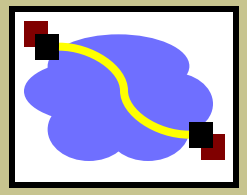
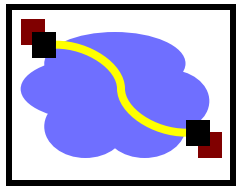


416 Distributed Systems

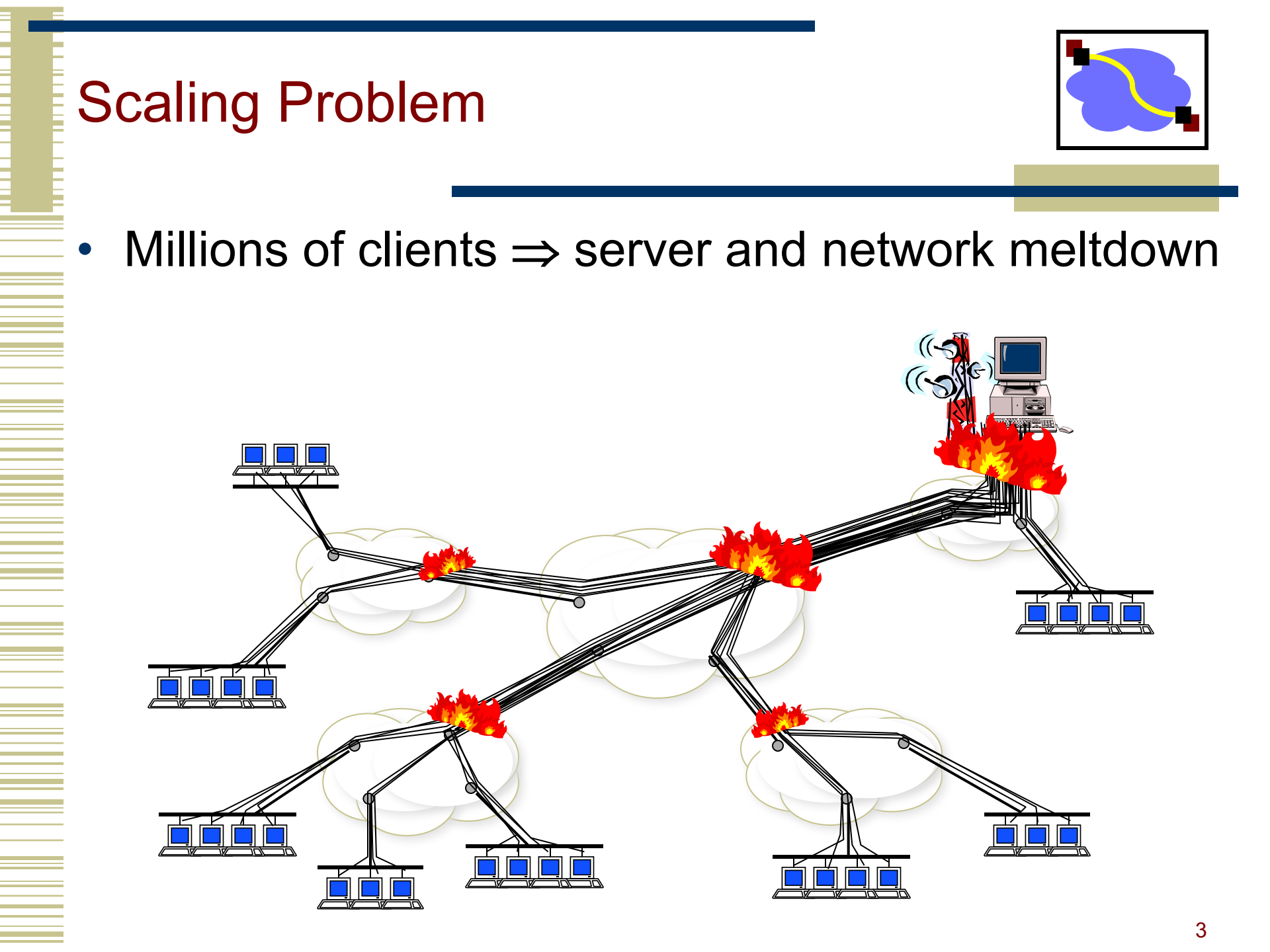
Jan 31, Peer-to-Peer

Outline



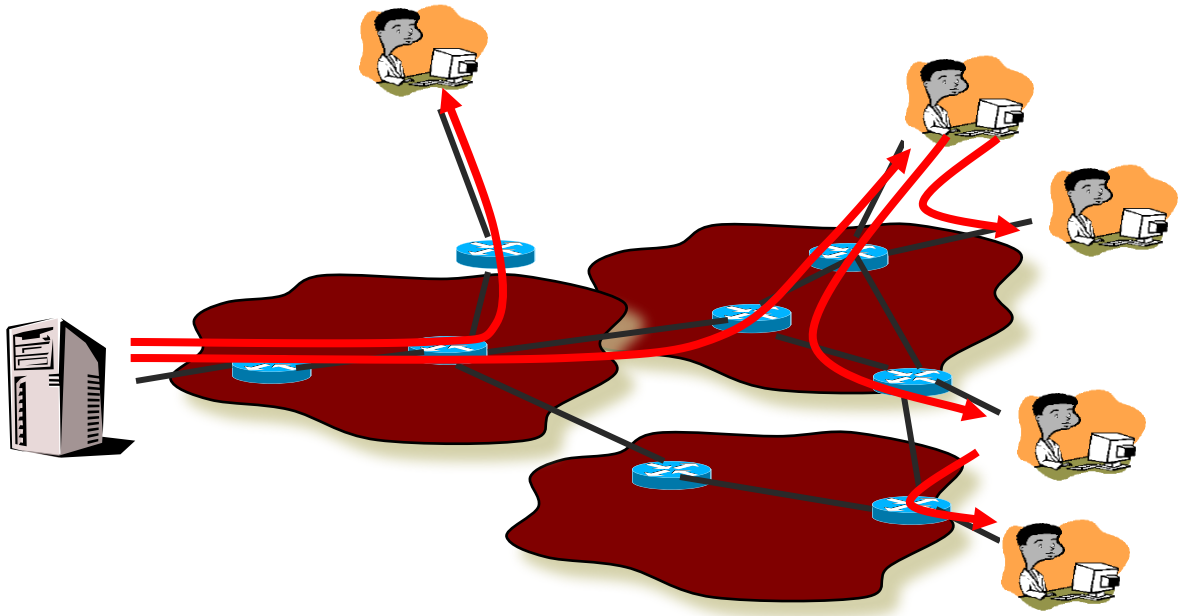
- P2P Lookup Overview
- Centralized/Flooded Lookups
- Routed Lookups – Chord
- BitTorrent

Scaling Problem



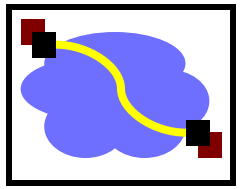
- Millions of clients ⇒ server and network meltdown

P2P System



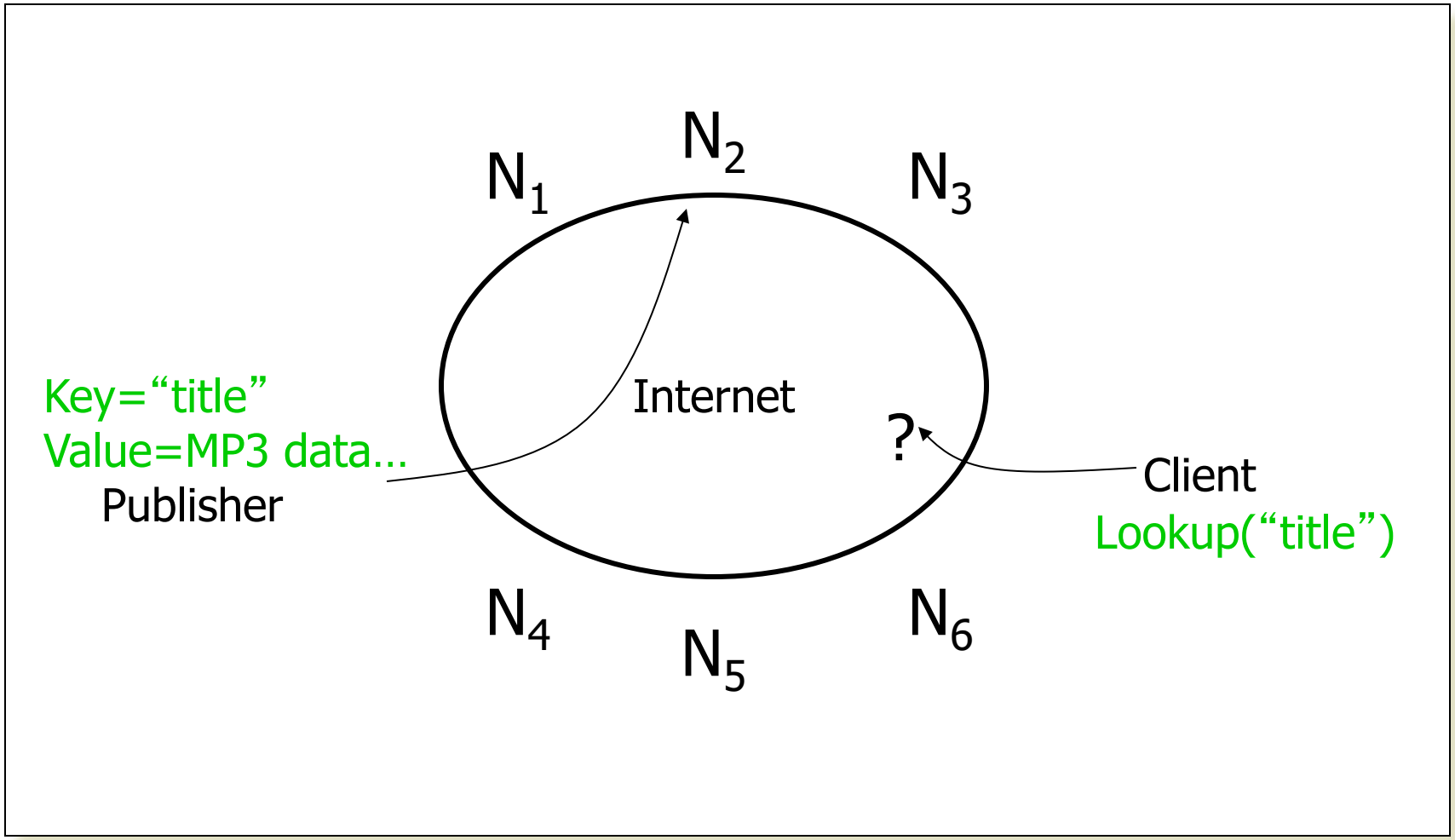
- Leverage the resources of client machines (peers)
  - Traditional: Computation, storage, bandwidth
  - Non-traditional: Geographical diversity, mobility, sensors!

Peer-to-Peer (storage) Networks

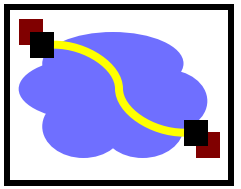


- Typically each member stores/provides access to content
- Basically a replication system for files
  - Always a tradeoff between possible location of files and searching difficulty
  - Peer-to-peer allow files to be anywhere → searching is the challenge
  - Dynamic member list makes it more difficult
- What other systems have similar goals?
  - Routing, DNS

The Lookup Problem

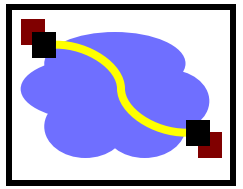


Searching



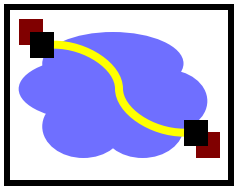
- Needles vs. Haystacks
  - Searching for top 40, or an obscure punk track from 1981 that nobody's heard of?
- Search expressiveness
  - Whole word? Regular expressions? File names? Attributes? Whole-text search?

Framework



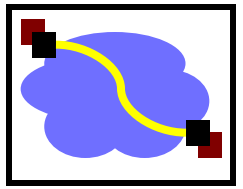
- Common Primitives:
  - **Join**: how do I begin participating?
  - **Publish**: how do I advertise my file?
  - **Search**: how to I find a file?
  - **Fetch**: how to I retrieve a file?

Outline



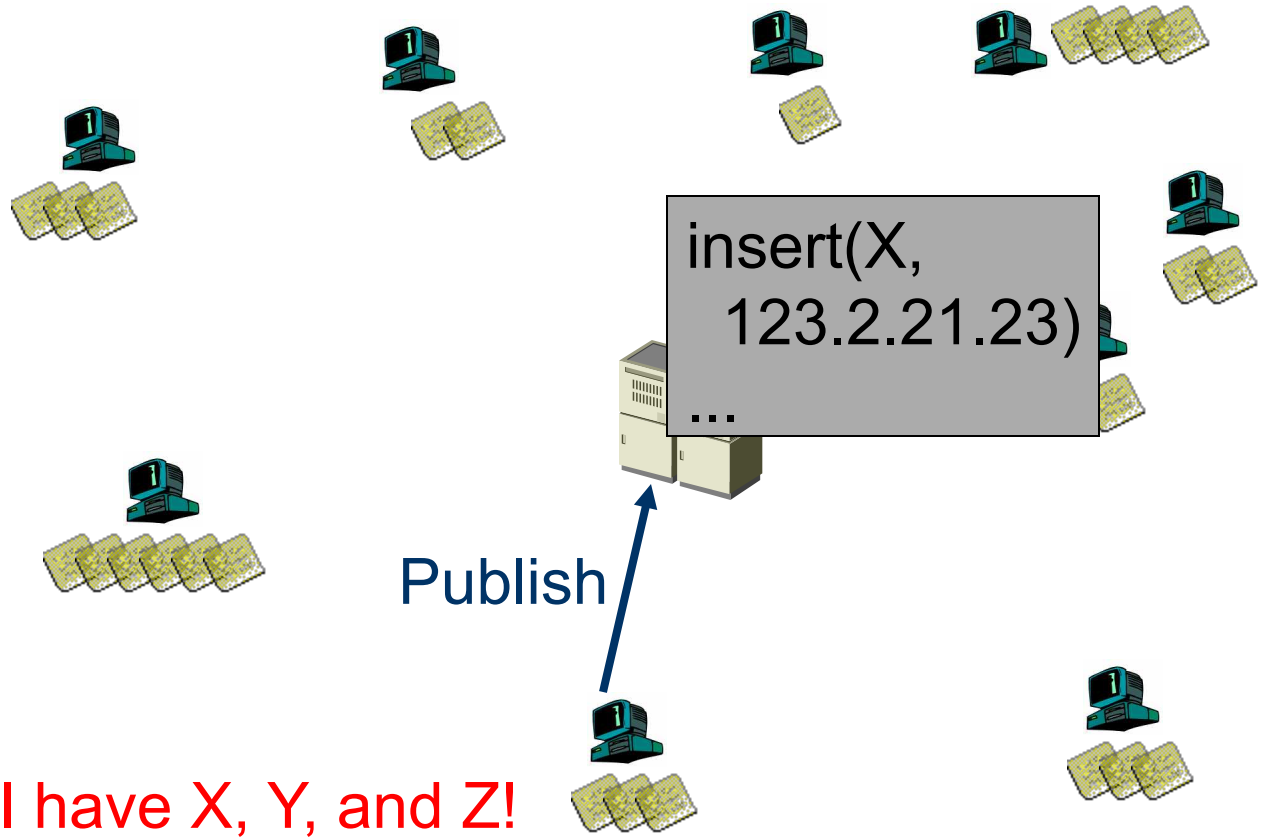
- P2P Lookup Overview
- Centralized/Flooded Lookups
- Routed Lookups – Chord
- BitTorent

Napster: Overiew

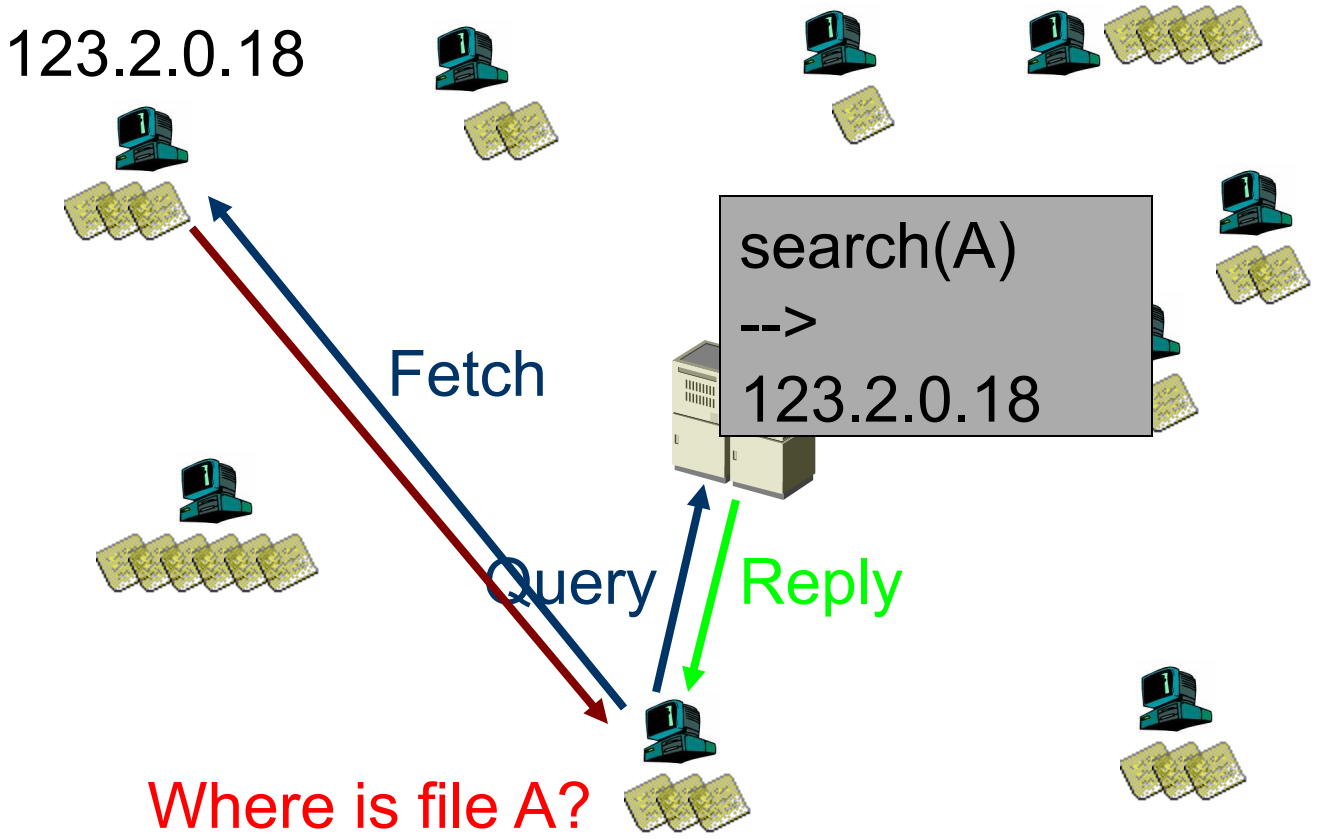


- Centralized Database:
  - **Join:** on startup, client contacts central server
  - **Publish:** reports list of files to central server
  - **Search:** query the server => return someone that stores the requested file
  - **Fetch:** get the file directly from peer

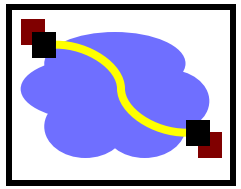
Napster: Publish



Napster: Search

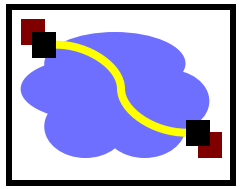


Napster: Discussion



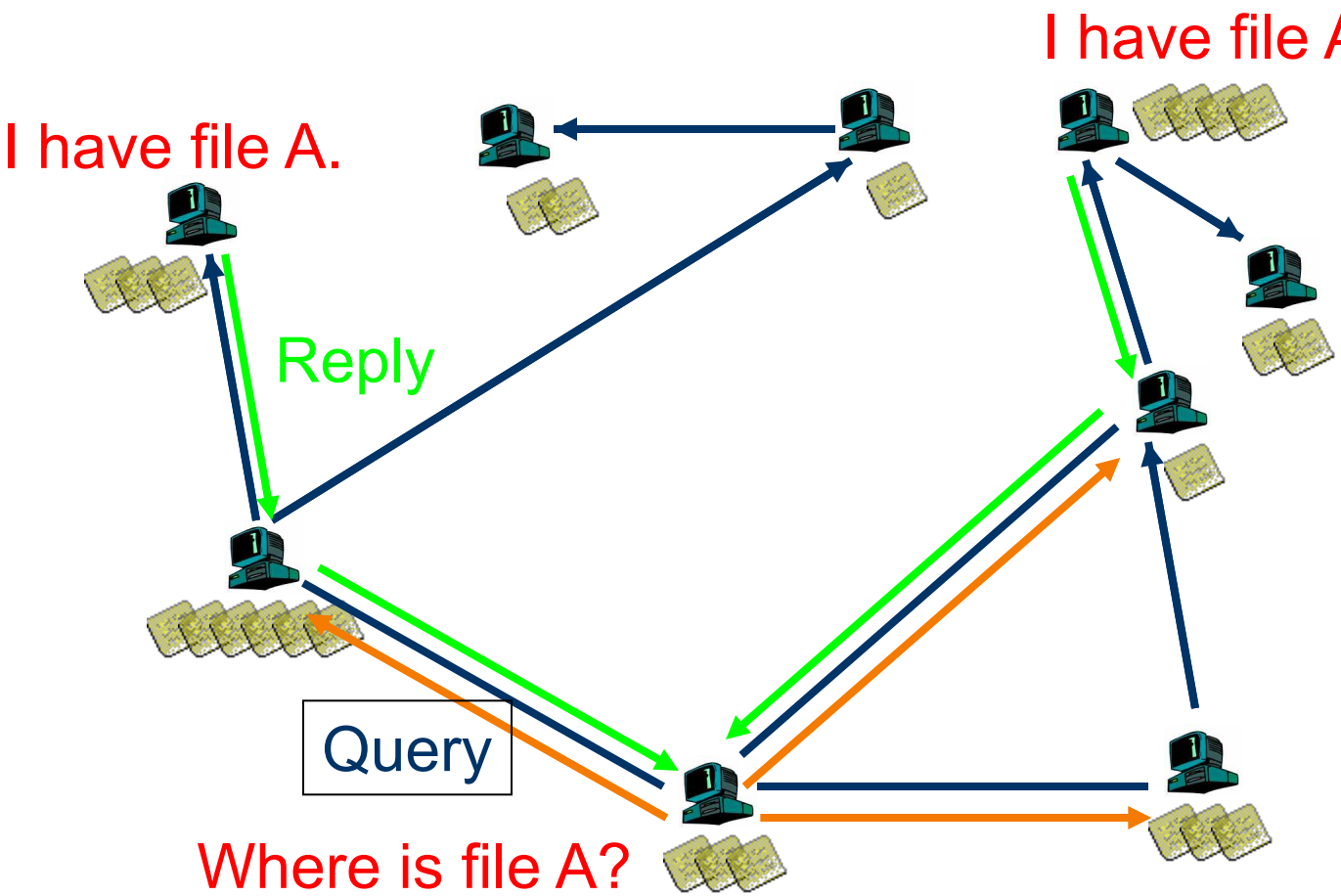
- Pros:
  - Simple
  - Search scope is O(1)
  - Controllable (pro or con?)
- Cons:
  - Server maintains O(N) State
  - Server does all processing
  - Single point of failure

“Old” Gnutella: Overview

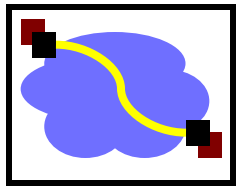


- Query Flooding:
  - **Join**: on startup, client contacts *a few* other nodes; these become its “neighbors”
    - “unstructured overlay”
  - **Publish**: no need
  - **Search**: ask neighbors, who ask their neighbors, and so on... when/if found, reply to sender.
    - TTL limits propagation
  - **Fetch**: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

Gnutella: Search

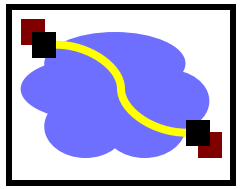


Gnutella: Discussion

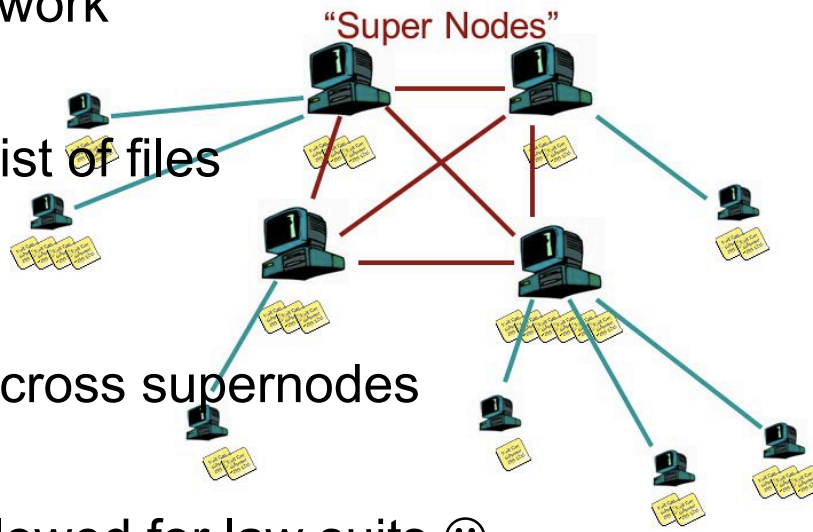


- Pros:
  - Fully de-centralized
  - Search cost distributed
  - Processing @ each node permits powerful search semantics
- Cons:
  - Search scope is O(*N*)
  - Search time is O(???)
  - Nodes leave often, network unstable
- TTL-limited search works well for haystacks.
  - For scalability, does NOT search every node. May have to re-issue query later; no guarantee that it will find the file!

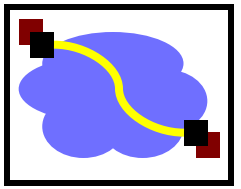
Flooding: Gnutella, **Kazaa**



- Modifies the Gnutella protocol into two-level hierarchy
  - Hybrid of Gnutella and Napster
- Supernodes
  - Nodes that have better connection to Internet
  - Act as temporary indexing servers for other nodes
  - Help improve the stability of the network
- Standard nodes
  - Connect to supernodes and report list of files
  - Allows slower nodes to participate
- Search
  - Broadcast (Gnutella-style) search across supernodes
- Disadvantages
  - Kept a centralized registration → allowed for law suits ☹

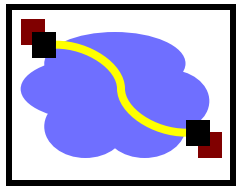


Outline



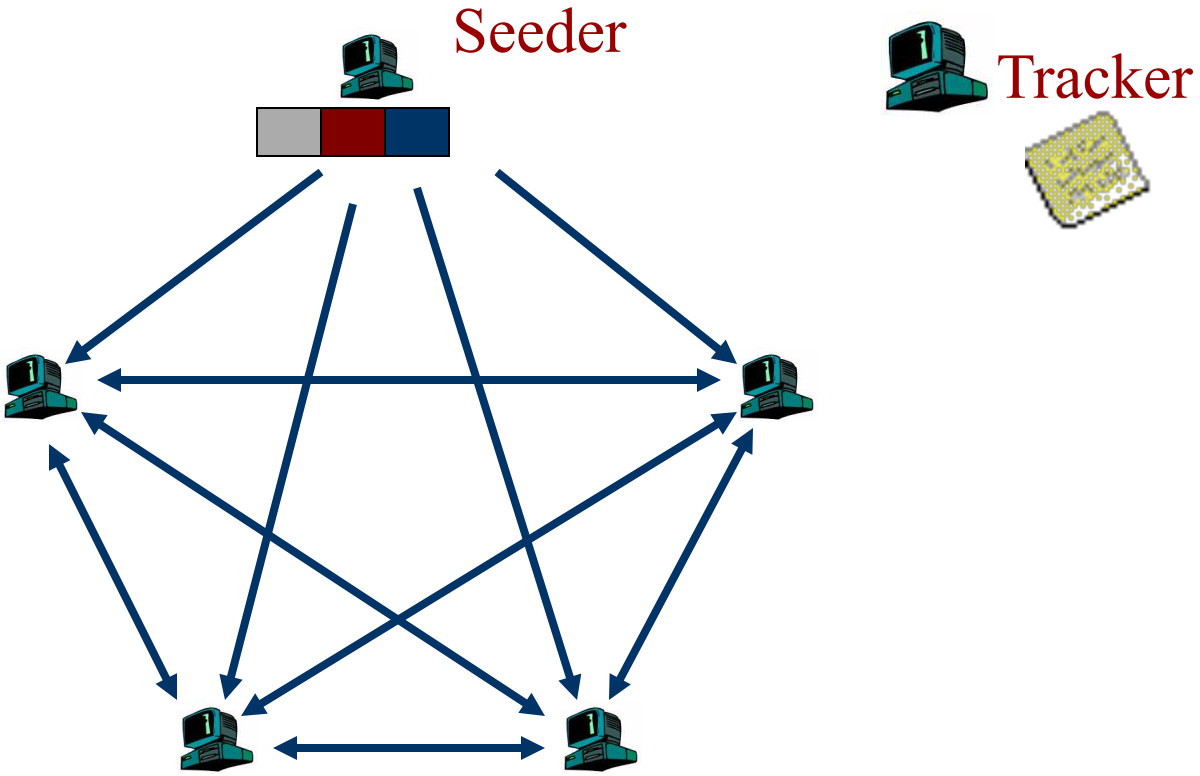
- P2P Lookup Overview
- Centralized/Flooded Lookups
- Routed Lookups – Chord
- BitTorent

BitTorrent: Overview

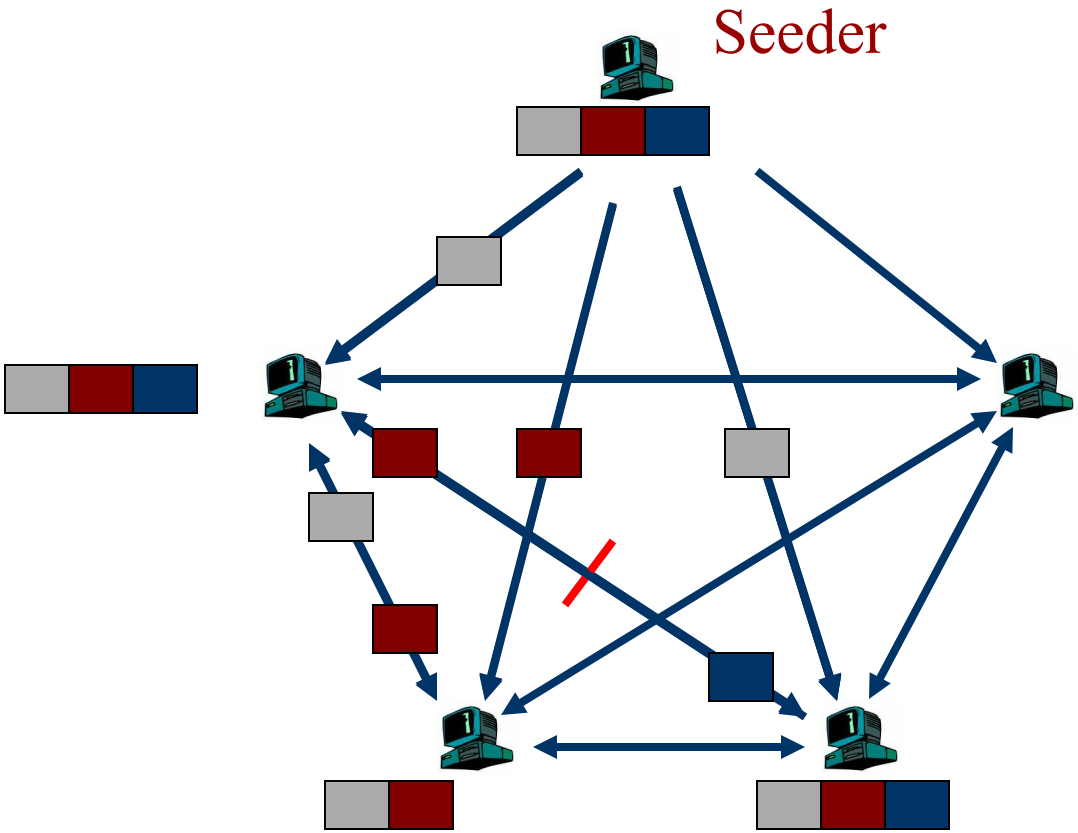


- File swarming:
  - **Join**: contact centralized “tracker” server, get a list of peers.
  - **Publish**: Run a tracker server.
  - **Search**: Out-of-band. E.g., use Google to find a tracker for the file you want.
  - **Fetch**: Download chunks of the file from your peers. Upload chunks you have to them.
- Big differences from Napster:
  - Chunk based downloading
  - “few large files” focus
  - Anti-freeloading mechanisms

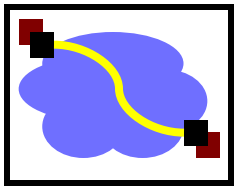
BitTorrent: Publish/Join



BitTorrent: Fetch

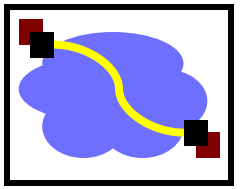


BitTorrent: Sharing Strategy



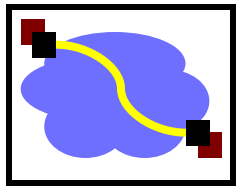
- Employ "Tit-for-tat" sharing strategy
  - A is downloading from some other people
    - A will let the fastest N of those download from it
  - Be optimistic: occasionally let freeloaders download
    - *Optimistic unchoke*
    - Otherwise no one would ever start!
    - Also allows you to discover better peers to download from when they reciprocate
- Goal: Pareto Efficiency
  - Game Theory: "No change can make anyone better off without making others worse off"
  - Does it work? How would you cheat?
  - (not perfectly, but perhaps good enough?)

BitTorrent: Summary



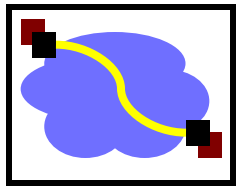
- Pros:
  - Works reasonably well in practice
  - Gives peers incentive to share resources; avoids freeloaders
- Cons:
  - Pareto Efficiency claim is not true … a lie
  - Central tracker server needed to bootstrap swarm
    - Alternate tracker designs exist (e.g., DHT-based trackers)

A Peer-to-peer Google?



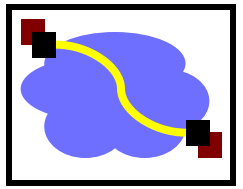
- Complex intersection queries (“the” + “who”)
  - Billions of hits for each term alone
- Sophisticated ranking
  - Must compare many results before returning a subset to user
- Very, very hard for a DHT / p2p system
  - Need high inter-node bandwidth
  - (This is exactly what Google does - massive clusters)

Writable, persistent p2p



- Do you trust your data to 100,000 monkeys?
- Node availability hurts
  - Ex: Store 5 copies of data on different nodes
  - When someone goes away, you must replicate the data they held
  - Hard drives are *huge*, but edge network upload bandwidth is tiny
  - May take days to upload contents of a hard drive. P2P replication/fault-tolerance expensive.

P2P: Summary



- Many different styles; remember pros and cons of each
  - centralized, flooding, swarming, and structured routing
- Lessons learned:
  - Single points of failure are very bad
  - Flooding messages to everyone is bad
  - Underlying network topology is important
  - Not all nodes are equal
  - Need incentives to discourage freeloading
  - Privacy and security are important
  - Structure can provide theoretical bounds and guarantees