Name: Prashanth Mallyampatti
Unity ID: pmallya
Student ID: 200250501

CSC724- Advanced Distributed Systems
Paper Review

Chord: A Scalable Peer-to-peer Lookup service for Internet Applications

Summary:

This paper discusses on how to easily scale a distributed hash table for service lookups. Earlier algorithms required each node to maintain a list of all other active nodes to route requests. Even though the lookups in this case is very quick but updating whenever a node leaves or joins is costly. To address this the authors present Chord a distributed hash table algorithm that uses consistent hashing. This algorithm aims for scalability, load balancing, availability, flexibility, minimizing the number of hops and provides correctness and guarantees of the system.

Chord uses consistent hashing which adapts to a continuously changing environment to map every node and data to an m bit identifier using SHA1. Identifiers are arranged in a circle and data item maps to a node whose id is equal or is close to the next id. Each node maintains routing information called the finger table about nodes 1,2,4,… hops away from it. Thus, when a key is given for a lookup, it refers to this table which reduces the amount of each state required at each node to O (log N) hops. This accounts for scalability and faster lookups of nodes.

Also, nodes follow a stabilization protocol which keeps the finger table correct and consistent during failures. In addition, they maintain a successor list which has the information about a set of successors in the circle. During node failures these tables are updated by looking at the node's predecessors, and successors (like a linked list). Chord adjust its tables to reflect the newly joined nodes as well as left/failed nodes, ensuring that, a node can always be found id there's key associated with it. Thereby handling all the vulnerabilities in the system and making the system always available. In Chord no node is more important than the other thus a nature of decentralization is achieved. The above are some of the main points discussed in the paper.

Strong Points:

1) It is highly efficient for real time systems which require efficient data lookups as their primary requirement, which is predictable in fixed time span with certainty of definite failure or success.
2) In addition to minimized number of hops, Chord also provides tolerance to failures and transparent remapping of keys to nodes, which can be used in global lookup services.
3) Implications of naming conventions, load imbalance, hierarchy, and super nodes have been overcome by Chord.

Weak Points:

1) The first weak point I found out is that Chord doesn't discuss (or provide) any authentication policies for purpose of security.
2) Chord is not extensible for a wide range of feature list like local peer selection which increases the overhead when provided.

3) The network bandwidth or the latency have not been discussed or considered.
4) It is left to the user's system for maintaining the replication procedures.