Chapter Outline

- Overview of (previously) deployed P2P systems in 3 areas
- P2P file sharing and content distribution:
  - Napster, Gnutella, KaZaA, BitTorrent
  - Differences, strengths, weaknesses
- P2P Communication
  - Typical instant messaging setup
  - Skype
- P2P Computation
  - SETI@Home example

Napster

- Napster was the first P2P file sharing application
- Only sharing of MP3 files was possible

- Napster made the term “peer-to-peer” known
- Napster was created by Shawn Fanning
  - “Napster” was Shawn’s nickname

- Do not confuse the original Napster and the current Napster
  - Latter is an online music store, nothing to do with P2P
  - Uses Napster name mainly to attract people

History of Napster

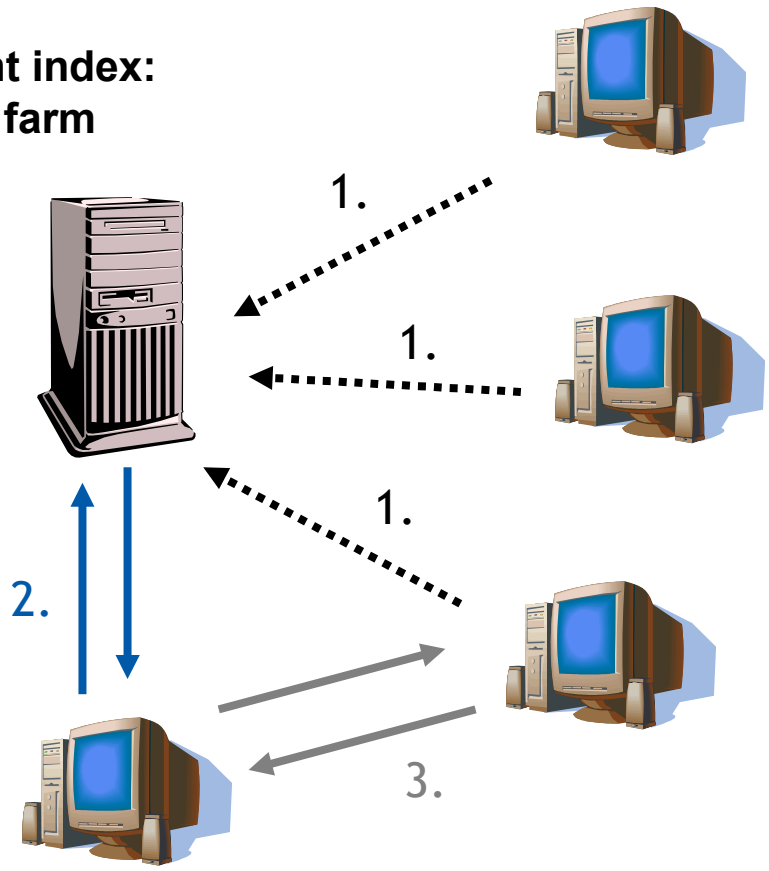
- Napster started in fall of 1999
- First lawsuit in December 1999 from several major recording companies
- Napster grew in popularity
  - Peaked at 13.6 million users in February 2001
- July 2001 judge ordered Napster to shut down
- Case partially settled on September 24, 2001
  - Napster paid $26 million for past and future damages
- Bertelsmann AG bought Napster on May 17, 2002
  - Napster filed Chapter 11 bankruptcy protection
  - On September 3, 2002, Napster forced to liquidate (Chapter 7)
- On October 29, 2003 Napster came back as an online music store

Napster: How It Worked?

- Napster was based on a central index server
  - A server farm, actually
- User registers with the central server
  - Sends list of files to be shared
  - Central servers know all the peers and files in network

- Searching based on keywords
- Search results were a list of files with information about the file and the peer sharing it
  - E.g., encoding rate, size of file, peer's bandwidth
  - Some information entered by the user, hence unreliable

Napster: Queries



1. Peers register with central server, send list of files to be shared
2. Peers send queries to central server which has content index of all files
3. File transfers happen directly between peers

Last point is common to all P2P networks and is their main strength as it allows them to scale well

Napster: Strengths

- Consistent view of the network
  - Central server always knows who is there and who is not (in theory, to some extent...)
- Fast and efficient searching
  - Central server always knows all available files
  - Efficient searching on the central server
- Answer guaranteed to be correct
  - “Nothing found” means none of the current on-line peers in the network has the file

Napster: Weaknesses

- Downloading from a single peer only (manual load balancing…)
- Central server is a single point of failure
  - Both for network attacks…
  - … as well as all kinds of attacks
  - Ultimately this was a big factor in the demise of Napster
- Central server needs enough computation power to handle all queries
  - Then again, Google handles a lot more…
  - This weakness can be solved with money, by adding hardware
- Results unreliable
  - No guarantees about file contents (as in most P2P networks)
  - Server know nothing about local situation at peers (already uploading..)
  - Some information (e.g., user bandwidth) entered by the user, not guaranteed to be even close to correct (i.e., not measured)

    (This weakness applies to all networks to a large degree)

Gnutella

- Gnutella came soon after Napster
- Answer to some of Napster's weaknesses
- Gnutella introduces entirely different problems

- Gnutella is at the opposite end of the spectrum
  - Napster is centralized
  - Gnutella is fully distributed

- Open protocol specifications
  - Other P2P systems are proprietary
  - Popular for research work

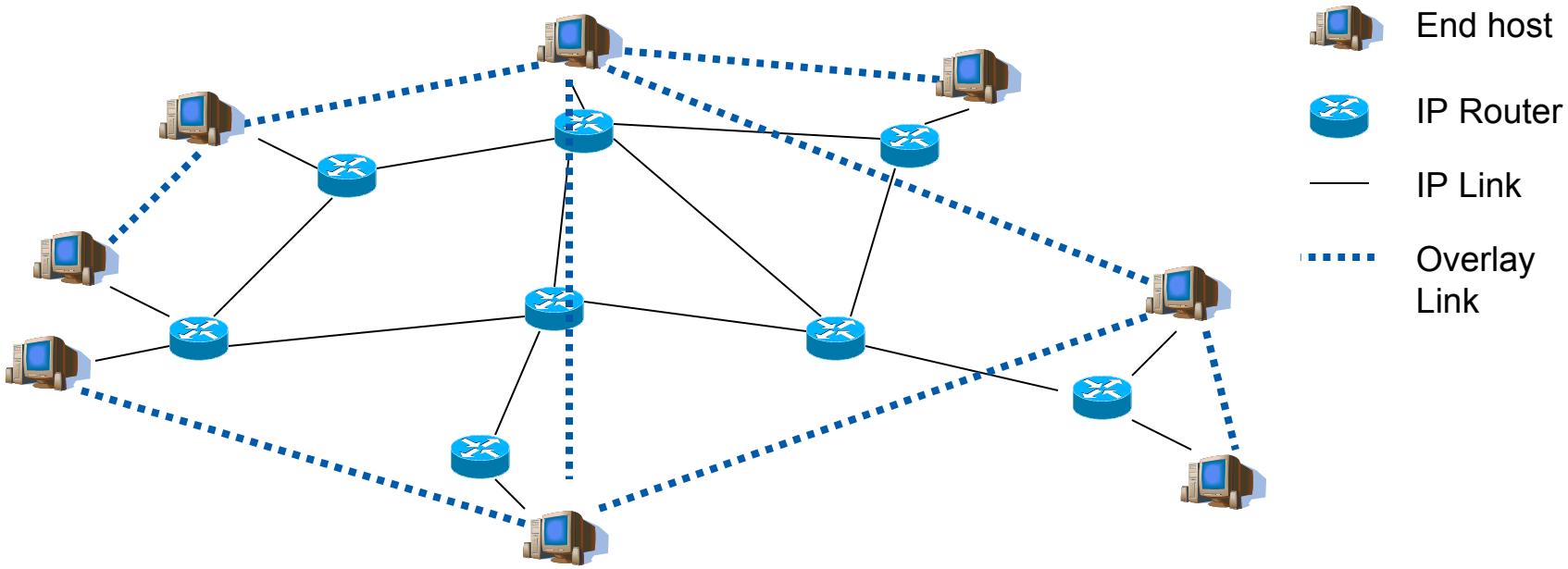
Gnutella History

- Gnutella software originally developed by Nullsoft (*)
  - Nullsoft bought by AOL
- Accidentally released on the website, quickly taken out
  - But damage had already been done, and code was available
- Version 0.4 is covered here (= original Gnutella version)
- Current version 0.6 is similar to FastTrack (KaZaA)
- Gnutella was never a big network
- It provided an alternative to Napster, but was quickly surpassed by other (= better) networks like KaZaA
- Currently old Gnutella is not in use anymore

(*) also known for: WinAmp, ShoutCast, W.A.S.T.E

Gnutella: Overlay Network

- Gnutella is based on an overlay network
- Overlay network means a virtual network on top of the underlying IP network

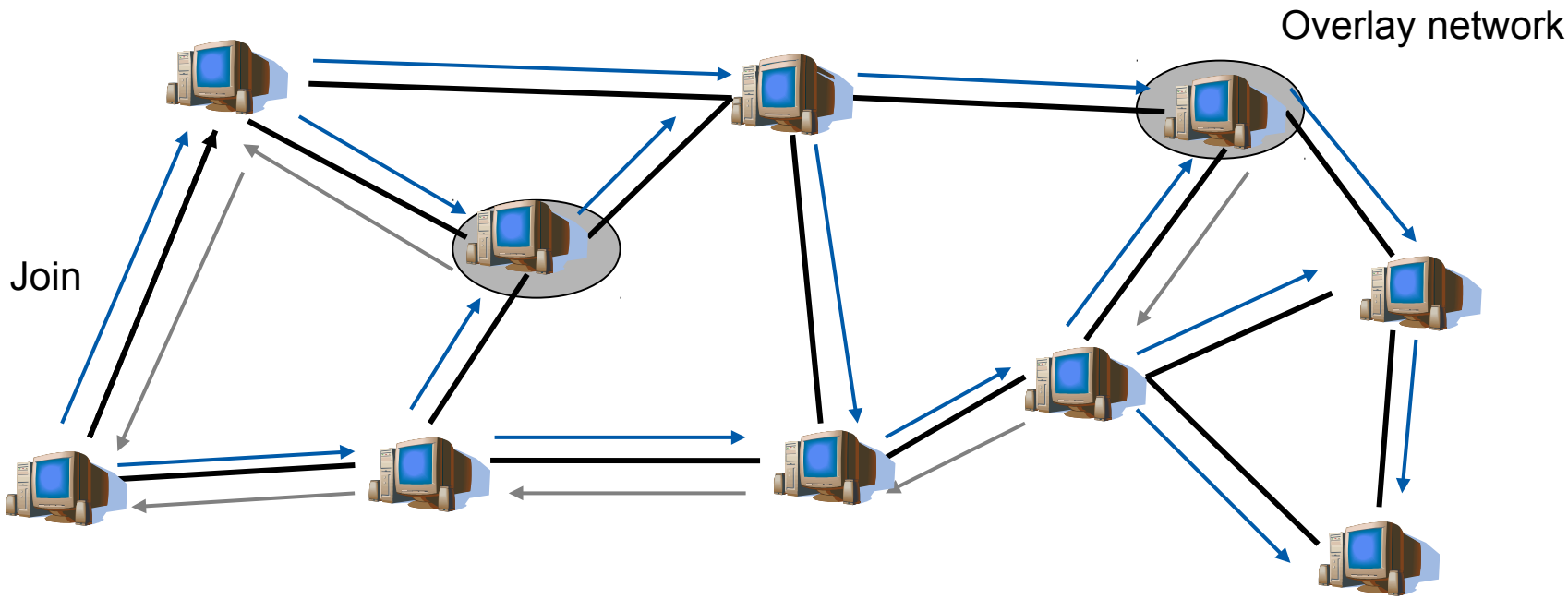


Most current P2P systems based on some kind of overlay

Gnutella: How it Works

- Gnutella network has only peers
  - All peers are fully equal
  - Peers called servents (**serv**er + cli**ent**)
- To join the network, peer needs the address of another peer that is currently a member
  - Out-of-band channel, e.g., get it from a website
- Once a peer joins the network, it learns about other peers and learns about the topology of the network
- Queries are flooded into the network
- Downloads directly between peers

Current Systems: Gnutella



- To join, peer needs address of one member, learn others
- Queries are sent to neighbors
- Neighbors forward queries to their neighbors (flooding)
- Replies routed back via query path to querying peer

Gnutella: Joining the Network

- A peer who wants to join the Gnutella network, needs the address of one peer who is already a member
- New peer sends connect message to existing peer
  - GNUTELLA CONNECT
- Reply is simply “OK”
  - No state involved at this point
- The point of this message is not very clear..
  - Gnutella uses TCP connections, no need to wave “hi” again
  - Receiving peer can deny the join (denial of service)
  - In fact, most of Gnutella 0.4 does not make much sense…
    - Mixing text and binary, little-endian and big-endian

Gnutella: PING/PONG

- A peer discovers other peers in Gnutella with the PING and PONG messages

- PING:
  - Used to actively discover hosts on the network. A servent receiving a Ping descriptor is expected to respond with one or more Pong descriptors.

- PONG:
  - The response to a Ping. Includes the address of a connected Gnutella servent and information regarding the amount of data it is making available to the network.
- PONGs sent back along the same path as PING took

Gnutella: QUERY/QUERYHIT

- Finding content happens with QUERY and QUERYHIT messages
- QUERY:
  - A servent receiving a Query descriptor will respond with a QueryHit if a match is found against its local data set.
- QUERYHIT:
  - The response to a Query. This descriptor provides the recipient with enough information to acquire the data matching the corresponding Query.
- Servents receiving QUERY messages forward them to their neighbors (unless TTL expired)
- Replies returned along the same path

Gnutella: Download

- Peer sends QUERY to find files it wants
- If QUERY reached a peer with matching content, the querying peer will receive QUERYHIT
- QUERYHIT contains the IP address and port number of the peer who sent it

- Transfer: contact that peer directly
- Downloading happens over HTTP
  - Use the given port number, but HTTP syntax for request
- Download directly between peers
  - Gnutella network not involved in downloads

Gnutella: Extra Features

- One additional feature for clients behind firewalls
- If a peer is behind a firewall, it may be impossible to contact it
  - If that peer wants to share files, it cannot do so easily
- Gnutella defines the PUSH message
  - Peer outside firewall sends PUSH to peer inside firewall via relay
    - Assumption: Peer inside firewall keeps a TCP connection open to some neighboring peers in the overlay
  - Peer inside firewall contacts peer who sent PUSH
  - File transfer happens normally

Gnutella: Strengths

- Fully distributed network, no weak points
  - At least on paper…
- Open protocol
  - Easy to write clients for all platforms
  - For example, KaZaA not available for Linux
- Gnutella is very robust against node failures
  - Actually, this is only true for random failures
  - Why only random failures?
  - Answer: Gnutella forms a power-law network

Side note: Power Law Networks

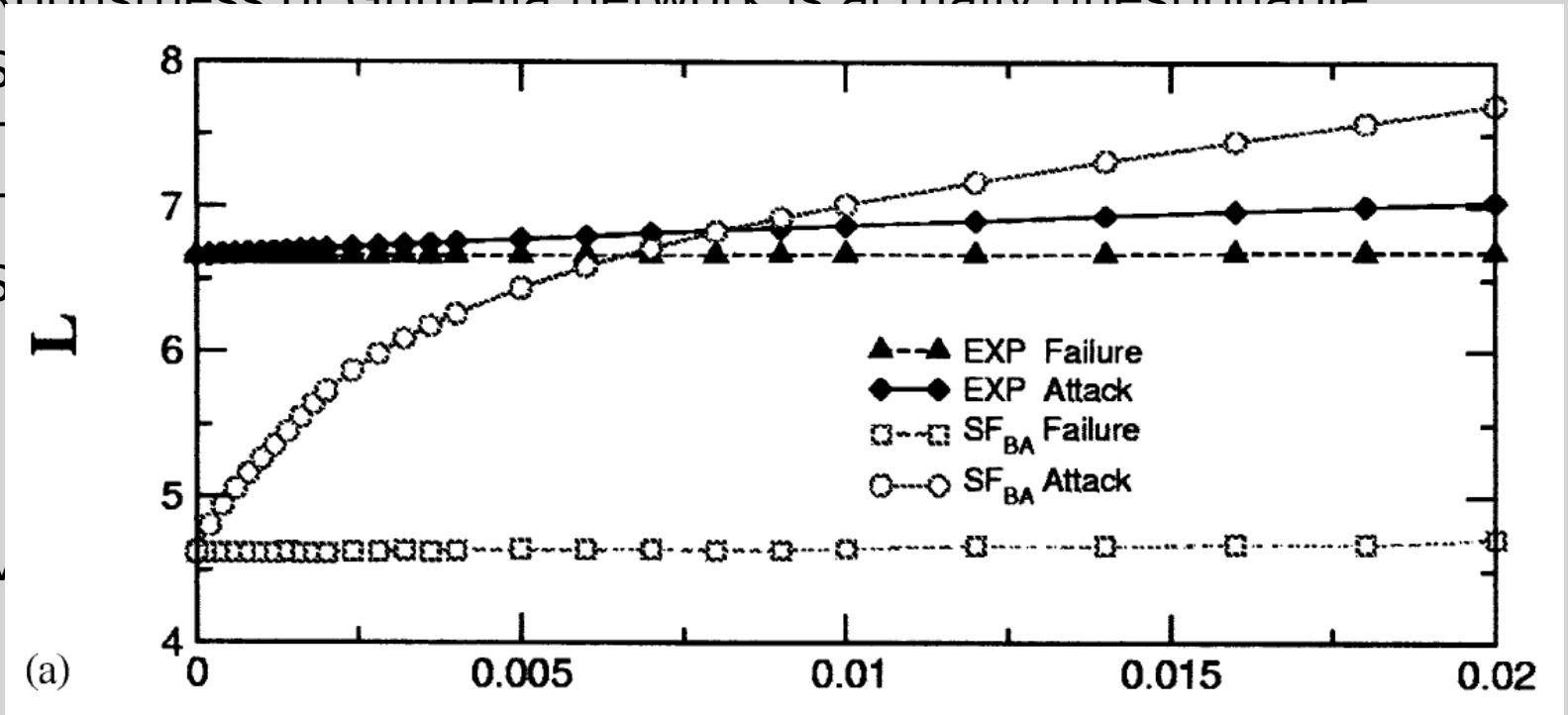
- Power law:

$$y = ax^k$$

- Power laws are very common in nature
- Internet also follows some power laws
  - Popularity of Web pages (cf. Zipf's law for English words)
  - Connectivity of routers and Autonomous Systems
  - Gnutella's topology ;-)
- Has been shown:
  - In a network where new vertices (nodes) are added and new nodes tend to connect to well-connected nodes, the vertex connectivities follow a power-law
  - In Gnutella's case, the exponent is 2.3 (actually -2.3)

Robustness in Power Law Networks

- Networks with power law exponent < 3 are very robust against random node breakdowns
- Robustness of Gnutella network is actually questionable
- S
- T
- T
- S
- M



(cf.) Crucitti et al.: Efficiency of Scale-Free Networks: Error and Attack Tolerance, Elsevier

(*) Albert et al.: Error and attack tolerance of complex networks, nature 406.

(*) http://www2.econ.iastate.edu/classes/econ308/tesfatsion/NetworkIntro.LT.htm

Gnutella Weaknesses

- Flooding a query is extremely inefficient
  - Wastes lot of network and peer resources
- How can we deal with that?
  - State (don't flood the same message twice, don't flood backwards)
  - Search horizon (6 degrees of separation: limit query radius)
- Gnutella's network management not efficient

How does this compare to IP and routing on the Internet?

- Queries in Gnutella not very efficient
  - Limited query radius
  - Only a subset of peers receives query
  - Only files at those peers can be found

FastTrack (a.k.a) KaZaA

- FastTrack (KaZaA, or also Kazaa) changed the game
  - Completely new architecture
  - Many networks followed in KaZaA's footsteps
- On a typical day, KaZaA had:
  - Over 3 million active users
  - Over 5000 terabytes of content (even 29000 TB?)
- KaZaA based on a supernode-architecture
  - All recent architectures are based on the same idea
- Many important lessons from KaZaA
  - Exploit heterogeneity of peers
  - Organize peers into a hierarchy

KaZaA History

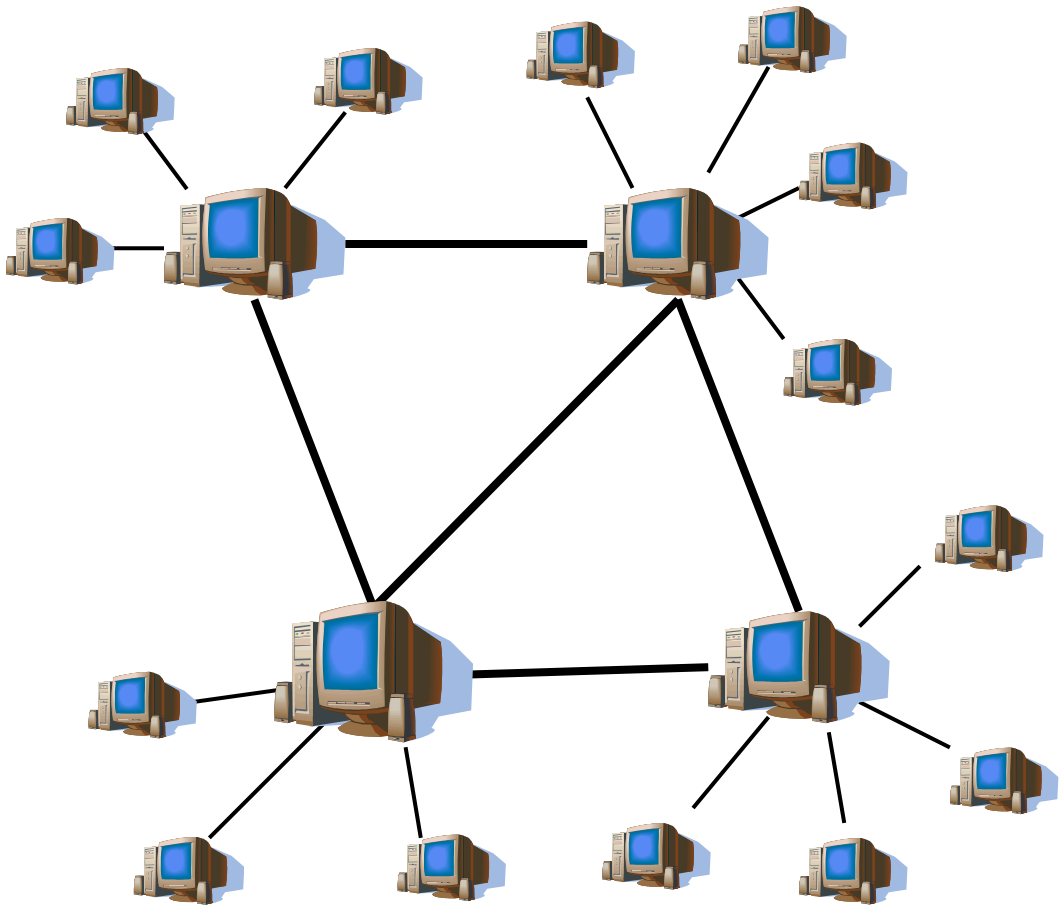
- KaZaA uses FastTrack protocol
  - FastTrack was also used by MusicCity and Morpheus
- KaZaA created in March 2001 (Niklas Zennström (*))
  - Company was called Consumer Empowerment (Dutch company)
- November 2001, KaZaA moved out of Netherlands
  - Result of law suit in Netherlands
  - Main holder became Sharman Networks (in Vanuatu)
- In March 2002, earlier judgment reversed
- Lawsuits also followed in other countries
  - USA (California), Australia
- Judgment in June 2006 against Sharman Networks
  - Settled by paying $100 M and convert Kazaa into a legal service

(*) Consumer Empowerment->Joltid; FastTrack, Skype, Joost

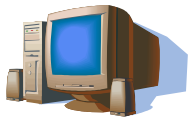
KaZaA: How It Works?

- Two kinds of nodes in KaZaA:
  - Ordinary nodes (ON)
  - Supernodes (SN)
- ON is a normal peer run by a user
- SN is also a peer run by a user, but with more resources (and responsibilities) than an ON
- KaZaA forms a two-tier hierarchy
  - Top level has only SN, lower level only ON
- ON belongs to one SN
  - Can change at will, but only one SN at a time
- SN acts as a Napster-like “hub” for all its ON-children
  - Keeps track of files in those peers (and only those peers)

KaZaA: Hierarchy



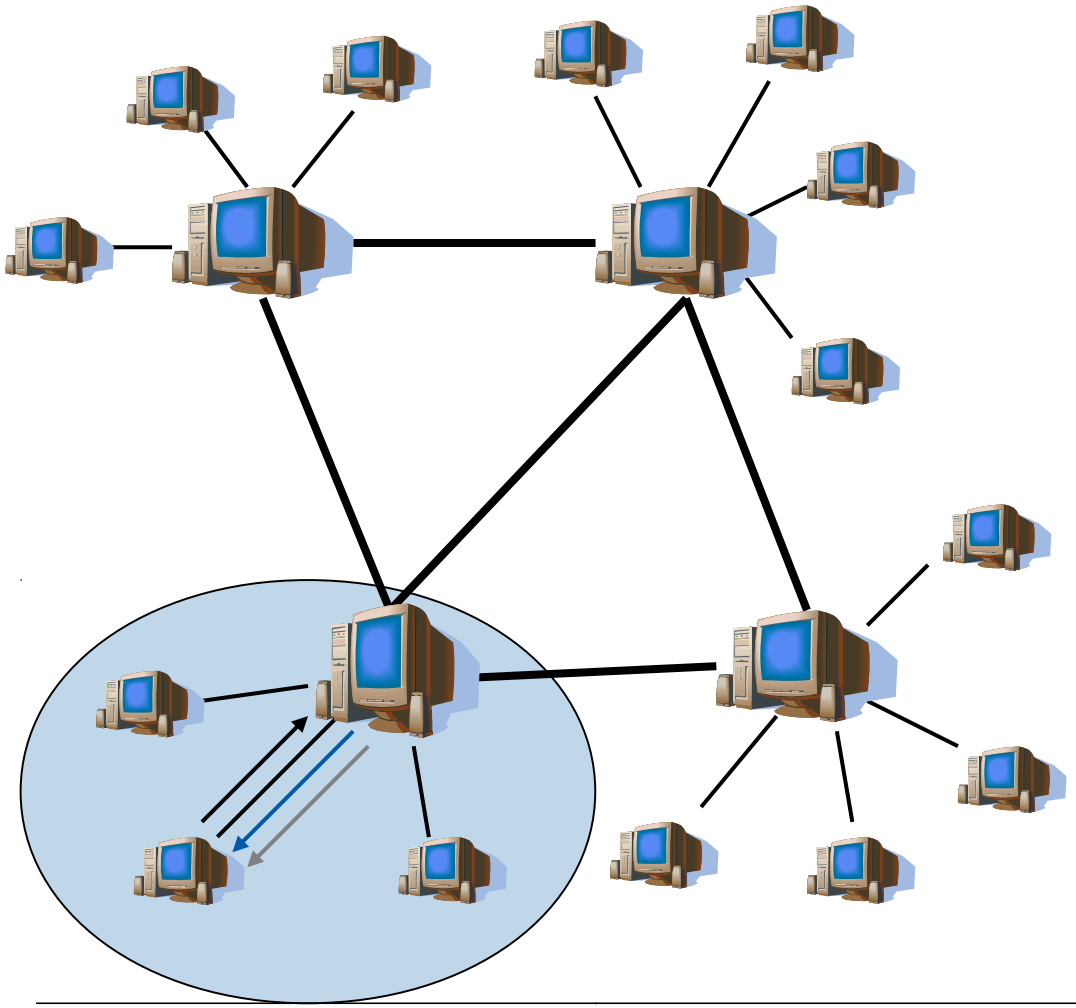
Ordinary Node



SuperNode

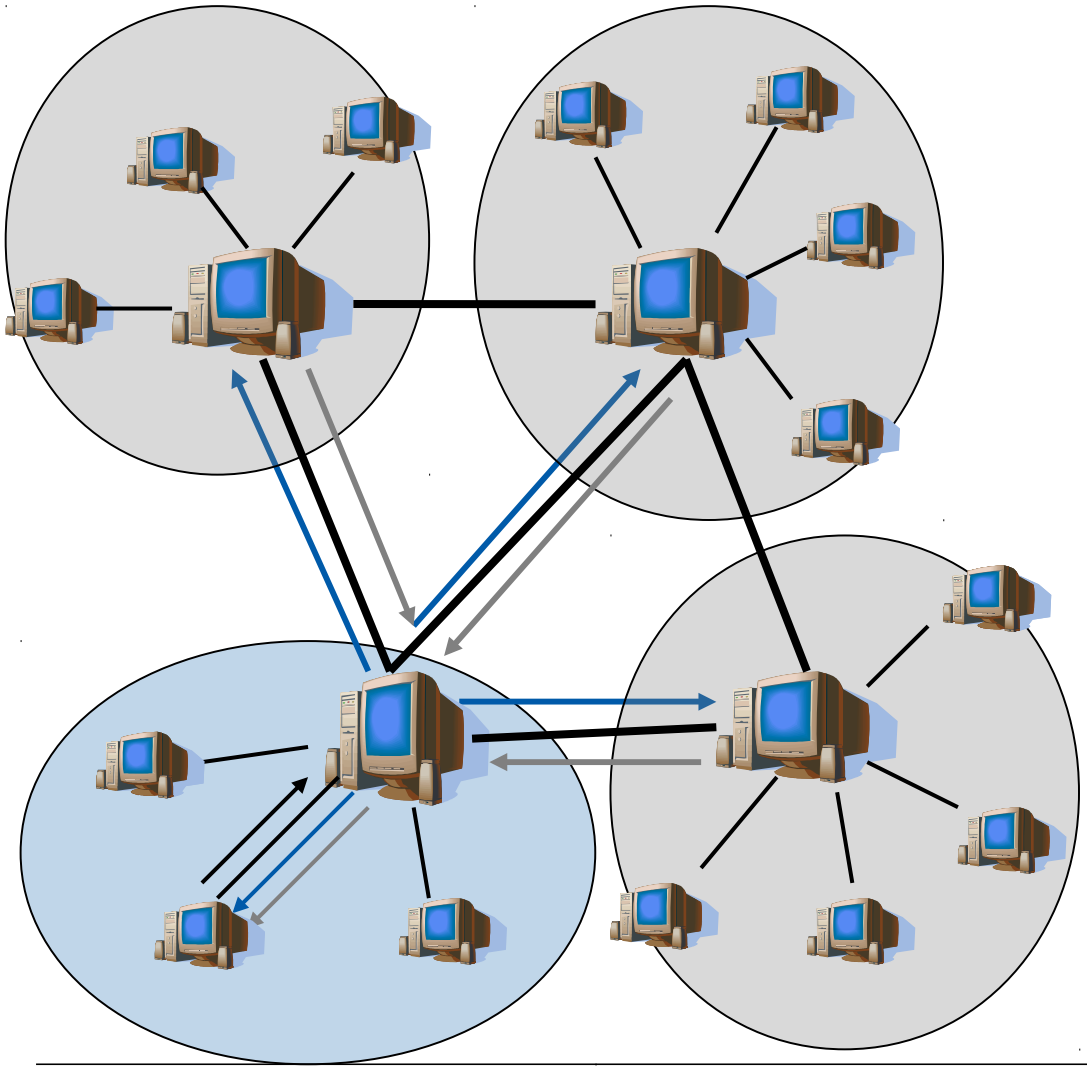
- Ordinary nodes belong to one Supernode
  - Can change SN, but not common (Kazaa-Lite)
- Supernodes exchange information between themselves
- Supernodes do not form a complete mesh

KaZaA: Networking



- Peer obtains address of SN from “somewhere”
  - Bootstrap server or included in software
- Peer sends request to SN, gives list of files to share
- SN starts keeping track of this peer
- Other SN not aware of the new peer

KaZaA: Finding Stuff



1. Peer sends query to its own supernode
2. Supernode answers for all of its peers and forwards query to other supernodes
3. Other supernodes reply for all of their peers

KaZaA: Downloading

- KaZaA (re-)introduces chunks (split large files into small pieces)
- Files are identified using „UUHash“
  - Hash first 300KB → 128 bit
  - CRC32 each 300KB at every 2nMB offsets
  - CRC32 of last 300KB of file
  - Concatenate 128bit + result of CRC32 → 160bit
- Requesting peers download from multiple sources

KaZaA: Ordinary vs. Super Nodes

- ON can be promoted to SN if it demonstrates sufficient resources (bandwidth, time on-line)
  - User can typically refuse to become a SN
  - Typical bandwidth requirement for SN 160-200 kbps

OK for cable and universities, but problem for DSL, let alone modems!

- SN change connections to other SN on a time scale of tens of minutes
  - Allows for larger range of network to be explored
  - Average lifetime of SN 2.5 hours, but variance is high
- SN does not cache information from disconnected ON
- Estimated 30,000 SN at any given time
  - One SN has connections to 30-50 other SN
- 13% of ON responsible for 80% of uploads