Interims Report

Peer-to-Peer Systems and Security

Network Size Estimation

Group 37
Dominik Winter
Stefan Armbruster

1 Process Architecture

We decided to use the event loop library `asyncio` for the process architecture. In addition, as Python's threading library does not actually utilize multiple cores simultaneously for computation and thus will not lead to an increased performance, we decided to stick to a single threaded architecture. The combination of a single thread and the event loop library should give us the least overhead and should moreover be more secure than other methodologies.

2 Inter-Module Protocol

For our protocol we chose GNUnet's implementation of the network size estimation [1] as it has several benefits compared to other implementations. GNUnet NSE requires only $O(1)$ state per peer and sends controlled floods of $O(|E|)$ messages per round. In addition, it is more secure than other implementations by requiring Proof-of-Work-based peer identities as well as signatures attached to every protocol message.

In short, all peers that use the GNUnet NSE algorithm compute a distance estimation at a frequency of *f*, by comparing the own identity with a random key T and comparisons of other peers received over the last few iterations. The first matching bits of the peer identity and the random key T are counted and compared to the proximity calculated by other peers. The network size is then derived by computing $2^{p-0.332747}$, where p is the highest proximity of matching bits in the current round.

The following section introduces the message format used in our implementation of the GNUnet NSE algorithm.

2.1 Message Format

Due to the intension to keep the communication between the different peers as simple as possible, we decided to use the Json format for the messages. The usage of the Json format enables the access to the information within the message without using bit offsets. This will help us to implement the parts of code, which read or write the respective protocol messages much more clearly and understandable. Furthermore, Json support can be easily added to Python by importing the respective library.

Every key and its value are decribed in the following:

- **Proximity:** Contains the amount of equal bits of the random key and the peer identity.
- **Round-time:** Contains the time of the corresponding round of the NSE.
- **Pub-key:** Contains the public-key of the peer which sends the message.
- **PoW:** Contains the proof of work which consists of a time stamp, a random number and a hash.
- **Sign:** Contains the signature, which is used to authenticate the sender.

```
{
 "Proximity": String,
 "Round-time": String,
 "Pub-key": String,
 "PoW": {
         "Time": number,
         "Random-number": number,
         "Hash": number
        },
 "Sign": String
}
```

Figure 1: Message Format

2.2 Peer Authentication

The authentication of the communication is proceeded by a proof of work and a signature. The proof of work authentication uses a hash function which calculates a value with special properties. Thereby, the sending time of the message and an arbitrary random number are used as inputs. An example of such a property may be a certain number of zero bits at the beginning. Therefore, the proof of work can be used to give a time validity of the message. The signature is created with the help of Python's crypto library. The module takes the message and the RSA key of the sender to sign the message. Subusequently, the signature is appended to the Json message. Thus, the signature can be used to confirm the integrity of the message and the authentication.

2.3 Exception Handling

We are handling exceptions by creating *try-, except* blocks in every defined method, in oder to be able to output appropriate messages for every recieved error. This strategy will also be helpful when creating testcases with `pytest` and asserting certain exceptions.

References

[1] N. Evans, B. Polot, C. Grothoff, L. Kencl, L. Li, J. Widmer and H. Yin. *Efficient and Secure Decentralized Network Size Estimation.* In: *Networking 2012.* (2012), pp. 304-317. ISBN: 978-3-642-30045-5.