February 15, 2021

Reading Summary: Kademlia: A P2P Information System Based on the XOR Metric
Alexander McRae, V00890516

1 Please describe the problem(s) in your own words. Is the problem important at the time of paper publication, and how about now? Why?

This paper addresses 3 main problems:

1. Current systems require explicit messages to gain knowledge of peers
2. Current systems lack flexibility of routing which can lead to high latency request
3. Current systems do not permit parallel requests for improved latency

At the time of publication, hyperscale computing was coming to age and the problems the paper identifies are key to success at scale. In modern computing DHTs are used less than expected but some of the ideas seen is this paper are prevalent today. (ie. CassandraDB)

2 Please describe the main idea(s) in your own words. How is the idea different from the existing work at the time of paper publication? How does the idea impact the follow-on work till now?

One of the main ideas is using XOR for a distance function between two keys, this has the property that two keys can never be the same distance apart from another key. Another idea from the paper is to use k-buckets for routing. K buckets contain k entries between $[2^i, 2^{i+1}]$ for i in [0, 160). This creates a cover of the keyspace without any overlap and allows lookups in O(LogN) like Chord and other systems. The benefit of this system over chord is that whereas Chord only stores a single preceding node for an interval $[2^i, 2^{i+1}]$ Kademlia allows queries within an interval. Another main idea is to query multiple close-by nodes at once to avoid high latency routes. It can do this because it doesn't just store a single preceding node like Chord. Lastly, all RPC pass IDs with them such that they can be put into buckets.

3 Please list at least three most important things in this paper. Why do you think they were important at the time of paper publication? How about now?

K buckets for dividing key space at the time of the paper would have been an improvement over Chord and Pastry. In modern times we could probably allow K to become large and get even more benefits.

Parallel queries available due to k buckets both at the time of the paper and now are incredibly important to keeping latency low.

Including configuration with messages allows routing tables to be kept up to date and full. Both then and now this is important as we expect systems to scale extremely well.

4 Please list at least three things you think may need further improvement in this paper. Has the improvement appeared in the follow-on work already?

I felt the paper didn't express the ideas well and lacked both pseudocode and examples. Luckily there are online resources that clear things up.

Glossed is how Kademlia handles nodes leaving and joining at realistic rates (https://static.usenix.org/publications/library/proceedings/usenix04/tech/general/rhea/rhea_html/usenix-cr.html)

Although it is mentioned briefly security isn't discussed heavily in the paper. I have not found much analysis on the topic.

5 Do you have some ideas of your own on this problem? Can you do something better or differently? How can you show that?

February 15, 2021

Kademlia makes no guarantees about <k,v> persistence and correctness which is often a good property to have. Using the number of nodes in the system and real-world data it may be possible to make some guarantees about the reliability and persistence of the data. It is not clear to me from the paper that the results returned for a value lookup are required to be up to date or correct at all.