Chair of Network Architectures and Services
Department of Informatics
Technical University of Munich

Peer-to-Peer Systems and Security

Team number:	52
Team members:	Roland Bernhard Reif, Manfred Stoiber
Advisor:	Richard von Seck, Filip Rezabek, Jonas Jelten
Supervisor:	Prof. Dr.-Ing. Georg Carle
Begin:	04/2021
End:	09/2021

Midterm Report

1. Changes to Assumptions in Initial Report

The main technical assumptions of our initial report still remain and are represented in our progress. Due to personal developments (change of study subject, slipped disk) we are a bit behind our intended schedule (e.g. we only did rudimentary testing of our current implementation). But we are fully dedicated to catch up in August and to fulfill all assumptions in our initial report.

In addition to the mentioned libraries in the initial report, we also use the following ones: .encoding/binary, .bytes, and .math/bits.

2. Architecture

We have separated our implementation into two main modules, namely the API communication and the peer-to-peer communication. The API communication listens for newly to be established connections and listens on these connections for new orders for our distributed hash-table. These orders can be of type dhtPUT and dhtGET. As an answer, a dhtSUCCESS and dhtFAILURE message can be received.

The peer-to-peer module currently consists of three submodules. The first one handles the k-buckets (an implementation specific datastructure) e.g. populates the k-buckets and updates them. The second submodule is responsible for parsing the messages we use for peer-to-peer communication.

The last submodule (P2PCommunication) is responsible for everything else related to Kademlia. It dispatches incoming messages, handles connections and is responsible for node lookups and finding and storing key-value pairs.

In our main function, first the configuration file is parsed and the provided parameters are used to configure certain implementation specific parameters (e.g. the k in k-Buckets). Afterwards the API module starts listening and parsing requests. This is done concurrently with goroutines. For handling the requests, the peer-to-peer module is used.

For ensuring availability and hence security, we store key-value pairs multiple times and the Kademlia protocol that we use has built in resilience against node-poisoning attacks. Further security measures (encryption, authentication) are not yet implemented but will be part of our tasks in the upcoming weeks.

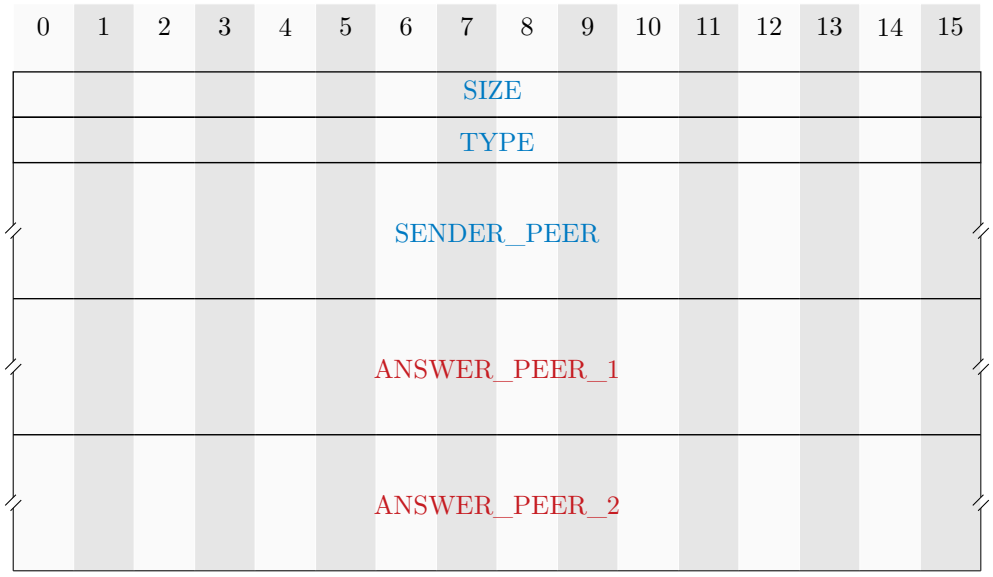

Figure 1: Structure of a KDM_FIND_NODE_ANSWER for an α of 2.

3. Peer-to-Peer Protocol

As a peer-to-peer protocol, we use our own version of a Kademlia [1] protocol as it is one of the securest distributed hash table implementations.

Up to now, our implementation uses 4 kinds of message exchanges:

Check liveliness	KDM_PING, KDM_PONG
Storing key-value pairs	KDM_STORE
Retrieving key-value pairs	KDM_FIND_VALUE, KDM_FIND_VALUE_ANSWER
Locating responsible nodes	KDM_FIND_NODE, KDM_FIND_NODE_ANSWER

Our peer-to-peer messages in general consist of a message type and a message size and a field to identify the sending peer. Depending on the message type, the payload (if existing) varies. In a KDM_FIND_NODE_ANSWER e.g. the payload consist of the peer IDs, IPs and ports of the α (a configuration parameter) closest nodes to a specified key/ID that was searched for. An example for an α of 2 can be seen in Figure 1.

These messages are needed to fulfill our requirements (e.g. storing and retrieving data) or to ensure security/availability (e.g. with ping-pong messages). Basic error handling and sanity checking for messages and specified parameters is implemented but will be extended.

4. Future Work

For some messages the parsing and sending of them is not fully implemented yet. This will be our first priority. As a second priority we want to focus on testing our implementation. Thirdly, we want to build in more security measures to ensure confidentiality, availability and integrity of our data. We will also try to audit these security measures. Lastly, we will write the documentation and final report.

5.-6. Workload Distribution and Effort Spent

Manfred Stoiber

Manfred has focused on the implementation of the peer representation, the implementation of the peer-to-peer message dispatcher and the ping-pong liveliness check. His estimated time spent is about 30 hours.

Roland Reif

Roland has implemented the API communication, the parsing of the configuration file, the k-buckets sub-module and the functions responsible for node lookups and parsing the respective messages (including e.g. creating the byte representation of peers). His estimated time spent is about 50 hours.

References

[1] P. Maymounkov and D. Mazières, "Kademlia: A Peer-to-Peer Information System Based on the XOR Metric," in Peer-to-Peer Systems, P. Druschel, F. Kaashoek, and A. Rowstron, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 53–65.