Peer-peer Computing & Networking

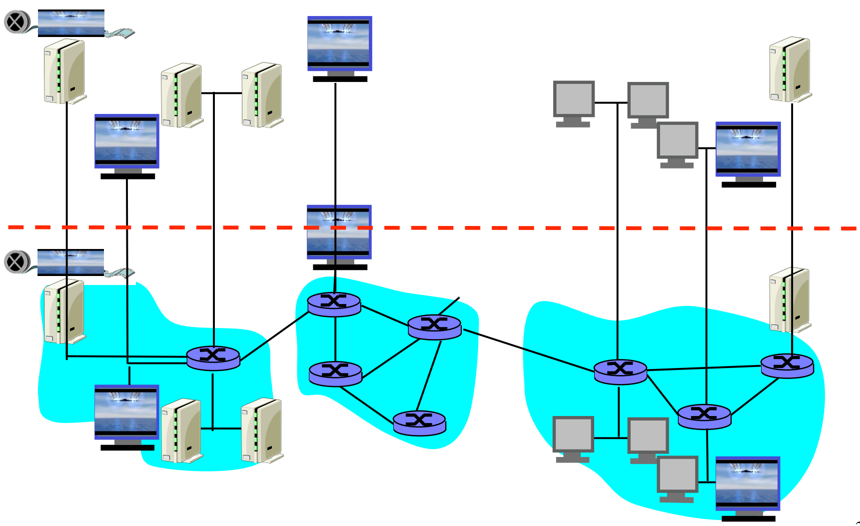
CS 707

Acknowledgements

Some of the followings slides are based on the slides made available by the authors of
Computer Networking: A Top Down Approach Featuring the Internet, 2nd edition.
Jim Kurose, Keith Ross
Addison-Wesley, July 2002.

and from talks by Robert Morris (MIT)

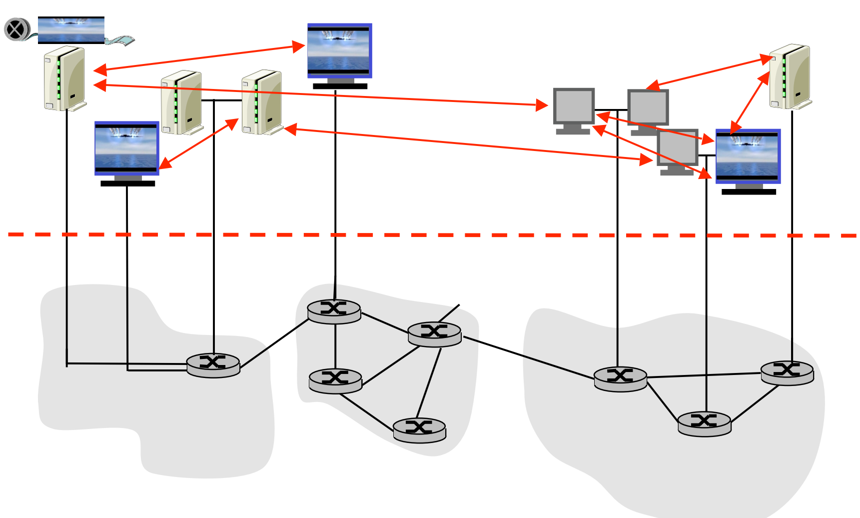
Peer-peer computing and networking



3

Peer-peer network

Focus at the application level



4

Peer-to-Peer: Some Definitions

- A P2P computer network refers to any network that does not have fixed clients and servers, but a number of peer nodes that function as both clients and servers to other nodes on the network.
  Wikipedia.org
- The sharing of computer resources and services by direct exchange between systems
  Intel P2P working group
- The use of devices on the internet periphery in a non-client capacity
  Alex Weytsel, Aberdeen Group
- P2P is a class of applications that takes advantage of resources – storage, cycles, content, human presence – available at the edges of the internet.
  Clay Shirky, openp2p.com

5

Peer-peer applications

- File sharing
  - Napster, Gnutella, KaZaa
  - Second generation projects
    - Oceanstore, PAST, Freehaven
- Distributed Computation
  - SETI@home, Entropia, Parabon, United Devices, Popular Power
- Other Applications
  - Content Distribution (BitTorrent)
  - Instant Messaging (Jabber), Anonymous Email
  - Groupware (Groove)
  - P2P Databases

6

Is Peer-to-peer new?

- P2P concept certainly not new
  - Usenet - News groups first truly decentralized system
  - DNS - Handles huge number of clients
  - Basic IP - Vastly decentralized, many equivalent routers
- What is new?
  - Scale: people are envisioning much larger scale
  - Security: Systems must deal with privacy and integrity
  - Anonymity: Protect identity and prevent censorship
  - (In)Stability: Deal with unstable components at the edges

7

P2P: Related Technologies

- Distributed computing.
  - How is P2P different from distributed computing?
- Grid computing.
  - How is the computational grid different from P2P networks?

  KEY DIFFERENCES: Peers are on the edges of the Internet, are autonomous, have variable connectivity, and temporary network addresses
- Application-level networking.
  - Resilient overlay networks for multicast, video distribution, etc.

8

P2P: Related Technologies

- Wireless ad-hoc networks.
- Sensor networks.
- P2P devices/ubiquitous computing.
  - JINI.
- Web services.
  - .NET framework, SOAP, UDDI.

9

Why the hype???

- File Sharing: Napster (+Gnutella, KaZaa, etc)
  - High coolness factor
  - Served a high-demand niche: online jukebox
- Anonymity/Privacy/Anarchy: FreeNet, Publis, etc
  - Libertarian dream of freedom
  - Extremely valid concern of Censorship/Privacy
  - In search of copyright violators, RIAA challenging rights to privacy
- Computing: The Grid
  - Scavenge the numerous free cycles of the world to do work
  - Seti@Home most visible version of this
- Industry/Management
  - Looking for the next big thing
  - A lot of interest/hype in "autonomic computing"/Computing as a utility

10

P2P Applications Taxonomy

- Content and File Sharing
  - Napster, Gnutella, KaZaa, etc.
  - Most research has focused on this class of apps
- Parallelizable
  - Compute Intensive (Same task on every peer using different parameters)
  - Componentized applications – different components on each peer (not yet widely supported/recognized)
- Collaborative
  - Instant messaging, groupware, games
  - Many startups but not that much academic research

P2P file sharing

Example

- Alice runs P2P client application on her notebook computer
- Intermittently connects to Internet; gets new IP address for each connection
- Asks for "Hey Jude"
- Application displays other peers that have copy of Hey Jude.
- Alice chooses one of the peers, Bob.
- File is copied from Bob's PC to Alice's notebook: HTTP
- While Alice downloads, other users uploading from Alice.
- Alice's peer is both a Web client and a transient Web server.

All peers are servers = highly scalable!

P2P Content Location & Routing

- Three approaches
  - Centralized directory (Napster)
  - Decentralized directory + Flooding-based search (Gnutella)
    - Unstructured P2P systems
  - Distributed Hash Tables (DHT) based document search and publication
    - Structured P2P systems (Chord, CAN, Tapestry, etc)
    - Presented in weeks 2 & 3

13

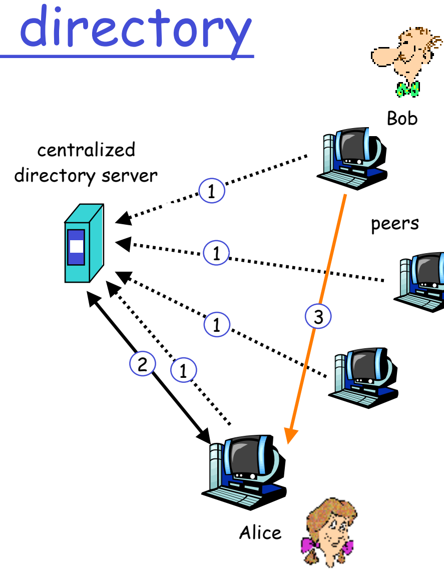
P2P: centralized directory

original "Napster" design

1) when peer connects, it informs central server:
   - IP address
   - content

2) Alice queries for "Hey Jude"

3) Alice requests file from Bob



14

P2P: problems with centralized directory

- Single point of failure
- Performance bottleneck
- Copyright infringement

> file transfer is decentralized, but locating content is highly centralized

15

Napster

- program for sharing files over the Internet
- a killer application?
- history:
  - 5/99: Shawn Fanning (freshman, Northeasten U.) founds Napster Online music service
  - 12/99: first lawsuit
  - 3/00: 25% UWisc traffic Napster
  - 2000: est. 60M users
  - 2/01: US Circuit Court of Appeals: Napster knew users violating copyright laws
  - 7/01: # simultaneous online users:

    Napster 160K, Gnutella: 40K, Morpheus: 300K
  - 2001: Napster shut down; Bertelsmann acquire assets, etc.
- Today
  - Napster 2.0 music download service (Roxio)
  - Also OpenNap (open source napster server)

16

Napster: how did it work

Application-level, client-server protocol over point-to-point TCP

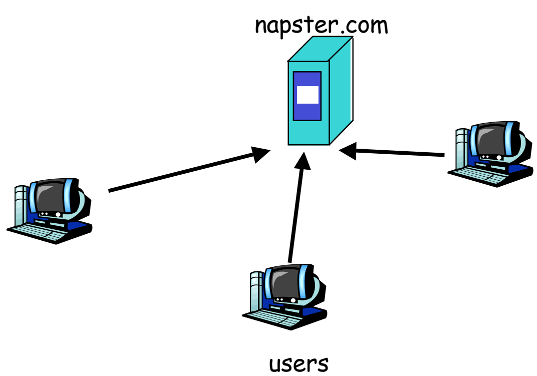
Four steps:

- Connect to Napster server
- Upload your list of files (push) to server.
- Give server keywords to search the full list with.
- Select "best" of correct answers. (pings)

17

Napster

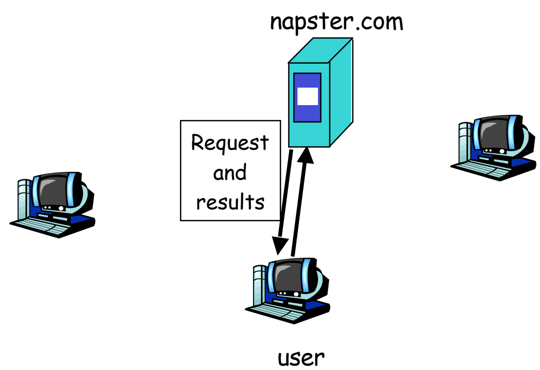
1. File list is uploaded



18

Napster

2. User requests search at server.

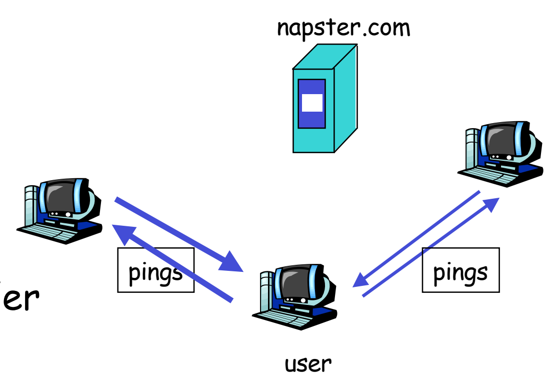


19

Napster

3. User pings hosts that apparently have data.

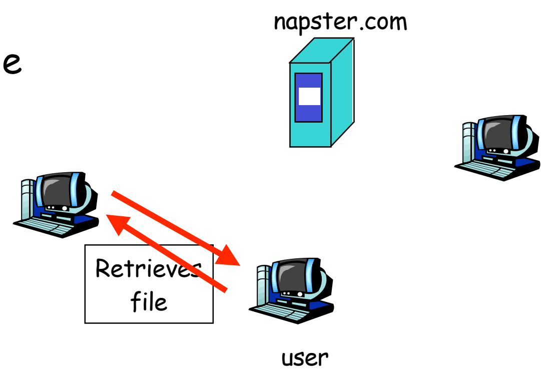
Looks for best transfer rate.



20

Napster

4. User retrieves file

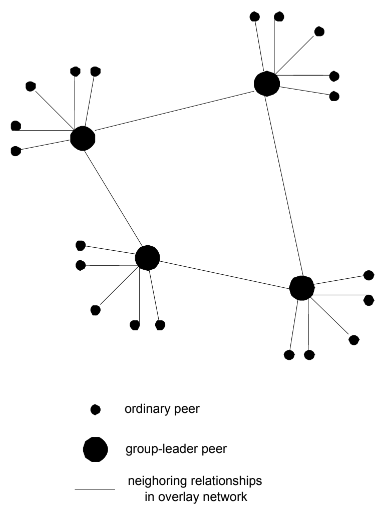


Napster: architecture notes

- centralized server:
  - single logical point of failure
  - can load balance among servers using DNS rotation
  - potential for congestion
- no security:
  - passwords in plain text
  - no authentication
  - no anonymity

P2P: decentralized directory

- Each peer is either a group leader or assigned to a group leader.
- Group leader tracks the content in all its children.
- Peer queries group leader; group leader may query other group leaders.



23

More about decentralized directory

overlay network

- peers are nodes
- edges between peers and their group leaders
- edges between some pairs of group leaders
- virtual neighbors

bootstrap node

- connecting peer is either assigned to a group leader or designated as leader

advantages of approach

- no centralized directory server
  - location service distributed over peers
  - more difficult to shut down

disadvantages of approach

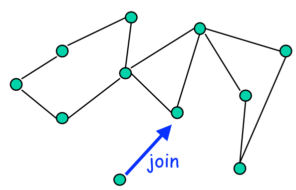
- bootstrap node needed
- group leaders can get overloaded

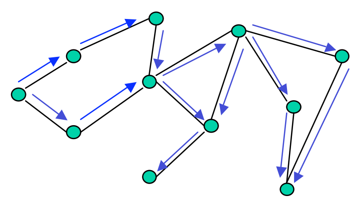
24

P2P: Query flooding

- Gnutella
- no hierarchy
- use bootstrap node to learn about others
- join message

- Send query to neighbors
- Neighbors forward query
- If queried peer has object, it sends message back to querying peer





25

P2P: more on query flooding

Pros

- peers have similar responsibilities: no group leaders
- highly decentralized
- no peer maintains directory info

Cons

- excessive query traffic
- query radius: may not have content when present
- bootstrap node
- maintenance of overlay network

26

Gnutella

- peer-to-peer networking: applications connect to peer applications
- focus: decentralized method of searching for files
- each application instance serves to:
  - store selected files
  - route queries (file searches) from and to its neighboring peers
  - respond to queries (serve file) if file stored locally
- Gnutella history:
  - 3/14/00: release by AOL, almost immediately withdrawn
  - too late
  - many iterations to fix poor initial design (poor design turned many people off)
- What we care about:
  - How much traffic does one query generate?
  - how many hosts can it support at once?
  - What is the latency associated with querying?
  - Is there a bottleneck?

27

Gnutella: how it works

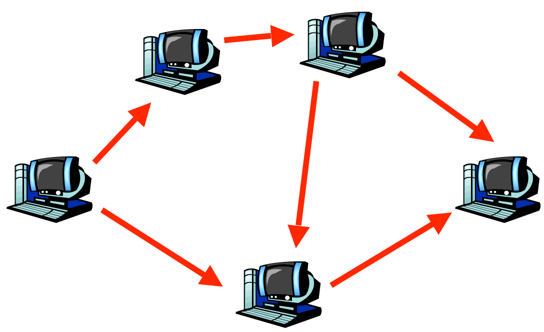
Searching by flooding:

- If you don't have the file you want, query 7 of your partners.
- If they don't have it, they contact 7 of their partners, for a maximum hop count of 10.
- Requests are flooded, but there is no tree structure.
- No looping but packets may be received twice.
- Reverse path forwarding

Note: Play gnutella animation at:
http://www.limewire.com/index.jsp/p2p

28

Flooding in Gnutella: loop prevention



Seen already list: “A”

29

Distributed Computing

- Current supercomputers are too expensive
  - ASCI White (#1 in TOP500) costs more than $110 million and needed a new building
  - Few institutions or research groups can afford this level of investment
- There are more than 500 million PCs around the world
  - some as powerful as early 90s supercomputers
  - they are idle most of the time (60% to 90%), even when being used (spreadsheet, typing, printing,...)
  - corporations and institutions have hundreds or thousands of PCs on their networks

⇩

Try to harness idle PCs on a network and use them on computationally intensive problems

30

How it works

- Embarrassingly parallel applications
  - Large computation to communication ratio
  - Master/worker model
  - Applications can use local disk for checkpointing
- Provider farms out work to idle PCs across the internet
  - PC owners volunteer idle cycles (for money or altruistic purposes)

31

Entropia network

- Born in 1997 to apply idle computers worldwide to problems of scientific interest
- In 2 years grew to more than 30,000 computers with aggregate speed of over 1 Tflop/second
- Several scientific achievements, e.g. Identification of largest known prime number
- Gone commercial: www.entropia.com and used for applications from:
  - Life sciences
  - Financial services
  - Product design, etc.
- Today: appears to not have succeeded as a business
  - Business model for distributed computing not yet successful

32

SETI @ home project

setiathome.ssl.berkeley.edu

- SETI = Search for Extraterrestrial Intelligence
- Started in 1996 to enlist PCs to work on analyzing data from the Arecibo radio telescope
- Good mix of popular appeal and good technology

⇨
- Now running on more than _ million PCs
- delivering ~ 1,200 CPU years per day
- ~ 35 Tflops/sec
- fastest (but special-purpose) computer in the world

33

DHTs

- Distributed Hash Tables: a building block for P2P applications
- First generation of DHTs
  - Tapestry (Zhao et al -- UC Berkeley)
  - Pastry (Rowstron et al - Microsoft Research)
  - Chord (Morris - MIT)
  - CAN (Ratnasamy et al - UC Berkeley)
- Several other DHTs have been proposed
  - Symphony, Kademlia, etc.

34

What Is a DHT?

- Single-node hash table:
  - key = Hash(name)
  - put(key, value)
  - get(key) -> value
    - Service: O(1) storage
- How do I do this across millions of hosts on the Internet?
  - *Distributed* Hash Table

What Is a DHT?

Distributed Hash Table:

key = Hash(data)

lookup(key) -> IP address (Chord)

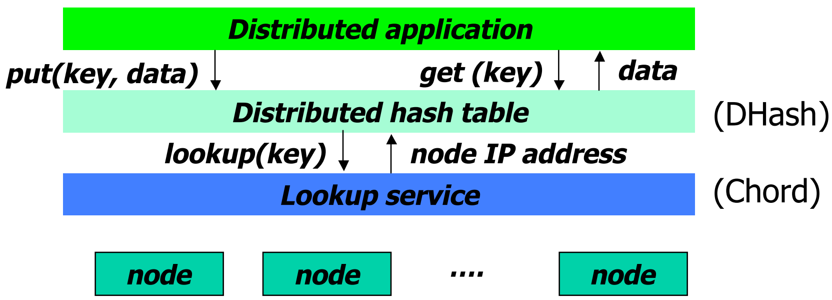
send-RPC(IP address, PUT, key, value)

send-RPC(IP address, GET, key) -> value

Possibly a first step towards truly large-scale distributed systems

- a tuple in a global database engine
- a data block in a global file system
- rare.mp3 in a P2P file-sharing system

DHTs



- Application may be distributed over many nodes
- DHT distributes data storage over many nodes

37

Why the put()/get() interface?

- API supports a wide range of applications
  - DHT imposes no structure/meaning on keys
- Key/value pairs are persistent and global
  - Can store keys in other DHT values
  - And thus build complex data structures

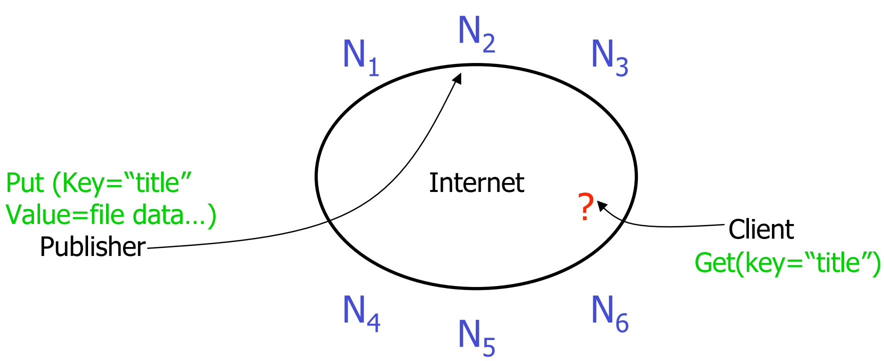
38

Why Might DHT Design Be Hard?

- Decentralized: no central authority
- Scalable: low network traffic overhead
- Efficient: find items quickly (latency)
- Dynamic: nodes fail, new nodes join
- General-purpose: flexible naming

39

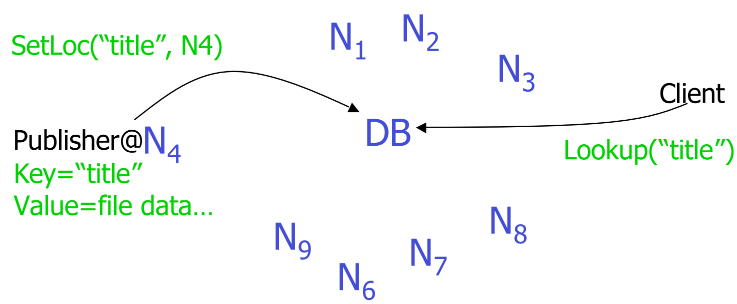
The Lookup Problem



- At the heart of all DHTs

40

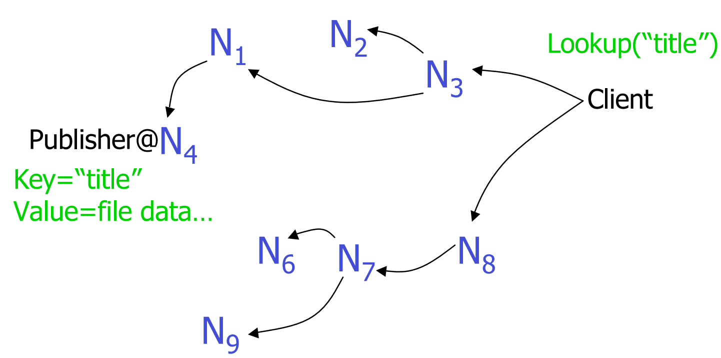
Motivation: Centralized Lookup (Napster)



Simple, but O(N) state and a single point of failure

41

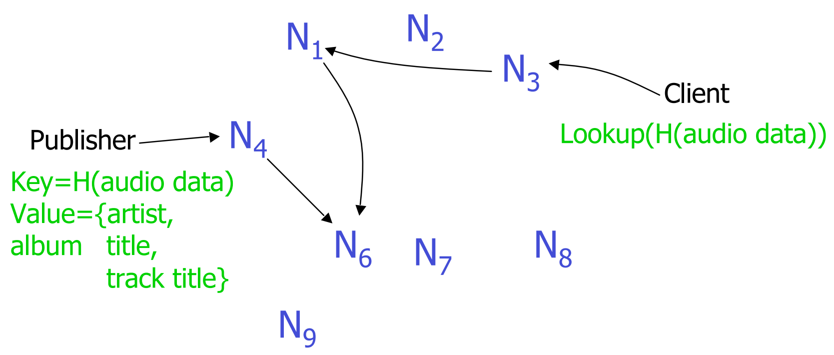
Motivation: Flooded Queries (Gnutella)



Robust, but worst case O(N) messages per lookup

42

Motivation: Routed DHT Queries (Tapestry, Pastry, Chord, CAN, etc)



DHT Applications

- global file systems [OceanStore, CFS, PAST, Pastiche, UsenetDHT]
- naming services [Chord-DNS, Twine, SFR]
- DB query processing [PIER, Wisc]
- Internet-scale data structures [PHT, Cone, SkipGraphs]
- communication services [i3, MCAN, Bayeux]
- event notification [Scribe, Herald]
- File sharing [OverNet]

Chord Simplicity

- Resolution entails participation by O($log(N)$) nodes
- Resolution is efficient when each node enjoys accurate information about O($log(N)$) other nodes
- Resolution is possible when each node enjoys accurate information about 1 other node

"Degrades gracefully"

45

Chord Algorithms

- Basic Lookup
- Node Joins
- Stabilization
- Failures and Replication

46

Chord Properties

- Efficient: O($log(N)$) messages per lookup
  - N is the total number of servers
- Scalable: O($log(N)$) state per node
- Robust: survives massive failures

- Proofs are in paper / tech report
  - Assuming no malicious participants

47

Chord IDs

- Key identifier = SHA-1(key)
- Node identifier = SHA-1(IP address)
- Both are uniformly distributed
- Both exist in the same ID space

- How to map key IDs to node IDs?

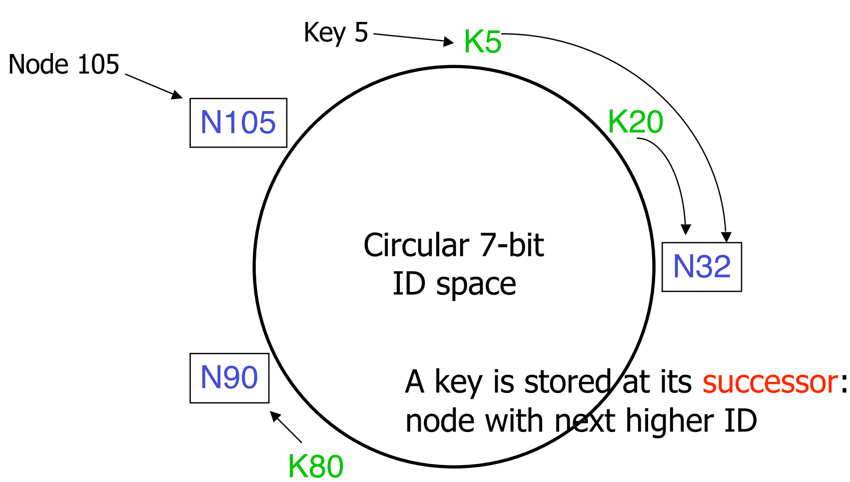
48

Consistent Hashing[Karger 97]

- Target: web page caching
- Like normal hashing, assigns items to buckets so that each bucket receives roughly the same number of items
- Unlike normal hashing, a small change in the bucket set does not induce a total remapping of items to buckets

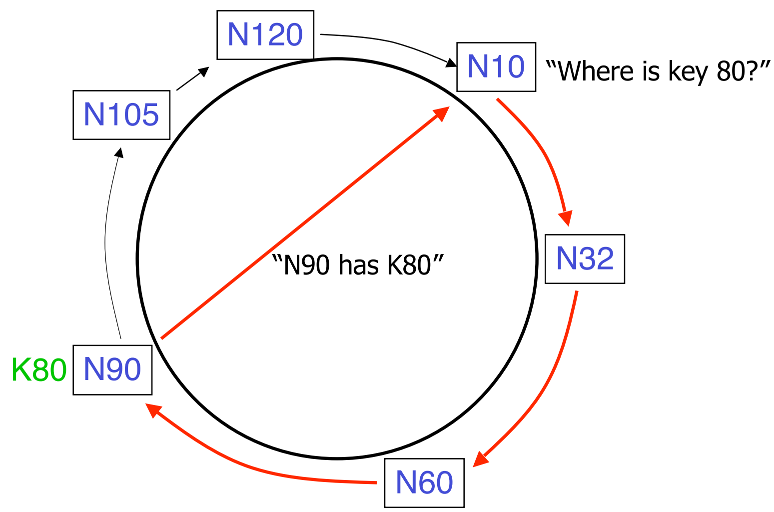
49

Consistent Hashing [Karger 97]



50

Basic lookup



51

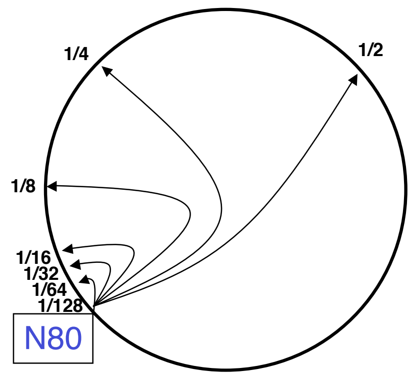
Simple lookup algorithm

```
Lookup(my-id, key-id)
   n = my successor
   if my-id < n < key-id
        call Lookup(id) on node n   // next hop
   else
        return my successor          // done
```

- Correctness depends only on successors

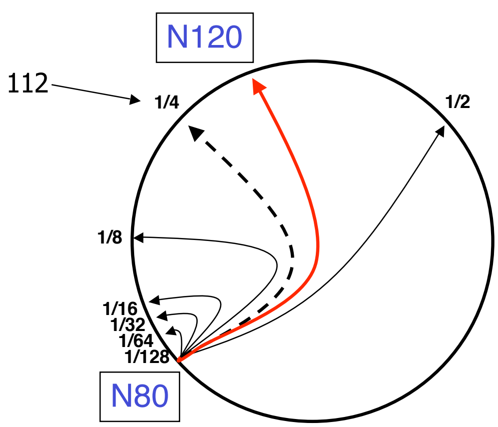
52

"Finger table" allows log(N)-time lookups



53

Finger *i* points to successor of $n+2^{i-1}$



54

Lookup with fingers

Lookup(my-id, key-id)

look in local finger table for

highest node n s.t. my-id < n < key-id

if n exists

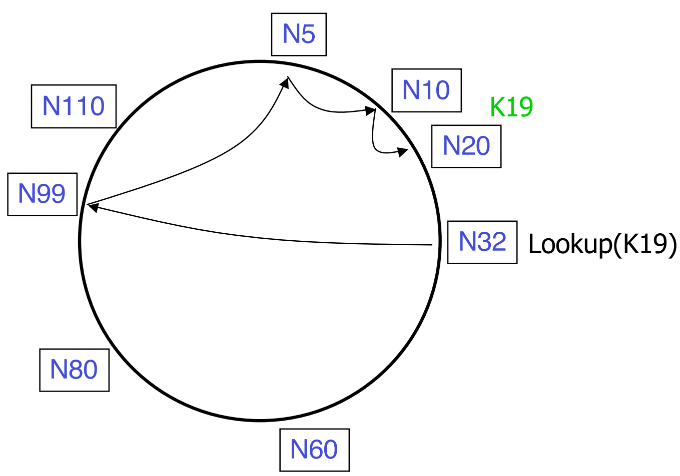
call Lookup(id) on node n *// next hop*

else

return my successor *// done*

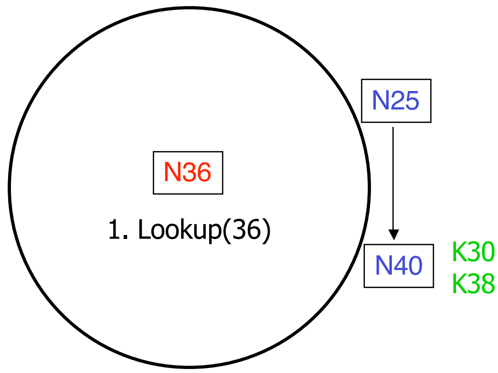
55

Lookups take O(*log(N)*) hops



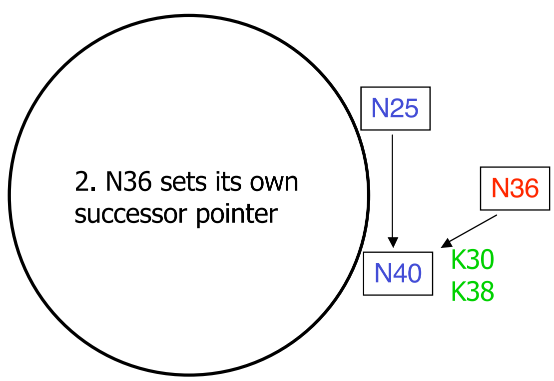
56

Node Join - Linked List Insert



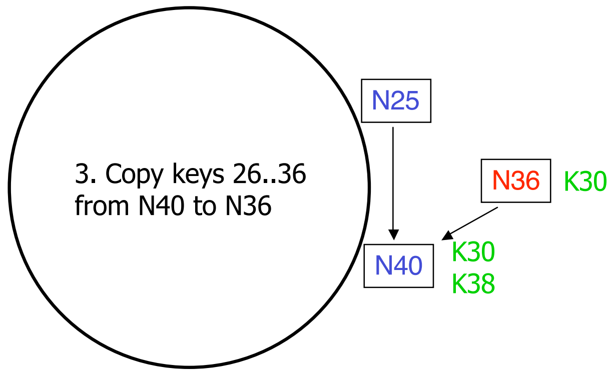
57

Node Join (2)



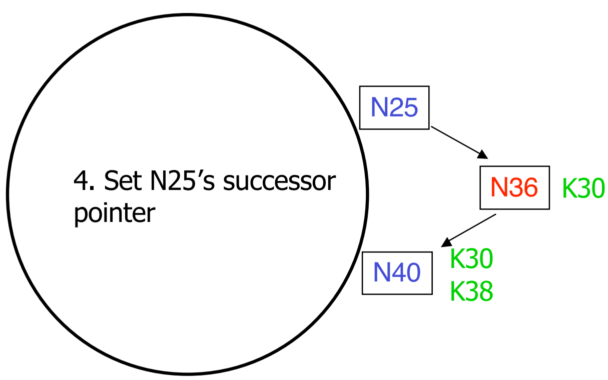
58

Node Join (3)



59

Node Join (4)



Update finger pointers in the background
Correct successors produce correct lookups

60

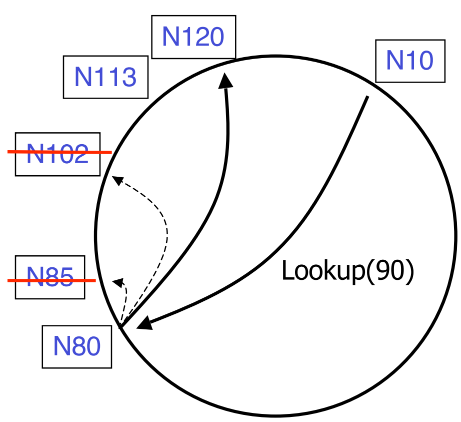
Stabilization

- Case 1: finger tables are reasonably fresh
- Case 2: successor pointers are correct; fingers are inaccurate
- Case 3: successor pointers are inaccurate or key migration is incomplete

- Stabilization algorithm periodically verifies and refreshes node knowledge
  - Successor pointers
  - Predecessor pointers
  - Finger tables

61

Failures and Replication



N80 doesn't know correct successor, so incorrect lookup

62

Solution: successor lists

- Each node knows r immediate successors
- After failure, will know first live successor
- Correct successors guarantee correct lookups

- Guarantee is with some probability

Choosing the successor list length

- Assume 1/2 of nodes fail
- P(successor list all dead) = $(1/2)^r$
  - I.e. P(this node breaks the Chord ring)
  - Depends on independent failure
- P(no broken nodes) = $(1 - (1/2)^r)^N$
  - $r = 2log(N)$ makes prob. = $1 - 1/N$

Chord status

- Working implementation as part of CFS
- Chord library: 3,000 lines of C++
- Deployed in small Internet testbed
- Includes:
  - Correct concurrent join/fail
  - Proximity-based routing for low delay
  - Load control for heterogeneous nodes
  - Resistance to spoofed node IDs

65

Chord Summary

- Chord provides peer-to-peer hash lookup
- Efficient: $O(\log(n))$ messages per lookup
- Robust as nodes fail and join
- Good primitive for peer-to-peer systems

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

66

Readings

- P2P Survey Article on Class web page
- Article on Chord

67