PEER-TO-PEER SYSTEMS

• Peers are identical

• Peers are both senders and recipients

• Communication is indirect: space uncoupled

Peer-to-peer middleware

**Functional requirements:**

• enable clients to locate and communicate with any individual resource made available to a service

• ability to add new resources (ref to the data(source)) and to remove them

**Non-functional requirements:**

• Globally scalable

• Load balancing (that we share the load among all peers)

• Optimization for local interactions between neighbors:

place resources close to the nodes that access them the most

Routing overlay: distributed algorithm responsible for locating nodes and objects

It functions

- Routes clients requests to the desired resource(the ref)

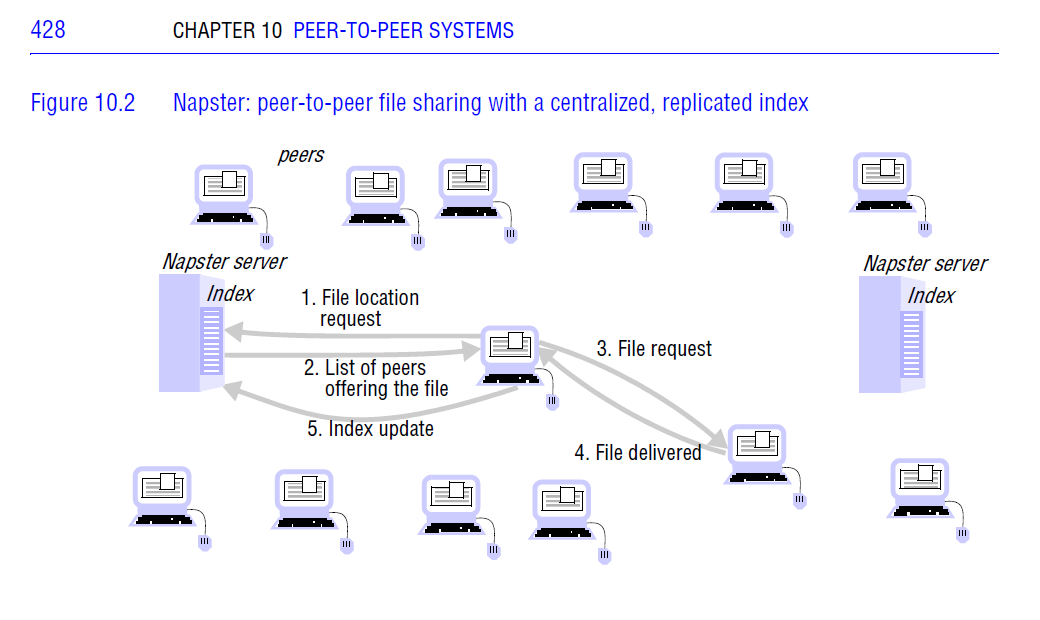
- Handle new resource(ref) insertion/deletion

- Handle node addition / removal

• Objects(resources) are identified by GUID: unique (global unique identifier)

• Tracked in DHT: distributed hash table (the hash contains the resource ref and its location)

• GUID is not human readable (hash value)



**Algorithm: Pastry**

• A peer has a table of some peers

• **Algorithm** Needs GUID (other algorithms only need a reference)

• Treats GUID table as a sorted circle

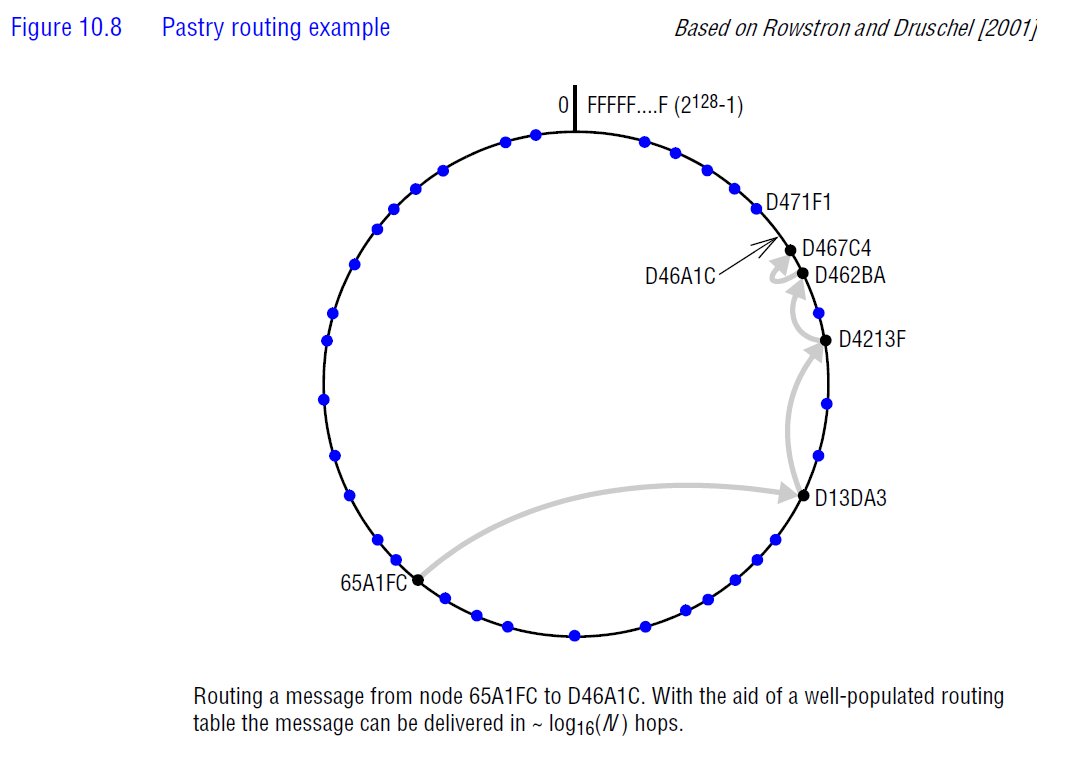
• Nodes send heartbeat messages to neighboring nodes in their leaf sets, to signal they are

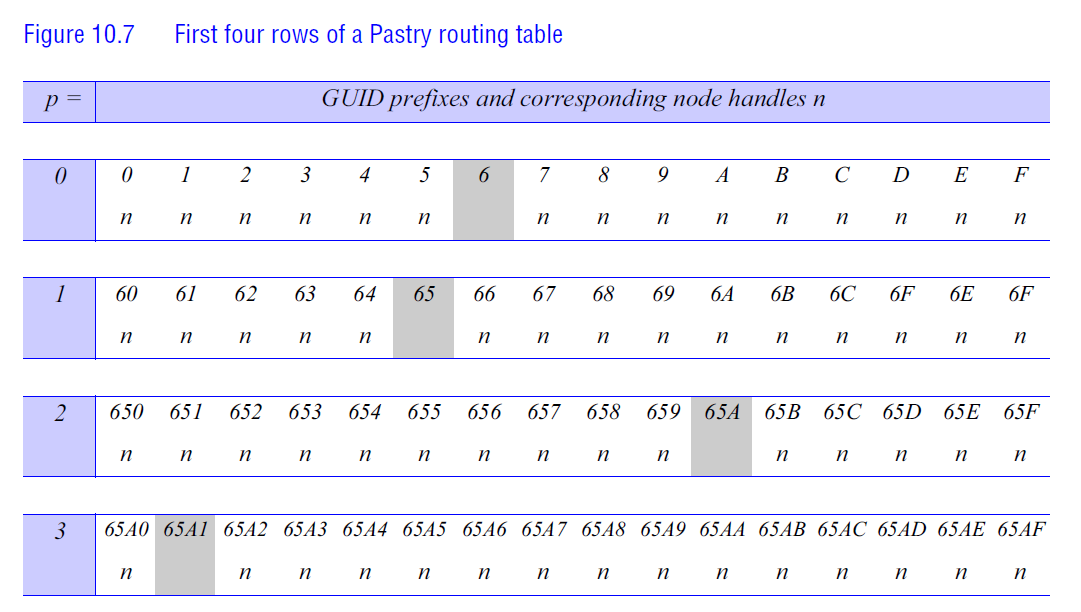
Alive.

Lead set (is a group of nodes (how many))

• Nodes Know at least 1 peer (preferably more)

• Addressee: GUID (globally unique identifier); hexadecimal





**Strategies for effective search**

Binary search

• Starts in the middle of a sorted list, throws away irrelevant half, repeats for the remainder

**Algorithm: Flooding**

• The peer sends the message (the request) to all the peers it knows, and so on

• Breaks down in larger networks

**Algorithm: Random walk**

• Peer sends the message to a fixed number of randomly selected peers

**Algorithm: Gossip**

• Peer sends to a fixed percentage of peers

**Algorithm: Ultrapeers**

• Skype

