Peer-to-Peer Systems and Security

Team 23, SS21
Initial Report

Alexander Liebald, Kevin Ploch

April 2021

1 General Team Information

- Team number: 23
- Members: Alexander Liebald, Kevin Ploch
- Module: Gossip

2 Language and OS

We chose Python 3 as it's very portable and works on Linux and Windows operating systems through the python interpreter instead of a binary executable. It's also considered easy to read and offers great support in terms of libraries and documentation. For a non-production ready project like this, the performance downside of Python can be neglected.
Additionally, both of us only have limited experience in Python and wanted to improve our Python skills with this project.

3 Build System

Since we are using Python, we do not require an extra build system.
For the installation, we will use a requirements.txt, which allows pip to install all non-default libraries with just one command.

4 Quality of Software Checks

To guarantee the quality and correctness of our software we plan to make use of unittests in python. These can be included in a Continuous Integration pipeline in gitlab to test code that's in development. For the unittests themselves, we try to cover every possibly malformed input to avoid throwing exceptions. To test the whole system itself we plan to use the provided mockups.

5 Libraries

- **struct**: to parse byte-objects to other datatypes and backwards
- **configparser**: parsing .ini config files
- **asyncio** and/or **threading**: to handle multiple parallel connections
- **argparse**: parse (optional) cmd parameters
- **socket**
- **hexdump**

6 Licensing

We choose the GNU GPLv3 License for our project. The main argument for this license is that the code must always be distributed open-source which allows independent security checks. Besides that GNU GPLv3 requires developers to state changes they make to the original code. This makes it clear what changes were made by whom.

7 Previous Programming Experience

Last year both of us participated in the same team for the computer architecture practical course. We build a simple program to resize images which was written in Assembly with a c wrapper. For that, we also used git. With little experience in c / Assembly, we had no problems working together and achieved a good grade.

8 Planned Workload Distribution

We have already created a general plan for the project in the form of a flow chart (together). See figure 1.
After that, both of us started programming and refining the current plan for a specific module.
We also split up the writing of this report 50/50 and plan to do so in the future.

Currently, we are still working on the first modules and check our progress in regular meetings. For the future, we do not yet have a clear workload distribution. Every time we finished a task, we will check with the other one and assign a new task. Since we had no problems regarding an even workload distribution in the past, we believe this will work. If one of us feels like it doesn't, we will make a more detailed plan for future workload distribution.
Nevertheless, we have already worked out a general plan dividing the project into tasks and sub-goals.

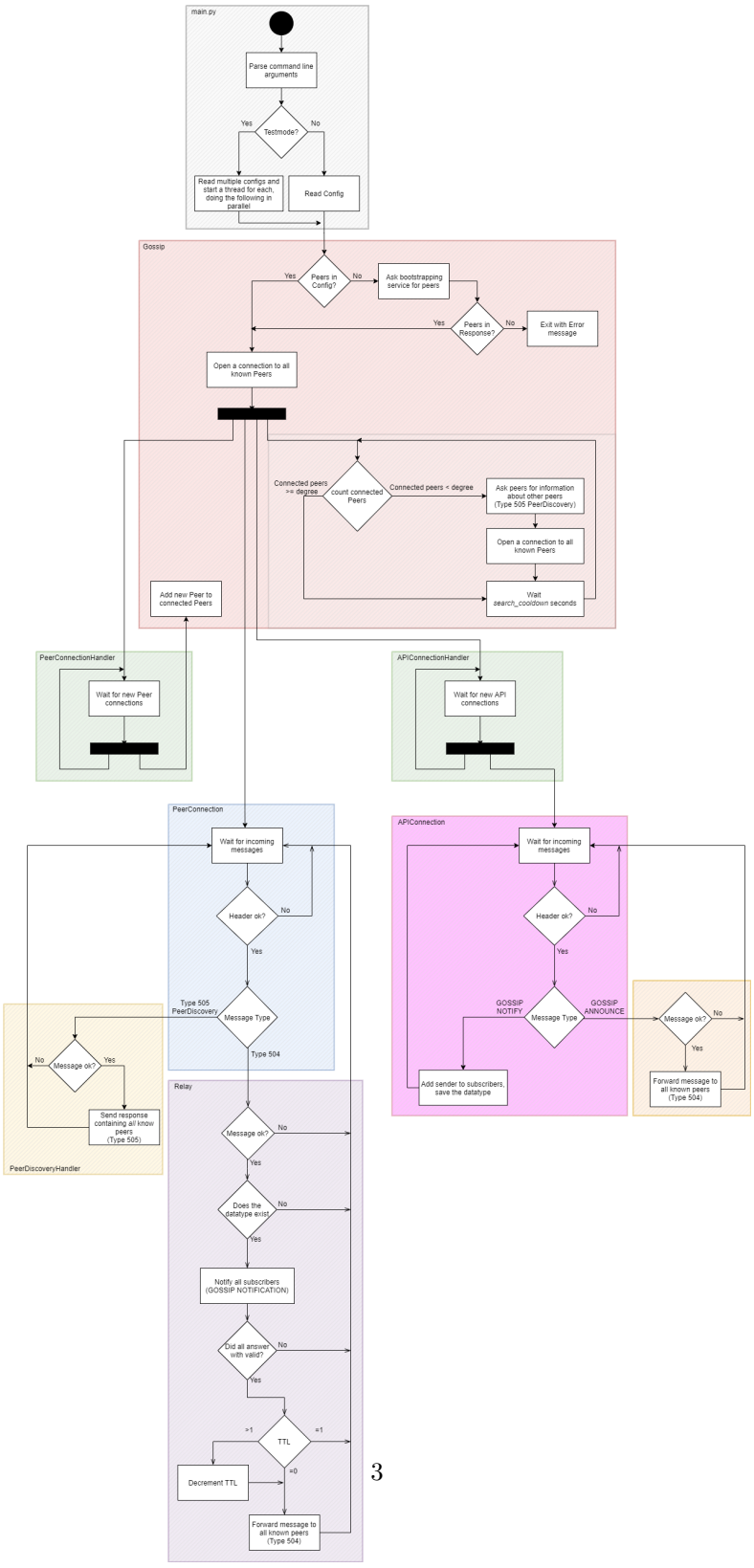

Figure 1: Current work in progress version of our flow chart for Gossip. Colored boxes represent sub modules.