CS 425/ECE 428/CSE 424
Distributed Systems
(Fall 2009)

Lecture 13

Peer-to-Peer systems (Part II)

Acknowledgement

- The slides during this semester are based on ideas and material from the following sources:
  - Slides prepared by Professors M. Harandi, J. Hou, I. Gupta, N. Vaidya, Y-Ch. Hu, S. Mitra.
  - Slides from Professor S. Gosh’s course at University o Iowa.

Administrative

- **HW 2**
  - Deadline, October 6 (Tuesday), 2pm
  - Solutions will be released in the evening
- **Midterm on October 13 (Tuesday)**
  - Exam will take place in class
  - You are allowed one cheat-sheet (one side only)
  - Exam will include all topics covered in HW1-HW2, plus P2P material (Lectures 1-13)
  - No TA office hours on **October 12**, instead the **instructor will hold office hours on Monday, October 12, 3-4pm in 3104 SC**
- **MP1 code** is released on the class website
  - Score of MP1 will be posted on **October 6, 2009 evening**

Administrative

- **MP2** posted **October 5, 2009**, on the course website,
  - **Deadline November 6 (Friday)**
  - **Demonstrations** , 4-6pm, 11/6/2009
  - You will need to **lease** one Android/Google Developers Phone per person from the CS department (see lease instructions)!!
  - **Start early** on this MP2
  - **Update groups** as soon as possible and let TA know by email so that she can work with TSG to update group svn
  - **Tutorial for MP2** planned for **October 28** evening if students send questions to TA by **October 25**. Send requests what you would like to hear in the tutorial.
  - During October 15-25, Thadpong Pongthawornkamol (tpongth2@illinois.edu) will held office hours and respond to MP2 questions for Ying Huang (Ying is going to the IEEE MASS 2009 conference in China)

Administrative

- **MP3 proposal instructions**
  - You will need to submit a proposal for MP3 on top of your MP2 before you start MP3 on November 9, 2009
  - Exact Instructions for MP3 proposal format will be posted **October 8, 2009**
  - Deadline for MP3 proposal: **October 25, 2009, email proposal to TA**
  - At least one representative of each group meets with **instructor or TA during October 26-28** during their office hours) watch for extended office hours during these days.
    - **Instructor office hours: October 28 times 8:30-10am**

Administrative

- To get Google Developers Phone, you need a **Lease Form**
  - You must pick up a **lease form** from the instructor during **October 6-9** (either in the class or during office hours) since the lease form must be signed by the instructor.
  - Fill out the lease form; bring the lease form to **Rick van Hook/Paula Welch** and pick up the phone from **1330 SC**
- **Lease Phones**: phones will be ready to pick up starting **October 20, 9-4pm** from room 1330 SC (purchasing , receiving and inventory control office)
- **Return Phones**: phones need to be returned during **December 14-18**, **9-4pm** in 1330 SC

Plan for Today

- Fast-Track
- Summary of unstructured P2P networks
- Introduction of distributed hash table concept

Gnutella Summary

- No index servers
- Peers/servents maintain "neighbors" (membership list), this forms an overlay graph
- Peers store their own files
- Queries flooded out, ttl restricted
- Query Replies reverse path routed
- Supports file transfer through firewalls (one-way)
- Periodic Ping-Pong to keep neighbor lists fresh in spite of peers joining, leaving and failing
  - List size specified by human user at peer : heterogeneity means some peers may have more neighbors
  - Gnutella found to follow power law distribution:

P(#neighboring links for a node = *L*) ~ L^{-k} (*k* constant)

Gnutella Problems

- Ping/Pong constitutes 50% traffic
  - Solution: Multiplex, *cache* and reduce frequency of pings/pongs
- Repeated searches with same keywords
  - Solution: *Cache* Query and QueryHit messages
- Modem-connected hosts do not have enough bandwidth for passing Gnutella traffic
  - Solution: use a central server to act as proxy for such peers
  - Another solution:
    - ➔FastTrack System (in a few slides)

Problems (contd.)

- Large number of *freeloaders*
  - 70% of users in 2000 were freeloaders
- Flooding causes excessive traffic
  - Is there some way of maintaining meta-information about peers that leads to more intelligent routing?

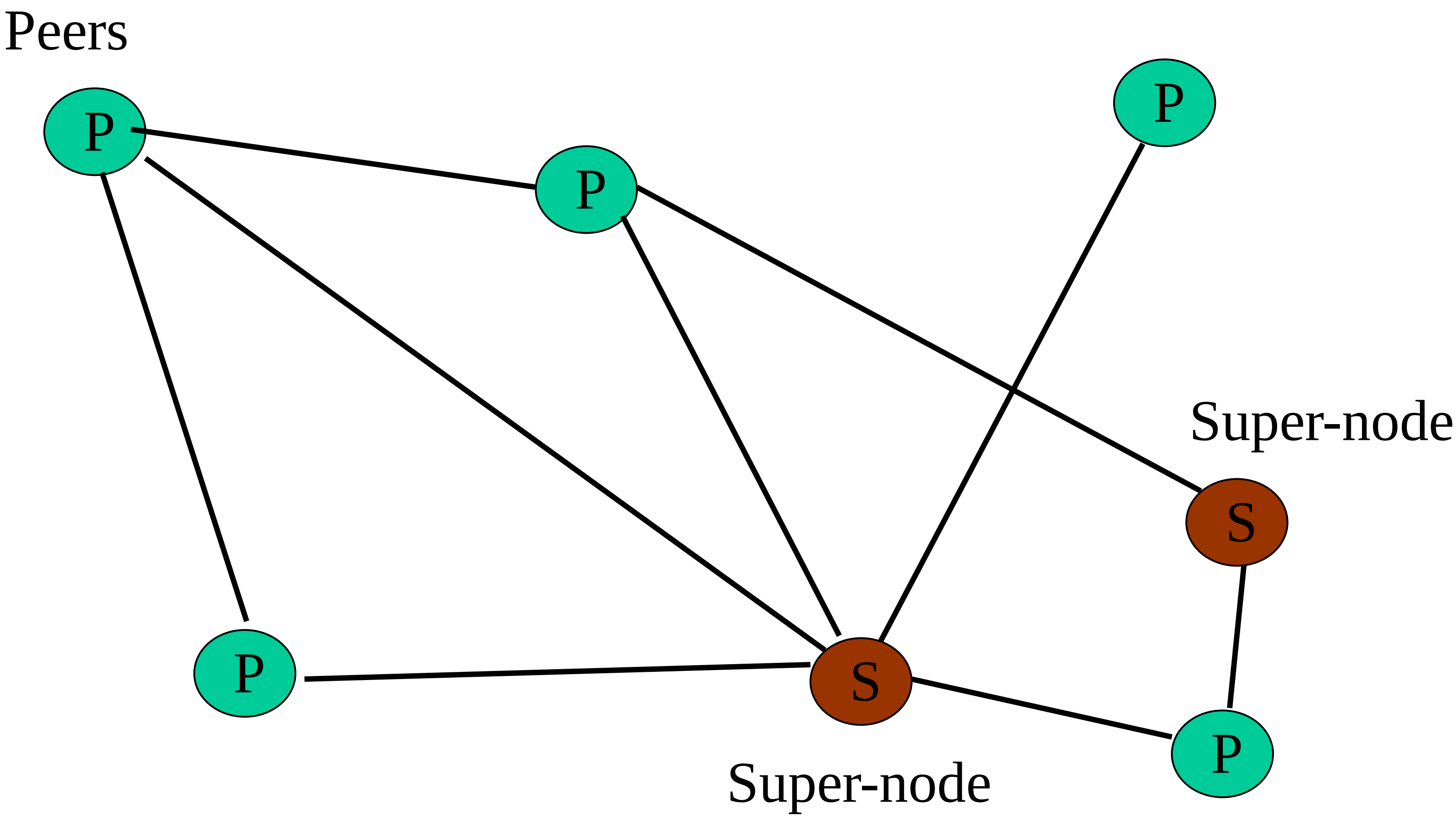
➔ Structured Peer-to-peer systems

e.g., Chord System (next lecture)

FastTrack (KaZaA)

- Unstructured Peer-to-Peer System
- Hybrid between Gnutella and Napster
- Takes advantage of "healthier" participants in the system (higher bandwidth nodes, nodes that are around most of the time, nodes that are not freeloaders, etc.)
- Underlying technology in Kazaa, KazaaLite, Grokster
- Proprietary protocol, but some details available
- Like Gnutella, but with some peers designated as ***supernodes***

A FastTrack-like System



FastTrack (contd.)

- A supernode stores a directory listing **(<filename,peer pointer>),** similar to Napster servers
- Supernode membership changes over time
- Any peer can become (and stay) a supernode, provided it has earned enough *reputation*
  - Kazaalite: participation level of a user between 0 and 1000, initially 10, then affected by length of periods of connectivity and total number of uploads from that client
- A peer searches for a file by contacting a nearby supernode

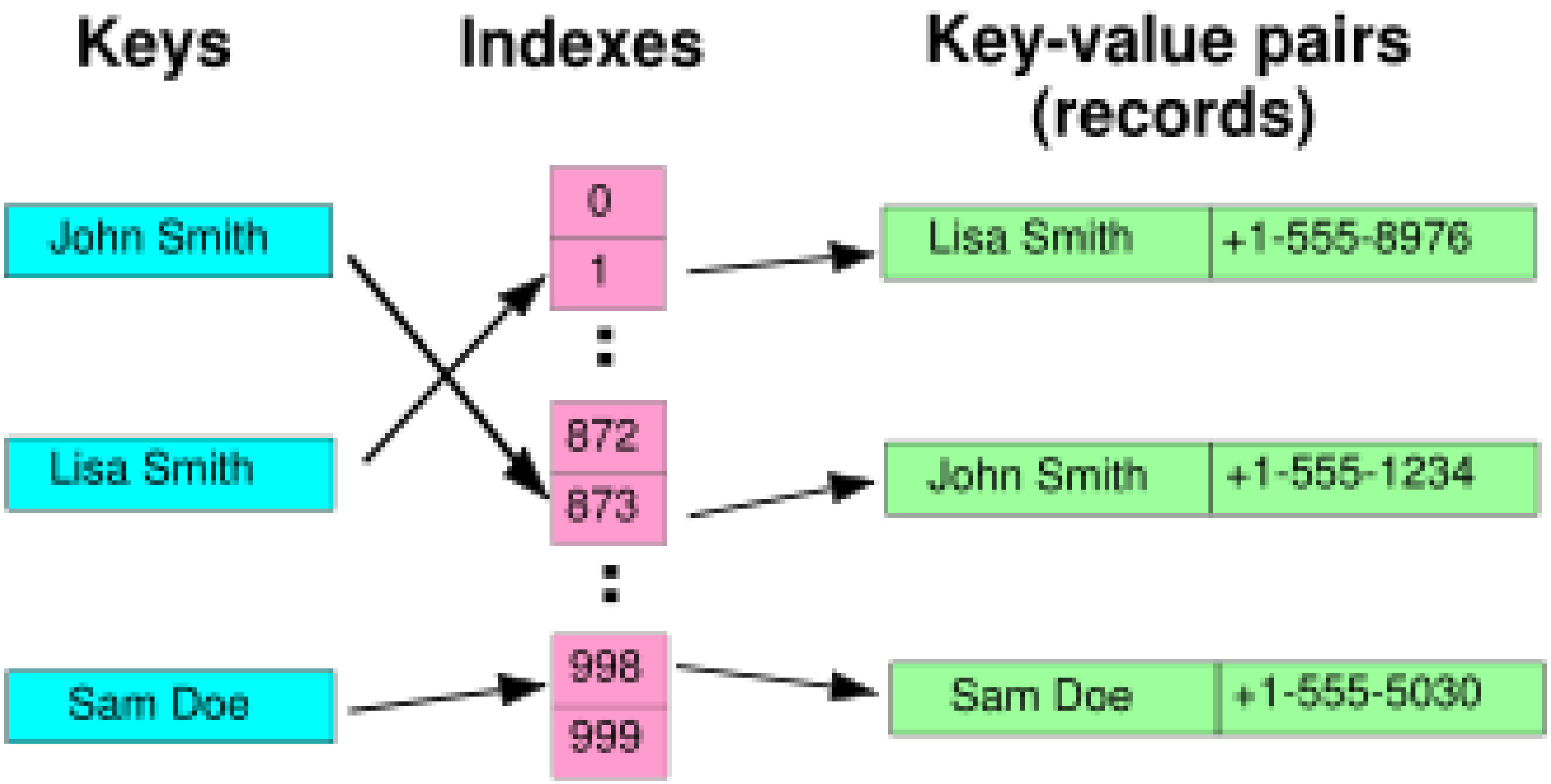
Final Comments on Unstructured P2P Systems

- How does a peer join the system (bootstrap)
  - Send an http request to well known URL `http://www.myp2pservice.com`
  - Message routed after DNS lookup to a well known server that has partial list of recently joined peers, and uses this to initialize new peers' neighbor table
- Lookups can be speeded up by having each peer cache:
  - Queries and their results that it sees
  - All directory entries (filename, host) mappings that it sees
  - The files themselves

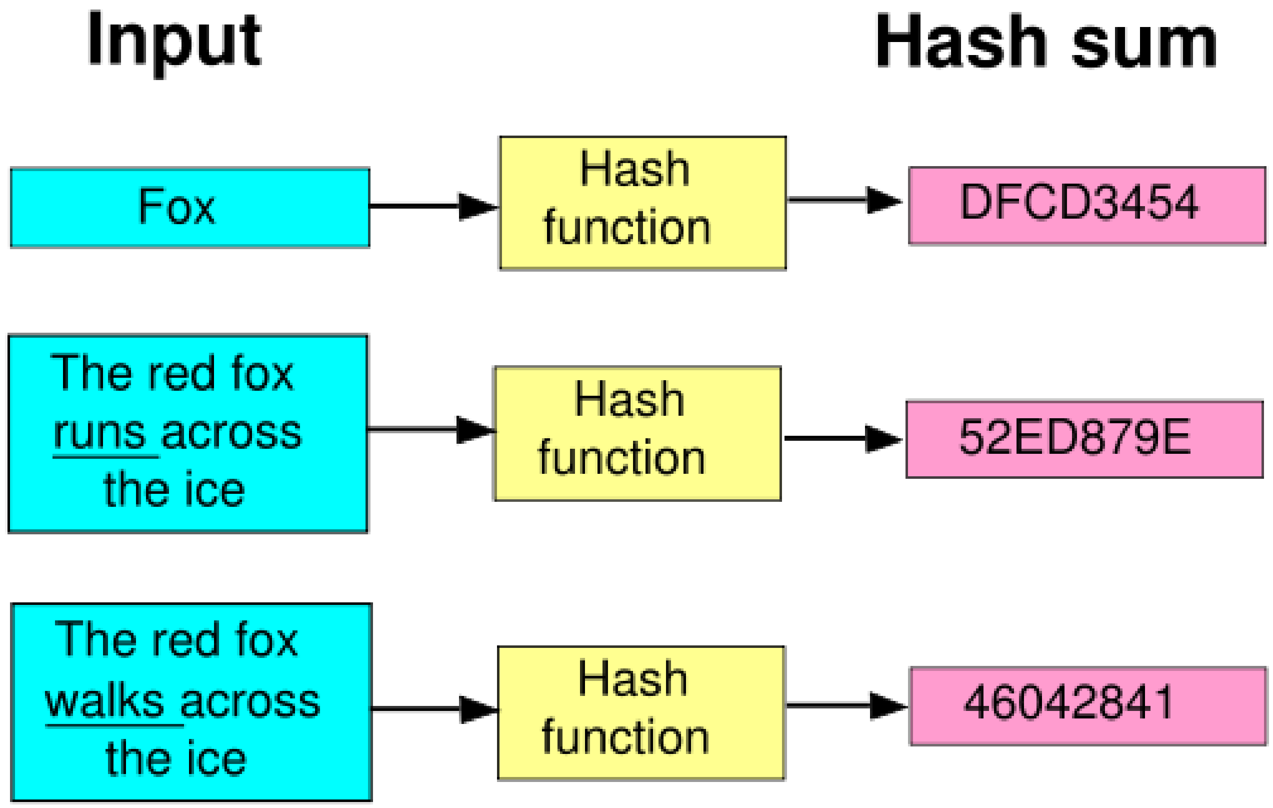
Comparative Performance

	Memory	Lookup Latency	#Messages for a lookup	
Napster	$O(1)$@client, $O(N)$@server	$O(1)$	$O(1)$	
Gnutella	$O(N)$	$O(N)$	$O(N)$	

Hash Table (Phone Book)



Hash Functions



DHT=Distributed Hash Table

- Hash table allows you to insert, lookup and delete objects with keys
- A *distributed* hash table allows you to do the same in a distributed setting (objects=files)
- Performance Concerns:
  - Load balancing
  - Fault-tolerance
  - Efficiency of lookups and inserts
  - Locality

DHTs

- DHT research was motivated by Napster and Gnutella
- First four DHTs appeared in 2001
  - CAN
  - Chord
  - Pastry
  - Tapestry
  - BitTorrent
- Chord, a structured peer to peer system that we study next

Comparative Performance

	Memory	Lookup Latency	#Messages for a lookup	
Napster	*O(1)* (*O(N)*@server)	*O(1)*	*O(1)*	
Gnutella	*O(N)*	*O(N)*	*O(N)*	
Chord	*O(log(N))*	*O(log(N))*	*O(log(N))*	

Chord

- Developers: I. Stoica, D. Karger, F. Kaashoek, H. Balakrishnan, R. Morris, Berkeley and MIT
- Intelligent choice of neighbors to reduce latency and message cost of routing (lookups/inserts)

Base Chord Protocol

- Uses concepts such as
  - Consistent hashing
  - Scalable key lookup
  - Handling of node dynamics
  - Stabilization protocol

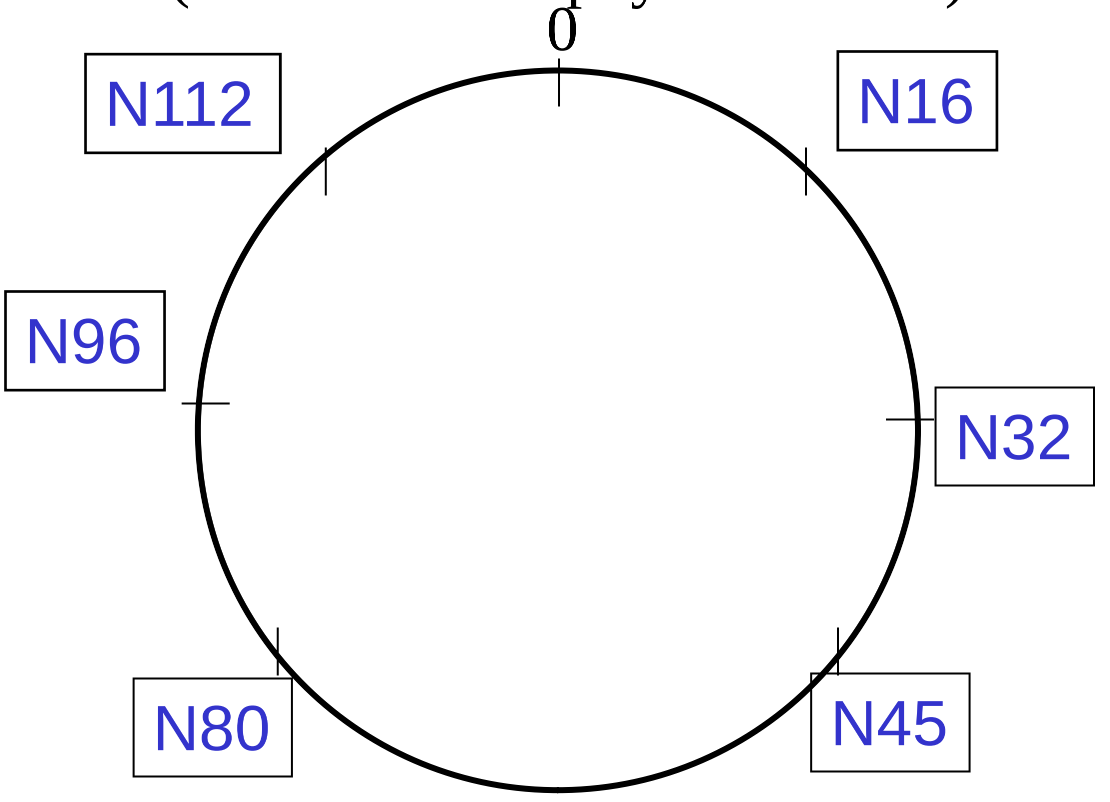
Consistent Hashing

- *Consistent Hashing* on peer's address
  - SHA-1 = Simple Hash Algorithm-1, a standard hashing function)
    - (In 2005 security flaws were identified in SHA-1)
  - **SHA-1(ip_address, port) →160 bit string**
    - Output truncated to *m* (< 160) bits
    - Called **peer *id*** (integer between 0 and $2^m - 1$)
    - Example: SHA-1(140.45.3.10, 1024) = 45 (0101101) with m = 7
    - Not unique but **peer id** conflicts very very unlikely
  - Any node A **can calculate the peer id** of any other node B, given B's IP address and port number
  - We can then map peers to one of $\mathbf{2^m}$ logical points on a circle

Ring of Peers

Example: *m=7 (number of logical nodes is 128)=* 2^7

N=6 peers/nodes (*N* - number of physical nodes)

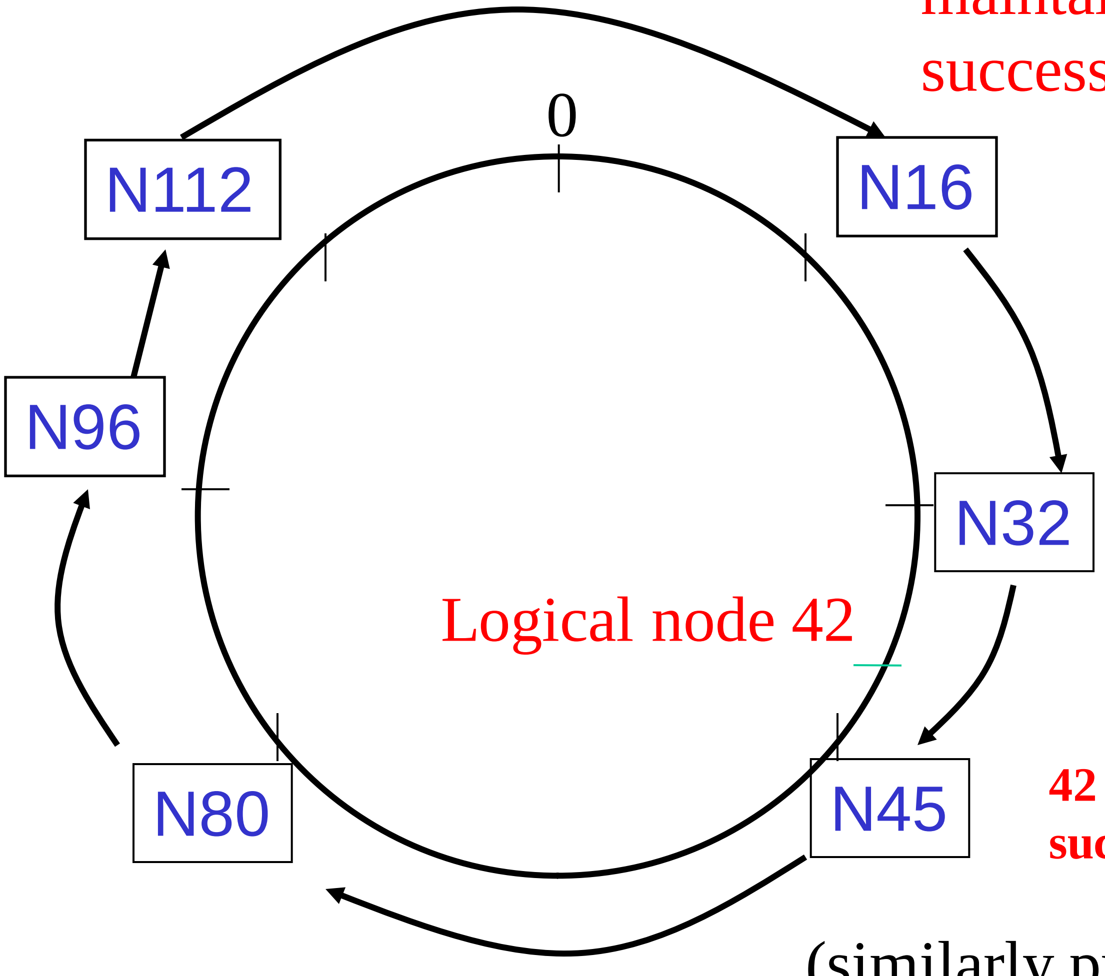


Peer pointers (1): *successors*

Say *m=7*

SHA-1(140.45.3.12, 1245) = 42 (0101010)

Each physical node maintains a successor pointer



42 stored in the successor N45

(similarly predecessors)

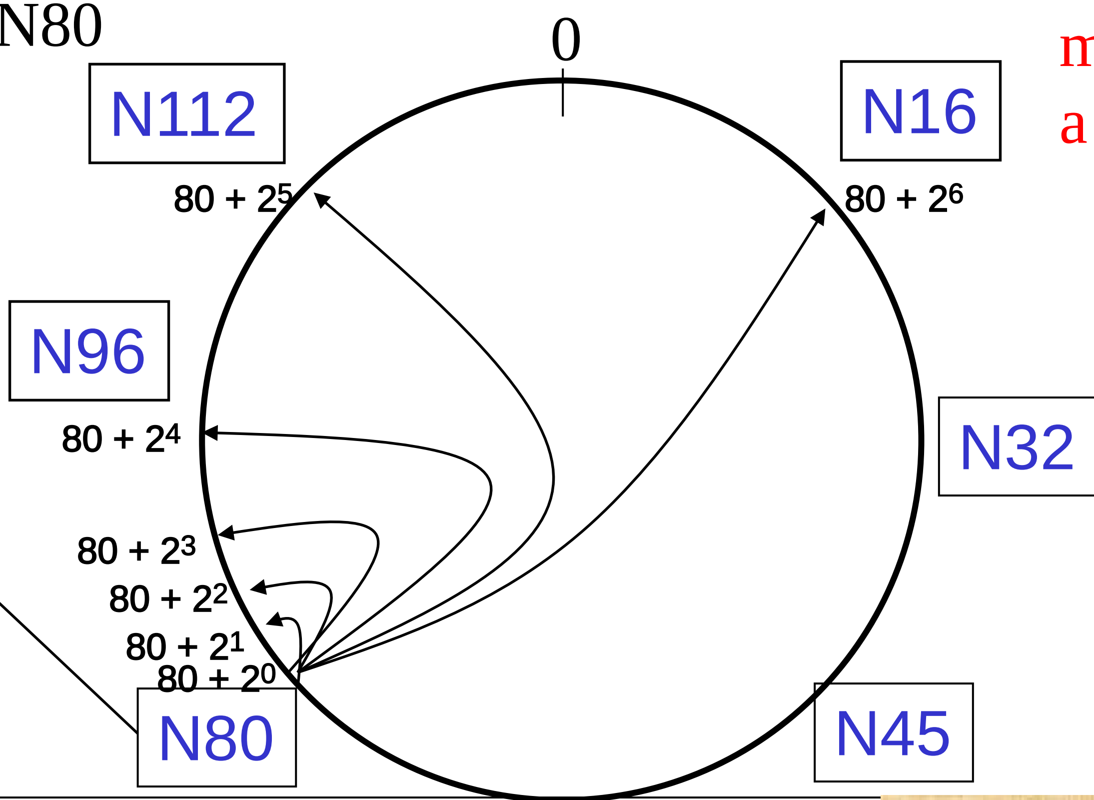
Peer pointers (2): *finger tables*

(Scalable Key Location)

Each node maintains a finger table

Finger Table at N80

i	$ft[i]$
0	96
1	96
2	96
3	96
4	96
5	112
6	16



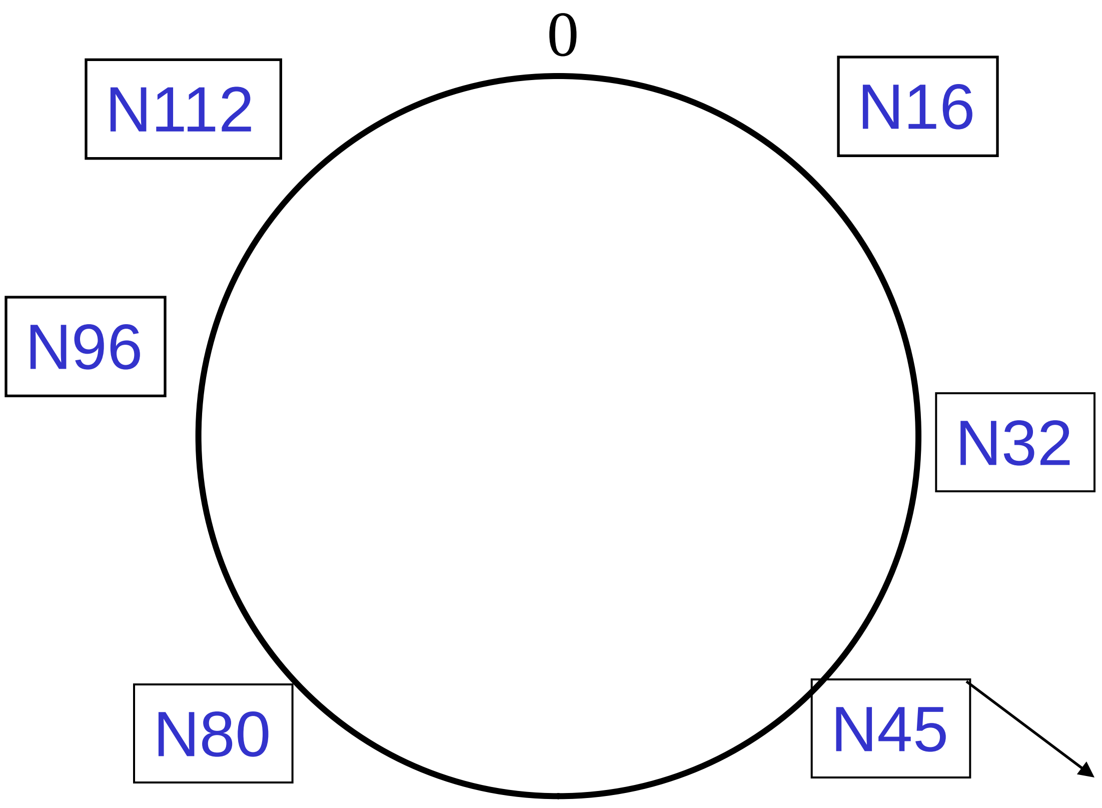
*i*th entry at peer with id *n* is first peer with id >= $n+2^i \pmod{2^m}$

What about the files?

- Filenames are also mapped using **same consistent hash function**
  - SHA-1(filename) →160 bits, truncated to *m* bits=file id or *key*
  - Assume *K* keys
- Example:
  - **File cnn.com/index.html** that maps to file id /key **42** is stored at *first peer* with id >= 42
- Note that we are considering a different file-sharing application here : *cooperative web caching*
  - "Peer"=client browser, "files"=html objects
  - Peers can now fetch html objects from other peers that have them, rather than from server
  - The same discussion applies to any other file sharing application, including that of mp3 files

Mapping Files

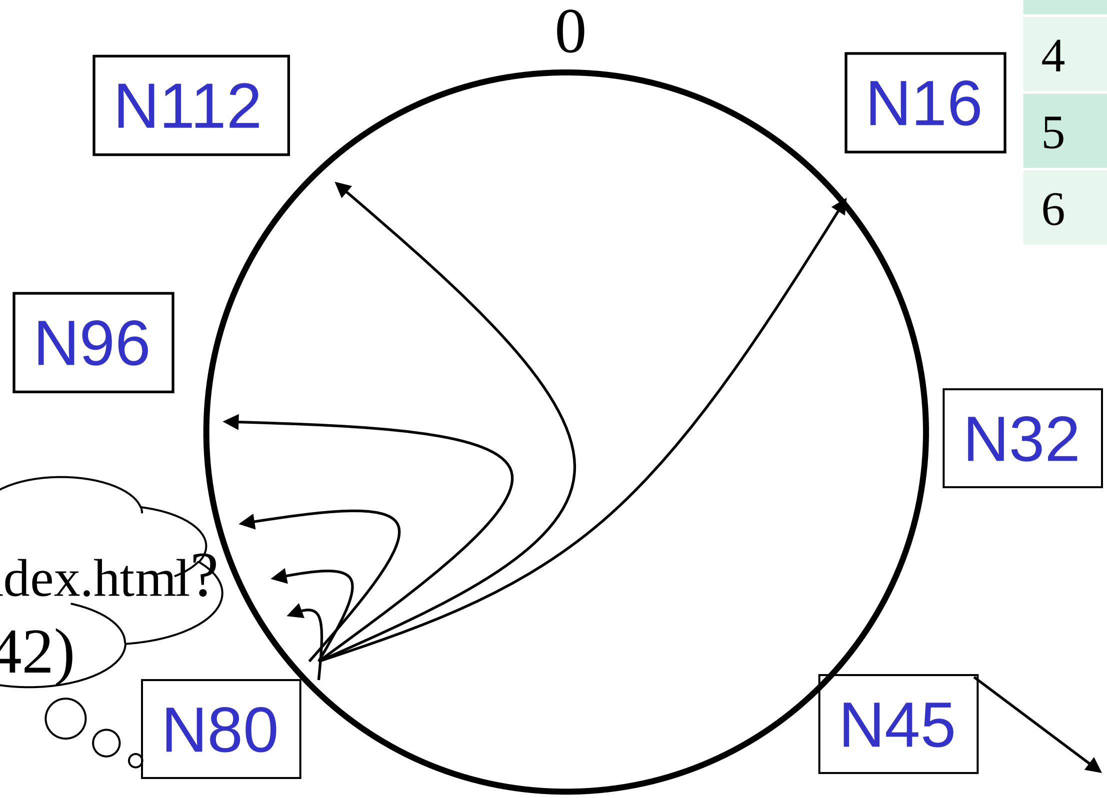
Say *m=7*



File cnn.com/index.html with key K42 stored here

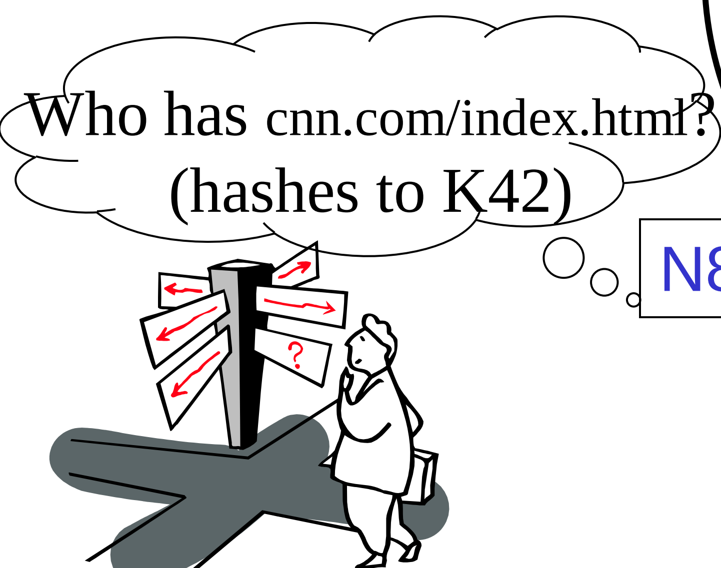
Search

Say m=7



0	32
1	32
2	32
3	32
4	32
5	80
6	80

0	45
1	45
2	45
3	45
4	80
5	80
6	96



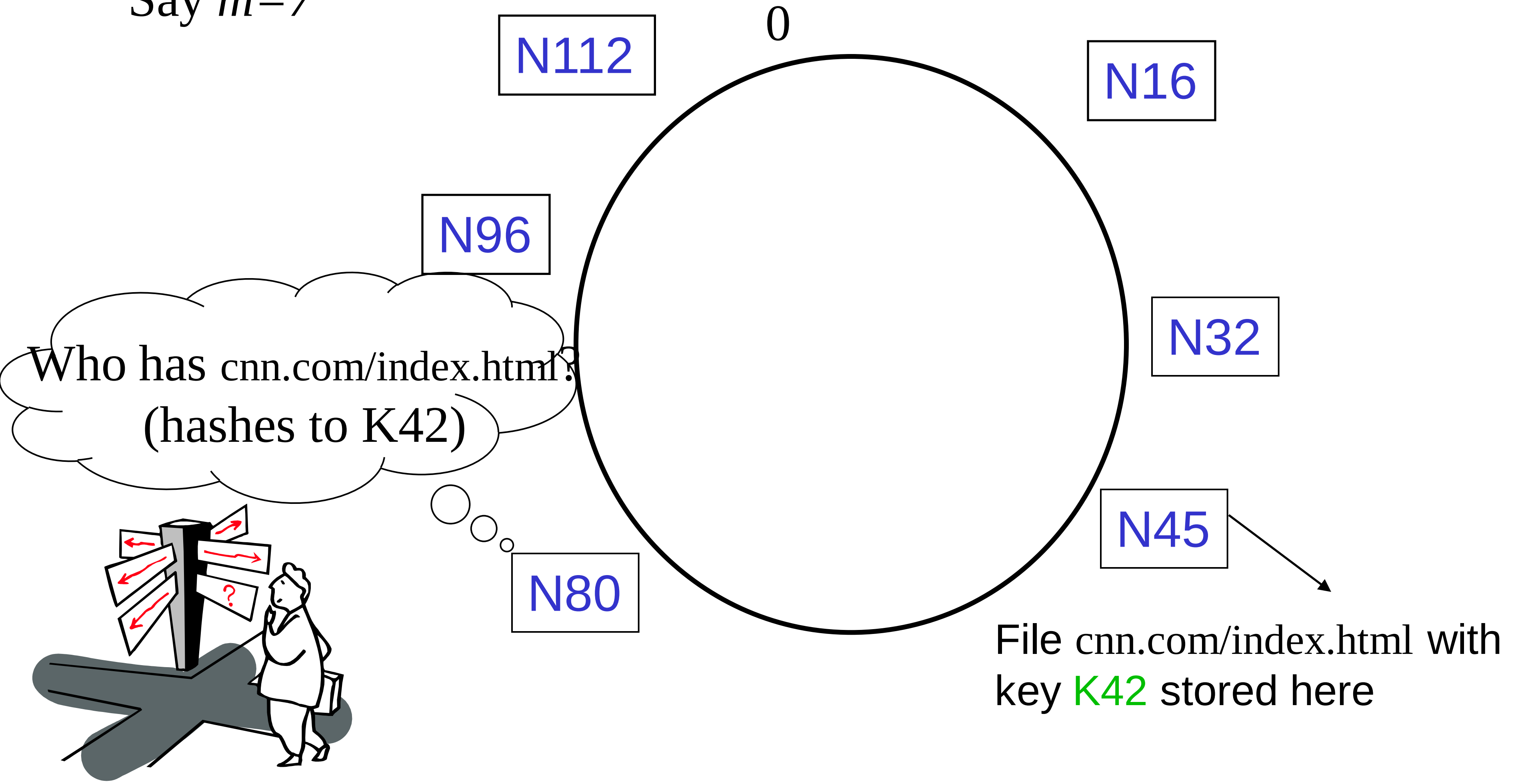
File cnn.com/index.html with key K42 stored here

Search

At node *n*, send query for key *k* to largest successor/finger entry < *k* (all modulo *m*)

if none exist, send query to *successor(n)*

Say *m=7*

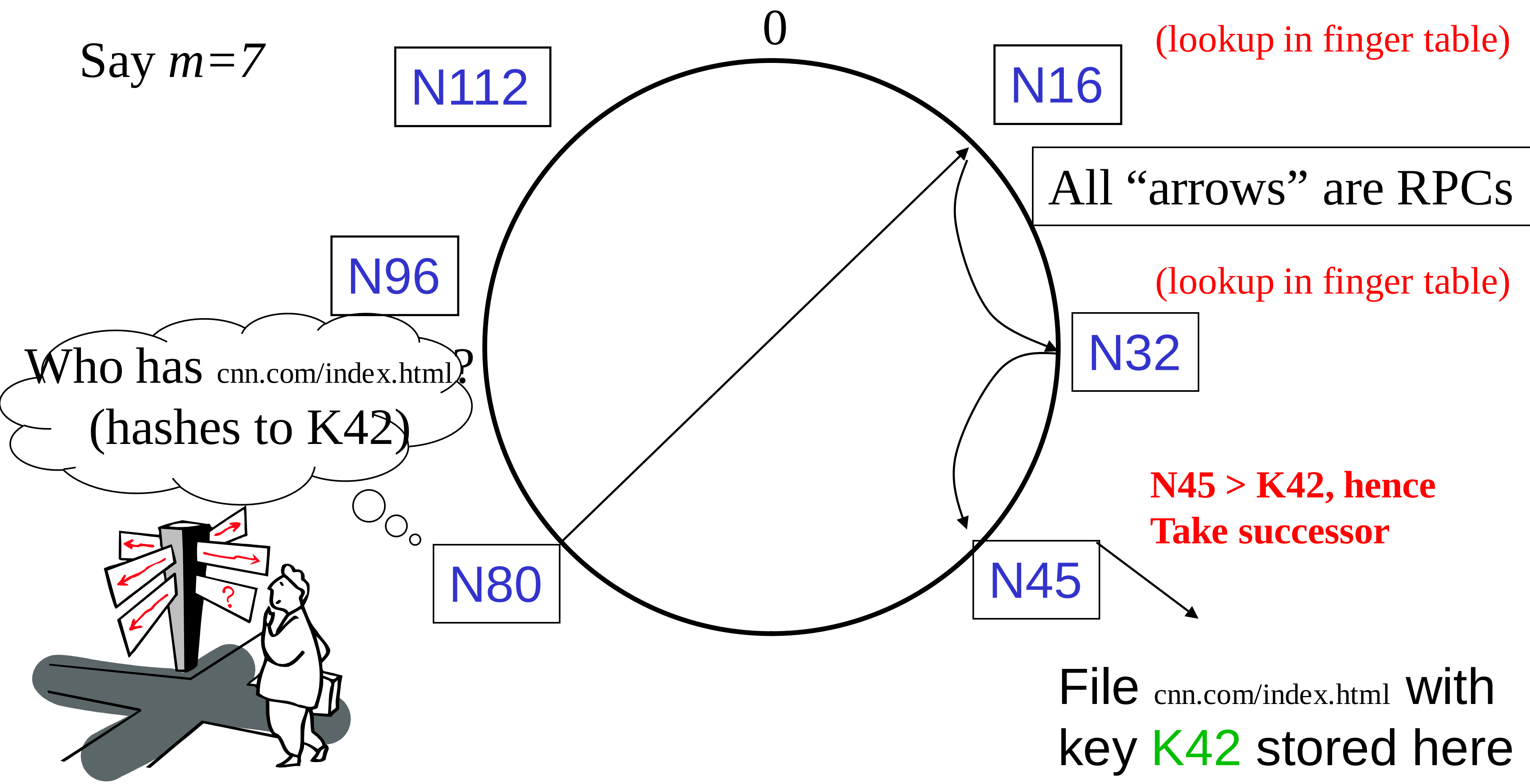


Search

At node *n*, send query for key *k* to largest successor/finger entry < *k* (all mod *m*)

if none exist, send query to *successor(n)*

Say *m=7*



Summary

- Chord protocol
  - *Structured P2P*
  - *O(log(N))* memory and lookup cost
  - Simple lookup algorithm, rest of protocol complicated
- Next lecture – look at the rest of the Chord protocol