# Peer-to-Peer Systems and Security - Midterm Report Team 4

Kevin Burton

## 1 Changes to the Initial Report

So far the implementation of the module is faithful to the initial report, although the building using Setuptools and code scrutiny using Flake8 have not been performed yet. Furthermore the evaluation is planned with a different testing method than the one described in the intial report: instead of starting multiple independent nodes, each with their own NSE and Gossip modules, only a single node with the to be tested code is spawned, while a testing node simulates an entire network behind itself. The reason for this change lies in the gossip module. The gossip dummy distributed via the `voidphone_pytest` repository has no bootstrapping functionality for a new node to join the network. Furthermore, since communication between peers is handled by their respective gossip modules, a message of type GOSSIP ANNOUNCE would need to be forwarded by the gossip that it was announced to, to another gossip instance, before ultimately reaching the actual subscriber, in our case another NSE module. In conclusion, these constraints mean that it would be necessary to modify the gossip dummy beyond the scope of work of this project, which leads to the aforementioned method of evaluation, where the testing node calculates network estimates based on how many nodes are to be simulated and communicate these via a single gossip instance sitting between the real NSE node and the testing node.

## 2 Architecture and Protocols

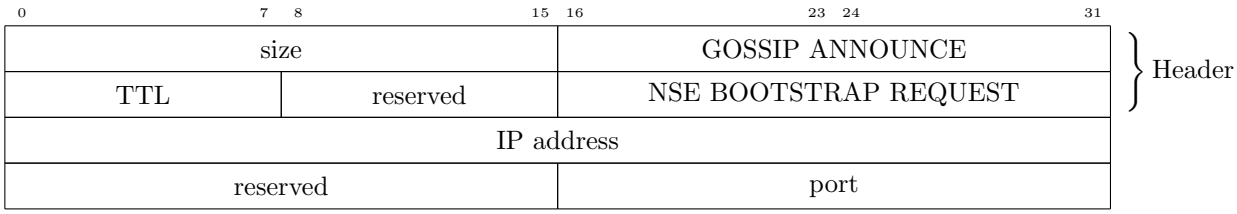
The module architecture is quite simple: the `main` file is responsible for bootstrapping and everything protocol related, namely receiving and sending messages, while the `nse_util` file is used to outsource frequently used calculations such as getting an estimate from a leading bit proximity and calculating proof of work.
The NSE protocol itself will be explained using the messages sent throughout. It is an implementation of GNUnet's NSE which is described in Nathan Evans 2012, with a few compromises that are further specified in section 4.

### 2.1 Bootstrapping

First of all every NSE module has to subscribe to GOSSIP ANNOUNCE messages of data type NSE BOOTSTRAP REQUEST by sending a GOSSIP NOTIFY to gossip. Therefore every peer in the network is able to respond to a new node requesting its first estimate. This request is sent out right after the subscription and is further specified in the following subsection.
Furthermore an event loop is set up at the beginning of module execution, on which two tasks are created to run asynchronously: the first being the server which listens for incoming messages, and the proximity calculation function, which is responsible for starting estimation rounds and calculating a proximity based on a node identifier and a random key.

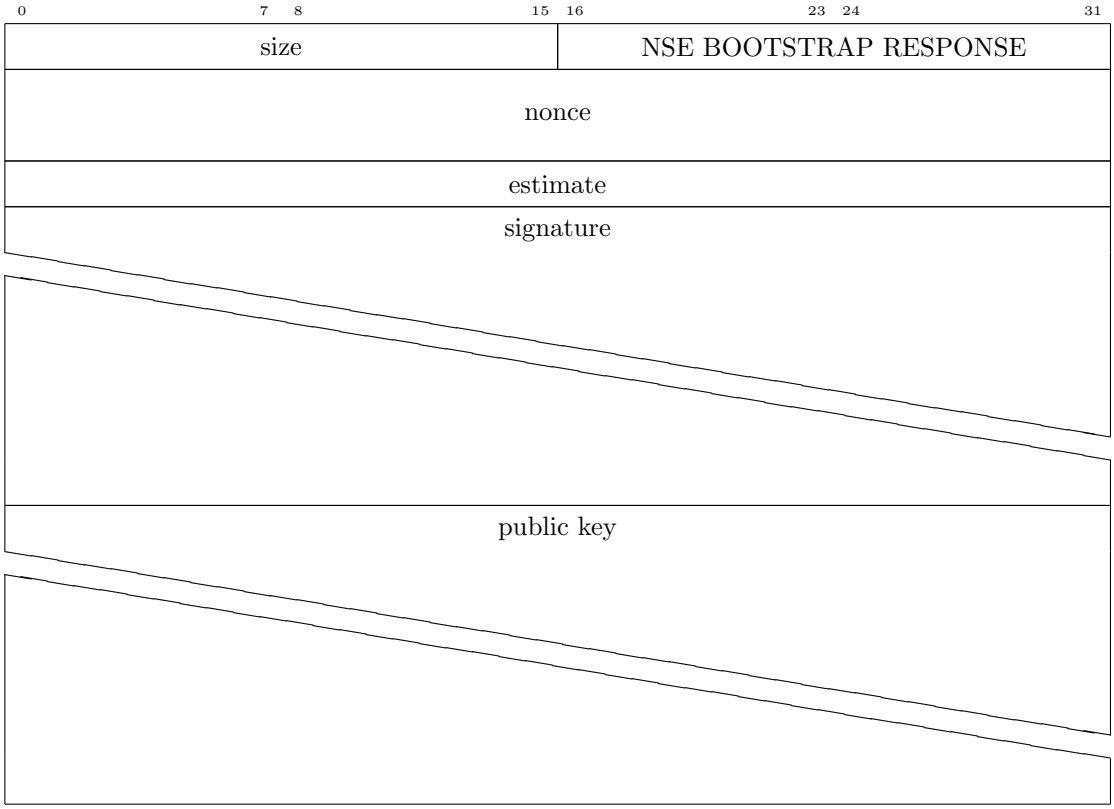
#### 2.1.1 Bootstrap Request

The bootstrap request announcement is sent to the local gossip instance at the start of module execution so that the NSE module has a first estimate to work with. In addition to the regular GOSSIP ANNOUNCE header fields it also contains the senders IP address and the port on which the NSE module can be reached.



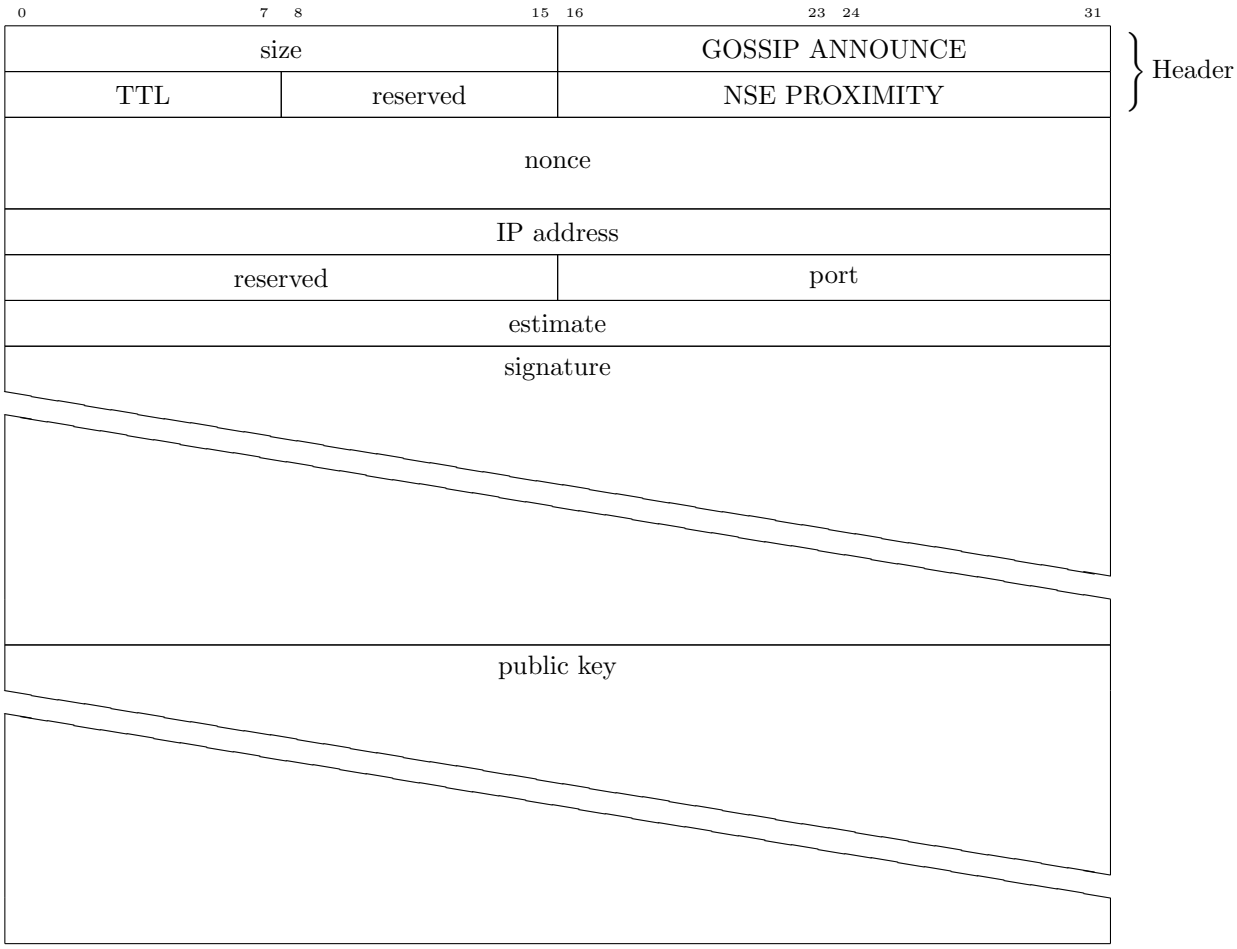

### 2.1.2 Bootstrap Response

Whenever an instance of an NSE module receives an announcement with NSE BOOTSTRAP REQUEST datatype, it answers with the following message mainly containing an estimate to bootstrap with. The random nonce provides proof of work, while the signature provides authenticity and integrity, which can be verified using the public key. The proof of work is achieved by a certain number of overlapping leading bytes between a hash of the data of the message and the node identifier, which is a hash of its public key.




## 2.2 Announcing and Relaying an Estimate

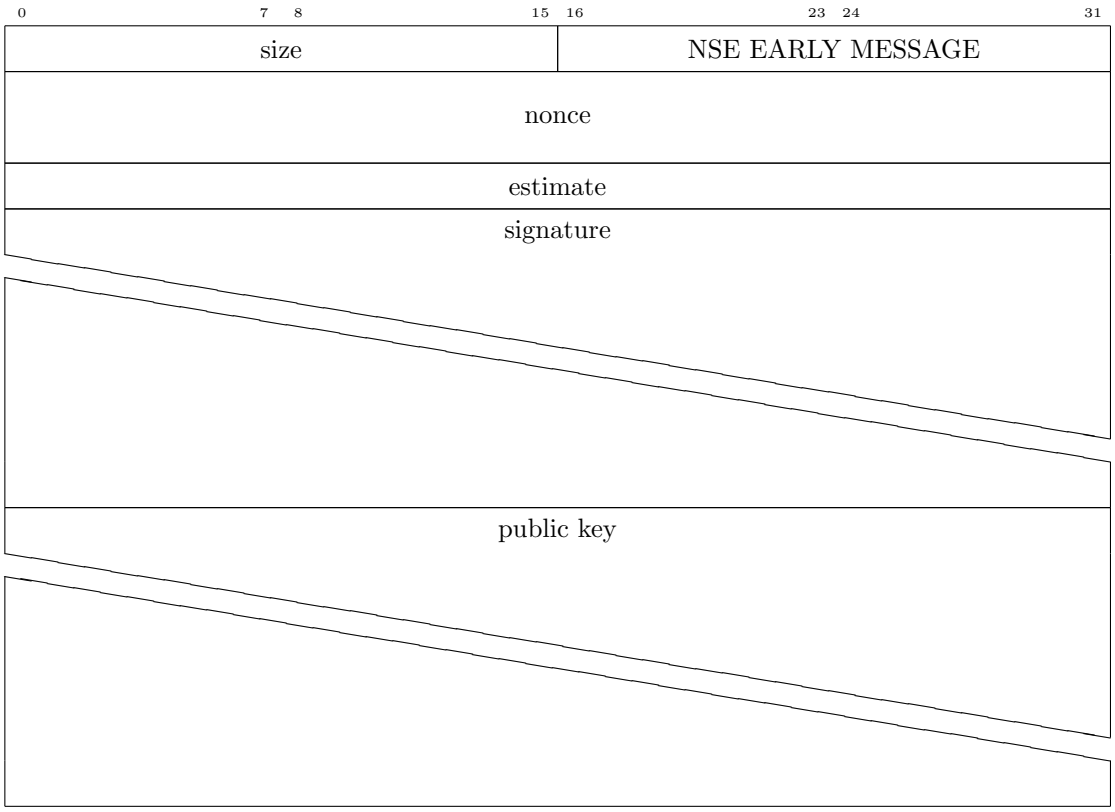
At the beginning of a round every node calculates a proximity by determining the number of overlapping leading bits between a random key and the node identifier. The maximum expected proximity in the network can then be translated to an estimate of how many peers are participating in the network. The calculation and proof is further explained in Nathan Evans 2012, section 3. From the locally calculated estimate a node can then determine when to announce it to its peers. Lower estimates are announced later in the round. The exact method of calculating this waiting time is explained in Nathan Evans 2012, section 3.2. When this waiting time ends the NSE module announces its estimate using the following message:




IP address and port are necessary to be included in the announcement so if another node notices that the message was sent too early for the estimate it transported, it can answer with the message type presented in section 2.3. Nonce, signature and public key are included for the same reason as in the BOOTSTRAP RESPONSE message type.
If our module receives such a message it has to compare the received estimate with the highest estimate of the round so far. In case it is higher we don't transmit the above message with our own calculated estimate and instead relay the received message after replacing IP address, port and public key with our own parameters, which also means that a new proof of work and signature has to be created. The other case is that a lower estimate is received, which indicates clock skew in the sending node. This leads to a sending of the message type shown in the following subsection.

## 2.3 Answering Clock Skew

This message is received when another node has detected that a message was sent too early, indicated by an estimate that was too low to be sent that early in the round. It mainly contains a proper estimate which can be immediately used to update the current one within the receiving node. Again, a nonce is included for proof of work, as is a signature and the corresponding public key for verification purposes.




# 3 Exception Handling

There are a couple of noteworthy exceptions that can occur and for which the respective handling has to be specified. The first kind happens whenever a connection breaks: this could be either between the NSE module and gossip or between two NSE modules when NSE EARLY MESSAGE or NSE BOOTSTRAP RESPONSE messages are sent. The former means that our module is cut off completely from the network and there's nothing we can really do other than keep listening and retrying for a connection. The latter is a non-issue as the messages leading to such a response are announcements and therefore most likely received by another peer in the network, which can then send their estimate, without ours being needed.
The other type of exception that could occur during service is that a message is corrupted or malicious. This is visible by the proof of work, integrity and authenticity checks performed when receiving a message containing an estimate. If any of these checks fail we respond to gossip with a GOSSIP VALIDATION with the last bit set to 0 if the received message was of type GOSSIP NOTIFICATION, which signals it to not further propagate that particular message. In any case we then simply discard the message.

# 4 Future Work

The first aspect that has to be tackled during the final stretch of the project is proper diagnostic output of the module during service. This will facilitate proper debugging while using the testing suite envisioned. As mentioned, evaluation of the main functionality will be done using a test module that connects to the same gossip instance that the NSE module which is to be tested, is connected to. According to different parameters given to the test on execution it will then begin simulating multiple NSE instances at the same time, calculating many proximities and announcing them at both appropriate and skewed times via gossip. This simulation will run for multiple rounds, each with a changed number of peers to simulate churn. After every round the current estimate of the tested NSE module is the queried and compared to the correct number of nodes in the simulated network.
Further future work that most likely will not be completed during the scope of this project are the compromises that had to be made because of gossip. The first one is evaluation using a real network, which was already mentioned in section 1. The second compromise is made in the relaying of a higher estimate. GNUnet's NSE handles the issue of a flood of messages in the network by delaying the relay of the message to each neighbour with a peer specific random delay (Nathan

Evans 2012, section 3.3). As the distribution of messages is handled by gossip, this is impossible without changing the gossip module or circumventing it completely.

# 5 Workload

All of the work was done by myself. It is very hard to gauge how much effort was spent on the project, but with the help of git I can track around eleven full days of work on the module itself so far.

# References

Nathan Evans Bartlomiej Polot, Christian Grothoff (2012). *Efficient and Secure Decentralized Network Size Estimation.* DOI: 10.1007/978-3-642-30045-5_23.