POCS Recitation: Chord

Katerina Argyraki

School of Computer & Communication Sciences

Problem

Design a scalable decentralized system that maps keys to nodes, under significant node churn.

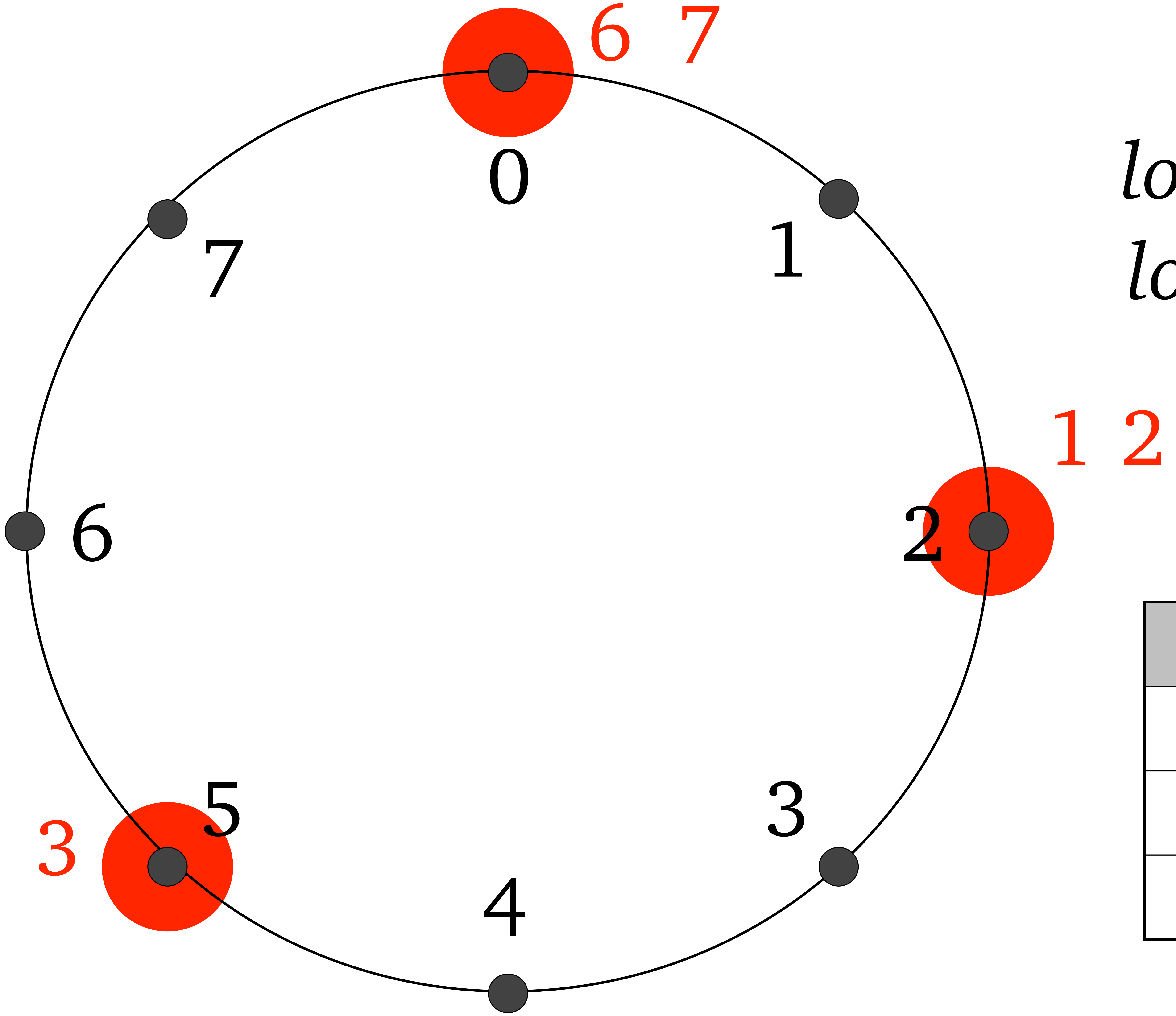
Solution

- Consistent hashing assigns to each node and each key a flat m-bit ID
  - *IDs organized in a circle from 0 to 2^m-1*
- Key k assigned to the first node n>=k
  - *successor(k)*
- Correctness: each node knows the successor of its own ID
- Performance: each node knows the successors of up to m IDs
  - *fingers*

App/Chord Interface

- downcall: app calls lookup(key), gets IP address of node resp. for key
- upcall: Chord notifies app about key changes

Lookup Implementation



lookup(key 3)
lookup(key 7)

key (start)	node (succ)
3	5
4	5
6	0

Scalability

- Per-node state (memory): up to m entries
- Nodes contacted during lookup (latency): O(logN) [Theorem 2]

Joins

- New node n must know an active node n*
- Asks n* to find n's successor

- Each node periodically "stabilizes" =
  updates its own successor + its successor's predecessor

- Each node periodically updates its finger table

- $O(\log^2 N)$

Scalability

- Per-node state (memory): up to m entries
- Nodes contacted during lookup (latency): $O(\log N)$ [Theorem 2]
- Messages exchanged on join (bandwidth): $O(\log^2 N)$ [Theorem 3]

Failures

- For correctness: each node maintains list of r nearest successors
- For performance: if a contacted node does not respond,
  contact the nodes preceding it in the finger table (or one of the successors)

Fault tolerance

- Theorems 7 and 8

Interesting points

- Scalability is quantified through formal bounds
- Separation of correctness from performance concerns
- Relaxation of correctness guarantees to handle churn