**Project Architecture**

**Title**

Peer to Peer chatSDK with distributed hash tables

**Student(s)**

Benedikt Kaiser BSc.

**Project Advisor**

Erik Sonnleitner FH-Prof. DI Dr.

**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

The proof-of-concept implementation for our distributed ChatSDK on iOS, utilizing LibP2P, realizes several key functionalities essential for decentralized communication within a local network. These functionalities include:

1. **Chat Functionality:** The primary feature of the ChatSDK is to enable users to engage in real-time messaging within a local network environment. This includes sending and receiving text messages, multimedia content, and other forms of data supported by the application.
2. **Network Address Translation (NAT) Handling:** The implementation incorporates mechanisms to handle NAT traversal, allowing devices behind NAT devices to establish direct communication channels. This feature is crucial for enabling peer-to-peer communication within private networks where NAT is commonly encountered.
3. **Peer Discovery:** The ChatSDK implements peer discovery mechanisms to locate and connect with other devices running the application within the same local network. This involves identifying available peers and establishing connections to facilitate messaging.

Within the ChatSDK implementation, various internal interfaces are required to facilitate communication between different modules and components. These interfaces include:

* **Messaging Interface:** Facilitates the exchange of messages between users, handling the sending, receiving, and processing of messages within the application.
* **Network Interface:** Manages network communication tasks such as peer discovery, connection establishment, and data transmission/reception over the local network.
* **Encryption Interface:** Provides functions for encrypting and decrypting message content, ensuring secure communication between peers.
* **Autonat Interface:** This interface manages the automatic NAT traversal functionality within the ChatSDK. It handles the detection of NAT devices and dynamically establishes traversal mechanisms, such as hole punching, to enable direct peer-to-peer connections.
* **Circuit Relay Interface:** Responsible for coordinating relayed communication between peers when direct connections cannot be established due to network restrictions or NAT traversal failures. The interface manages the setup and maintenance of relayed circuits through intermediate nodes to facilitate communication between peers.

The intended artefacts collaborate with external components and interfaces to fulfill their functionalities :

* LibP2P Library: The ChatSDK leverages the LibP2P library to implement peer-to-peer communication capabilities, including network abstraction, peer discovery, and secure messaging.

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:

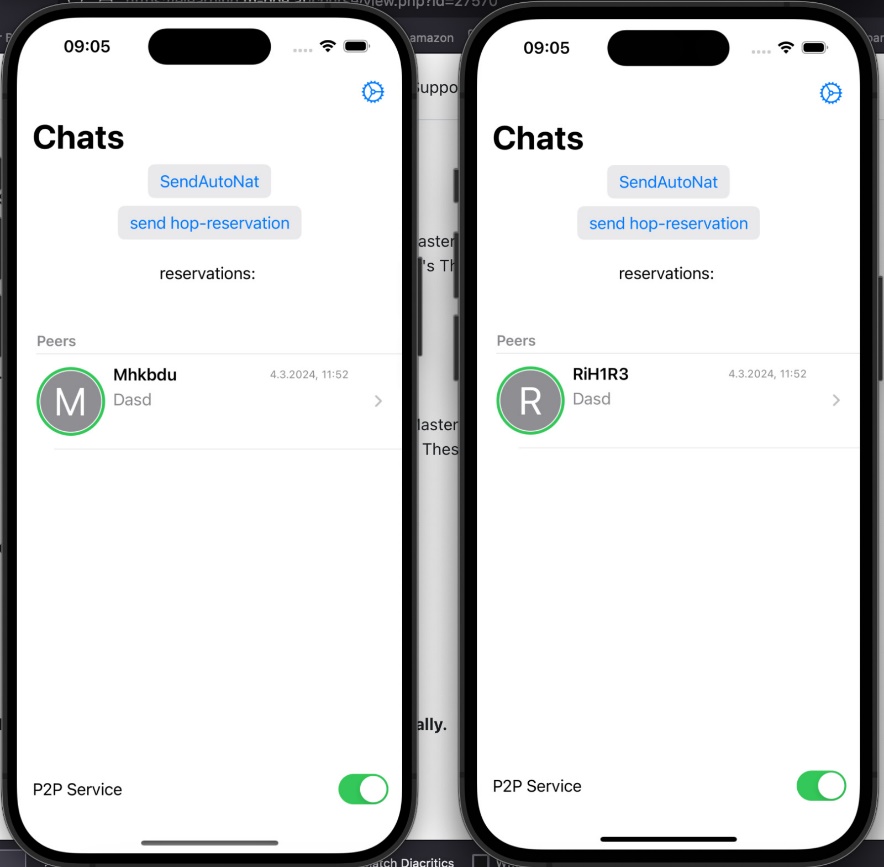
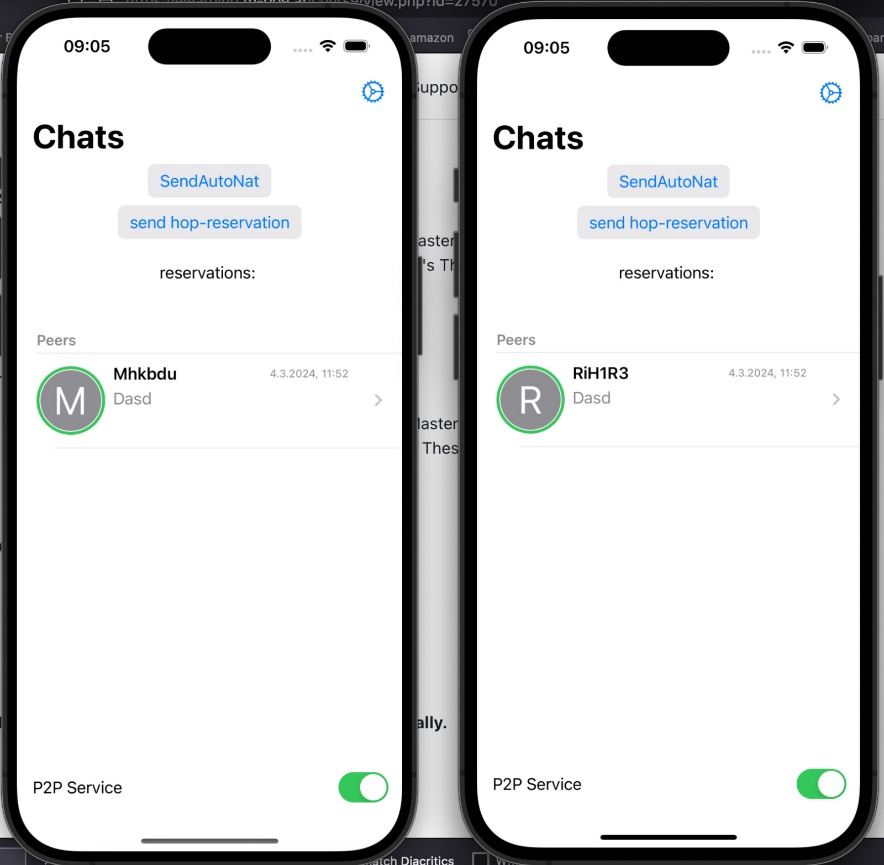
* The DHT Implementation relies on the cryptographic libraries for secure communication and storage.
* Peer Discovery and Networking functionalities depend on the DHT Implementation to locate and connect with other peers.
* User Management and Authentication functionalities interact with the DHT Implementation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

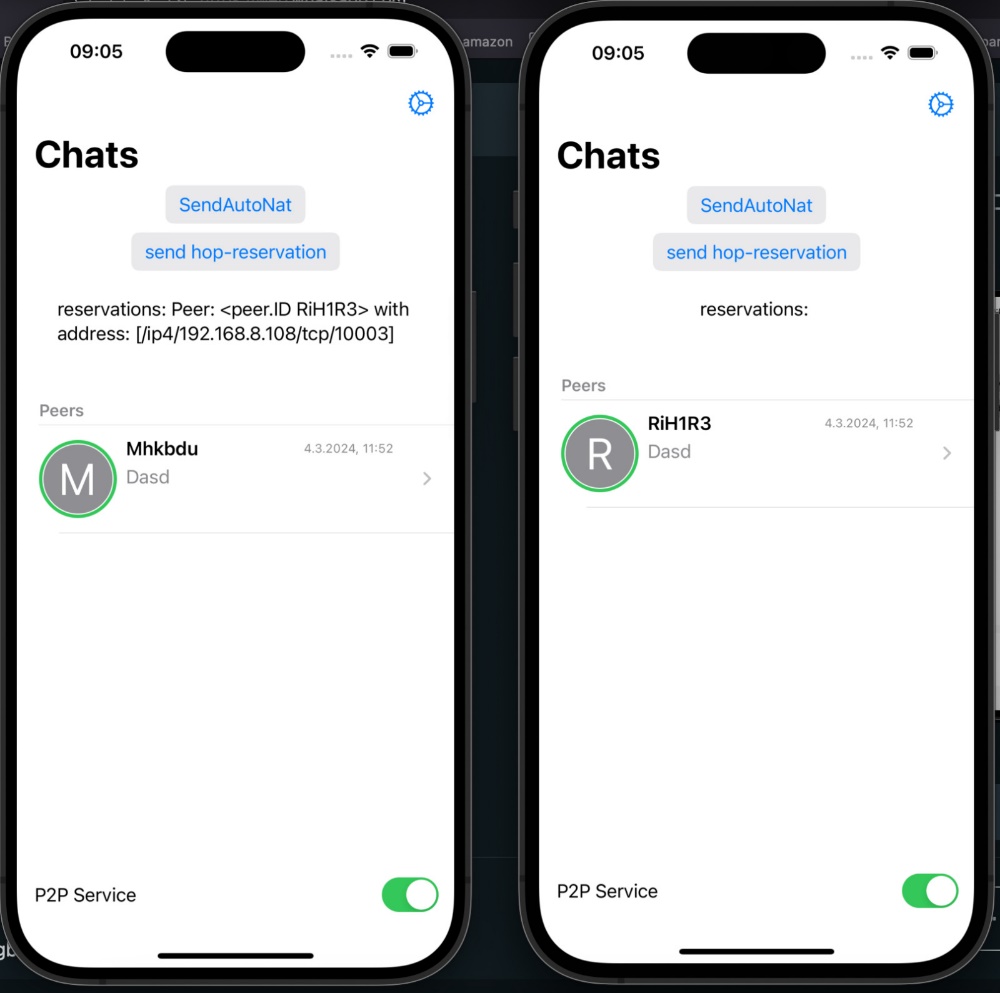
Now start at least two simulators and deploy the app to both simulators.



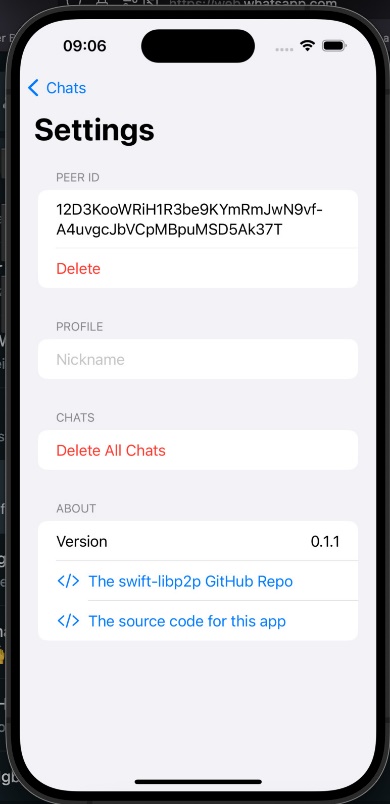
As soon as you enable the P2P Service, the two nodes find each other and the profile pictures get a green border as an indicator that they are connected. Now it is possible to press on the chat and get to the chat screen:



This is the chat screen, here it is possible to just exchange messages.



It is also possible to make reservations at the other node. In this case the phone on the right pressed on the “send hop-reservation” button and the phone on the left stored the peer id and the address of the phone. This connection is going to be open until the phone on the right closes it again.



In the settings screen it is possible to set a nickname and to see your peerID