DISTRIBUTED HASH TABLES

Soumya Basu

November 5, 2015

CS 6410

OVERVIEW

- Why DHTs?
- Chord
- Dynamo

PEER TO PEER

- What guarantees does IP provide?
- What features do you get?
- What happens if you want more?
  - Overlay networks!

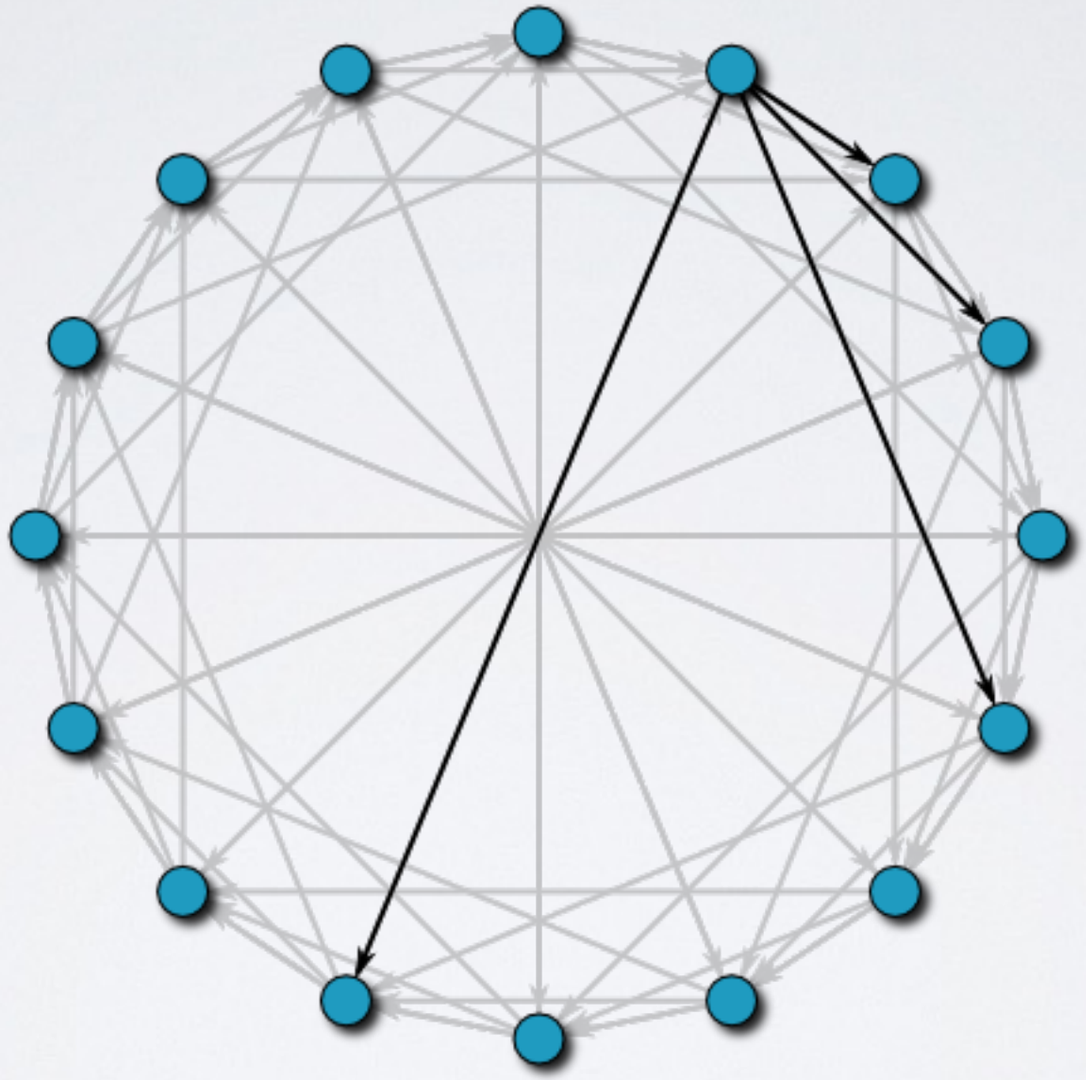
CHORD PROTOCOL

- Intended as another building block
- Supports one operation:
  - Mapping keys to nodes

FEATURES OF CHORD

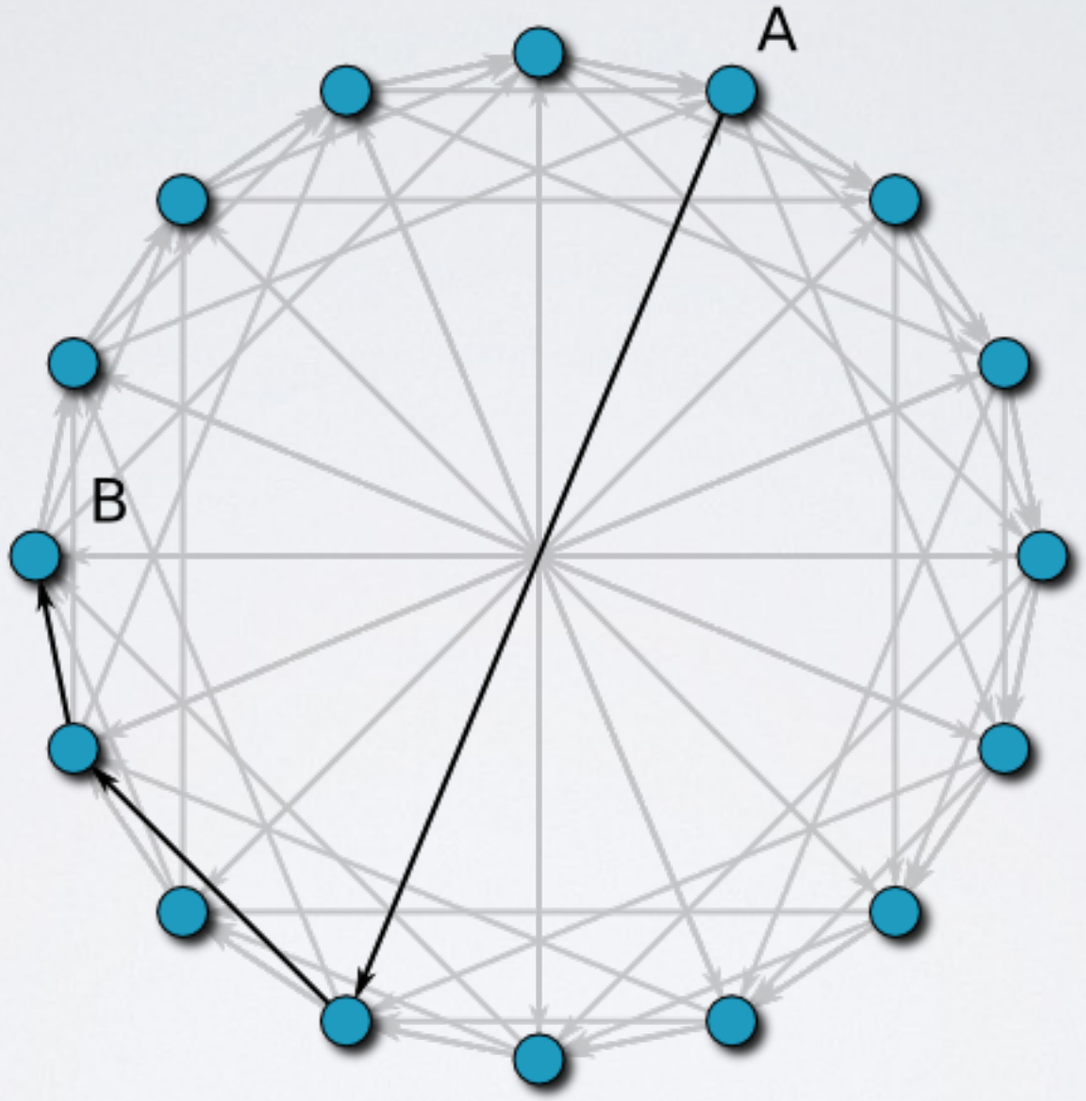
- Scalability
- Provable correctness and performance
  - O(log(N)) lookups
- Simplicity

HOW CHORD WORKS



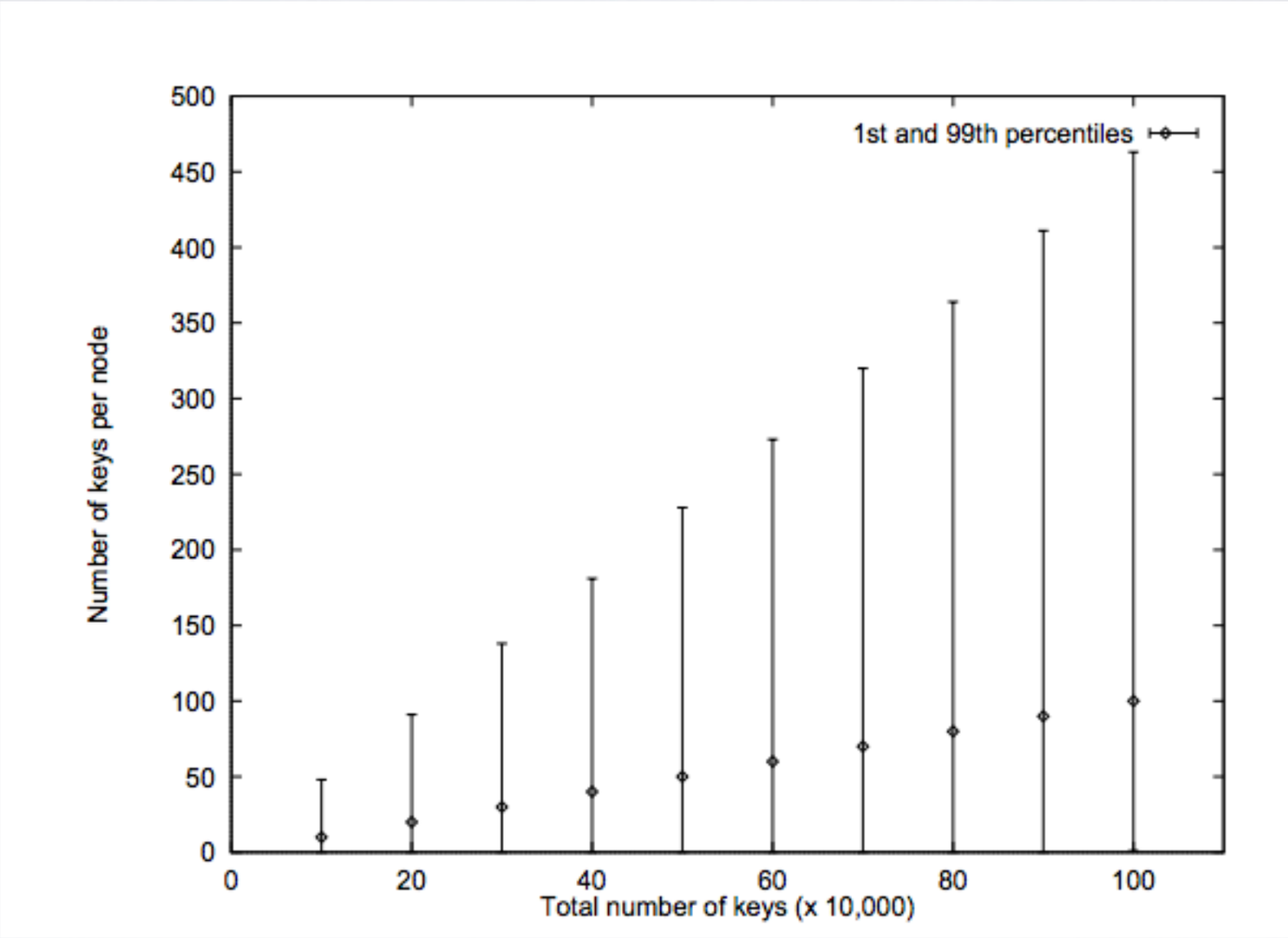
Finger Table for a node

HOW CHORD WORKS

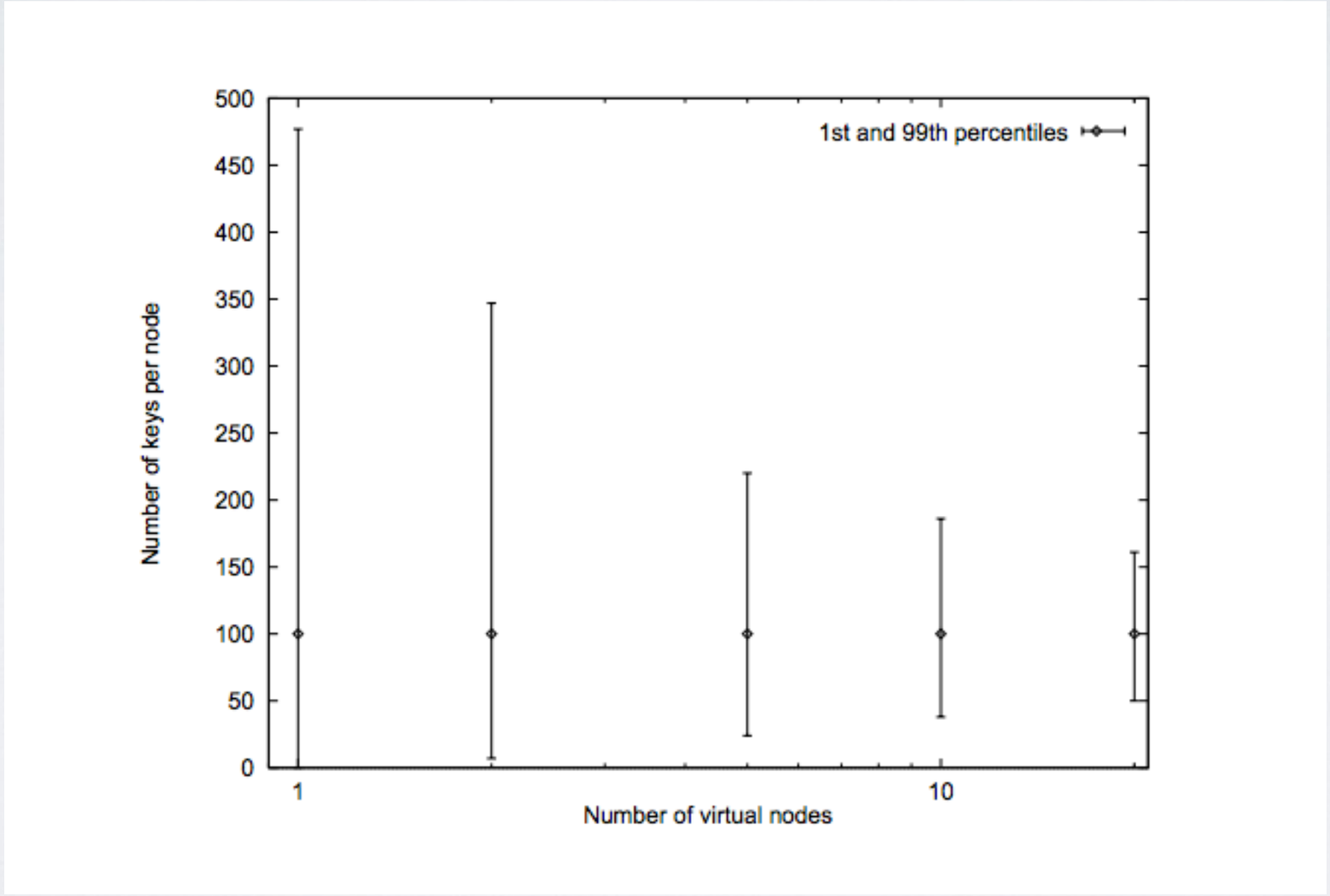


How routing works

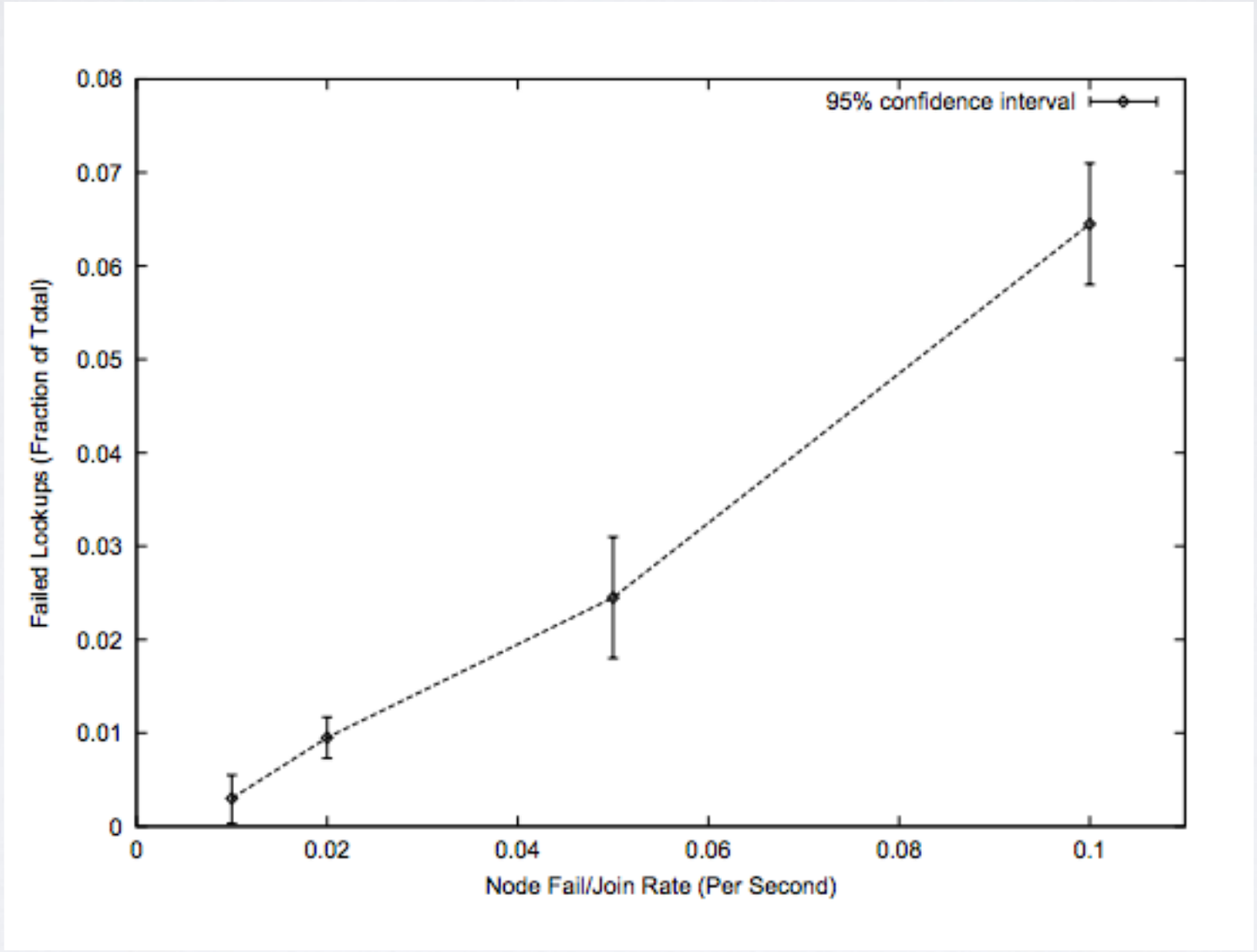
UNFAIR LOADS



LOAD BALANCING



FAULT TOLERANCE



IMPACT

- Distributed Hash Tables were a hot topic!
  - Chord: 12193* citations
  - Pastry: 9606* citations
  - CAN: 9010* citations

*According to Google Scholar

DISCUSSION

- Why was this so impactful?
- What limitations are there to Chord? Is it easy to overcome? Why/why not?

DYNAMO

- Another distributed hash table
- Similar structure to Chord
  - Ring
  - Only supports get() and put()
- Follows the CAP theorem (no strong consistency)

STRICT PERFORMANCE

- Service level agreements in 99.9th percentile
  - Availability
  - Latency
- Explicitly don't care about averages!

FAULT TOLERANCE

- Nodes fail all the time
  - Keys can't be lost
- Solution: replicate keys for next N successors

REPLICATION

- Sloppy quorum
  - Each nodes maintains a "preference list" of replicas
  - Requests are made on first N healthy nodes
    - Need R nodes to respond for read
    - Need W nodes to respond for write

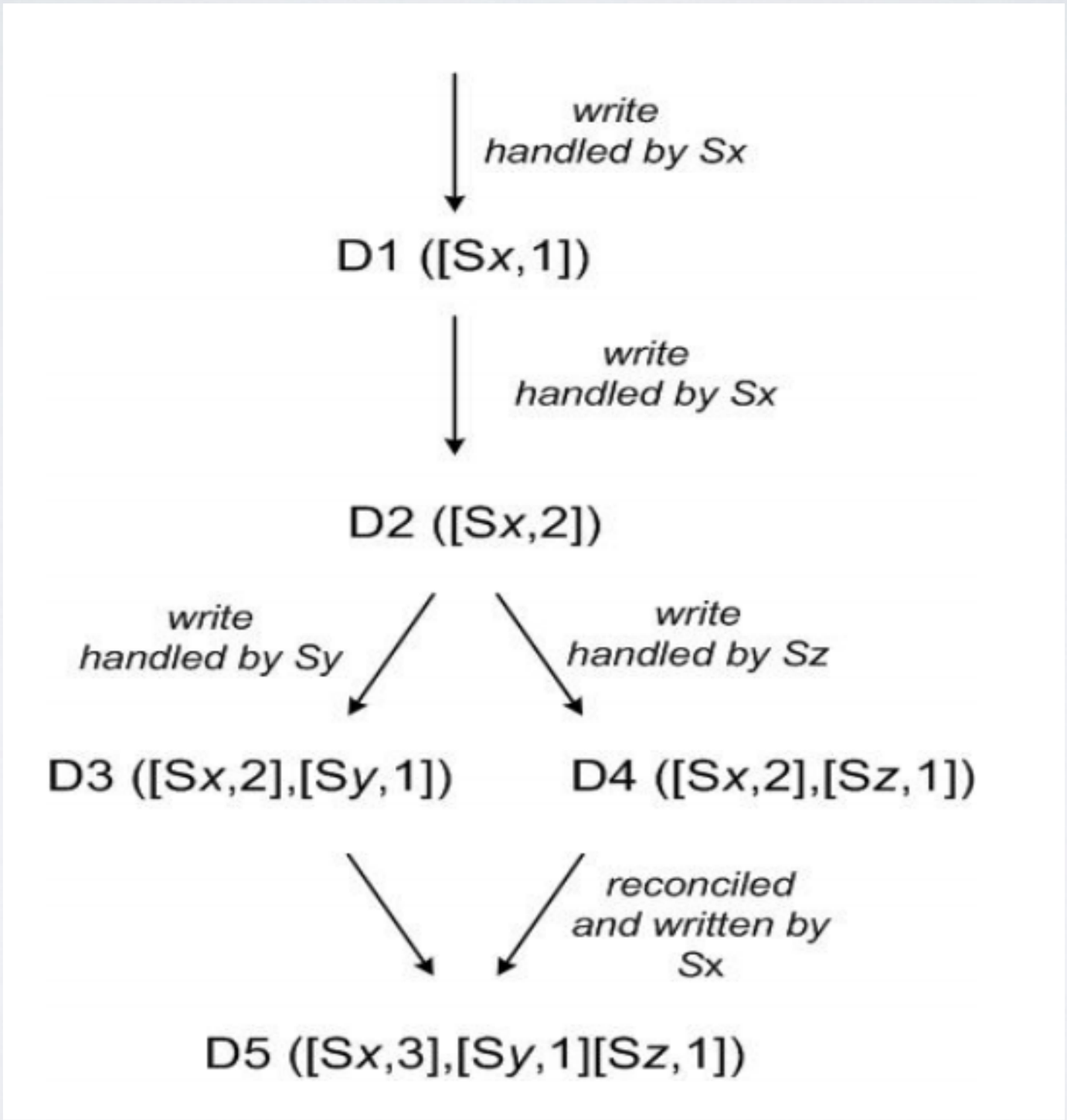
REPLICATION

- Sloppy quorum
  - Developers can tune R, N and W
- Hinted handoff
  - If node is down, periodically check for recovery
  - Include "hint" declaring original replica for key

CONSISTENCY

- Replication leads to consistency problems
- Most systems resolve conflicts on writes
- Amazon needs high write throughput
  - e.g. adding to a cart
- Gives up on consistent reads: "eventual consistency"

HANDLING CONFLICTS



PERFORMANCE