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:	08/2021

Initial Report

1. General

Team name:	Peer-Pioneers
Team members:	Manfred Stoiber, Roland Bernhard Reif
Module:	Distributed Hash Table
Implementation:	Kademlia[1]

2. Programming Language

Programming language:	Go
Operating system:	Linux

Our main reason for choosing Go[1] is its outstanding support for concurrency, e.g. with the help of so-called "Goroutines". This can become handy for managing multiple DHT requests. Furthermore, Go is quite lightweight, offers an easy way for error handling, and enforces a simple, readable coding style. The language also utilizes hardware in an efficient way (e.g., with automated garbage collection / memory management). We especially like Go's support for scalable networking services. Go runs on both Windows and Linux, which is convenient for us.

We both use Windows machines for developing the module, as we use it in our daily work. However, we want to build the application for Linux. We will use virtual machines, the Windows Subsystem for Linux[2] and our remote access to the LRZ for running and testing our code in Linux environments.

3. Build Systems

With the command "go run" Go automatically compiles and executes our code and runs the resulting executable. We plan to make use of a make file to resolve potential dependencies and execute the aforementioned command. Alternatively we could generate a binary executable by using the command "go build", which then would not require dependencies on the target system.

[1] https://golang.org/
[2] https://docs.microsoft.com/de-de/windows/wsl/

4. Quality Assertion

Go in itself already enforces a neat coding style for example by forcing the developer to avoid unused variables. We will of course extensively use the provided testing module for testing the interaction between instances of our module with the provided interface specifications. To ensure software functionality and function verification we will be using the builtin "testing" package. To avoid software regression and maintain professional source-code, we intend to write unit tests reaching at least 90% statement coverage.

For further quality improvement we intend to use tools for static code analysis developed by the Go community. One of them is the golangci-lint project[3]. This is a tool for automatically running a huge number of Go linters in parallel. We also intend to implement an interface for centralized logging mechanisms. This will also include different levels of detail.

5. Available Libraries

We intend to use the following libraries:

Networking	.net[4]
Binary Conversions	.encoding/binary[5]
Cryptography	.crypto[6]
Flags	.flag
Logging	.log
Testing	.testing

6. License

We are leaning towards a permissive license, orientated at the GNU all permissive license. This grants the rights to perform, display, copy, modify and distribute the code under the condition to include the license itself. We also want to include a statement of excluding liability against us as broad as possible under applicable law. We do not intend to monetize our system and feel comfortable with granting other developers the right to use our system. However we do not want to be liable for any negative consequences of the usage of our system.

7. Previous Experience

Manfred Stoiber

Manfred has worked on several smaller software-projects for studies and in a company. Most of them were written in Java and a few in C++ and Python. None of them had to handle topics like networking or protocol design and implementation. Therefore this is his first networking-project and his first project written in Golang.

[3] e.g. https://github.com/golangci/golangci-lint

Roland Reif

Roland has worked on two relevant programming projects so far. Both of them were conducted at the networking Chair of Prof. Carle and both projects were implemented in Golang. The first project was for his Bachelor thesis ("Analysis of EDNS Client-Subnet Loadbalancing"). The second project is his IDP ("Detecting BGP Hijacks in Real-Time"), which he will finish on 2nd of June 2021. He started learning Go approximately 1,5 years ago for his Bachelor thesis.

8. Workload Distribution

We intend to equally distribute the workload and to work together frequently. For example, we want to schedule regular pair-programming sessions. For these sessions we have created a "common" branch in Gitlab. Changes committed in this branch were created and contributed to by both of us equally. Same goes for commits from other branches where the commit message starts with "[common]. In the case of individually assigned subtasks we still want to collaborate closely and advice/support each other. Therefore, we have established frequent communication channels (e.g. WhatsApp).

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.