Initial Report

Andreas Bauer
andi.bauer@tum.de

Chung Hwan Han
hanc@in.tum.de

May 2021

1 Objective

We, Andreas Bauer and Chung Hwan Han, form the team *Gossip-10* in the Peer-to-Peer Systems and Security course. Within the project we implement the Gossip module of the *VoidPhone* Voice over IP system. The system provides anonymity and understandability to voice communication through a peer-to-peer architecture. The Gossip module is used to spread information like availability information among communicating peers. Other modules from other teams might rely on the Gossip module to spread information required for their operations through the network. For this purpose, the project specification defines a socket-layer API for inter-module communication. The protocol spoken between Gossip instances is the core product of this project work. We follow the material and contents taught in the lecture – considering best practices, common attacks and security measures – for the instantiation of our P2P protocol.

2 Tooling and Infrastructure

For our implementation we chose Java 17 LTS[1] as the programming language. Lower level languages such as C/C++ may have better performance, but we identify performance to not be that hard of a requirement for this project as the focus is on the network-layer and not really consisting of many computational intensive tasks. Therefore, we decided to focus on readability, type safety and faster development iterations. After all, while not being a compiled language, Java still delivers incredible performance. While we don't expect any major platform incompaitibilities, we primarmly develop for UNIX-based systems with macOS 12 and Ubuntu 22.04 LTS being the focus.

For our build system we use Gradle. It is a modern and well estbalished build system in the Java community and itegrates with Maven central for fast and easy dependency managment. We aim to use the provided GitLab instance for our development workflow (Issues, PRs and Git workflow). That includes relying on the GitLab CI infrastructure for automatic build and testing pipelines.

We intend to rely on the following libraries:

- **Netty 5**[2] is used an asynchronous and event-driven networking I/O framework. We rely on it as the core for both the API protocol and the P2P protocol implementations.

- **Bouncy Castle**[3] will be used to perform any crypto operations that are part of the developed P2P protocol.

- **ini4j**[4] is used for parsing the INI-style configuration files.

---

[1]https://openjdk.java.net/projects/jdk/17/

[2]https://netty.io

[3]https://bouncycastle.org

[4]http://ini4j.sourceforge.net/index.html

- **JUnit 5**[5] is used as the testing library. It is widely used in the Java development community and integrates natively with the Gradle build system.

This project is licensed under the MIT license. The license text is available in the LICENSE file at the root of the repository.

3 Previous Experiences

We both have a strong background with the Java programming language. Both of us were involed in bigger software projects, building experiences with common project management and development workflow tools and pratices. We provide some references to past work in the following.

I, Chung Hwan Han, have done a project about arp spoofing[6], which is relevant with this project in terms of the network security. Also, I gained experiences in networks and access control while working on the project about simulating CSMA[7]. Please refer my GitHub profile for more information[8].

I, Andreas Bauer, want to highlight my participation in the open-source project Homebridge[9] as one of the primary maintains. Furhter, I gained extensive experiences practicing agile development workflows as part of the Apodini[10] research project. General references and experiences can be taken from by GitHub profile[11].

4 Planned Work Distribution

We identify the following fundamental milestones for our project work:

- **Project Setup**: The initial step in the project is the project setup itself. Part of it is the setup of the Java project and the Gradle build environment. Further, it deals with the configuration of the GitLab CI/CD pipeline with ensures that new code is compliant with code style and unit tests pass as expected. Lastly, we intend to use GitLab features like *Push rules* to enforce our PR-based Git workflow.

- **API**: The implementation of the inter-module API will be the first step in the implementation work. It enables us to build core networking layer which we can later reuse for the P2P protocol implementation. The API-layer itself it properly define within the project specification and therefore a good candiate for the first implementation step. Predefined python scripts to interact with the API server are provided from the start and serve as an initial test vector.

- **P2P Protocol**: The implementation of the core logics of the project.

  - **Session Management**: Establishing and maintaining connections between peers. It should provide security measures such as authentication and encryption.
  - **One-shot Messages**: Messages that are sent once and then discarded. Since some messages of Gossip do not expect any response, we can implement them as one-shot messages.
  - **Notification System**: Implementation of notification protocol that represented on the specification.

---

[5] https://junit.org/junit5
[6] https://github.com/hanchchch/arp-spoof
[7] https://github.com/hanchchch/csma-ca
[8] https://github.com/hanchchch
[9] https://github.com/homebridge/homebridge
[10] https://github.com/Apodini/Apodini
[11] https://github.com/Supereg

- **Tests**: Writing test cases to guatantee the functionalities. Each point would have its own test cases. We will use JUnit 5 for this purpose. Further, we will use the provided GitLab CI/CD pipeline to automatically run the tests and ensure that the tests pass as expected.

- **Documentation & Reports**: The final step in the project is the documentation and writing reports.

We aim for a fair and eqaul workload distribution. We will follow a agile development workflow, staying in continuous contact, realigning on project goals, milestones and current workload distribution. Nonetheless, we want to sketch a rough workload distribution in the context of the above outline milestones. For simplicity reasons, the initial *Project Setup* will be done by a single person. Implementation of the *API* might be split into implementation of the networking-layer and the implementation of the message formats. Work on the *P2P Protocol* is split according to the sub-milestones identified above. We will allow ourselves to assign workload distbution for this milestones dynamically and by need. While *Tests* are mentioned as a explicit milestone, we expect that those are implemented with every incremental development step. Lastly, work invested into wiriting the midterm and final report is split equally.