

CS 598IG  
 Adv. Topics in Dist. Sys.  
 Spring 2006

Indranil Gupta  
 Lecture 3  
 January 24, 2006

DHT=Distributed Hash Table

- A 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
- Napster, Gnutella, FastTrack are all DHTs
- So is Chord, a structured peer to peer system that we study next

Comparative Performance

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

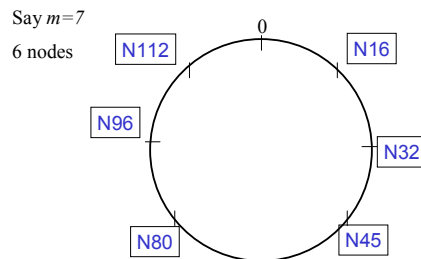
Comparative Performance

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

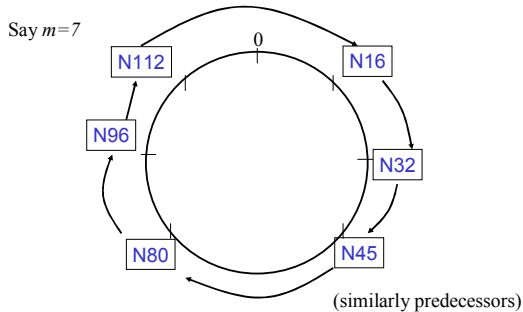
Chord

- 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)
- Uses *Consistent Hashing* on peer's address
  - SHA-1(ip\_address,port)  $\rightarrow$  160 bit string
  - Truncated to  $m$  bits
  - Called *peer id* (number between 0 and  $2^m - 1$ )
  - Not unique but id conflicts very unlikely
  - Can then map peers to one of  $2^m$  logical points on a circle

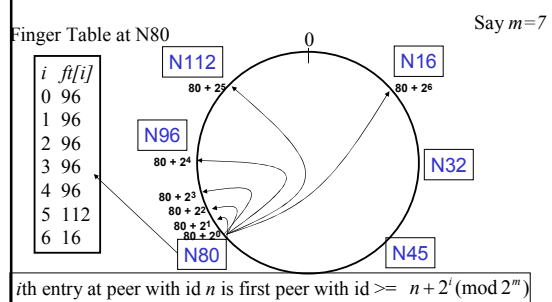
Ring of peers



## Peer pointers (1): successors



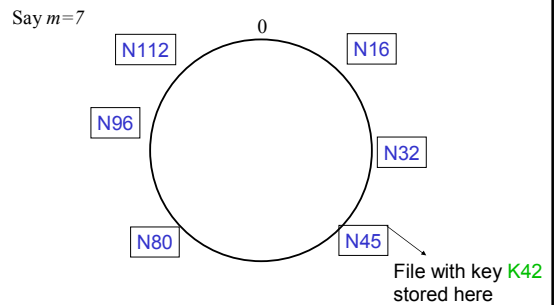
## Peer pointers (2): finger tables



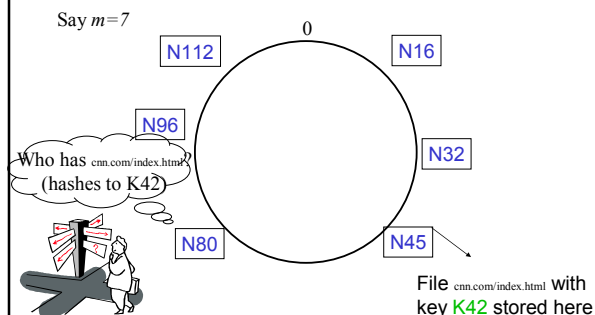
## What about the files?

- Filenames also mapped using same consistent hash function
  - $\text{SHA-1}(\text{filename}) \rightarrow 160 \text{ bit string}$
- File `cmn.com/index.html` that maps to key K42 is stored at first peer with id greater than 42
- Note that we are considering a different file-sharing application here : *cooperative web caching*
  - The same discussion applies to any other file sharing application, including that of mp3 files.

## Mapping Files

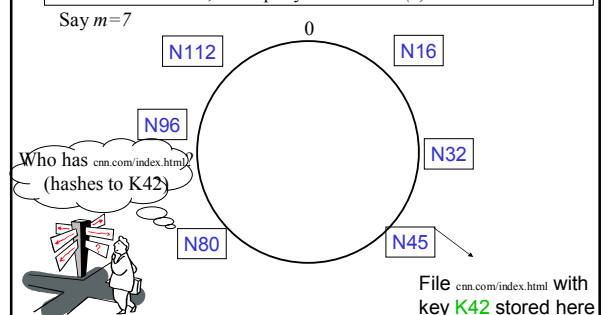


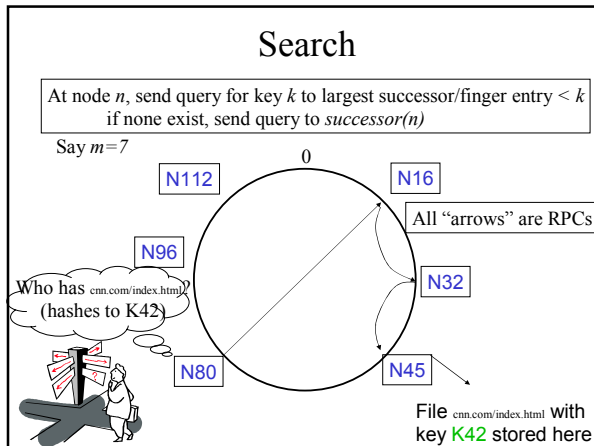
## Search



## Search

At node  $n$ , send query for key  $k$  to largest successor/finger entry  $< k$  if none exist, send query to  $\text{successor}(n)$





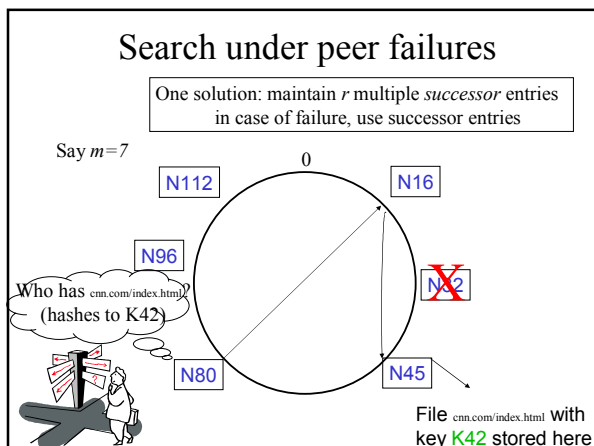
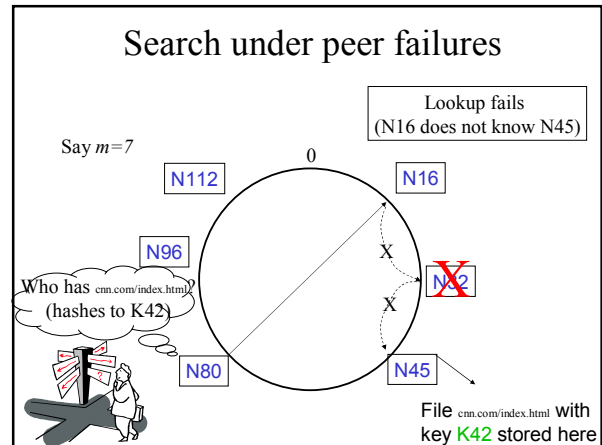
### Analysis

Search takes  $O(\log(N))$  time *w.h.p.*

- Proof 1 (intuition): at each step, distance between query and peer-with-file reduces by a factor of at least 2 (why?)  
Takes at most  $m$  steps: ok if  $2^m$  is at most a constant multiplicative factor above  $N$
- Proof 2 (intuition): after  $\log(N)$  forwardings, distance to key is at most  $2^m / N$  (why?)  
Number of node identifiers in a range of  $2^m / N$  is  $O(\log(N))$  with high probability (why? SHA-1!)  
So using *successors* in that range will be ok

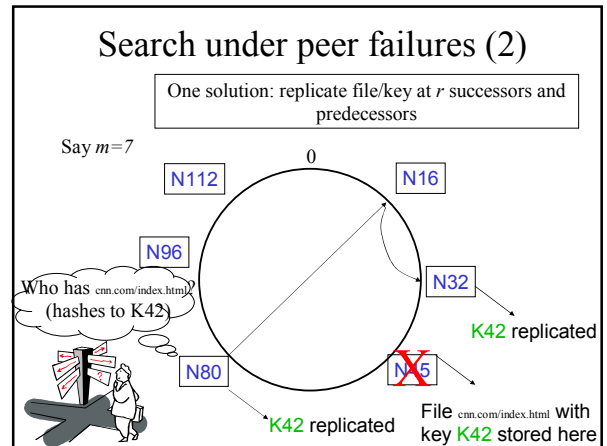
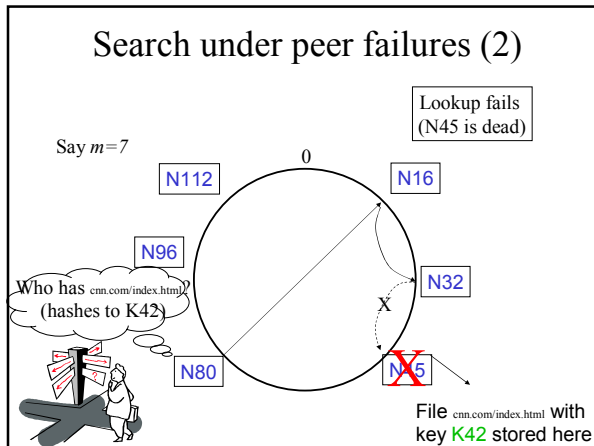
### Analysis (contd.)

- $O(\log(N))$  search time holds for file insertions too (in general for *routing to any key*)
  - "Routing" can thus be used as a building block for other applications than file sharing [can you name one?]
- $O(\log(N))$  time true only if finger and successor entries correct
- When might these entries be wrong?
  - When you have failures



### Search under peer failures

- Choosing  $r=2\log(N)$  suffices to maintain correctness *w.h.p.*
  - Say 50% of nodes fail
  - Pr(at least one successor of given node alive)=  
$$1 - \left(\frac{1}{2}\right)^{2\log N} = 1 - \frac{1}{N^2}$$
  - Pr(above is true at all alive nodes)=  
$$\left(1 - \frac{1}{N^2}\right)^{N/2} = e^{-\frac{1}{2N}} \approx 1$$

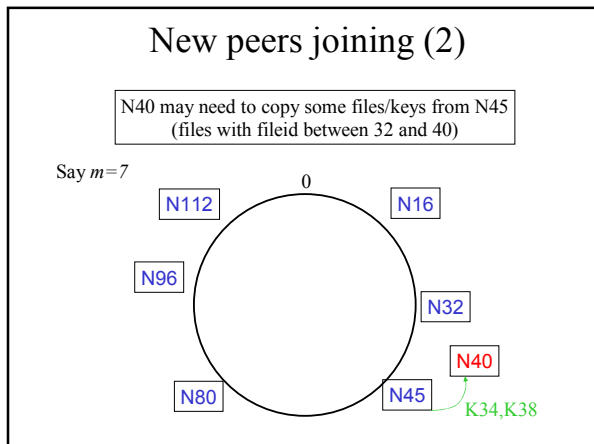
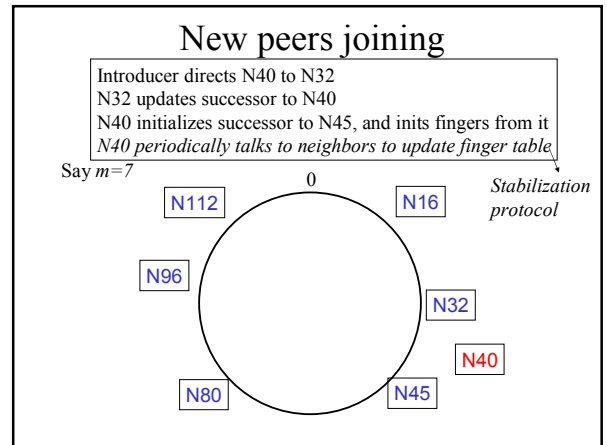


### Need to deal with dynamic changes

- ✓ Peers fail
- New peers join
- Peers leave

All the time

→ Need to update *successors* and *fingers*, and copy keys



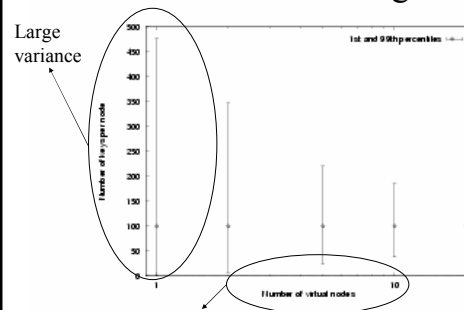
### New peers joining (3)

- A new peer affects  $O(\log(N))$  other finger entries in the system
- Number of messages per peer join =  $O(\log(N) * \log(N))$
- Similar set of operations for dealing with peers leaving

## Experimental Results

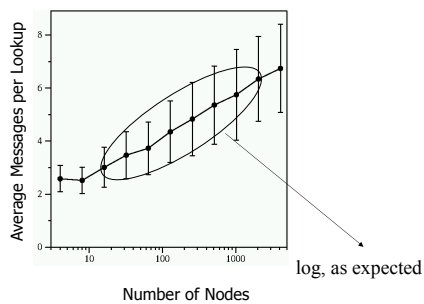
- Sigcomm 01 paper had results from simulation of a C++ prototype
- SOSP 01 paper had more results from a 12-node Internet testbed deployment
- We'll touch briefly on the first set
- 10000 peer system

## Load Balancing



Solution: Each real node joins as  $r$  multiple *virtual nodes*  
 smaller load variation, lookup cost is  $O(\log(N*r)) = O(\log(N) + \log(r))$

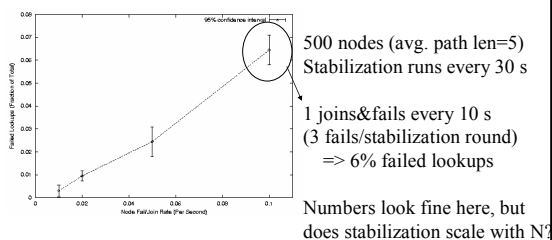
## Lookups



## Stabilization Protocol

- Concurrent peer joins, leaves, failures might cause loopiness of pointers, and failure of lookups
  - Chord peers periodically run a *stabilization* algorithm that checks and updates pointers and keys
  - Ensures non-loopiness of fingers, eventual success of lookups and  $O(\log(N))$  lookups w.h.p.
  - [TechReport on Chord webpage] defines *weak* and *strong* notions of stability
  - Each stabilization round at a peer involves a constant number of messages
  - Strong stability takes  $O(N^2)$  stabilization rounds (!)

## Fault-tolerance



## Wrap-up Notes

- Memory:  $O(\log(N))$  successor pointer,  $m$  finger entries
- Indirection: store a pointer instead of the actual file
- Does not handle partitions (can you suggest a possible solution?)

## Wrap-up Notes (2)

- When nodes are constantly joining, leaving, failing
  - Significant effect to consider: traces from the Overnet system show *hourly* peer turnover rates (*churn*) could be 10-15% of total number of nodes in system
  - Leads to excessive (unnecessary) key copying (remember that keys are replicated)
  - Stabilization algorithm may need to consume more bandwidth to keep up
  - There exist alternative DHTs that are churn-resistant
    - E.g., Kelips

## Wrap-up Notes (3)

- Virtual Nodes good for load balancing, but
  - Effect on peer traffic?
  - Result of churn?
- Current status of project:
  - Protocol constantly undergoing change
  - File systems (CFS,Ivy) built on top of Chord
  - DNS lookup service built on top of Chord
  - Spawned research on many interesting issues about p2p systems

<http://www.pdos.lcs.mit.edu/chord/>

## Summary

- Chord protocol
  - More structured than Gnutella
  - $O(\log(N))$  memory and lookup cost
  - Simple lookup algorithm, rest of protocol complicated
  - Stabilization works, but how far can it go?

## Administrative Announcements

### *Student-led paper presentations*

- **Start from February 7th**
- **Groups of up to 2 students** each class, responsible for a set of 3 “Main Papers” on a topic
  - 45 minute presentations (total) followed by discussion
  - ***Set up appointment with me to show slides by 5 pm day prior to presentation***
- List of papers is up on the website
- Each of the other students expected to **read the papers before class** and turn in a one to two page **review** of the **any two** of the main set of papers (summary, comments, criticisms and possible future directions)

## Announcements (contd.)

- Presentation Deadline: **form groups by midnight of January 31 (next Tuesday) by dropping by my office hours (4 pm – 5 pm, Tu, Th in 3112 SC)**
  - Hurry! Some interesting topics are already taken!
  - I can help you find partners
- *Use course newsgroup for forming groups and discussion: [class.cs598ig](mailto:class.cs598ig)*

## Announcements (contd.)

### *Projects*

- Groups of 2 (need not be same as presentation groups)
- We’ll start detailed discussions “soon” (a few classes into the student-led presentations)

## Next Lecture

- Sensor Networks and Theoretical Distributed Computing – Basics
  - Please read papers / references from web page.  
Please print out yourself
- Signup for presentations by next Tuesday

## A question before you go

What new *design techniques* about p2p systems have you learnt in this class?

As you head out that door, think about at least 3 new design techniques (about p2p systems) that you have learnt in this lecture.