Peer-to-Peer Computing: DHTs and Overlay Networks

Distributed Systems

Romaric Duvignau
https://www.cse.chalmers.se/~duvignau/
duvignau@chalmers.se

Today’s lecture

Unstructured Overlays / History

- Napster 1999
- Gnutella 2000
- BitTorrent 2001

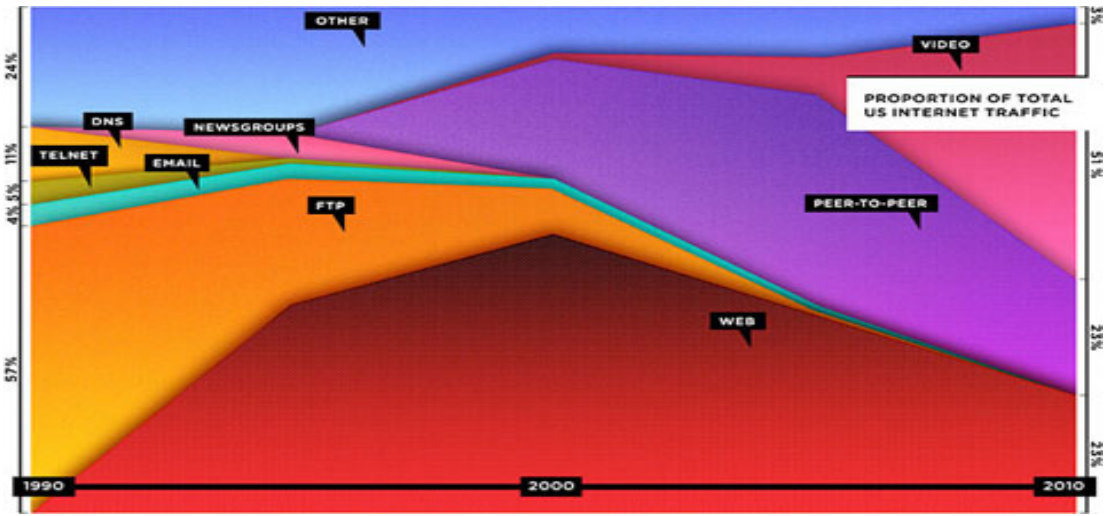
- **Structured Overlays**
  - How?
  - Chord
  - Sybil Attacks
- **Blockchains (briefly)**

Do you recognize this logo?

If yes, then you should recognize that one as well...

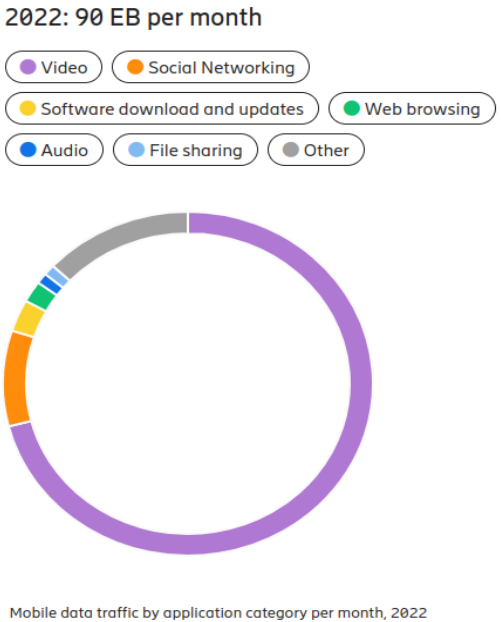
What is P2P?

- It is (used to be) every ISP's nightmare
  - In 2007, estimations between 50% to 70% of Internet traffic is P2P [1,2,3,4].



00-10 “10 years of glory” for P2P networks

https://www.ericsson.com/en/reports-and-papers/mobility-report/dataforecasts/traffic-by-application



Mobile data traffic by application category per month, 2022

What is P2P again?

- “A distributed network architecture may be called a **Peer-to-Peer (P-to-P, P2P)** network, if the participants share a part of their own hardware resources (processing power, storage capacity,network link capacity, printers).”
- “These *shared resources* are necessary to provide the **Service** and content offered by the network (e.g. file sharing or shared workspaces for collaboration).
- “They are peerible by other peers directly, without passing intermediary entities. The participants of such a network are thus resource (Service and content) **providers** as well as resource (Service and content) **requesters** (Servant-concept). “

What is a P2P Overlay?

- Distributed systems, **without** any hierarchical organization or **centralized control**.
- Peers form self-organizing overlay networks that are *overlayed* on the Internet Protocol (IP) networks.
  - They are virtual networks

Unstructured overlays

“the birth of P2P”: Napster

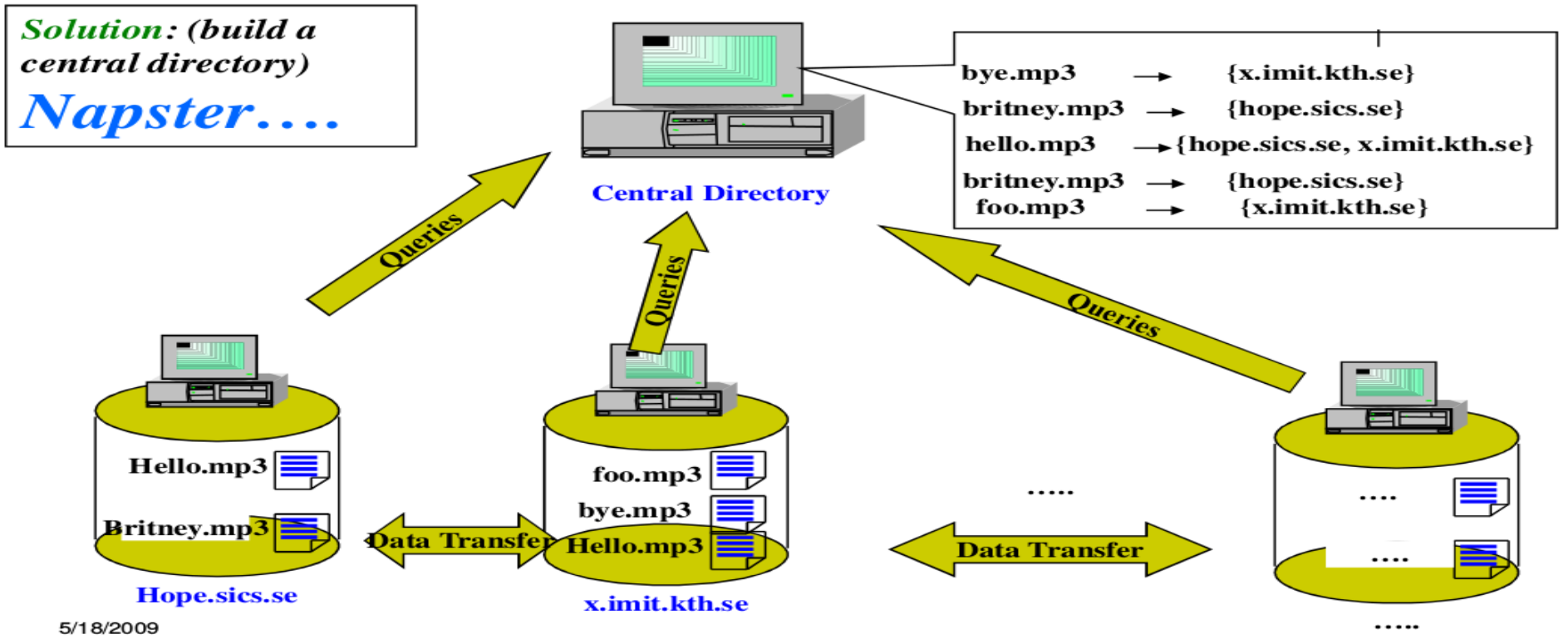


Figure Courtesy of Sameh El-Ansary's and Seif Haridi's course at SICS/KTH

History: Basic operations in Napster

- Join
  - Connect to the Napster name server
- Leave/Fail
  - Server detects you are not there any more
  - Data removed from directory
- Share (Publish/Insert)
  - Hi server, I have these files I want to share
- Search
  - Hi server, does anyone online have these files?
- Download
  - From peers

History: The end

FEATURES

Metallica vs. Napster: The lawsuit that redefined how we listen to music

MUST READ: Top programming languages: Most popular and fastest growing choices for developers

Dr. Dre sues Napster -- and users?

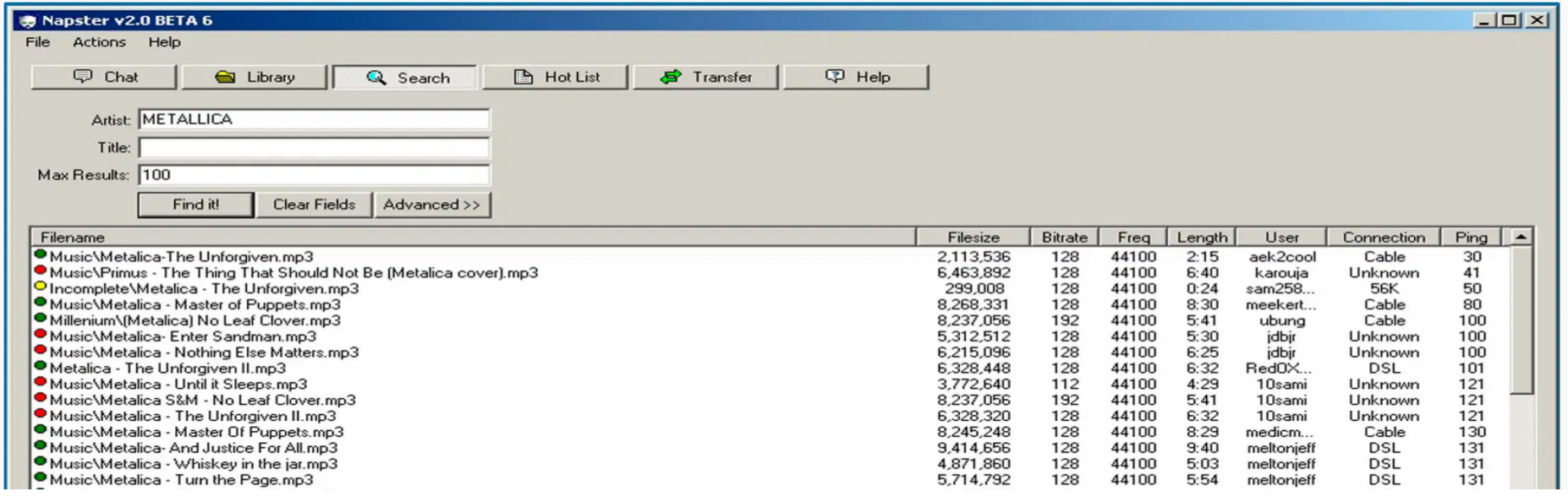
The lawsuit was expected, but the possibility that people who use the music exchange program might also be hauled

History: End of Napster & the birth of “real” P2P

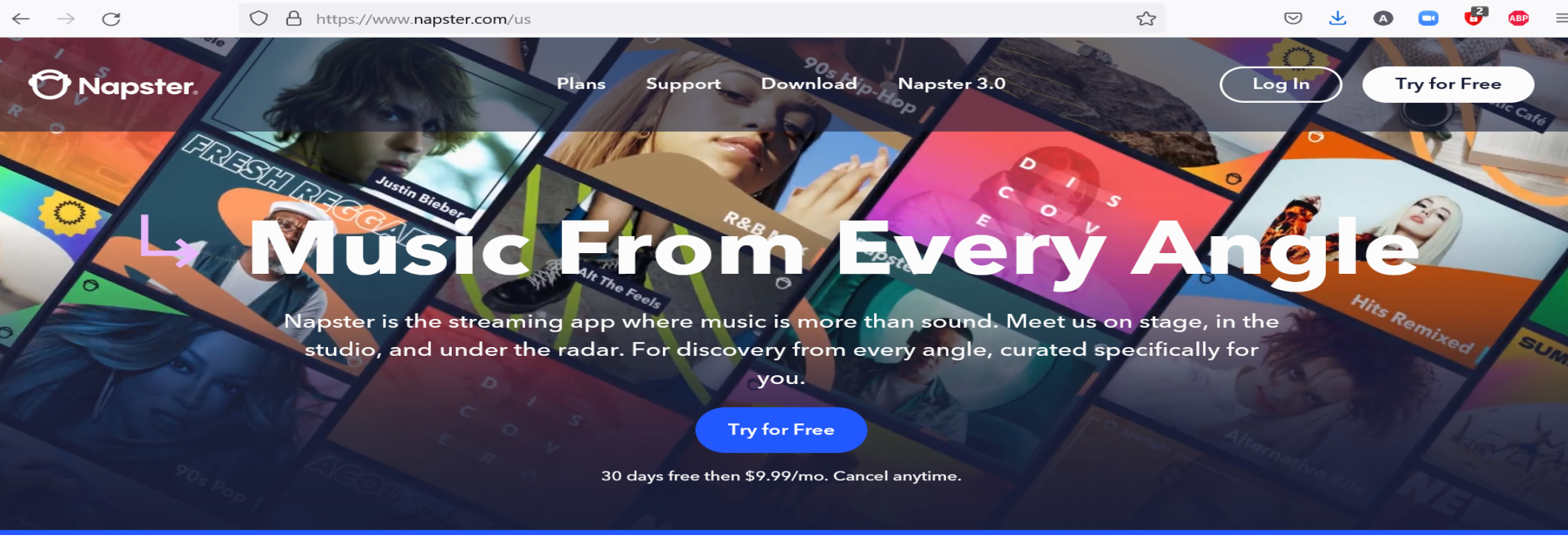
- Napster was brought down when Metallica and Dr. Dre filed a lawsuit for copyright infringement.
- Its name was sold at a bankruptcy auction for 70 M$
  - https://us.napster.com/
  - (Note: Napster was created for fun and never made money during its existence.)

KICKING AND STREAMING

Napster, yes that Napster, was just bought for $70 million



Yet another Spotify...



Reflections on Napster

- Why could it be sued / plugged off?
- Can we build something that is *un-sueable*?

Rise of the Unstructured overlays

History:Random (Unstructured) Overlay Networks Distributed Directory + Distributed Storage

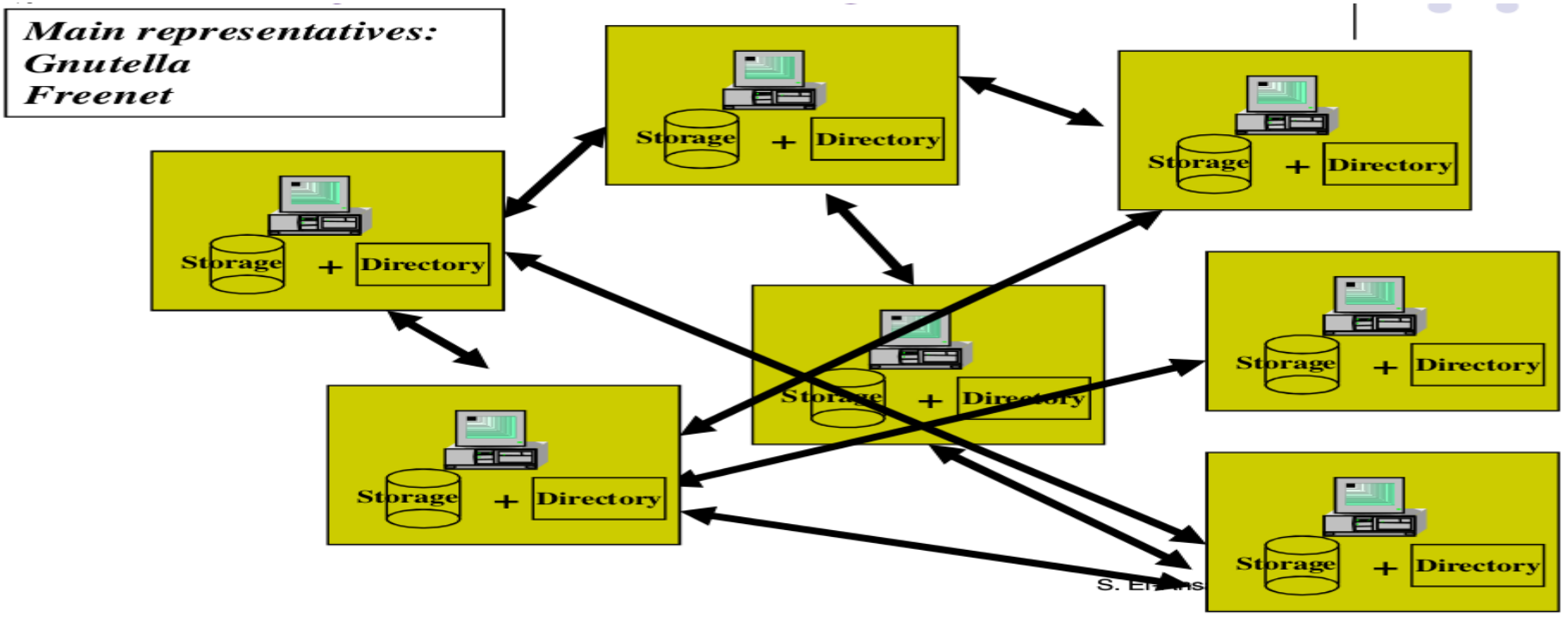


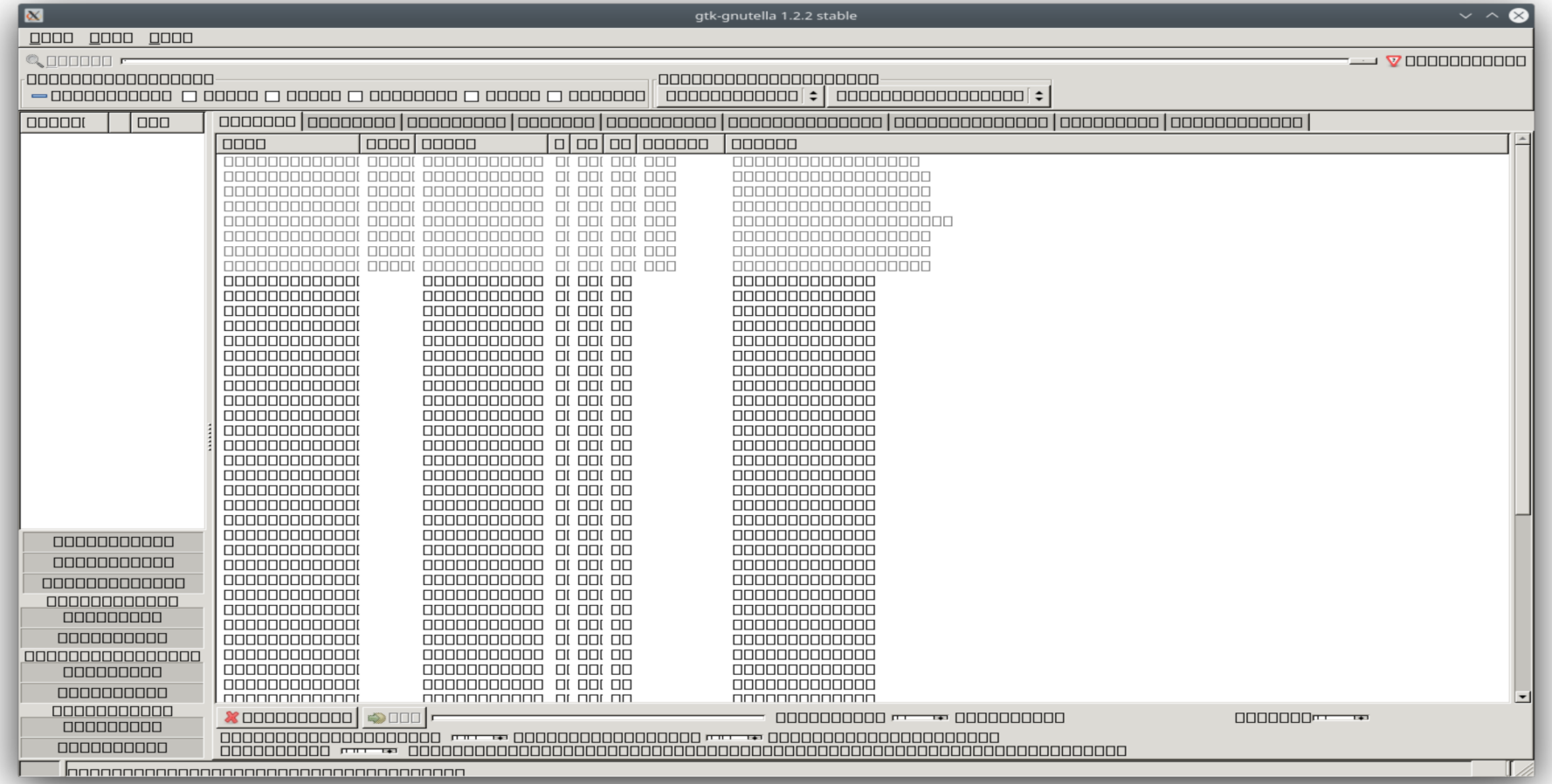
Figure Courtesy of Sameh El-Ansary's and Seif Haridi's course at SICS/KTH

History: Gnutella

- The authors wanted to make a GNU program and ate a lot of Nutella while coding*.
- This is still a functional network with updates rolled out as we speak
- Peers join network by connecting to one of the (always there) peers
  - **Ahmed's claim**: "Used to be available from a link that is now a porn web page"

*According to https://en.wikipedia.org/wiki/Gnutella.

Gtk-Gnutella: tested in 2023



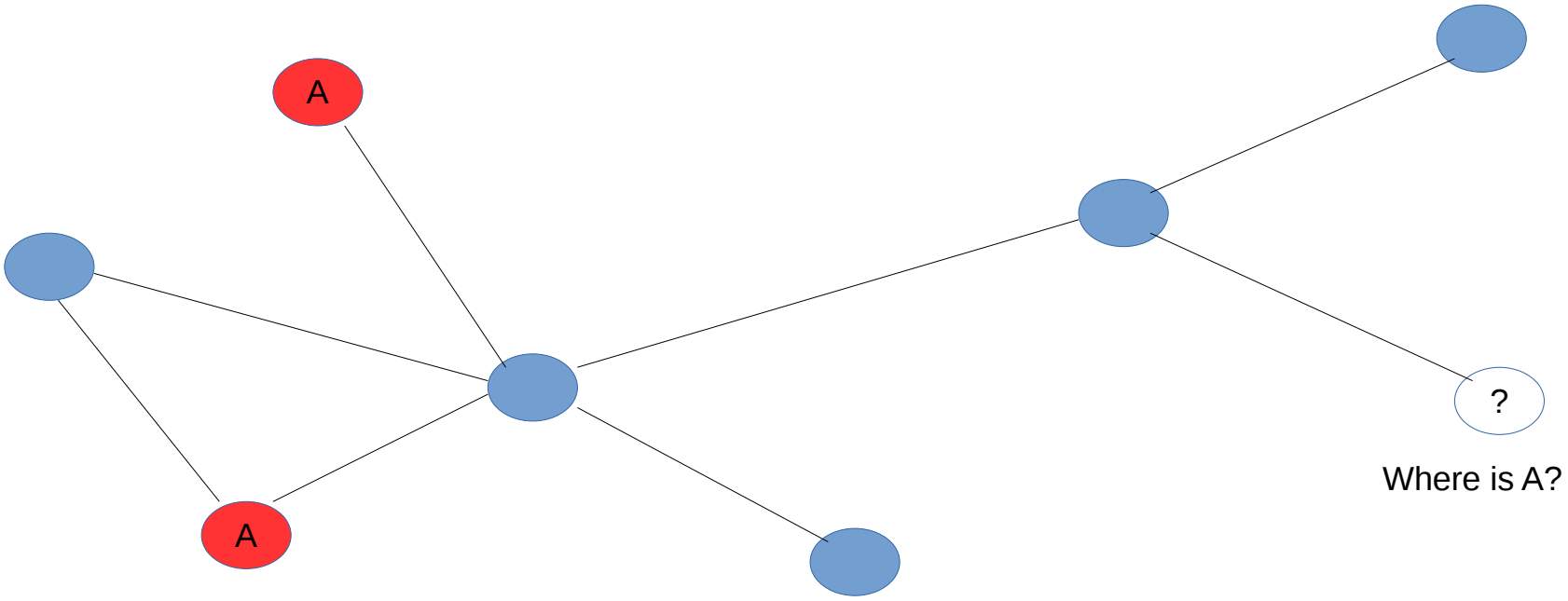
History: Gnutella messages

- **Group Membership**
  - Ping: “Hi I am here, add me to the network”
    - Broadcasted to neighbors, neighbors forward the message
    - Pong: “Cool, my IP-address is x.x.x.x, etc..
      - From all peers who get the Ping

History: Gnutella messages

- **Search**
  - Query: “Does any one have 'foo.mp3'?”
    - Contains a Time-to-Live (TTL)
    - Floods the network until TTL
  - Query response: “ I do, you can get the file through...” or no response
    - “Maybe it is just the TTL, let me increase it and try again”
- **File Transfer**
  - GET or PUSH (if the peer is behind a firewall)

Example for Gnutella (Historical)



History: Gnutella

- a peer periodically PINGs its neighbors to discover other participating peers.
  - And that was problematic
- Flooding
  - Too much traffic
  - Impossible to break the network without taking down everyone (or at least the always on peers)
  - Or targeting the people who originally wrote the software
  - Again, court order against **Limewire** (which introduced some backdoor code for disabling the network in last official version)
  - But the legacy and the network are still there: (http://www.gnutellaforums.com/)

Design question

- How to reduce flooding?

Bittorrent: Content distribution with a twist

- Large number of peers want to download a certain file as fast as possible.
- All previous techniques discussed do not do that
  - Has to wait for a peer already downloading the file
  - Then another peer can start retrieving the file from one of the two peers

Bittorent

- Philosophy:
  - Use the bandwidth of everyone in the network to the maximum

Bittorrent: Layman's example

- One file at Peer1 of size 10 GiB on a 1 Mb/s link.
- 100 peers who want that file at the same time
- A few options to distribute the data

Layman's example: Option 1

- Send it to Peer2 in ~**1 day**.
  - (85899345920/1000000)/3600 = 23.86h → 24h!
- Then send it to Peer3 in another **day**.
- To distribute the file to all peers it will take **100 days**, i.e. roughly 3 months.
- Even with all peers acting like sources once they get the file, you need **6 days** (How did I get this number?)
  - 1+2+4+8+16+32+64 = 127 = 6 rounds = 6 days!
    (the first is the original sender)

Layman's example: Option 2

- Divide the file in to **100 distinct chunks**
- Send to each peer out of the 100 peers **one chunk** of the file
  - Try to be fair with bandwidth allocated to each peer
- Each peer now has a small part of the file after the first day.
  - But the whole file is in the network
  - Peers can then fetch from each other the parts they are missing
  - So all peers will have the file in roughly two days in a very optimal case (since each one can connect to 100 different peers to get 100 different chunks of the file)

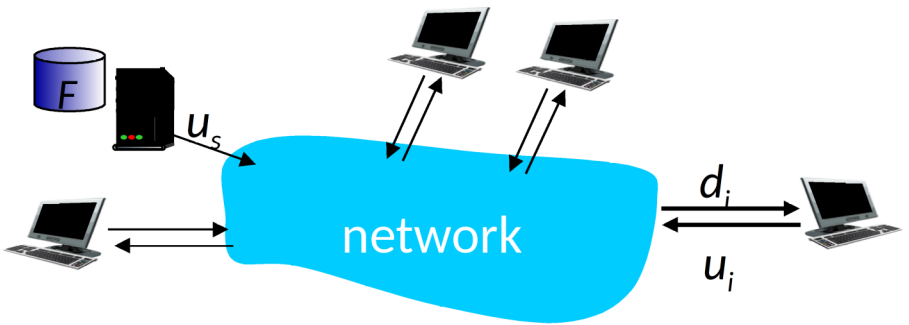
In reality

- “Piece length maps to the number of bytes in each piece the file is split into.”
- “For the purposes of transfer, files are split into **fixed-size pieces** which are all the same length except for possibly the last one which may be truncated.”
- “piece length is almost always a power of two, most commonly 2^18 = 256 K (BitTorrent prior to version 3.2 uses 2 20 = 1 M as default).”

http://www.bittorrent.org/beps/bep_0003.html

Layman's example: Option 2

- Even faster downloads:
  - Smaller chunks will be sent in a few seconds, then each peer in the network will have something to share.



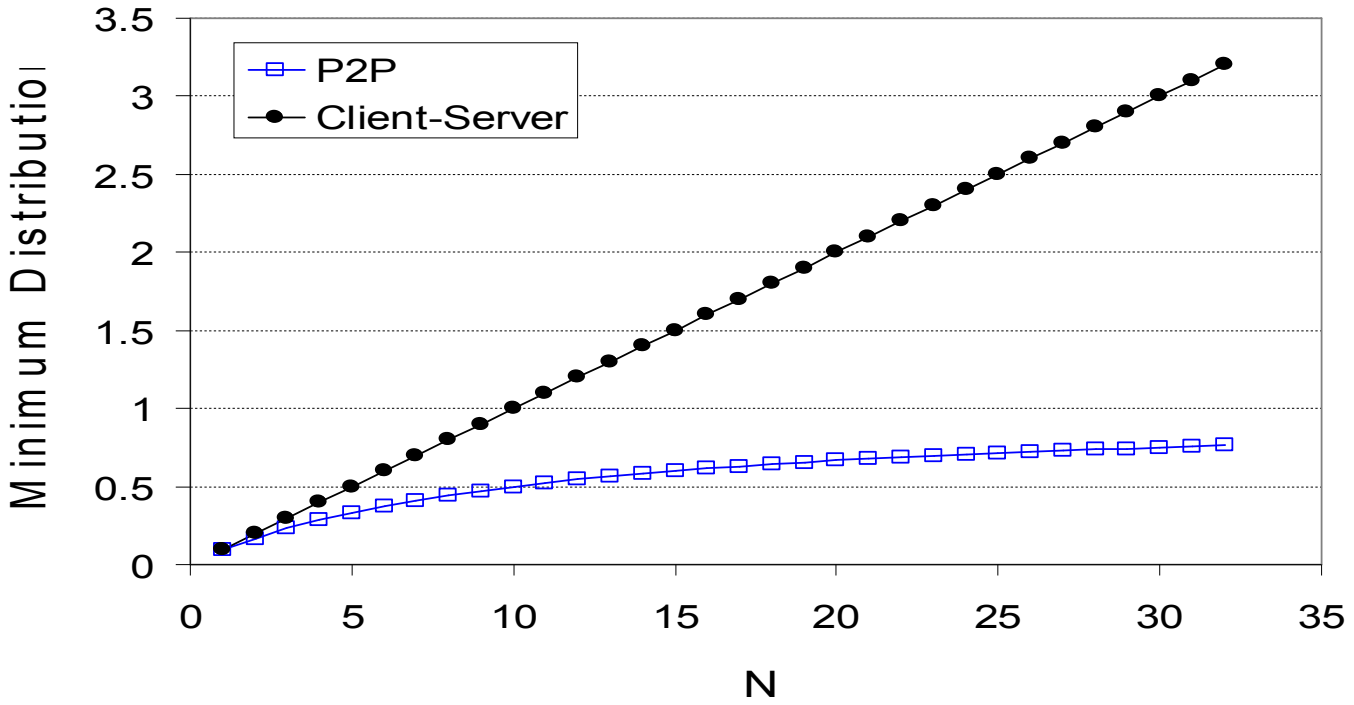
- Making the optimal transfer length for 100 peers just a little over 1 day!

time to distribute F to N clients using P2P approach

$$D_{P2P} > max\{F/u_s, F/d_{min}, NF/(u_s + \Sigma u_i)\}$$

Client-server vs. P2P: example

client upload rate = u, F/u = 1 hour, $u_s = 10u$, $d_{min} \geq u_s$



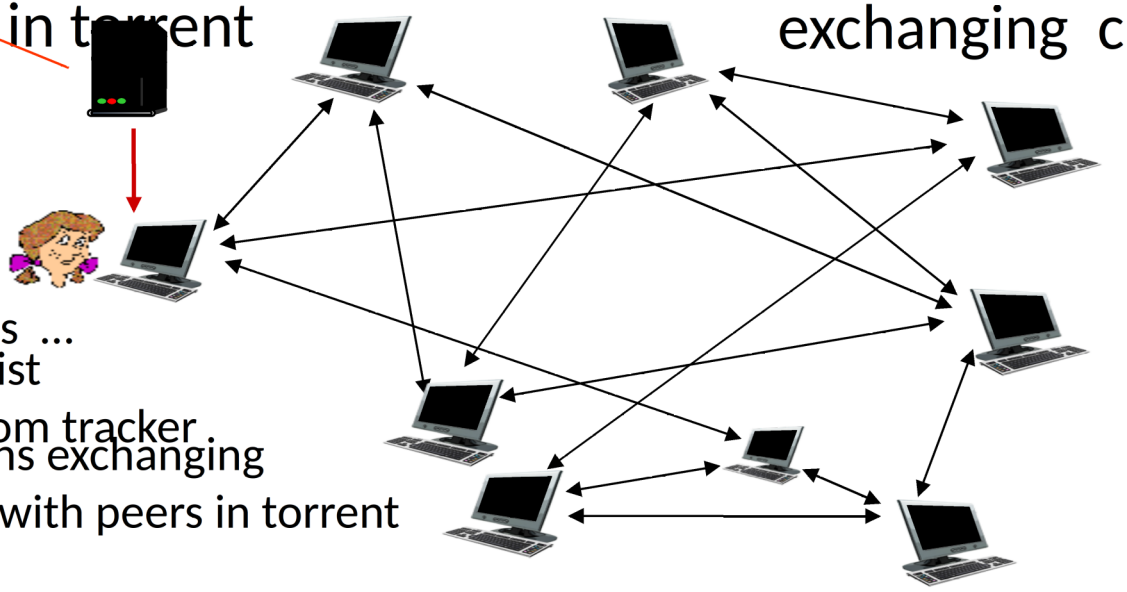
In reality

- You almost never have 100 peers asking for the same file at the exact same instance
- Some will start the download earlier than the others
- The early downloaders will, with high probability, eat up the upload bandwidth of the file source since they establish the connection ahead
  - And each uploader/downloader can set the maximum number of simultaneous connections

Bittorent in action

tracker: tracks peers participating in torrent

torrent: group of peers exchanging chunks of a file



A swarm is the set of peers
that are participating in distributing the same file

Peer of type1: “Seeders”

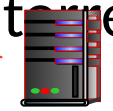
- Machines that have a **complete** copy of the file
- They are usually selfish and do not want to wait after they get the file
- At least the first seed has to stay to serve one complete copy of the file
- In general at all times all pieces of a file must be around or the process fails

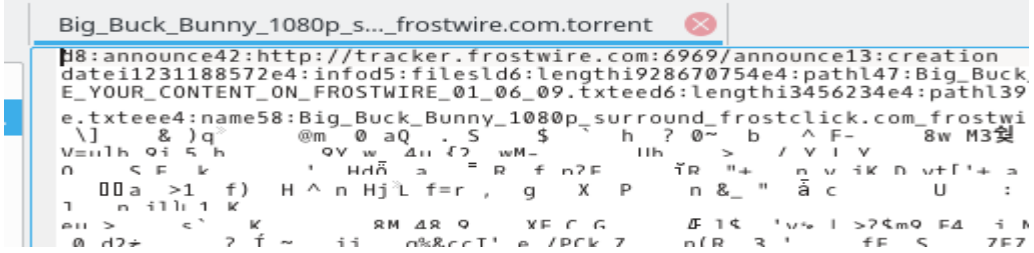
Peer of type2: “Leechers”

- Those who do not have a full copy of the file.
- Once they have the full file, they become seeders.

Tracker (Membership server)

tracker: tracks peers participating in torrent



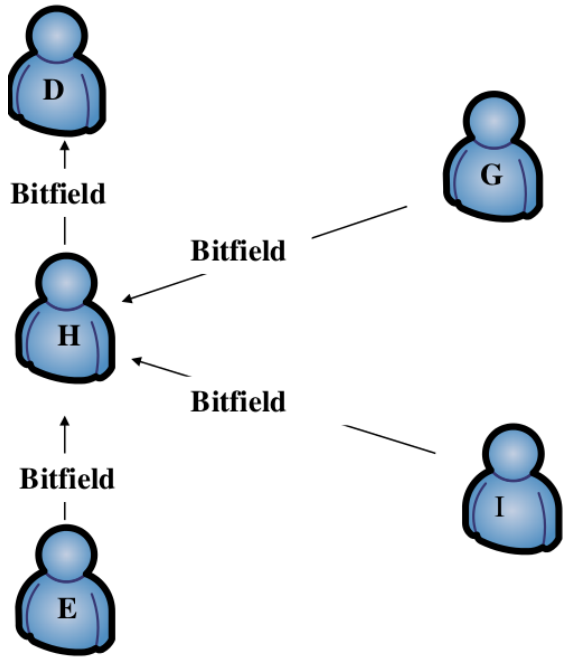


- A peer that keeps track of the seeds and the leechers
- Connected to when you download a .torrent file, which contains information about the file, its length, name, and hashing information, and URL of a tracker

How to find torrent files?

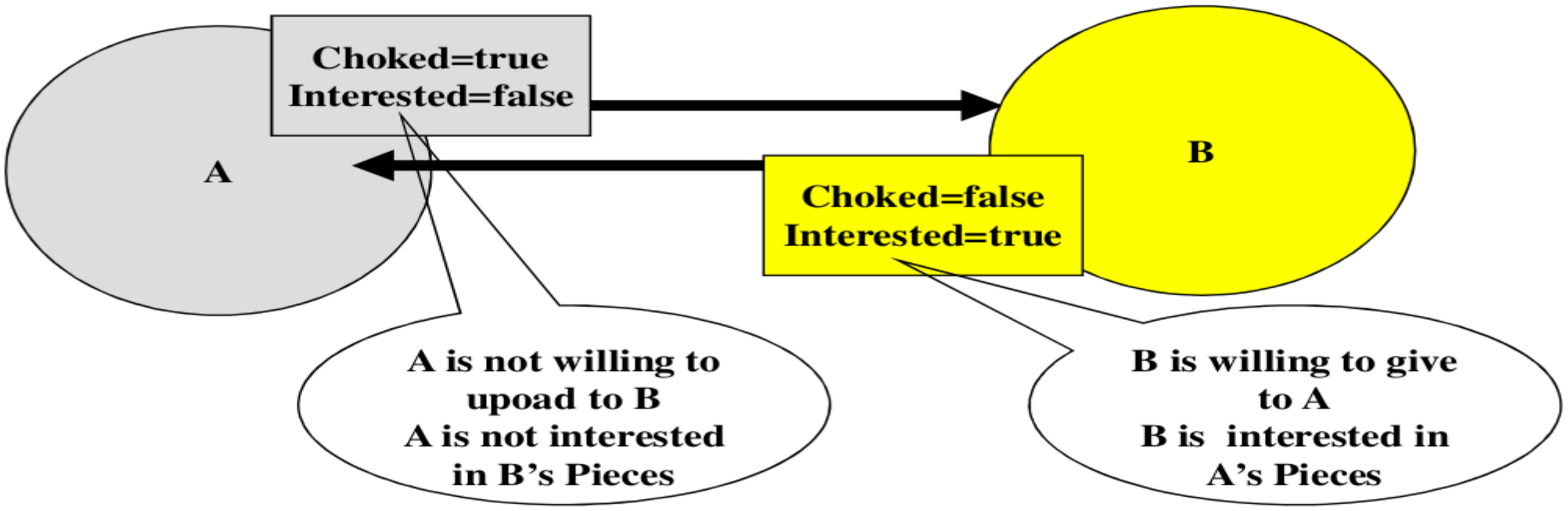
Use eg a popular website like the Pirate Bay.

Handshaking and exchanging Bitfields



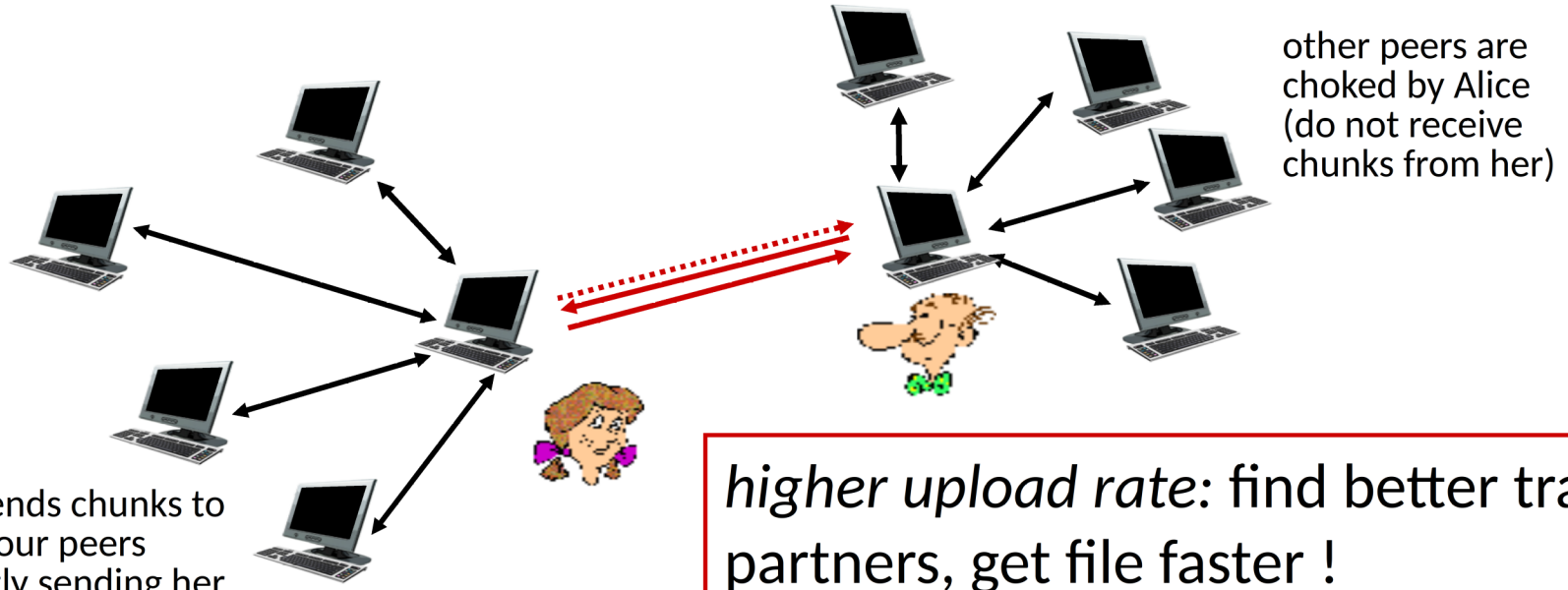
- Send handshake to all peers received from the tracker
- Exchange bitfields
- Then choose who can provide you with the pieces that you are missing!

Choking algorithm: One sided relationship



Choking algorithm: Mutual love or Tit-for-tat

(1) Alice “optimistically unchokes” Bob (eg every 30 secs)
(2) Alice becomes one of Bob’s top-four providers; Bob reciprocates
(3) Bob becomes one of Alice’s top-four providers



other peers are choked by Alice (do not receive chunks from her)

Alice sends chunks to those four peers currently sending her chunks *at highest rate (check every 10 secs)*

higher upload rate: find better trading partners, get file faster !

Which piece to get first

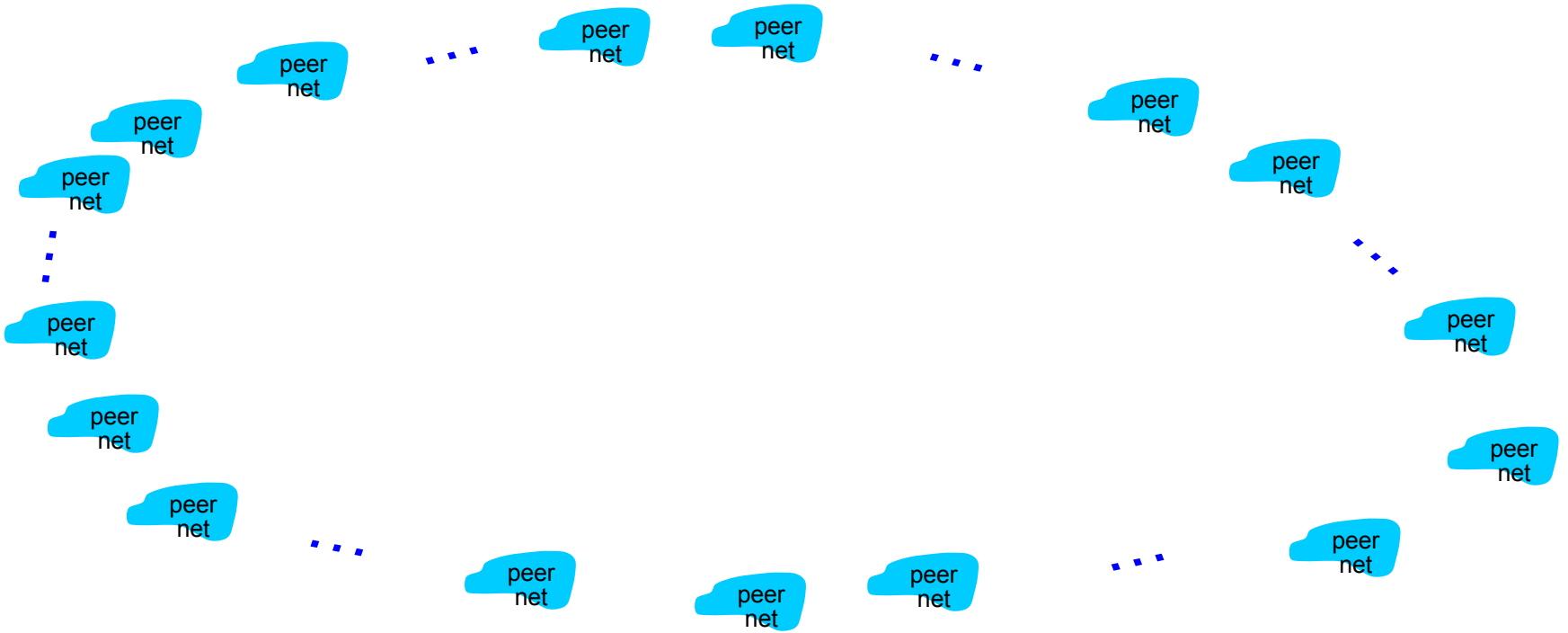
- Rarest-first
  - Choose the piece that is rarest among your peers
  - If possible
- Random-first piece
  - As we already know, the first piece is crucial for someone to be useful to the community
  - So, rarest-first is not applied
  - Get first piece as fast as possible, even if you download subpieces from different peers
- “Streaming” (latest BitTorrent versions)

BitTorrent legal uses

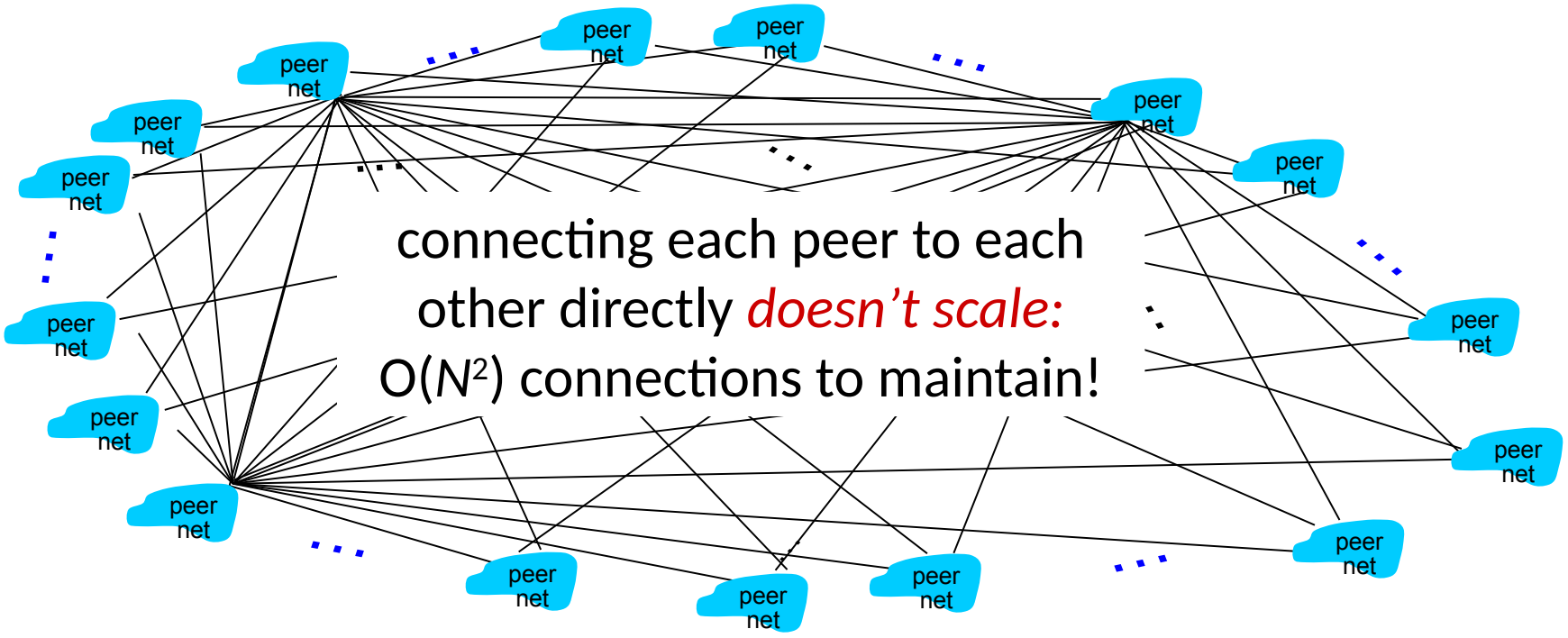
- https://www.makeuseof.com/tag/8-legal-uses-for-bittorrent-youd-be-surprised/

Structured Overlays

How to structure a P2P network?

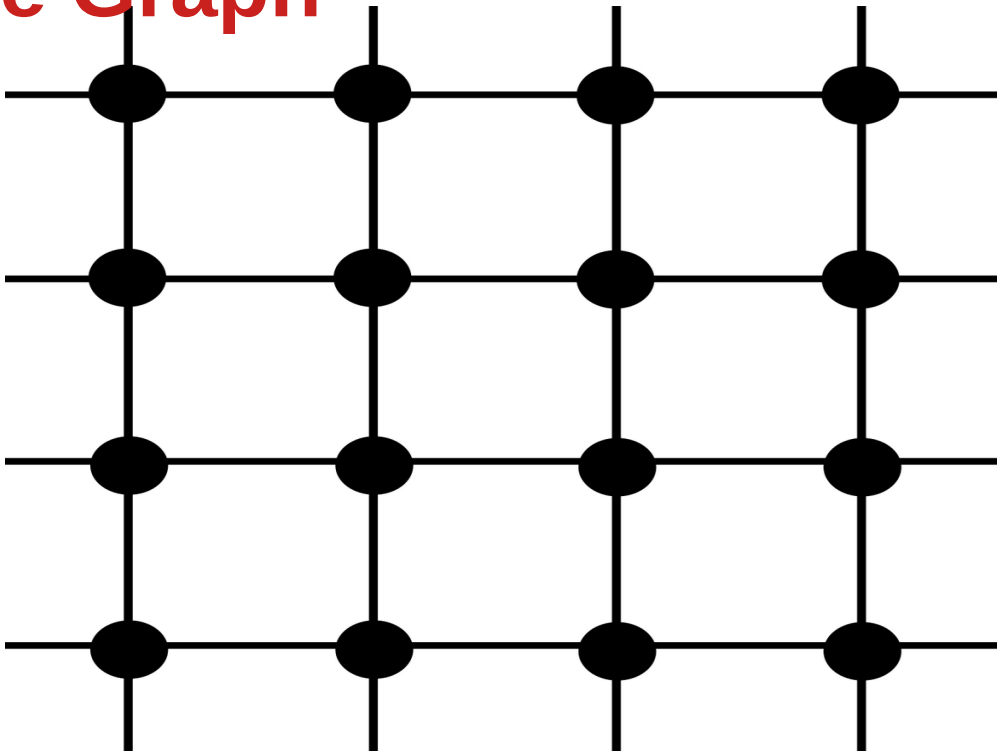


How to structure a P2P network?



Better?

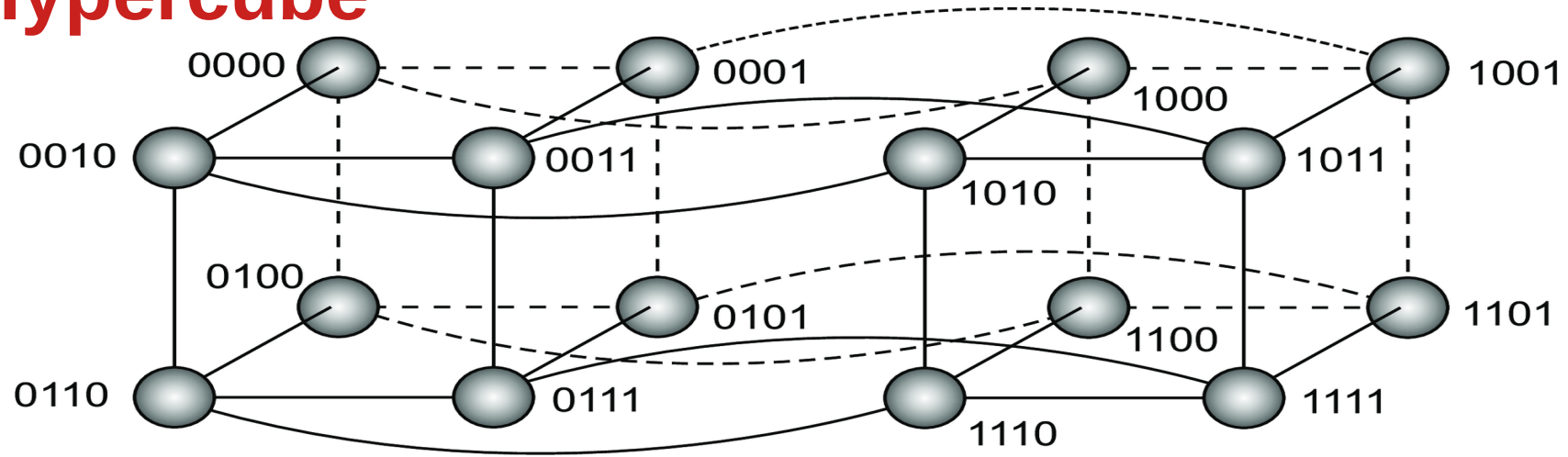
Lattice Graph



- **Degree**:
  - Constant!
- **Diameter**:
  - Linear!

Even Better?

Hypercube

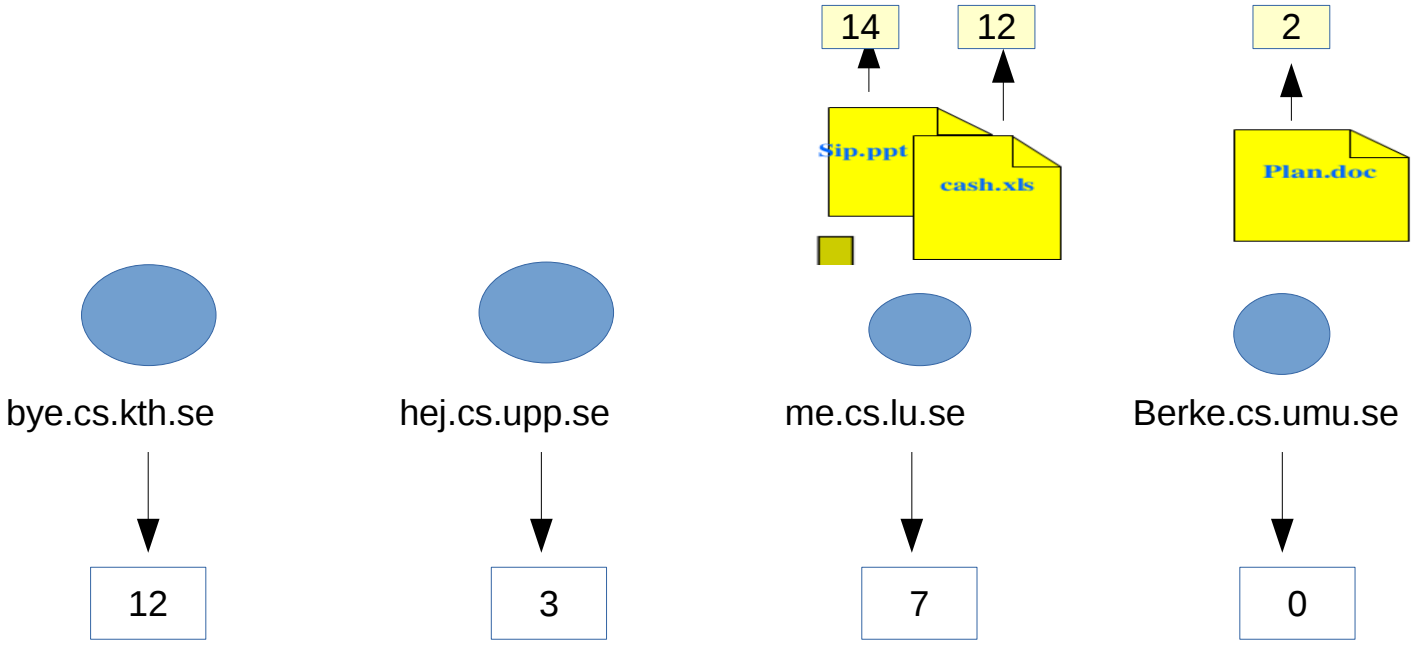


- **Degree**:
  - Logarithmic!
- **Diameter**:
  - Logarithmic!

Premise of Chord / DHT

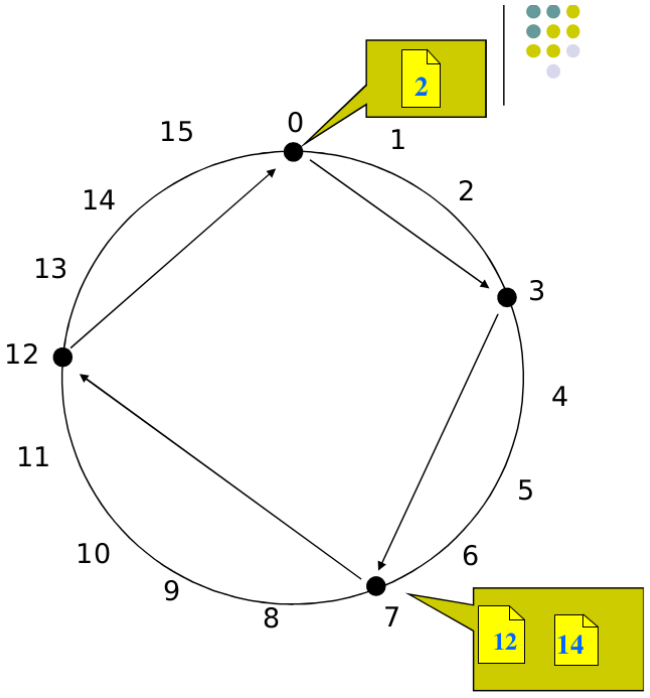
The ID of a data item determines the machine on which it is going to be stored

Simplest Example: Consistent Hashing



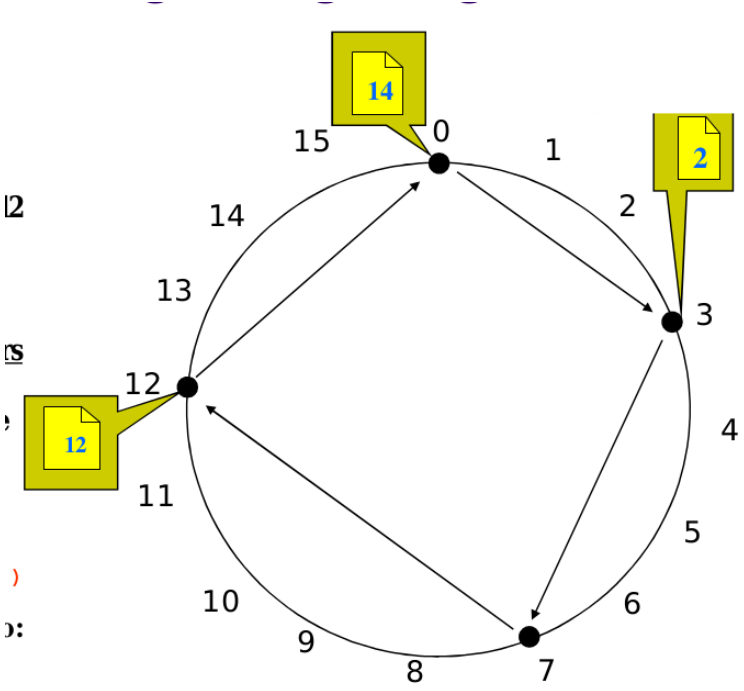
- 4 machines that need to share data
- Each machine will have an id based on the hash of its IP address
- Data will also have ids using the same function (SHA, MD5sum..)

Simple Example: Build a ring



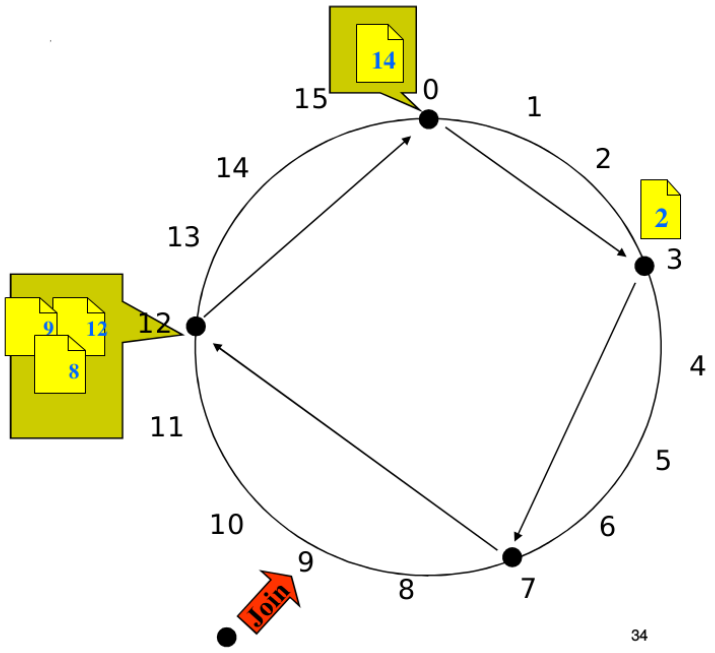
- Original structure

Simple Example: update the ring



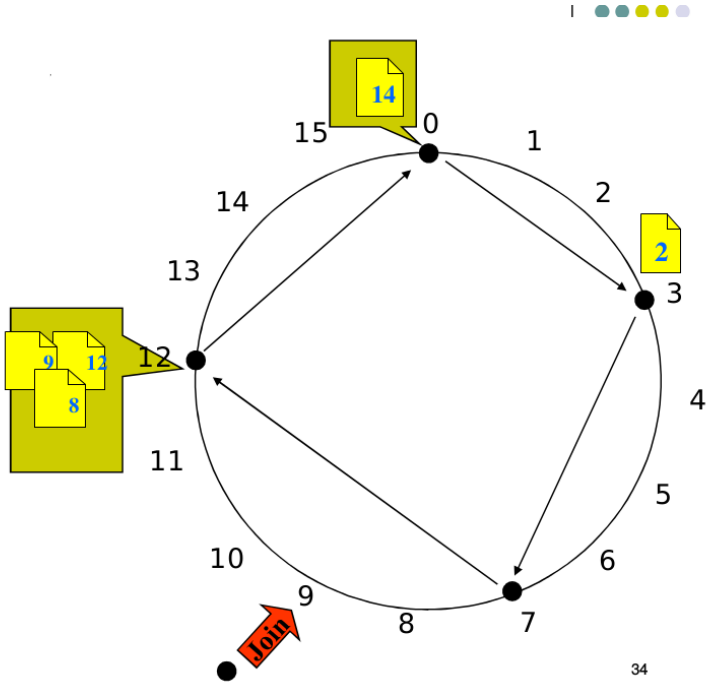
- Policy
  - A doc with id y, would be stored at Succ(y)
- Lookup
  - ask node n which you know about to find the successor of id.

Handling Joins



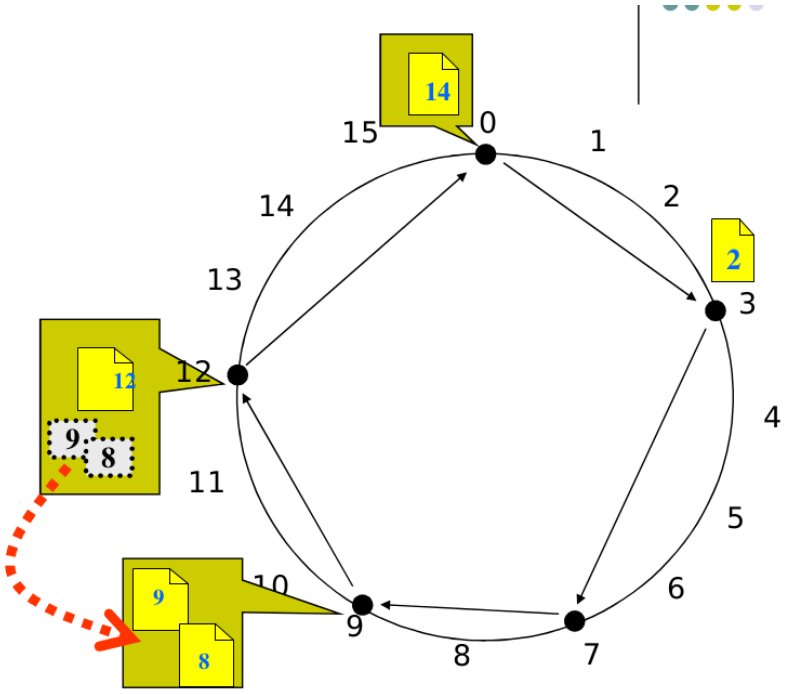
- You need to know the successor
- Your predecessor needs to know about you

New example: Handling Joins



- A node 9 joins the ring

Handling Joins: data redistribution



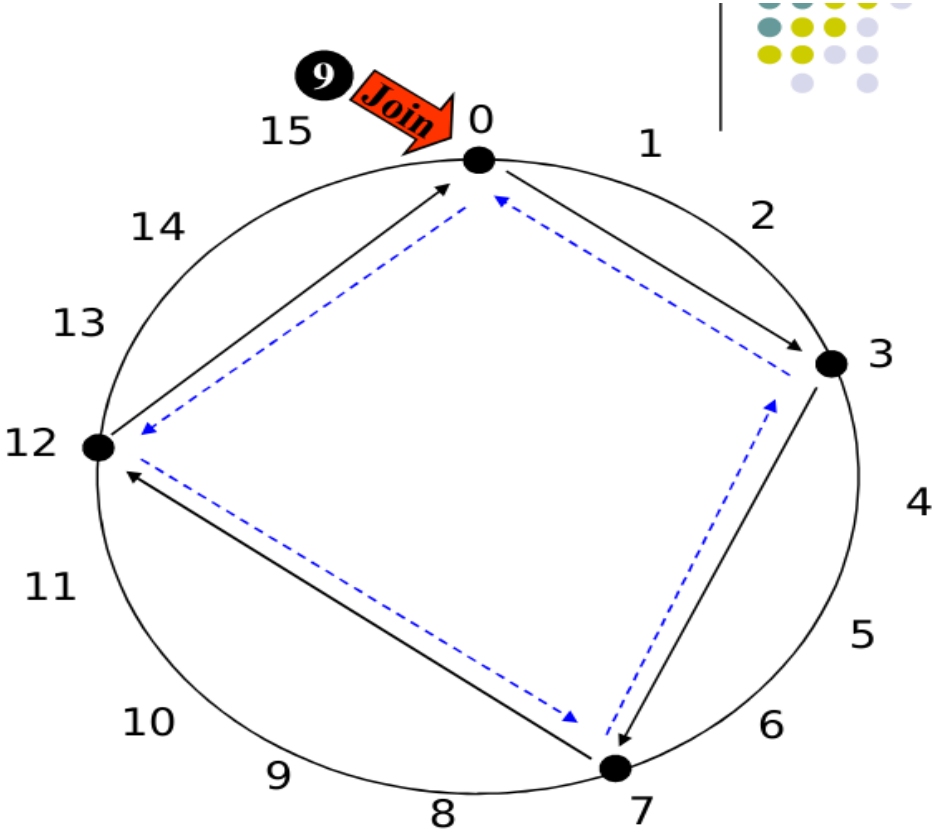
- Can you think what happens on data leaves?

Chord DHT

- Developed at MIT
- Very popular since it is very simple to understand
- In addition to the successor pointer, every node has a predecessor pointer as well

Chord Join

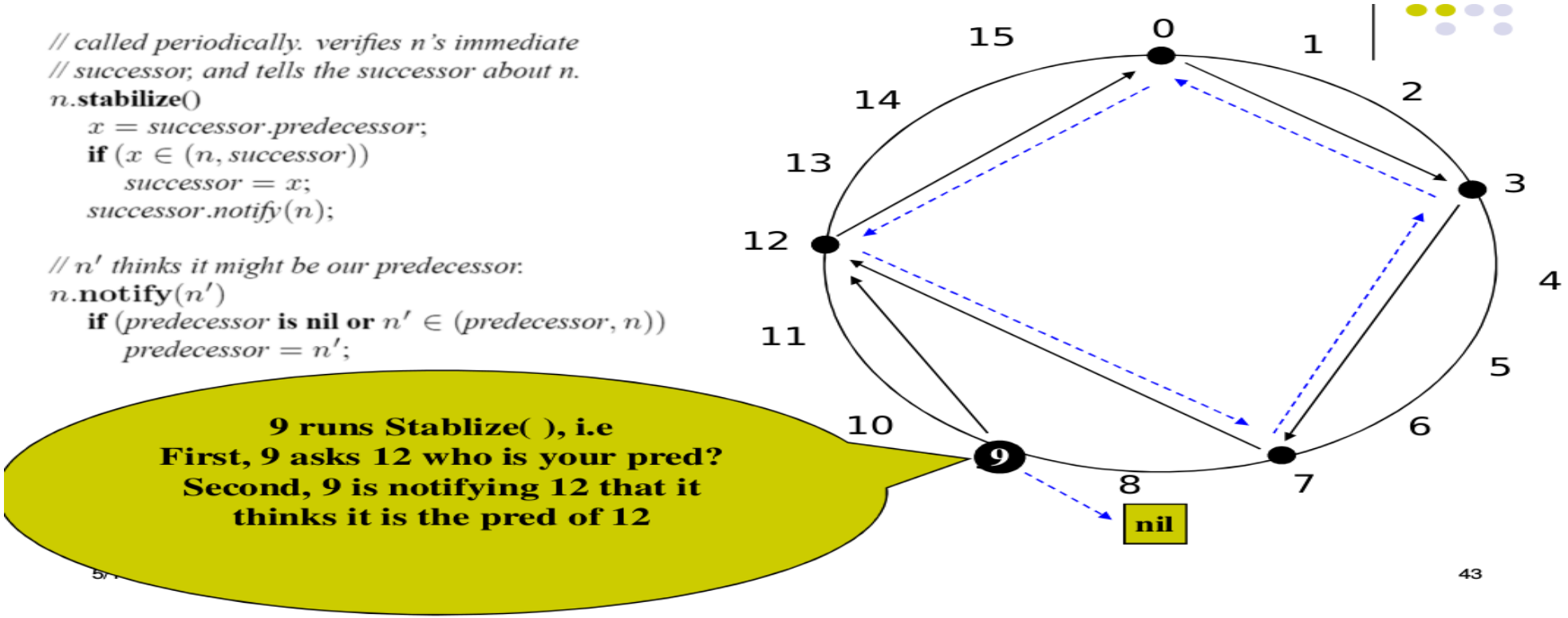
- 9 can join through any other node, take 0 for example.
- 9 will set its predecessor to nil
- 0 will help 9 to find its successor
- 0 will tell 9 that its successor should be 12



Chord Join

```
// called periodically. verifies n's immediate
// successor, and tells the successor about n.
n.stabilize()
    x = successor.predecessor;
    if (x ∈ (n, successor))
        successor = x;
    successor.notify(n);

// n' thinks it might be our predecessor.
n.notify(n')
    if (predecessor is nil or n' ∈ (predecessor, n))
        predecessor = n';
```



- 12 changes its predecessor to 9
- 7 will learn that 9 is it successor when it runs stablize
  -12 will tell

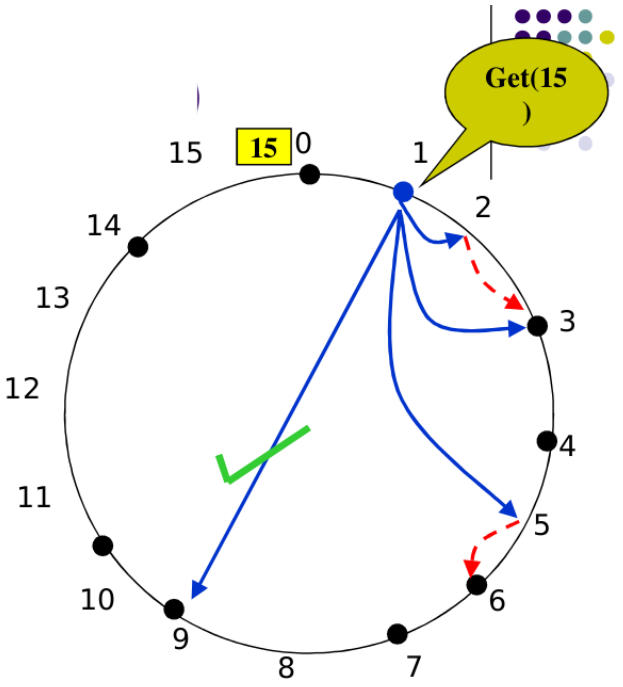
Can their be collisions in the peer names?

- Can the hash function produce two peers having the same hash ID?
  - H(a) = H(b), and a ≠ b?
  - Yes?
  - No?

Can their be collisions in the peer names?

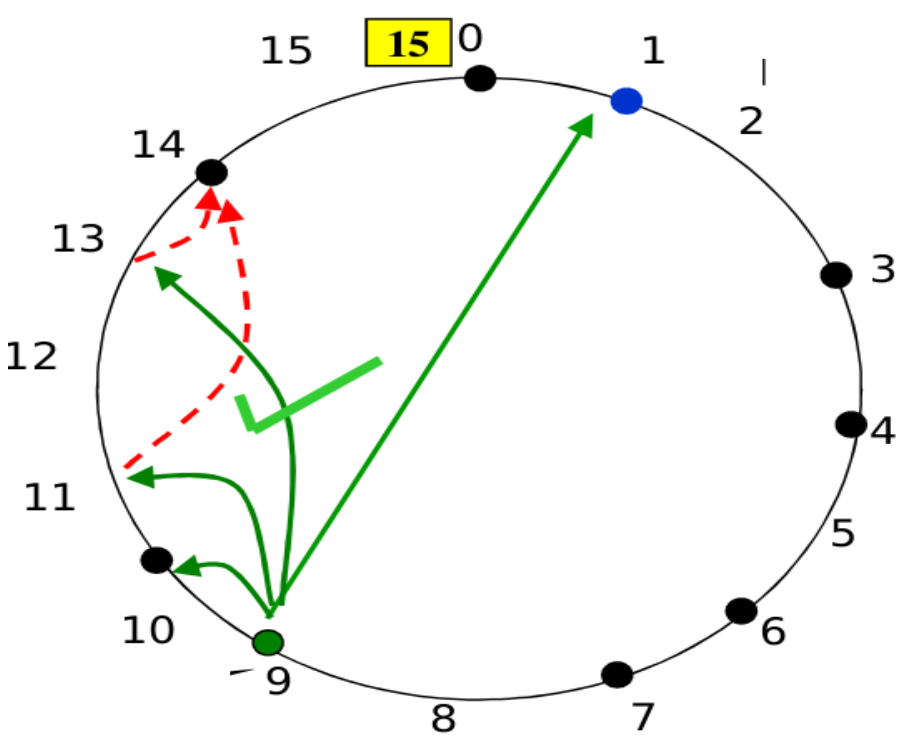
- Can the hash function produce two peers having the same hash ID?
  - No, if you use a good function (and some other hypothesis on hardness to solve certain problems)
    - Called Provably secure hash functions
  - MD5 and SHA-I are not Provably secure hash functions
  - Still we use non provably secure hash functions and seldom use the provably secure ones
    - Can you think why?
      - https://en.wikipedia.org/wiki/Security_of_cryptographic_hash_functions
    - But we pick one that is hard to have these collision (very low probability that you can do the attack)

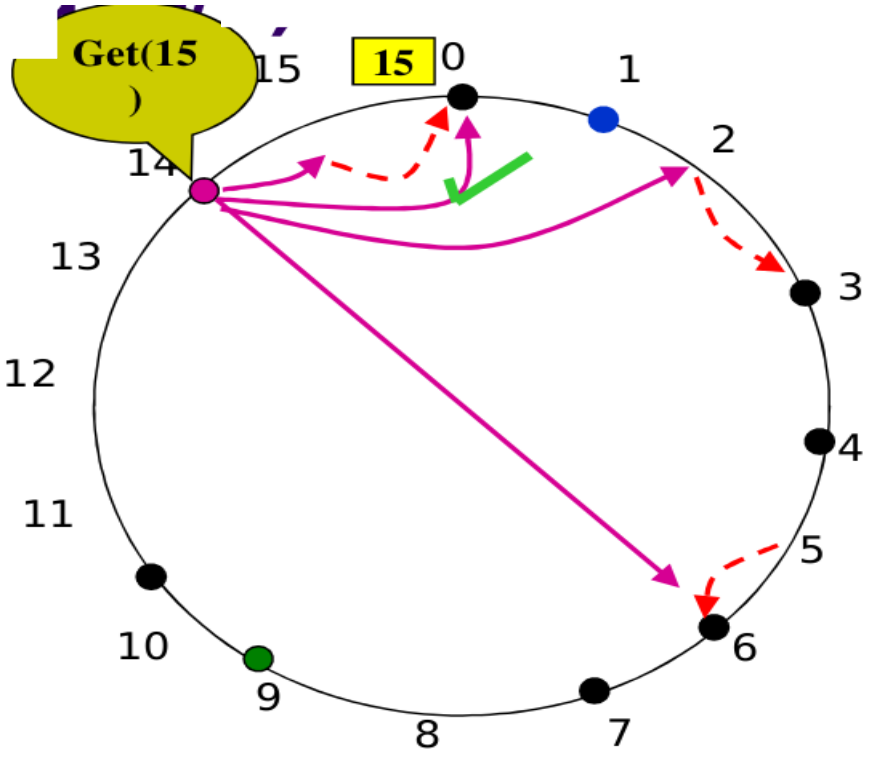
Routing: Fingers table



- Keep a routing table of M peers at each node
  - Where $N=2^M$
  - Maximum distance to any other node is then $\log_2(N)$, i.e., M hops

Routing: Fingers table





Chord: Routing

- In fact, if nodes are uniformly distributed, the maximum is log (# of nodes), i.e. log (8) hops between any two nodes
- The average complexity is: 1⁄2 log(#nodes)

Chord: Handling failures

- If one successor fails, total ring collapses (Why?)
  - Each node maintains a successor list of length r
  - If a node’s immediate successor does not respond, it uses the second entry in its successor list
    - Update the successors' list

Would not it be nice to have something with locality?

- A huge issue with chord is its inability to take locality in account
- Also some practicalities are abstracted (how to transfer the files, eg use pointers instead)

Drawbacks of P2P systems?

- Major drawbacks with P2P systems [7]
  - Churn and performance guarantees
  - The power of cloud computing
    - Skype abandoned their P2P architecture (2017)
    - Spotify abandoned their P2P streaming option (2014)
  - Decreasing cost of content delivery

Security

- We have mostly talked about robustness and efficiency
- P2P adds a new dimension to security
  - With newer attacks and newer challenges
- Example:
  - Sybil attacks
  - File Poisoning
  - Rational Attacks
- Plus the good old ones like DdoS, man-in-the-middle,..etc.

Security: Sybil attacks

- Adversary introduces a large number of peers so to control the P2P system
  - Subverting the reputation system
  - Corrupting the network
  - Tricking the users
  - Controlling/influencing the network
- Problem for consensus algorithms! (Blockchain, eg Bitcoin)

"One can have, some claim, as many electronic personas as one has time and energy to create."

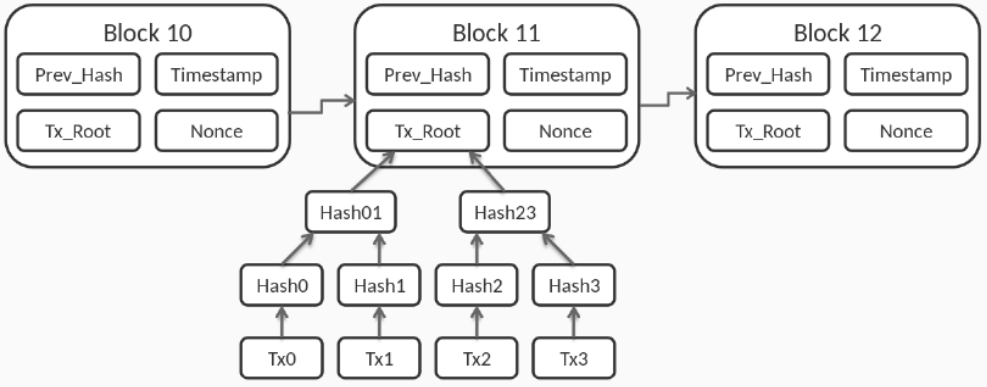
Judith S. Donath

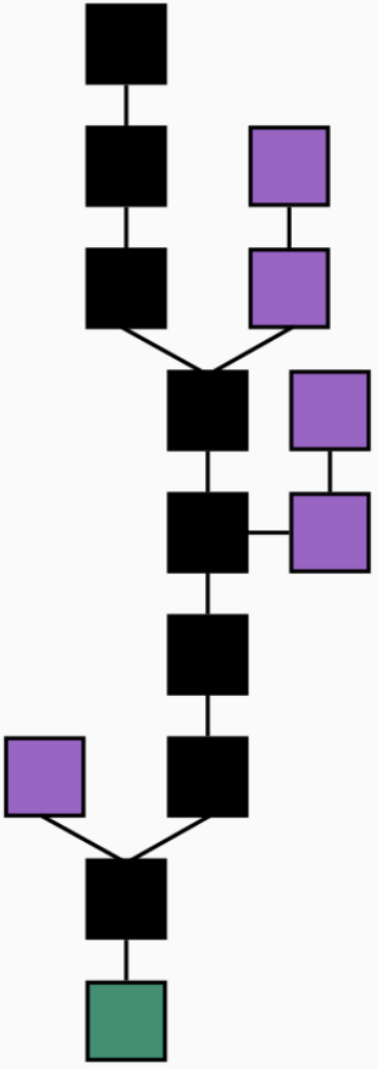
Where does P2P live today

- Internal within companies and datacenters
  - Eg Dynamo at Amazon
- Blockchains

Blockchain

- Invented in 2008 by "Satoshi Nakamoto"
- **Decentralization**
- Public-key cryptography
- No retroactive modifications
- Mining nodes validate transactions paid
- **Bitcoin**: Hashcash puzzles





Consensus in blockchains

Problem for consensus algorithms! (Blockchain, eg Bitcoin)

- **Proof of Work**: show that you own computational capabilities
- **Proof of Stake**: show that you own assets
  - Sept 2022, Ethereum switched to Proof of Stake, 99% cut in energy consumption.

Some extra extra extra readings :)

- (Original slides from Ahmed Ali-Eldin Hassan <ahmed.hassan@chalmers.se>, please be kind to forward all complaints and copyright queries to him.)
  - (some material from From Kurose & Ross Book, Computer Networking: A Top-Down Approach, 8th edition, Jim Kurose, Keith Ross, Pearson, 2020.)
- [1] http://spin2013.cs.sunysb.edu/~phillipa/CSE390/choffnes08.pdf
- [2]https://www.net.t-labs.tu-berlin.de/papers/AFS-CIP2PCIP-07.pdf
- [3] https://www.usenix.org/legacy/event/imc05/tech/full_papers/karagiannis/karagiannis.pdf
- [4] http://research.cs.washington.edu/networking/websys/pubs/osdi_2002/osdi.pdf
- [5] http://www.computer.org/csdl/proceedings/p2p/2001/1503/00/15030101.pdf
- [6] http://www.cs.rice.edu/Conferences/IPTPS02/101.pdf
- [7] Li, Baochun, Yuan Feng, and Bo Li. "Rise and fall of the peer-to-peer empire." Tsinghua Science and Technology 17.1 (2012): 1-16. http://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=A744210E174E5775327F6B8F844D91E8?doi=10.1.1.296.1725&rep=rep1&type=pdf
- [8] https://www.allthingsdistributed.com/2007/10/amazons_dynamo.html