CHORD: A SCALABLE PEER-TO-PEER LOOKUP SERVICE FOR INTERNET APPLICATIONS

Mema Roussopoulou

*Slides based in part on Sylvia Ratnasamy's talk slides.

What kind of paper is this?

- □ A New big idea?
- □ A Measurement paper?
- □ An Experiences/Lessons Learnt paper?
- □ A System Description?
- □ A Performance Study?
- □ A Refute-Conventional-Wisdom paper?
- □ A Survey paper?

Chord’s Intentions

- Given a key, Chord maps the key to a node
- Each node should maintain information for a few nodes, O(logN)
- It tends to balance the load by distributing roughly evenly keys to nodes
- Involves little movement of keys when nodes join or leave the system, O($\log^2$N)

Chord DOs and DONOTs

- DOs
  - Storage load balance
    - spread keys over nodes evenly
  - Decentralization
    - fully distributed, no single point of failure
  - Scalability
    - Chord lookup grows logarithmically in the number of nodes
  - Availability
    - adjusts tables when nodes join/leave
  - Flexible naming
    - no constraints on naming

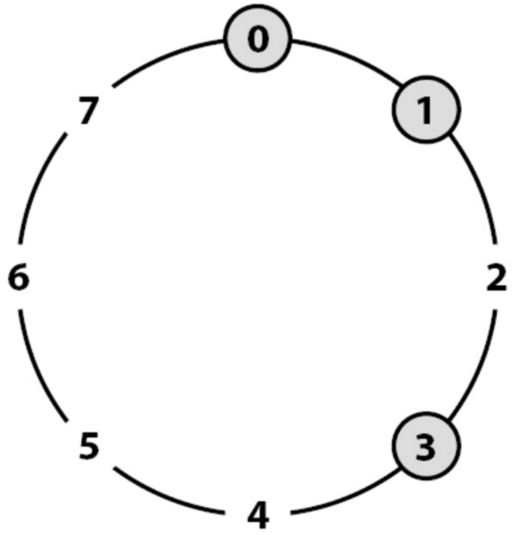
Chord DOs and DONOTs

- DONOTs
  - Authentication
  - Caching
  - Replication
  - Naming of data

Suggested applications?

Chord Base Protocol (1/2)

- Keys are ordered binary numbers of length m
- Nodes are also assigned a random ID in the same number space
- Nodes are ordered in a circle according to their IDs
- For a given key k the responsible node n is the one with the smallest id larger than k, also called successor(k)



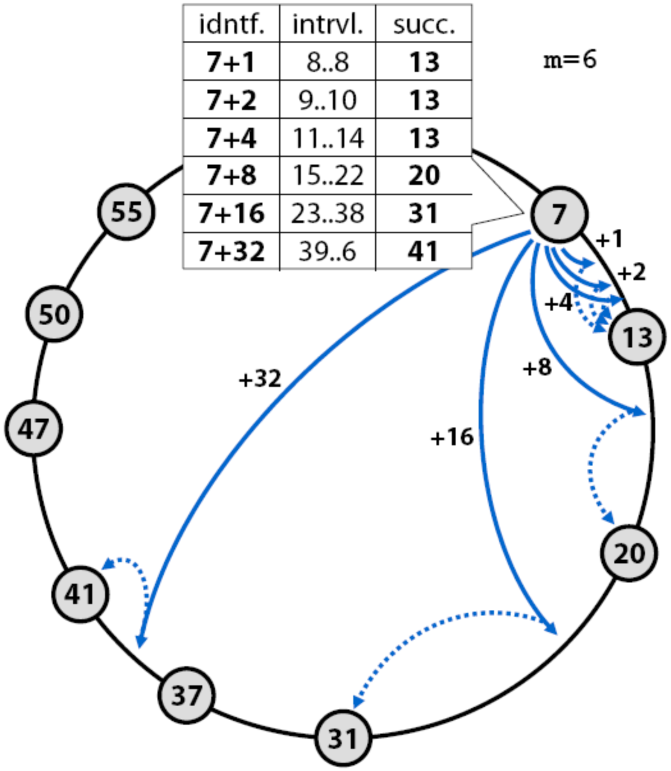
- successor(0) = 0
- successor(1) = 1
- successor(2) = 3
- successor(3) = 3
- successor(4..7) = 0

Chord Base Protocol (2/2)

- Each node holds a pointer to its next node based on node ID order

- To speed things up, each node n has a finger table where its i^{th} entry contains successor($(n + 2^{i-1}) \bmod 2^m$)

- Thus finger table contains O(logn) entries



idntf.	intrvl.	succ.
7+1	8..8	13
7+2	9..10	13
7+4	11..14	13
7+8	15..22	20
7+16	23..38	31
7+32	39..6	41

Questions

- How does lookup work?
- What is the single piece of information that must be correct for lookup to work?
- What must happen on a join?

Node join

- Main challenge is to preserve ability to locate all keys
  - each node has a valid pointer to its successor
  - for each k, node successor(k) is responsible for k
  - in order to have fast search, finger table has to be consistent
- Chord algorithm when node n joins
  - initialize predecessor and finger table of n
  - update fingers and predecessors of existing nodes
  - notify application software to transfer state to new node

Concurrent operations/failures

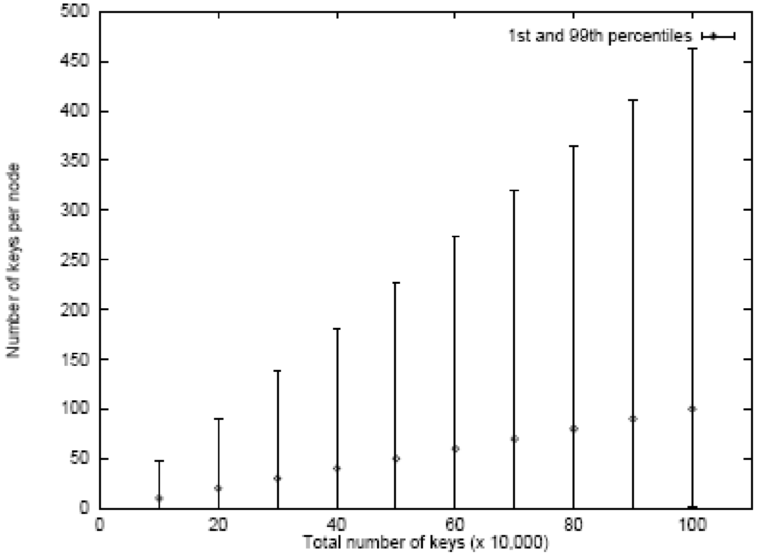
- Previous algorithm does not work in the face of concurrent joins and leaves in the system
  (unless a global locking algorithm is applied)
- Chord prefers to guarantee availability of keys even at a high cost and let the system over time optimize itself to provide fast access to those keys
  - ensure successor links are valid *(correctness)*
  - fingers will converge over time *(performance)*

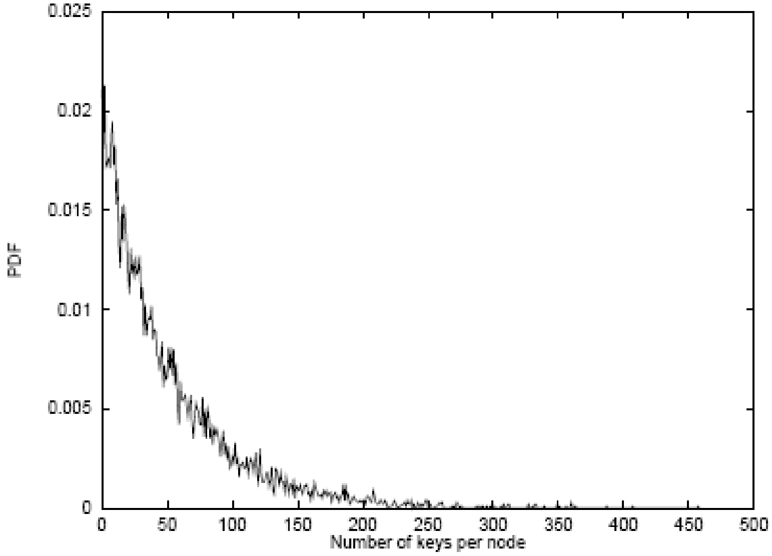
- Algorithm for concurrent operations (Stabilization)
  - when n joins it just locates its successor and updates successor's predecessor
  - nodes periodically validate their successors by asking for their predecessors
  - nodes periodically refresh their finger table

Storage Load Balance

- Number of keys stored per node in (10^4 nodes)
  - ideally distribution of keys to nodes would be K/N

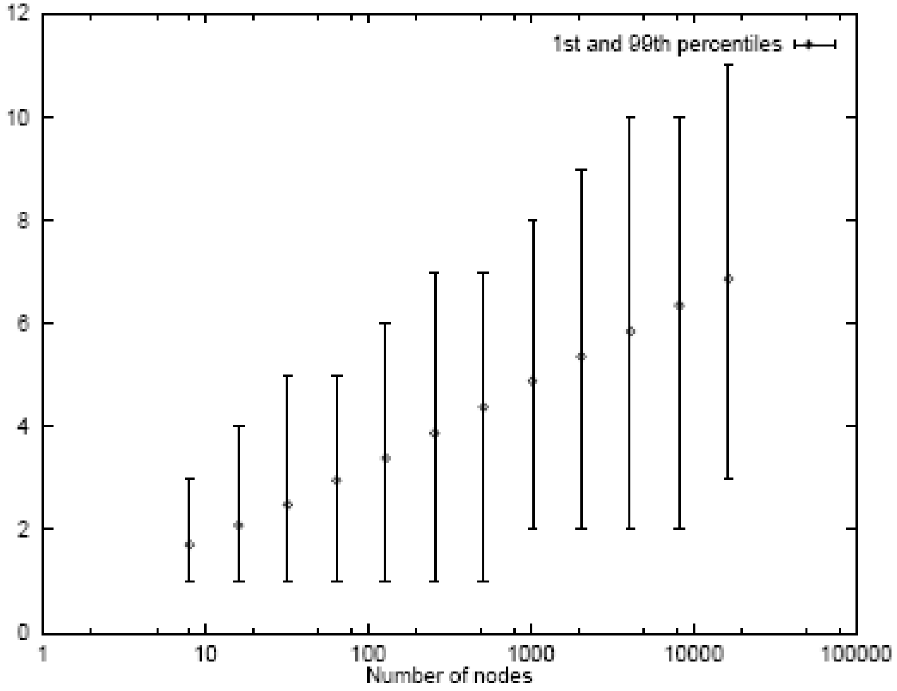
- PDF of number of keys per node (5x10^5 keys, 10^4 nodes)

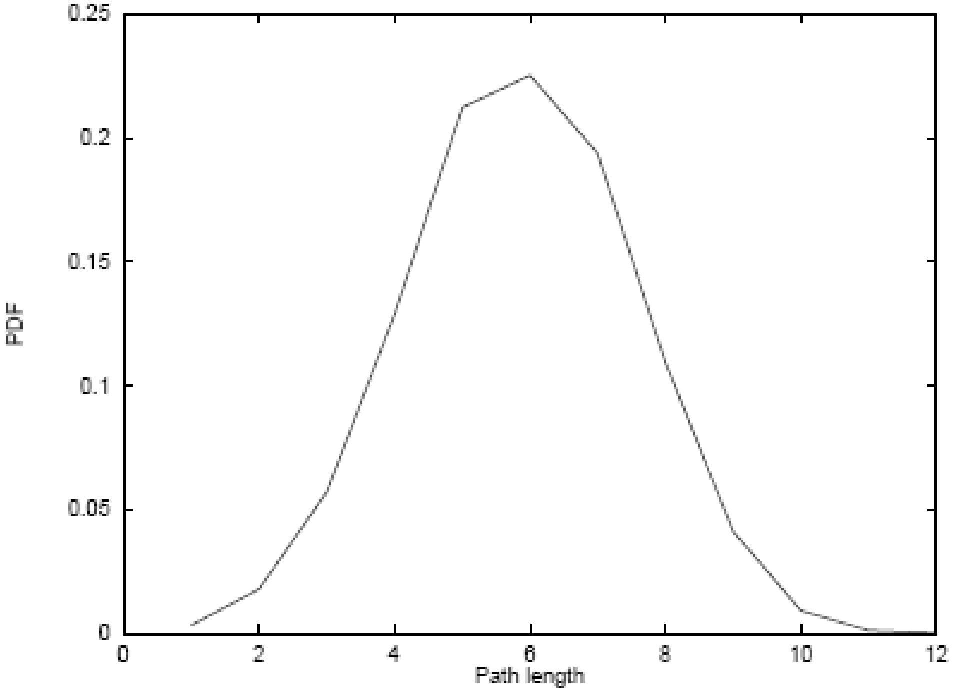




Path length

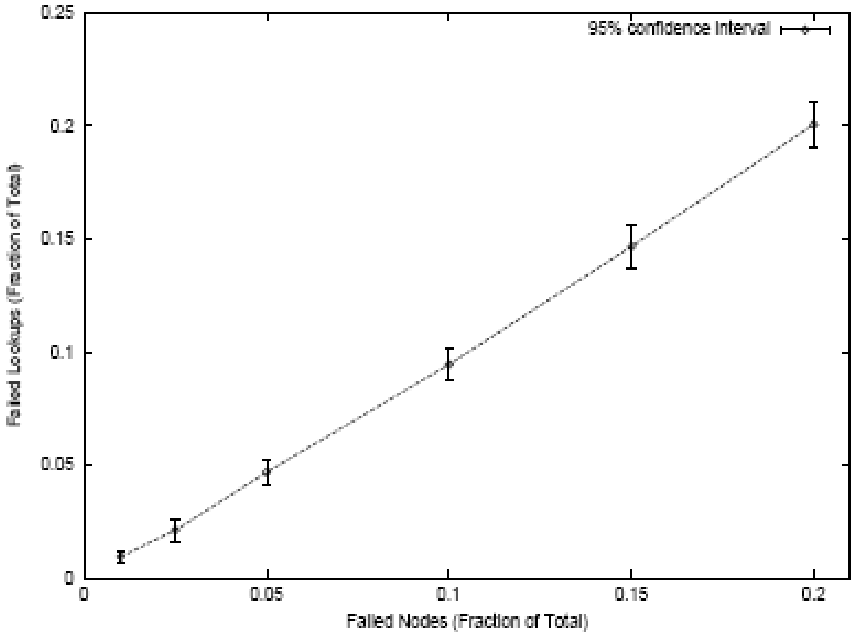
- □ The path length as a function of network size
  - path length is almost ½logN
- □ PDF of path length in a 2^{12} node network

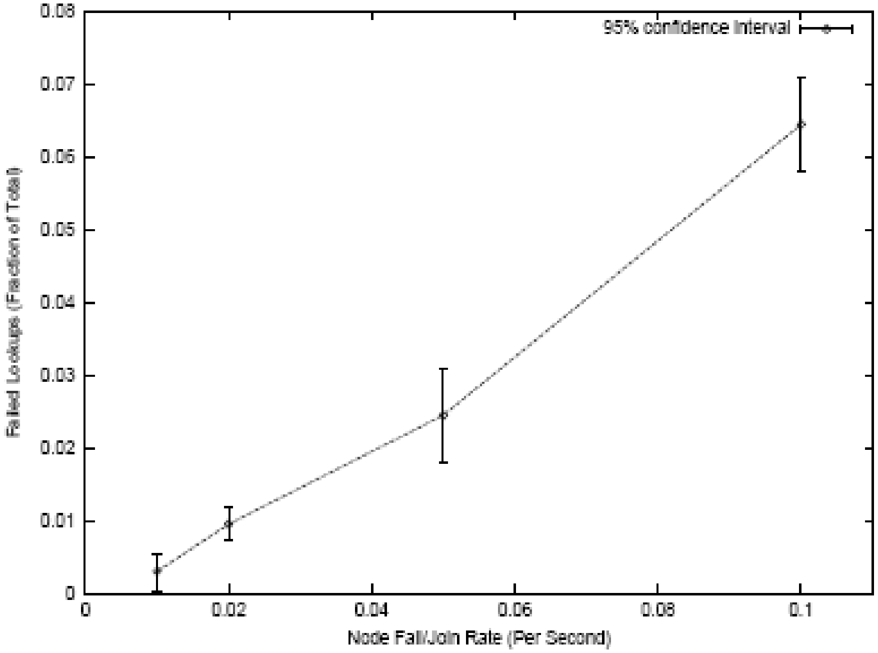




Node failures

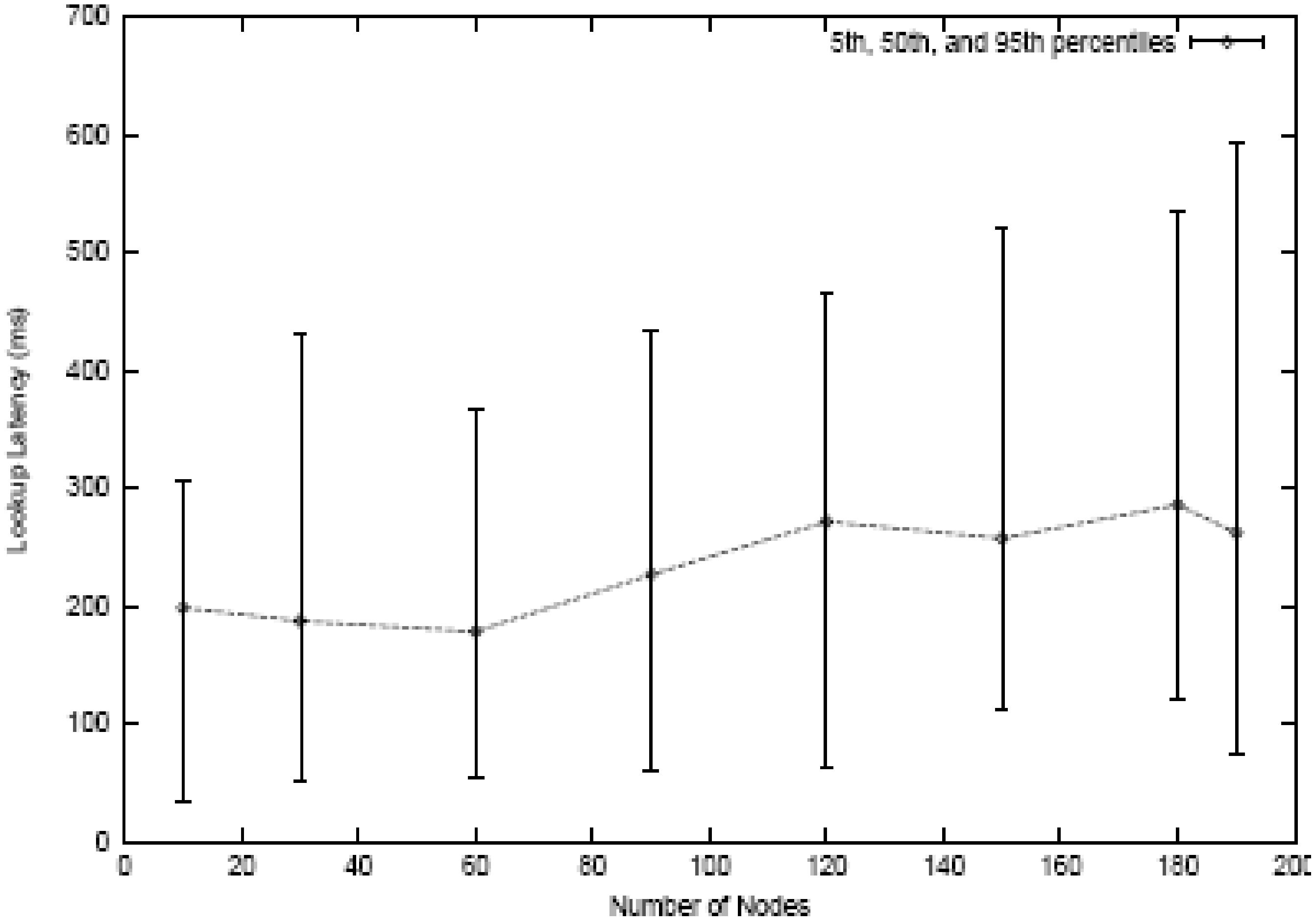
- First case lookups happen after stabilization
  - fraction of failed queries proportional to fraction of lost nodes
- Second case lookups happen during stabilization
  - nodes stabilize every 30 sec
  - Chord’s performance is sensitive to the frequency of node joins and leaves versus stabilize frequency
  - only failures due to chord inconsistency are considered, not failures due to lost keys





Lookup Latency

- **Experiment over internet hosts**
  - only 10 hosts
  - experiments run with virtual nodes on the 10 physical hosts



Future Work?

Future Work

- Suggested future directions:
  - heal partition rings
  - address consistency attack
  - address deny attack
  - reducing hops
  - RTT combined with recursive style of execution