**Project Architecture**

**Title**

Peer to Peer chatSDK with distributed hash tables

**Student(s)**

Benedikt Kaiser BSc.

**Project Advisor**

Alexander Palmanshofer MSc.

**Date and Time**

Hagenberg, 07.07.2023

Version: 03/2023

Table of Contents

[1. Required Modules 3](#_Toc128577851)

[2. System Architecture 3](#_Toc128577852)

[3. Software Architecture 3](#_Toc128577853)

[4. Infrastructural Aspects 3](#_Toc128577854)

# Required Modules

To fulfill the requirements outlined in the specification document for the development of a decentralized Chat SDK for iOS using distributed hash tables (DHT), several logical modules or units need to be implemented. These modules can be categorized as follows:

1. DHT Implementation:
   * DHT Data Structure: Design and implement the data structure to store and manage the distributed hash table.
   * Key-Value Store: Create mechanisms to store and retrieve key-value pairs within the DHT.
   * Routing Mechanism: Develop algorithms for efficient routing and lookup of IP addresses within the DHT.
   * Peer Discovery: Implement mechanisms to discover and connect with other peers in the network.
2. Communication and Messaging:
   * Peer-to-Peer Communication: Establish direct communication channels between peers for message exchange.
   * Message Encryption: Incorporate encryption techniques to ensure secure transmission and storage of messages.
   * Message Queueing: Develop mechanisms to handle message queuing and delivery in case of offline recipients.
3. User Management and Authentication:
   * User Registration: Design a system for users to register within the decentralized network, generating unique public-private key pairs.
   * Authentication and Authorization: Implement mechanisms to authenticate users and authorize their access to the network.
4. Swift Package Manager Integration:
   * Package Configuration: Prepare the Chat SDK to be compatible with the Swift Package Manager (SPM) for easy integration into new or existing iOS applications.
   * Dependency Management: Specify external libraries and dependencies required by the Chat SDK within the SPM manifest file.

External functionalities and libraries that can be utilized to fulfill the requirements include:

* Cryptographic Libraries: Incorporate established cryptographic libraries like CryptoKit or OpenSSL to handle encryption and digital signatures.
* Networking Libraries: Utilize networking libraries such as Alamofire or URLSession for peer discovery and communication.
* DHT Protocol: Leverage existing DHT protocols like Kademlia as a reference to implement the necessary functionality.

The interdependencies between these functionalities include:

* The DHT Implementation relies on the cryptographic libraries for secure communication and storage.
* Peer Discovery and Networking functionalities depend on the DHT Implementation to locate and connect with other peers.
* User Management and Authentication functionalities interact with the DHT Implementation and Networking components to establish secure communication channels and validate user identities.

# System Architecture

This chapter covers a small installation guide and how to use the current software.

It is possible to get the XCode project from the gitlab repository. Once pulled, you can start XCode and build the app.

Now start at least two simulators and deploy the app to both simulators.

A screen shot of a phone

Description automatically generated

This is the only screen of the application. On this screen you can select your port and start listening on it.

Now the second simulator instance is able to connect to the first instance by writing the corresponding port into the correct textfield.

You can select an endpoint you want to trigger, in the screenshot above ping is selected, and press connect&send.

Now a udp package gets send to the first simulator, which responds with the answer to the request.

This answer is currently only mocked and will be valid as soon as kademlia is implemented.