Peer-to-Peer Networks

Chapter 2: Initial (real world) systems

Thorsten Strufe

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

Current P2P Content Distribution Systems

- Most intial 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., BitTorrent) and name of client (e.g., Vuze)

BitTorrent

- BitTorrent is an approach to sharing large files
- BitTorrent used widely also for legal content
  - For example, Linux distributions, software patches
  - Official movie distributions (WB)
  - BBC series over the Internet (iPlayer), Octoshape
  - Game distribution (Pando Networks, Akamai’s Netsession Interface)
- Goal of BitTorrent:
  - Quickly and reliably replicate one file to a large number of clients
- BitTorrent more appropriately called “peer-to-peer content distribution”

P2P Content Distribution

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

Big advantage of BitTorrent:

- Can send “link” (.torrent) to a friend
- “Link” always refers to the same swarm -> the same file (remember identifiers?)

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

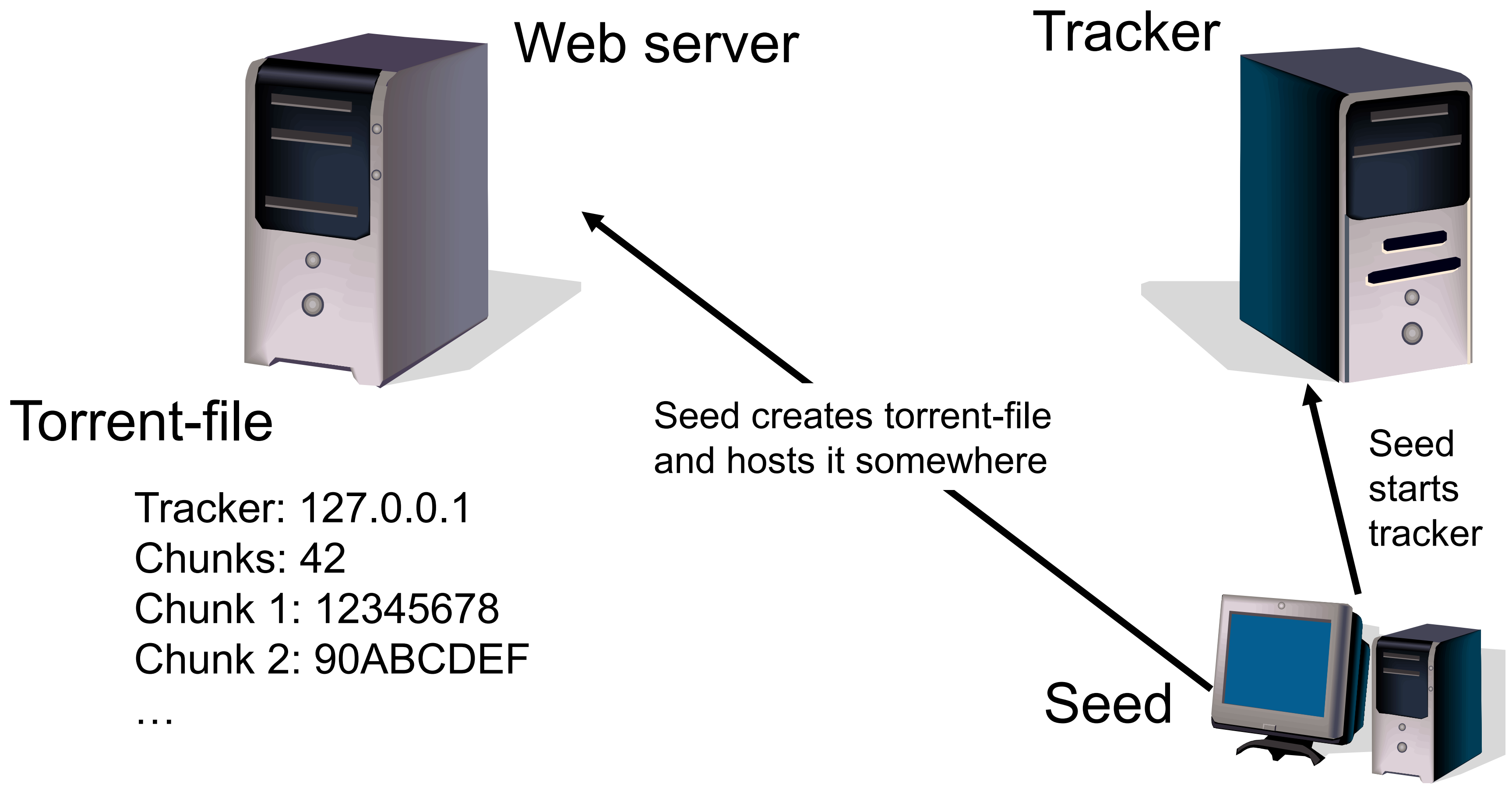
- BitTorrent originally used only seldom for illegal content
  - No search functionality?
  - Original source easily identified?
  - Currently lots of illegal content on BitTorrent too (decreasing)…

BitTorrent: How does it Work?

- Components:
  - Location Service
  - Tracker
  - Peers (Seed)
- Data:
  - shared file (broken into chunks)
  - “torrent” file contains metadata about the shared file
    - Tracker, size and checksum of chunks
- Cycle:
  - 1st Seed creates torrent file, publishes via tracker and location service
  - Client discovers resource, downloads torrent file (“name service”)
  - Registers with, and retrieves peers from tracker (“location service”)

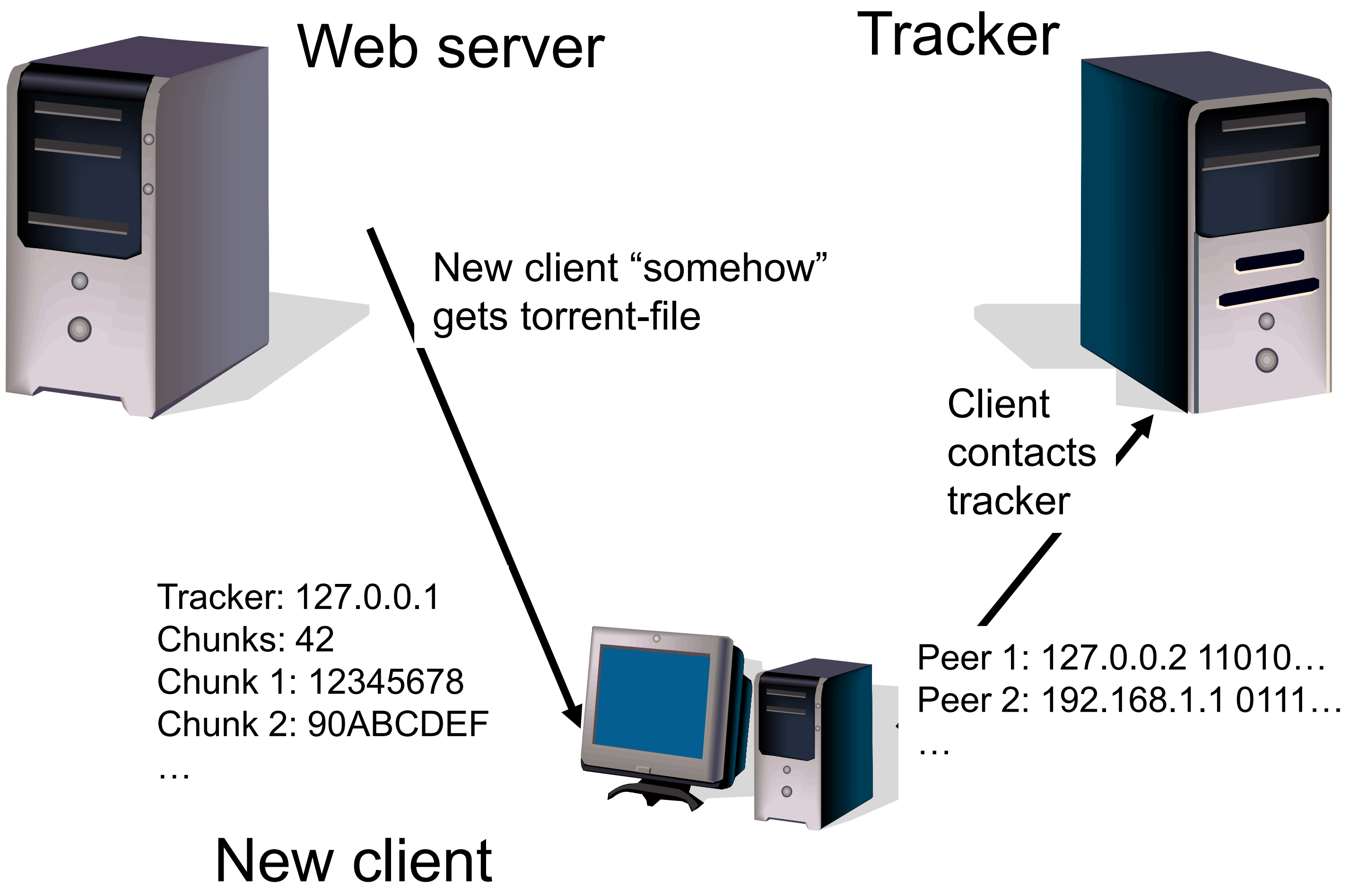
BitTorrent: Players

- 3 entities needed to start distribution of a file



BitTorrent: Starting Up

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



Tracker: 127.0.0.1
Chunks: 42
Chunk 1: 12345678
Chunk 2: 90ABCDEF
…

Peer 1: 127.0.0.2 11010…
Peer 2: 192.168.1.1 0111…
…

BitTorrent: How does it Work (Detailed)?

- 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 and their chunk availability
- 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
  - Chunk size typically 256 KB
  - Chunks make it possible to download large file in parallel

BitTorrent: Tit-for-Tat and Chunk Selection

- BitTorrent uses ***tit-for-tat*** policy
- A peer serves peers that serve it
  - Encourages cooperation, discourage free-riding
  - (not "eye for an eye", but quid pro quo!)

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

BitTorrent: Choke/Unchoke

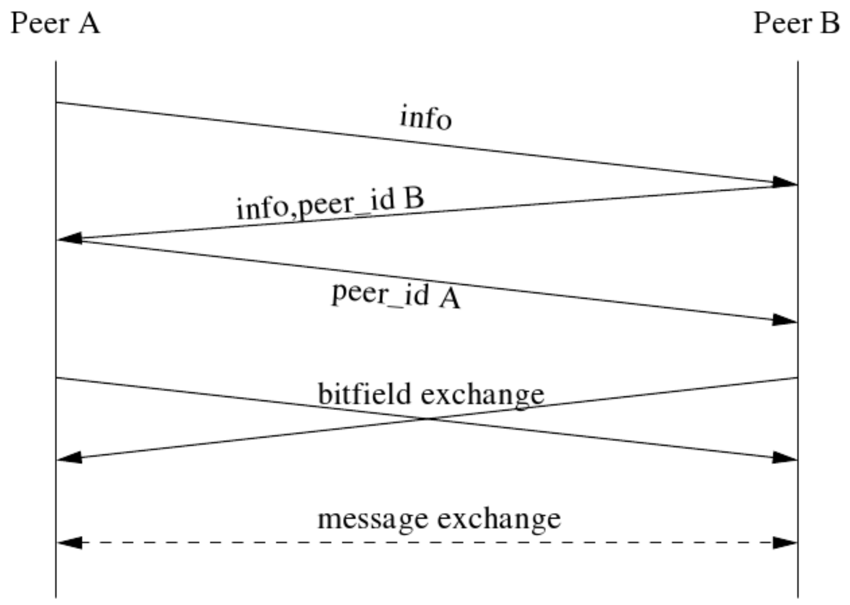
- Peer serves 4 peers in peer set simultaneously
  - Seeks best (fastest) downloaders if it’s a seed
  - Seeks best uploaders if it’s a leecher
- Choke is a temporary refusal to upload to a peer
  - Leecher serves 4 best uploaders, chokes all others
  - Every 10 seconds, it evaluates the transfer speed
  - 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

BitTorrent: Choke/Unchoke

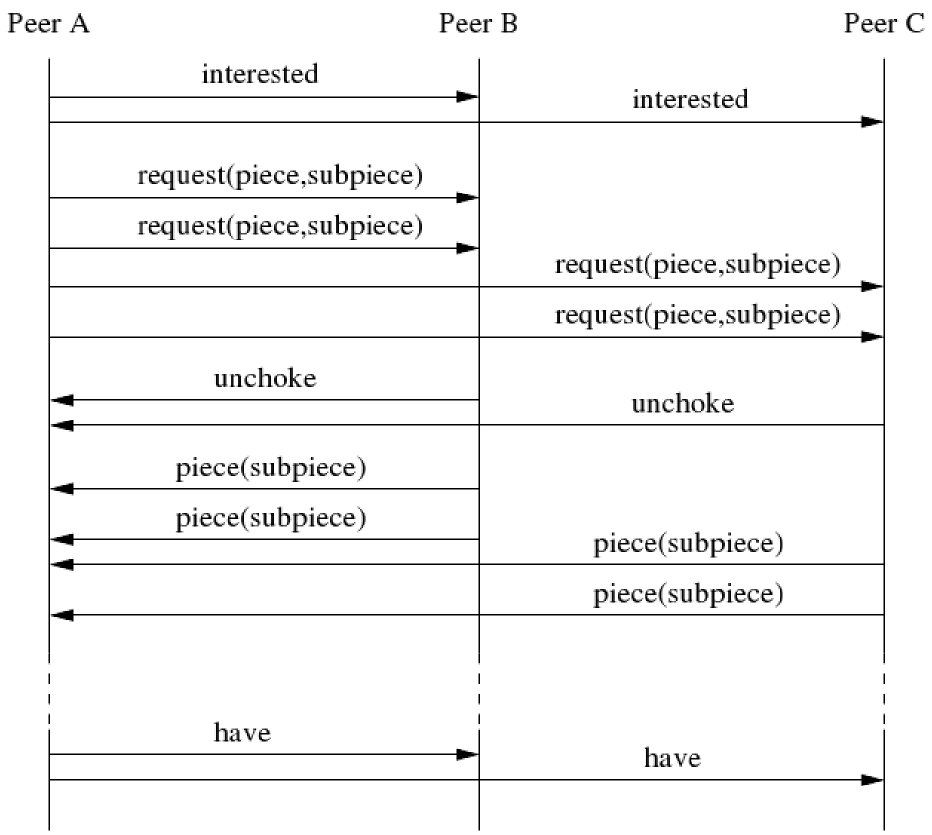
Why only 4?

What happens if one is much slower than all the others?

BitTorrent: a Quick Look at the Protocol



BitTorrent Handshake



BitTorrent Exchange

BitTorrent: How Users Behave

What does this information really tell?

What kind of content is downloaded and which external triggers have significant impact on peer (user) behavior?

BitTorrent: Strengths

- Good performance
  - Download starts ***slow***, but ***speeds up*** with increasing chunk availability
  - Adapts to the capacity of the peer
- Users keep their peers connected as seeds
  - Legal content, so no need to worry?
  - Large download, leave running over night?
- 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, Pando Media Booster, Akamai’s Netsession Interface, BitTorrent DNA

BitTorrent: Weaknesses

- File needs to be quite large
  - 256 KB chunks
  - Rarest first needs large number of chunks for good load balancing

- Everyone must contribute
  - Problem for clients behind a firewall?
  - Low-bandwidth clients have a disadvantage?

BitTorrent: Open Issues

What does „nearby“ mean?

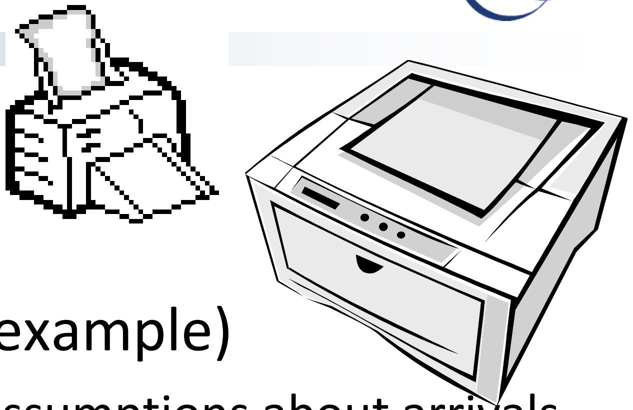
Why „endgame“ mode?

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!
- Sure? 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
  - *What do you, which printer do you use?*

  ➔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

Again: What does this graph really tell?
Could the sizes of the regions differ? Drastically, even, may be?
Is it always the slow peer who's freeriding?

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)
   - Offered/requested resources homogeneous (only dsl, only dorms)
3. Freeriding is beneficial to everyone
   - Slow (significantly slower!) peers do not offer anything
   - Large gains for everyone!

Freeriding: Recap

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

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

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

- Current research: Incentives and cooperation enforcement
- Remember: Forced contributions from everyone not necessarily 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)

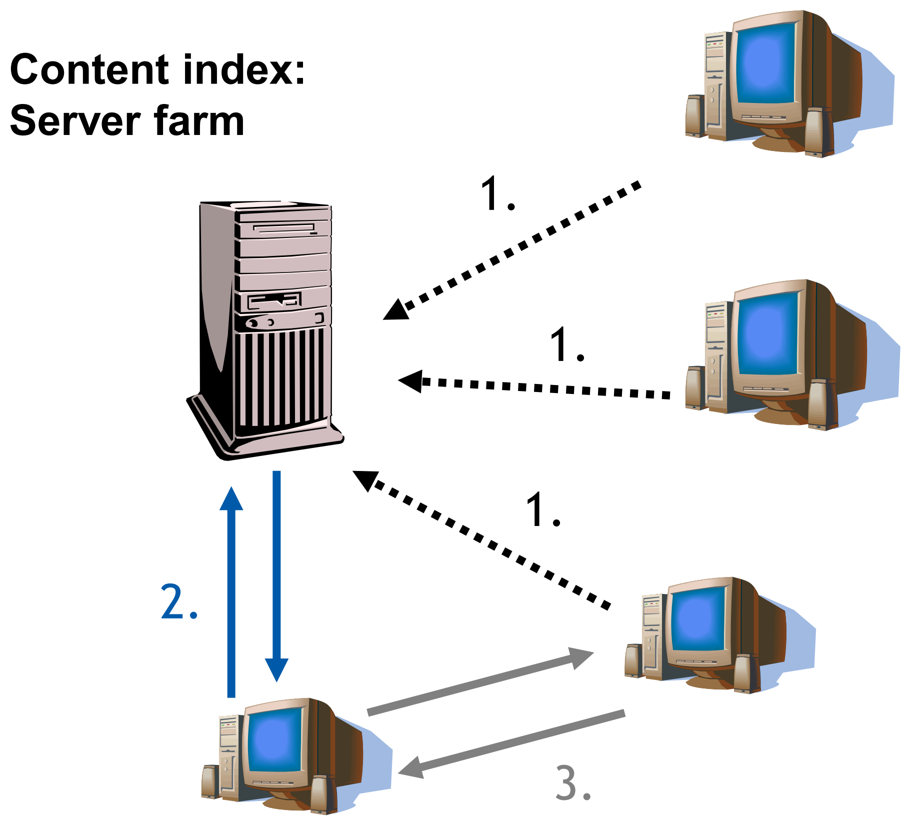
- On October 29, 2003 Napster came back as an online music store

Napster: How it Worked

- Napster was based on a central index server
  - 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 most 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
- Search with complete coverage (answer is “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 other 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
  - Server knows 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)

    (These weaknesses apply to all p2p 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

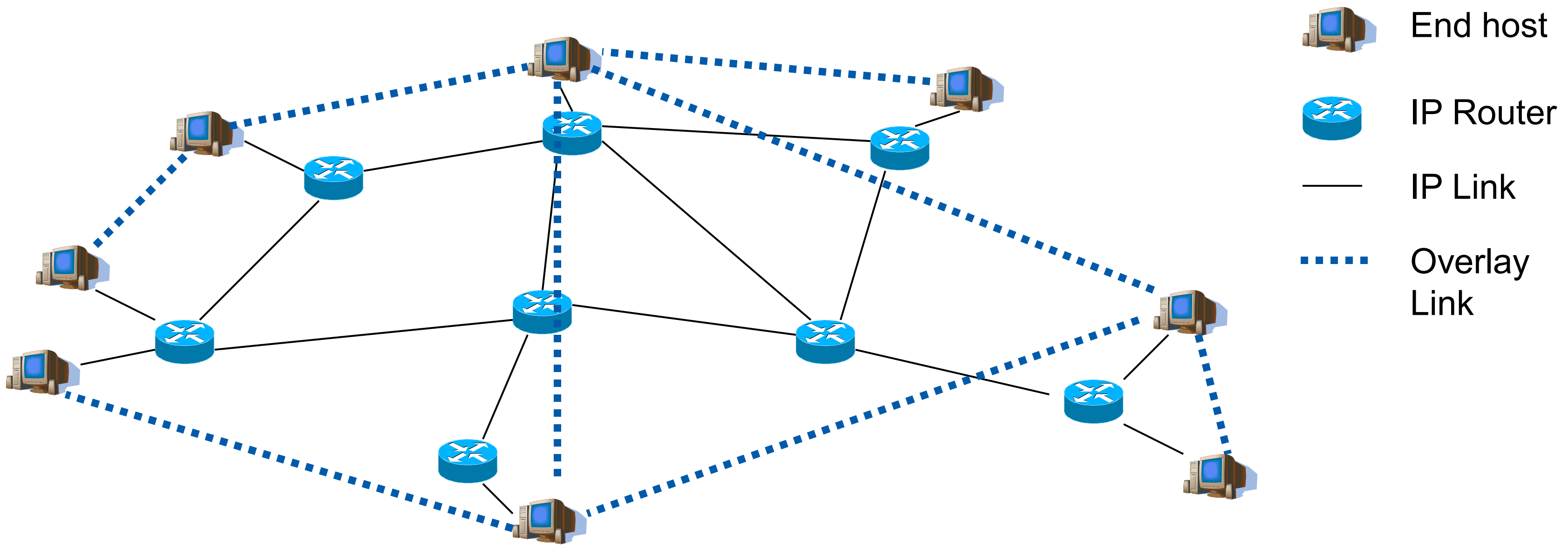
- Version 0.4 is covered here (= original Gnutella version)
- Current version 0.6 is similar to FastTrack (KaZaA)

- Gnutella has never been very big
- It provided an alternative to Napster, but was quickly surpassed by other (“better”) networks like KaZaA
- Original Gnutella 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

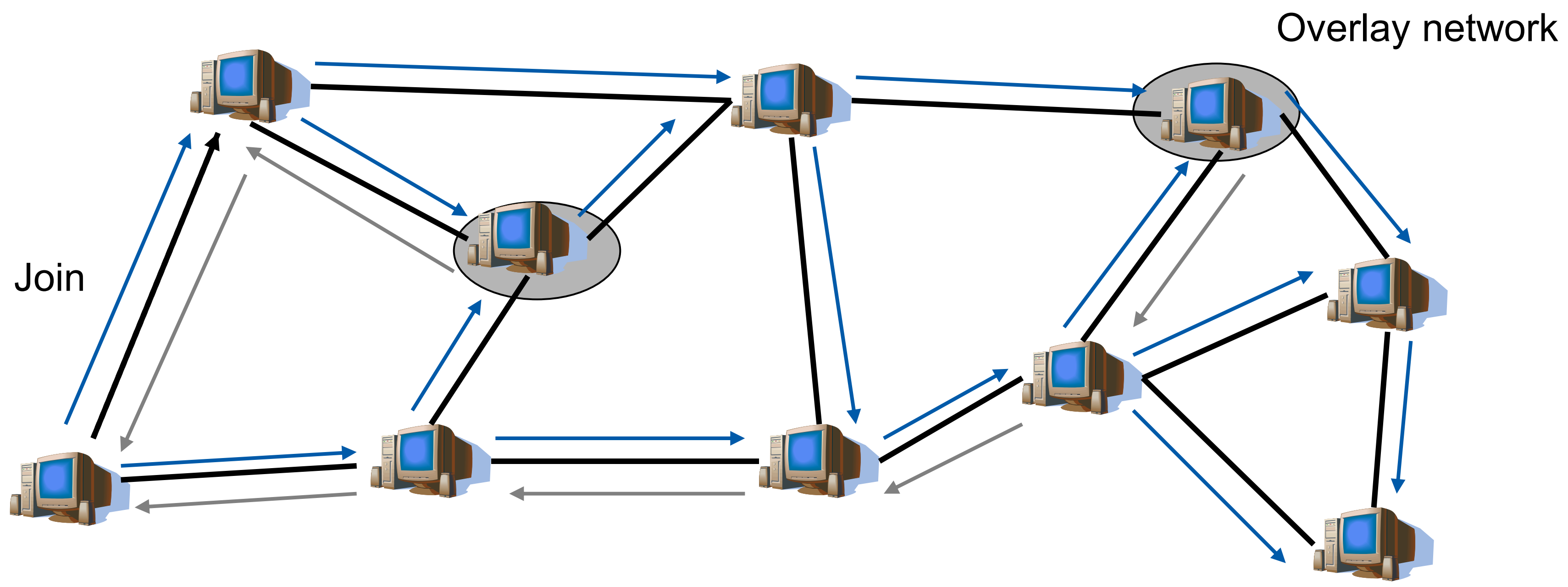


P2P and other Layer 7 systems based on overlays…

Gnutella Overview

- Gnutella network has only peers
  - All peers are fully equal
  - Peers called servents (**serv**er + cli**ent**)
- Bootstrapping: peer needs the address of an online peer
  - Out-of-band channel, e.g., get it from a website
- Peer discovery: Once a peer joins the network, it learns about other peers (and about the topology of the network..)
- Name-/Location Service: Queries are flooded into the network
- Data Service: 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” back
  - 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

- Peer discovery in Gnutella uses PING and PONG messages
- PING:
  - Used to actively discover hosts on the network. A servent receiving a Ping is expected to respond with one or more Pongs.
- 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

- Content location uses QUERY and QUERYHIT messages
- QUERY:
  - Message containing search request (name service)
- QUERYHIT:
  - The response to a Query.
  - A servent receiving a Query responds with a QueryHit if a match is found against its local data set.
  - provides recipient with information to acquire the data matching the corresponding Query (location service at the same time)
- 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

- Data service
- Download directly between peers
  - Gnutella network not involved in downloads
- Downloading using HTTP
  - Use the given port number, but HTTP syntax for request

Gnutella and 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

- *Later: "NAT whole punching"*

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:

$$y = ax^k$$

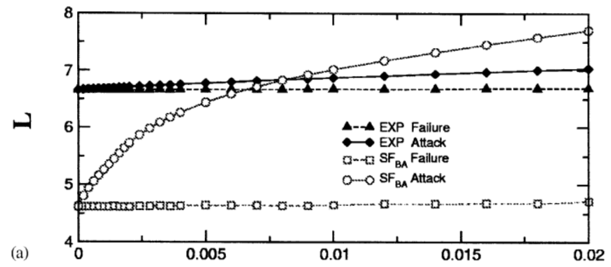
- Power laws are very common in nature
- Internet also follows some power laws
  - Popularity of Web pages (cf. Zipf's law for English words)
  - Connectivity of routers and Autonomous Systems
  - Gnutella's topology ;-)

- Has been shown:
  - Consider adding nodes to a network, preferably to well connected nodes: vertex connectivity follows a power-law
  - In Gnutella's case, the exponent is -2.3

Robustness in Power Law Networks

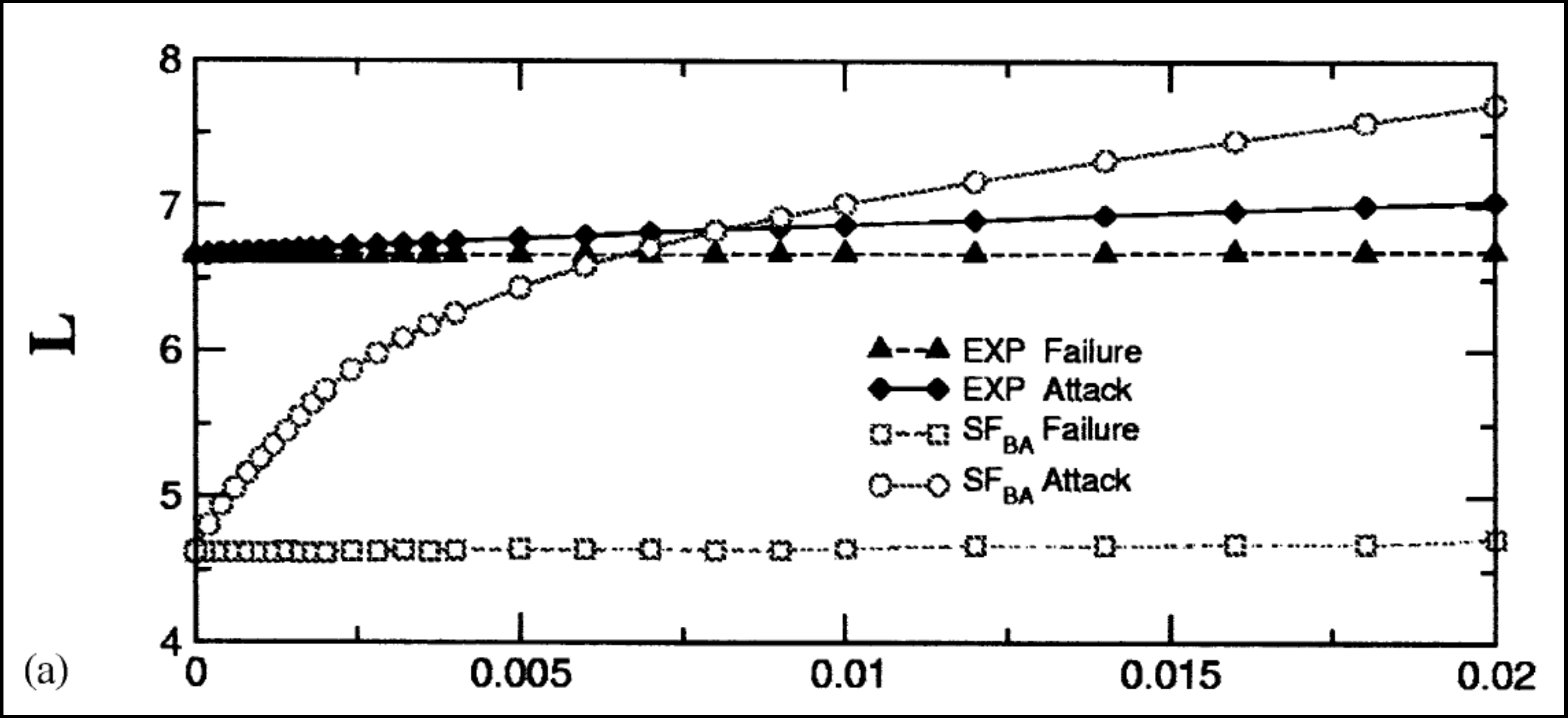
- Networks with power law exponent < 3 are very robust against random node breakdowns

- 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



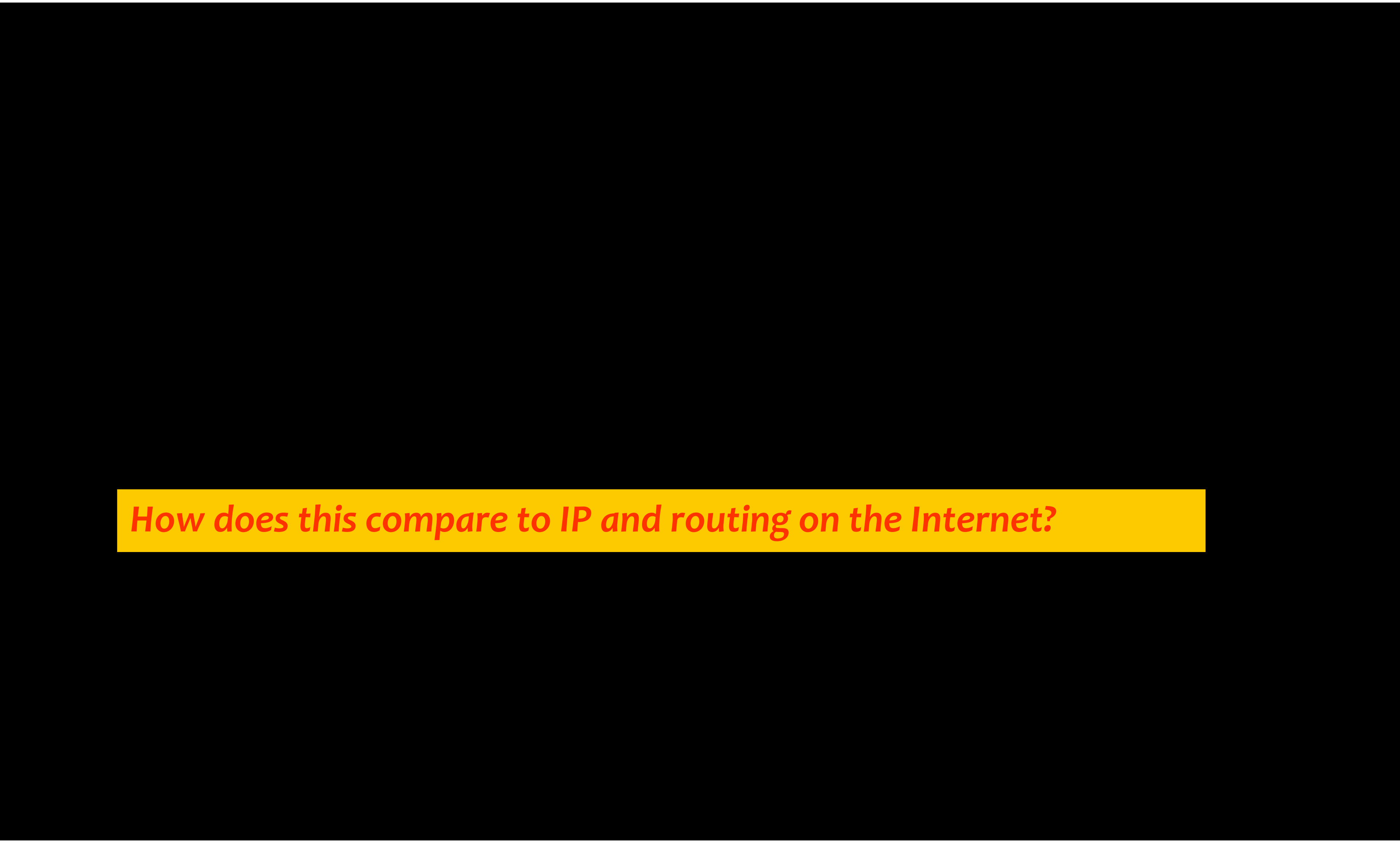
- More on power law and other kinds of networks in next chapter

Robustness in Power Law Networks



- More on power law and other kinds of networks in next chapter

Gnutella Weaknesses



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

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
  - 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

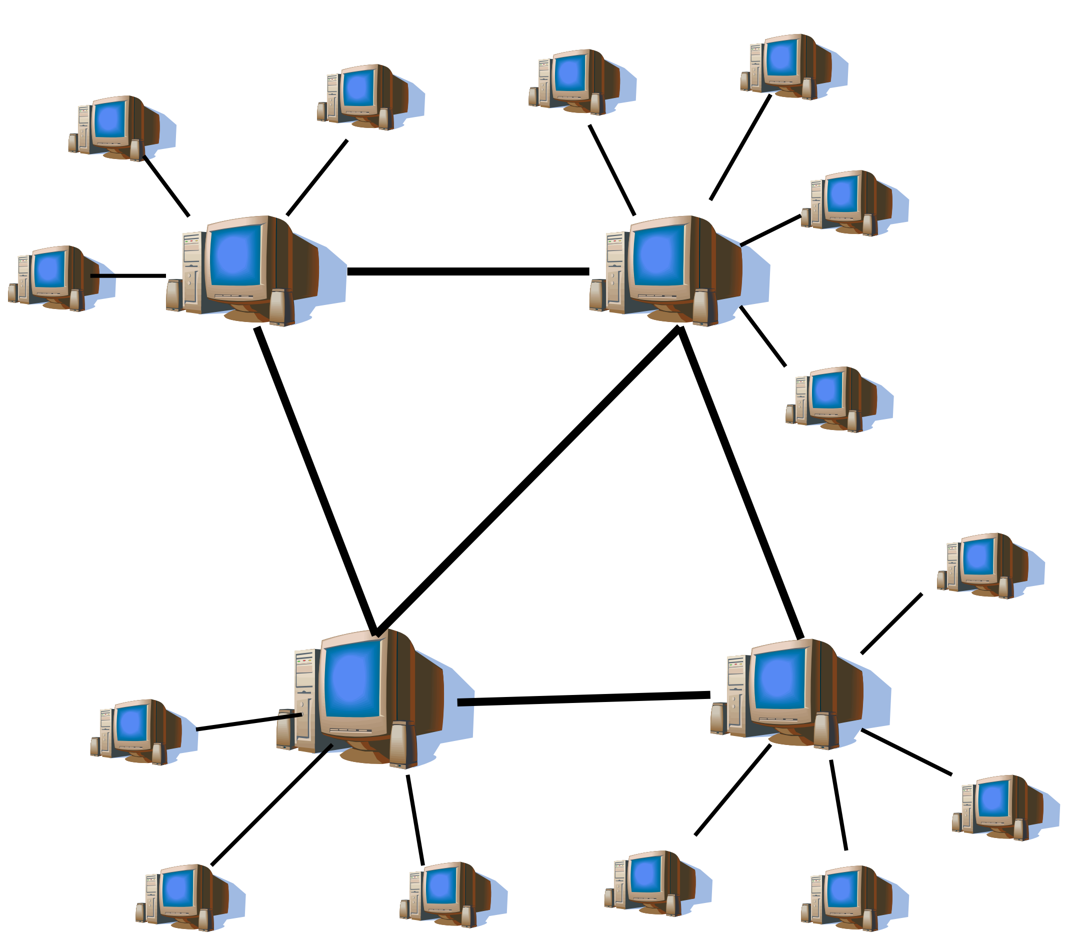
KaZaA: How it Works

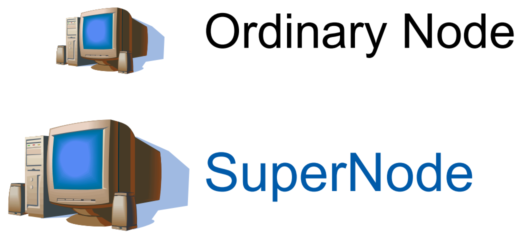
- Two kinds of nodes in KaZaA:
  - Ordinary nodes (ON)
  - Supernodes (SN)

- 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)

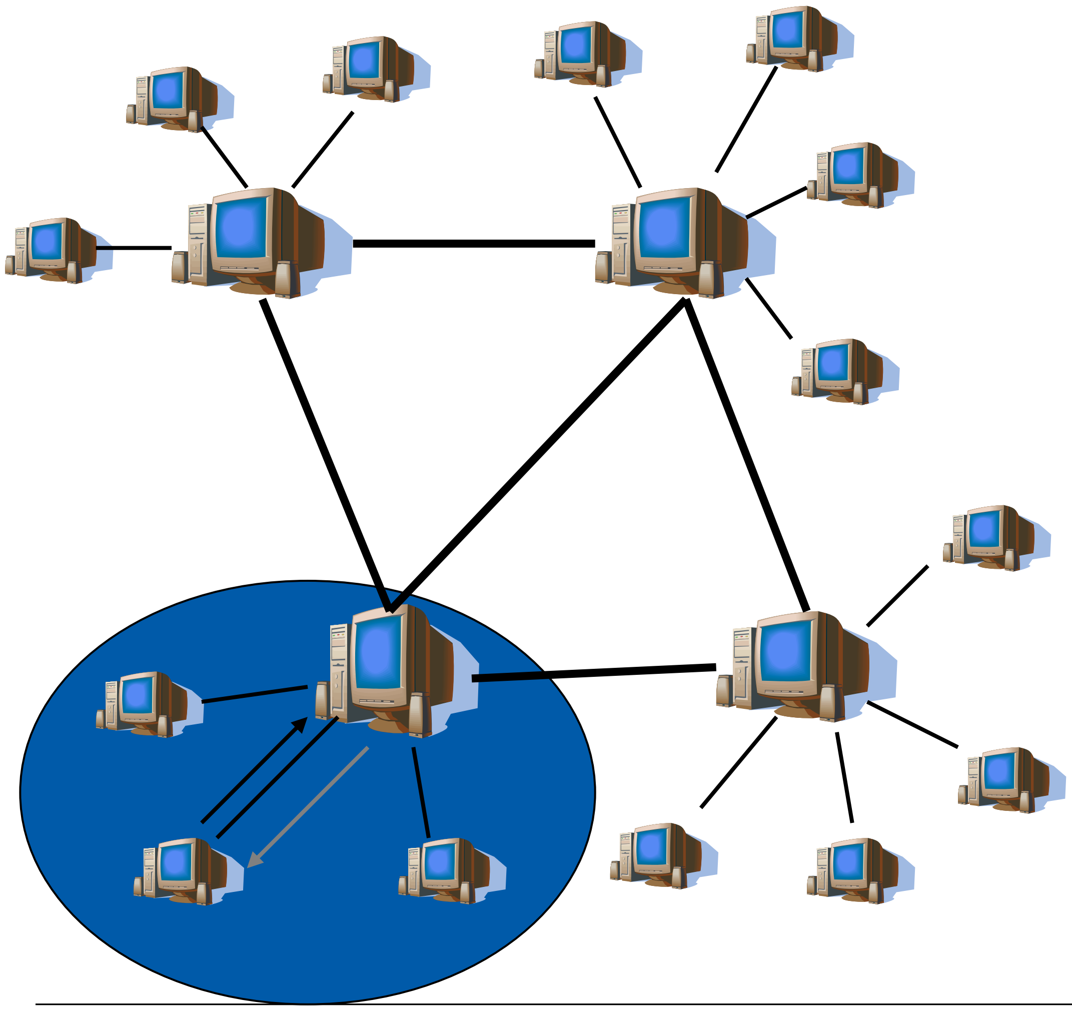
KaZaA: Hierarchy





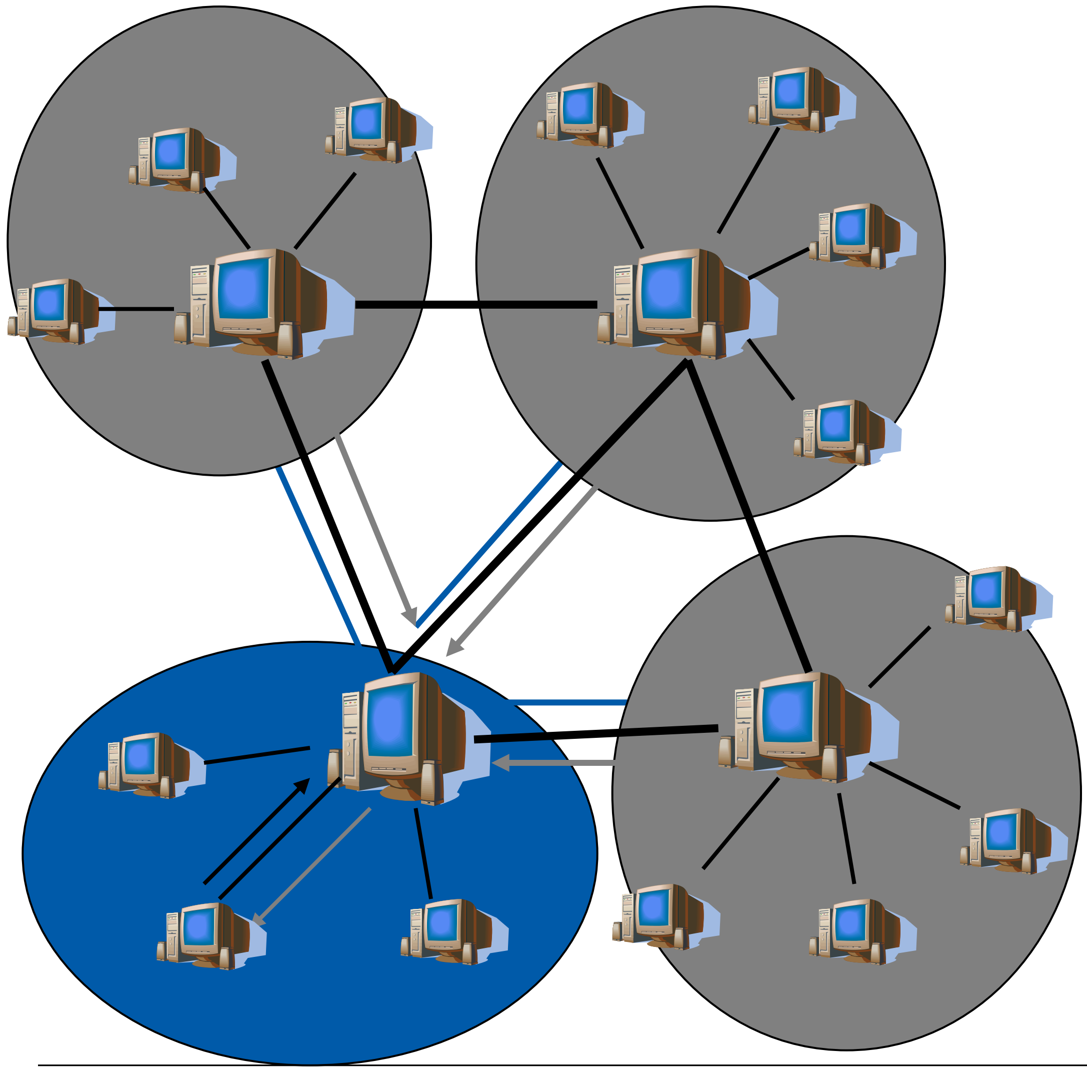
- Ordinary nodes belong to one Supernode
  - Can change SN, but not common (Kazaa-Lite)
- Supernodes exchange information between themselves
- Supernodes do not form a complete mesh

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 and Identifiers

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

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

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
  - Removal software reportedly failed often…
  - Spyware-free versions available for download (sometimes for sale)
  - “Spyware-free KaZaA” (malware) for download…

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)
  - Lots of content, lots of users
  - Some networks more popular in some areas (eDonkey DE, eMule EU)

KaZaA: Weaknesses

- Search not comprehensive
  - Can still miss a file even though it exists
  - Better coverage than Gnutella
- Easy to create collisions for UUHash (feed-in corrupted files)
- 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

Hierarchical Systems, Annex

- Many other systems followed FastTrack
  - Gnutella 0.6 extension of Gnutella to supernodes

  - „Gnutella2“ (publicity stunt) „walks“ „hubs“ (iterative queries at SN)

  - eDonkey (eMule, aMule, iMule, jMule, lMule, etc.)
    - hierarchical: users can run server or client
    - Closed source (caused by shame, because it was coded so badly…)

    - FOSS (aMule/eMule) included DHT later on
    - Most common ***file-sharing*** system, today

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	N?
Popularity	+++	-	+++

For Your Long Term Memory

- Problems that file sharing systems solve well
  - Connectivity (find neighbors, stay connected)
  - Name- and location services
  - Request delegation/routing
- Name space (index) implemented
  - Centralized index (full namespace, complete index)
  - Distributed index (full namespace, partial index)
  - “Hybrid” index (full namespace, partial but aggregated index)
- Delegation/“Routing” depending on the impl. of namespace

Putting the Systems into Perspective

- Entirely different approaches exist
  - Information dissemination: Gossiping / Epidemic routing
  - Stochastic: Random Walks / Percolation Search

- Questions remain
  - Hit rate: search not comprehensive (routing not deterministic unless Napster)
  - Neighbor selection?
  - Load balancing?
    - Requests / storage of namespace?
    - Forwarded messages?
    - Stored resources (“shared” files)?
  - Effectiveness (load on the underlying network)?
  - Fairness?
  - Resilience to malicious behavior?

Next Generation P2P...?

- P2P networks seen so far are called unstructured
  - Content can be placed anywhere in the network
  - Centralized/fully distributed sometimes called 1st generation P2P
  - Hybrid, Gossiping, Random Walk st. called 2nd generation P2P

- Contrast: Structured networks
  - Every file has a well-defined place (distribution of the namespace)
  - Sometimes called 3rd generation P2P (will there be a 4th?)
  - See DHTs in Chapter 3

File Sharing: Current State

- Most bigger file sharing networks sued into submission
  - Napster, Kazaa, eDonkey, (the pirate bay)…
- Many networks still up and running
  - Because many open clients are available
  - New target are the users (hadopi/three strikes, ask Piratenpartei…)

- 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
  - "If it was for a fair price…" – but what is a fair price?
  - (*Plus:* we are hunters and gatherers)

- Many different kinds of file sharing networks
- Old ones go, new ones come (pace slowing down?)
- 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 $\rightarrow$ name service
  - Keyword matches either file name or metadata
  - Must use same keywords (or pattern matched) 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”
- Currently mainly PirateBay+BitTorrent or eMule

P2P Communication

- P2P communication is a communication architecture based on the P2P principle
- Examples: Email, network news, instant messaging, telephony

- Current email and news systems are P2P to some degree
  - See below for details

- Generally possible to implement any communication using P2P
  - Remove central management
  - Remove any dedicated servers

P2P Communication Example: IM

- Typical instant messaging system is P2P
  - Centralized server has buddy lists
  - User logs on to server, sees buddies on-line
- Chatting directly between peers
  - Including audio, video, and file transfers
- Role of centralized server: (similar to Napster)
  - Discovery of other individuals
  - Gatekeeper / authentication
  - Helps with firewalled clients
- Jabber, P2P IM
  - jabber/xmpp connects distributed servers (encrypted, if wanted)
  - Servers interconnect, making use of DNS for location/routing!

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

- ***By the way: which problems do we experience here? :-)***

- 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 (Gossiping)

From user's point of view, P2P as implementation detail 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
  - But: Skype is perfectly legal (the affected industry is “only” telcos, they sell DSL…)
- Architecture of Skype very similar to that of KaZaA, until early 2012
  - Supernodes and ordinary nodes
- Claimed ~550 million registered, ~35 million concurrent users online
- Clients for Windows, Mac, Linux, Smartphones…

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:
  - ***Central server*** for login and billing
  - Everything encrypted, key-pair allocated by centralized server
  - ***Supernodes*** behave much the same way as in KaZaA
    - Normal nodes connect to SN, SN are not firewalled, etc..
  - ***Directory*** (who is online) spread over peers
    - Details unknown, Skype claims that system knows all users who were online in the last 72 hours
    - Keeping state of 35m nodes (160 bit ID?) → 668MB of IDs

  - Skype goes through firewalls (it does, try! :)
    - As long as firewall allows (some) outgoing connections

Skype: Supernodes and Calls

Why is skype closed-source? Keys assigned, not much known?

P2P Computation

- P2P computation: share computation between many peers
- Computationally intensive problem to solve
  - For example, crack encryption or find messages from aliens
  - Problem: needs to be easy to parallelize and distributed to peers
- Typically, centralized server manages work distribution and aggregation
  - 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 does P2P Computation Work?

- 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
  - (Usually hidden behind other, “philantropic” tasks)
- *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 were the first
- SETI@Home project run from UC Berkeley
  - Now a project in 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 is P2P (distributed) computation 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 around about 39 TB of data
- Data divided into work units at UC Berkeley
  - Work units sent to clients (picked up by clients, actually)
  - 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
    - Would people cheat for the Kudos? ;-)
- Communications over HTTP
  - Consider clients behind a firewall

SETI@Home: Some Old Numbers

- Most importantly: No alien signals detected yet ☹ (or ☺?)
- Client available for 47 OS/hardware combinations
- Millions of users, many with multiple machines
  - 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…
  - Grain of salt: 330k “stable” users for seti@home, 200k for folding@home
- Total throughput: 10.3 petaFLOPS (2009, 8.8: folding@home)
  - “RoadRunner” aimed: 1.7petaFlops (1.4), 133Million US$ (2009)
- And still: new signals added faster than they are processed

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 types of system (content distribution, communication,computation)
  - Several different file sharing networks, each with good and bad points
  - Several communication networks
  - Many computation projects
- No single solution, approach or system ruling over others