P2P: Peer-to-Peer or Pir8-to-Pir8 ?

Paolo Costa
costa@cs.vu.nl
http://www.cs.vu.nl/~costa
Vrije Universiteit Amsterdam,

Intro

- Quickly grown in popularity
  - Hundreds of file sharing applications
  - about 31% of all Internet traffic is due to BitTorrent
  - Audio/Video transfer now dominates traffic on the Internet
- But what is P2P?
  - Searching or location? -- DNS, Google!
  - Computers "Peering"? -- Server Clusters, IRC Networks, Internet Routing!
  - Clients with no servers? -- Doom, Quake!
- First, let's say what is not…

Why P2P is different

- Fundamental difference:

> *"Take advantage of resources at the edges of the network"* (Clay Shirky, O'Reilly)

- What's changed:
  - End-host resources have increased dramatically
  - Broadband connectivity now common

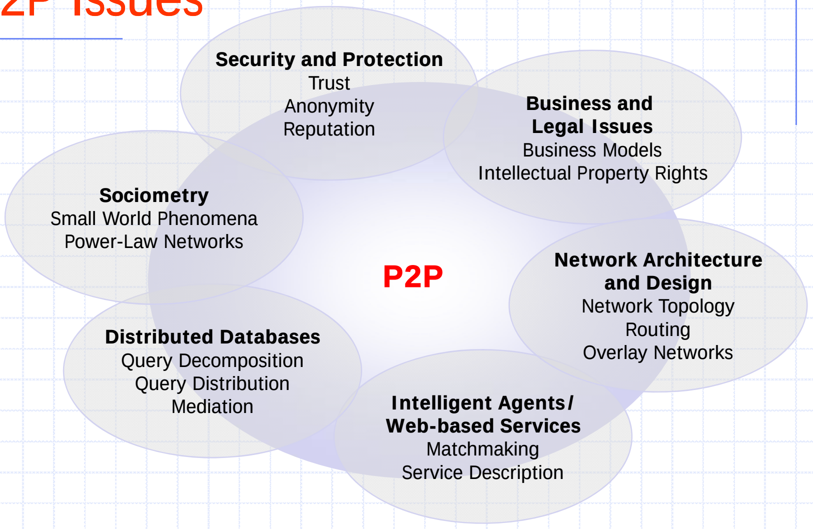
Client Server Model

- Most commonly used paradigm in today's Internet
- A client requests data or a service
- A server satisfies the request
- Successful: Web, FTP, Web Services
- However:
  - Hard to scale
  - Presents single point of failure
  - Requires administration
  - Leaves some resources unused

The alternative: peer-to-peer

- Peer-to-peer can be seen as the alternative proposed to overcome the limitations of the client-server model.
- It is a paradigm.
  - It specifies how, not what.
- P2P promotes the sharing of resources and services through direct exchange between peers.
- Resources can be:
  - processing cycles (SETI@home)
  - collaborative work (ICQ, Skype, Waste)
  - storage space (Freenet)
  - network bandwidth (ad hoc networking, internet)
  - data (most of the rest)

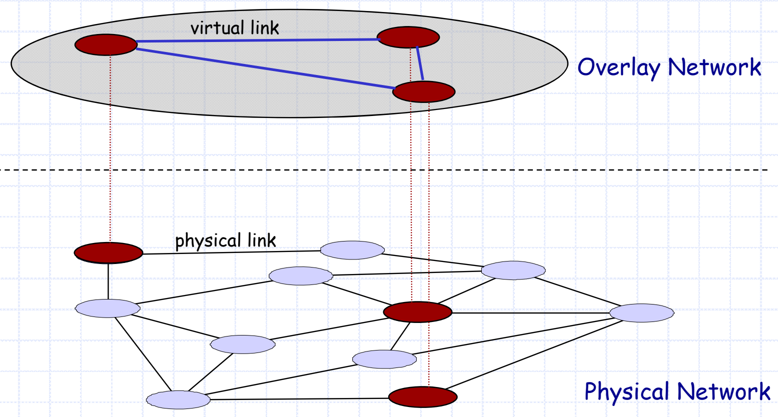
P2P Issues



What we will concentrate on

- Retrieving resources is a fundamental issue in peer-to-peer systems due to their inherent geographical distribution
- The problem is to direct queries towards nodes that can answer them in the most efficient way

Overlay Network



Characteristics of peer-to-peer

- All nodes are potential users of a service and potential providers of a service
  - Nodes act as servers, clients as well as routers (ad-hoc communication)
- Each node is independent of the other: no central administration is needed
- Nodes are dynamic: they come and go unpredictably
- Capabilities of nodes are highly variable
- The scale of the system is internet-wide
  - No global view of the system
  - Resources are geographically distributed

Data Retrieval: Lookup and Search

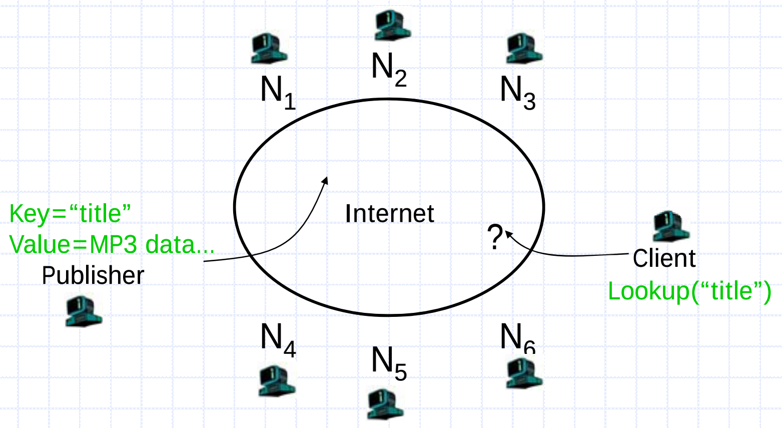
- We can distinguish two forms of retrieval operations that can be performed on a data repository
  - Search for something

    “Locate all documents on ‘Networking’ ”

    “Locate all papers which contain the expression ‘Peer-to-Peer’ ”
  - Lookup a specific item

    “Locate a copy of “RFC 3268”

How to Retrieve Data



How to Retrieve Data (2)

- Common Primitives:
  - **Join**: how do I begin participating?
  - **Publish**: how do I advertise my file?
  - **Search**: how do I find a file?
  - **Fetch**: how do I retrieve a file?

What to Retrieve

- The actual data
  - It can become a burden if query results are routed through the overlay network.
  - It is only meaningful in lookup operations…
    - Search operations return multiple items….
- A reference to the location from where the data can be retrieved
  - It is used both in lookup and in search

Overview

- **Centralized Database**
  - Napster, EDonkey, DirectConnect, Emule
- **Query Flooding**
  - Gnutella
- **Intelligent Query Flooding**
  - KaZaA, Skype
- **Swarming**
  - BitTorrent

Napster

- It was the first p2p system of the new generation.
- A killer application:
  - Free music over the Internet
- Key idea: share the storage *and* bandwidth of individual (home) users.
- In 2000, 50M users downloaded Napster client. Napster had a peak of 7TB traffic a day with 1.5 million simultaneous users.

http://www.napster.com

Napster History

- 1987: MP3 format developed by Karlheinz Brandenburg of Fraunhofer Gesellschaft. “CD ripping” now feasible.
- 1999: Shawn Fanning develops Napster, believing he has “bypassed” copyright law. Napster has >25M users in its first year.
- Dec., 1999: RIAA sues Napster for “contributory and vicarious” copyright infringement.
- April, 2000: Metallica sues Napster, Yale, Indiana Univ., and USC. (Yale bans the use of Napster within a week.)

Napster History (2)

- July, 2000: US District Judge Patel grants RIAA’s request for an injunction. The injunction is temporarily stayed soon thereafter.
- October, 2000: Napster announces a partnership with Bertlesmann AG (one of the “major labels” in the industry whose trade association is suing it!).
- January, 2001: Napster and Bertlesmann say that they will roll out a “subscription service” by “early summer” and will use “DRM technology.”
- February, 2001: Ninth Circuit upholds lower court’s findings that Napster is guilty of contributory and vicarious infringement.

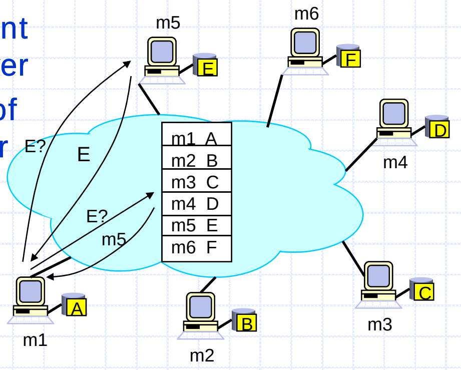
Napster History (3)

- Summer, 2001: Napster and Bertelsmann fail to roll out subscription service.
- September, 2001: Napster reaches a settlement with music publishers (but not with RIAA record labels). However, CNET.com reports the number of users has “dropped from tens of millions…to almost zero.”

Napster, R.I.P. !

Napster

- Join: on startup, client contacts central server
- Publish: reports list of files to central server
- Search: query the server => return someone that stores the requested file
- Fetch: get the file directly from peer



A pure P2P system ?

- Many researchers argued that Napster is not a peer-to-peer system (or at least not a pure one) since it depends on server availability.
- IMHO, they confuse topology with function.
- Napster is a P2P system since allows small computers on edges to contribute
  - All peers are active participants as service provider not only as consumer

Napster: Pro and Cons

☑ PROs:
- ☑ Simple
- ☑ Search scope is O(1)

☒ CONs:
- ☒ Server maintains O(N) State
- ☒ Server does all processing
- ☒ Single point of failure
- ☒ Napster "in control" (freedom is an illusion)

Gnutella

- No central authority
  - Need to find a connection point in the network
- Gnutella employs the most basic search algorithm
  - Flooding
- Each query is forwarded to all neighbors
- Propagation is limited by a HopToLive field in the messages

Gnutella ?!?

- Mixture of GNU (<u>G</u>NU's <u>N</u>ot <u>U</u>nix) and Nutella

Gnutella History

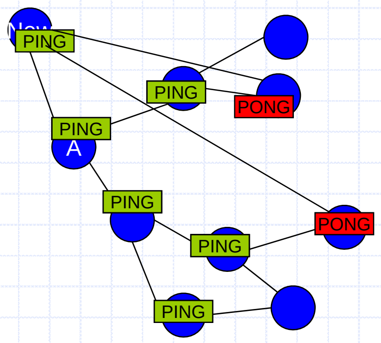
- Gnutella was written by Justin Frankel, the 21-year-old founder of Nullsoft, in just 14 days
- Nullsoft (maker of WinAmp) posted Gnutella on the Web, March 1999.
- Nullsoft acquired by AOL, June 1999.
- A day later AOL yanked Gnutella, at the bequest of Time Warner.
- People had already downloaded and shared the program.
- Gnutella continues today, run by independent programmers.

Gnutella: Overview

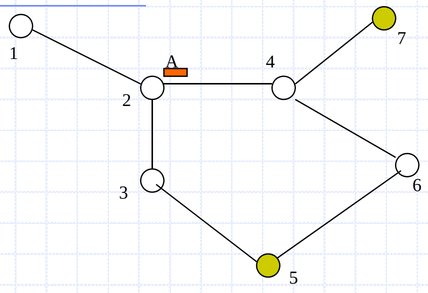
- Query Flooding:
  - **Join**: on startup, client contacts a few other nodes; these become its “neighbors”
  - **Publish**: no need
  - **Search**: ask neighbors, who as their neighbors, and so on... when/if found, reply to sender.
  - **Fetch**: get the file directly from peer

Gnutella: joining the network

- The new node connects to a well known ‘Anchor’ node.
- Then sends a PING message to discover other nodes.
- PONG messages are sent in reply from hosts offering new connections with the new node.
- Direct connections are then made to the newly discovered nodes.

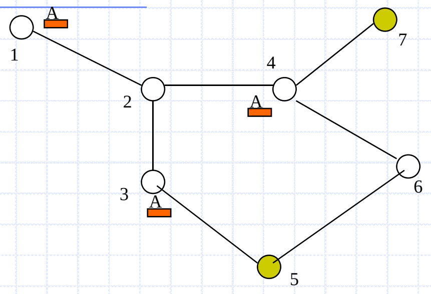


Gnutella search mechanism



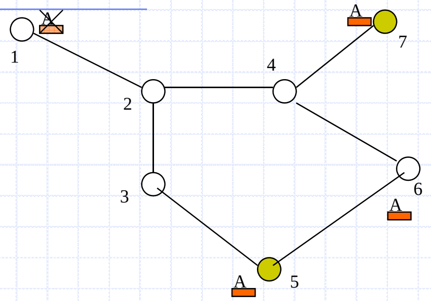
- Node 2 initiates search for file A

Gnutella search mechanism



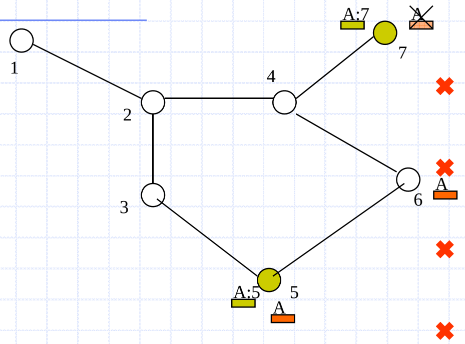
- Node 2 initiates search for file A
- Sends message to all neighbors

Gnutella search mechanism



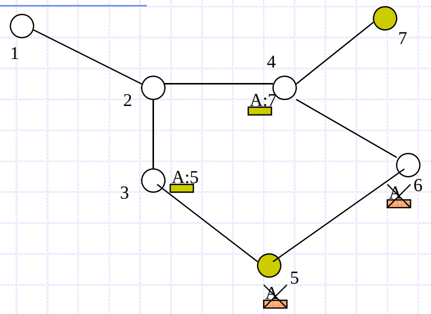
- Node 2 initiates search for file A
- Sends message to all neighbors
- Neighbors forward message

Gnutella search mechanism



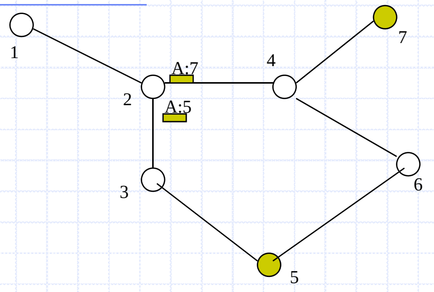
- Node 2 initiates search for file A
- Sends message to all neighbors
- Neighbors forward message
- Nodes that have file A initiate a reply message

Gnutella search mechanism



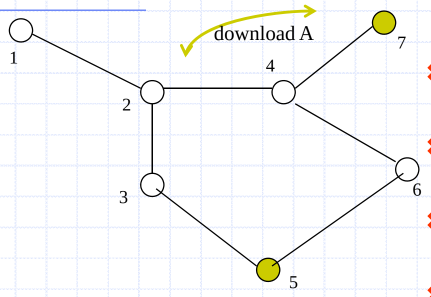
- Node 2 initiates search for file A
- Sends message to all neighbors
- Neighbors forward message
- Nodes that have file A initiate a reply message
- Query reply message is back-propagated

Gnutella search mechanism



- Node 2 initiates search for file A
- Sends message to all neighbors
- Neighbors forward message
- Nodes that have file A initiate a reply message
- Query reply message is back-propagated

Gnutella search mechanism

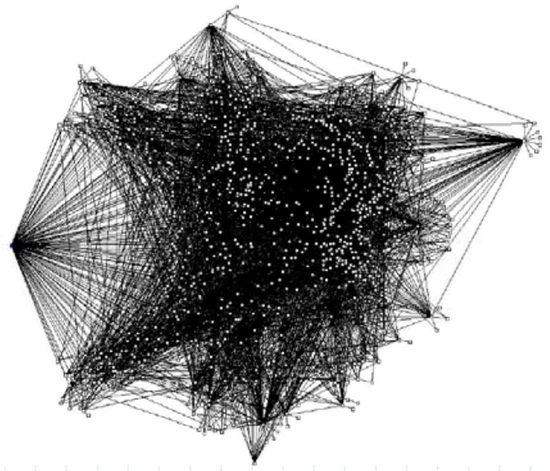


- Node 2 initiates search for file A
- Sends message to all neighbors
- Neighbors forward message
- Nodes that have file A initiate a reply message
- Query reply message is back-propagated
- File download

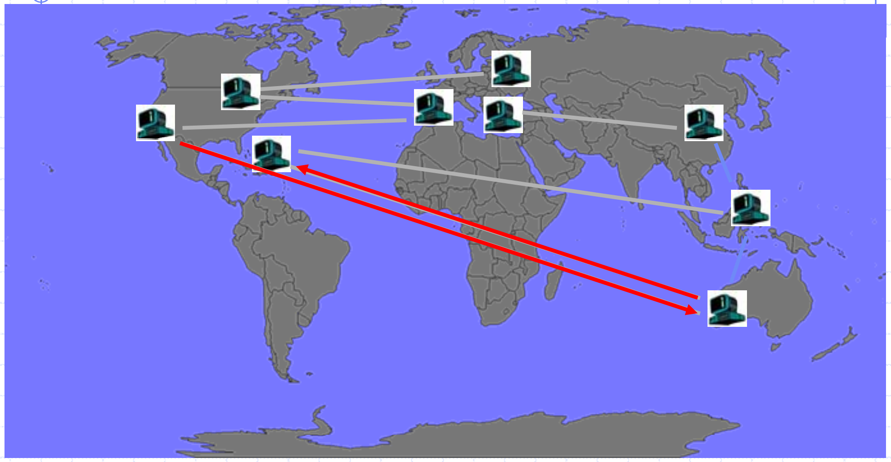
Gnutella: Pro and Cons

- ☑ PROs:
  - ☑ Fully de-centralized (no central coordination required)
  - ☑ Search cost distributed
  - ☑ “Search for S” can be done in many ways, e.g., structured database search, simple text matching, “fuzzy” text matching, etc.
- ☒ CONs:
  - ☒ Flood” of Requests. If average number of neighbors is C and average TTL is D, each search can cause CD request messages.
  - ☒ Search scope is O(N)
  - ☒ Search time is O(???)
  - ☒ Nodes leave often, network unstable

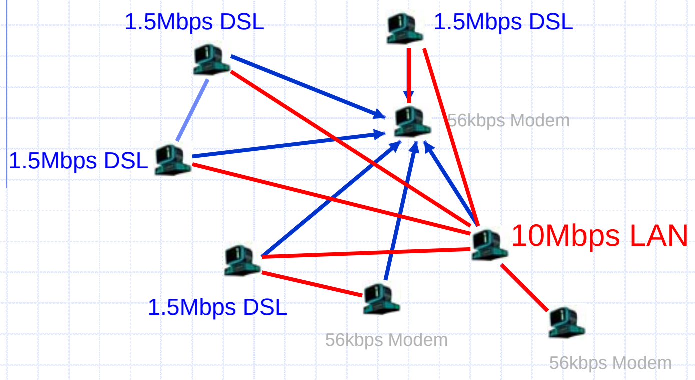
Gnutella Network (27/1/2000)



Aside: Search Time?



Aside: All Peers Equal?



Kazaa

- Peers connect directly to each other; content is distributed and there is no central server.
- Search requests are sent to the "nearest" **supernode**, which tries to locate the content; if it fails, the request is sent to other supernodes.
- Any node running Kazaa with a good Internet connection can become a **supernode**.
  - Other Kazaa users upload lists of shared files to neighboring supernodes.
  - Supernodes facilitate search but do not host content; peers connect directly to download files.
  - Supernode status is controlled by the software based on user settings (permission to become a supernode, bandwidth restrictions, *etc.*).

http://www.kazaa.com

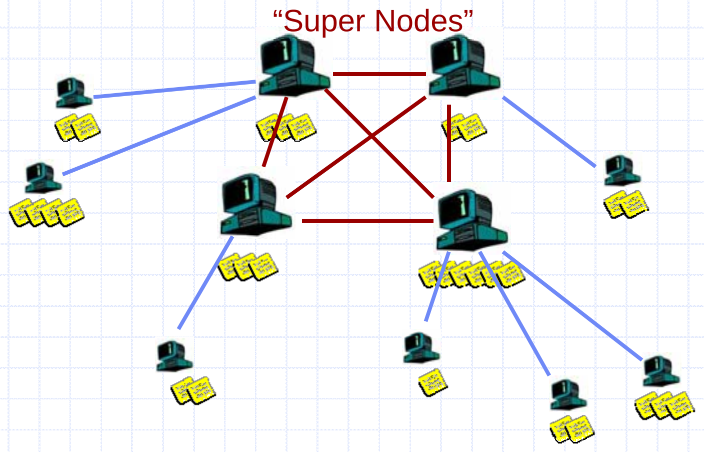
KaZaA: History

- In 2001, KaZaA created by Dutch company Kazaa BV
- Kazaa Media Desktop now is produced by Sharman Networks, Ltd., "a consortium of private investors with multimedia interests" (see company website). Based in Australia with offices in Europe.
- Single network called FastTrack used by other clients as well: Morpheus, giFT, etc.
- Eventually protocol changed so other clients could no longer talk to it
- Most popular file sharing network today with >10 million users (number varies)

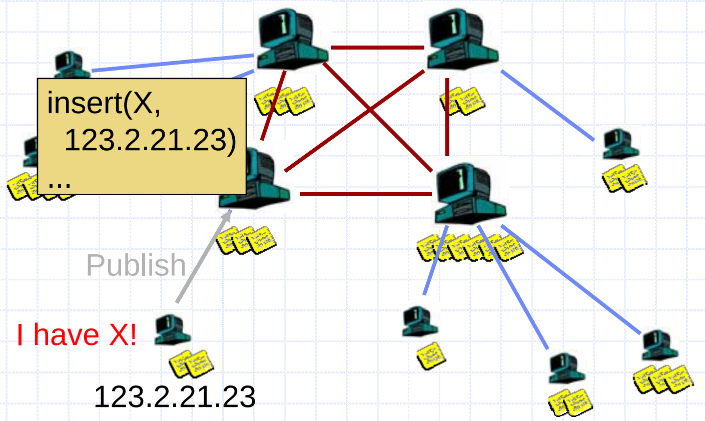
KaZaA: Overview

- "Smart" Query Flooding:
  - **Join**: on startup, client contacts a "supernode" … may at some point become one itself
  - **Publish**: send list of files to supernode
  - **Search**: send query to supernode, supernodes flood query amongst themselves.
  - **Fetch**: get the file directly from peer(s); can fetch simultaneously from multiple peers

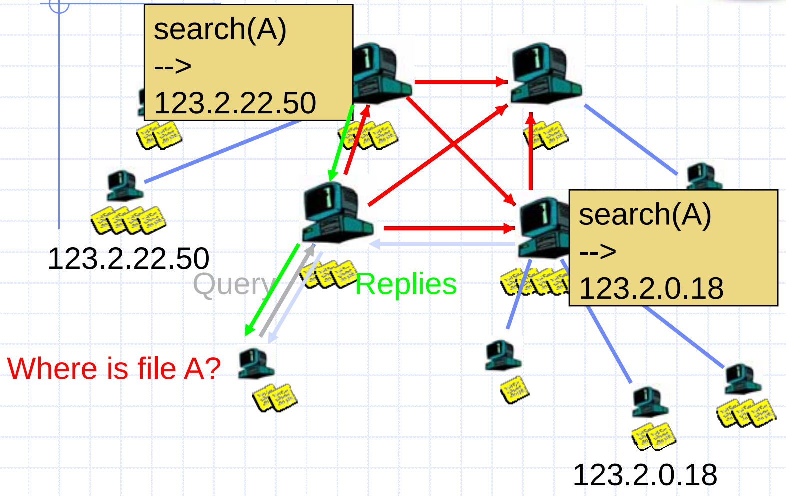
KaZaA: Network Design



KaZaA: File Insert



KaZaA: File Search



KaZaA: Fetching

- More than one node may have requested file…
- How to tell?
  - Must be able to distinguish identical files
  - Not necessarily same filename
  - Same filename not necessarily same file…
- Use Hash of file
  - KaZaA uses UUHash: fast, but not secure
  - Alternatives: MD5, SHA-1
- How to fetch?
  - Get bytes [0..1000] from A, [1001…2000] from B
  - Alternative: Erasure Codes

KaZaA: Discussion

☑ Pros:
- ☑ Tries to take into account node heterogeneity:
  - ☑ Bandwidth
  - ☑ Host Computational Resources
  - ☑ Host Availability (?)
- ☑ Rumored to take into account network locality

☒ Cons:
- ☒ Mechanisms easy to circumvent
- ☒ Still no real guarantees on search scope or search time
- ☒ Proprietary Protocol

Gnutella Reloaded

- Originally, Gnutella peers connected to other peers randomly. This was ok with broadband users but not modem users
- Solution: provide a hierarchy to the network
  - Ultrapeer
    - Acts as a proxy
    - Reduces the number of nodes in the network involved in message handling and routing
    - Forwards a query to the leaf only if it believes the leaf can answer it
    - Connencted to other Ultrapeers and other normal Gnutella hosts
  - Leaf
    - Normal Gnutella hosts
    - Leaves a small number of connections open (usually to ultrapeers)
    - Usually connects to 3 ultrapeers at a time

Skype

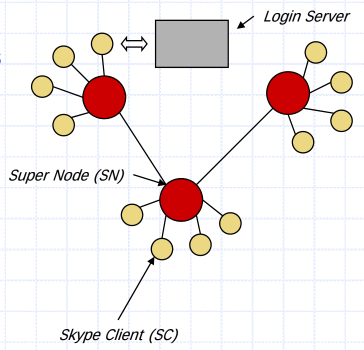
- Developed by Danes Niklas Zennstrom and Janus Friis, creators of Kazaa
  - Now owned by eBay
- First peer-to-peer VOIP application
- Started in 2003, grew in only 1.5 years from nothing to a rapidly expanding Internet telephony operator
  - 12,049,795 people on-line today
- Protocol not fully disclosed
  - reverse-engineering

Skype Architecture

- ✖ Based on peer-to-peer network
  - minimal network infrastructure
  - Utilize its users' computers to do the work
  - three types of hosts:
    - ordinary hosts: Skype users
    - super nodes: Skype users with sufficient computing power, memory and network bandwidth
    - login servers: for authentication
- ✖ Skype owns only the login servers

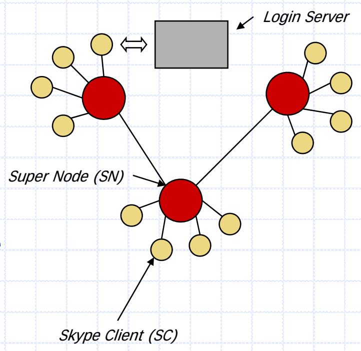
Login

- ✖ Authenticate the user name and password with the Skype login server
- ✖ After that, the client's host cache is filled with IP addresses of 7 bootstrap super nodes
  - Any client with a public IP address and sufficient performance can be a supernode
- ✖ Establish a connection with one of them
- ✖ The host cache is periodically updated with the IP address of new super nodes



Call Establishment

- ✖ User lookup
  - "Global Index" technology
  - Use flooding among supernodes to find users
- ✖ After acquiring the callee's IP address, the caller establishes a connection to the callee and send signaling messages.
- ✖ After the callee answered the call, voice packets are sent directly between the two parties.
- ✖ If one or both of the parties are behind NAT or firewall, they communicate through another Skype online node



Other Issues

- ✖ Security
  - All communication is encrypted with Advanced Encryption Standard (AES)
  - The AES encryption keys are transmitted between hosts using the Rivest, Shamir, & Adleman (RSA) algorithm.
- ✖ Voice quality
  - wideband codecs: allow 50 – 8000 Hz to pass through
- ✖ NAT and firewall
  - Skype uses Simple Traversal of User Datagram Protocol (STUN) and Traversal Using Relay NAT (TURN) algorithms to determine the type of firewalls and NAT

The Others

- WinMX (http://www.winmx.com):
  It began its life as an OpenNap client capable of connecting to several servers simultaneously. Then, it moves towards a decentralized Kazaa-like network whereby **supernodes** (ultra-peers, super-peers, etc) act as temporary indexing servers.
- EDonkey / EMule (http://www.emule-project.net):
  Edonkey adopts a Napster-like scheme but multiple servers are availble and searches may be spread across them. Emule is a GPL client that fixes some flaws (see next slide)
- Direct Connect (http://dcplusplus.sourceforge.net):
  It uses hubs which connect a group of users. The mentality on the Direct Connect network is a bit different from (most) other p2p-networks, as it has a very strong social ingredient in the form of chat, and privateness in the form of small hubs or even "reghubs" (hubs whereby user/passwd is required).

Free Riders

- Free riders only download without contributing to the network.
- First attempts in Napster were user-based
  - *"I refuse to upload you if you don't share anything worth"*
- Direct Connect++ requires you explicitly but, of course, it does not guarantee, share something valuable (unless hub administrator takes care of)
- Emule has a credits system whereby a client stores the amount of data it has uploaded and downloaded from every client and gives clients that have net upload to it a higher priority in the queue
- Kazaa in 2002 introduces the Participation Level that increases when you upload and decrease when you download. Then when you upload a file to someone else the person with the highest Participation Level gets it first, then they upload it on to the person with the next highest Participation Level, and so on.
  - Unfortunately the system adopted by KaZaA is flawed as it relies on the client accurately reporting their Participation Level and therefore it is easy to cheat with the many "unofficial" clients

Bittorent

- BitTorrent allows many people to download the same file without slowing down everyone else's download.
- It does this by having downloaders swap portions of a file with one another, instead of all downloading from a single server. This way, each new downloader not only uses up bandwidth but also contributes bandwidth back to the *swarm*.
- Such contributions are encouraged because every client trying to upload to other clients gets the fastest downloads.

http://bittorrent.org

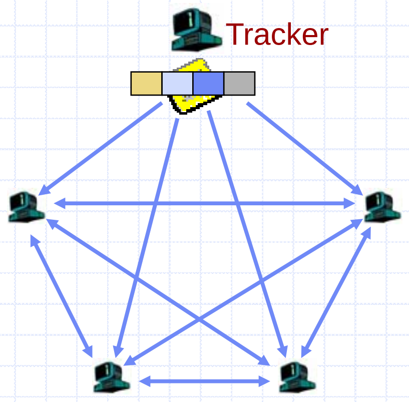
BitTorrent: History

- In 2002, B. Cohen debuted BitTorrent (written in Python)
- Key Motivation:
  - Popularity exhibits temporal locality (Flash Crowds)
  - E.g., Slashdot effect, CNN on 9/11, new movie/game release
- Focused on Efficient *Fetching*, not *Searching*:
  - Distribute the *same* file to all peers
  - Single publisher, multiple downloaders
- Has some "real" publishers:
  - Blizzard Entertainment using it to distribute the beta of their new game
  - Linux Distros

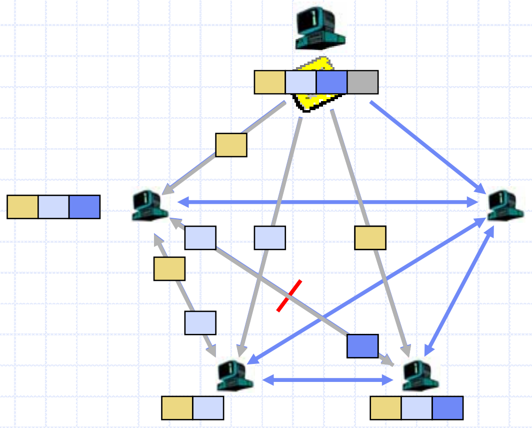
BitTorrent: Overview

- Swarming:
  - **Join**: contact centralized "tracker" server, get a list of peers.
  - **Publish**: Run a tracker server.
  - **Search**: Out-of-band. E.g., use Google to find a tracker for the file you want.
  - **Fetch**: Download chunks of the file from your peers. Upload chunks you have to them

BitTorrent: Publish/Join



BitTorrent: Fetch



BitTorrent: Terminology

- Torrent: A meta-data file describing the file(s) to be shared. A *.torrent* file holds:
  - Names of the files
  - Sizes
  - Checksum of all blocks
  - Address of the tracker
  - Address of peers
- Seed: A peer that has the complete file and still offers it for upload
- Leech: A peer that has incomplete download
- Swarm: All seeders/leeches together make a swarm
- Tracker: A server that keeps track of seeds and peers in the swarm and gathers statistics. When a new peer enters the network, it queries the tracker to provide a list of peers

BitTorrent: Content Distribution

- ✖ Breaks the file down into smaller fragments (256KB in size). The metadata holds the SHA1 hash to verify data integrity
- ✖ Peers download these missing fragments from each other and upload to those who don't have it
- ✖ The fragments are not downloaded in sequential order and needs to be assembled by the receiving machine
- ✖ Clients start uploading what they already have (small fragments) before the whole download is finished

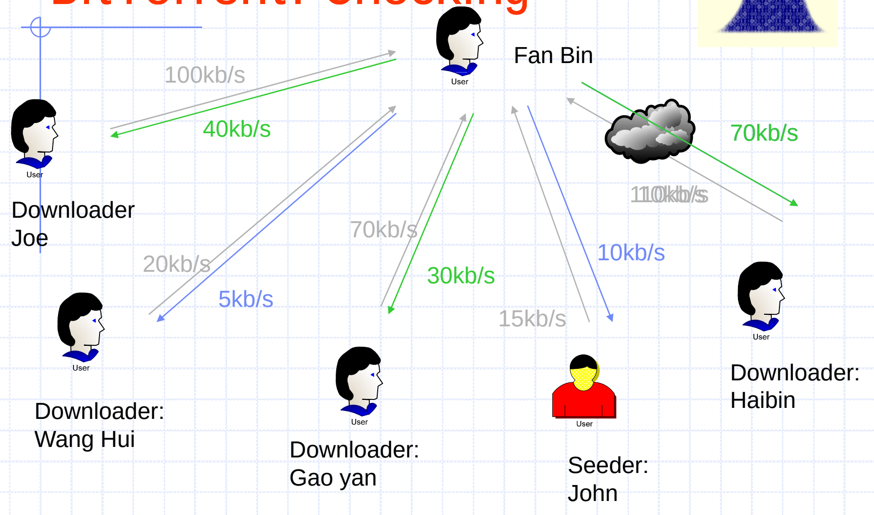
BitTorrent: Content Distribution (2)

- ✖ Once a peer finishes downloading the whole file, it should keep the upload running and become an additional seed in the network
- ✖ Everyone can eventually get the complete file as long as there is "one distributed copy" of the file in the network, even if there are no seeds

Where all the magic happens

- ✖ Choking is a temporal refusal to upload
  - Choking evaluation is performed every 10 seconds.
  - Each peer unchokes a fixed number of peers (default = 4)
  - The decision on which peers to un/choke is based solely on download rate, which is evaluated on a rolling, 20-second average
- ✖ A BitTorrent peer has a single '*optimistic unchoke*' which is uploaded regardless of the current download rate from it. This peer rotates every 30s
  - This allows to discover currently unused connections are better than the ones being used

BitTorrent: Chocking



BitTorrent: Game Theory

- Employ "Tit-for-tat" sharing strategy
  - "I'll share with you if you share with me"
  - Be optimistic: occasionally let freeloaders download
    - Otherwise no one would ever start!
    - Also allows you to discover better peers to download from when they reciprocate
  - Similar to cooperative move in Prisoner's Dilemma
- Approximates Pareto Efficiency
  - If two peers get poor download rates for the uploads they are providing, they can start uploading to each other and get better download rates than before

BitTorrent: Summary

- ☑ Pros:
  - ☑ Works reasonably well in practice
  - ☑ Gives peers incentive to share resources; avoids free-riders
- ☒ Cons:
  - ☒ Pareto Efficiency relative weak condition
  - ☒ Central tracker server needed to bootstrap swarm (is this really necessary?)

Video Streaming

- BitTorrent is becoming attractive also for live video streaming
- Traditional (centralized) approaches suffer from scalability when the number of users increases
- In BitTorrent, instead, the more users there are, the better it is
  - more users = more upload bandwidth available