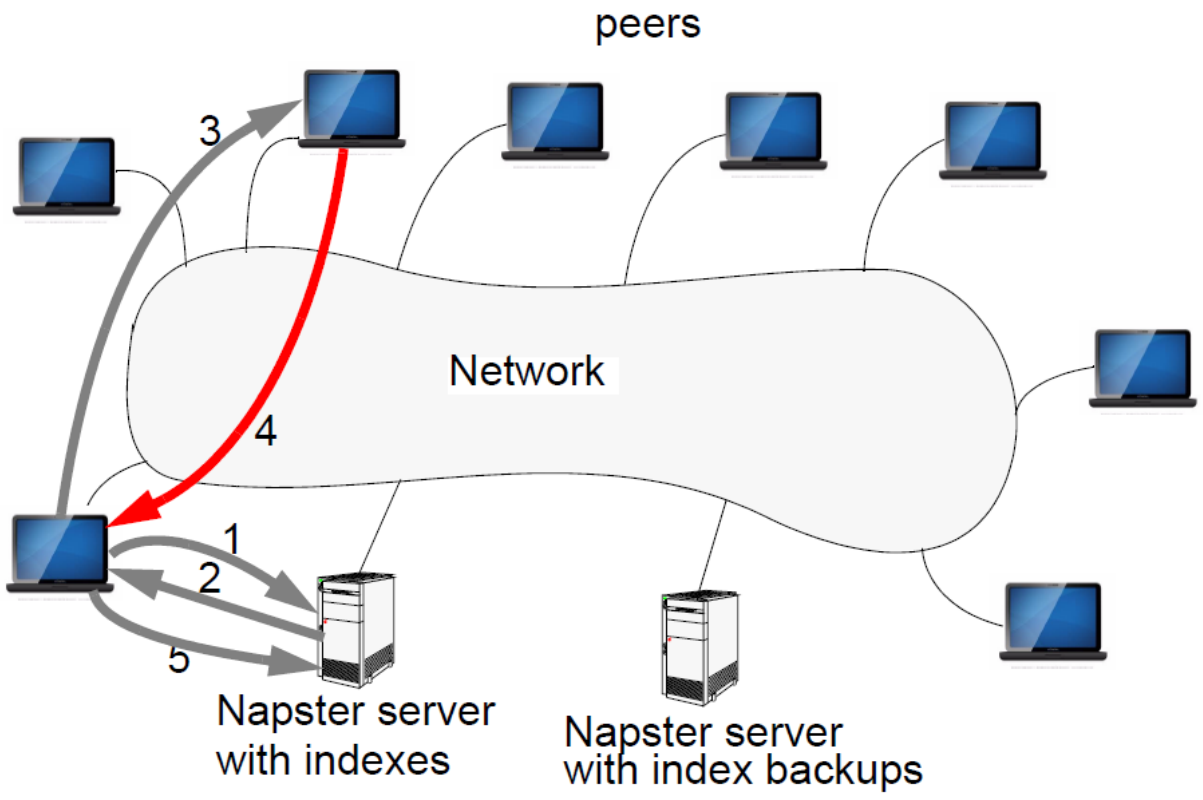
The Napster File Sharing System



Step 1: File location request;
Step 2: List of peers offering the files;
Step 3: File request;

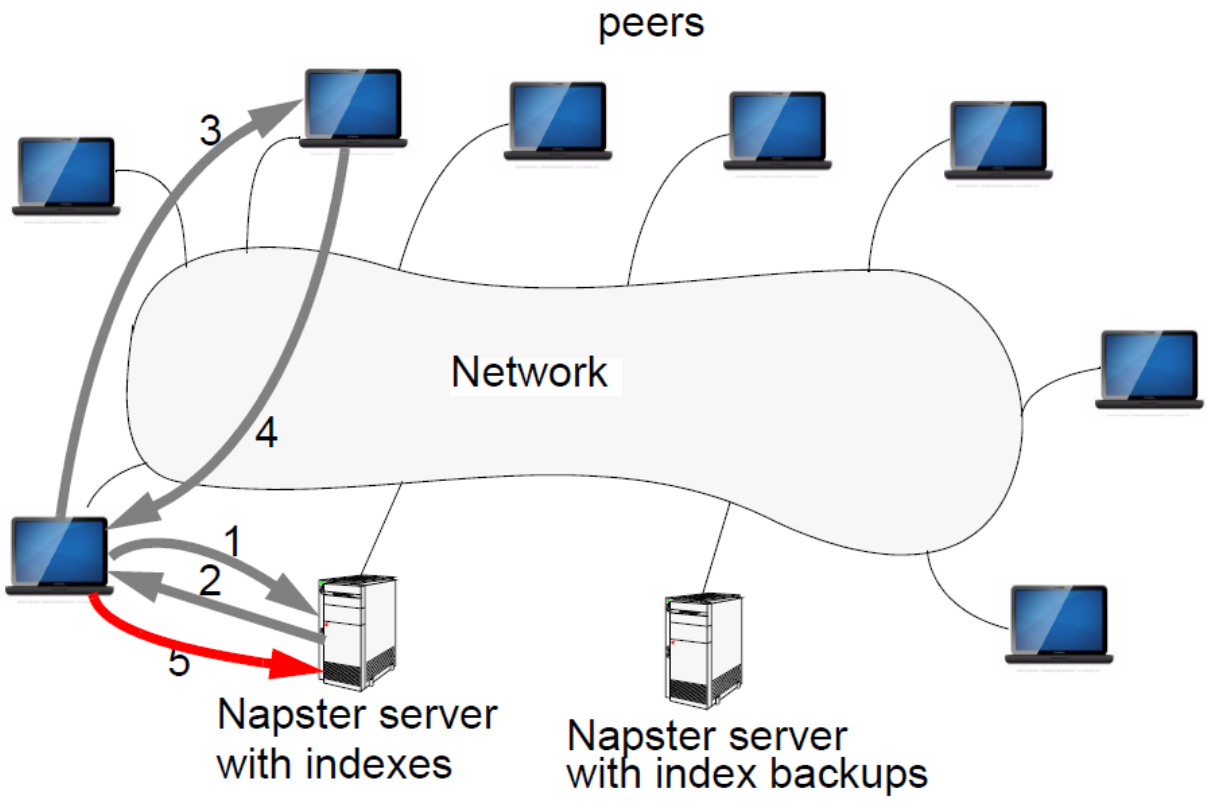
Possible extension in Step 3:
ping peers that offer the file, and request it from the one replying with best transfer rate

The Napster File Sharing System



Step 1: File location request;
Step 2: List of peers offering the files;
Step 3: File request;
Step 4: File loading;

The Napster File Sharing System



Step 1: File location request;
Step 2: List of peers offering the files;
Step 3: File request;
Step 4: File loading;
Step 5: Index update (user adds own files to pool of shared resources).

The Napster File Sharing System

- Napster uses a centralised index
  (with replicas for increased availability).

- The whole pool of files is distributed over the computers of the peers.

- In order to achieve load balancing:
  - When creating and sending the list of peers that offer the file (step 2), Napster takes into account **locality** (the distance between the client and the potential peers).

Problems with Napster

- Centralised index:
  - **Scaling** problem (server capacity and network bandwidth).
  - **Anonymity of operators** is not possible:
    legal responsibility for copyright issues can be put on operators maintaining the central index.
- A completely distributed index can provide better scaling and anonymity.
- Napster did not provide solutions for **consistency of replica updates** nor for **guaranteed availability**.

  This was no problem because of the particular application, music files:
  - Music files are *immutable* (do not change after being created)
    → no need to maintain replicas consistent.
  - If a file is unavailable at a certain moment,
    it can be downloaded later.
- Later systems, like BitTorrent, tried to solve some of the above problems by applying various specific ad-hoc solutions.

BitTorrent

- Similar to Napster, **BitTorrent** is a peer-to-peer file-sharing application
  - much more decentralized than Napster
  - Designed by Bram Cohen; first release 2001; several versions followed

- Main problems considered:

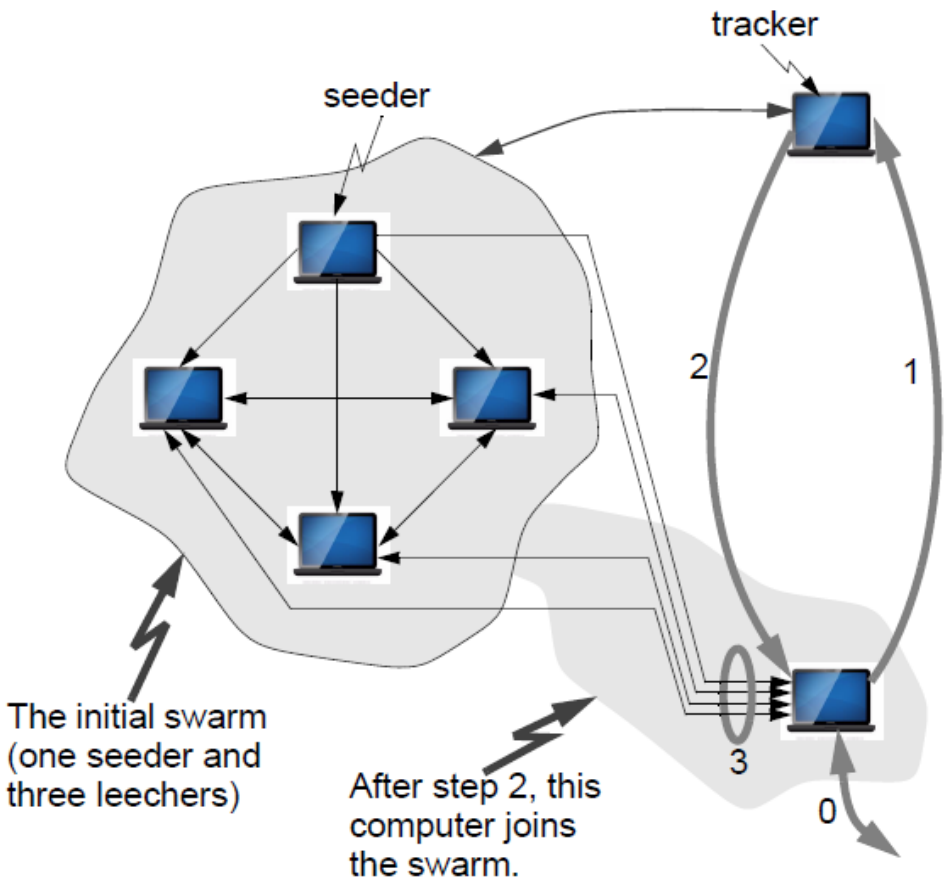
  - Files are **very large**;
    if the whole file is downloaded from a single peer
    → poor performance, processor overload, network congestion.

  → So, why not **divide the file into chunks**
    and download different chunks from different peers, **in *parallel***?

  - Centralised indexing creates problems with scalability and availability.

  → **Avoid the need for centralised indexing.**

BitTorrent



A **swarm** is composed of several computers interested in downloading/uploading a given file:

- **seeders**: have the complete file;
- **leechers**: have only a part of the file and are in the process to get the whole file.

- A swarm consists of, at least, one seeder
  - For a download of a file to operate, at least one seeder is needed to be available.

BitTorrent

Before being made available, a file to be distributed is broken into pieces (**chunks**); the chunk size can be between 64KB and 4MB.

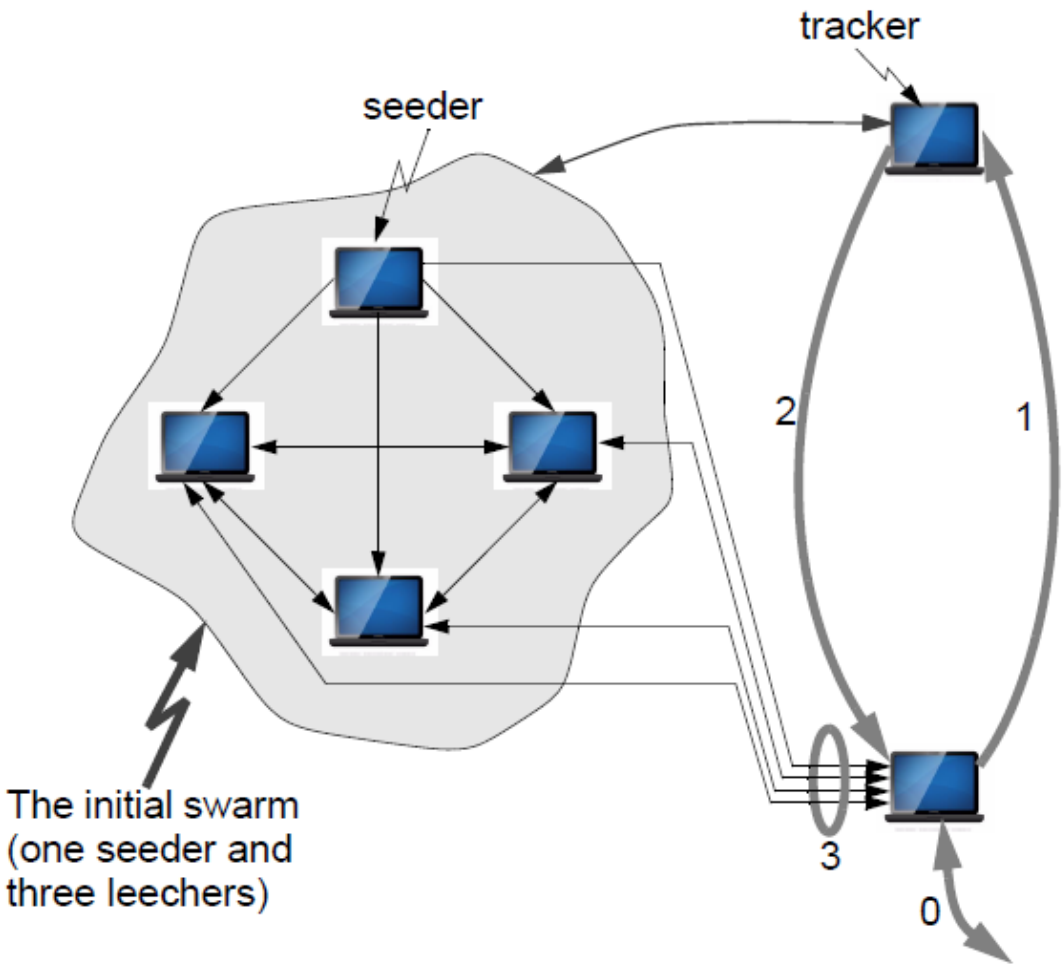
- BitTorrent downloads different chunks of the file simultaneously from multiple computers.
- The more computers in the swarm, the faster the download
  - BitTorrent is particularly useful for files that are large and ***popular*** (many simultaneous downloads).
- Chunks are received non-sequentially and rearranged into the correct order by the receiving client (based on information from the .torrent file*).

* The actual way to identify the .torrent file corresponding to the file one is interested in, is *not* part of the protocol: google, or go to specialised pages (e.g. PirateBay, but also many other less controversial ones).

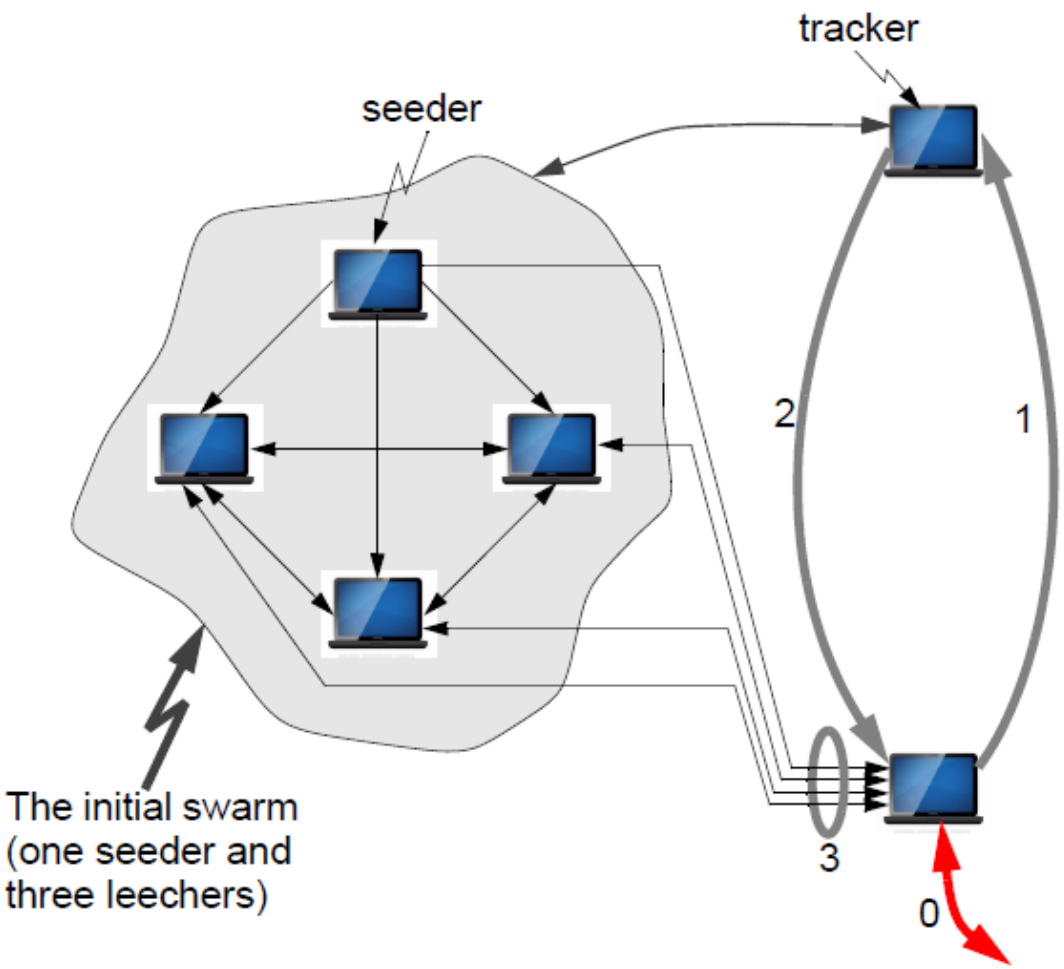
BitTorrent

- The **.torrent file** contains **metadata** needed for downloading a certain file:
  - name of the shared file,
  - file size,
  - chunk size,
  - checksum for each chunk (checked for integrity at download),
  - URL of the tracker.
- The .torrent file is created and made available by a user wanting to share a file.
- A download begins with identifying and downloading a .torrent file.

BitTorrent

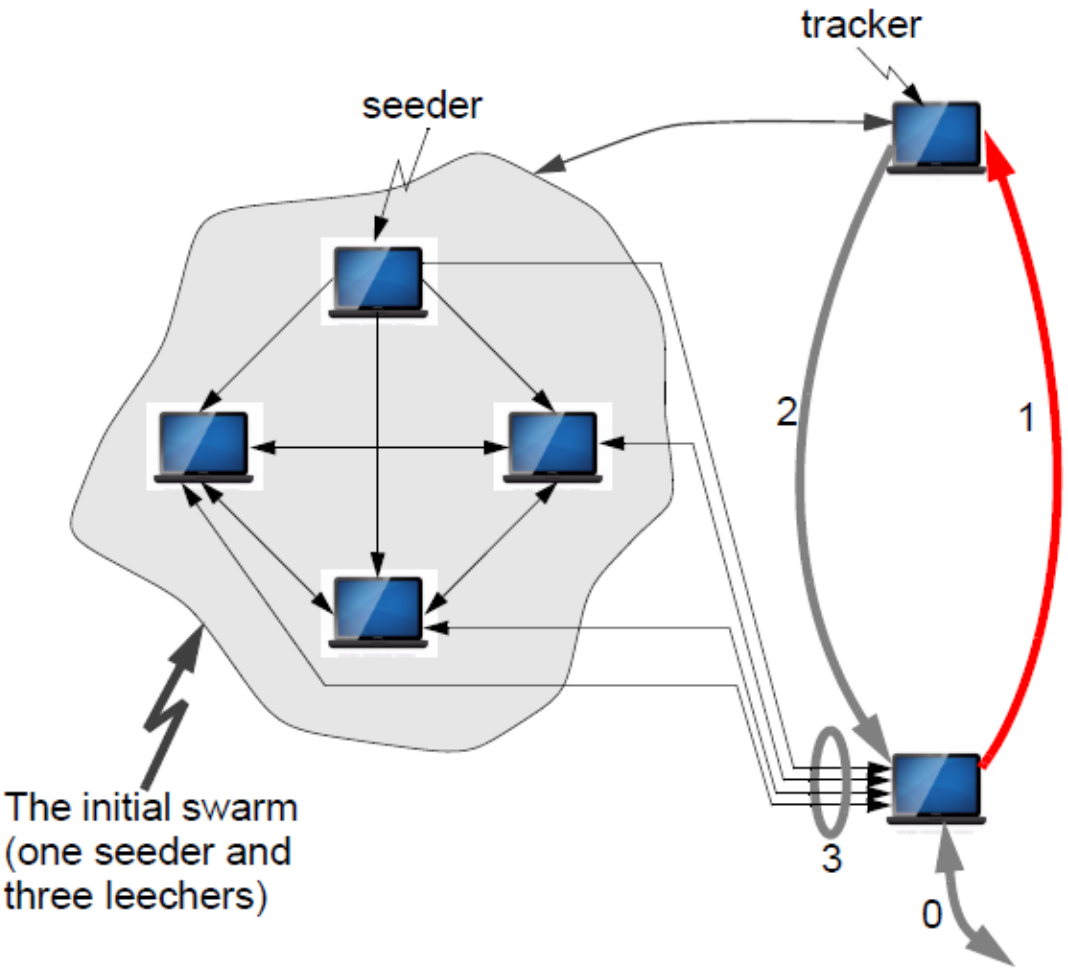

tracker
seeder
2
1
3
0
The initial swarm
(one seeder and
three leechers)

BitTorrent



Step 0: Search for the .torrent file and save it;

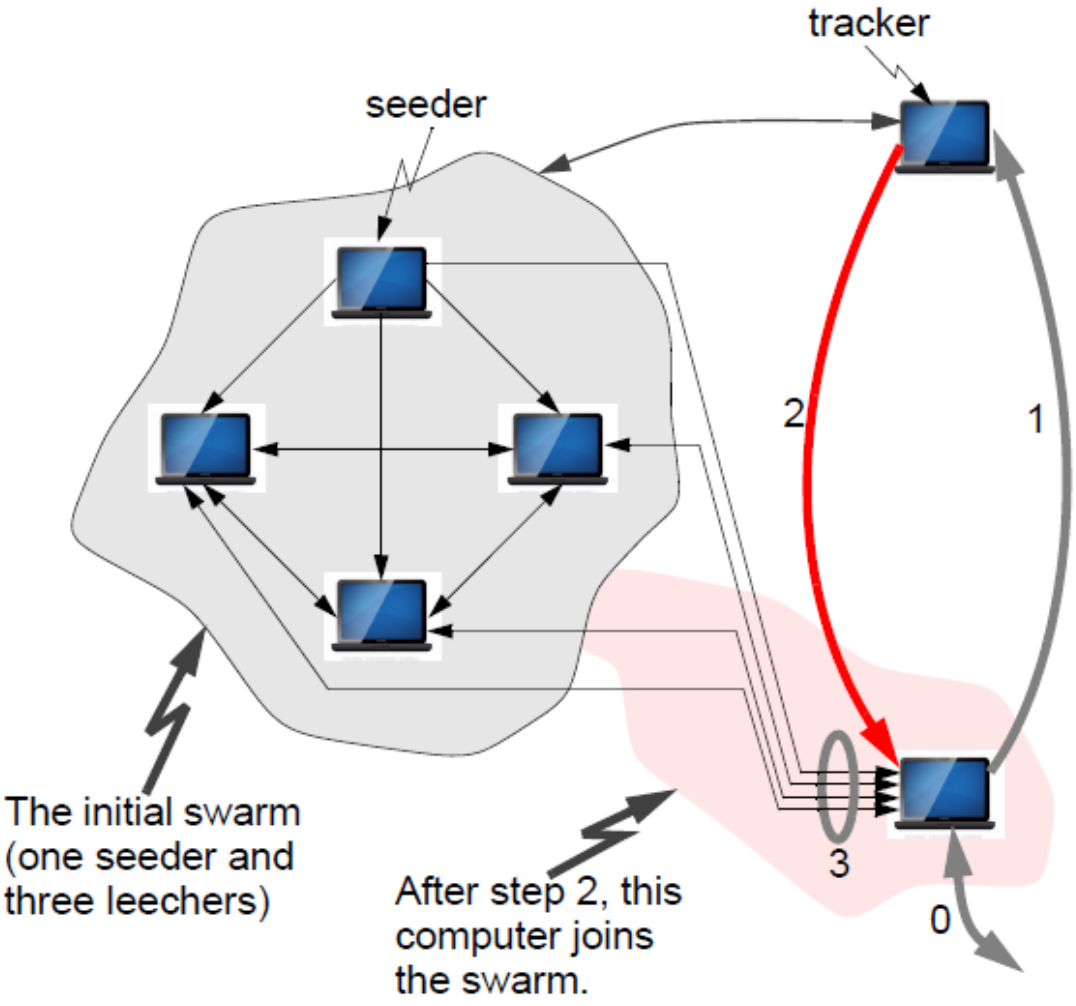
BitTorrent



Step 0: Search for the .torrent file and save it;

Step 1: The BitTorrent client on the computer contacts the *tracker* identified in the .torrent file;

BitTorrent

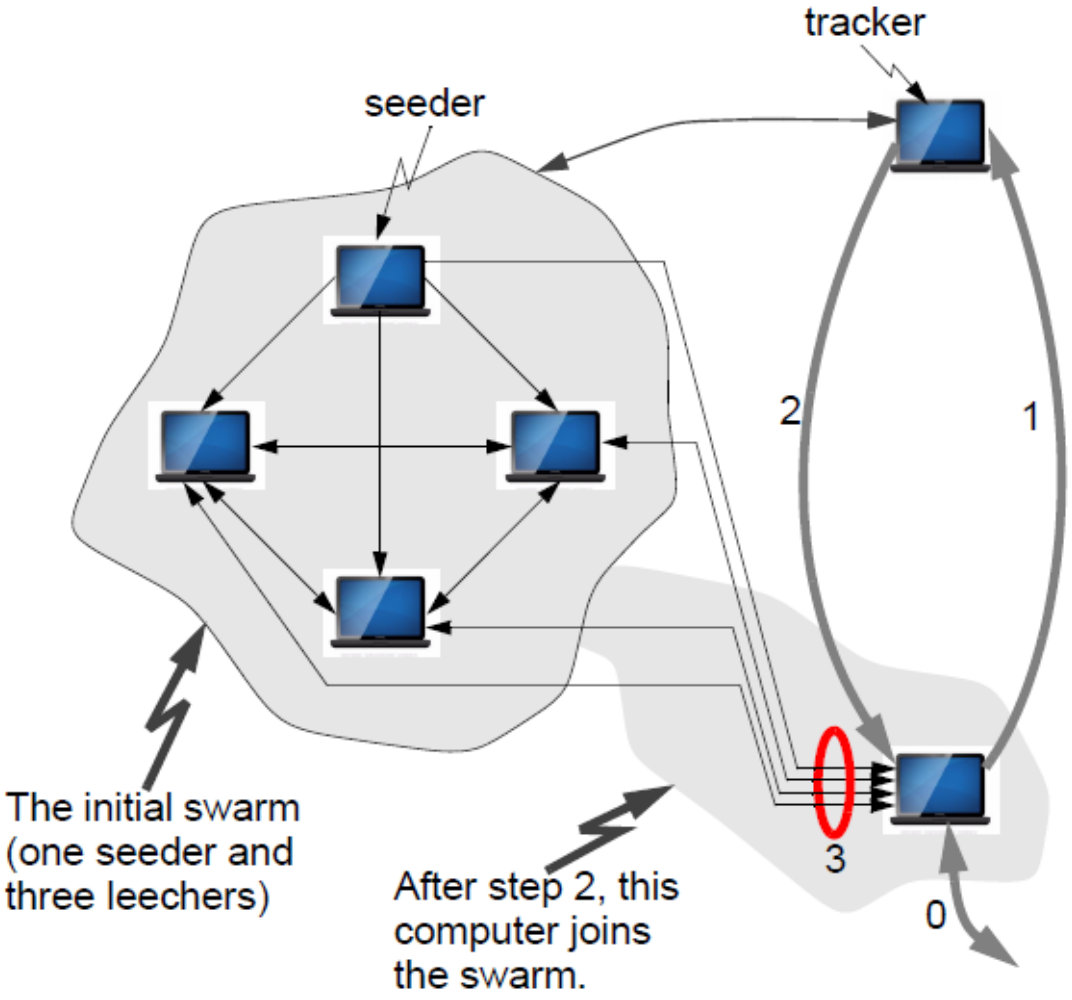


Step 0: Search for the .torrent file and save it;

Step 1: The BitTorrent client on the computer contacts the tracker identified in the .torrent file;

Step 2: The tracker identifies the corresponding *swarm* and helps the computer join it;

BitTorrent



Step 0: Search for the .torrent file and save it;

Step 1: The BitTorrent client on the computer contacts the tracker identified in the .torrent file;

Step 2: The tracker identifies the corresponding swarm and helps the computer join it;

Step 3: The computers in the swarm trade *pieces* of the file to be downloaded; the computer receives multiple pieces in parallel.

BitTorrent

The **tracker** is the computer in charge of managing the transfer of a file:

- The tracker's URL is extracted from the .torrent file
- It keeps track of the connected computers;
  it facilitates the computers in the swarm to connect to each other
  by sharing their IP addresses.
- NB the file is not downloaded from the tracker!
  The tracker *coordinates* the swarm.

BitTorrent

Tit-for-tat reward system

- The reward system tries to avoid peers only downloading but not contributing with uploading:
  - In order to receive files, you also have to give.
  - Clients reward other clients who upload, preferring to send data to clients who contribute more upload bandwidth
    → the more files you share with others,
    the *faster* your downloads are.
  - After you have got the whole file, you should continue to run the client
    → you stay as a potential seeder which others can use
    → your rates improve in the tit-for-tat system.

Napster vs. BitTorrent: Scaling, Availability, Developments

- **Napster**:
  - centralized indexing service – if it fails → 
  - the whole file is downloaded from the same peer – if it fails → 

  → Potentially reduced scalability and availability

- **BitTorrent**:
  - no indexing system (just need a .torrent file);
  - pieces of the file are downloaded (in parallel) from multiple seeders and leechers from the swarm.

  → Increased scalability, availability, performance.

BitTorrent: Scaling, Availability, Developments

A potential point of failure is the *tracker* supervising the swarm!

Two alternative **solutions** proposed in later BitTorrent versions:

- **A decentralized, trackerless torrent system**:
  - Clients communicate to each other without a central tracker.
  - A *distributed hash table* (*DHT*) technique is used, by which nodes identify other nodes to build the swarm.
    - See the Colouris book, Sec. 10.4-10.5 on DHT and overlay networks
  - The swarm is managed collectively by its members.
- **Multi-tracker implementations**:
  - Multiple trackers can be used for one torrent; they are specified in the .torrent file.

APPENDIX

Distributed Hash Tables

Overlay Networks

Distributed Hash Table (DHT) [Colouris p.176]

- P2P distributed system that implements the abstract data type **Dictionary**: *Keyspace* → *Valuespace*
  (e.g., the *index* for a file-sharing application)
  - Used like an ordinary hash table, but in a distributed manner
  - Entries are (*key*, *value*) pairs,
    each denoting a shared data object
    - **Keyspace** of limited size, e.g. 128-bit integers
      - Data identifiers, e.g. file names, are **hashed** to keys by a **hash function**
    - Values e.g. pairs [*file-name*, file contents]
      or [*file-name*, *locations (URLs) of file-holders*]

- The keyspace is **partitioned** and **distributed** across (possibly many) nodes in the distributed system
  - Each node is responsible for some unique **segment** of the keyspace
    - Possible: replication for improved availability and fault tolerance
  - DHT nodes (segments) are linked by an **overlay network** →

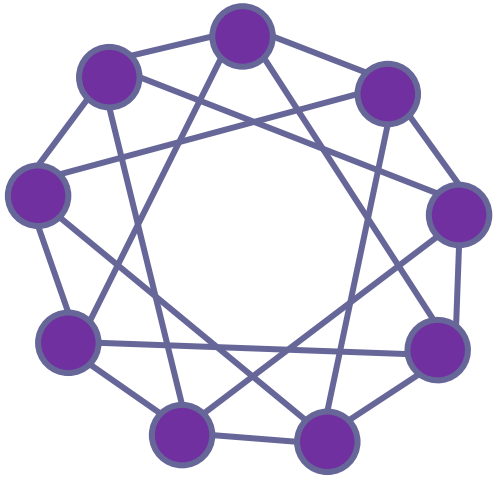
Overlay Networks

Overlay network = application/service - defined **virtual** network layered atop an underlying network, e.g. the internet

- No changes to the underlying network
  – just an additional top layer on the network stack
- Open and extensible structure, not bound to existing network standards
- Multiple overlay networks can coexist independently atop the same underlying network
- Usually dynamic → offers operations for **dynamic addition, lookup**, and **removal** of nodes

- **Possible uses:**
  - offer a special service not provided by the underlying network, e.g. file sharing or multimedia content distribution
  - customize operation or services of the underlying network, e.g.
    - more efficient operation, e.g. routing, caching
    - extend the underlying network by additional functionality, e.g. multicast communication, secure communication (e.g. VPN) etc.
- **Disadvantages**
  - Overlays introduce an extra level of indirection → may incur a performance penalty

Overlay Networks (cont.)



Realization

- Common **virtual topology**:
  Nodes linked to form a ***ring*** structure
  - circular linked list of nodes
    each holding part of the data
  - also: ***chordal ring***
    (ring with additional ”shortcuts” for faster search)
  - Each node only needs to know its direct neighbors in the overlay network.
- **Self-organizing**,
  nodes can dynamically join or leave the overlay network.
- **Searching** by following references to neighbor nodes in the overlay network from a known start point
  - a node either owns the key
    or has a link to a node that is closer to the owner
    = **key-based routing** → Colouris-book, Sect. 10.5
  - potentially many IP-hops required

Overlay Networks (cont.)

Examples of overlay networks:

- The **internet** itself was originally an overlay network over the telephone network.
- **Peer-to-peer file sharing** e.g. distributed tracker in BitTorrent
- **Distributed hash tables** ←
  - Addressing scheme based on a keyspace
  - Topology, e.g. ring: **Skype** for voice-over-IP communication (→ Colouris-book, Sect. 4.5.2)
- **Wireless ad-hoc networks** and disruption-tolerant networks in the context of mobile and ubiquitous computing (→ book, Ch. 19)
- **Multimedia content delivery** (→ book, Ch. 20)
- The *Interplanetary File System* (IPFS)

Acknowledgments

- Most of the slide contents is based on a previous version by Petru Eles, IDA, Linköping University.