DECENTRALIZED REPUTATION MODEL AND GENERAL TRUST FRAMEWORK BASED ON BLOCKCHAIN & SMARTCONTRACTS

Dissertation in partial fulfillment of the requirements for the degree of

MASTER PROGRAMME IN COMPUTER SCIENCE

UPPSALA
UNIVERSITET

Uppsala University
Department of Information Technology

SUJATA TAMANG

May 9, 2018

DECENTRALIZED REPUTATION MODEL AND GENERAL TRUST FRAMEWORK

Dissertation in partial fulfillment of the requirements for the degree of

MASTER PROGRAMME IN COMPUTER SCIENCE

Uppsala University
Department of Information Technology

Approved by

Supervisor, Jonatan Bergquist

Reviewer, Björn Victor

Examiner, Prof. fName lName

May 9, 2018

Abstract

Abstract here

Acknowledgements

Acknowledgements here

Table of Contents

List of Tables

List of Figures

List of Abbreviations

1 Introduction

1.1 Definition

1.1.1 Trust and Reputation

Trust and Reputation encompass a broad spectrum of domains and is context dependent. Therefore, a universally agreed upon single definition doesn't exist. From a game theoretic sense, trust can be interpreted as a subjective probability, by which an individual, A, expects another individual, B, to perform a given action on which its welfare depends according to a previous agreement. [1] Reputation, on the other hand, is the perception of an individuals character or standing. Individuals in online systems are identified by their online identities which can be anything and not necessarily attached to real-world identities.[2] Online identities play a crucial role in digital interactions and require unknown entities to trust each other based on the reputation system of the platform in use.

1.1.2 Blockchain

Blockchain can be defined as a distributed record of state changes that let anybody on the network audit state changes and proves with mathematical certainty that the transactions transpired according to the blockchain rules. There exist several definitions of blockchain technology each specific to their closest use case. A formal standard definition of Blockchain is under development as ISO/TC 307.[1] Vitalik Buterin, the founder of Ethereum, puts it this way. "A blockchain is a magic computer that anyone can upload programs to and leave the programs to self-execute, where the current and all previous states of every program are always publicly visible, and which carries a very strong cryptoeconomically secured guarantee that programs running on the chain will continue to execute in exactly the way that the blockchain protocol specifies." This definition provides a broad overview of what blockchain does.[2] As a continually developing discipline, it needs to keep adapting to a new definition while maintaining the essence. This thesis will discuss the topic in more detail in the background section.

[1] https://www.iso.org/committee/6266604/x/catalogue/p/0/u/1/w/0/d/0
[2] https://blog.ethereum.org/2015/04/13/visions-part-1-the-value-of-blockchain-technology

1.2 Motivation

Consider a simple scenario where Alice wants to buy a pair of headphones for which she browses a buy/sell platform. When she finds a relevant product on the platform published by Bob, unknown entity to Alice, she needs to rely on the ratings/feedback that Bob has received on the platform from his previous customers and also on the platform in use for not tampering with the data in any form. The entity claiming to be Bob could be Eve who found a way to bypass the platform's security and inflate his reputation on the system. Eve could delete the ad and associated account when the payment is complete, or she could gather Alice's personal details to misuse it later. Any malformed decision on the trustworthiness of an entity could be expensive and deal severe damage to the user. Thus, it is interesting to study about reputation model and methods to make it more reliable and accurate in its measure. Reputation model offers a way to measure the trustworthiness of entities to aid interacting users in making an informed decision about carrying forward the transaction or dropping it.
Studying interactions between entities and analyzing their behavior to generalize a trust framework is, therefore, a riveting problem. Graph theory and network flow algorithms have been researched in both centralized and decentralized environment before. This thesis proposes a blockchain based solution to record users behavior and compute a trust score for each of them.

1.3 Purpose and research questions

The main goal of this thesis is to use blockchain technology and smart contracts to simulate an endorsement network where entities can endorse each other based on physical or digital acquaintance. The endorsement values will be quantified to infer reputation score and a trust value that can be used on any transaction network. The nodes and their relationship will be studied to identify honest or malicious participants. Generalization of this endorsement network to serve other use cases will also be discussed. The research questions that this thesis aims to address are :

1. How can graph theories and relevant reputation algorithms be used to model the interaction between entities and detect/identify honest and malicious nodes in the network? How can the interaction graph be modeled?

2. What are the requirements for storing trust values and linking them to associated identities stored off a blockchain network? How can a blockchain application be built to define a general trust framework for a transactional network? How could the overall system architecture look like?

3. How can the discussed endorsement network ensure trustworthiness while also preserving users anonymity and how can it be generalized to other transactional

network or added on top of it to serve other use cases such as content filtering, E-Commerce etc?

1.4 Scope

This thesis work attempts to answer all the research questions mentioned in section 1.3.

To answer research question1, literature survey will be performed on existing reputation algorithms along with the presentation of background overview that will lead to graph simulation of endorsement network.

For research question2, interpretation and quantification of reputation scores and trust metrics will be manifested. Comparative analysis of on chain and off-chain storage requirements will be studied resulting in an overall design of endorsement system architecture.

For research question3, relevant use cases will be presented, and the network will be tested on with various predefined cases and attack models to see how well it behaves in a dynamic environment.

1.5 Structure of Report

1.5.1 acronyms

2 Literature Review

2.1 Existing Reputation Systems

2.2 Problems & Limitations

Existing reputation models aggregate feedbacks and evaluate actions and interactions of users and store them in a centralized database. i.e., A trusted node has the access control and rights to publish information to the network which implies that it could tamper with the data at will. The traditional client-server architecture is also susceptible to DDOS attack as the target is known and holds a single point of failure. Another challenge that is not limited to the centralized system is Sybil attack. In any digital platform that doesn't require one to reveal personally identifiable information, creating multiple pseudonymous identities to exploit the system is usually cheaper with nothing to lose. Sybil attack is one of the most significant challenges in a distributed computing environment. It is usually challenging to detect and has been mathematically proven to be impossible to prevent in a distributed environment.

2.2.1 Sybil Attack

Sybil attack is a widely used attack model in the peer-to-peer reputation system. Peers in the network create multiple pseudonymous identities with a purpose of inflating their reputation or damaging some other peers reputation. If a peer gets a bad reputation in the system for its activity or other reputation models defined parameters, then usually it is both cheaper and faster to create a new identity and start afresh then to try and recuperate the damaged reputation. As the network makes it so easy to create identities with nothing at stake, participants opt for it and exploit this feature to perform Sybil attack.

3 Background

3.1 Reputation algorithms

3.1.1 Graph properties

A graph, as the name suggests can be used to represent objects and their relationships graphically. A graph G is an ordered triple (V,E,φ_G) where V is a non empty set of vertices v, E is a non empty set of edges e that connects two vertices and $v \in V, e \in E$. φ_G is an incidence function that assigns pair of vertices to each edge of the graph G. φ_G(e) = uv represents that e is an edge that joins vertices u and v. Graph properties can be leveraged to serve as an interaction graph of network for reputation system. Each node on the network, v can represent individuals and the edges that connect the nodes can represent the relationships between those nodes. The edge can have varying weights to represent the strength of relationship between the nodes. [3]

3.1.2 EigenTrust

EigenTrust is a reputation management algorithm for P2P network that aims to minimize malicious behaviour in the network and is based on the notion of transitive trust. i.e. If a peer i trusts a peer j then all other peers trusted by j is also trusted by i. In EigenTrust, global reputation of each peer i is given by local trust value assigned to peer i by other peers and is weighted by the global reputation of assigning peers. A local trust value s_{ij} is calculated by each peer i which represents the opinion i has of j. s_{ij} is the difference of satisfactory and unsatisfactory transactions peer i had with other peers j.

$$s_{ij} = sat(i,j) - unsat(i,j) \tag{3.1}$$

where sat(i,j) represents number of satisfactory transactions that i had with j whereas unsat(i,j) represents number of unsatisfactory transactions.
To prevent malicious peers from assigning arbitrarily high local trust values to other malicious peers, the local trust value is normalized as c_{ij} before aggregating them.

$$c_{ij} = \frac{max(s_{ij}, 0)}{\sum_j max(s_{ij}, 0)} \tag{3.2}$$

C_{ij} keeps changing depending on the good or bad interaction between peer i and peer j. Based on the local trust value assigned by other peers, each peer has a global trust value that determines their standing in the network. To aggregate the normalized local

trust values, the approach used is friend-friend reference where a peer i would ask its acquaintances about their opinion about other peers. Trust that peer i places in peer k by asking his friends can be denoted by t_{ik} as :

$$t_{ik} = \sum_{j} c_{ij} c_{jk} \tag{3.3}$$

Each peer asks other peers about their opinion which is weighted based on how much peer i trusts them. If we define C as a matrix $[c_{ij}]$ and t_i as a vector containing values t_{ik}, then $t_{ik} = C^T \vec{c_i}$. This helps a peer get a wider view of the network more than its own experience. This can continue for many nodes until peer i asks his friend's friend's and friend's friend can be consulted further to receive a broader view of the network. For n nodes, we can represent t as t = $(C^T)^n c_i$. For a large enough value of n , trust vector $\vec{t_i}$ will converge to same vector for every peer i and could give complete view of the network. t is the global trust vector where t_j quantifies the trust system places in peer j. EigenTrust is robust to malicious peers and good for decreasing inauthentic file downloads in a P2P network. However, it doesn't address the issues such as inactive peers, where a peer doesn't download from anywhere else, malicious collectiveness, where malicious peers collude to inflate the trust value. It also doesn't have a way to calculate negative trust and is entirely based on user feedback. [4]

3.1.3 Net flow Rate convergence

Net flow rate convergence can help to determine anomaly in the network. By looking at how fast the net flow converges to zero, it can detect unusual behaviour in the network. The flow in a network can be measured by looking at inflow and outflow edges and calculating their differences. Inflow edges are all incoming edges in the graph and outflow edges are all outgoing edges . (diagram) Net flow convergence rate is the rate at which the net flow converges to the global net flow which is zero. Depending upon how fast the net flow in a graph converges to zero, it can be useful to detect anomaly.(example diagram)

3.2 Cryptography

3.2.1 Basic Concepts

Cryptography offers algorithms to achieve confidentiality, integrity, authenticity, and non-repudiation. Confidentiality and integrity ensure that the information being communicated is not disclosed or has been modified to or by any unauthorized parties. The data is hidden or encrypted such that only the authorized parties can make sense out of it, i.e. decrypt using the previously agreed upon key.
Asymmetric key cryptography makes use of key pairs, private key, known only to the owner and public key, that can be publicly distributed. It ensures authenticity, a proof

that sender is who he claims to be and non-repudiation, the sender cannot deny having sent the message. Public key verifies the holder of the private key and encryption of the message. That paired private key can only decrypt this encrypted message. One of the significant application of public key cryptography is Digital Signatures, described in more detail in section (Blockchain section ..) which is useful in preserving the properties of authenticity and non repudiation.
A cryptosystem can be seen as a five tuple (P,C,K,E,D) that satisfies the following conditions:
P is a finite set of plain texts.
C is a finite set of cipher texts.
K, the keyspace is a finite set of keys
E, set of encryption rules e_k: P $\Rightarrow$ C
D, sete of decrytion rules d_k : C $\Rightarrow$ M.
for each k $\in$ K, there is $e_k \in$ E and $d_k \in$ D such that $d_k(e_k(m))$ = m for every plaintext m $\in$ P.

3.2.2 Hash functions

Cryptographic hash functions are a one-way function, also known as mathematical trapdoor function that transforms an input message into a fixed length binary output. It is one way because although converting a message input to a hash value or a message digest can be done in constant time, reversing the operation is practically impossible to achieve as its computationally inefficient. Earlier hash functions include MD5 which produces a 128-bit hash value but is vulnerable and can be cracked by brute force attack. The predecessors hash functions are sha-256 preceeded by sha-1, sha-2 and others. Their applications include the digital signature, message authentication both of which are interesting for blockchain as will be discussed in section(name). The essential characteristics of hash functions are their deterministic output, meaning given a fixed input; it will always generate the same output. It offers collision resistant property, i.e. it is impossible or extremely rare to get the same hash value for two different messages. If m1 and m2 are the message and h(m1) and h(m2) are hash functions applied to them respectively, collision resistant ensures that h(m1) != h(m2). Another important characteristic of a hash function is that the hash value does not indicate the original information that was hashed thus making it efficient for hiding information.

3.2.3 Digital Signature

A digital signature acts as an intermediary to prove that an entity A, has the password without ever requiring A to reveal it. To create a digital signature, one would need to apply signing algorithm to the private key along with the message. Likewise, anyone can verify the generated signature by applying it to a verification algorithm along with

the public key and the message. If a node A intends to send a transaction to B on a blockchain network, A needs to prove that he is the rightful owner of the public address from where the message originated. This is done by creating a digital signature using A's private key from the transaction message. Once the transaction is broadcasted, any node in the network can verify that signature corresponds with A's public key. The signature is dependent on the message, and thus any attempt by a malicious node on the network to modify the message will refute the signature.

3.3 Blockchain Technology

Blockchain, as the name suggests is a chain of blocks, where each block had the consensus of all the nodes on the network before being unlocked. A block consists of a list of transactions that a miner accumulated at a particular point in time. Blockchain shows the ordering of transactions in the network.

Block0 -> Block 1 -> Block 2 -> . . .
————————-Time————————>

3.3.1 Evolution & Categories

Bitcoin was the first application that made use of Blockchain technology which was a peer-to-peer electronic cash system. The major contribution of this project was distributed trust at scale without using a trusted intermediary. Along the dimension of validation and access control, blockchain can be categorized as public permissionless, public permissioned, private permissioned.
While various use cases and diverse domains have shown an interest in the technology, it is crucial to make a distinction on when and for what purpose does it make more sense.
Hashgraph, which is a new project that claims to have solved the scalability issue of blockchain while maintaining security describes the different stages of evolution as:
Leader Based System
Proof-of-work blockchain
Economy based systems
Voting based systems
Hashgraph with virtual voting

3.3.2 Consensus algorithms

Consensus algorithm is the defining element in any blockchain network based on which all nodes agree about the transactions ordering and timestamps of all messages.
proof-of-work: This was the first consensus algorithm in use and has been proven to be

robust regarding security. However, it comes with a trade offs such as scalability.

proof-of-stake :

DAGs: Byzantine Fault Tolerance

3.3.3 Smart contracts

Smart Contracts

3.3.4 Applications

4 Methodology and Implementation

The problem of measuring the trustworthiness of communicating entities is an essential aspect of any online system where they interact with each other for any purpose, be it shopping, content delivery or file sharing. This chapter follows on a discussion of a proposed endorsement network where physically or digitally acquainted entities can endorse each other or their presented information. The model will address several concerns such as the roles and requirements of participants as endorser and endorsee, why a participant would play by the rule and what is to stop them from not doing so, threat models, etc. With a system of smart contracts, PoC design will confer interaction between entities, aggregation of information and assignment of scores for final computation. The storage of data both on and off-chain will be discussed.

4.1 Problem Statement

To be able to rely on the trustworthiness of an entity as presented by any online systems, the underlying reputation system needs to be robust and as transparent as possible. The assurance that available information has not been tampered with and correctness of claimed identity should be provided to sustain minimal risk of fraud. The immutable, trustless, decentralized and distributed attribute of blockchain protocol is a recommended solution on a public permissionless network.

4.2 User stories & Requirements

Anyone can join the network and become a participant in the endorsement system. The two notable roles of a user are endorser and endorsee. An endorser can initiate the transaction by sending an endorsement to the participant they wish to. The same user can assume both the roles of endorser and endorsee as long as a set of predefined requirements are met.

The user stories for each role that defines the system requirements for each user type is presented in table 4.2.

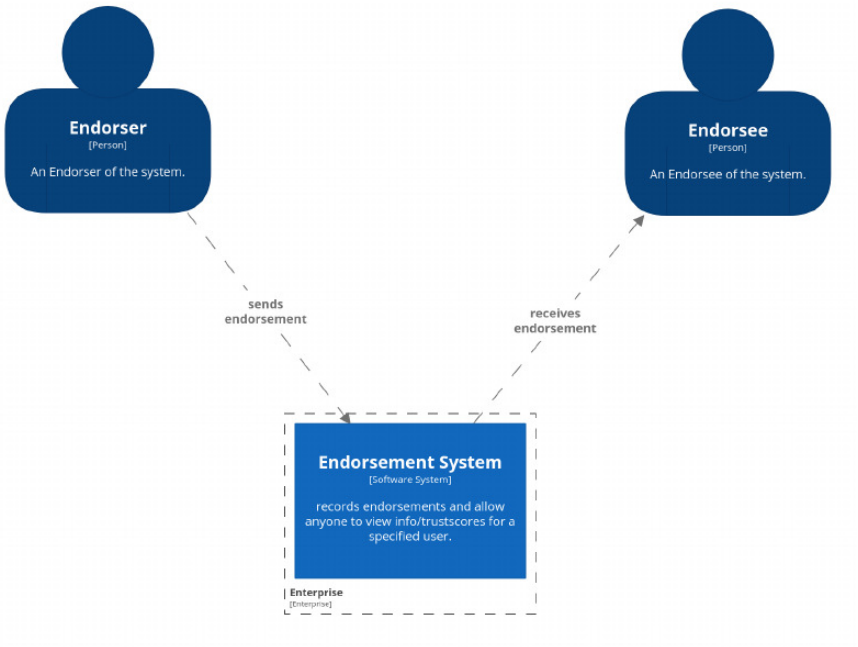

Figure 4.1: Context Layer

As an	**I need to be able to..**	**Traceability**
Endorser	send an endorsement so that the endorsement is received by the endorsee.	R1
	remove endorsement so that the endorsement is removed from the endorsee.	R2
	view a list of endorsees so that i can see to whom i have sent endorsements.	R3
	view/edit my personal information so that i can keep it up to date	R5
Endorsee	view a list of endorsers so that I can see from whom I have received endorsements.	R3
other users	compute the total endorsement impact(i.e., final computed score) of any registered members so that I can make an informed decision about the future transactions.	R4
	make a request to join the endorsement network so that I can start sending/receiving endorsements.	R1

The functional requirements can be listed in points as :

1. It must be impossible to make an Endorsement if the endorser and endorsee is same address or not a registered participant.

2. It must be impossible to remove an endorsement if the endorser doesn't belong to the list of endorsers for the given endorsee.
3. All the endorsements must be stored such that, it is possible to see:
   - endorser and endorsee for the given endorsement.
   - degree of incoming and outgoing connections for all endorsers and endorsees.
4. There must be a way to link the public key hashes to the corresponding computed trust scores.
5. It must be possible for a participant to edit their own profile if the editor is the same as the profile owner.

The non-functional system requirements are :

1. Security: smartcontract security.
2. Reliability: reliability of data, tamperproof and verifiable,immutable traceability.

4.3 The Model - Endorsement Network

The initial assumption is that all nodes are honest and as such receive equal points that they can spend at will once registered on the network. This received points are the consumable power that keeps depleting with every endorsement connection made along the way. As depicted in figure 4.2 , these points follow a convergent sequence that converges to the limit 0 as the number of connection 'n' increases. As such, increasing the number of connection alone will not be enough to achieve a higher impact on the network.

4.3.1 Computation of Total Endorsement Impact(tei)

The total endorsement impact corresponds to the total impact a participant has made on the network by sending or receiving endorsements. The two factors that are primarily responsible for computation of final trust score of a participant are the number of incoming and outgoing connections.

The final trust score, associated with the total endorsement impact a participant/node has made on the network requires familiarity with some new terminologies which is briefly discussed below.

nEG_A: number of endorsements sent by a participant 'A'.

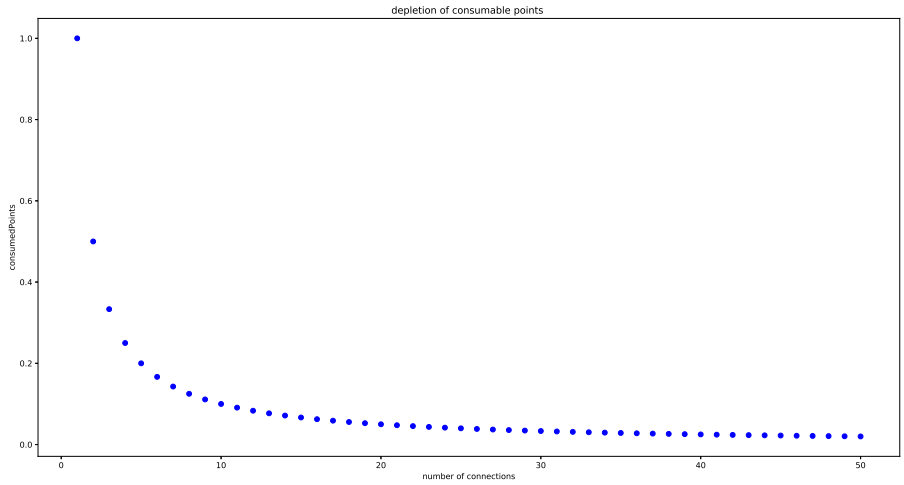

Figure 4.2: Convergent behaviour of consumable points as 'n' increases

nER_A: number of endorsements received by a participant 'A'.

$ratio_A$: ratio of nEG_A to nER_A. This value ensures that sent and received endorsement are not far off from each other. $ratio_A$ is always assumed to be less than 1 and is given by:

$$ratio_A = \frac{min(nEG_A, nER_A)}{max(nEG_A, nER_A)} \tag{4.1}$$

cp_A: Total amounts of points spent by a participant 'A' out of the initially received consumable points. 1 being the initial consumable points received by everyone who joins the network, cp_A is given by $1/nEG_A$.
TRP_A: This corresponds to A's total received points which is the accumulated sum of consumed points by all endorser of A. If a peer 'A' receives an endorsement from 'n' number of peers, then the TRP_A is calculated as:

$$TRP_A = \sum_{i=0}^{n} cp_i \tag{4.2}$$

Finally, the total endorsement impact made by 'A' is given by:

$$TEI_a = ratio_a * cp_a * TRP_a \tag{4.3}$$

4.4 Design of PoC

This chapter focuses on overview and design details of PoC based on the requirements mentioned in section 4.2 on page 10. It starts with design considerations, smart contract

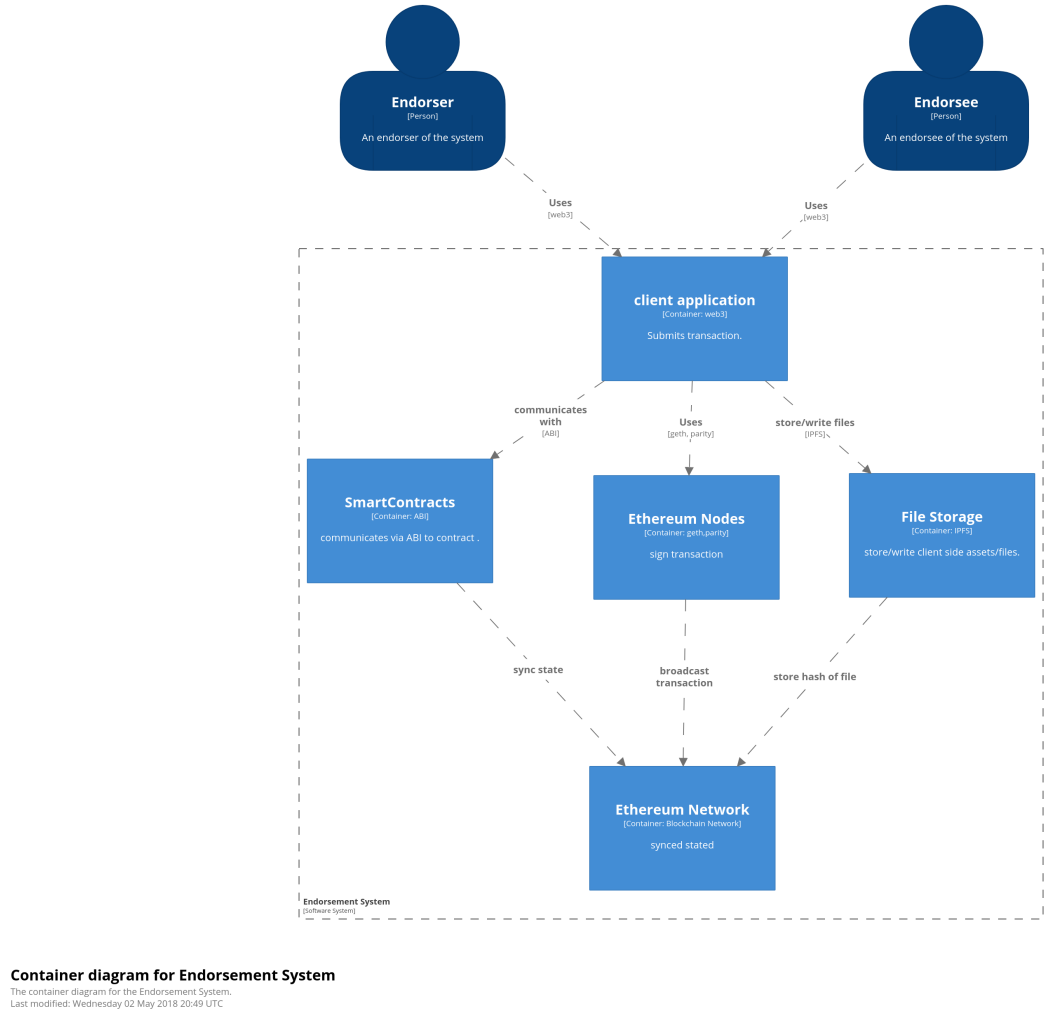

Figure 4.3: Container Layer

setup, data storage on and off blockchain. The high level system overview is presented in 4.3

4.4.1 Design Consideration

Honest and malicious participation in the network and the possible behavior that can result from the interaction between nodes were considered during the design. From a game-theoretic perspective of a behavioral outcome, following definition were made for network influencing factors.

1. **Fake endorsements with pseudonymous identities:**Endorsement system being on a distributed, public permissionless blockchain network allows anyone to join and start sending endorsements immediately to whoever they wish to. This creates the possibility that an entity could create multiple pseudonymous identities with an aim to inflate their impact on the network by increasing nEG or nER for their associated persona. There is no straightforward way to detect and stop this behav-

ior right away. However, if doing so doesn't provide any significant advantage, then the assumption is that a rational decision would be not to do it. One factor that is believed to stop a participant from making too many endorsements is the convergent behavior of consumable points. While, there is no limit to the amount of connection a participant can form, as the number of connection increases, the value of consumable points decreases.

2. **Transaction cost:** Ethereum is a programmable blockchain that supports a Turing complete programming language. Thus, to avoid running in infinite loops or DDOS attacks, gas was introduced. Every operation has a gas cost. One could write a program to do anything they wish to do on the network as long as the account that initiates the transaction can pay the gas cost of all operations. The gas consumption is an imperative aspect of endorsement system for two reasons.

   - *Standard transactions on Ethereum:* A participant that makes a call to 'sendEndorsement' function is responsible for paying all the required gas costs. This function updates the state variable nEG and nER. While the price may not seem too high for making one transaction, a malicious node with multiple pseudonymous identities has to pay for all the operations initiated by all personas. For instance, given the interaction graph in the figure, if Alice is an honest node, then she only needs to pay for the operation of one transaction. Whereas if both Bob and Charlie are the pseudonymous identities of Alice, she needs to pay for six transactions. Thus, the assumption is that they may make ether transfer at some point between their accounts. This information is publicly available for anyone on the ethereum blockchain network to view the chain of ownership. If some interactions in the endorsement network look suspicious, one could look up this detail. This method is not guaranteed to detect a Sybil node on the endorsement network but is just another additional factor that might be useful before making a decision.
   - *Local information of all neighbouring nodes:* Whenever an endorser makes a new connection, the nEG, and consumable point change accordingly. This change in consumable point has to be reflected for the list of all endorsees. This is not to be confused with a one time update. Every new connection made by an endorser changes the state for all his/her old endorsees. Therefore, all the neighboring nodes of an endorser should be stored previously. The impact of a participant is dependent on his/her direct interaction as well as the endorsers(the participants that have endorsed them). There is no way to make constant cost lookups and updates for this operation. It requires iterating through the list of arrays and computing the impact of every endorsee based on the updated state variable. While it is possible to iterate through items in the array, the general recommendation is to avoid them if possible. One

could surely assume that the list will not grow too big for two reasons: (a)A rational node will not make too many connections for reasons mentioned earlier in 1 (b)Dunbar's number suggests a cognitive limit of 150-250 stable social relationships for humans.
One way to approach this problem is to store the list of endorsees and endorsers for a participant but not change the state. The computation can be done on the client-side using language such as javascript. The final score will be done by the client. However, all the variables necessary to compute the final score will be stored and updated on blockchain as a publicly verifiable information.

3. **Dynamism:** The dynamic social behavior of human is that trust between two entities is not perpetual. Alice may have trusted Bob yesterday but refuses to endorse him today. Trust is dynamic and so is the endorsement decision that an entity can take. Therefore, the design also considers removal of endorsement previously assigned. The removal of endorsement is captured by the figure 4.4.

4. **Free-Riders Problem:** Free riders problem is addressed by making it necessary to maintain the ratio between nEG and nER. A peer without a balanced proportion cannot have a significant impact score on the endorsement network. This method also discourages Sybil nodes because each identity needs to have an almost equal bi-directional connection. If they are only receiving from their own pseudo identity that don't have too many connections, then the impact is ignorant and thus useless and not worth the time.

4.4.2 SmartContract

The startup process for registering the contract on the public blockchain network is shown in figure 4.5. Smart contracts can be either deterministic which doesn't require any information from outside blockchain or non-deterministic which does need to get oracles from external sources.(cite) For this PoC, endorsement contract is deterministic and so all the data and variables required are stored and executed on the blockchain.
The system of smart contracts on the component level is depicted in 4.6. The main contracts written for this PoC are :

- Ownable: tracks the owner of the contract. i.e., the creator of contract.
- killable: inherits from ownable and can be killed by owner only.
- Participants: set participant and store their information. An index to access each participant.
- Endorsement: It inherits from participants and can be called by participants only. Endorsement contract handles the core logic of endorsement system, accesses/-

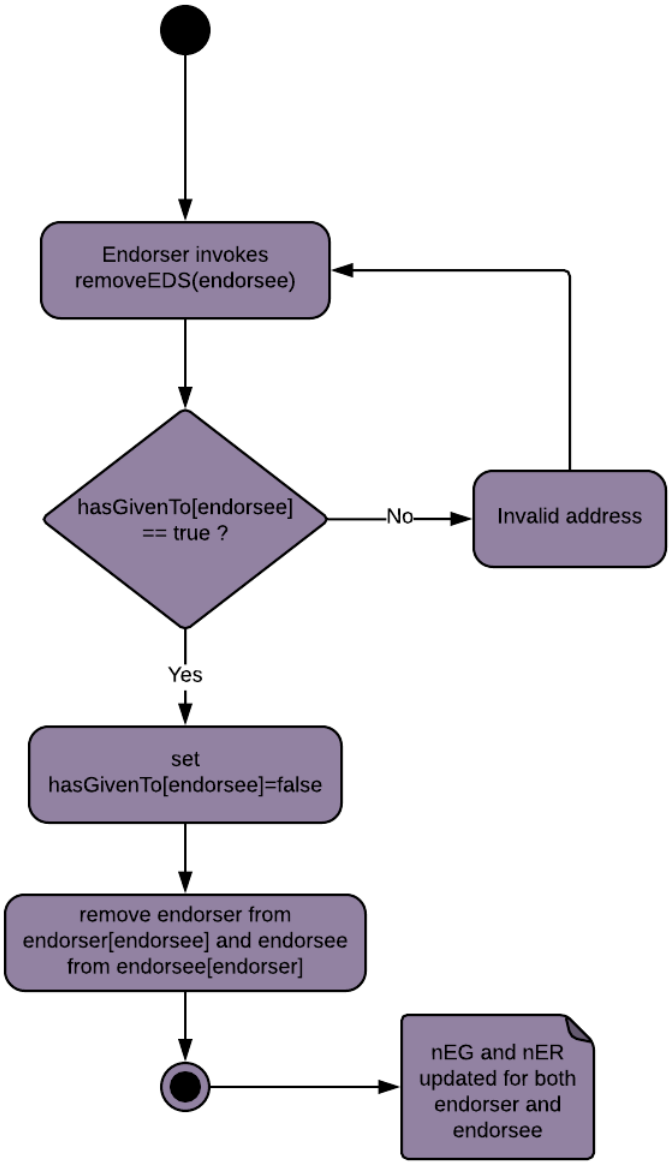

Figure 4.4: Activity diagram for removing endorsement

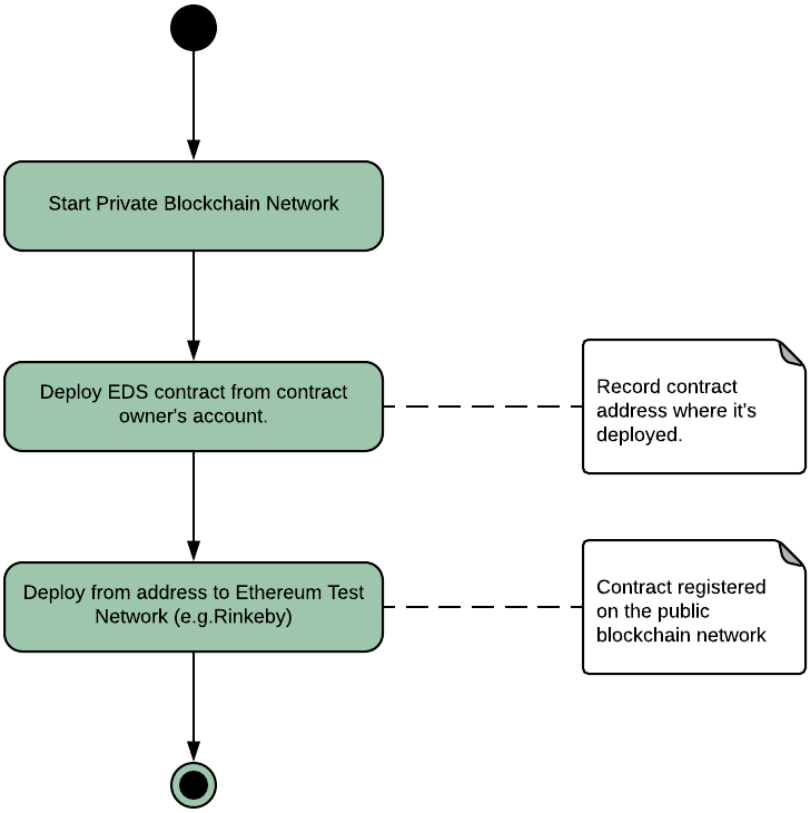

Figure 4.5: Startup activity for registering contract on the network

queries addresses from Participants and is used for storage of data along with CRUD operations on them.

- Computation: inherits from Endorsement contract and allows anyone to get the final score by accessing the current state from an Endorsement contract.
- Marketplace: stores the buyers, sellers information and allows them to buy or sell the product. Also, allows buyer/seller to compute the score of the involved entity before doing a transaction.

'Marketplace' contract was written to test the endorsement network on a transactional network. However, when deploying endorsement system in the real world, other transactional network/online systems are assumed to have their reputation platform. The reputation platform should have assigned a score to the corresponding users based on the behavior on that network. The endorsement system can act as additional conformity for deciding on a transaction. Say, Alice is registered on Endorsement network and has made a decent score. If she wants to sell a product on Marketplace(or any other transactional network), she can claim about her score and anyone who wants to buy from Alice can verify the claim by checking the score that corresponds to her public address. If both Alice and buyer are registered on the endorsement network on the blockchain, they can send a pre-transaction message to each other to verify that Alice is who she claims to be and vice-versa. In case the buyer is not registered on the endorsement network then Alice can prove the claim by signing a cryptographic challenge with her private key.

4.4.3 Data and variables on and off blockchain

For this PoC, the data required to identify the users are stored on the blockchain. But, it does preserve the anonymity requirement mentioned in section 4.1.1, as the only public information is the link between the public key hash and individual trust score. Even though the users are required to register with a pseudonym, it is not needed that the aliases be linked to real-world identity. A user might like to share more information(other online account ids, address, etc.). As mentioned earlier in the gas consumption section, every non-zero byte data or code of a transaction costs a certain amount of gas. (cite: eth yellow paper)(cite: eth gas station chart on the gas section above). Storing this data can become an expensive operation for real-world usage. The right approach can be to use an off-blockchain storage solution such as IPFS, swarm (cite). The hash that points to the file in IPFS can then be stored on the blockchain. Generally, client-side assets (HTML, js) are stored on these distributed off-chain file system that can communicate to the contracts registered on blockchain network.

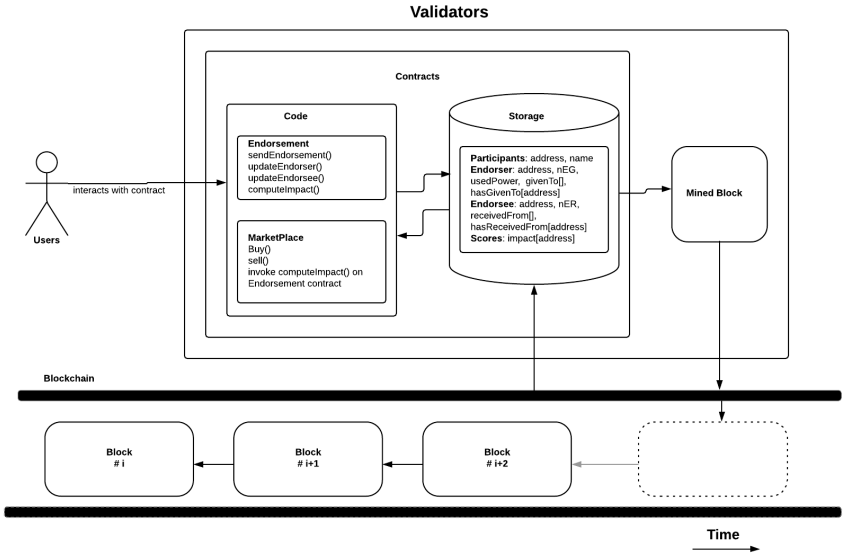

Figure 4.6: Smart contract system

4.4.4 Blockchain and Consensus algorithms

4.5 Trust Metrics

Every node keeps track of its neighbouring node and whenever an intera

4.6 Experimental Setup

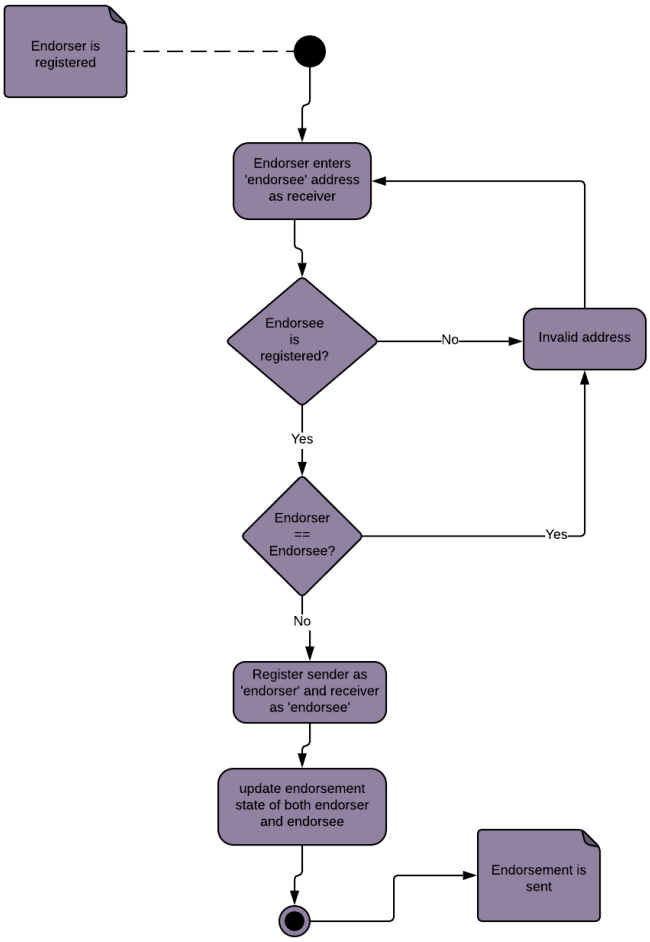

Figure 4.7: Activity Diagram for sending an endorsement

5 Results

5.1 Interaction graph

A participant sends an endorsement to their acquaintances. There is no absolute way to find out if it's an honest interaction or not. i.e., if the participating nodes are the distinct or pseudonymous identity of the same node. The only information visible about the participants on the network are the public address and information they chose to disclose to selected members. Therefore, an ideal way is to view the endorsements as an interaction graph where nodes are entities participating and edges define the interaction along with the direction. Graph algorithms can help to determine the anamoly in the network. Using Net flow rate convergence, anamoly detection is simplified and explained further.

5.2 Analysis

5.3 Measurement

5.4 Comparison

6 Discussion & Analysis

6.1 Generalization

6.2 first section

The results presented in Chapter 4 are discussed and analyzed, including comments and reflections from the author. It may include the following: Comparison of obtained results with discussion,interpretation and evaluation of results. Results of analysis or modeling are described. Interpretations are drawn and connected to previous work

7 Conclusion

7.1 first section

7.1.1 first subsection

Synopsis of findings, limitations, further proposals for future work on the subject. Clear conclusions are drawn that stem from the previous analysis. Present the conclusions drawn and the evidence and arguments that support the conclusions.
Do not include new findings, but only refer to results already discussed in the thesis. Relevant further work in the field is summarized.

Literature

[1] C. Castelfranchi and R. Falcone, "Trust and control: A dialectic link", *Applied Artificial Intelligence*, vol. 14, no. 8, pp. 799–823, 2000.

[2] J. Sabater and C. Sierra, "Review on computational trust and reputation models", *Artificial Intelligence Review*, vol. 24, no. 1, pp. 33–60, 2005.

[3] J. A. Bondy, U. S. R. Murty, *et al.*, *Graph theory with applications*. Citeseer, 1976, vol. 290.

[4] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina, "The eigentrust algorithm for reputation management in p2p networks", in *Proceedings of the 12th international conference on World Wide Web*, ACM, 2003, pp. 640–651.

[5] R. A. Hill and R. I. Dunbar, "Social network size in humans", *Human nature*, vol. 14, no. 1, pp. 53–72, 2003.