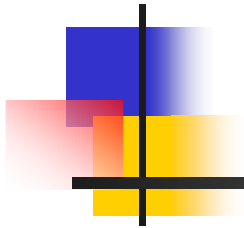
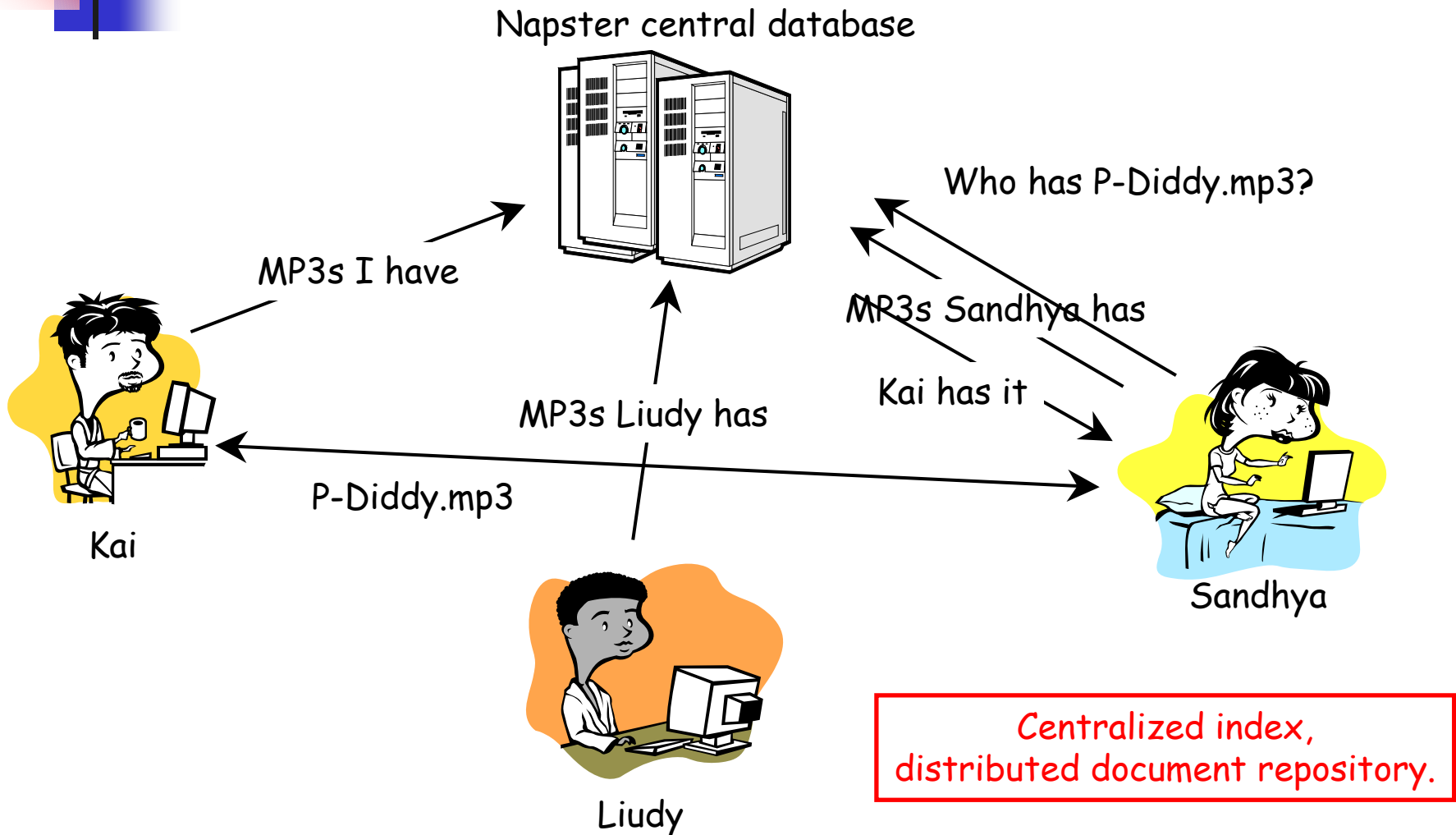


Peer-to-Peer Networks

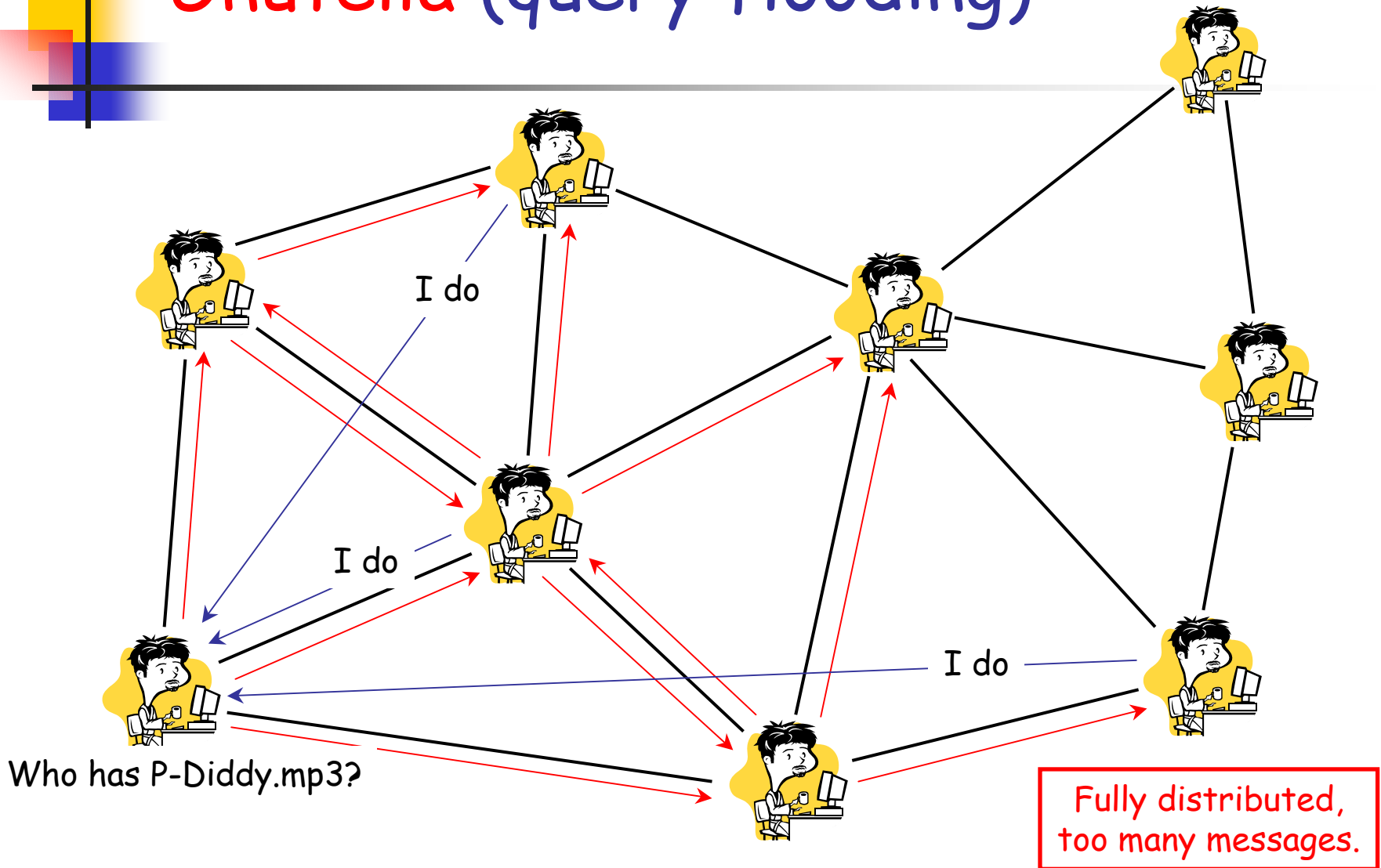


Dept. of Computer Science, University of Rochester

Distributed Search I: Napster (central index)



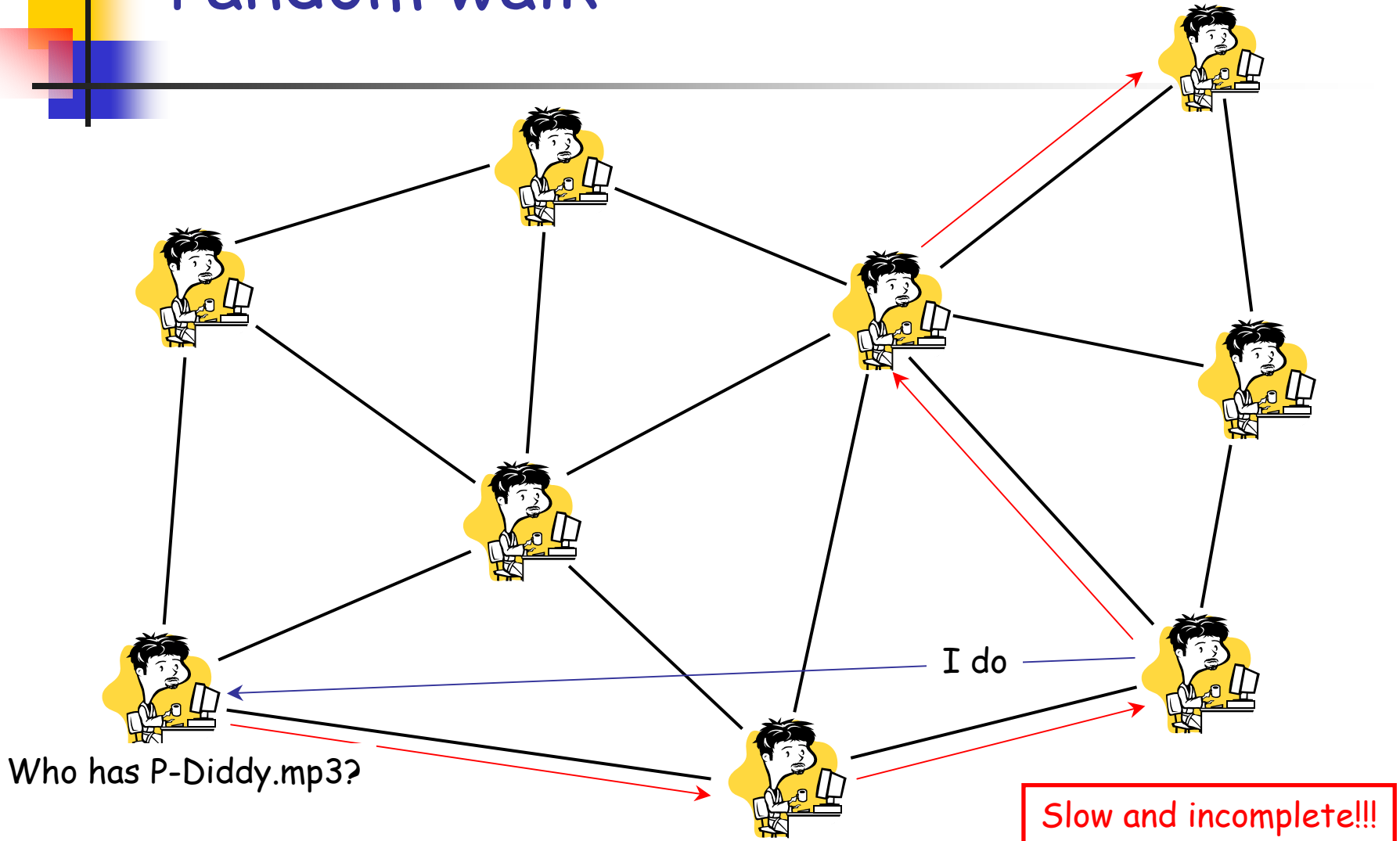
Distributed Search II: Gnutella (query flooding)



Who has P-Diddy.mp3?

Fully distributed,
too many messages.

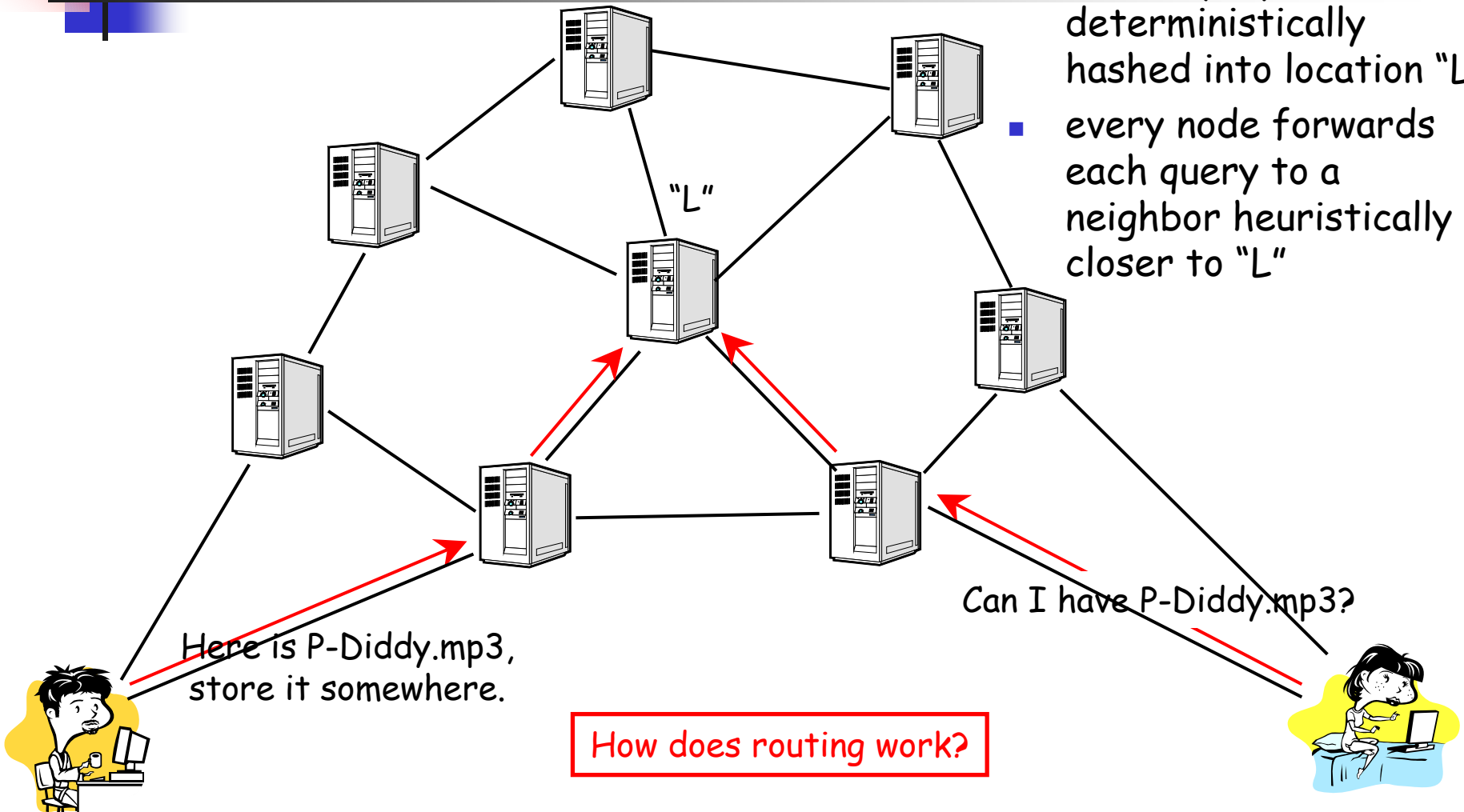
Distributed Search III: random walk



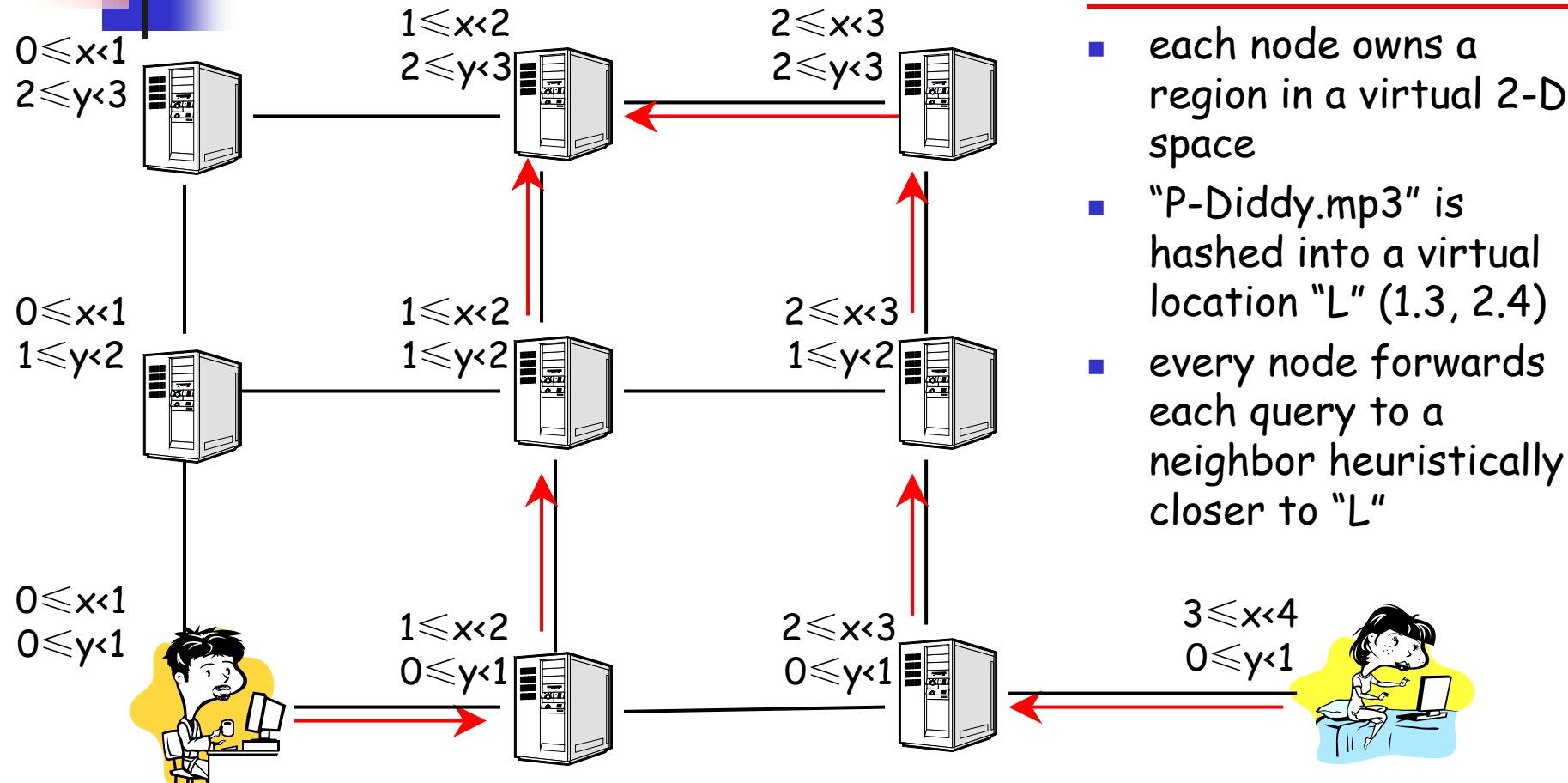
Distributed Search IV:

Object-addressable network

- "P-Diddy.mp3" is deterministically hashed into location "L"
- every node forwards each query to a neighbor heuristically closer to "L"



Scalable Object-addressable Net



Content-addressable net:

- each node owns a region in a virtual 2-D space
- "P-Diddy.mp3" is hashed into a virtual location "L" (1.3, 2.4)
- every node forwards each query to a neighbor heuristically closer to "L"

Here is P-Diddy.mp3,
store it somewhere.

Can I have P-Diddy.mp3?



Analysis

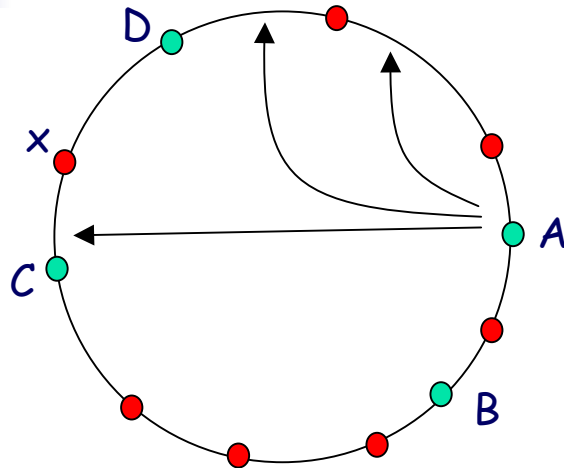
Content-addressable network [Ratnasamy et al. 2001]:

- each node owns a region in a virtual 2-D space
- each object is hashed into a virtual location "L"
- every node forwards each query to a neighbor heuristically closer to "L"

Questions:

- Space consumption? Lookup cost for a network of N nodes?
 - 4 links per node; $O(N^{1/2})$
- Can the lookup cost be improved?
- How to take into account the actual link latency?

Chord [Stoica et al. 2001]



● hash code of a data object

● hash code of a network node

- each object is mapped to the first node in the clockwise direction on the ring.
- every node maintains links to
 - the node half-way in the circular ID space
 - the node a quarter-way across the circular ID space
 - ...
- Space consumption? Lookup cost for a network of N nodes?



Distributed Hashtables

- Content-addressable network (CAN) [Ratnasamy et al. 2001] and Chord [Stoica et al. 2001] are also called scalable distributed hash tables (DHTs)
- There are other scalable DHT protocols.



Peer-to-peer Networks

- Peer-to-peer networks: distributed systems of **no hierarchy** - each node is a peer of any other in terms of equal functionality
 - DNS? Napster? Gnutella? Random walk? DHTs (CAN and Chord)?
- Fundamental advantage of p2p networks
 - better scalability?
 - no performance bottleneck
 - better robustness?
 - more tolerant to intentional attacks; better worst-case scenario under random failures



More on Scalability & Robustness

- **Scalability**: able to support large number of nodes
 - cost of each operation is below linear-scaling - goes up slowly when the system size increases: $O(n)$ is terrible, $O(n^{1/2})$ is OK, $O(\log n)$ is good.
 - space requirement at each node is below linear-scaling.
- **Robustness**: complexity of repair mechanism is a very important issue
 - Gnutella? Random walk? CAN? Chord?



Where are we now?

- Three approaches for searching large, distributed systems:
 - central index
 - query flooding
 - random walk
 - hash + heuristics-based routing (DHT)
- Concept of peer-to-peer networks.
- DHT can serve as a fundamental building block for other network services
 - p2p distributed file systems
 - DNS
 - Keyword search in peer-to-peer fashion.

Keyword Search

- user inputs a few keywords, the system returns a list of documents matching the keywords - **Google**
- Google maintains a central search index:
 - a search index contains a list of all searchable words, each of which contains a list of documents relevant to the word
 - intersection of document lists for multiple-word queries

Java:

Page #123	Page #157
-----------	-----------	--------

Sun:

Page #157	Page #468
-----------	-----------	--------

... ..

Peer-to-peer Keyword Search

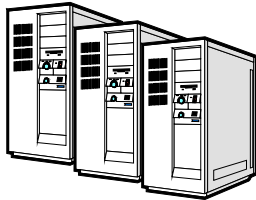
Solution 1: Split based on keywords

Split the index database to many pieces based on **keywords** and distribute them to many nodes in the network.

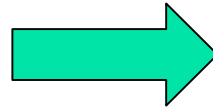
Java

Sun

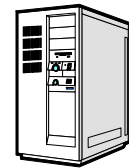
... ..



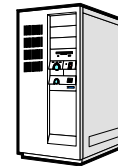
Central index - Google



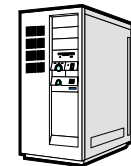
Sun



Java



... ..

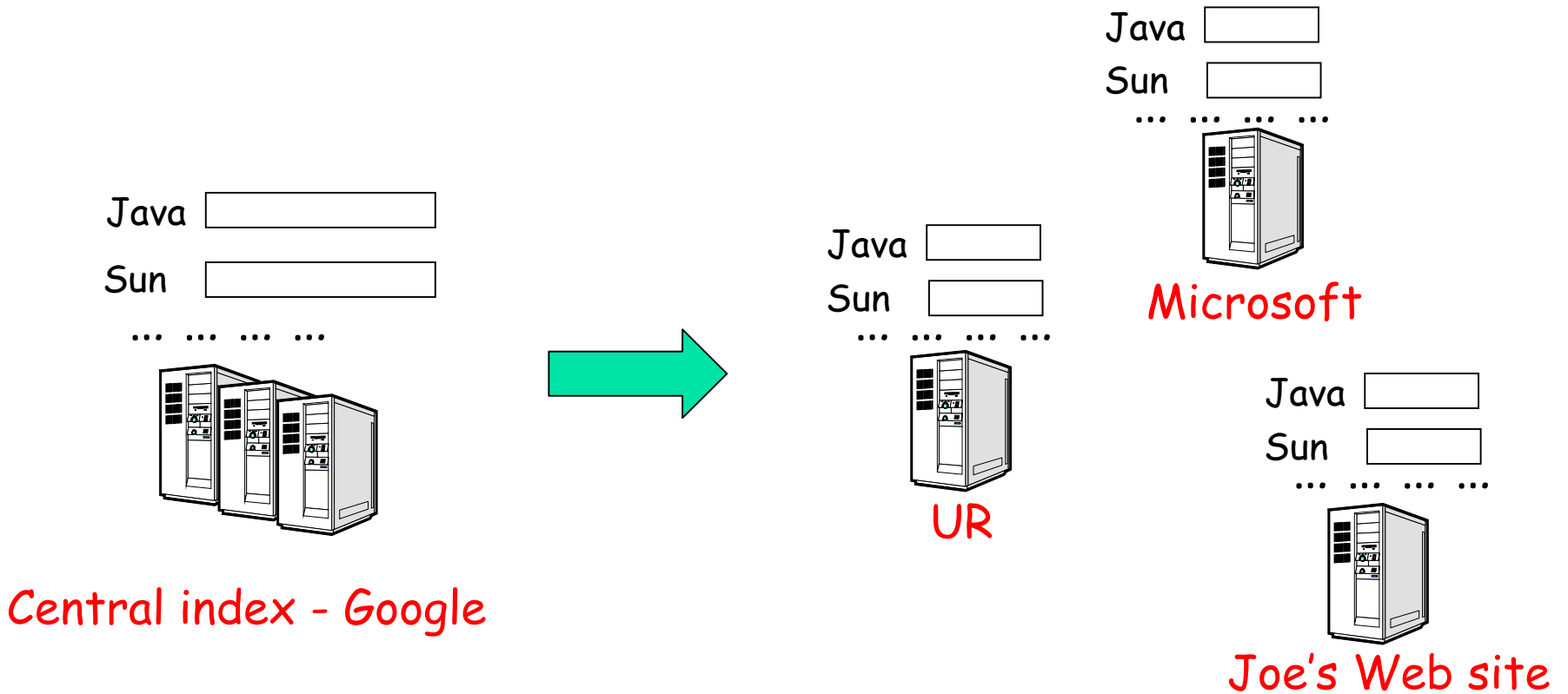


Distributed index

Peer-to-peer Keyword Search

Solution 2: Split based on documents

Split the index database to many pieces based on **documents** and distribute them to many nodes.





Putting Them Together

- Split based on keywords
 - Weakness: transferring large data segments for multiple keyword queries.
- Split based on documents
 - Weakness: too many sites to visit for each query.