Initial Report. DHT Peer-to-Peer Project

Denis Grebennicov

May 14, 2020

1 Team information

- team number: dht-4
- team member: Denis Grebennicov
- project: DHT (Chord)

2 Programming language and operating system

- Programming Language: Scala (with Akka Framework).
- Operating System: OS X / Linux.

Scala is used due to various reasons:

- Together with the Akka framework (based on actor model) makes it easy to implement distributed applications
- Actor model has several characteristics which are advantageous for this project:
  - Since actors communicate via async message passing, it's asynchronous by default
  - Location transparency - meaning actor doesn't really care if the target actor is local or remote one. This eases the development and testing, since there is no need for spawning multiple jvm processes for multiple peer communication.
  - In order to test a node/peer all we have to do is send a message and assert the received one. This makes testing easy.

Since Scala project run everywhere, where JVM runs, platform, is not really a problem. My operation system of choice is OS X, the development will be done on it, but it shouldn't have any impact on the project itself.

3 Build system used

sbt - Scala build tool.

4 Quality management

Quality management consists of following measures:

1. Unit and integration tests via Akka TestKit and ScalaTest
2. Implementation of a centralized monitoring web-service (with web frontend) for introspection of the state of the system/peers. E.g. checking nodes successor lists, its predecessor, finger table and/or stored keys.

5 Available libraries

Akka framework (including actor/http/stream) for various parts of the system:

- peer actor
- web frontend

6 Assigned license

MIT license. As its copyright says: "Hey, it's free, no legal restrictions, why not try it out?". But GNU GPL could work as well. I am pretty open to any license.

7 Previous programming experience of your team members

Last year I was attending this course and started implementing the project (also Chord DHT). However, due to the fact that I finished my bachelors and failed my masters application at TUM, I was exmatriculated and therefore couldn't and didn't finish the project. Long story short, I continue developing a system I started implementing a year ago for the Peer-to-Peer lecture.

8 Planned workload distribution

- implement routing module where peer routing table consists of just a successor list and a predecessor node
    - spawn actors within one jvm process for easier testing
- implement monitor service for visualization and testing purposes
- implement chord stabilization protocol
- implement storage module with key-value getting/setting functionality
- support one peer - one jvm process and implement the remote actor communication
- implement TCP server which serves as a public API for the DHT module/project
- load config file and use it for bootstrapping and dht module setup