

CS 525
Advanced Topics in Distributed
Systems
Spring 08

Indranil Gupta
Lecture 3
Introduction to Peer to Peer Systems - II
January 22, 2008

Two types of P2P Systems



Systems that work well in practice but with no big/famous names
• *Non-academic P2P systems*
e.g., Gnutella, Napster (previous lecture)

Systems with big/famous names but that may or may not work well
• *Academic P2P systems*
e.g., Chord (this lecture)



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

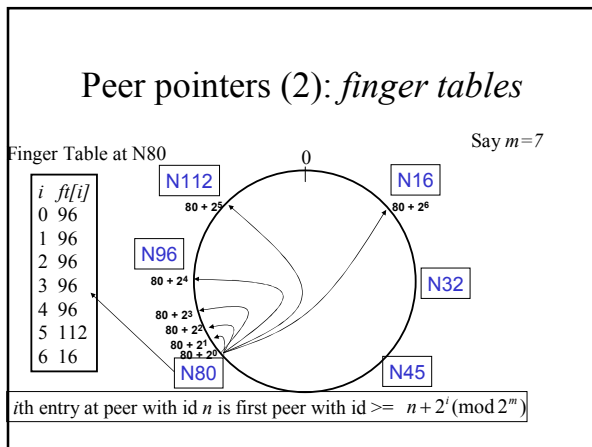
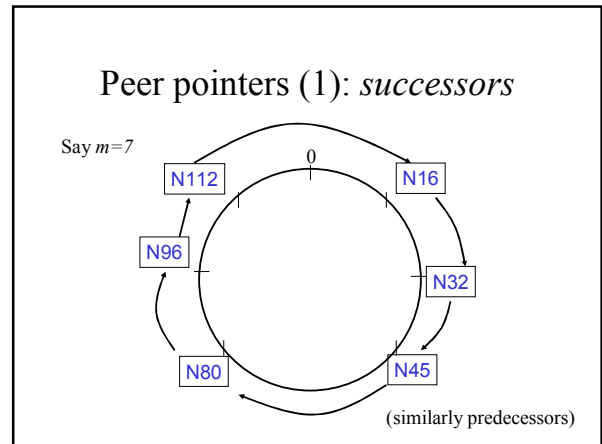
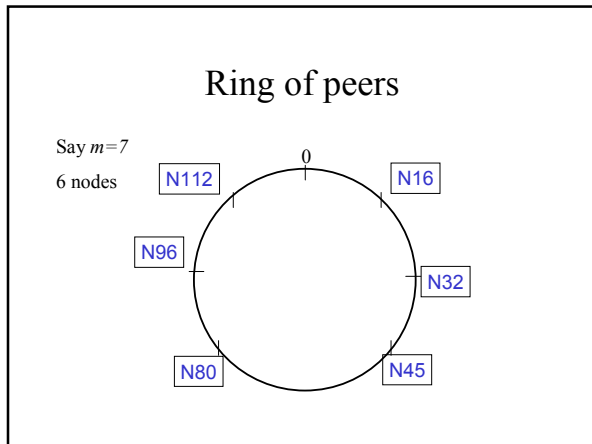
	Memory	Lookup Latency	#Messages for a lookup
Napster	$O(1)$ $(O(N)@server)$	$O(1)$	$O(1)$
Gnutella	$O(N)$	$O(N)$	$O(N)$

Comparative Performance

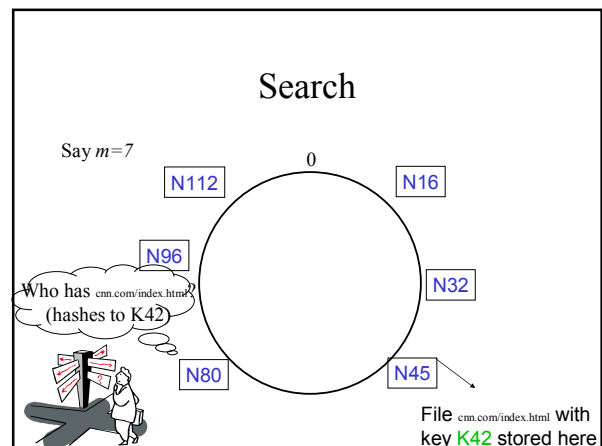
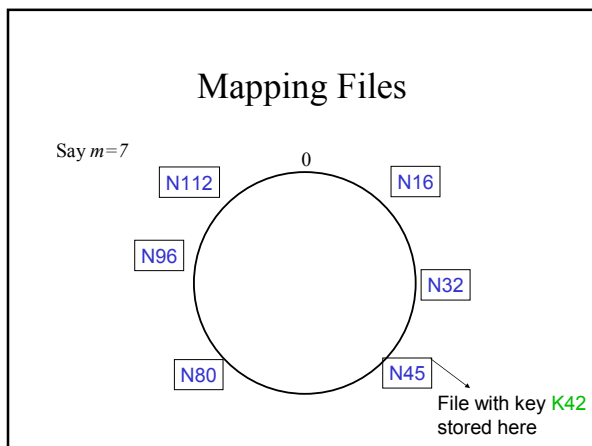
	Memory	Lookup Latency	#Messages for a lookup
Napster	$O(1)$ $(O(N)@server)$	$O(1)$	$O(1)$
Gnutella	$O(N)$	$O(N)$	$O(N)$
Chord	$O(\log(N))$	$O(\log(N))$	$O(\log(N))$

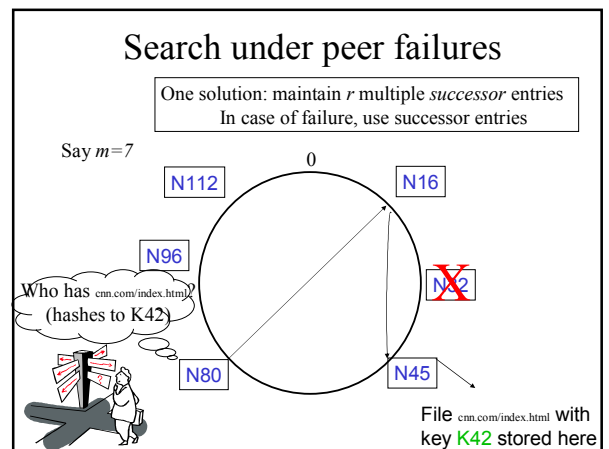
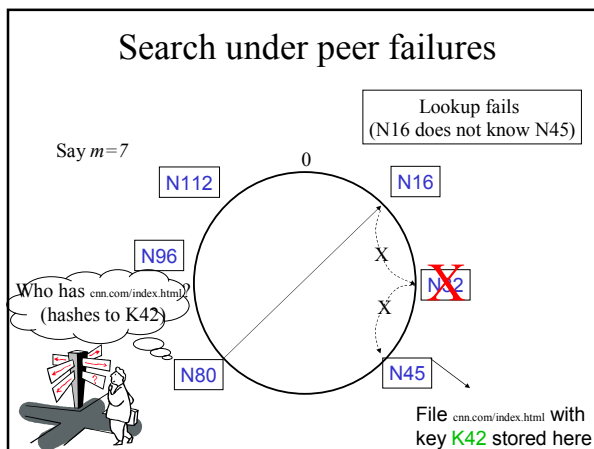
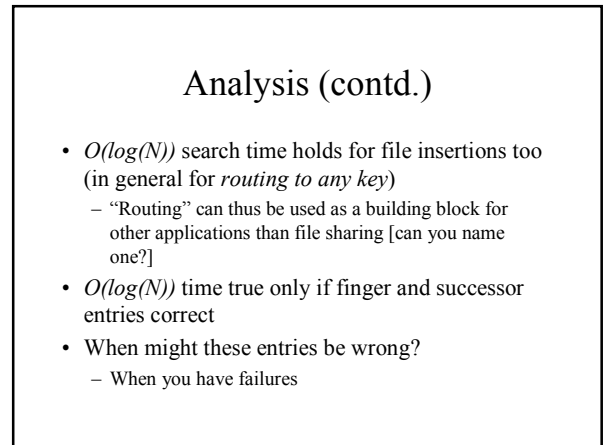
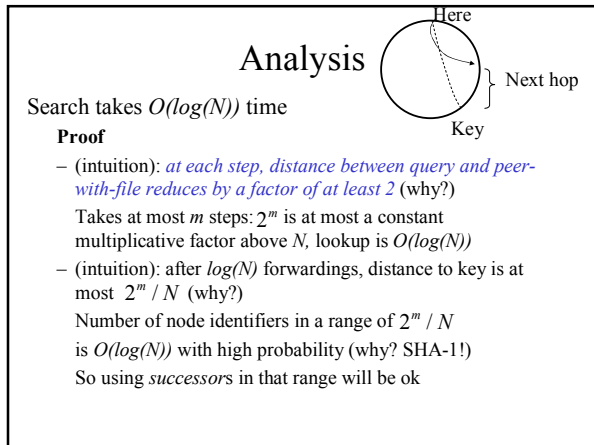
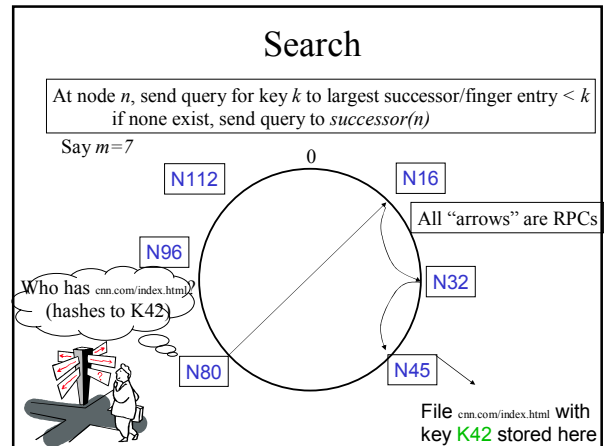
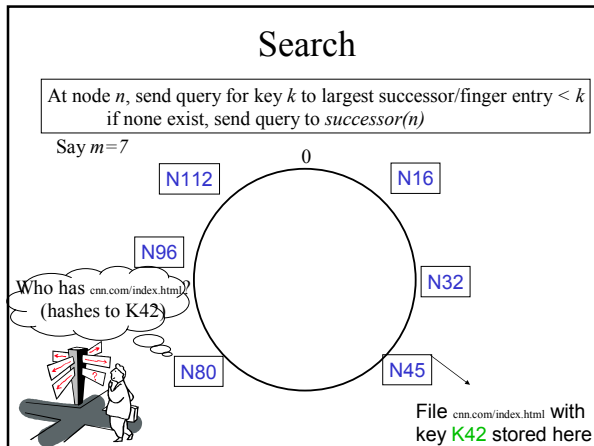
Chord

- 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 node's (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



- ### What about the files?
- Filenames also mapped using same consistent hash function
 - SHA-1(filename) \rightarrow 160 bit string (*key*)
 - File is stored at first peer with id greater than its key
 - File `cnm.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.





Search under peer failures

- Choosing $r=2\log(N)$ suffices to maintain *lookup correctness* w.h.p.
 - Say 50% of nodes fail
 - Pr(at given node, at least one successor alive)=

$$1 - \left(\frac{1}{2}\right)^{2\log N} = 1 - \frac{1}{N^2}$$
 - Pr(above is true at all alive nodes)=

$$\left(1 - \frac{1}{N^2}\right)^{N/2} = e^{-\frac{1}{2N}} \approx 1$$

Search under peer failures (2)

Lookup fails (N45 is dead)

Say $m=7$

Search under peer failures (2)

One solution: replicate file/key at r successors and predecessors

Say $m=7$

Need to deal with dynamic changes

- ✓ Peers fail
- New peers join
- Peers leave
 - P2P systems have a high rate of *churn* (node join, leave and failure)
 - 25% per hour in Overnet (eDonkey)
 - 100% per hour in Gnutella
 - Lower in managed clusters, e.g., CSIL

So, all the time, need to:

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

New peers joining

Introducer directs N40 to N32
N32 updates successor to N40
N40 initializes successor to N45, and inits fingers from it
N40 periodically talks to neighbors to update finger table

Say $m=7$

New peers joining (2)

N40 may need to copy some files/keys from N45 (files with fileid between 32 and 40)

Say $m=7$

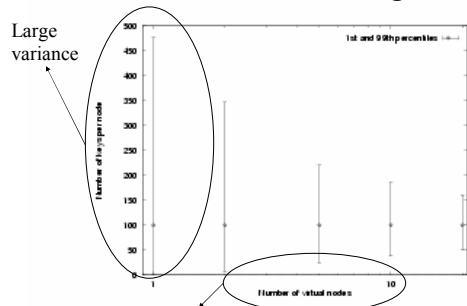
New peers joining (3)

- A new peer affects $O(\log(N))$ other finger entries in the system, on average [Why?]
- Number of messages per peer join = $O(\log(N) * \log(N))$
- Similar set of operations for dealing with peers leaving
 - For dealing with failures, also need *failure detectors* (we'll see these later in the course!)

Experimental Results

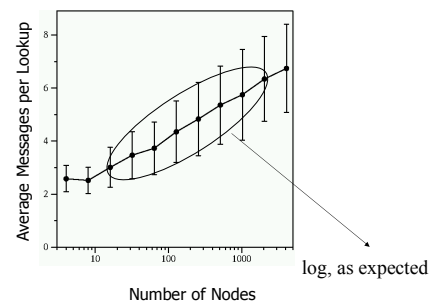
- Sigcomm 01 paper had results from simulation of a C++ prototype
- SOSp 01 paper had more results from a 12-node Internet testbed deployment
- We'll touch briefly on the first set
- 10000 peer system

Load Balancing



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

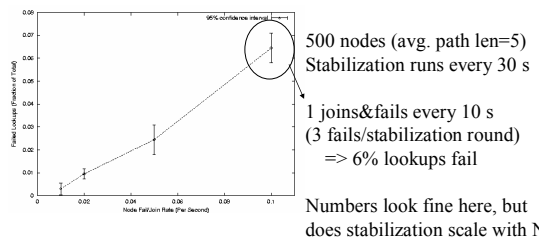
Lookups



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



Numbers look fine here, but does stabilization scale with N^2 ?

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 25-100% 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 (later in the course)

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 12th**
- **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 (non-presenters) 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 Thursday) by dropping by my office hours (10.45 am – 12 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.cs525*

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)

- Please turn in filled-out "Student Infosheets" today or next lecture.

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 Thursday

A question before you go

What new *design techniques* about p2p systems have you learnt in this lecture and the previous lecture??

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