HELSINGIN YLIOPISTO
HELSINGFORS UNIVERSITET
UNIVERSITY OF HELSINKI

Peer-to-Peer Networks

Chapter 2: Current Peer-to-Peer Systems

Chapter Outline

- Overview of deployed P2P systems in 4 areas
- P2P file sharing and content distribution:
  - BitTorrent, Napster, Gnutella, KaZaA
  - Differences, strengths, weaknesses
- P2P Communication
  - Typical instant messaging setup
  - Skype
- P2P Computation
  - SETI@Home example
- P2P Collaboration
  - Collaboration according to the P2P principle

Current P2P Content Distribution Systems

- Most current P2P content distribution systems targeted at one application: File sharing
- Users share files and others can download them
- Content typically music, videos, or software
  - Also often illegally shared… :-(
  - Legal uses becoming more common? (see BitTorrent)
- Content distribution has made P2P popular

- Note: Distinguish between name of network (e.g., Gnutella) and name of client (e.g., LimeWire)

BitTorrent

- BitTorrent is a new approach for sharing large files
- BitTorrent used widely also for legal content
  - For example, Linux distributions, software patches
  - Official movie distributions are also happening (WB)
- Goal of BitTorrent:
  - Quickly replicate one file to a large number of clients
- File sharing networks attempt to provide as much content for download
- BitTorrent more appropriately called peer-to-peer content distribution
- BitTorrent has also had its share of litigation

P2P Content Distribution

- BitTorrent builds a network for every file that is being distributed

Big advantage of BitTorrent:

- Can send "link" to a friend
- "Link" always refers to the same file
- Same not really feasible on Napster, Gnutella, or KaZaA
  - These networks are based on searching, hard to identify a particular file
  - Downside of BitTorrent: No searching possible
    - Websites with "link collections" and search capabilities exist

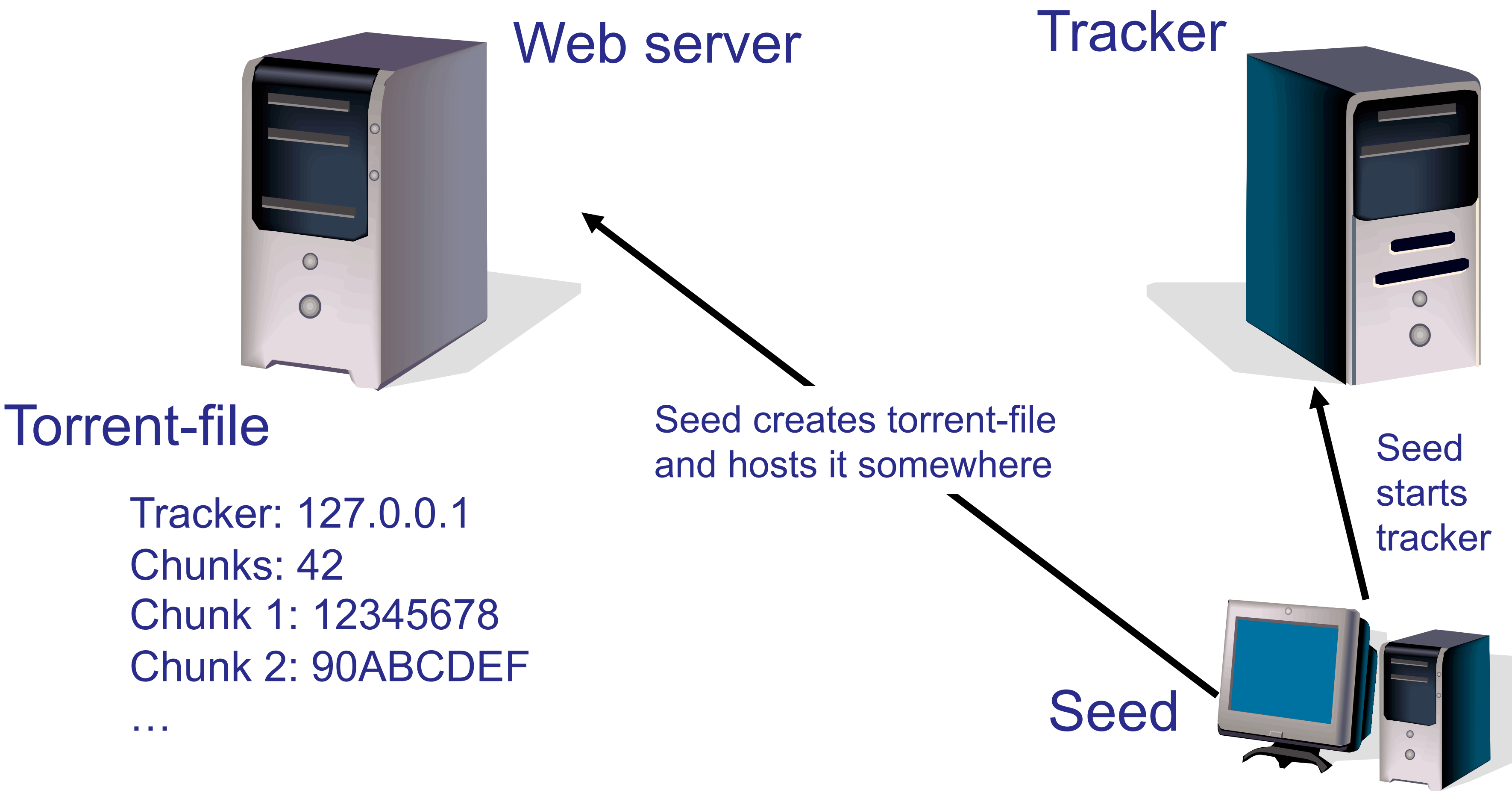
BitTorrent History

- BitTorrent developed by Bram Cohen in 2001
  - Written in Python, available on many platforms
- Uses old upload/download-ratio concept from BBSs
  - “The more you give, the more you get”
  - Participation enforced in protocol
  - Other P2P systems have adopted similar ideas
- Why BitTorrent originally had little illegal content?
  - No search functionality?
  - Original source easily identified?
  - Currently lots of illegal content on BitTorrent too…

BitTorrent: How it Works?

- For each file shared on BitTorrent, there is (initially) one server which hosts the original copy
  - File is broken into chunks
- A “torrent” file which gives metadata about the file
  - Torrent file hosted typically on a web server
- Client downloads torrent file
  - Metadata indicates the sizes of chunks and their checksums
  - Metadata identifies a tracker
- Tracker is a server which tracks the currently active clients
  - Tracker does not participate in actual distribution of file
  - Law suits against people running trackers have been successful, even though tracker holds no content (see later Chapter…)

BitTorrent: Players



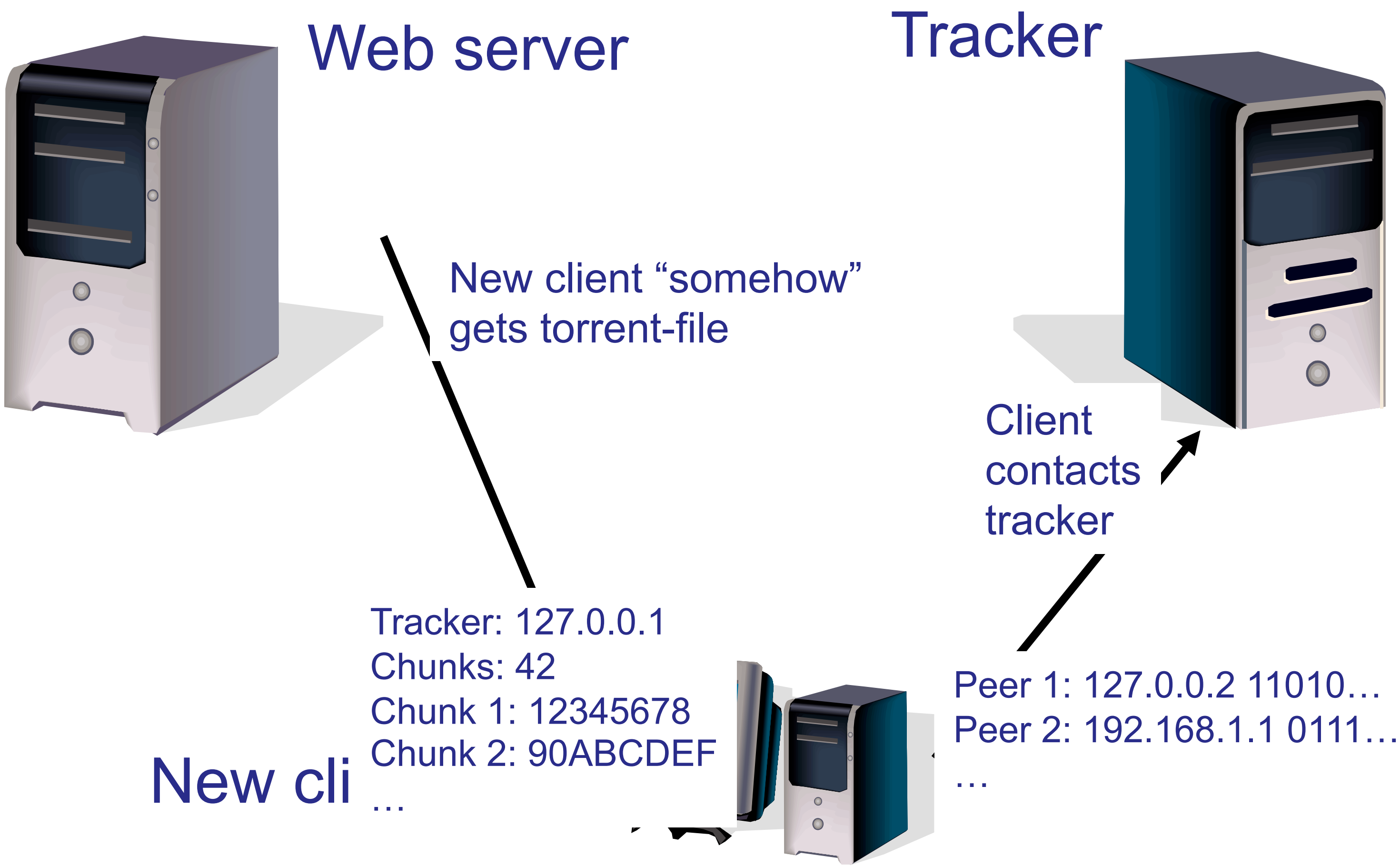
- 3 entities needed to start distribution of a file

BitTorrent: How it Works?

- Terminology:
  - Seed: Client with a complete copy of the file
  - Leecher: Client still downloading the file

- Client contacts tracker and gets a list of other clients
  - Gets list of 50 peers
- Client maintains connections to 20-40 peers
  - If number of connections drops below 20, it contacts tracker
- This set of peers is called peer set
- Client downloads chunks from peers in peer set and provides them with its own chunks
  - Chunks typically 256 KB
  - Chunks makes it possible to use parallel download

BitTorrent: Starting Up



- New client gets torrent-file and gets peer list from tracker

BitTorrent: Tit-for-Tat

- BitTorrent uses tit-for-tat policy
- A peer serves peers that serve it
  - Encourages cooperation, discourage free-riding

- Peers use rarest first policy when downloading chunks
  - Having a rare chunk makes peer attractive to others
  - Other wants to download it, peer can then download the chunks it wants
  - Goal of chunk selection is to maximize entropy of each chunk
- For first chunk, just randomly pick something, so that peer has something to share

BitTorrent: Choke/Unchoke

- Peer serves 4 peers in peer set simultaneously
  - Seeks best (fastest) downloaders if it’s a seed
  - Seeks best uploaders if it’s a leecher
- Choke is a temporary refusal to upload to a peer
  - Leecher serves 4 best uploaders, chokes all others
  - Every 10 seconds, it evaluates the transfer speed
  - If there is a better peer, choke the worst of the current 4
- Every 30 seconds peer makes an optimistic unchoke
  - Randomly unchoke a peer from peer set
  - Idea: Maybe it offers better service
- Seeds behave exactly the same way, except they look at download speed instead of upload speed

BitTorrent: Strengths

- Works quite well
  - Download a bit slow in the beginning, but speeds up considerably as peer gets more and more chunks
- Users keep their peers connected as seeds
  - Legal content, so no need to worry?
  - Large download, leave running over night?
  - How necessary is this?
- Those who want the file, must contribute
  - Attempts to minimize free-riding
- Efficient mechanism for distributing large files to a large number of clients
  - Popular software, updates, …
  - See also Avalanche from Microsoft Research

BitTorrent: Weaknesses

- File needs to be quite large
  - 256 KB chunks
  - Rarest first needs large number of chunks
- Everyone must contribute
  - Problem for clients behind a firewall?
  - Low-bandwidth clients have a disadvantage?

BitTorrent: Open Issues

- What is the impact of BitTorrent on the network?
  - Fast download != nearby in network (at least not always)
  - Topic of on-going research
  - Preliminary results underline importance of selecting nearby peers for downloading
- What is the optimal chunk selection algorithm?
  - Rarest-first seems to work well in practice
    - Beginning of download, endgame mode, …
  - Is it also optimal?
  - What is optimal? Fastest for single peer? Overall fastest?
- Is tit-for-tat really necessary?
  - Are there situations where free-riding should be allowed?
  - *Are there situations where free-riding should be encouraged?*

Freeriders: Problem or Not?

- Freerider is someone who does not contribute
  - Sometimes: Contributes much less than consumes
- Measurement in original Gnutella:
  - 80% of users share little or no files at all
  - Even among the remaining 20%, sharing uneven
- "Rash" conclusion: We must do something about this!
- WHY?!?
- "Logic": It's not fair!

- True, but is "fairness" the right thing to aim for?
  - How do you define fairness?
- How about optimizing system performance?

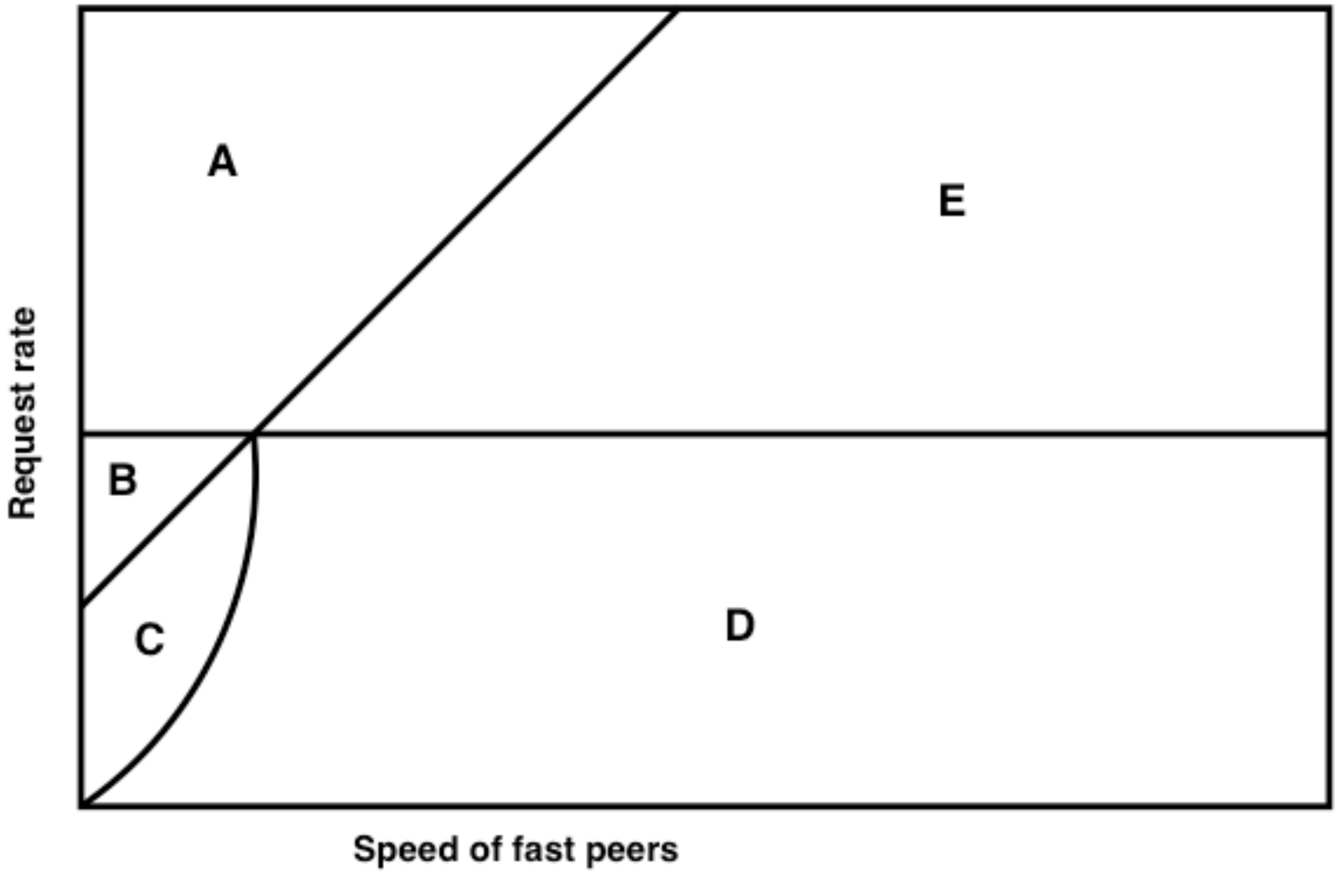
Are Freeriders Really a Problem?

- Short answer: Usually not
- Long answer starts here…
- First, let's look at queueing theory: (classic example)
  - Two printers, fast and slow + standard Poisson assumptions about arrivals and service times
  - ➔ You always send print job to fast printer
  - On average you win (as does everyone)
- So what's the relationship to BitTorrent?
- We have two peers: fast and slow
- Where do you want to download from?
- Duh, the fast one of course…
- So: Why should the slow peer even offer the file?

Let's Test This in Practice

- 2 peers, fast and slow, want to download 1 chunk
  - Exponential inter-request times, deterministic service times
  - Model as M/D/1 queue
- Vary arrival and service rates
- Question: How should we split requests between fast and slow peer?
- Can identify 5 possible cases:
  - A. Request rate too high to handle, nothing works
  - B. Both peers must participate
  - C. Every configuration is possible, best if both participate
  - D. Every configuration possible, best if only fast sends
  - E. Only fast peer is possible

Graphically Speaking



- Sizes of regions vary according to details
- Most of the time we have case D or E (= only fast peer)

Freeriding in General

- Same kind of reasoning can be pushed further
- Three main findings:

1. Freeriding is bad when:
   - Request rate extreme
   - Number of freeriders extreme (over 90%)
2. Freeriding is technically bad, but not noticeable
   - Moderate to high freeriders (50-80%)
   - Increase in download times negligible (~ few % at most)
3. Freeriding is beneficial to everyone
   - Slow peers do not offer anything
   - Large gains for everyone!

Freeriding: Recap

- Real-world systems exhibit lot of freeriding
- Gut reaction: Must do something!

- Reality: Not really a major problem to begin with
- Reality: Can be highly beneficial to everyone

- What happens when fast peers become freeriders?
  - This is of course very bad for everyone…

- Topic for future research: Incentives
- Remember: Forced contributions from everyone not necessary the best thing to do

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

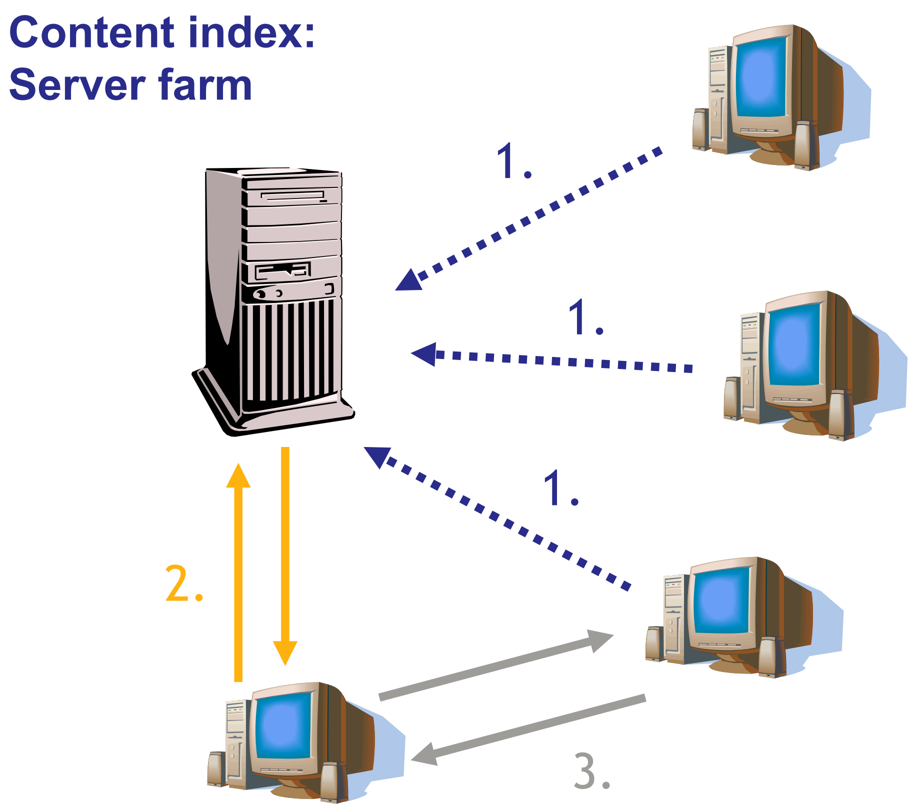
History of Napster

- 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
  - Actually a server farm
- User registers with the central server
  - Give list of files to be shared
  - Central server 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
  - For example, encoding rate, size of file, peer’s bandwidth
  - Some information entered by the user, hence unreliable

Napster: Queries



1. Peers register with central server, give 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
- 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

- 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)
  - 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
- But Gnutella introduces its own problems

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

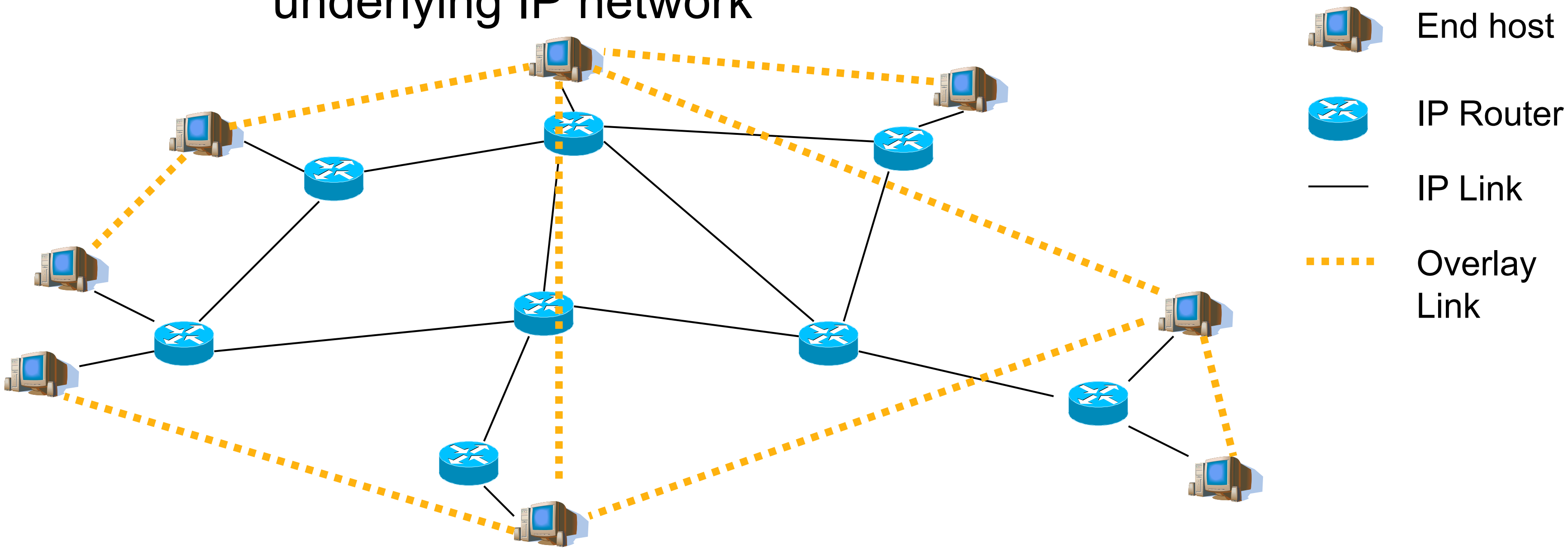
- Gnutella is at the opposite end of the spectrum
  - Napster is centralized
  - Gnutella is fully distributed

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 out
- Version 0.4 is covered here (= original Gnutella version)
- Current version 0.6 is similar to 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

Gnutella: Overlay Network

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

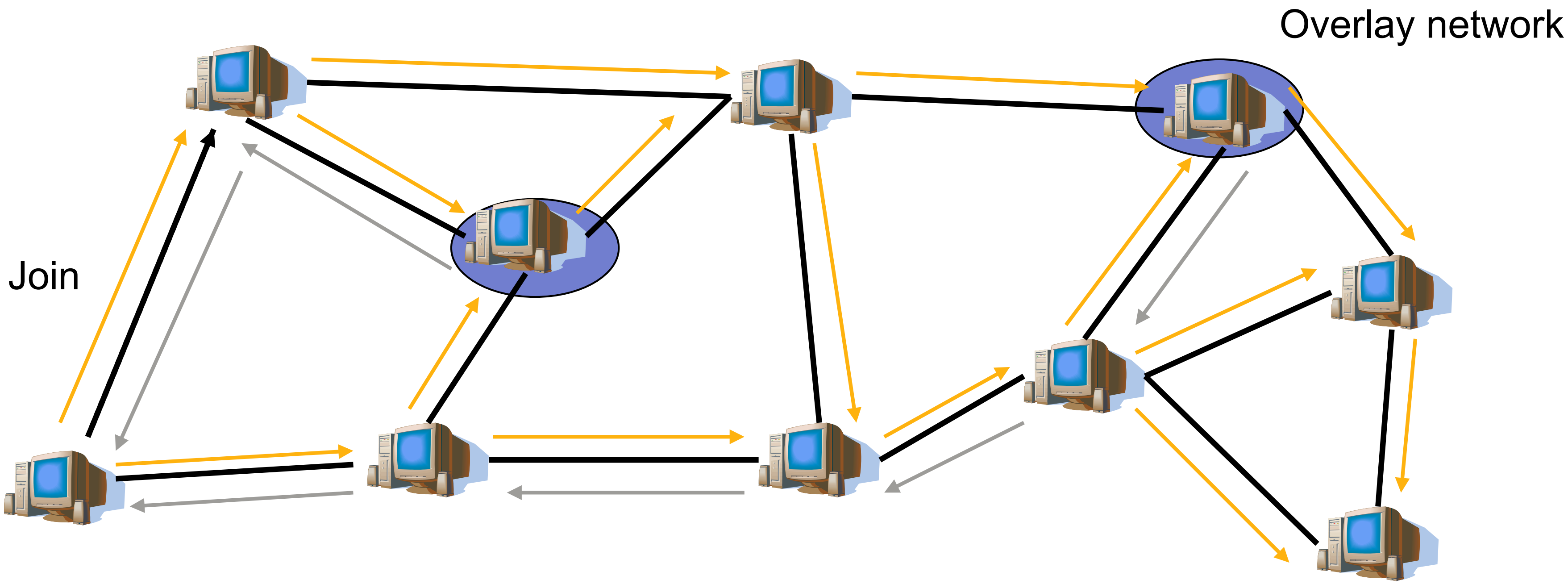


Most current P2P systems based on some kind of overlay

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…
  - 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

- 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
- Gnutella has PUSH message
  - Peer outside firewall sends PUSH to peer inside firewall
    - 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 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
- Subset of Gnutella with 1771 nodes
- Take out random 30% of nodes, network still survives
- Take out 4% of best connected nodes, network splinters
- Still, Gnutella survives attacks (all kinds) better than Napster

- More on power law and other kinds of networks later

Gnutella Weaknesses

- Flooding a query is extremely inefficient
  - Wastes lot of network and peer resources
  - Solution: Limit query radius
- Gnutella's network management not efficient
  - Periodic PING/PONGs consume lot of resources
- Queries in Gnutella not very efficient
  - Limited query radius
  - Only a subset of peers receives query
  - Only files at those peers can be found

KaZaA

- KaZaA (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
  - Currently all recent architectures are based on a similar idea
- Many important lessons from KaZaA
  - Exploit heterogeneity of peers
  - Organize peers into a hierarchy

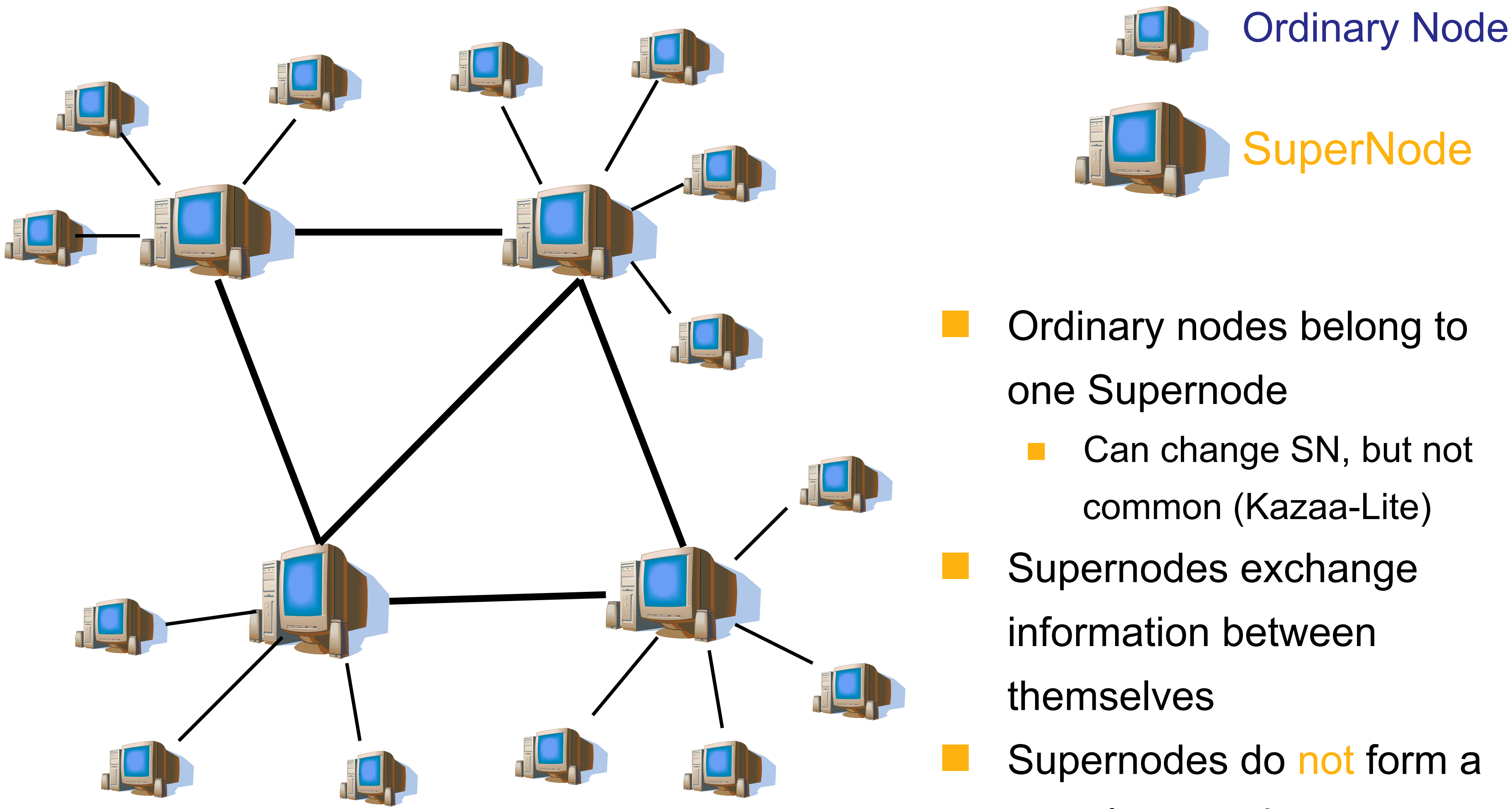
KaZaA History

- KaZaA uses FastTrack protocol
  - FastTrack was also used by MusicCity and Morpheus
- KaZaA created in March 2001 (Niklas Zennström)
  - Company was called KaZaA BV (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
  - California, Australia
- Judgment in June 2006 against Sharman Networks
  - Settled by paying $100 M and convert Kazaa into a legal service

KaZaA: How it Works

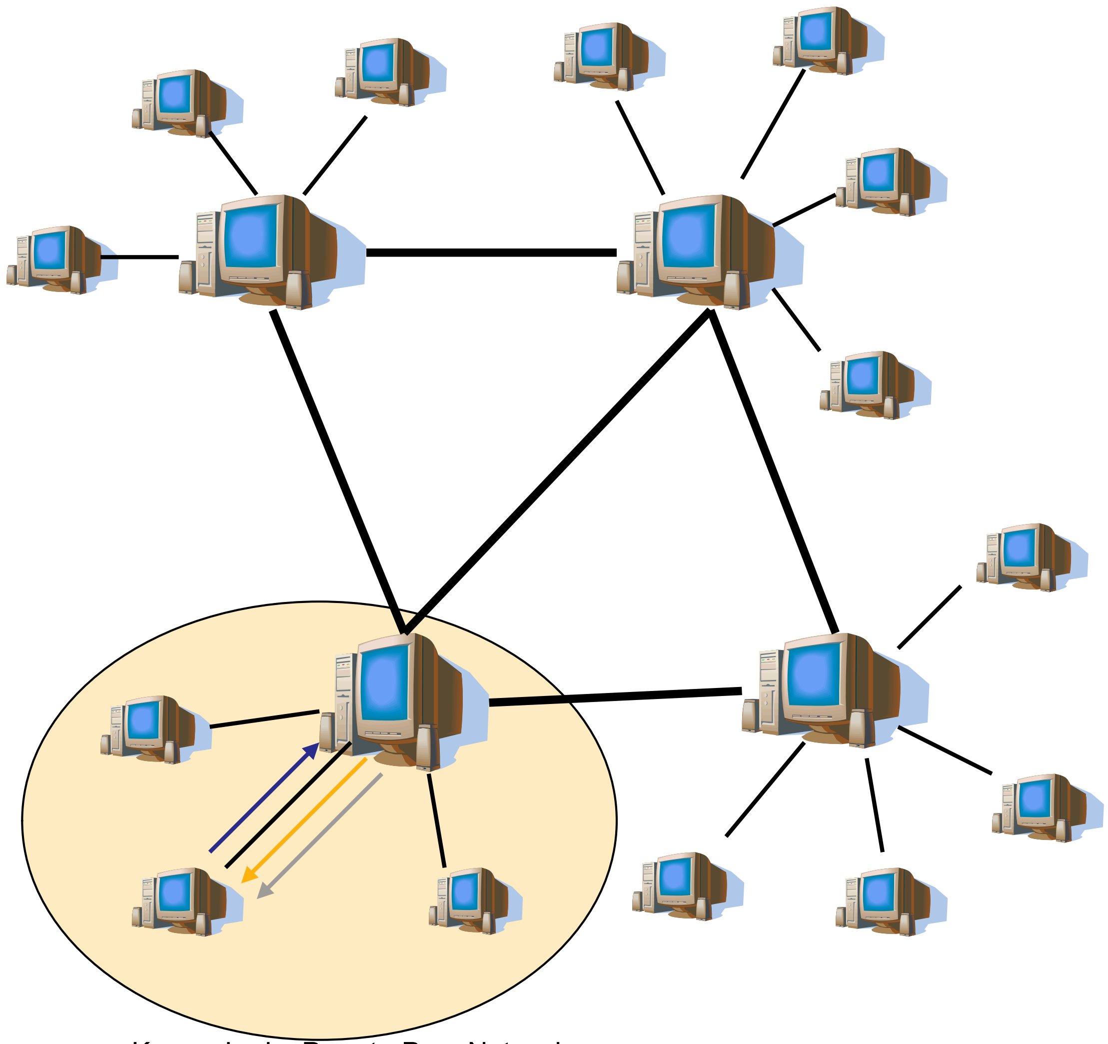
- Two kinds of nodes in KaZaA:
  - Ordinary nodes (ON)
  - Supernodes (SN)
- ON is a normal peer run by a user
- SN is also a peer run by a user, but with more resources (and responsibilities) than an ON
- KaZaA forms a two-tier hierarchy
  - Top level has only SN, lower level only ON
- ON belongs to 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



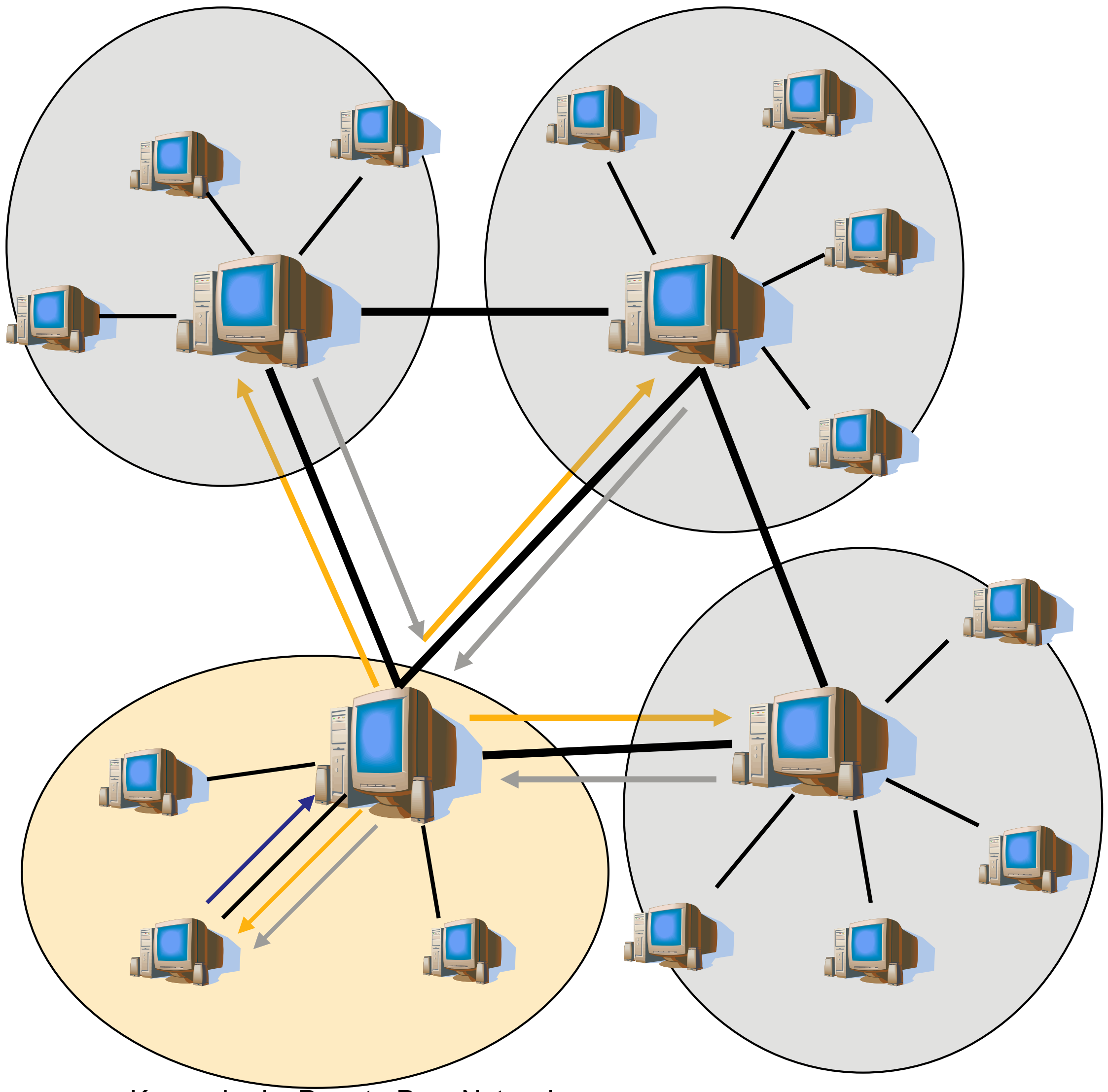
- Ordinary nodes belong to one Supernode
  - Can change SN, but not common (Kazaa-Lite)
- Supernodes exchange information between themselves
- Supernodes do not form a complete mesh

KaZaA: Building the Network



- 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: Inner Workings

- 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!
- 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

KaZaA: Spyware

- KaZaA included spyware in their program
- Spyware does things like:
  - Sends all DNS queries to a tracking server
  - Monitors visited websites
  - Additional popup windows on “partner” sites
- KaZaA originally denied existence of spyware
- In theory, possible to disable spying functions
  - But removal software reportedly failed often…

KaZaA: Strengths

- Main strength of KaZaA: Combines good points from Napster and Gnutella
  - Efficient searching under each supernode
  - Flooding restricted to supernodes only
  - Result: Efficient searching with “low” resource usage
- Most popular network (globally)
  - Lot of content, lot of users
  - Some networks more popular in some areas (e.g., eDonkey in Germany)
  - Currently most big file sharing networks have been shut down

KaZaA: Weaknesses

- Queries not comprehensive
  - Can still miss a file even though it exists
  - But better reach than Gnutella
- Single point of failure?
  - Lawsuits against KaZaA eventually successful
  - Software comes with list of “well-known” supernodes
    - Increases robustness?
    - More targets for lawyers?
- In general, solves many problems of Napster and Gnutella

Napster vs. Gnutella vs. KaZaA

	Napster	Gnutella	KaZaA
Type of Network	Centralized	Distributed	Hybrid
Efficient Searching	+++	---	+
Resilience to Attacks	---	++?	+
Open Protocol	N	Y	N
Spyware-free	Y	Y	Y/N?
Popularity	+++	-	+++

For Your Long Term Memory

- Many different kinds of file sharing networks
- Old ones go, new ones come (pace slowing down?)

- Three main architectural solutions for indexing
  - Centralized index
  - Distributed index
  - Hybrid index

- File sharing networks also called unstructured
  - Content can be placed anywhere in the network
- Contrast: Structured networks
  - Every file has a well-defined place
  - See DHTs in Chapter 3

File Sharing: Current State

- Most bigger file sharing networks sued into submission
  - Napster, Kazaa, eDonkey, …
- Many networks still up and running
  - Because of many open clients available
- Future is uncertain
- Content owners (record companies and movie studios) are moving into online delivery of content
  - iTunes and others for music
  - iTunes, Amazon for movies and TV content
- Remains to be seen… Stay tuned!

P2P File Sharing: Summary

- File sharing networks extremely popular
  - Different networks come and go
- File sharing based on keyword searches
  - Keyword matches either file name or metadata
  - Must use same keywords as provider
    - Usually not a problem
- No guarantees about file being what it claims to be
  - Record companies inject files with dummy content
  - Solution: Each file has hash, make public list of “bad files”
- Future looks uncertain

P2P Communications

- P2P communications are a communication architecture based on P2P principle
- Examples: Email, network news, instant messaging, telephony
- Current email and news systems are P2P to some degree
  - See below for details

- P2P communications aim at bringing people together
  - Remove intermediate servers
  - “P in P2P means people” (D. Wiener, Userland)
- Possible to implement all forms of communication

P2P Communications Example: IM

- Typical instant messaging system is P2P
  - Centralized server has buddy lists
  - User logs on to central server, sees buddies on-line
- Chatting directly between peers
  - Including audio, video, and file transfers
- Role of centralized server: (similar to Napster)
  - Bring people together
  - Centralized server also helps with firewalled clients

P2P Communications: Email and News

- Current email and news systems have P2P components
- In Email, Mail Transfer Agents (MTA, mail servers) exchange email directly between them
  - No central coordination, except through DNS
  - Automatic transfer of messages, according to DNS MX records
- In News, NNTP servers exchange articles between them to build news feed
  - Again, no central coordination except DNS
  - Feeds typically set up through agreements between admins
- From user's point of view, P2P is hidden
  - User always has to access the same mail server to get her mail
  - Same for news (although technically this could be avoided…)

P2P Communications: Skype

- Skype is a popular Internet telephony software
- Allows the user to
  - Make calls to other computers on Internet
  - Make calls to real phone network (costs money)
  - Have calls made to a real phone number forwarded to Skype (also costs money)
- Skype developed by same people as KaZaA
  - Big difference: Skype is perfectly legal
- Architecture of Skype very similar to that of KaZaA
  - Supernodes and ordinary nodes
- Skype is very popular, ~200 million downloads, ~10 million concurrent users online
- Clients for Windows, Mac, Linux

Skype: Details

- Skype is a proprietary and encrypted protocol
- No real details available :-(
- Best study about Skype: “it sends 48 bytes over TCP to some IP address, then 512 bytes to this address”…
- What is known from Skype:
  - One central server for login and billing
  - Supernodes behave much the same way as KaZaA
    - Normal nodes connect to SN, etc.
  - Directory of who is online is spread over the peers
    - Details unknown, Skype claims that system knows all users who were online in the last 72 hours
  - Skype claims to go through firewalls
    - As long as firewall allows (some) outgoing connections

Skype: Supernodes and Calls

- Supernodes (and some other nodes?) have more responsibilities in Skype than in KaZaA
- Supernodes are responsible for forwarding actual data traffic (calls) between (firewalled) peers
- No (easy) was to disable this in client software
  - Configure your own firewall, restrict Skype’s bandwidth, …

- One big advantage of Skype is high call quality
  - Better than normal telephone in many cases!
- Skype has highly efficient voice codec
  - About 5 kB/s of traffic generated (even during silence)

P2P Computation

- P2P computation is peers doing computations for others
  - Others = typically central administrator, not “other peers”
- Computationally intensive problem to solve
  - For example, crack encryption or find messages from aliens
  - Problem needs to be easily distributed to peers
- Typically, centralized server handles problem-solving
  - Distributes work to peers
  - Peers only perform their computation, send back result
  - Each peer contributes at its own speed
  - Results verified somehow (problem dependent)
- Usually no special reward for participation
  - Common goal for all peers
- Uncontrolled and un-administered
  - Peers free to join and leave when they wish, contribute what they want

Why P2P Computation Works?

- P2P computation works because common goal appeals to people running peers
  - Read: People do it because they think it is worthwhile
- People participating are “techno-nerds”
  - Cracking encryption and SETI@Home are “cool”
  - Common, non-profit purpose
  - Often run on campuses and dorms (= lot of “free” computers)
- What if run by a private company for proprietary purposes?
  - For example, a car company wants to model a wind tunnel
  - Or military wants to simulate a nuclear detonation
- *Is it possible to build a P2P computation system where users are paid for their contributions?*

P2P Computation: Example

- Several P2P computation projects active
- SETI@Home, distributed.net, etc.
- SETI@Home project run from UC Berkeley
  - Now many projects under BOINC (Berkeley Open Infrastructure for Network Computing)
- Search for Extraterrestrial Intelligence (SETI):
  - Goal of SETI project is to discover signals from extraterrestrial civilizations
  - SETI@Home uses P2P computation to identify those signals
- Why P2P (distributed) computation is needed in SETI?
  - Signal parameters are unknown, sensitivity of search depends on available computation power
  - Need to scan large frequency bands, correct for Doppler shift, filter out local interference (from Earth)

SETI@Home

- SETI scans 2.5 MHz wide band around 1,420 MHz
  - Assumed to be universally of interest (hydrogen line)
- Total amount of data from survey expected to be 1100 tapes of 35 GB each, total about 39 TB of data
- Data divided into work units at UC Berkeley
  - Work units sent to clients
  - Client can work offline, takes several hours per work unit
- Clients reply with results from computation
  - Each work unit calculated by several clients
    - Undetected errors occur once every 10^{18} machine instructions
    - SETI would see several such errors per day!
    - Communication errors
- Communications over HTTP
  - For clients behind a firewall

SETI@Home: Some Old Numbers

- No alien signals detected yet ☹ (or ☺?)
- Client available for 47 OS/hardware combinations
- Millions of users (5,213,884 in 2004)
  - Users organized in teams
  - Teams “compete” against each other
  - SETI relies on volunteers, no rewards offered
    - Except prestige from being in “leading team”
    - And the distant possibility of finding a signal…
- Total CPU time: 2,095,302 years (2004)
- Average CPU time per work unit: 11 h 29 min
- New signals added faster than they are processed

P2P Collaboration

- P2P collaboration is users sharing their resources for a common project
  - Resource typically time (of the person)
- For example online encyclopedias, e.g., Wikipedia
  - Individual users write articles, can edit articles from others
  - Guiding principle: Enough many iterations result in a factually correct and unbiased article (?)
- Differs from Computer Supported Collaborative Work
  - CSCW aimed more at immediate collaboration
  - Meetings, video conferencing, shared whiteboards, …
- One of best examples of P2P principle in action
- Is open source software development P2P collaboration?

Chapter Summary

- P2P systems in active use in many areas
  - Main focus in content distribution (file sharing networks)
- Show well properties of P2P principle
  - Autonomous
  - Exploit edge resources
  - Intermittent connectivity
- Different system in different areas (content distribution, communication, computation, collaboration)
  - Several different file sharing networks, each with good and bad points
  - Several communication networks
  - Many computation projects
- No single system ruling over others