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

Alessandro Cacco
mat. 203345
alessandro.cacco@studenti.unitn.it

Andrea Ferigo
mat. 207486
andrea.ferigo@studenti.unitn.it

Enrico Zardini
mat. 207465
enrico.zardini@studenti.unitn.it

1 Introduction

This work aims at illustrating an implementation of *Chord*, a scalable distributed lookup protocol described in [1]. Basically, *Chord* provides a primitive, i.e. *lookup*, that allows to determine the responsible of a key in an efficient way. Hence, it represents a great solution to the data location problem: each data item needs just to be associated to a key and stored in the node the key maps to.
In practice, the nodes are logically arranged in a ring topology and each of them is responsible of the ids belonging to the interval $(predecessorId, nodeId]$ [1]. The key-data pairs are assigned to nodes based on the hash value of the key and consistent hashing is used in order to keep the load balanced. Moreover, each node is required to maintain information about only a few other nodes: the predecessor, some successors and the elements of the finger table[2]. Therefore, *Chord* scales well to large numbers of nodes without affecting performance. Actually, it adapts effectively also in dynamic environments with frequent joins and leaves thanks to a simple stabilization algorithm.
Finally, it is worth mentioning that the iterative version of *Chord* has been implemented. Hence, a node resolving a lookup initiates all the communications needed to reach the target.
Starting from this, Section 2 will describe in detail the implementation, Section 3 will present the graphical simulator that has been developed to show the protocol's functioning and Section 4 will describe the simulations that have been performed and the results obtained.

2 Implementation

This section presents the methods and the behaviour of the five classes that have been used to implement the protocol. In particular, these classes can be subdivided into three groups:

- the control class (`TopologyBuilder`) that instantiates the nodes, initialises the ring, manages joins/leaves and initiates the lookups;

[1] All the arithmetic is modulo 2^m, where m is the number of bits of the identifiers.

[2] The finger table is a routing table: the i^{th} entry points at the responsible of the identifier $nodeId + 2^{i-1}$.

- the class (`Node`) that defines the behaviour of the agents in the simulation, i.e. the nodes;
- the support classes (`FingerTable`, `Lookup` and `Utils`).

2.1 TopologyBuilder

3 Simulator

4 Analyses and Results

References

[1] I. Stoica, R. Morris, D. Liben-Nowell, D. Karger, F. Kaashoek, F. Dabek and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup protocol for internet applications", *IEEE Transactions on Networking*, vol. 11, Feb. 2003. DOI: 10.1109/TNET.2002.808407.