Peer-to-Peer Systems and Security

Final Report

Network Size Estimation

Group 41
Ankur Sinha
Saquib Shah

1.1. Dependencies Required:

The following project has been written in Python 3.5.2 on Ubuntu 16.04 LTS System.

The project makes use of the libraries, some of which are standard, some of which had to be installed specifically. They are: math, datetime, pycrypto, base64, string, sys, copy, struct, statistics, asyncio, asyncio, configparser and unittest.

To install any particular library, use: `pip install <library_name>`

1.2. Program Setup:

The project directory is setup as follows:

- p2p-vp-nse
  - code
    - main.py
    - crypto_op.py
    - test_cases.py
    - proof_of_work.py
    - tcp_connect.py
    - nse_protocol_validator.py
    - time_compute.py
    - config.ini
    - hostkey.pem
    - nse_protocol_handler.py
    - gossip_handler.py

The *hostkey.pem* file contains the private key.

The *config.ini* file contains the configuration parameters such as API address, ports, history length, and other. The config file looks like:

To run the program, go to console and type: `python main.py`
Or run the *main.py* file from your suitable IDE.

The *main.py* file automatically calls the necessary files and functions to keep the program up and running.

To run the test cases, go to console and type: `python test_cases.py`
Or run the *test_cases.py* file from your suitable IDE.

1.3. Issues:

Proper error messages, catch statements, and outputs have been added to wherever applicable to trace the issue. However, the program has been run with relatively low number of peers and has worked fine. It needs to be seen if there is any change with more peers and how the estimates get calculated based on the proximity. A history length of 5 has been maintained.

The test cases have all been included in one file. It takes around 30 seconds to run with various outputs coming on screen due to some functions being called. Please have the patience, and sorry for the inconvenience for waiting.

2. Protocol Implementation:

The protocol has been pretty much what was mentioned in the interim report.

We have finalised (from [1] and [2]) on using the methodology mentioned in GNUnet's paper titled "Efficient and Secure Decentralized Network Size Estimation" [1]. The algorithm computes the estimation using the identity of the peer and the random key T with the matching prefixes of its own and other messages received, over the last few iterations (typically, last 64 iterations), in short.

This protocol has been chosen because it employs Proof-of-Work mechanism to maintain a good level of security preventing malicious nodes from causing huge deviations in the size estimation. In addition, it is highly efficient as it has a complexity of O(1).

The main components of the message format are as follows: S corresponds to the starting time of the particular round, typically rounded off to the beginning of the particular hour. Proximity p is calculated by the hashing of starting time and public key. Ás mentioned earlier, PoW mechanism helps in keeping the system secure from malicious nodes, if any. Signature is calculated by hashing the other fields of the message. It is used to validate the fact that the public key has been derived from the private key. Hop count functionality has been skipped.

Offset	***Contents***
0	*Message header magic code*
4	*Hop-Count (updated at each peer)*
8	*Signed data header magic code*
16	*Time S of the round*

24	*Proximity p in bits*
28	*Public key (2048 bit RSA)*
288	*Proof-of-Work*
296	*Signature (signing bytes 8–295)*

Table 1: Message Format [1]

3. Future Enhancements:

Currently, the proof-of-work generated is completely static based on the rounded starting hour of the current time just like the protocol uses it. In the near future, the difficulty of this could be made more complex and the generated problem of rounded starting hour could be replaced with something better to not give the attacker any possibility of guesses. In addition, one could also add other alternative consensus mechanisms such as proof-of-stake and proof-of-burn which are gaining popularity in cryptocurrencies sector.

4. Work Distribution:

During the initial phase of the semester, Saquib did give some inputs and ideas. However, Ankur has done bulk of the work and all of the implementation and documentation part himself. Saquib also did not register for the course.

5.1. Team Effort:

Description	**Effort (in hours)**
Discuss about initial approach	1
Discuss about interim approach	3

5.2. Individual Effort

Description	**Effort (in hours)**
Read previous initial reports	1
Write current initial report	1
Read and understand [1]	6
Read and understand [2]	6

Write current interim report	2
Look into existing implementations and understand them	10
Develop config parsing module	1
Understand proof-of-work concept, read about blockchain and bitcoin [3]	8
Understand and develop working prototype of PoW	2
Understand and develop time computation module	1
Understand struct and unstruct messaging concept	2
Understand and develop tcp connect module with asyncio	5
Understand and develop gossip message handler module	7
Understand and develop cryptographic operations	2
Incorporate struct and unstruct message handling	4
Understand and develop protocol validation module	3
Code integration with different modules	4
Code debugging within modules	4
Code clean up and repository maintenance	1
Write sample test cases	2
Miscellaneous discussions with people with experience in such projects	5
Final report	2

References:

[1] Evans, Polot, Grothoff - *Efficient and Secure Decentralized Network Size Estimation*

[2] Van de Bovenkamp, Kuipers, Van Mieghem - *Gossip-based counting in dynamic networks*

[3] Satoshi Nakamoto - *Bitcoin: A peer-to-peer electronic cash system*