Initial Report - Team 13 - DHT Module

Dimitri Tyan, Lars Schwegmann

23. May 2019

1 Team Composition

Our team consists of Dimitri Tyan and Lars Schwegmann. We both study Informatics and are in the first semester of our Masters degree program.

2 Project Choice & Approach

We chose the DHT (Distributed Hash Table) module of the voidphone as our project. Furthermore, we decided that we want to look at an existing implementation (2018-dht-rust), understand it's P2P protocol and implement it in a different language, namely Swift 5[1].

3 Existing DHT Module analysis

We want our Swift-based implementation to be fully compatible with its Rust-based counterpart which already exists. The Rust implementation uses Chord[2] for its DHT, so we will be implementing the DHT according to the Chord protocol as well. Based on the code of the Rust implementation provided in the repository and the documentation, there are limitations that can be worked on and added as features to our Swift implementation. Specifically, the Rust implementation does not support handling in case of unexpected behavior of a node such as when a node leaves the P2P network.

4 Operating System

For development we are using macOS 10.14.5. Due to the fact that our build system, Swift Package Manager[3], supports Linux, it is also possible to execute the DHT module on devices that run Linux.

[1] `https://swift.org/`
[2] `https://de.wikipedia.org/wiki/Chord`
[3] `https://swift.org/package-manager/`

5 Build System

Our build system of choice is called Swift Package Manager. A package defines one or more targets and can contain dependencies that are used by the package. That way it is possible to add modules that may be needed for the implementation of the DHT module. We chose this build system since it is the official build tool for Swift applications. Swift Package Manager currently supports macOS and Linux.

6 Quality Measures

Using Swift Package Manager, it is possible to execute unit tests. We also plan on using the given testing repository to ensure interoperability with the other components of the voidphone application as well as the existing Rust implementation.

7 Libraries

We plan on using SwiftNIO[4], a library developed by a dedicated team at Apple for non blocking IO in order to make our DHT implementation fast and reliable. The library is used in a major web framework for Swift (Vapor[5]) and supported by Apple, so it reasonable to assume that it is well maintained and suitable as a stable library.

Currently, we do not plan on using any other libraries. Should we determine that additional third party libraries are required during the development of our implementation we will carefully review them in order to keep dependencies and third party code minimal.

8 Workload Distribution between Team Members

The first step will be to analyze and understand the existing implementation in depth in order to determine all the components of the system which need to be implemented by us and agree on the interfaces between these components. This gives us the possibility to work on the components separately, as we both have different schedules and can not meet up regularly.

[4] `https://github.com/apple/swift-nio/`
[5] `https://vapor.codes/`