Peer-to-Peer (P2P) Systems:

Peer-to-Peer (P2P) systems are a type of network architecture where computers, or "peers," in the network act as both clients and servers. In a P2P network, each computer can share resources (such as files or processing power) and request resources from other computers directly, without the need for a centralized server. P2P systems are often associated with file sharing and decentralized communication.

Key Characteristics of P2P Systems:

1. Decentralization: In P2P systems, there is no central server that controls or manages the network. Instead, all peers have equal status, and they can communicate and share resources directly with one another.
2. Shared Resources: Peers in a P2P network can share files, data, or services with other peers. This makes P2P networks efficient for tasks like file sharing, where users can upload and download content from each other.
3. Scalability: P2P networks are often scalable because adding new peers to the network does not require significant changes to the existing infrastructure.
4. Redundancy: P2P networks can be more robust because multiple copies of resources can exist on different peers. If one peer goes offline, others can still provide the same resources.

Aspect	P2P Model	Client-Server Model
Resource Sharing	All peers share resources with each other.	Clients request resources from a central server.
Network Topology	Decentralized; no central server.	Centralized; a central server manages resources.
Scalability	Typically more scalable as new peers can be easily added.	May require more centralized infrastructure to scale.
Dependency on Server	No central server dependency.	Clients depend on the central server.
Resource Availability	Resource availability depends on the presence of peers with the desired resources.	Resource availability depends on the server's availability.
Network Traffic	Peers communicate directly, resulting in distributed traffic.	Traffic is routed through the central server, which can lead to bottlenecks.
Redundancy	Redundancy is achieved by the presence of multiple peers with the same resources.	Redundancy is achieved through backup servers.
Example Use Cases	File sharing (BitTorrent), distributed computing (SETI@home).	Web hosting, email services, online gaming.

First Generation P2P – Napster

- Centralised server

First Generation P2P – Napster

- Centralised server
- Each node registers list of files that is has to the central server

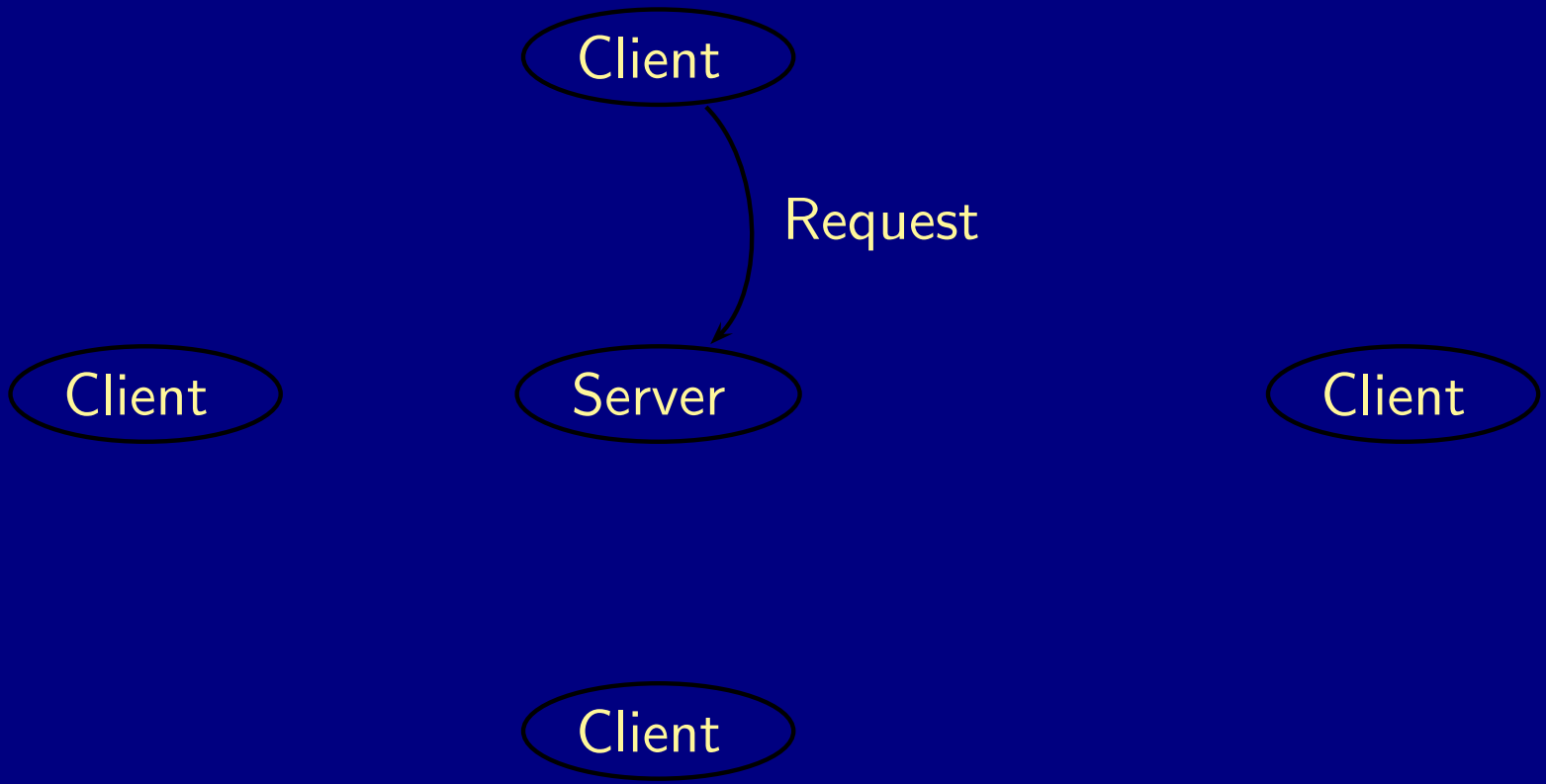
First Generation P2P – Napster

- 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

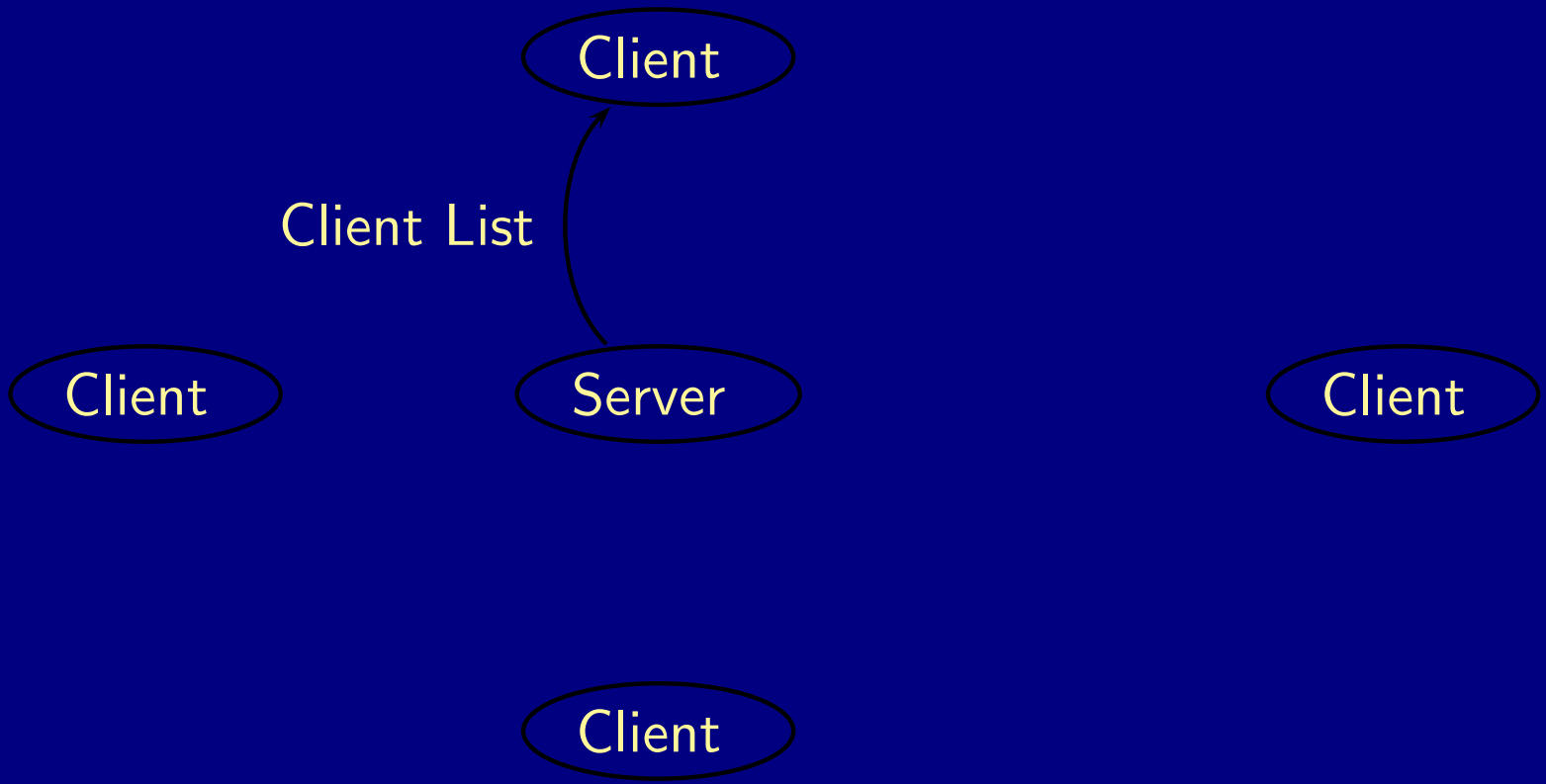
First Generation P2P – Napster

- 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.

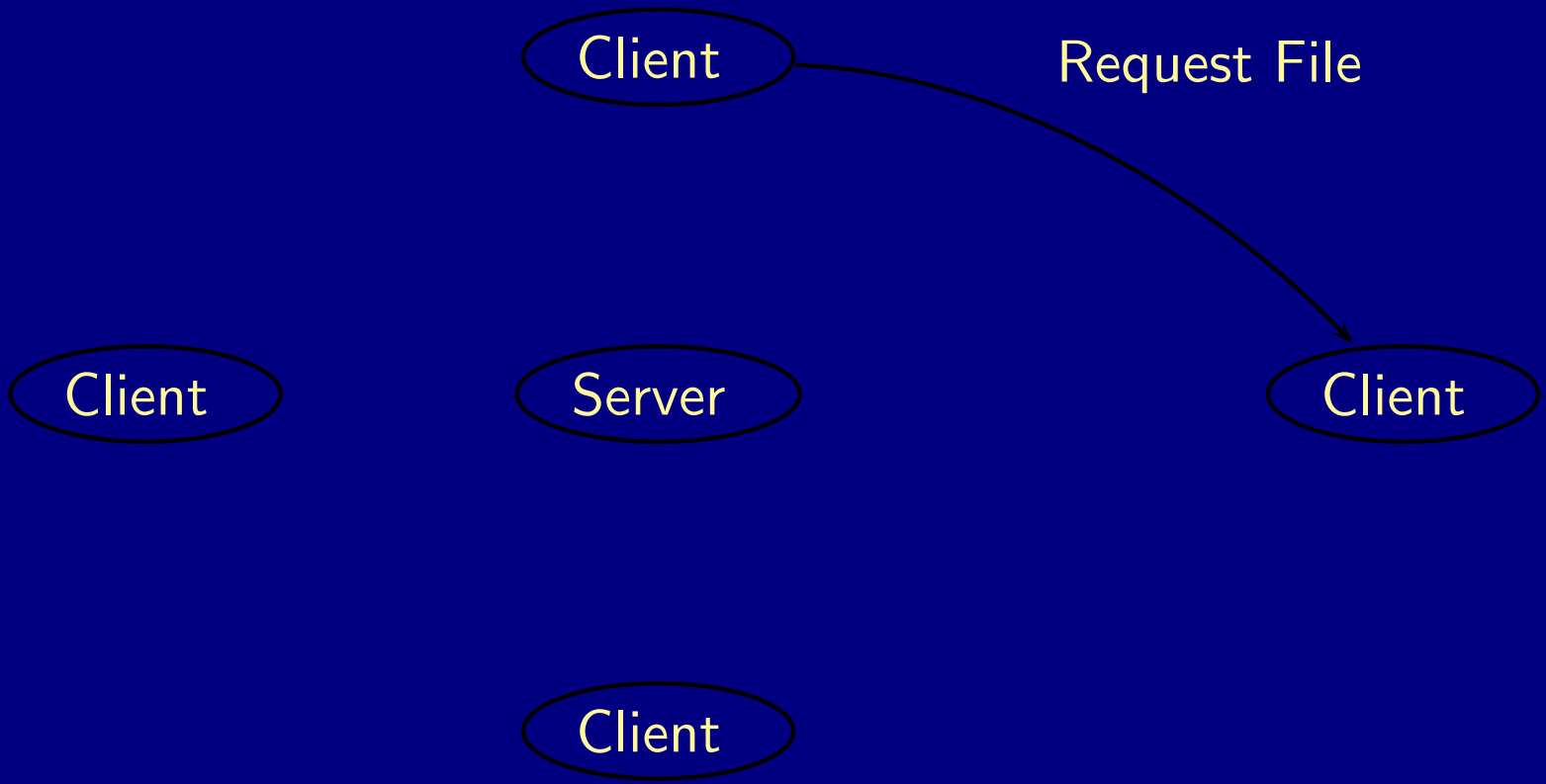
First Generation P2P – Napster



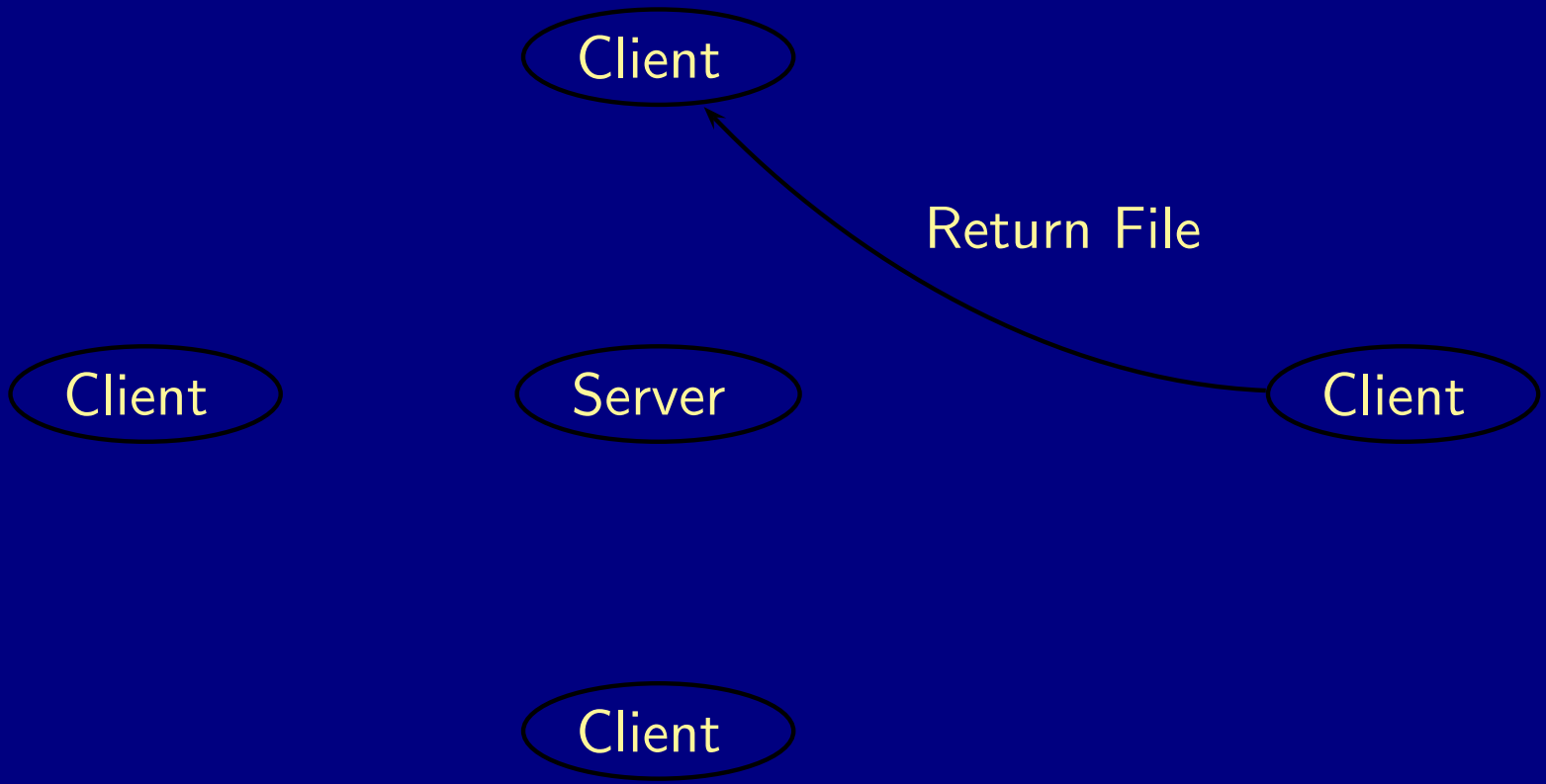
First Generation P2P – Napster



First Generation P2P – Napster



First Generation P2P – Napster



First Generation P2P – Napster

- Problems with Napster like protocols

First Generation P2P – Napster

- Problems with Napster like protocols
  - Single point of failure – The server.

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.

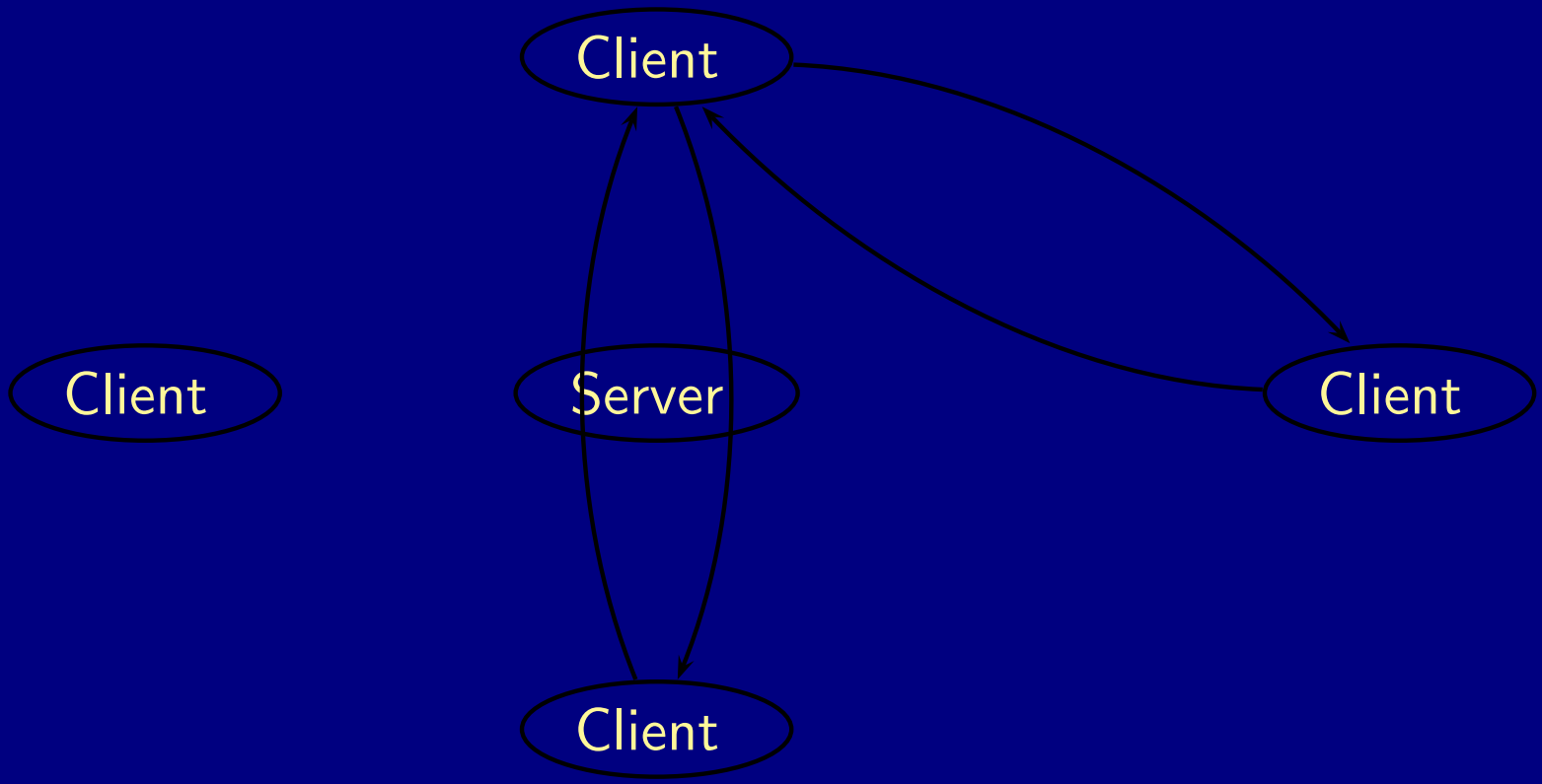
First Generation P2P – Napster

- 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.

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

First Generation P2P – Gnutella

- Again files are distributed across the network
- But no central server

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

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

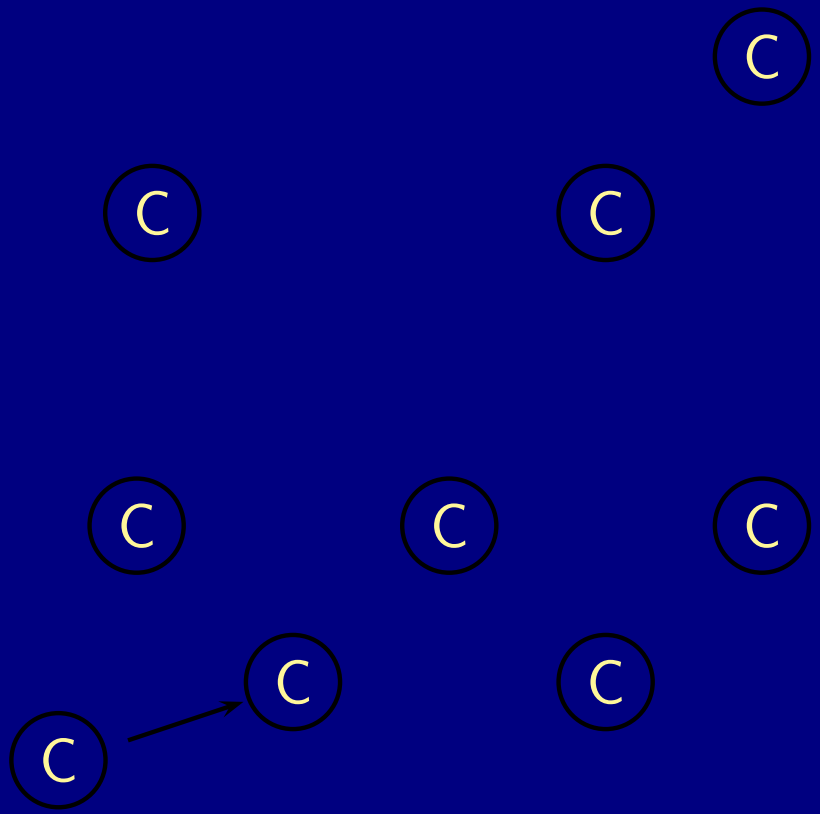
First Generation P2P – Gnutella

- 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

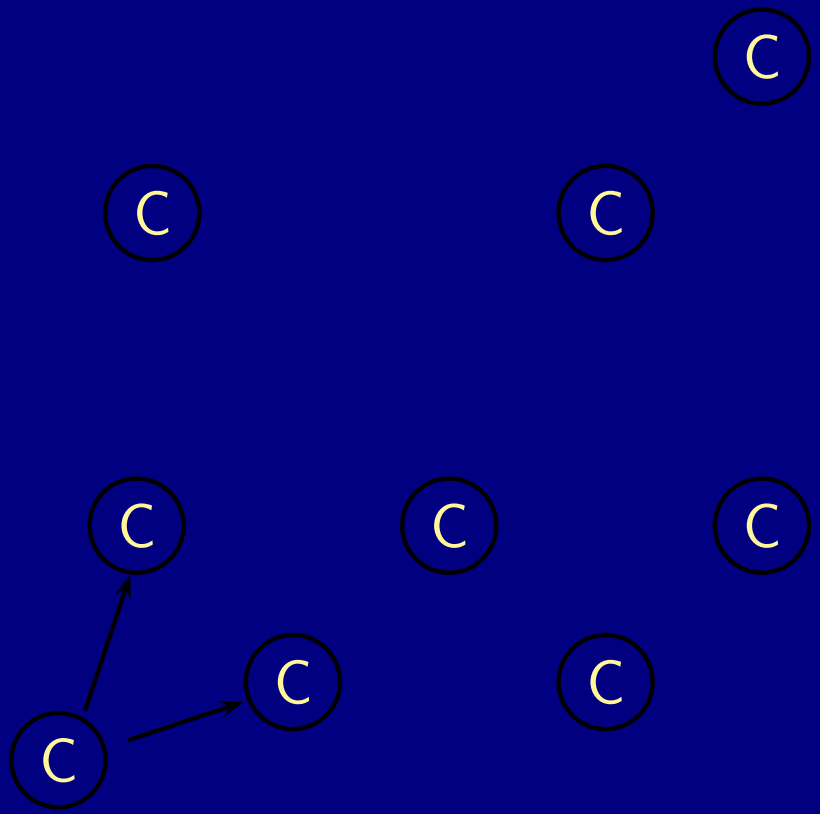
First Generation P2P – Gnutella

- 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).

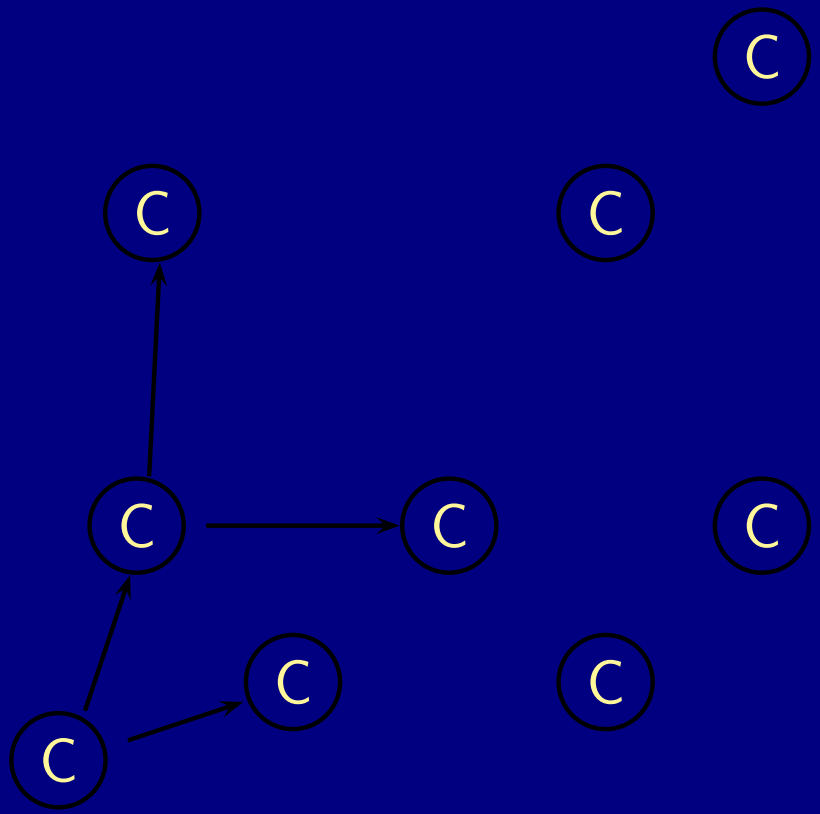
First Generation P2P – Gnutella



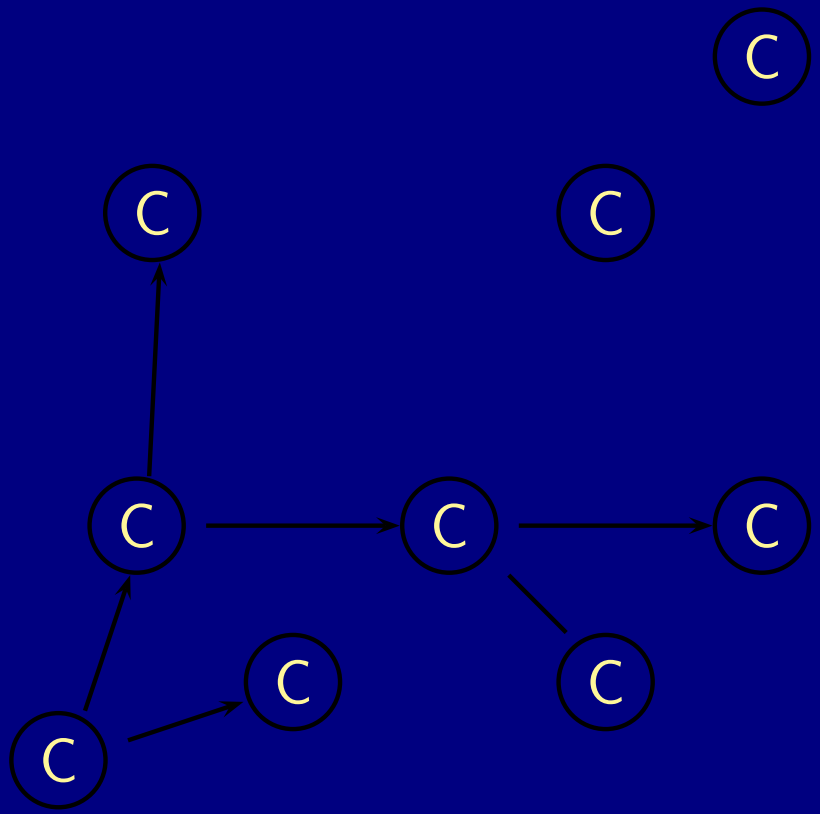
First Generation P2P – Gnutella



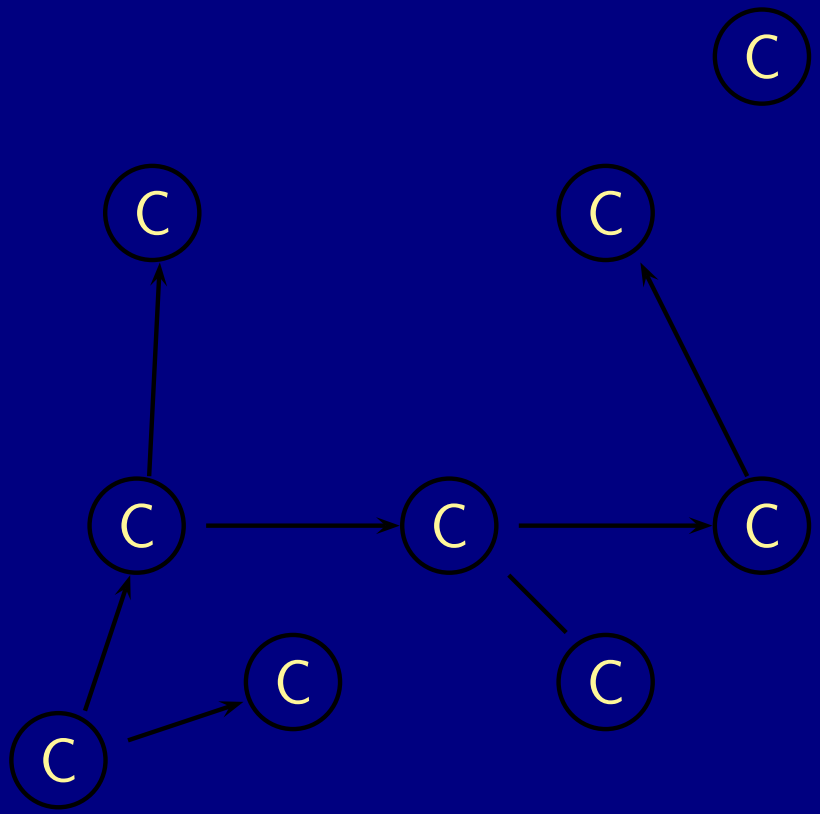
First Generation P2P – Gnutella



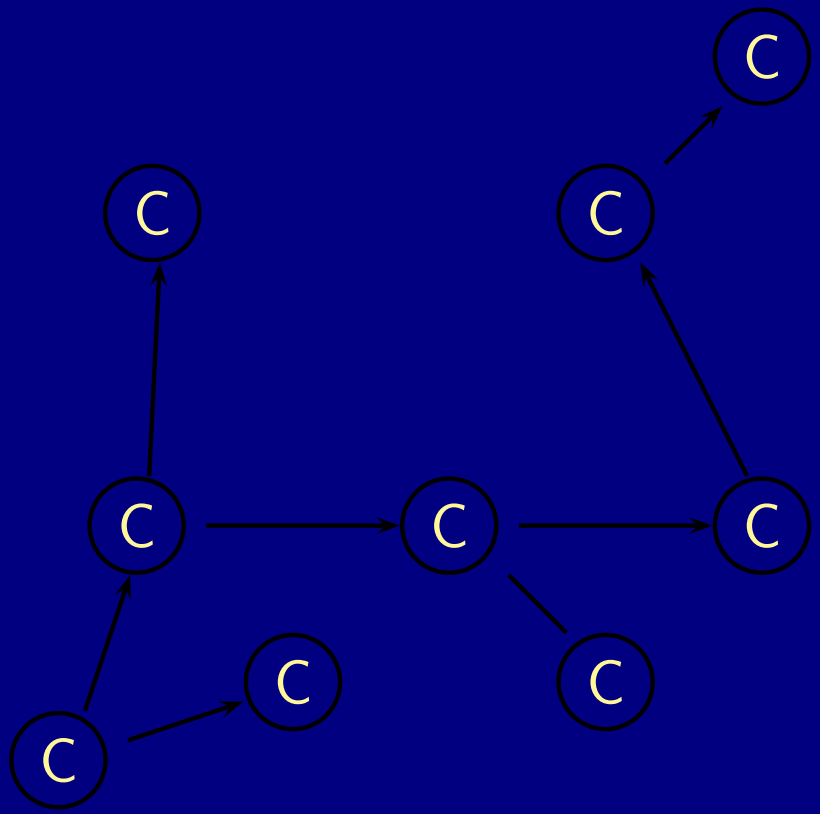
First Generation P2P – Gnutella



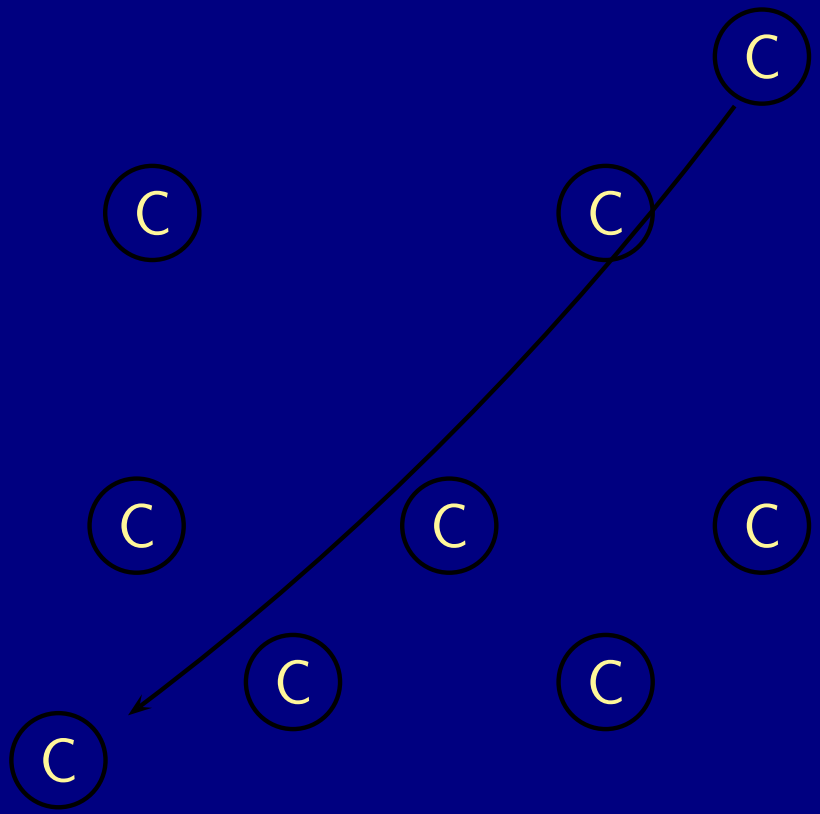
First Generation P2P – Gnutella



First Generation P2P – Gnutella



First Generation P2P – Gnutella



First Generation P2P – Scalability

- Napster did not last long enough to test scalability issues, but

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.

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.

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.

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?

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?

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?

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

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?

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.

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.

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.

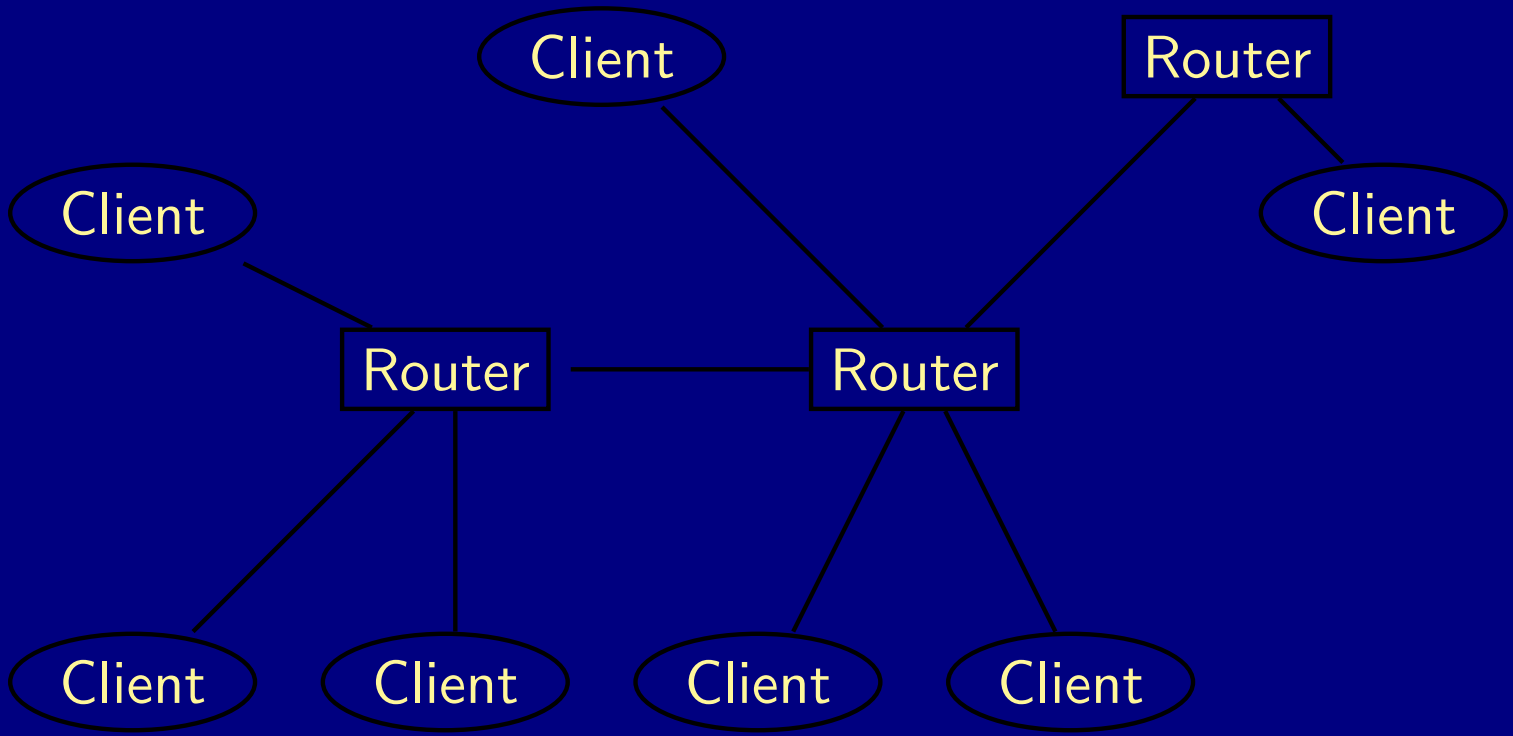
Overlay networks

- 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.

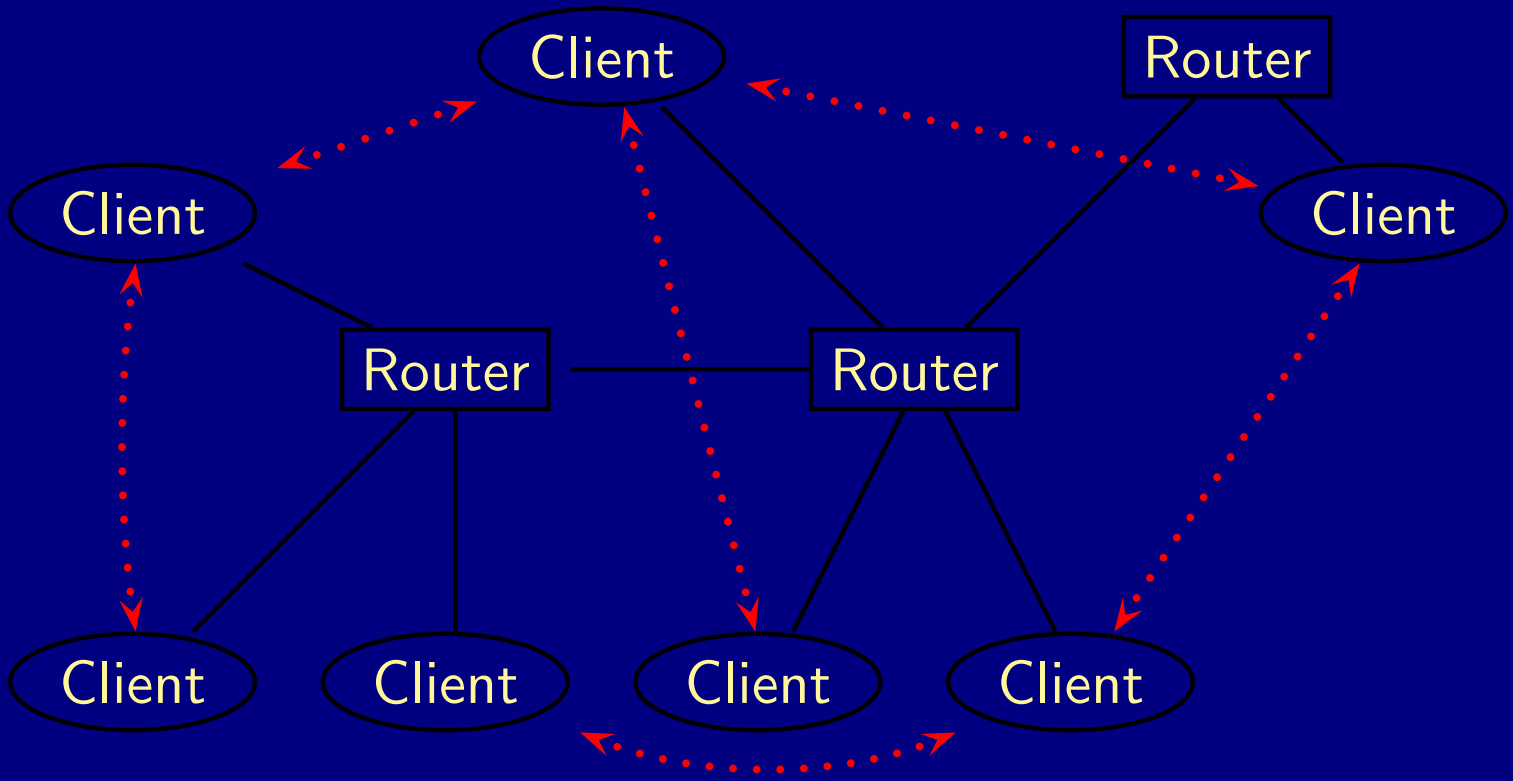
Overlay networks

- 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



Overlay Networks

- The overlay network has a different notion of neighbour to the underlying network.

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.

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.

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*

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

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.

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.

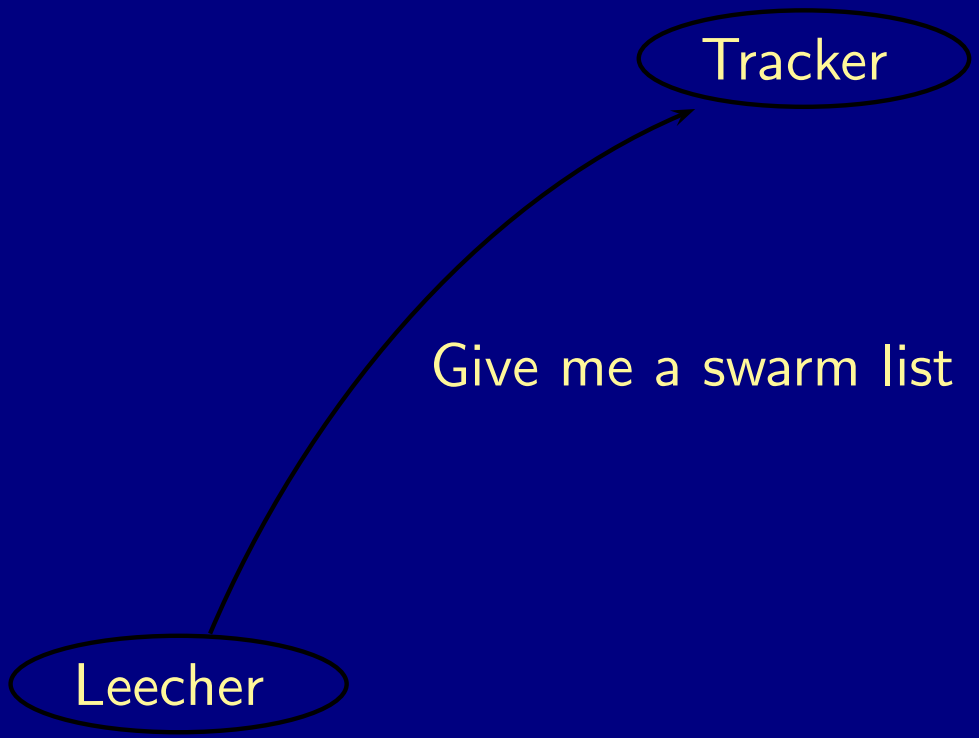
Bit torrent

- 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.

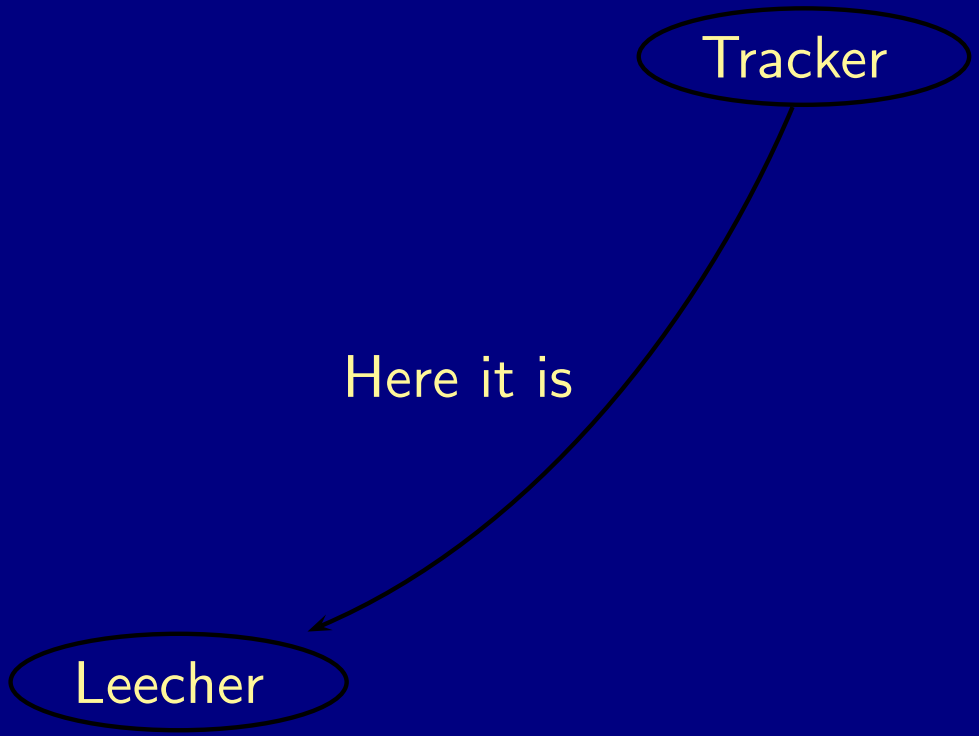
Bit torrent

- 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

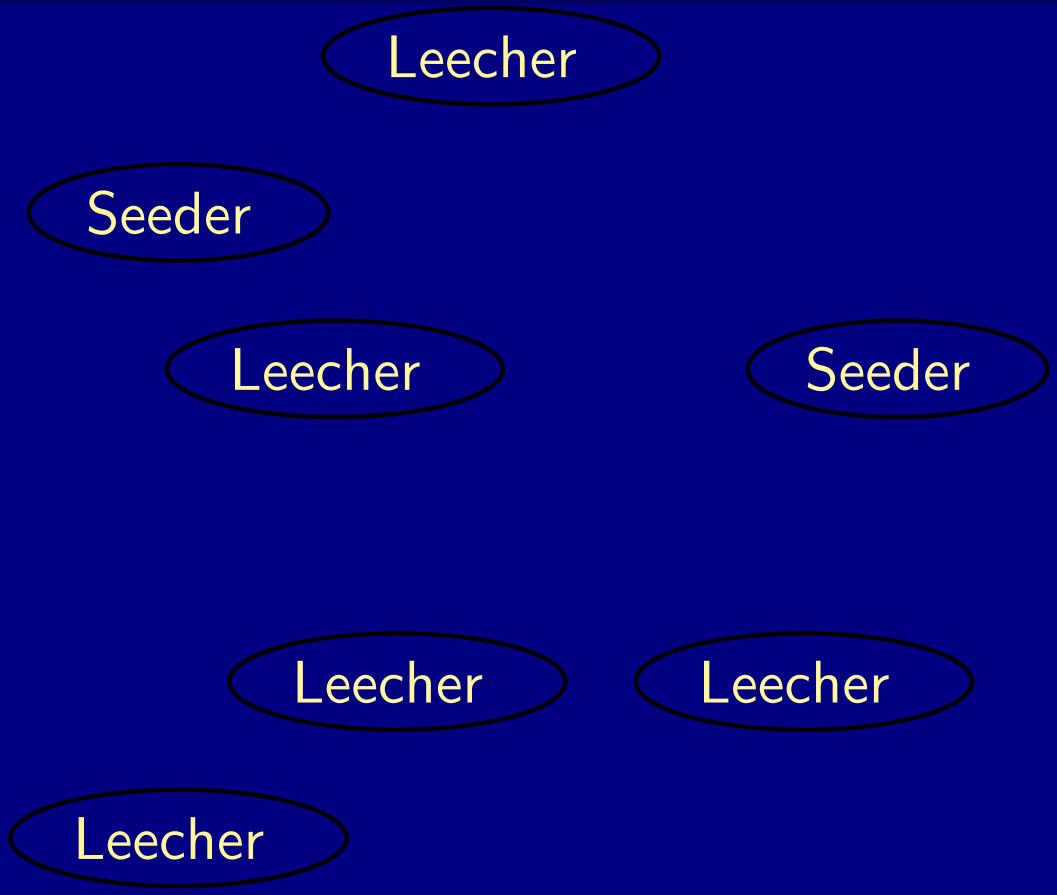


Bit torrent



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

Bit torrent



Bit torrent

- The actual mechanism of how the client downloads from the current list of seeders and leechers (the swarm) can be quite complicated.

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?

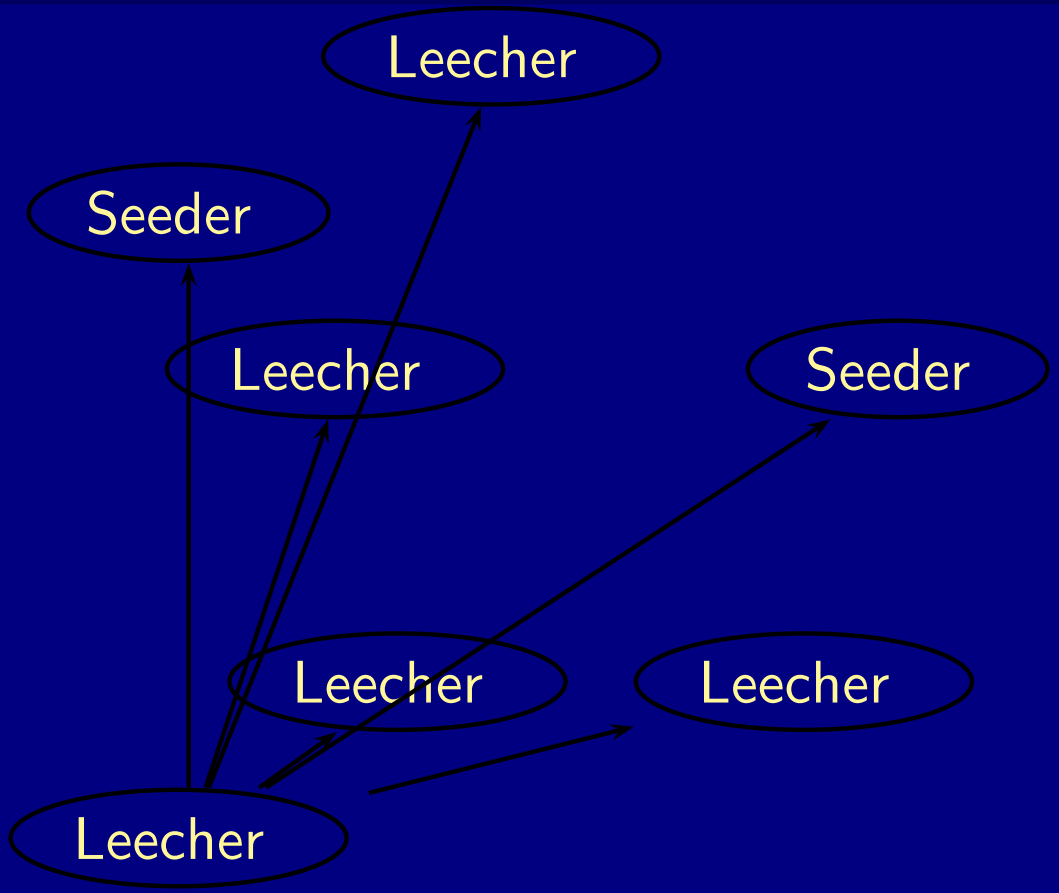
Bit torrent

- 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.

Bit torrent

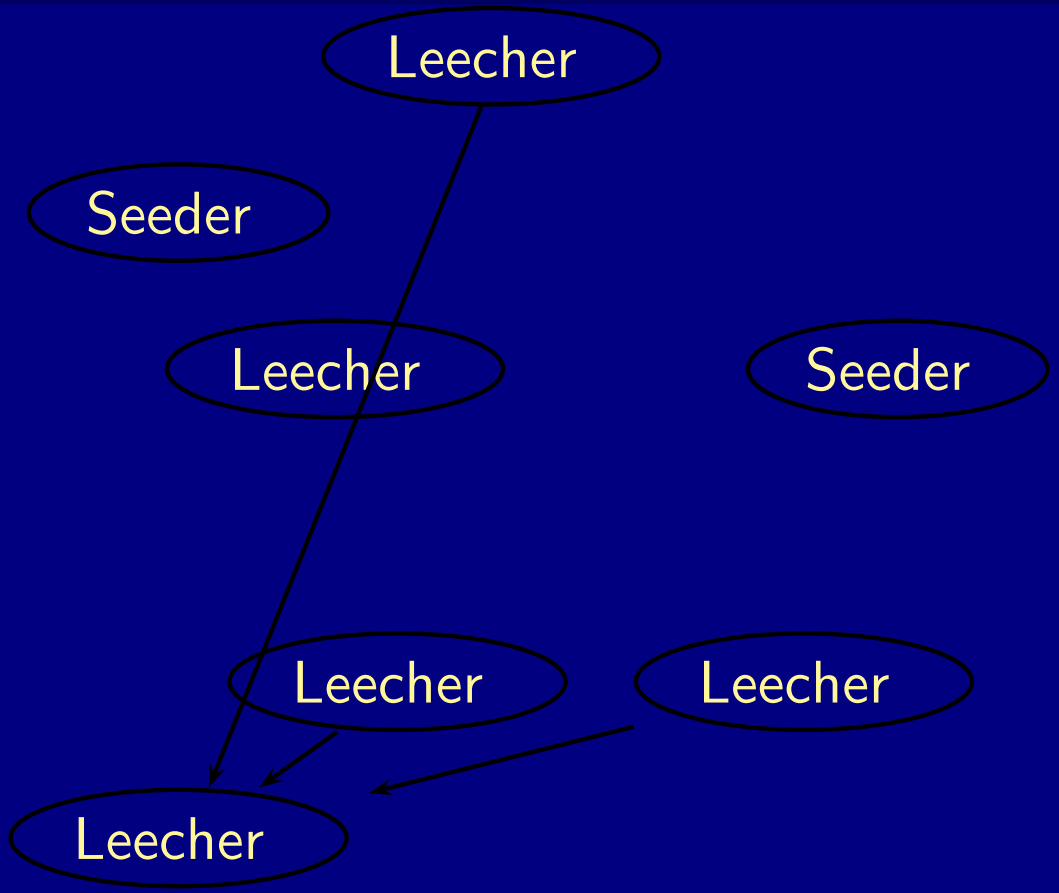
- The actual mechanism of how the client downloads from the current list of seeders and leechers (the swarm) can be quite complicated.
- Essentially the client asks each other client what pieces does it have?
- Then according to some strategy the client then asks for chunks form the other members of 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



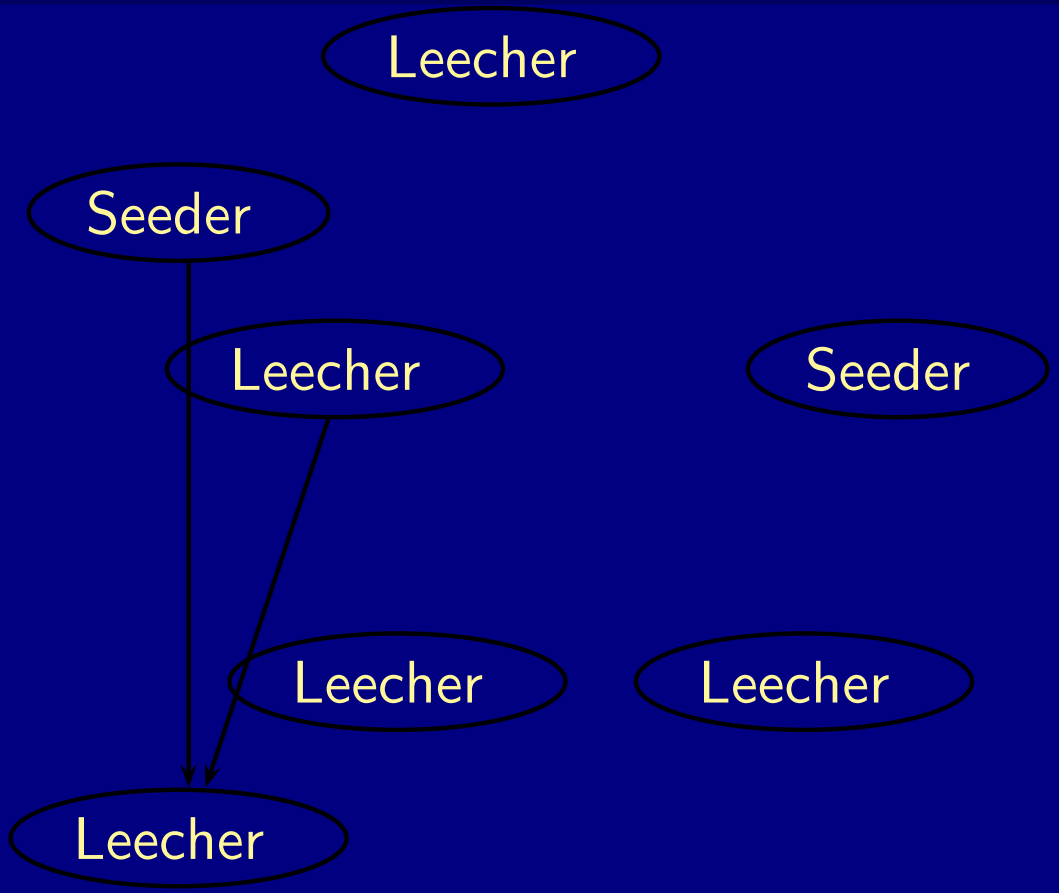
- What chunks do you have?

Bit torrent



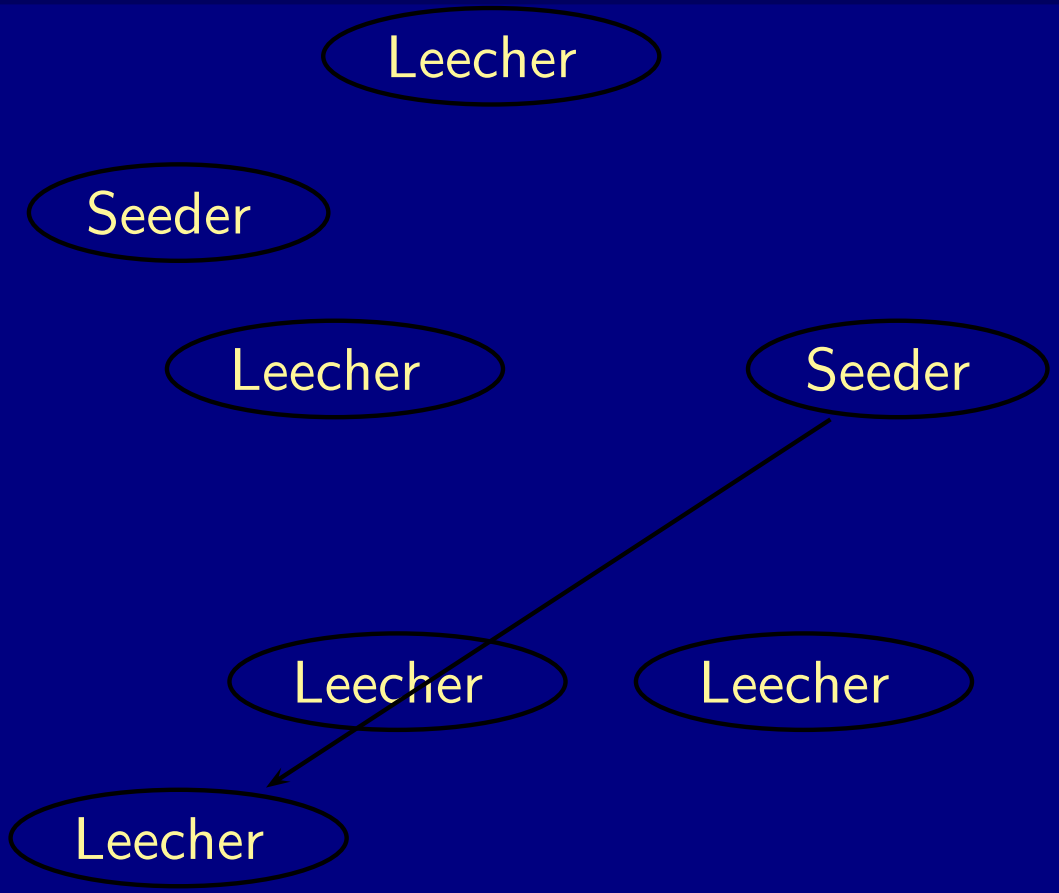
- Here is a list of chunks that I have.

Bit torrent



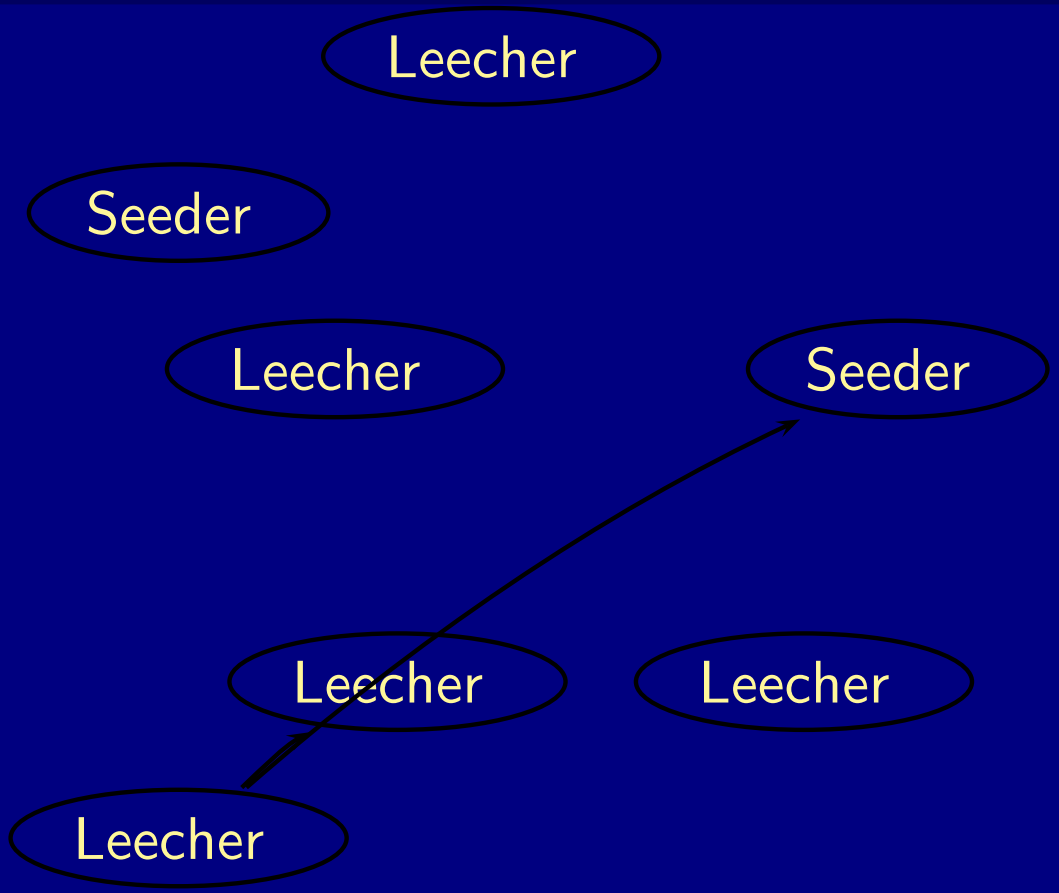
- Here is a list of chunks that I have.

Bit torrent



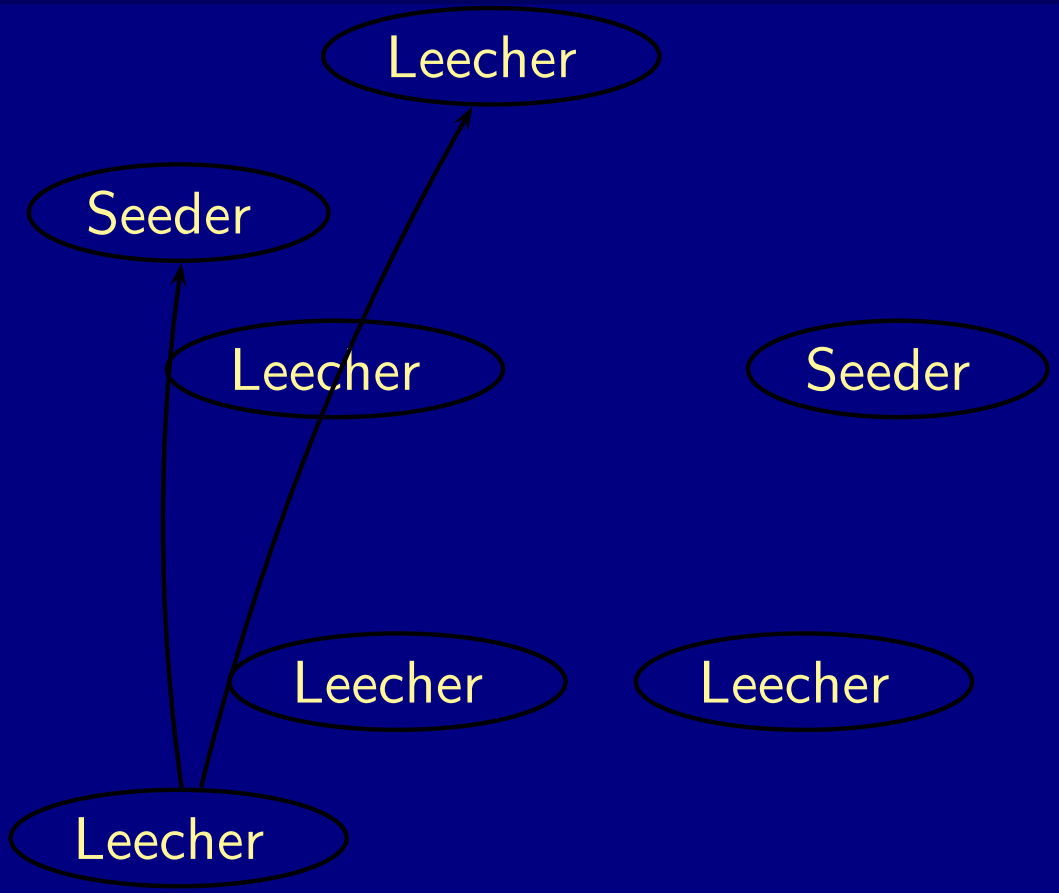
- Here is a list of chunks that I have.

Bit torrent



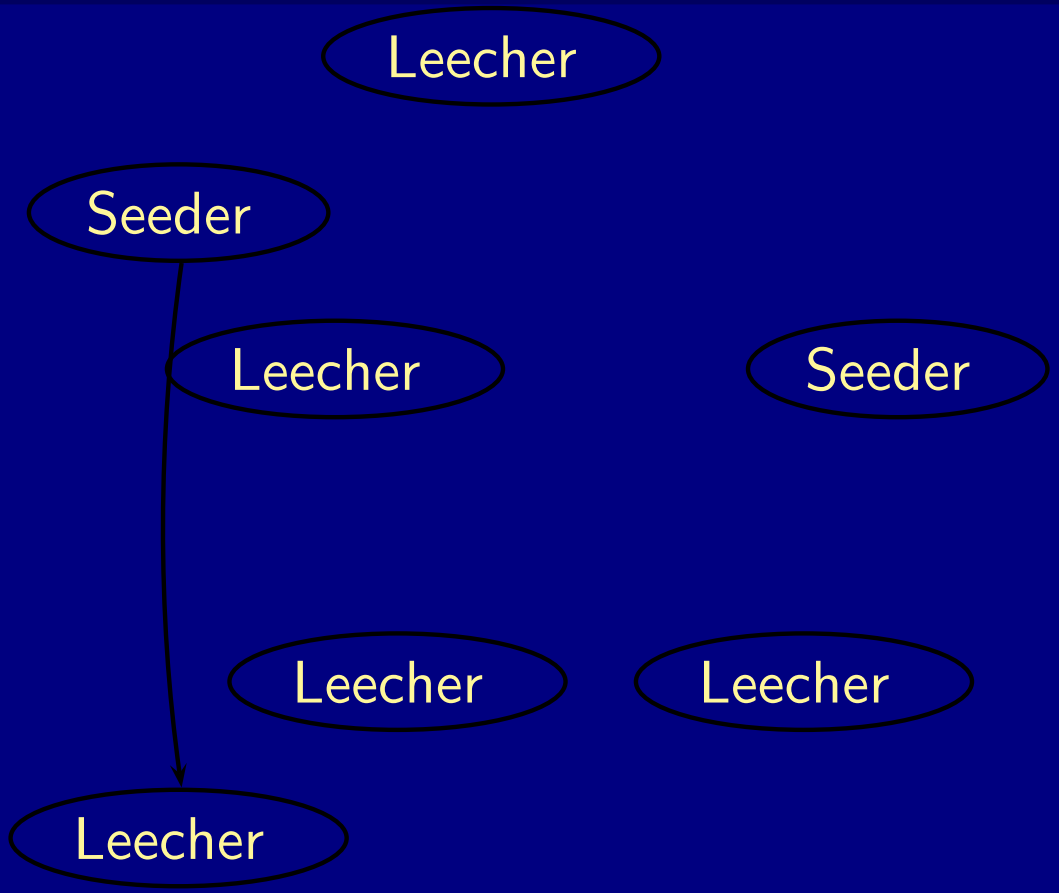
- Request some chunk

Bit torrent



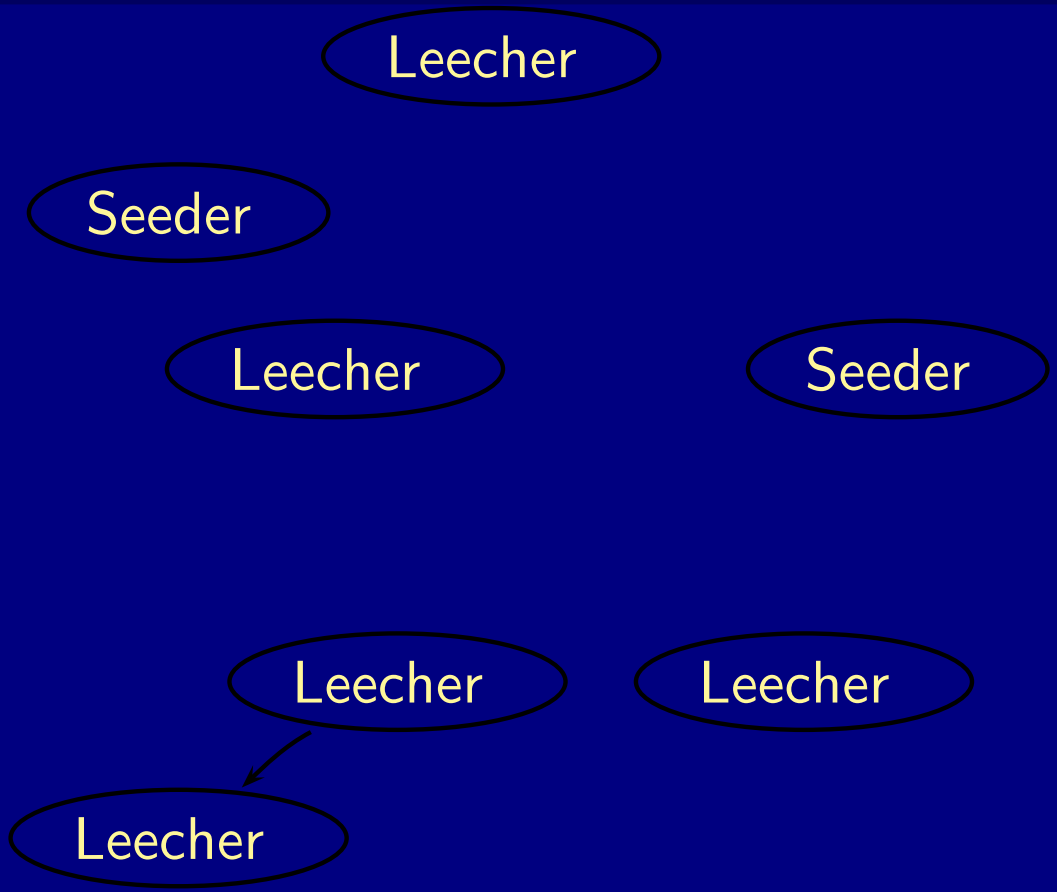
- Request some chunk

Bit torrent



- Here are the requested chunks.

Bit torrent



- Here are the requested chunks.

Bit torrent

- Note, at all times the downloading client is serving requests for chunks. Helps with Tit for Tat.

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.

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.

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.

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.

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).

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.

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.

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 $m < n$.

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 $k - 1$.

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.

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

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.