Distributed Systems

Peer-to-Peer

Rik Sarkar
James Cheney

University of Edinburgh
Spring 2014

Recap: p2p

- We studied properties of p2p systems
- Examples of p2p system
- Arpanet – Internet
- SETI@home
- Napster
- Gnutella
- Bittorrent

Skype

- Communication software
- Central server to find IP address or for initial contact to user
- After that, communication occurs directly, server does not see messages
- Means receiver does not get messages until both sender and receiver are online and aware of each-other
- Uses Voice over IP (VoIP) for audio

Skype

- Allows phone calls with credit
  - Skype has an office phone line in country X
- When user calls a number in country X
  - The call goes to skype office in X through Internet (free of cost)
  - Then it is routed to the regular phone (cost of a local call)
  - To skype, it costs like a local call
  - User charged a bit more for profit
  - Still cheaper than International call

What is P2P good for?

- In principle, can be used for all sorts of sharing
- Possible to rebuild entire Internet as p2p
  - Everyone participates
  - Any resources can be anywhere, found and delivered through p2p
  - Not very practical, hard to do efficiently
- Problem: peers are too dynamic, unreliable
- Adapting to that, makes the system inefficient
  - Think of Gnutella search
- Still some interesting questions remain
  - Can we use it to distribute data better? i.e. What if users stored data in general, and not only what they downloaded
    - Issues of privacy, reliability etc
  - Can we use it to distribute computation in general?

Some criteria for using p2p design

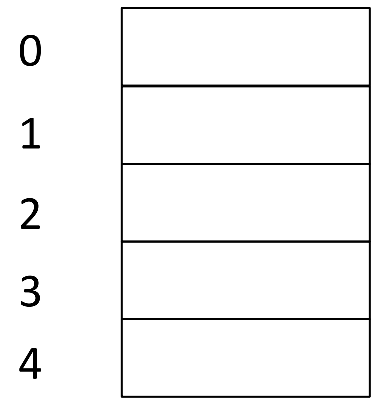
- Budget – p2p is low budget solution to distribute data/computation
- Resource relevance/popularity – if the items are popular, p2p is useful. Otherwise the few users may go offline..
- Trust – if other users can be trusted, p2p can be a good solution.
  - Can we build a secure network that operates without this assumption?
- Rate of system change – if the system is too dynamic, p2p may not be good. (Imagine peers joining/leaving too fast)
- Rate of content change – p2p is good for static/fixed content. Not good for contents that change regularly, since then all copies have to be updated.
- Criticality – p2p is unreliable, since peers cats independently, may leave/ fail any time.
  - P2P is good for applications that are good to have but are not critical to anything urgent

Better p2p design: Some theory

- File transfer in p2p is scalable (efficient even in large systems with many nodes)
  – Occurs directly between peers using Internet
  – Bittorrent like systems can download from multiple peers – more efficiency
- The problem in p2p:
  – Search is inefficient in large systems

Hash tables

- A hash tables has b buckets
  - Any item x is put into bucket h(x)
  - h(x) must be at most b for all x



- Example: a hash table of 5 buckets
  - Any item x is put into bucket x mod 5
  - Insert numbers 3, 5, 12, 116, 211

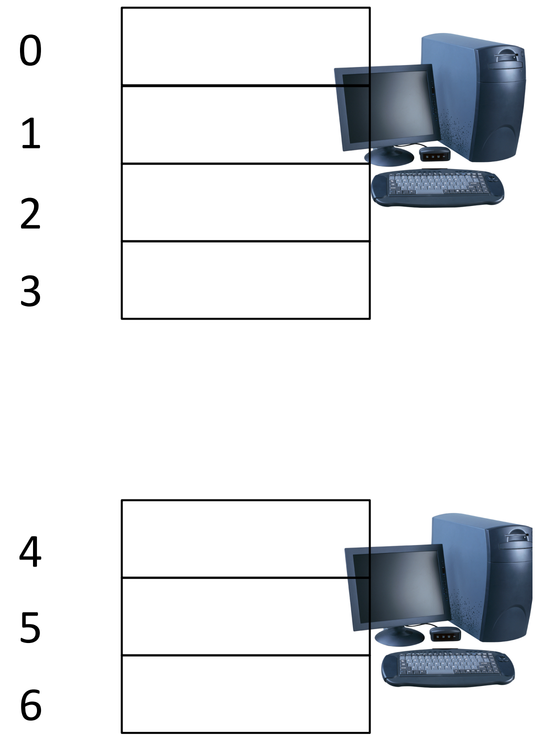
Hash tables

- Hash tables are used to find elements quickly
- Suppose we use hash on the file name “fname”
- Then h(“fname”) takes us to the bucket containing file fname
- If the bucket has many files, then we will still have to search for the file inside the bucket
- But if our hash table is reasonably large, then usually there will be only a few files in the bucket – easy to search

0	5
1	116, 211
2	2
3	3
4	

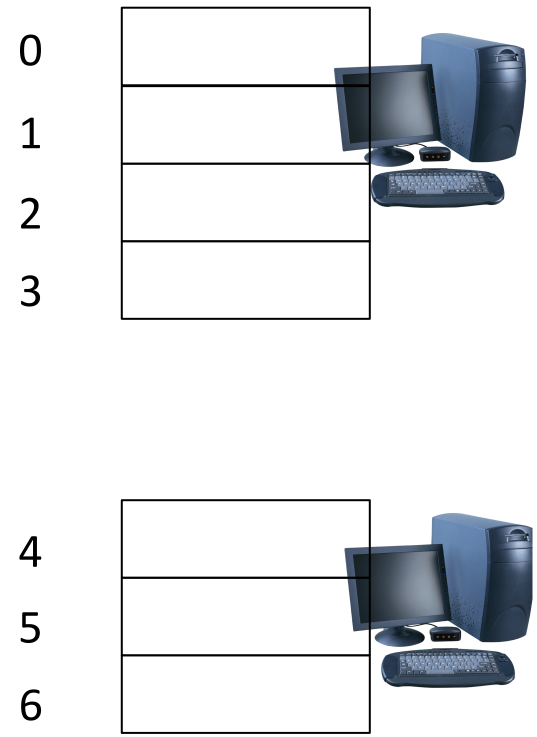
Distributed hash tables

- Each computer knows the hash function
- Each computer is responsible for some of the hash buckets
- Different parts of the data are stored in different computers



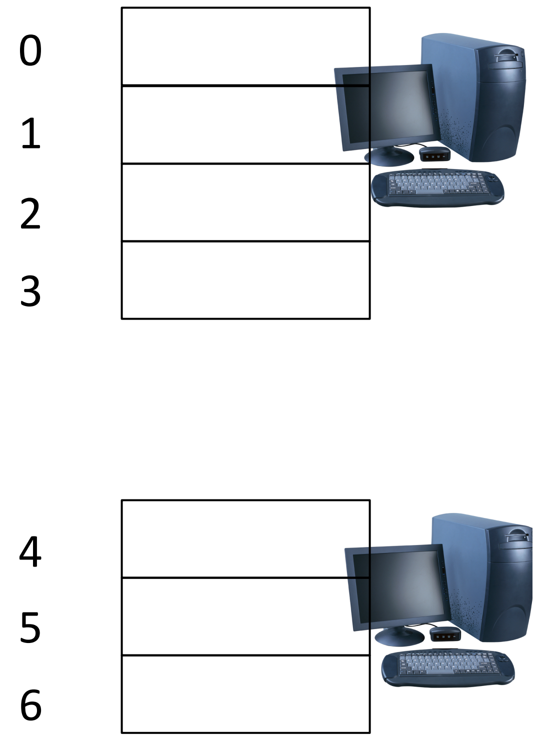
Distributed hash tables

- Elements can be inserted/ retrieved as usual to the corresponding bucket
  - But need to ask the computer responsible for that bucket
- Need efficient mechanism to find the responsible node
  - Using communication between nodes



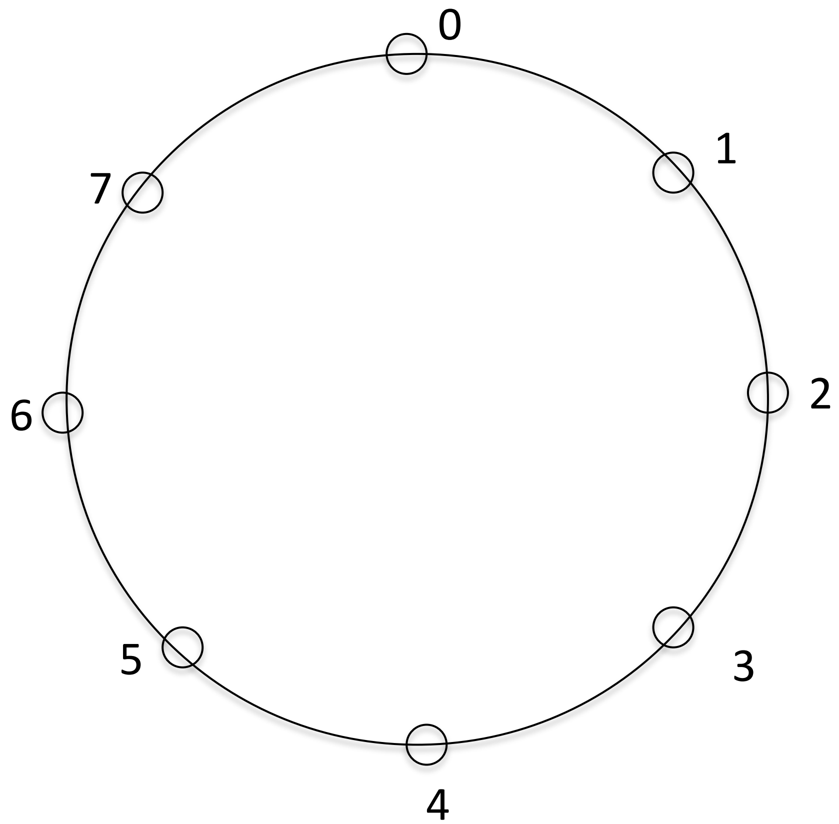
Distributed hash tables

- P2p systems are dynamic
  - Nodes join/leave all the time
  - Need a mechanism to shift responsibilities with change



Example system: Chord

- P2P system from MIT (2001)
- Operates using a ring overlay for the set of node ids
- Each id has a *slot* in the overlay
  - Each slot may not be occupied

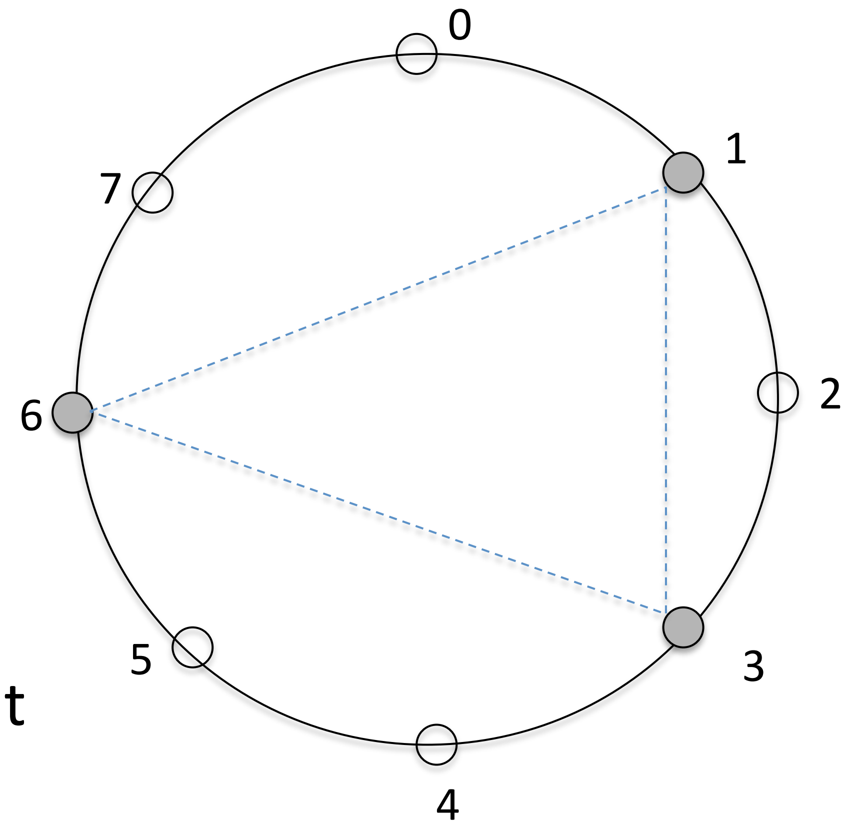


Example system: Chord

- Each node knows the *next* and *previous* occupied slots in the ring

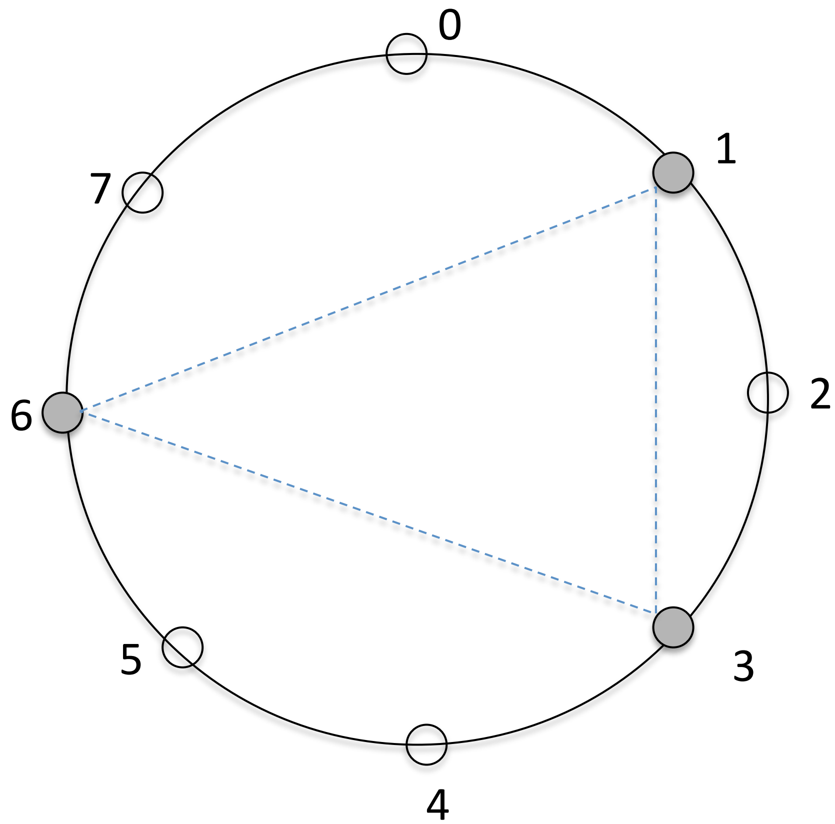
- Storage using hash tables

- To store/retrieve data, forward message to *next* until reaching the node with the bucket
- If the slot is not occupied, (for example, 5 in the figure), store it at the next occupied slot (eg. 6)



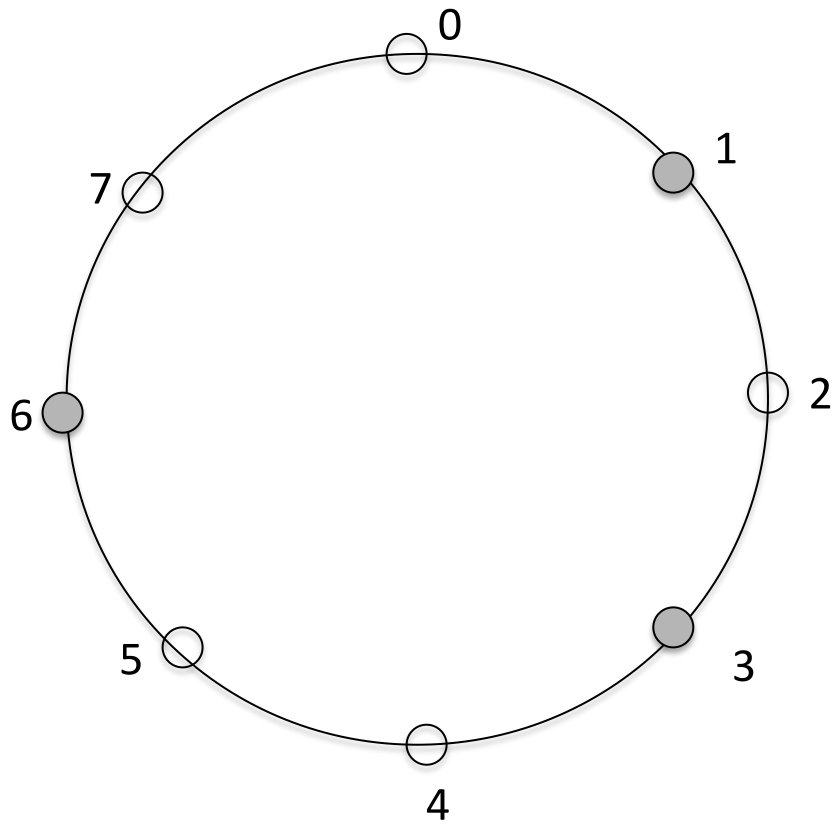
Example system: Chord

- When a node wants to join, it finds occupied slots just before/after itself
- Example: 5 wants to join
  - 5 has to know at least one node already in system, say node 1.
  - 5 sends search message to 1
  - The message gets forwarded using *next* pointers
  - Node 3 and 6 realize that they are neighbors of 5
  - Message sent back to 5



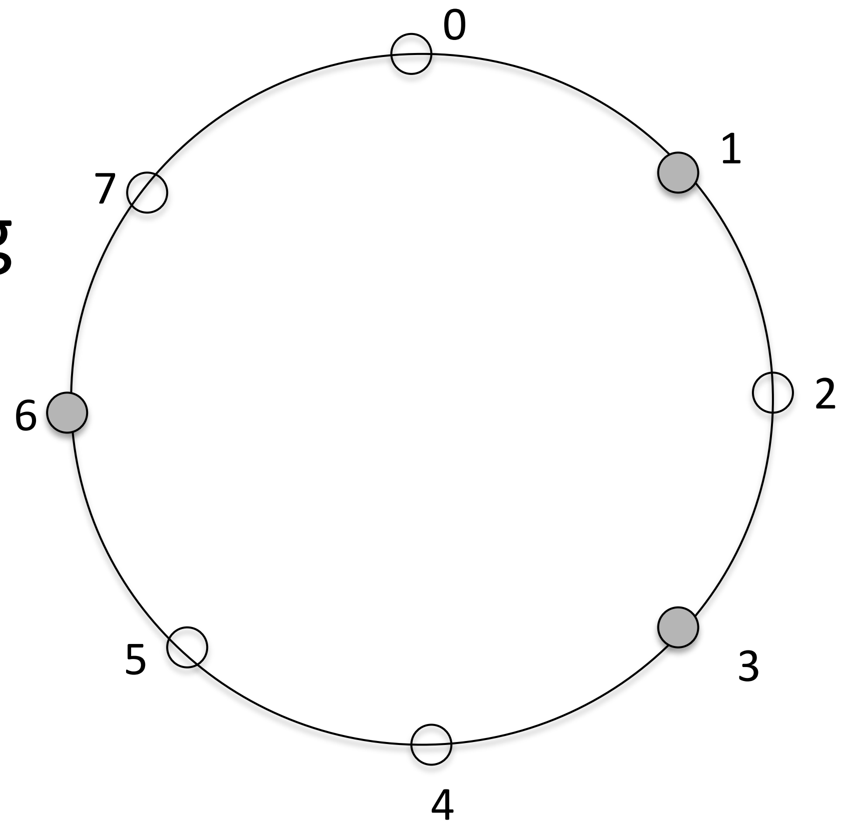
Example system: Chord

- 6 can send 5's hash table to 5
- Each node replicates all the data for several nodes before/after itself
- If a node fails, its data is still preserved



Example system: Chord

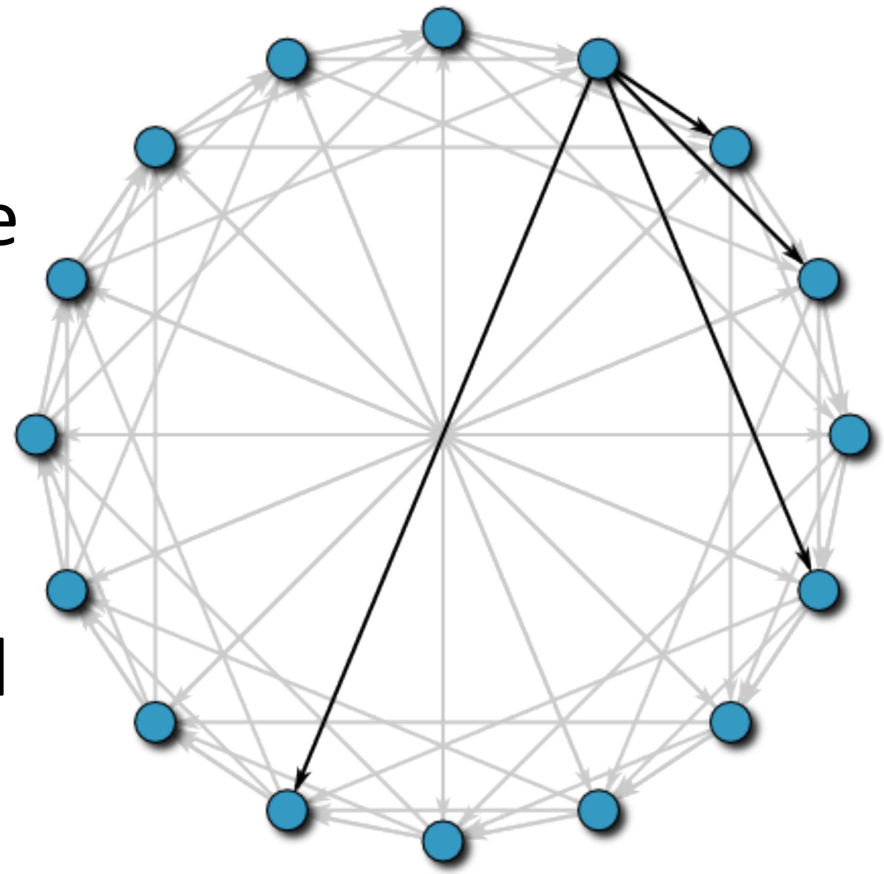
- Problem: search is still inefficient
- It goes sequentially along the ring
- Cost: O(n)
- Now imagine a ring with a million nodes!



Chord: more efficient search

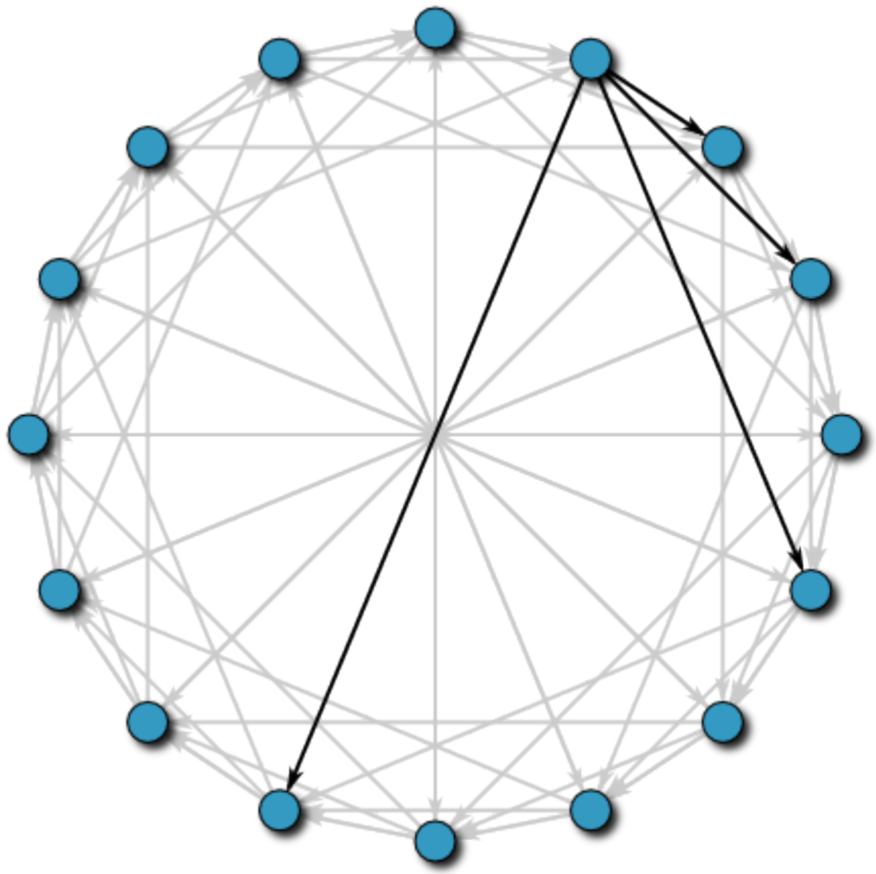
- Add some extra links in the overlay graph
- To find node x, go to the neighbor that is nearest to the destination

- Which extra links to add to the network?



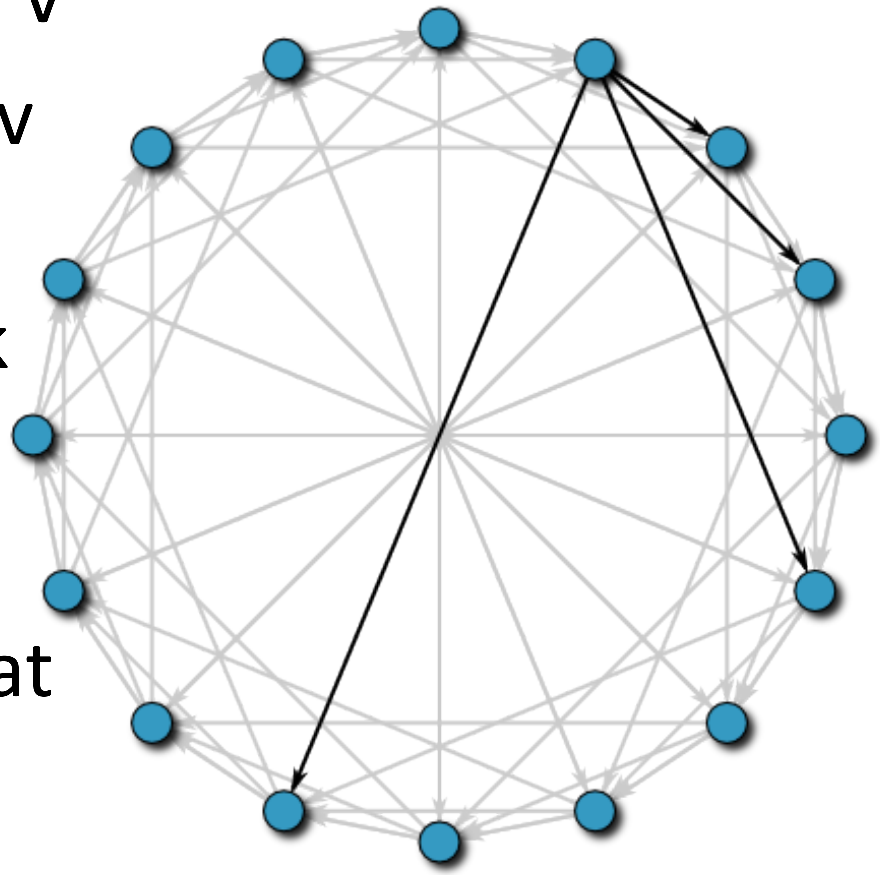
Chord: more efficient search

- At node v, add links to
  - $(2^i+v) \bmod n$
  - Or the first occupied slot after
- Each node has log n additional links
  - O(log n) storage
- Search is efficient



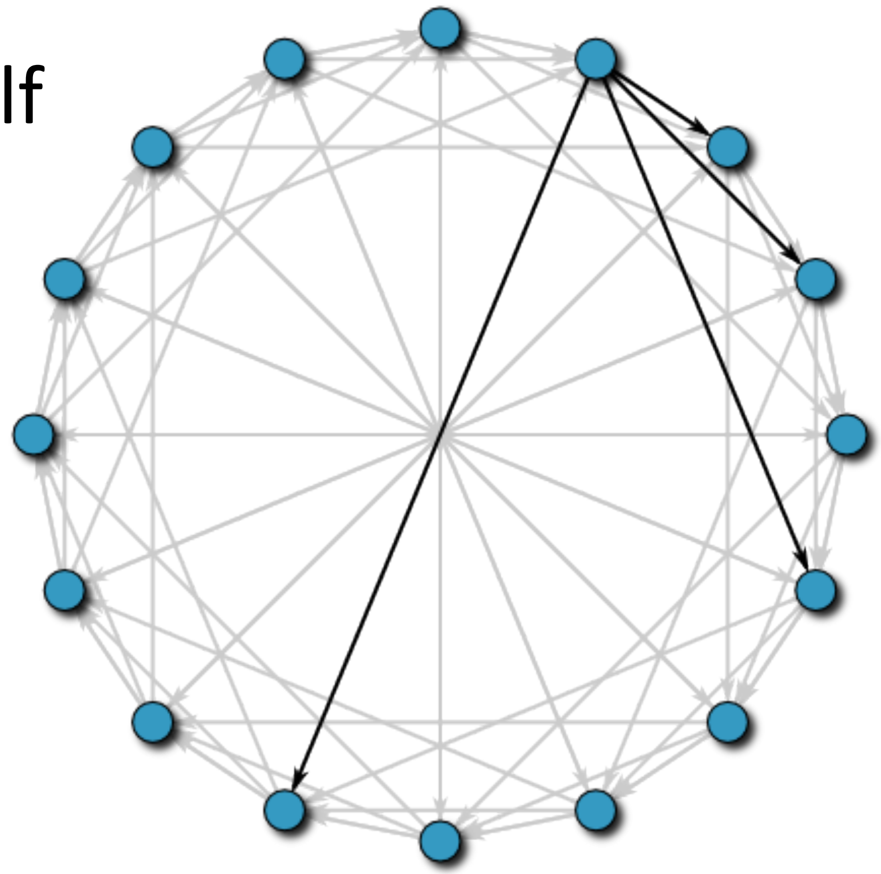
Chord: more efficient search

- Suppose we are at node v
- And searching for node v + x
- There is at least one link to a node between v + x/2 and v+x
- The message goes to that node



Chord: more efficient search

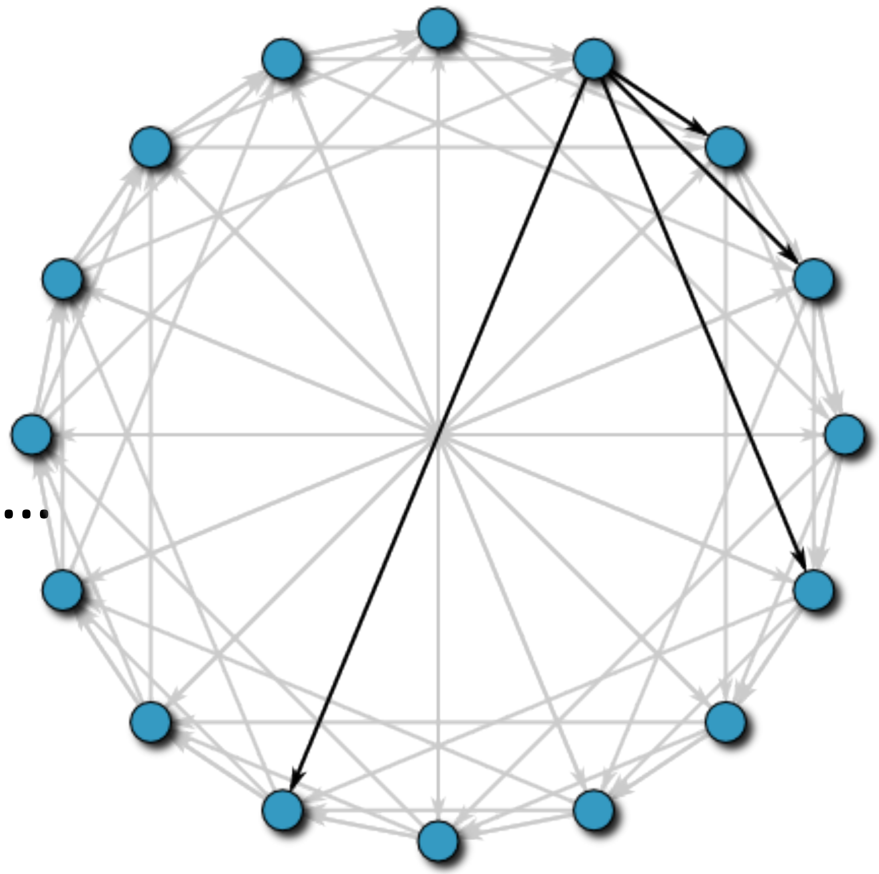
- The distance to the destination becomes half in each step
- How many steps does it take?



Chord: more efficient search

- The distance d to the destination becomes half or less in each step
- How many steps does it take?

- The sequence d, d/2, d/4 … converges to 1
- In O(lg n) steps
  - (since d<=n)

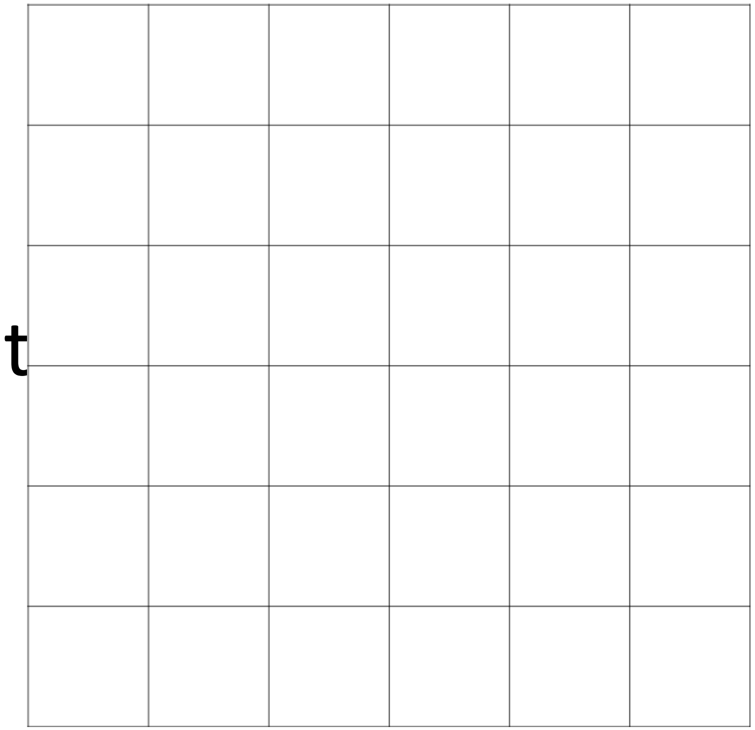


Magnet links

- Instead of a .torrent or other descriptor file, use a "link" which eventually gets the file or equivalent data
  – Can be used in any system, currently popular in bittorrent
- Can be of different types
  – Some links direct to the "trackers", and give the hash of the file
  – Other links lead into a DHT, to find .torrent file/info
    - Assumes the user agent knows how to enter and find content in the overlay network of the DHT
    - Several slightly different formats for magnet links
- Overall, bittorrent is moving toward using DHT magnet links
- But the formats/protocols are not yet standardized or well documented

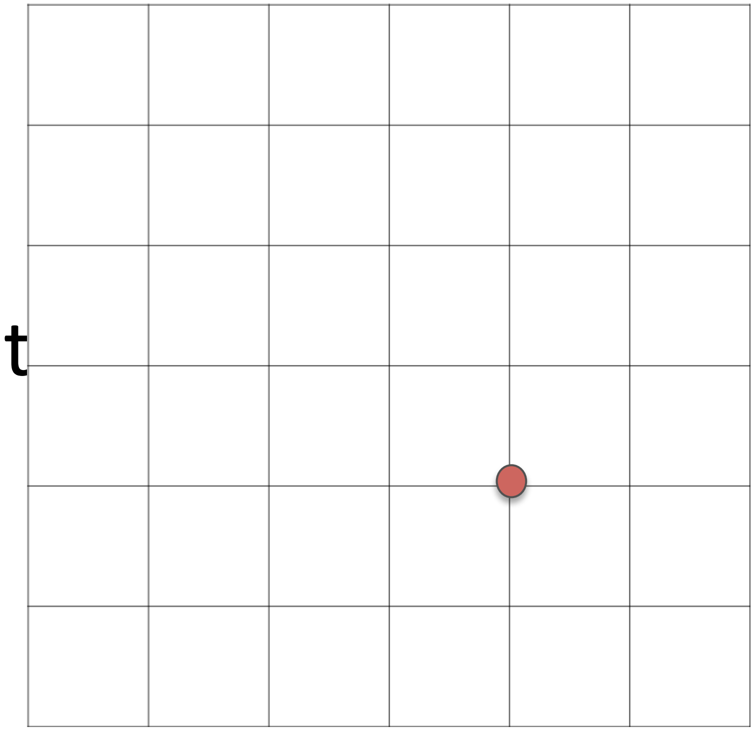
Few other methods of doing DHT/search

- The overlay network is a grid
- Each node knows its neighbors in the grid
- The DHT hash stores item at some node in the grid



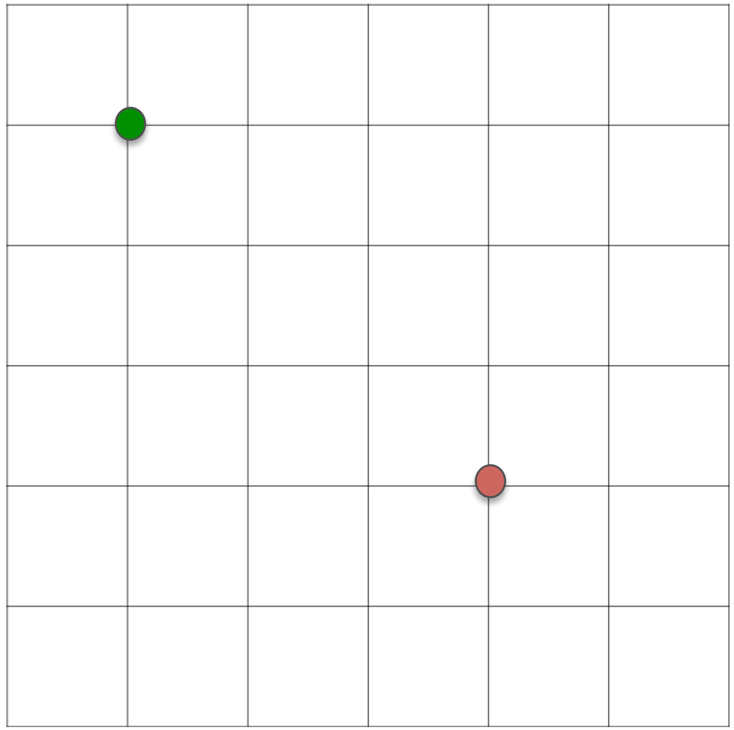
Few other methods of doing DHT/search

- The overlay network is a grid
- Each node knows its neighbors in the grid
- The DHT hash stores item at some node in the grid



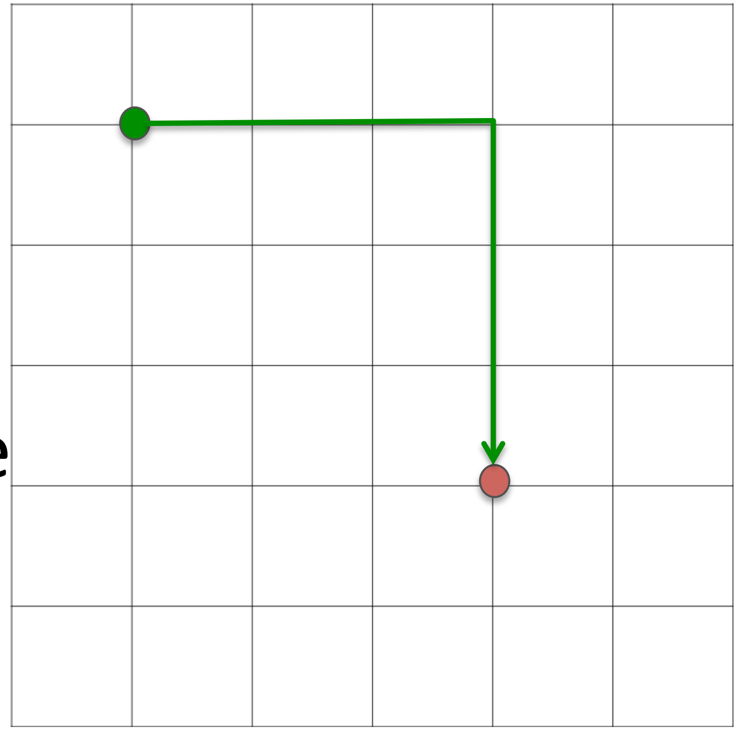
Few other methods of doing DHT/search

- To access content, just need to route to the node using the grid
- Routing is easy!



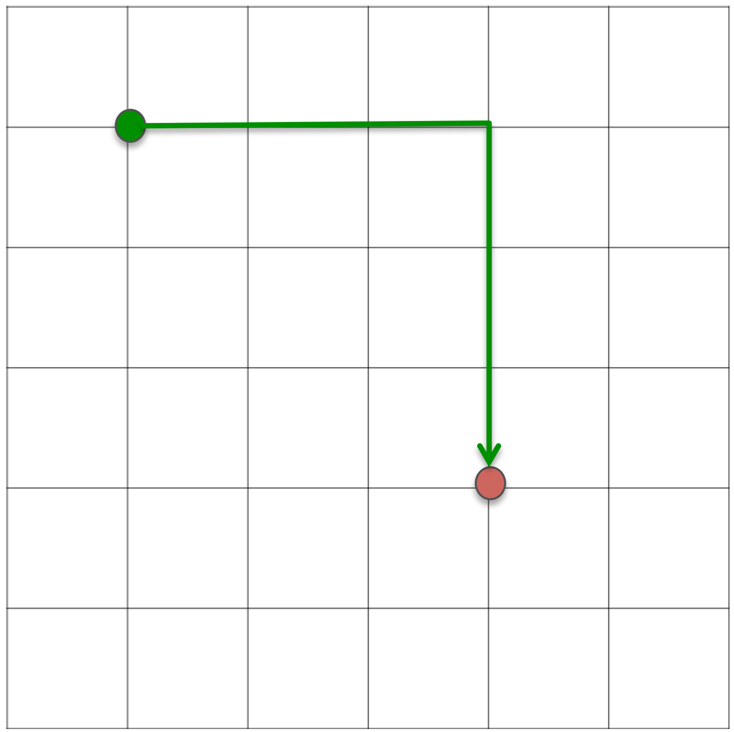
Few other methods of doing DHT/search

- To access content, just need to route to the node using the grid
- Routing is easy!
- Get to the x coordinate, then get to the y coordinate
- What is the cost of the search?



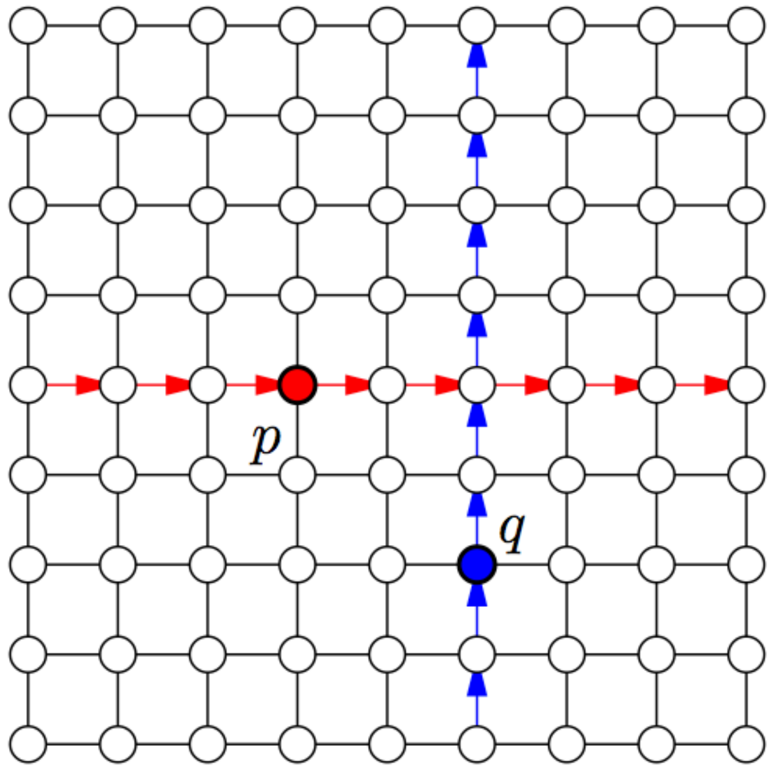
Few other methods of doing DHT/search

- A grid with n nodes is
  - √n x √n in size
  - The expected and maximum distance between a pair of nodes id O(√n)
  - The expected cost is O(√n)



Few other methods of doing DHT/search

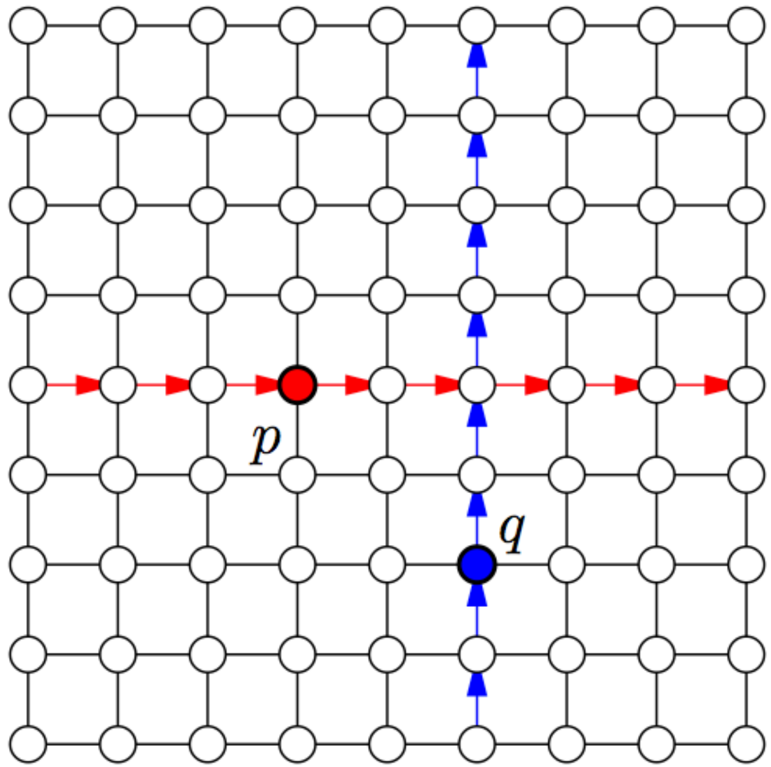
- Double rulings
- Suppose node q has content to share
- q stores it (or may be the .torrent file) at all nodes in the same column
- Node p searches for the content
- Along all nodes in its row
- Guaranteed to find content!



Double rulings

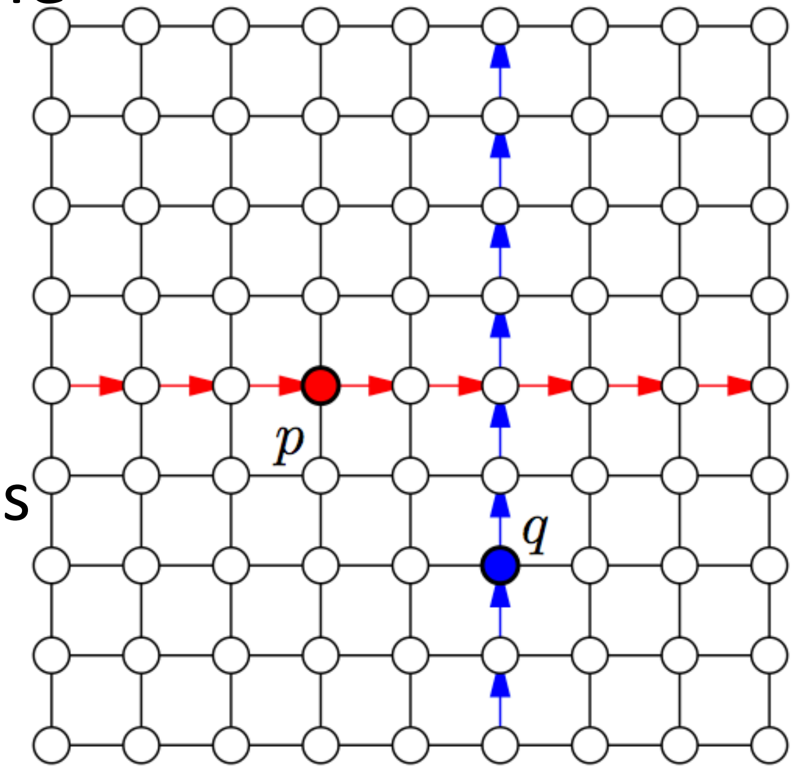
- Storage cost:
  - O(√n) per item
- Search cost:
  - O(√n) per search

- Does not need DHT!



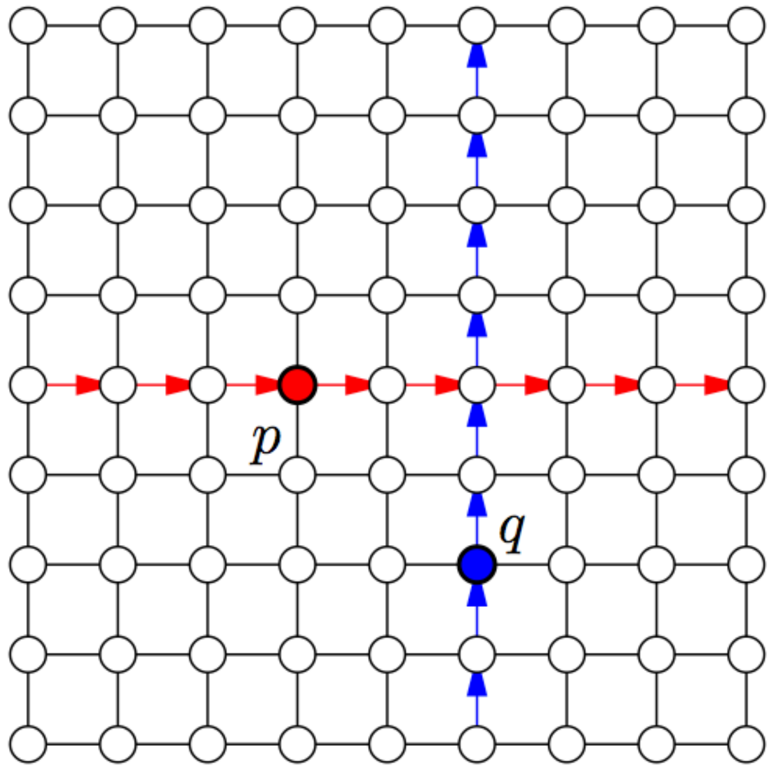
Double rulings

- A different way to doing the search:
  - (remember the efficient leader election)
  - p searches in phases
  - In phase i, p checks 2^i nodes to the right and left (using TTL messages)



Double rulings

- In phase i, p checks 2^i nodes to the right and left (using TTL messages)
  - Until it hits q's row
- If the distance between p and q is d
- Then the message hitting the content could have traveled at most distance d
- In previous phases, messages could have traveled at most
  - d + d/2 + d/4 +….



Double rulings

- The cost of the search is O(d)
- Where the distance between p and q is d
- This is called *distance sensitive search*
- Even better if the grid approximates the underlying network distances, then the cost is proportional to the actual distance between p and q
- Imagine the map of the city laid on the grid. Then "distance" is actual distance.