Peer to Peer — P2P

- There are many non-technical questions around P2P because of the ease of use in distributing content.
- The technical aims of P2P can be summarised as follows:

  Load balancing Get rid of central servers, less load on one node in the network.

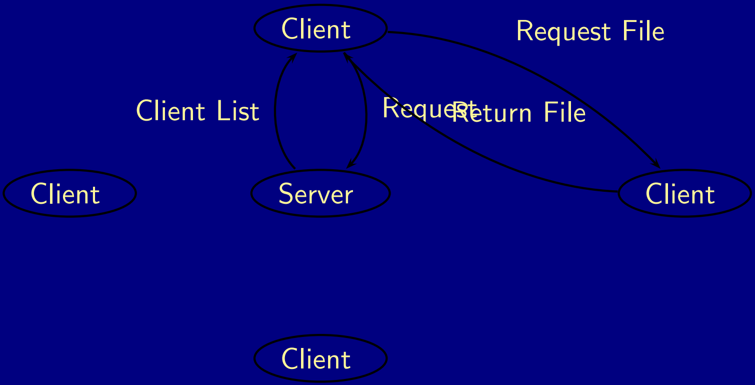
  Fault Tolerance No single point of failure, if the server goes down the network can still carry on.

  Efficient use of resources There are often lots of wasted resources on network (spare file space, spare computation power ....).
- Also coupled with this is the fact in most P2P systems it is very easy for clients to participate (`SETI@Home`,`folding@home`), posting a torrent is easy.

First Generation P2P – Napster

- Centralised server
- Each node registers list of files that is has to the central server
- When a node wishes to retrieve a file it request from the central server a list of client nodes that have that file
- Then the client picks a node from which to download the file.

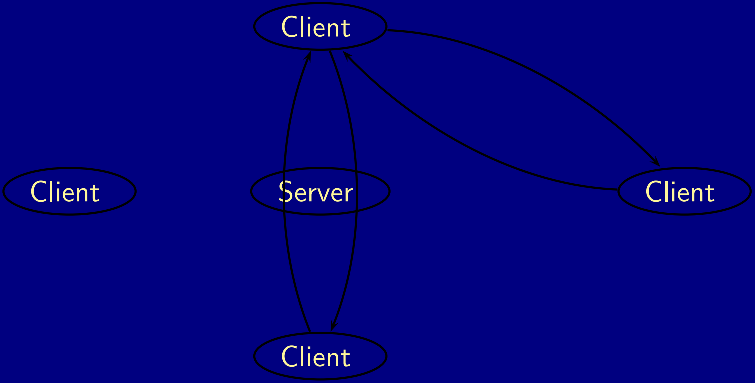
First Generation P2P – Napster



First Generation P2P – Napster

- Problems with Napster like protocols
  - Single point of failure – The server.
  - Client only downloads from one other client at a time.
- Solutions
  - Have more than one server.
  - Make the clients more complicated and download from multiple clients (essentially what Bit-torrent does)

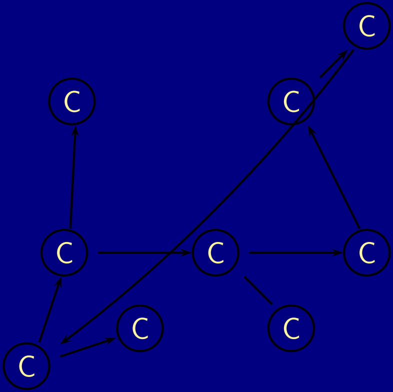
First Generation P2P



First Generation P2P – Gnutella

- Again files are distributed across the network
- But no central server
- A node must know the IP address of at least one other Gnutella node. Clients initialised with a set of working nodes
- Each node request each node in its working set
- If a node receives a request either:
  - The file is there
  - Otherwise the request is propagated on
- Requests have a lifetime TTL (Time to live).

First Generation P2P – Gnutella



First Generation P2P – Scalability

- Napster did not last long enough to test scalability issues, but
- Think of Google with a central server, scalability is less of a problem today.
- Gnutella essentially the protocol tries to find a node by flooding the network.
- Gnutella can have the problem that the network has more request messages floating around than anything else.
- Instead of flooding do a random walk from node to node, works but it can take a can take a long time to find the file.

The second generation of P2P systems

Issue a P2P file sharing systems must address:

File placement	Where to publish the file to be shared by others?
File Look up	Given a named item, how do you find it or download it?
Scalability	How does the performance degrade with the network size?
Self-Organization	How does the network handle nodes joining and leaving the network?

The second generation of P2P systems

Additionally users often find attractive:

Censorship resistance How does the network function if nodes are shut down in an attempt to censor items?

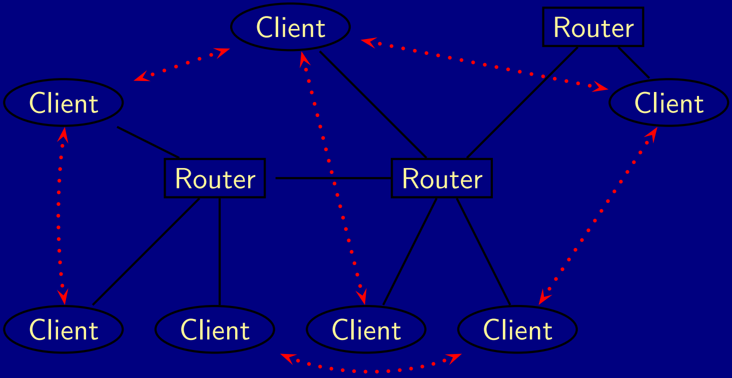
Fault-tolerance How can performance be kept in the presence of node failures.

Free-rider elimination Discourage nodes that only download and never upload.

Overlay networks

- Gnutella type protocols flood the network with lots of request.
- What is needed is some map that of nodes in the network that have files.
- The basic idea is that of an overlay network, a network over a network.

Overlay networks



Overlay Networks

- The overlay network has a different notion of neighbour to the underlying network.
- In the overlay network we need some way of storing routing tables and a routing algorithm.

Pastry

- Pastry is a system developed at Microsoft research as a middle ware layer for P2P systems.
- Each node is given a (almost certainly) unique 128 bit idea via a cryptographic hash function on the node's IP address or public key.
- Cryptographic hash functions have random looking outputs, things with similar IP address will get very different 128 bit ids, and there is very little chance that two different IP addresses get given the same hash.
- In a N-node system a message can be routed in

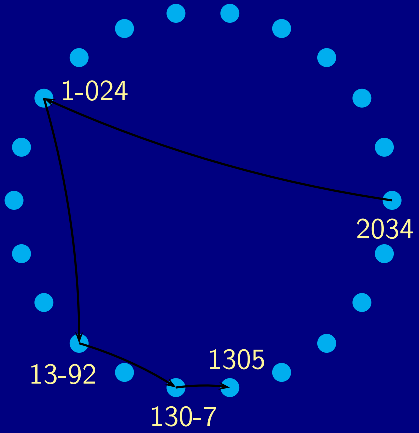
  $$O(log_r N)$$

  hops, where $r = 2^b$ and b is typically 4.

Pastry Routing

- Pastry routes messages using prefix routing.
- If node P wants to send a message to an address x it sends it to the next node in its neighbourhood set that have the closet common prefix.

Pastry – Routing



A route from 2034 to 1305

Pastry Routing Implementation

- Each node maintains the following data:
  - Leaf Set L. Each node n maintains a list of nodes that are between $(n + L/2)$ and $(n - L/2)$
  - Routing Table R Each row j of the table points to a node whose id shares the first j digits with n with with digit $j + 1$ being different.
  - Neighbourhood set M The set of nodes that are nearest to n w.r.t. the network underlying network topology.
- If a destination node is not in the leaf set then the node is forwarded to a node whose id shares a larger common prefix with the destination id.

Chord

- Chord uses an alternative addressing scheme based on a m-dimensional hypercube.
- Each node only connects via its outgoing edges.
- With $N = 2^m$ nodes a message can be sent in $m = \log_2 N$ hops.
- Chord distributes files over the hypercube.
  - A file with hash K is put in the node with address K otherwise the node with the closets higher key.
  - Each node has a routing table (called a finger table) with $m = log_2 N$ entries, each entry directs it to the a neighbour.
  - Look up is in a greedy fashion, go via the neighbour which will get you there quickest.

Incentive Mechanisms

- In the previous arrangements we have assumed that all nodes are willing to share the resources that they have got.
- One of the primary (technical) aims of file sharing is to distribute load.
- Instead everybody downloading a popular file from a central server, we let clients host all or some of the file and other clients download not just from the central server.

Incentive Mechanisms

There are three main mechanisms to manage incentives:

Reputation
: Each node gets a better service if it has built up a reputation of offering a better service.
- For example peers in the KaZaA network build up their reputation scores by uploading files to others.
- Without security possible to fake your reputation.

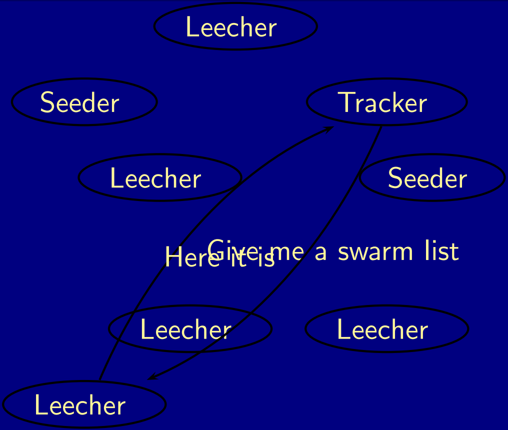
Barter
: If a node provides a good service to you then provide a good service back. Essentially the model in Bit Torrent. It is argued that this gives some economically optimal situation.

Currency
: Peers earn money by giving resources to the system and spend money by downloading and using resources.

Bit torrent

- Bit torrent uses a central server (for each file) called a *tracker* which keeps track of all peers that have the file. Note that generally the tracker does not actually have the file to be downloaded.
- A file is divided up into a number of *chunks*
- Each peer can have some or all of the chunks
- A *seeding peer* has all the chunks.
- A *download peer* has some of the chunks.
- The idea is that even while a peer is downloading it can still be serving chunks.
- Each chunk has a hash to verify if it has been downloaded properly (stops people injecting bogus chunks).

Bit torrent

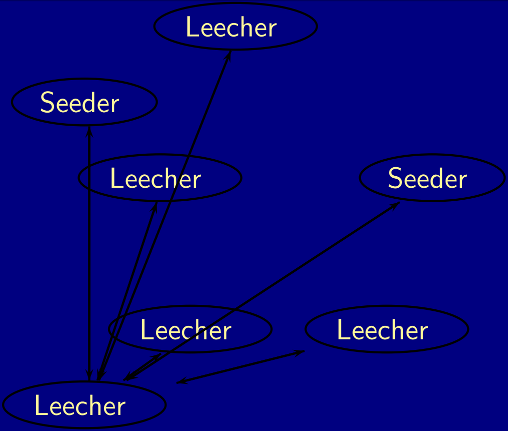


Note that the tracker need not give all the files in the swarm.

Bit torrent

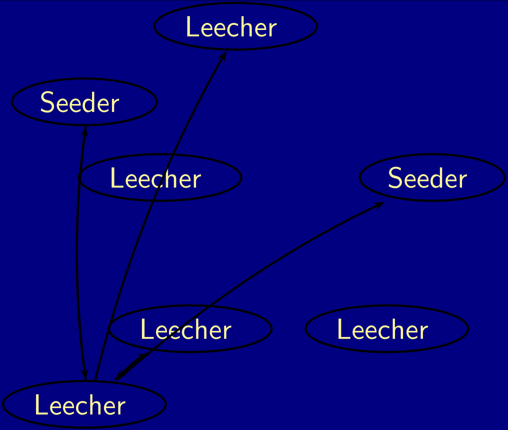
- The actual mechanism of how the client downloads from the current list of seeders and leechers (the swarm) can be quite complicated.
- Essentially the client asks each other client what pieces does it have?
- Then according to some strategy the client then asks for chunks form the other members of the swarm.
- Tit for Tat, means that you don't have to answer a request if you not getting something back from requester (bandwidth). This can make start up times a bit slow.

Bit torrent



- What chunks do you have?
- Here is a list of chunks that I have.

Bit torrent



- Request some chunk
- Here are the requested chunks.

Bit torrent

- Note, at all times the downloading client is serving requests for chunks. Helps with Tit for Tat.
- Client might periodically ask for the chunk list from members of the swarm.
- You don't have to serve a downloaded chunk. There are many reasons why you might not:
  - You don't have the bandwidth.
  - Tit for tat scheme says no.
- The idea is that the more you upload the better service you have.

Bit torrent

Bit torrent like protocols are used in quite a few places:

- Games
  - Blizzard's World of Warcraft uses bit torrent to deliver updates
  - GnuZ The Duel (online multiplayer shot and kill game)
- Bit Torrent Inc. Legal version of Bit torrent download.
- Amazon S3 uses bit torrent in parts.
- Lots of Linux distributions offer bit-torrent downloads.

Security

Two aspects:

- Verification of identities and verification of money (if been used as an incentive mechanism).
- This can be solved using standard techniques from cryptography, public/private keys.
- Secure Storage, is a bit harder.

Secure Storage

Self-Certifying Data Use cryptographic hash.

Information Dispersal Files encoded into m blocks s.t. any n is sufficient to reassemble the original data with $m < n$.

Secret Sharing Encrypt the data into l shares, so that any k nodes can decrypt but not $k - 1$.

Other topics, *secure routing*, *distributed stenographic file systems*

Anonymity

- With bit-torrent it is easy to find a list of people downloading a file. Just connect and look at the list of peers.
- Various types of anonymity are desirable:
  - hide the author or publisher of the content
  - hide the identity of a node storing the content
  - hide the identity and details of the content
  - hide details of queries for content.

Anonymity

Freenet
: peer-to-peer content distribution system that makes it infeasible to discover the true origin or destination of a file passing through its network.

Onion routing
: provides a mechanism for anonymous connection between nodes (neither node knows the identity of each other but messages still get through).

Note that these schemes can be quite sophisticated. Via the use of techniques from cryptography it can be impossible (almost) to break the anonymity. It is more complex than just throwing away server logs.

Other uses of P2P

Skype uses peer-to-peer protocol to forward phone calls around the net. Closed protocol, not sure how it works.

Joost Peer-to-peer internet television.

OcenStore `http://oceanstore.cs.berkeley.edu/` large scalable, fault tolerant storage system.

Distributed Databases takes files up to the next level.

Distributed Computation Seti@Home, look for messages from the little green men, or folding@home find out how proteins fold.