Distributed Hash Tables

Carol Wang

CMU

October 29, 2010

Looking Up Data In P2P Systems

- Centralised and hierarchical lookup rely too much on certain nodes for correctness and reliability.
- Distributed hash tables allow each node to be equally important, generally with $O(\log N)$ lookup.

Chord: A Scalable P2P Lookup Service for Internet Applications

- Chord maps keys onto nodes (can assume to be values) in a network.
- Good performance guarantees whp or wrt standard hardness assumptions.
- Nodes maintain a 'finger table' which allows each forward to cut the search space in half.

Issues from discussion on the survey paper:

- User accountability: is it desirable?
- Taking network proximity into account when routing (addressed with the Chord paper as well)
- Necessary to go to the extreme of all nodes being equal?

User accountability:

- This depends a lot on what we mean exactly by this, but the paper describes trying to protect against malicious participants.
- Of course, one must strike a balance between turning away users either by the overhead of authentication or over-paranoid criteria and protecting the users already in the system.

All nodes being equal:

- The authors make a convincing case that a centralised lookup protocol is prone to failure
- If a node is for some reason known to be more reliable or somehow preferable, though, it would make sense to possibly allow that node to carry a larger load.

Issues from the Chord paper:

- Different topologies (addressed in optional reading)
- Formal proof
- Assigning node ids not completely at random.

Formal proof

- The paper gives a number of formal results about the expected performance of Chord
- It is not that unified, and they don't address the issue of partitioning very well

Assigning node ids not completely at random

- While we are guaranteed to have $O(\log n)$ lookups, it may be that the paths the lookups follow are suboptimal from a network perspective
- The proofs for Chord correctness seem to depend heavily on using random assignments
- Maybe we can eliminate this by relaxing the requirements; perhaps nodes at certain distances can be allowed to be slightly correlated, etc.

Optional reading: The Impact of DHT Routing Geometry on Resilience and Proximity

The authors discuss the "flexibility" of various routing geometries. This is measured using the number of choices for routes and for neighbors. We have

topo	flexible neighbors	flexible routes
tree (Pastry, etc)	yes	no
hypercube (CAN)	no	yes
butterfly (Viceroy)	no	no
ring (Chord)	yes	yes

Notes: Butterfly is inflexible, but more efficient. Also, Chord doesn't nominally allow flexible neighbors, but the ring topology admits it.