Trust and Reputation Mechanisms for the CrowdBED Platform

RAFAIL LOIZOU

UNIVESITY OF CYPRUS

DEPARTMENT OF COMPUTER SCIENCE

JUNE 2020

UNIVERSITY OF CYPRUS

DEPARTMENT OF COMPUTER SCIENCE

Trust and Reputation Mechanisms for the CrowdBED Platform

Rafail Loizou

Thesis Supervisor

Dr. Chryssis Georgiou

Submitted to the Department of Computer Science in partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science at the University of Cyprus

June 2020

Acknowledgments

First and foremost, I would like to thank Professor Chryssis Georgiou for the opportunity to work under his supervision and also for the continuous support and guidelines.

This thesis allowed me to study some of the most trending technologies in computer science like blockchains, trust and reputation systems and crowdsourcing platforms and thus attain knowledge that will be most definitely useful in the future.

ABSTRACT

Trust and reputation management systems are used in variety of online applications; market-places, social networks, expert sites, crowdsourcing platforms, search engines, etc. and are essential for the functionality of the application they are supporting since they are responsible for confronting potential malicious activity, incentivizing good behavior and overall create a better community within the network that the application lives. At the same time, the blockchain technology brought to the world the ideas of consensus and transparency within a community, in a more robust form than before, by utilizing the advances in cryptography. These ideas enable the blockchain technology to be applicable almost everywhere; cryptocurrencies (e.g. Bitcoin), agriculture, charity, healthcare etc. Crowdsourcing platforms like Amazon's MTurk do not leverage on the blockchain technology so they do not achieve transparency within their community. The CrowdBED platform is a distributed crowdsourcing approach of a platform that leverages on blockchain technology to achieve reliability and decentralization. With the approaches of the trust and reputation management schemes that are proposed in this work we aim to leverage on the blockchain technology that CrowdBED provides in the best way possible to achieve the goals of CrowdBED as well as the goals of a trust and reputation system. In this work, after the bibliographic study of several proposed academic reputation schemes as well as online reputation systems, two separate schemes are proposed for trust and reputation management in the CrowdBED platform.

Table of Contents

Chapter 1

Introduction

1.1 Motivation for the Work

It is impossible not to notice how the blockchain technology is taking over all the various sectors of applications. A blockchain is essentially a database that consists of blocks of data. Unlike a traditional database, each new block of data is appended to the end of the blockchain only after all the nodes of the network, that the blockchain works on top of, have reached a consensus on the new block being appended to the blockchain. The consensus is based on a proof protocol (e.g. proof of work used in Bitcoin) which dictates the node that will append the new block. All the nodes of the network have a copy of the blockchain. All those copies are the same up to a certain block. Transparency of the transactions that take place within a community is vital for an economy to function and thrive. So, blockchains are important because of the transparency that they provide to a network as well as the distributed (i.e., everyone in the network has it) and immutable (i.e., cannot be changed by anyone) nature of the database they support.

Crowdsourcing platforms such as Amazon’s MTurk [43] are nowadays on the rise and this can be attributed to the fact that the provision of goods and services is being transferred to the online world. Crowdsourcing platforms are platforms with the objective to allow entities that seek to receive goods or services and are willing to provide a payment in return, to reach out and connect with other entities that are willing to provide the requested service for a reward. The various goods and services that one can search or provide in some Crowdsourcing platform can vary between different platforms. The usual model of work of Crowdsourcing platforms is that the entity that needs the good or service advertises its need to the network of the platform and the interested users respond, either to negotiate terms or to immediately assume the responsibility of providing the requested service, depending on the platform's mechanisms.

At the same time, the use of trust and reputation systems to govern a lot of online applications has been increased. A trust and reputation system is the collection of: (1) data that are used to describe how confident the system is about an entity within a community not being malicious (i.e., being willing to hurt the community for self-gain or even to attack it in general for no reason) and (2) mechanisms that will use the aforementioned data to firstly prevent the malicious entities from acting against the system successfully and secondly incentivize the members of the community not to be malicious, goals that if they are achieved the community governed by the trust and reputation system will grow and thrive. To give a better understanding of how a trust and reputation works the Master-Worker paradigm [42] will be used. In Master-Worker task computing systems the Master is the requester (i.e., requests a task's solution) and the Worker is the service provider (i.e., provides the result computed for the task). So, the aim of a reputation system for a Master-Worker task computing system is; (A) For the worker: (1) use trust values to describe the probability that a Worker will return the correct task result (i.e., is reliable) (2) prevent Workers from acting maliciously by not providing the correct task result and incentivize them to provide their services to provide correct task results and (B) For the Master: (1) use trust values to describe the probability that a Master will assign tasks that can be solved before the deadline the Master have set (2) prevent Masters from assigning tasks that cannot be solved in general or before the deadline set and incentivizing the Masters to assign tasks that can be solved before the deadline set.

CrowdBED [50] proposes the innovating joining of the aforementioned concepts in order to achieve reliability and decentralization, which we do not find in the currently widely used crowdsourcing platforms such as MTurk [43]. This innovation of CrowdBED motivates us to design trust and reputation management schemes that can work on top of its platform to secure reliability and decentralization in its community.

1.2 Goal of the Work and Contribution

First of all, a review is provided for each scheme/system, that is studied in this work. So, a collective idea is presented about the current state of the art, regarding trust and reputation management systems in the various online applications. The bibliographic study is done on a diverse set of schemes that are different in various sectors such as goal, topology, technology leveraged, etc.

In this work, two separate approaches (both of centralized mechanisms) are proposed for a trust and reputation management system, that will work on top of the CrowdBED platform; one with permissioned worker participation and one with permissionless worker participation. These approaches apart from leveraging on the provided platform also adopt ideas from the proposed scientific schemes and the online reputation systems that are studied, in order to better defend against the possible attacks and behaviors that malicious users of the platform might demonstrate.

There are no widely used crowdsourcing platforms that were designed with the aim of achieving reliability and decentralization within their communities, such as CrowdBED. Thus, no trust and reputation management systems that serve the purpose of working with this specific type of platforms, as CrowdBED, have been designed. So, we are called to design such systems that will leverage on the transparency that is provided by the blockchain and will also be resistant against the various types maliciousness that may appear in such a platform. So, the novelty of the mechanisms that are proposed is first of all the application where they are used as explained before and secondly the many weapons with which these mechanisms are equipped to deal with the various possible attacks.

1.3 Approach

First of all, a preliminary study was done of various schemes in order to detect the basics of a trust and reputation system with which a framework was designed afterwards. Then academic papers of proposed schemes as well as sources that analyzed live reputation systems were read. After this reading the studied schemes and systems were mapped to the framework that was designed and at the same time the approaches that would be proposed were formulated. In the end the reputation schemes for the CrowdBED platform are proposed and their functionality is showcased with scenarios.

1.4 Document Organization

What follows in brief is: In Chapter 2 the methodology is presented; explanation of how the resources were chosen for the study and suggestion of a framework that was later used for the mapping of the reputation schemes that were studied. In Chapter 3 we present an overview of the CrowdBED system for the reader to be able to form an abstract idea of the system and thus understand the proposed mechanisms. In Chapters 4 and 5 the extensive bibliographic study is presented. The study has been broken into two chapters to separate individually studied schemes proposed in academic papers from systems that were studied from surveys that presented a collection of them. In Chapter 6 and 7 the centralized mechanisms with permissioned and permissionless worker participation are presented along with scenarios that demonstrate the functionality of the mechanisms. Finally, in Chapter 8 a general discussion is given and directions for future work are suggested.

Chapter 2

Methodology

In this chapter, we first explain the goal we had in mind when the various resources for the bibliographic study were chosen and in which this has influenced the structure of the study. We explain how the survey covers a lot of diverse options in various sectors (i.e., topology, application etc.). Also, it is mentioned how the resources and papers were collected. Secondly, we suggest a mapping framework which is based on a preliminary study of schemes and is used when the bibliographic study is carried out.

2.1 Choosing and Collecting the Resources and Papers

Firstly, individual schemes were studied [1-9] that considered different topologies (P2P Grids [1,7], Mobile P2P networks [2], Unstructured P2P [3,4,8], SOPSys [6], VANETs [9]) so that a better understanding of how the network's topology dictates the decisions that must be made when designing a governing trust and reputation system, was formed. Also, the individual schemes were chosen is such a way that in different schemes, the authors had different priorities of what they needed to achieve with their work (e.g. RCert [8] prioritized in solving the problems of storage and integrity of reputation while the M-Trust's [2] primary goal was to be scalable and lightweight), so that the study would be even more diverse.

Afterwards two surveys [10] and [20] were studied. In these surveys various other schemes were mentioned, analyzed fully or just partly for the specific mechanisms that they used. The first study focused on the propagation methods (i.e. how reputation is exchanged from user to user) and how opinions worked within the schemes. The latter gave a more general description of all the aspects of a trust and reputation system, categorized schemes/systems based on the models of reputation computation that they applied and then analyzed various commercial and live reputation systems.

Then paper [42] was studied, because of the fact that in our case for each transaction we will too have a master and a worker, so this paper would expose to us the work that has be done with such systems that involve masters and workers and would also provide a scheme for us to study that is more related, to the application that our systems will be designed to work for, than the previously studied schemes.

Finally, we studied [49] which made sense to our work since it was showcasing the use of reputation to incentivize good behavior and maintain the ledger, so it combined two concepts that appear in our goal as well.

All the papers used for this work were either provided by my supervisors or were freely available online. In the case that, for a scheme there were no papers available or not published so that a study could be carried out, this is noted in our work and in the case that the survey that mentioned the specific scheme gave any information on it then this information would be utilized in this work.

2.2 Suggesting a Framework to Map Trust and Reputation Management Systems

Here firstly the two main parts of a trust and reputation system are included in the framework: the variable that will contain the reputation information for each network entity and the method in which this variable will be updated. Then the secondary properties are included, and these are the topology where the system works and the system’s strengths and weaknesses both from its authors perspective and from a critical point of view. Now we get into more detail and describe the parts in order.

The variable v_i: represents the overall collection of values that describe the trust and/or reputation of a member/party/entity labeled ***i*** in the network. There are several aspects of this variable we must determine when we design a reputation system:

Perception of the variable's value that different entities have:

We define that v_{yx} is the value of the variable v_y that entity x will see/know for entity y. We need to determine whether each entity of the network will maintain its separate value for v_i or the same as every other entity in the network, thus we have two perspectives:

(a) Global: Means that for any entity A and entities B and C where $(A \neq B, A \neq C, C \neq B)$ in the network, then v_{AC} must be equal to v_{AB}.

(b) Local: in the same example as above v_{AC} and v_{AB} can be different. Here we can also consider a case where a local perspective will not refer to a single entity but to a neighborhood of nodes, thus if B and C where $(B \neq C)$ are in the same neighborhood that has its own local perspective then for any entity A v_{AC} must be equal to v_{AB}.

Visibility and Sharing of the variable:

Here we need to know for any entity A if v_A will be visible to some entity in the network. By visible we mean that the entity will be able to form a perception of its own for v_A based on information that it will either receive from other entities or from direct interaction with entity A (i.e. if v_A is visible to entity B then v_{AB} exists, otherwise v_{AB} is null). Since sharing is closely related to the concept of visibility, we will also define properties of sharing that will apply to different levels of visibility. For some entities A, B and C where $(A \neq B, A \neq C, C \neq B)$ we have:

Symmetric property: That is, if v_A is shared to B and thus v_{AB} exists then B will share v_B with A so v_{BA} exists as well.

Transitive property: That is, if v_A is shared to B and thus v_{AB} exists AND v_B is shared to C thus v_{BC} exists, then B will share v_{AB} with C.

So, we will define four levels of visibility. For any entity A:

(a) Public: v_A is visible to all entities in the network. So, it is globally shared. So here both the transitive property and symmetric property exists (if it is public you must share it with everybody otherwise it is not public).

(b) Private: v_A is hidden to all entities in the network except to the system itself that needs it for calculations. Absolutely no sharing.

(c) Privilege depended: v_A is visible to some entities only (e.g. entities with higher status in the network).

(d) Other combinations of sharing properties: For example, entities are in a linear topology where they only share their knowledge of v_A (for any A) with their next neighbor, so here we have the transitive property but not the symmetric property.

Structure and Scale of the variable:

When we talk about the structure of the variable, we mean to list the various parameters for which a rating is received and that comprise the variable. For example, in an e-commerce system an item could receive ratings for price and quality so we would define the parameters of the variable v_i which would be v_i.price_rating and v_i.quality_rating. We also need to define the scale of every such parameter. It is important to know whether a rating is scaled to real numbers in a range (a,b) or to some number of stars from 1 to 5 or to phrases (e.g. "bad", "good", "very good" etc.), because this would matter when we talk about how the system calculates ratings. Some calculations can be applied to ratings that follow a certain scale and some can't. For example, it would make sense to find the average of ratings that are real numbers scaled in the range of (a,b) but if ratings are in the form of standard phrases it would not make sense to find the average value between "very good" and "very bad". I would rather find which of the two values occurs the most.

Note there are combinations of the above aspects that wouldn't make sense such as Global perspective and Private visibility. Thus, we don't have to consider every single combination and we can see that each choice we make for a certain aspect has direct influence on the subsequent choices we will make for the other aspects.

The method for calculation/updating of the variables: How will the system produce from a variable v_A (that represents the old state of trust and reputation for entity A) the variable v_A' (that will represent the new state) after a certain time or event? To define how this method will work we need to describe two essential steps that need to take place: the rating and the aggregation of ratings after which the new value(s) of the variable are ready.

Rating:

Who gets to have a word on how some entity's reputation value changes?

(a) All: Everyone will rate everyone.

(b) Only entities that had some past interaction (here we can choose to include only direct interactions or also include non-direct interactions) with that entity will rate (we might as well as set a limit of how old this interaction can be).

(c) Random: A random group of entities will be chosen each time to provide a rating.

(d) A local group of entities: If we consider that one entity belongs to the same local group of entities as the entity under question then it will rate.

When does rating occur?

(a) As soon as the entity interacts with some other entity then there is a rating given.

(b) After certain (or random, or system calculated) time a rating round will take place.

(c) After a certain number of interactions has taken place in the network.

Ratings aggregation:

Who will collect the ratings and do the aggregation?

(a) All: Everyone will aggregate ratings.

(b) Certain group of entities (or one entity).

(c) Random group of entities each time (or one entity).

Will everybody aggregate in the same way?

(a) No, everybody can aggregate in the way he chooses.

(b) Yes, certain way of aggregating must be followed.

If everybody aggregates in the same way, the what way is that?

There are a lot of formulas that can be applied. From averaging the votes, to weighting votes based on the reputation values of the voter, to counting hops from voter to the entity receiving the vote etc. Some reputation systems when aggregating involve a lot more than just a single formula.

The topology of the network in which the scheme/system works: Some reputation systems can work only with certain topologies (e.g. P2P grid, mesh, centralized topology etc.). If a certain reputation system works with a specific topology only, then studying it would be a waste of time if we are trying to implement it to another topology without changing it. Also, whether a system is set on a centralized architecture or not should matter because a centralized architecture means a central authority exists that can oversee everything since members need to first communicate with the central node so they can communicate with the other members of the community.

Strengths and Weaknesses of the scheme/system: These need to be listed so we can use them in the future to decide how will we try to implement efficient and working trust and reputation management scheme approaches based on the systems/schemes that will be mapped in this study.

Chapter 3

Overview of the CrowdBED Platform

In this chapter we give an overview of the CrowdBED [50] platform. Firstly, an outline is presented to give a better understanding of how transactions work within the CrowdBED platform and what are the basic processes that comprise the transaction. Then the basic architectures of the blockchain and the crowdsourcing platform are presented in figures. Finally, the position that the trust and reputation schemes, that will be proposed, will have within the system will be explained by listing the roles and responsibilities that they will have.

3.1 The Outline of a Transaction in CrowdBED

3.1.1 The Roles of a User within a Transaction

For some transaction *T* and a user *U* a user can either not participate in the transaction or participate with one of the following roles:

(1) MASTER: This is the user that has a task for which he needs an answer (for which answer he is willing to pay) and uses the platform to get the answer.
(2) WORKER: The user that possesses computational resources which he is willing to utilize to solve a task that the platform will provide for a reward.

Based on the definition of these roles, T will have one master and multiple workers.

3.1.2 Phases of a Transaction

First of all, here we will specify the kind of tasks we are focusing on. The schemes that will work with CrowdBED do not work with any possible family of tasks. Here we focus on microtask crowdsourcing [51]. Microtask crowdsourcing essentially means that the main task, that the master needs solved, is broken down into batches of smaller and quicker tasks called microtasks that are then given to the worker population to be solved and returned to the master.

The basic steps for a transaction to be carried out, simplified, are the following:

(1) The master submits the task that needs to be solved to the platform.
(2) The task is advertised to the network.
(3) The set of workers that have expressed interest in solving the task is formed.
(4) A selection process takes place to select the workers that will solve the task.
(5) Selected workers solve the task and submit their answers.
(6) The correct answer is selected form the set of provided answers.
(7) The users involved in the transaction receive the respective reward/punishment based on the correct answer.
(8) The master is provided with the correct answer.

3.2 The Architecture of CrowdBED by its Components

Here figures are used to present the simplified architectures of the system so that later it can be explained what parts of the system the proposed mechanisms will aim to influence to achieve trust and reputation management.

3.2.1 Crowdsourcing

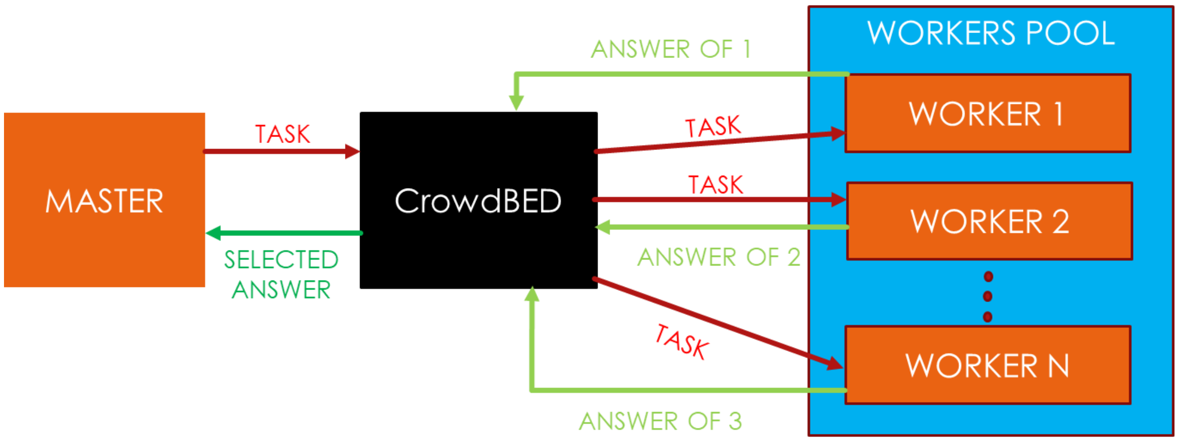

Figure 3.1: The architecture of crowdsourcing in CrowdBED

3.2.2 Blockchain

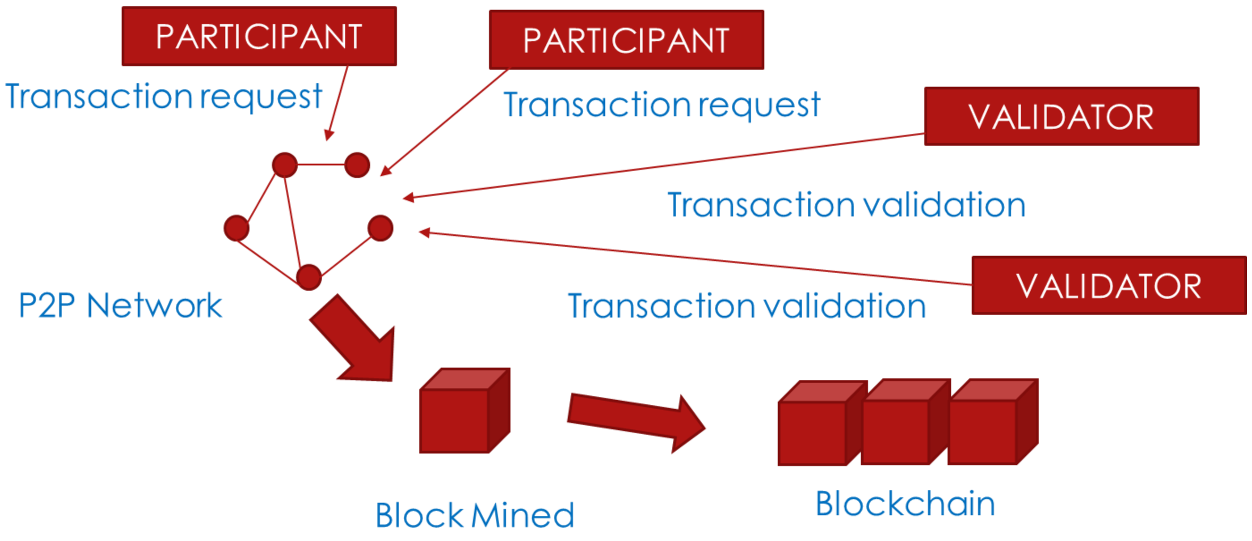

Figure 3.2: The blockchain architecture used in CrowdBED

From Figure 3.2 we can see that in order for the next block to be mined (i.e., be appended to the most recent blockchain version) we need a transaction request to take place so that the transaction is carried out and also a transaction validation by a validator so that the transaction is validated before the transaction data is ready to constitute the block. The validator is checking the important condition that the transaction has valid information (e.g. to be created from an authorized participant with the right permissions) before it can be recorded on a block.

3.3 The Role and Responsibility of the Proposed Mechanisms in CrowdBED

Now based on the architecture presented in the previous section, the roles and responsibilities of the mechanisms that will be proposed, are listed:

(1) As seen from Figure 3.1, after the MASTER submits the task to the platform the trust and reputation system must be used to create the WORKERS POOL (i.e. choose which of the interested workers will be assigned with the task).
(2) As seen in Figure 3.1, after the WORKERS submit their answers to the platform, one of these answers is selected to be returned to the MASTER. The mechanism here will be responsible to determine how will this answer be selected.
(3) A reward/punishment scheme must be provided by the trust and reputation management mechanism. First of all, for a transaction, after the answer to be returned to the MASTER is selected the mechanism will be responsible to reward/punish the involved users in the transaction under question in a fair way. Secondly the mechanism must reward/punish based on various other events and facts that are deemed important and not just solely after the end of a transaction.
(4) The mechanism must also determine how will the users be treated when they join the network (i.e. what will the initial reputation values be, if there will be specific policies for them for a certain time etc.).

Chapter 4

Individually Studied Schemes Proposed in Academic Papers

The following reputation schemes were studied individually. For each one of the schemes we present firstly the mapping to the framework that was defined in Section 2.2 and then the formulation of the mechanisms for the CrowdBED platform takes place. This formulation will be based on elements extracted from the mapping and so each scheme will contribute to the schemes that will be finally proposed for CrowdBED.

The purpose of this study and mapping of these reputation management schemes which is to formulate two new reputation management schemes, one permissioned and one permissionless, that will be used in the CrowdBED platform, was the factor that guided the study. Thus, after each mapping, a decision will be made, using critical thinking and self-debate, on whether certain characteristics/techniques/properties of the scheme under question could be used in the case of CrowdBED, for the reasons explained in Section 3.3, in a way that will improve the system we are trying to design.

4.1 H-Trust

H-trust [1] is a robust group trust management system.

This system works in 5 stages that are listed below for later reference:

1) Trust recording:

Record the information of the past services in a service history table which is maintained in the DHT-based overlay network.

2) Local trust evaluation:

A local trust score is calculated by a local trust manager using weighted trust aggregation algorithm.

3) Trust query phase:

Required when the trust information is not available locally.

4) Spatial-temporal update phase:

Activated periodically to renew the local trust scores and credibility factors.

5) Group reputation evaluation phase:

Aggregation of the group reputation using the H-trust algorithm.

The variable v_i:

Structure and Scale of the variable:

The variable here will have two main parameters:

- v_i.individual : represents the individual trust value of the peer.
- v_i.group : represents the group trust value of the peer.

Secondary parameters:

- v_i.group_reputation : used by a job distributor to assign jobs to work groups.
- v_i.credibility_factor : separate parameter for each peer that is used only in certain stages when the overall reputation is updated. The credibility factor is an integer in the range (0-10) and for new peers is set to 5.
- v_i.LSHT : the Local Service History Table (LSHT) is an object that we must consider as part of the variable. The Local Service History Table (LSHT) is used to record the history of past

transactions or services information in each peer. The trust rating for one peer towards the other is based on statistics collected form its LSHT. It is important to mention that in these tables not only the service quality is noted for each job but also the service importance which is used by the scheme to weigh the feedback from that service. Let's declare these sub-parameters:

○ v_i.LSHT(transaction_id,service_peer_id).$_{\text{quality}}$: integer from 1 to 5.

○ v_i.LSHT(transaction_id,service_peer_id).$_{\text{importance}}$: integer from 1 to 5.

Perception of the variable:

Personalized trust rating system; Trust is subjective and reflects an individual's opinion. So, we have local to the entity perception of the variable. Specifically, for the LSHT it is mentioned that every peer maintains an LSHT for the other peers which it had interactions with before, so a local to the peer perception exists.

Visibility and sharing of the variable:

• v_i.individual : is kept as a record in each peer's LTRT (Local trust ratings table). So, if a peer wants v_i.individual as seen by peer *j*, he will query *j* for v_{ij}.individual in *j*'s LTRT. The opinion that will be shared though might be false statements.

• v_i.LSHT : stores data that have to do with peers that the tables owner interacted with before. So, if no interactions happened between two peers then no data will be stored for each other in their LSHTs, thus we can say that v_i.LSHT has a visibility depending on whether interaction happened in the past.

• v_i.credibility : The results of the past references (in the trust query phase) of other peers are recorded in every peer's LCRT (local credibility rating table) which is managed as the LSHT. So, for the v_i.credibility we can say the same thing we said for the v_i.individual.

The method for calculation/updating of the variables:

Rating:

• In the trust query phase, the system relies on collective opinions from other peers. In this phase the whole network is queried yet using the credibility factor part of the reputation variable, that we mentioned above, we consider only opinions from partially qualified peers. A threshold T, which is based on the network's environment, helps to set a boundary on the responses to be considered in a queried trust aggregation. So, from the opinions that a querying peer receives only opinions from peers whose credibility ratings are greater than T are selected. So, opinions from all the peers are received but some are filtered out.

- All the peers give a rating in the group evaluation phase.

Ratings aggregation:

- Personalized reputation evaluation mechanism: Each peer stores its own Local History Table and may use different local inference algorithms to derive the trust values. So, everybody can aggregate in the way he chooses in the local trust evaluation phase.
- After opinions from peers that we consider dishonest are filtered out in the trust query phase we proceed to the trust aggregation phase. Here we use selective aggregation technique and the opinions are aggregated into local trust ratings using H-index aggregation. So, in the trust query phase everybody aggregates in the same standard way which is the H-trust aggregation algorithm.
- After a service cycle is completed the trust rating and credibility factors are updated. That is the service is recorded in the LSHT of the start peer and the LTRT and LCRT are updated. In the LCRT for all the peers that responded in the trust query phase, selected or not selected, we will increase or decrease their rating if they gave a correct recommendation or not respectively.
- In the group reputation evaluation phase, we aggregate the group reputation. Here the H-trust group reputation algorithm is used.
- When selecting a group to assign the job to if we come across draws between group reputation values then usually the trust rating is used as a tiebreaker (individual reputation) but not always.

The topology of the network in which the scheme/system works:

Peer-to-Peer Desktop Grid (P2PDG): Peer-to-Peer Desktop Grid (P2PDG) has emerged as a pervasive cyber-infrastructure tackling many large-scale applications with high impacts, such as SETI@Home and DNA@Home.

<u>Strengths and Weaknesses of the scheme/system:</u>

<u>From the system's author view:</u> Medium convergence speed. Medium message overhead. Low computation overhead. Simulation results demonstrate that H-Trust is robust and can identify and isolate malicious peers in large scale systems even when a large portion of peers are malicious. Malicious peer detection rate more than 99%.
<u>From a critical point of view:</u> This system managed a very good percentage of malicious peer detection and that can be attributed to the fact that it uses both personalized trust and collective opinions as well as to fact that peers are also separately rewarded/punished based on the recommendations they are giving for other users. This robustness strongly incentivizes peers not to be malicious but comes with the costs of memory overheads for each table a user must maintain and the fact that all users can give their opinion means that a lot of messages will have to travel through the network.

<u>Formulation of CrowdBED trust and reputation management systems from this system/scheme:</u>

Group trust in the case of CrowdBED would only make sense if members of the network could form groups before they join CrowdBED (so they join as a group) or while they are participating in the system decide to create a group. In CrowdBED neither situation can occur; we don't have groups but individuals, so we can't elaborate in the idea of group trust.

The credibility measure could be used in CrowdBED as well and will help users to decide to listen to one's opinion or not. Here the credibility measure is a separate parameter of the variable. In our case we could just use the overall rating of a client when deciding to listen to his opinion or not, but this would mix what the two values are describing and cause the loss of feedback. For example, a user might be a good worker but would give bad recommendations or the opposite might occur; a user is a bad worker but will provide correct feedback. So, it would be better that we keep these two concepts in separated values just like this scheme does.

This scheme uses a Local Service History Table to maintain transaction info which would not make sense in our case since in CrowdBED all transactions information is stored in the public ledger. If a user wants to maintain his own opinions, for other users, he can do so but the system will keep in the ledger only global values (no extra hidden data is kept so that it is used just by one user individually).

Just as in H-trust the trust rating is based on statistics on the LSHT the same could be done with CrowdBED whereas in this case it would be based on statistics from the ledger.

H-trust does not only use a service quality rating but also considers the importance of the service. We could choose to use this idea as well, but we must do so carefully, because considering a task's importance and difficulty might provide more accurate results but also can create ways in which the system can be manipulated (e.g. A master chooses to label a task as of high importance when assigning it to a user he wants to elevate and of low importance when assigning it to a user he doesn't want to elevate). So, if we were to adapt this idea to our case then the measure, related to the task, that would be chosen to influence the rewards/punishments, should be such that it can be verified by both the system and the community as well (e.g. size of task).

H-trust uses personalized trust (Local Trust Rating Tables) which we could implement in the ledger, but all personal values will also be publicly visible since they are in the ledger.

H-Trust uses collective opinions. This idea would make sense in our case as well especially because we are using a ledger. Collective opinions would lead us to not hearing opinions from just a group of users but from the whole population and thus it is possible that greater accuracy is achieved. The only question is if we will use a credibility factor to either weight these opinions or choose to completely ignore or consider them as H-trust does. If we choose to completely ignore or consider them then a lot of opinions from not so credible users would be lost and if most users are not credible enough for their opinions to be considered then this could lead to having some users that managed to reach the top tiers of credibility and thus enabling them to manipulate the system if they cooperate. So, weighing the opinions seems to be a better strategy than trying to continuously adjust the right credibility threshold so that the situation described before does not occur.

H-trust updates trust ratings as soon as a transaction is done which is something we must choose to do as well so that scores are always up to date so that the system is more accurate.

H-trust also rewards or punishes users that gave a correct or incorrect recommendation respectively and that is a good idea that we should also adopt but we must make sure that the reward or punishment given will not create imbalances in our system. A good idea would be to add a streak factor that should be considered after doing such updates. For example, if a user gave 5 correct recommendations in a row then he should receive a greater reward than one that just gave his first correct recommendation. This could incentivize the users to continually give correct recommendations.

4.2 M-Trust

M-Trust [2] is a robust, scalable and lightweight distributed reputation aggregation and trust management scheme for mobile P2P networks.

The variable v_i:

Structure and Scale of the variable:

All parameter values are real numbers in the (0,1) range.

- v_i.rating : the overall rating.
- v_i.confidence : the value of confidence which is used for deciding whether witness recommendations coming from the peer are legit.

Perception of the variable:

- v_i.rating : Local to the peer, each peer has his own “t_list” where the ratings are stored.
- v_i.confidence : The confidence values are also local since they are defined by the personal interactions of each peer.

Visibility and sharing of the variable:

The trust ratings are publicly computable since, as the scheme suggests, if a peer *X* wants the rating of a peer *Y* then he can compute it using the local trust ratings of peers that he already has and a mean aggregation error value.

<u>The method for calculation/updating of the variables:</u>

<u>Rating:</u>

• When it comes to parameter v_i.confidence the peer that will calculate it will use his own interactions, so it is based on his own rating of the peer under question.

• When rating only direct peers that are trusted rate. As trusted peers we define those that have a rating value higher than a threshold.

• For witness interactions only distant peers will rate and not the direct neighbors. A distant peer's rating to be considered his confidence value needs to be greater than a threshold.

<u>Ratings aggregation:</u>

• The value of confidence is calculated by individual peers based on the number of positive and negative interactions with target peers with the formula: $\mathrm{v_i}.confidence = \mathrm{abs}((x - y)/x)$ where *x* the positive and *y* the negative interactions. So, everybody aggregates in the same way when it comes to parameter v_i.confidence.

• Ratings that are given by direct trusted peers are aggregated using a standard algorithm.

• If there are no trusted direct peers, then rating is based on witness interactions. When using witness interactions, the ratings are aggregated using a standard algorithm that uses weighted averages and the confidence values of the raters.

<u>The topology of the network in which the scheme/system works:</u>

Mobile P2P networks. P2P over MANETs (Mobile Ad-hoc networks)

<u>Strengths and Weaknesses of the scheme/system:</u>

From the system’s author view: The results demonstrate that M-trust gives an excellent overall performance in terms of accuracy, reliability, convergence speed, and detection rate under various constraints of mobility, trust threshold and network out-degree.
From a critical point of view: This scheme is lightweight because it reduces computation as much as it can which we can see from how trust values are calculated. We can see that this lightweight cost approach is also chosen because of the topology where the system is working which is mobile P2P networks where calculations and messages should be reduced as much as possible while not disabling the system from working properly. Of course, this approach will not provide the same accuracy and will not detect malicious peers as well as a system that takes ratings from all peers each time would. But for the network topology, that this system works, this is the best outcome we could expect because otherwise the message congestion would make the system very heavy and not appropriate for this topology. Also, this scheme uses the t_list object which works in a way that reduces the memory usage, because records on t_list will exist only when they will have a use (i.e. records for trust information about a peer, that will never matter in the calculations a peer will make, will never exist).

Formulation of CrowdBED trust and reputation management systems from this system/scheme:

Again, here just as in H-trust a separate parameter of the trust variable is used for opinion trusting. Here a peer forms its own perception of another peer's opinion credibility using as input only interactions that it had with that peer. In our case, and as we had decided before from when studying the H-trust scheme, it would be better if we had a global credibility value for each peer and update that value with every recommendation that the user might provide so localizing this value to each peer will cause the system to lose this constant and rich feedback that can be extracted from every recommendation the user gives for each transaction. Also, M-trust bases credibility to the interactions that took place between users, while it intuitively seems that it would better to rate one’s opinion credibility based on past opinions that he gave and how credible these opinions have been.

Peers in M-trust have their own perception of other peers’ trust variable thus they maintain local values for all trust variables of other peers. So, in our case if we want to implement this local perception, we will need to include this data in our ledger since this is what our system uses to store data, or we could let users store their own local values if they want. Of course, our global trust values will still be in the ledger.

In M-trust all values are computable and thus public. Same will happen in our case since we have a ledger that deems all information transparent and public. Also, since we have this transparency there is no need for us to use a mean aggregation error value (in the same context that M-trust does).

In M-trust direct trust is mainly used and only in the case of no direct trusted peers witness interactions are used along with the opinion confidence values. We can assume that the author of this scheme either does this to make the scheme more lightweight or to give more gravity to firsthand experiences of a peer (direct interactions) in order to avoid problems associated with malicious peers giving false ratings which could happen if witness interactions are used. But these problems are not of our concern since we are using a global ledger so every peer has access to the ledger and can retrieve transparent information it needs from there. So, following the same idea with M-trust when aggregating would deem all the overhead, that comes with the ledger we are using in CrowdBED, extra and unnecessary, so we will not use this idea.

4.3 R^2-Trust

R^2-Trust [3] is a reputation and risk-based trust management framework for large-scale fully decentralized overlay networks.

<u>The variable v_i:</u>

<u>Structure and Scale of the variable:</u>

We have an overall trust parameter: v_i.overall being calculated from two main parameters:

• v_i.trust : the trust value calculated from sub-parameters:

○ v_i.direct : the direct trust value. The direct trust value is obtained only from direct transactions between two peers. The direct trust value can be calculated for a specific time window so we could go ahead and declare sub-parameters for each time window.

○ v_i.reputation : the reputation value.

• v_i.risk : the risk value.

Weights used in the calculations of the values should also be consider as parameters of the variable since they may differ from peer to peer; α and β used for calculating the overall trust value and play the role of optimism and pessimism respectively and ζ the confidence factor, we declare v_i.weight_a, v_i.weight_b and v_i.weight_z. All weights are real values in the range (0,1).] Also, the scheme uses an additional parameter:

• v_i.credibililty : represents how much a peer is trusted by some other peer on the recommendations that he gives, and it is used as a weight to the referrals that this peer gives. The credibility values are real numbers normalized in the range (0,1) and initially they are set to 0,5.

<u>Perception of the variable:</u>

• v_i.direct : local to the peer since it is calculated from direct and thus personal interactions with a target peer.

• v_i.reputation : local to the peer since it is calculated using referrals from other peers and since different peers may receive different referrals they might as well have different reputation values for some peer.

• v_i.credibililty : local to the peer, since two different peers may have different perceptions of how credible another peer is. Generally, as stated in the paper, every peer has his own personalized view about the community.

<u>Visibility and sharing of the variable:</u>

• v_i.trust : shareable between peers.

• v_i.risk : shareable between peers.

• v_i.credibility : not shared; each peer forms his own perception, namely the credibility value, for another peer and will not share it with other peers.

<u>The method for calculation/updating of the variables:</u>

<u>Rating:</u>

• For the direct trust value, the peer that receives the service will rate the servicing peer.

• The risk values are updated using interaction-driven information so only the participants in an interaction get to rate for the risk value.

Ratings aggregation:

• If peer *X* didn't have a direct transaction with another peer *Y* and thus hasn't a direct trust value (i.e. v_{YX}.direct) then only the reputation value v_{YX}.reputation will be used, otherwise both are used.

• To calculate a reputation value, we aggregate all the referrals from other peers.

• The credibility values are updated accordingly, meaning when a correct or wrong referral is given by a peer *X* to peer *Y* then v_{YX}.credibility will be increased or decreased respectively.

The topology of the network in which the scheme/system works:

Unstructured P2P networks.

Strengths and Weaknesses of the scheme/system:

From the system's author view: "Our experimental results show that, compared to the existing trust models, our model is cleanly a winner when security is the major concern of a system."

From a critical point of view: We can see that this scheme considers a lot of factors when calculating trust. So naturally, the more factors that are considered the more accurate the trust values calculation will be and that would lead in more effective detection of malicious users. But, if we were to compare this scheme, in terms of calculation and memory costs, with other schemes, that are working in a decentralized architecture as well (e.g. M-trust), we would see that it needs greater calculation and memory capacities to be functional. For a P2P network, if the goal is high security with little restrain in use of resources, this system would certainly achieve that goal.

Formulation of CrowdBED trust and reputation management systems from this system/scheme:

Here positive from negative ratings a separated in different parameters of the variable (although there is also a parameter that represents the overall rating). This is a good example to follow because knowing the amount of positive and negative votes will provide a better feedback when calculating trust rather than just knowing the difference between them (positive - negative) or just a ratio (e.g. positives divided by negatives) would.

Here the system uses both direct and witness interaction. Which is what we plan to do as well since we are using a ledger and thus can retrieve information given from both direct (that had an experience with) and indirect users.

Here confidence in one's opinion is used as a weight in calculations of trust and is different for each user and that's what we were planning to do from the two papers that were studied before [1] and [2]. That is, use a value, that represents how much we trust a user's opinion, as a weight when calculating trust values. It is important to highlight again this separation between the trust we have to one to complete a transaction well (i.e. be a good worker) and the trust we have to one's opinion (i.e. be a good informant).

Time windows can be used here when processing direct trust values and the same can be done in our system. If a user wants to see past trust values, he can take the ledger and by negating the last updates, in the opposite order of which they were done, manage to calculate the past trust values. But the only reasons we would use past trust values is to either detect malicious actions that might have caused sudden increase or decrease to one's trust value or to moderate and see if a user has had a negative (or under the maliciousness threshold) trust value in the past. For the case of detecting sudden increases/decreases in a user's trust variable we could either use a past trust value or keep for each K transactions a separate variable and perform this check for sudden changes once every K transactions which seems to be the lighter solution when it comes to memory kept in the ledger. Also, this solution can work even if we decide that the ledger will store data for only V transactions ($V < K$) case in which it is impossible for the look back to history solution to work out but our goal could still be achieved. In addition, if we want to change K to be smaller or greater, we can do so after the next check we perform. So, for detecting sudden changes, an extra variable seems to be the optimal solution as long as K is small or changes randomly and users can't infer when the next check will occur and thus schedule their actions. When it comes to detecting if a user has been malicious in the past we can resolve this with an extra boolean value that will be true if and only if the user has had a negative (or under the acceptable threshold) trust value, or we could just immediately kick any user that goes under this threshold out of the system and thus there would be no need for this boolean.

Again, here all values are localized to the peer and we have gone through before with localizing values and how the transparent ledger affects our decision on this matter.

This scheme chooses to share all but the credibility values between peers. In our case, since we chose to update credibility values, for users that had provided a recommendation, after each transaction has been completed and the majority voting for the answer (or the auditing if that's what we go with and not majority voting) is done as well, then that means that this credibility value can't be hidden since it can be calculated using data that it is stored in the ledger which is transparent and available to all users. If we want credibility values to be accurate then this can only be achieved by considering all possible feedback and so we will choose to have global and public credibility values unlike R^2Trust.

When updating trust values for v_i.direct and v_i.risk, in this scheme, only users involved in the transaction will provide a rating for the opposite party. In the case of CrowdBED the answer a worker gives could be that rating (i.e. if two workers have given the same answer then this is like they have rated positively for each other). So, in this sense, when it comes to updating reputation values based on the transaction only workers will rate for other workers.

Overall this scheme has been designed in a way that a lot of factors are considered when calculating trust so the more accurate the trust values will be. But in our case, with the transparency that the ledger will provide this complexity of using so many different factors just to achieve that extra accuracy in trust calculations seems to be unnecessary. We need to keep the ledger light and we will not overload it with data that will provide something that we already have using the transparency property of the ledger itself. A good example that showcases this point is that we have decided to use credibility already but including optimism and pessimism factors as well would add cost to the calculations the system will carry out and the memory of the ledger without providing any actual benefits.

4.4 EigenTrust

<u>The variable v_i:</u>

<u>Structure and Scale of the variable:</u>

• v_i.gtrust : A global trust value (also named as global reputation values in the paper) that reflects the experiences of all peers with the peer in question.

• v_{XY}.ltrust : a local trust value that represents the ratings of the individual transactions between *X* and *Y* when *Y* is served and *X* services.

<u>Perception of the variable:</u>

• v_i.gtrust : global.

• v_i.ltrust : local.

<u>Visibility and sharing of the variable:</u>

• v_i.gtrust : public.

<u>The method for calculation/updating of the variables:</u>

<u>Rating:</u>

• All peers participate when the global trust values are computed. The global reputation values v_i.gtrust are used to weigh the local trust values that peers will recommend for some peer *X* and then produce v_X.gtrust.

• The peer that receives the service also gives the rating for local direct trust values.

<u>Ratings aggregation:</u>

• The local trust values are normalized before they are being aggregated to produce the global trust value. The local trust values essentially are used to weight the opinions peers give for another peer but are not updated based on the opinions he gave before (they are update based on direct interactions as mentioned above).

The topology of the network in which the scheme/system works:

P2P network.

Strengths and Weaknesses of the scheme/system:

From the system's author view: This system has been shown to significantly decrease malicious transactions even under a variety of conditions where malicious peers cooperate.
From a critical point of view: We can see that this system does indeed use all the information possible when calculating trust but that entails communication costs (each peer sends his local ratings to the other peers to aggregate) and processing costs (all peers participate when the global trust values are computed). So, we could say that this is a secure approach but does not concern with resource management issues.

Formulation of CrowdBED trust and reputation management systems from this system/scheme:

Here v_i has two parameters; The global reputation and the local direct trust. As mentioned, before we choose that we will have a global reputation table but if we want to also add local trust values (not only for direct interactions but for reputation as well if we want to) we could do so, but they will be public since they will be stored in the ledger.

So, this scheme has chosen to make only reputation public while direct trust will be visible only to the peer that maintains it. But we can't do that since we are using a ledger that is public and we will then defeat the purpose of using a ledger in the first place.

We can see that in EigenTrust [4] all peers will give ratings before the aggregation is done. In our case since we update the values just as a transaction is done, we don't need to do that because only users involved in that specific transaction need to have their values updated. When it comes to when will the other users see this update that is something that has to do with how the ledger itself is updated (when consensus is reached). But when it comes to users providing opinions about how trustworthy a user is as a worker (e.g. before the assignment of a task) then we can use collective opinions since these opinions are stored in the ledger.

If we were to adapt this scheme to a centralized architecture approach, we would probably have problems regarding the communication capacity of the central node because when calculating trust all users will have to communicate for this to happen (i.e. sent their opinions to be aggregated). But since we are already using a ledger in CrowdBED we don't need to worry about communication overheads since the values we need to do the calculations are already in the ledger and so the idea of collective opinions can be adapted to our approach so that we achieve better accuracy.

4.5 FIRE

FIRE [5] is an integrated trust and reputation model for open multi-agent systems.

<u>The variable v_i:</u>

<u>Structure and Scale of the variable:</u>

The scheme states that it uses 4 types of trust thus we define 4 parameters of the variable:

- v_i.interaction : interaction trust value.
- v_i.rb : role-based trust value. The paper states that there is no general method for computationally quantifying trust based on this relationship type and so rules are used to assign values instead. The rules used for role-based trust are tuples of data which contain the roles of the two peers *a* and *b*, let *b* be the peer to provide the service, and three values which are a term named *c* which represents the quality, the expected performance *v* of b in an interaction with respect to the specified roles and the *c* term, where v in the range (-1,1), and finally the level of influence *e* which is in the range (0,1).
- v_i.wrep : witness reputation value.
- v_i.crep : certified reputation, which is a collection of certificates for a peer to show in order to convince other peers of his past work.

Also, we define v_i.trust for the overall trust calculated from any types of trust the peer decides to use. The paper suggests a formula that includes all the 4 types of trust. The v_i.trust is a normalized value in the range (-1,1).

Perception of the variable:

• v_i.interaction : the interaction trust values are local to the peer since they are based on personal experiences.

• v_i.rb : The role-based trust values are local to the peer since each agent has its own set of rules stored in a local database.

• v_i.crep : for the certificate trust values we can't decide between local and global perspective since it depends on the peer that puts the certificates forward when asked for them; if he chooses to put forward the best certificates each time, which a rational peer would do, then we have a global perspective of the ratings these certificates contain when on the case that the peer returns random certificates each time the perspective will certainly not be a global one.

Visibility and sharing of the variable:

• v_i.interaction : the scheme states that peers may not share an interaction trust value if they choose to so we can't declare with certainty the interaction trust values as public or private.

• v_i.wrep : the witness reputation values are publicly shared by the peers since the scheme uses them in the case that interaction trust values can't be collected and also highlights the assumption that it makes that peers are willing to share ratings that they gave and help others search for witnesses.

• v_i.rb : for the role-base trust values the paper doesn't explicitly state something about their visibility through the network but says that for peer *X* to determine the role-based trust values of peer *Y*, *X* will look up the relevant rules from its database. So, we can infer that the rules (and thus the role-based trust values) are private or shared between peers that are listed in the tuple of a rule.

• v_i.crep : for the certificate trust values it is stated that the information of the certificates is stored by the peer that has received the certificate and is made available to any other peer that wishes to evaluate his trustworthiness but then the paper states that the peer who is asked for the certificate can choose which ratings to put forward thus a rational agent will choose only the best ones, so we can't label certificate trust values as public or private, in addition the paper states that peers may refuse to give out their ratings in this case.

<u>The method for calculation/updating of the variables:</u>

<u>Rating:</u>

• The peers involved in a transaction will provide ratings for interaction trust values.

• All peers that interacted with the peer in question rate on witness interaction, since the paper states in order to evaluate witness reputation of *b* peer *a* needs to find the witnesses that have interacted with *b*.

• The role-based trust is domain-specific thus updated when the peer's domain changes and not with ratings.

• Certificate reputation is not something a peer can rate for after the certificates are created. The peer that received the service will rate on the behavior of the peer that provided it and that will be saved in the certificate which the providing peer will store.

<u>Ratings aggregation:</u>

• The trust value is calculated as the weighted mean of all ratings (ratings from other peers) available. Also, the scheme defines a reliability value in the range (0,1) which reflects the confidence of the trust model in producing each trust value given the data that it considered.

• Additional rules can naturally be added during a peer's life cycle for the role-based trust.

<u>The topology of the network in which the scheme/system works:</u>

Multi-agent systems so the model is open to any topology.

<u>Strengths and Weaknesses of the scheme/system:</u>

<u>From the system's author view:</u> The paper states: "It has been shown that taking various sources of trust information into account not only helps FIRE be able to make trust evaluations in a wide variety of situations, but also increases its precision; and that all the components contribute a significant amount to its overall performance." and "FIRE is empirically evaluated and is shown to help agents gain better utility (by effectively selecting appropriate interaction

partners) than our benchmarks in a variety of agent populations. It is also shown that FIRE can effectively respond to changes that occur in an agent's environment."

From a critical point of view: FIRE just as other schemes we have seen before (e.g. R^2Trust) follows a sophisticated approach on how it represents trust considering various factors to achieve greater trust accuracy. As the author proposes this variety of factors that are considered will also deem FIRE to be flexible with different environments and respond well to changes in the system (e.g. detect a user that turn malicious in a short period of time). FIRE proposes a trust and reputation scheme without considering the topology where it will be implemented. So, if we were to implement a system that uses the FIRE scheme as its blueprint, we could run into issues that FIRE was not designed to deal with. For example, if we implement FIRE for a mobile P2P network were resources are limited, we might have issues because FIRE demands usage of several types of trust, each with its own individual special features that might not be possible to implement in such a network.

Formulation of CrowdBED trust and reputation management systems from this system/scheme:

The only parameters of v_i, that FIRE uses, and our scheme does not (so far), are certified reputation and role-based reputation. Certified reputation is obsolete with our system; that is because we already have a transparent ledger so using certificates to prove the truth would serve us nothing. When it comes to role-based trust, since we don't force users that join the system to pick a role that they will keep for the rest of their lifetime in the system, it would make sense for us to use it. Because for example, a user can be a good worker but a bad master (e.g. asking for a task to be completed in a timeframe that it is not possible to be completed in, advertise a low task difficulty while the actual task is being difficult etc.). So, using role-based trust would be a good addition to our system so that we have better feedback. Of course, we will not use it to the depth that FIRE uses it since in our system only three roles can possibly exist (master, worker and auditor – and auditors will exist only if we choose to use auditing instead of multiple workers). In the case we don't want to use role-based trust to make the system more lightweight then enforcing users to pick a permanent role for the rest of their lifetime in the system as soon as they join it is the best next option.

Peers in FIRE can choose to hide interaction trust values something that we can't do in our case since everything regarding a transaction and related to trust is recorded in the ledger which is public.

4.6 Decentralized Trust Management in P2P Systems

Paper [6] proposes an anonymous and secure protocol for maintaining, accessing and updating trust information in decentralized peer-to-peer systems.

The variable v_i:

Structure and Scale of the variable:
Each peer has a trust rating v_i.trust representing the quality of service it provides.

Perception of the variable:
The trust rating is global since the set of peers that has each value will return the same value since the value is encrypted and can't be altered, so every peer that asks for the value will receive the same answer.

Visibility and sharing of the variable:
It's stated that the trust rating of each peer is stored by a randomly chosen set of peers who reply to the queries for all the trust values that they store.

The method for calculation/updating of the variables:

Rating:
- The ratings that peers give for other peers are secret and they remain in the system even after a peer logs out so the impact of the ratings in the updating of the trust values is forever.
- Since v_i.trust is an aggregation of all ratings obtained from the rest of the peers we infer that all peers that have a rating for that peer will provide it. The peer that received the service will inform about its feedback the trust manager of the peer that provided the service.

<u>Ratings aggregation:</u>

• The paper states that v_i.trust is an aggregation of all ratings obtained from the rest of the peers.

• The protocol considers that each peer X will be associated with a trust manager peer Y responsible to compute, update and store the trust value of peer X. So, aggregation is done by one peer for each peer. As it is stated after a new peer joins the network, he will have an associated parent (the direct parent in the tree overlay architecture) who will have the attribution to associate the peer with a set of trust managers and announce them.

<u>The topology of the network in which the scheme/system works:</u>

Decentralized P2P architecture with focus on SOPSys architecture; Self-organizing decentralized peer-to-peer system based on well balanced multi-way trees.

<u>Strengths and Weaknesses of the scheme/system:</u>

<u>From the system's author view:</u> Persistence, small decision time and ease of contribution.
<u>From a critical point of view:</u> This system does not consider complex aspects of trust as other systems that we have seen before did (e.g. FIRE, R^2Trust). We can suppose that this is done because of two possible reasons; the first one being the fact that this system uses encryption to ensure security and thus there is confidence in just a simple value and does not need the feedback that extra trust parameters would provide and the second reason might be that in order to support the cost that encryption has on resources the system prefers to use a lightweight representation of trust. Overall, this system aims to be very fast on calculating trust (as the author said small decision times) and capitalizes on the use of encryption and the topology where it is implemented in combination with using random trust managers to provide a high degree of security. We can note that this system would not do as well as other systems would in the case of high percentage of malicious within the network because random trust manager groups are responsible for managing trust.

<u>Formulation of CrowdBED trust and reputation management systems from this system/scheme:</u>

Here the system uses just a simple value to represent the quality of service of a user and uses encryption to ensure integrity. We already achieve this using a ledger in our system.

Here the system hides ratings that peers give for other peers. We will not do so because this would defeat the purpose of using a ledger which is total transparency.

Here calculation of new trust values is carried out by trust managers (each peer here has a random trust manager - another peer). In our case since everything is transparent we use a standard predefined way to calculate new trust values, that will consider all the various aspects that we added in the scheme before (other users results, updating of credibility values, "size of task" etc.), and thus we don't need to worry about trust managers since there is a consensus and if one user miscalculates new trust values on purpose this will immediately be detected and he will receive the respective punishment or the other peers will adjust their stance against him accordingly (e.g. will not work with him).

4.7 FR-Trust

FR-Trust [7] is a fuzzy reputation based model for trust management in semantic P2P grids.

<u>The variable v_i:</u>

<u>Structure and Scale of the variable:</u>

The paper states that each peer has a trust level that can be one of the three values; Low, Medium or High. So, a fuzzy logic is used here and so trust values are actually real number in the range (0,1) so we declare a single attribute the trust value of each node v_i.trust.

<u>Perception of the variable:</u>

Nodes are clustered into groups (called virtual organizations – VOs) thus we suppose a local perspective to the group for the trust values exist. The reputation scores are global since the paper states that reputation data are stored and collected by using super clusters to calculate global reputation scores.

Visibility and sharing of the variable:

We can infer that reputation and trust data are visible to the coordinator nodes of the clusters since the calculation of trust values the aggregation of the reputation scores from the agents and the queries are all done from the coordinators of the clusters.

The method for calculation/updating of the variables:

Rating:

• Peers that receive the service are the ones to judge the target node and thus rate for its reputation.

• Each VO has a coordinator node who is responsible for communicating with the other VOs. After a transaction between two nodes occurs, the source node of the link will determine a score according to its evaluation of the service of the destination node of the link and sent it to the coordinator of the group.

Ratings aggregation:

• The model calculates reputation scores by aggregating feedback from nodes to determine a node's trustworthiness.

• There is a trust agent in each super cluster which is responsible for storing reputation queries form nodes in other VOs and calculating the trust score of each node within the VO.

• Since the source node sends the score to the coordinator of the group, we can infer that the coordinators of the groups will aggregate the trust values. We can also back this up with what the paper states in the example that it gives; the trust agent is located on the coordinator node of the group and thus the aggregation of the reputation scores (ratings) happens there.

• The trust levels are also calculated by the trust agents in the coordinator nodes.

The topology of the network in which the scheme/system works:

Semantic P2P grids and P2P grids.

Strengths and Weaknesses of the scheme/system:

From the system's author view: From the paper: "our experimental results demonstrate that FR-TRUST combines low (and therefore desirable) and good computational complexity with high ranking accuracy".

From a critical point of view: FR-Trust uses fuzzy logic and so v_i has a parameter that scales to the words "Low", "Medium" or "High". The reason why this is done seems to be that the main target of this scheme is to achieve security in the P2P topology that is designed to work for. Thus, prefers that information that is exchanged between peers is of vague nature rather than absolute so that greater accuracy is achieved in detecting malicious peers. Example: Some peer A is 40% sure that B is malicious. If we used binary values and A had to choose between 1 or 0 whether he believes that B is malicious or not, respectively, then we would get 0 for an answer, but since fuzzy logic is used that percentage of belief that A has will not be lost and thus the system will get better feedback for B. Also, since aggregation is done only by the coordinators of clusters and thus the non-coordinator peers just need to give ratings when they receive a service this scheme manages to achieve smaller processing overheads. Overall, this scheme manages to provide a good level of security and speed. The only issue here would be that coordinators need to be trusted because if a significant malicious coordinator population somehow manages to work as a team, they can deem this scheme as a malfunctioning one. This scheme also denies transparency to the general peer population since only coordinators can see reputation values.

Formulation of CrowdBED trust and reputation management systems from this system/scheme:

This scheme clusters nodes into groups (i.e. Virtual Organizations) each with a coordinator node that is responsible for storing and calculating trust in cooperation with coordinators of other clusters. This idea would not work for us because we want total transparency since we are using a public ledger.

Fuzzy logic used here guarantees a better feedback and thus more accuracy. Accuracy of how trust is calculated leads to more security so it would be good that we use real values to represent trust in our system as well, or even integer values that map to a scale greater than one with just two values for ‘True’ and ‘False’.

4.8 Efficient Distributed Reputation Scheme for P2P Systems – RCert Trust

RCert trust [8] is a simple and efficient P2P reputation scheme that aims at solving the two problems of the storage and integrity of reputation.

<u>The variable v_i:</u>

<u>Structure and Scale of the variable:</u>

Each peer has a reputation value so we declare v_i.reputation that can be extracted from the certificate of each peer that contains the history of transactions of the peer.

<u>Perception of the variable:</u>

Here we have a global perception since every peer will see the same certificate for some peer and thus extract the same reputation value from it as the other peers will.

<u>Visibility and sharing of the variable:</u>

The reputation is stored by its owner in a certificate to keep its integrity but it is said after that the confidentiality part has been omitted from the targets of the cryptography used here so we can suppose that reputation values are global since they can be exchanged on demand from a peer searching for a serving peer.

<u>The method for calculation/updating of the variables:</u>

<u>Rating:</u>

The peer that is being serviced (rater) will generate a record of the transaction that will contain a rating and that record will be appended to the end of the certificate of the peer that provides the service (ratee) and will have the signature of the peer in order to show that it is legitimate. In the case that the rater does not provide a rating or does not provide a valid signature then the ratee can present the transaction acknowledgment to his previous rater and thus the current rater would have to either cooperate or be punished by cancelation of the revocation on his last time stamp.

Ratings aggregation:

The ratings are all stored in the certificate and can be processed in any way the peer that reads them wants.

The topology of the network in which the scheme/system works:

P2P networks.

Strengths and Weaknesses of the scheme/system:

From the system's author view: Security and transparency are the strengths of this scheme.
From a critical point of view: Indeed, this scheme addresses the transparency and security issues well but one important thing that this scheme does not explain (at least at the 2003 version that was studied) is how would the system prevent a rater from being malicious in the sense that he gives a malicious rating (e.g. negative rating for good service).

Formulation of CrowdBED trust and reputation management systems from this system/scheme:

The main reason that certificates are used here is clearly transparency which is what we achieve as good when we are using a ledger. So, in our case we do not need certificates to be transparent. We can even argue that a ledger is even more transparent since consensus from the whole community is needed for every bit of change whereas with certificates just the user that gives the rating needs to put his signature.

One could argue that a system that uses certificates would demand less memory from the overall community than a system that uses a ledger like ours. Which is true since it would be cheaper for each user to keep his own certificate that includes all transactions related to itself rather than a ledger that contains information for the whole community. But since less users are involved in updating a certificate than a ledger this would mean less transparency and also less security; What would stop a team of malicious users from elevating each other by choosing that they are serviced only by members of their team because they could make the transactions look legit but the only reason these transactions would be made would be to fill up the certificates of the malicious team members with positive transactions.

4.9 VANETs Reputation Scheme

VANETs reputation scheme [9] is a distributed reputation scheme for situation awareness in Vehicular Ad Hoc Networks (VANETs). Peers here are also referred to as vehicles.

The variable v_i:

Structure and Scale of the variable:

Each peer here has a reputation score which is a real number in the range (0,1), note the RSUs are considered to be trustworthy so we consider their opinions to be legit thus they have a constant value of 1 as their reputation score.

Perception of the variable:

Storage is done at node level. Each node has its own reputation list thus reputation has a local to the peer perspective.

Visibility and sharing of the variable:

The reputation scores are public since in the scheme the reputation values are updated based on opinions. So, if a vehicle is in the range of another and sends and opinion then a reputation score for it will be created in the reputation score list of the requesting peer.

The method for calculation/updating of the variables:

Rating:

Ratings are collected from all peers that are in the range of communication of a peer in the network and if the peers that are in this range are not enough then the RSU unit is used which is a device to provide connectivity between peers. We consider the RSU to be occasional peers since what the RSU is doing is collecting opinions from peers that had been inside its range sometime in the near past and forms its own opinion.

Ratings aggregation:

The receiving vehicle aggregates the opinion it has received from members of the crowd. The receiving vehicle further sets a threshold. If the average *t* score of all members of the crowd is above the threshold, the message from the sending vehicle is accepted as reliable.

The topology of the network in which the scheme/system works:

VANETs.

Strengths and Weaknesses of the scheme/system:

From the system's author view: As stated in the paper "we have proposed an integrated reputation mechanism that establishes trust from vehicles and from the Roadside Unit. It is our thesis that a combination of vehicle to vehicle, and vehicle to Roadside Unit reputation establishment enhances the ability of vehicles to identify malevolent peers." and "Our simulation results demonstrate that on average, our scheme has a 90 % accuracy in detecting malicious messages coming from malicious vehicles. In addition, the model is highly scalable."
From a critical point of view: The strength of this scheme is the proposal of the RSUs that will help users that have not enough other users in their communication range to aggregate opinions properly. Because what the authors of the scheme achieve with this is reliable feedback provision to members of such networks as the VANETs. With good feedback comes greater accuracy and better avoidance of malicious information. There is no significant weakness of this scheme, since most cases one can think of, that this scheme would not work well with, are not related to how the scheme is designed, but rather to the topology it is designed for (e.g. a vehicle has neither vehicles that tell the truth nor an RSU in its communication range but only has vehicles that lie and has no previous information for).

Formulation of CrowdBED trust and reputation management systems from this system/scheme:

This scheme would work very good with VANETs. But, since this is not the kind of a network that we are planning to work on, not a lot of this scheme’s ideas can be adopted. We can’t have something like the RSUs in our case since we are using a ledger all users share the same information, so we don’t need static bot users to collect data over time and pass it on to the next user that comes in contact with them. The only idea that we will use from here (and that we found in previous schemes as well) is using values that are not binary (0 or 1) to represent trust.

Chapter 5

Study of Surveys and Mechanisms Proposing Papers on Reputation Schemes

In this chapter two surveys [10] and [20] that analyze a collection of proposed and live reputation systems are studied. These surveys also highlight various important aspects that have to be taken in mind when designing a reputation scheme (e.g., the various attacks, the kinds of trust, the kinds of trust propagation etc.). For each survey, first of all its content is summarized to trim down details and list only the points that must remain in focus when our approaches will be designed and secondly an effort is made to map all the schemes and systems that are mentioned in the survey to the framework suggested in Section 2.2 and then formulate the mechanisms that will finally be proposed for CrowdBED. The mappings are done so that the bibliographic study becomes even more diverse and extensive.

Afterwards the same process is done for two papers [42] and [49] that are more closely related to our work than the previously studied schemes and systems.

Finally, we present three tables. The two first categorize all the studied schemes (both from Chapter 4 and 5) by application and by the network topology in which they work. The third table presents the attribute analysis of all schemes.

5.1 Study of Survey Paper: "A review on trust propagation and opinion dynamics in social networks and group decision making frameworks"

This survey [10] doesn't completely analyze any schemes but rather focuses on listing various techniques, elements and aspects and highlighting issues and threats found in trust and reputation management systems in general so what we will do is firstly summarize these ideas that are studied in the paper and make sense to the framework that has been set and then study all of the schemes/networks related to reputation and trust management and are mentioned in the paper (even in the introduction section because a lot of them are only mentioned there). While doing so, we will also in parallel keep formulating the two approaches for a trust and reputation management system for CrowdBED.

5.1.1 Summary of the Survey

Firstly, the survey specifies the categories of the trust and reputation management systems when it comes to their architecture and the way they calculate reputation.

- Reputation network architectures: The two main types of reputation network architectures are discussed in the paper; the centralized and the distributed.
- Reputation calculation mechanisms: Three mechanisms are mentioned and discussed; counting (positive minus negative votes), probabilistic models (using past binary ratings and probability distributions to guess future transactions outcomes) and fuzzy models (using fuzzy numbers or linguistic ratings as fuzzy sets with membership functions).

Next the survey lists the trust propagation approaches. Here it specifies that the models studied are known as Flow Reputation Systems which means that to estimate trust values between two agents with no previous interaction between them they rely on some kind of transitive property of trust via an iterative transitivity based aggregation along the different paths in the network that connect indirectly both agents until the estimated trust scores become stable for all agents. We list these trust propagation approaches here since they don’t give all the details in order to be mapped as schemes to the framework we created.

First is Guha’s et al. approach [11] where four different ways of carrying out atomic propagation of the trust are listed:

• Direct propagation: If A trusts B and B trusts C then A trusts C.

• Transpose trust: If A trusts B then B might present some level of trust towards A.

• Co-citation: If A trusts C and D, and B trusts D then we assume that B may trust C.

• Trust coupling: If A trusts B then this trust propagates to C if B and C trust agents in common.

Then these four ways are combined into one matrix that is based on a belief matrix and a weight vector. Then three schemes are proposed for propagating both trust and distrust that vary in the way the belief matrix and the matrix of propagation after some steps are generated by considering or not the distrust values.

• Trust only: The distrust values are completely ignored so trust scores are propagated.

• One-step distrust: Here the additive distrust propagation is assumed which means that if a user doesn’t trust someone all the judgements of this person are discarded and so distrust propagates only one step whereas trust propagates recursively.

• Propagate distrust: In this case it is assumed that both trust and distrust are propagated together, and so they are processed as the two ends of a continuum.

Then it’s Kamvar’s et al. approach [4] where there is no distrust propagation. For each node this approach calculates a global measure of trust. Each time agent A interacts with agent B the transaction maybe rated as positive or negative. Then a local trust value $v_{trustAB}$ is defined as the sum of all the ratings that A gave for his interactions with B. Instead of positive and negative interactions can also be voted as satisfactory or unsatisfactory. In this approach it is demonstrated that transitivity of trust leads to a system where global trust values correspond to the left principal eigenvector of a matrix of normalized local trust values. The scores values are normalized when aggregating local trust values. Following the normalization, the aggregation is carried out by asking the peer A’s neighbors about opinions on other peers which are weighted by the trust A places on each of them.

Next the paper lists trust and reputation distributed approaches.

First is the RateWeb [12] which is a decentralized and unstructured approach that calculates trust between services. Each time an entity needs a service it queries all the other entities, and for each service a list of the entities that used it is returned. Here each agent stores a personal vote for all the services it interacted with but also reputation based on the feedback the others gave. (For this approach the paper cited referring to it needed to be purchased to be read)

Then it is the R^2Trust [3] (which was one of the schemes that were studied individually before), which is a fully distributed system. Again, here both trust (direct interactions) as well as reputation (obtained recommendations) are used. The credibility mechanism for filtering out bad recommendations and the social relationships (the relationships we know e.g. the fact that two entities may belong to the same organization) are used here. The success of this approach is that irregularities in behavior are quickly detected and thus is maliciousness. Also, this approach uses time periods so past votes fade out as the time goes on.

Then Graft [13] is presented that uses both implicit (in-degree and out-degree in social networks) and explicit (scores and ratings) reputation information. This system does not process the information but instead it uses logical policies to generate trust based on the profiles of users.

Then another approach [14] is mentioned which is a decentralized graph-based reputation model which uses social context to fully build the trust relationships. This system develops a multi-graph based social network in which the users are characterized and interconnected by means of the context of each performed activity, the values associated with the activity and the relationship, all the performed activities and all the social relationships of the users.

Next the paper talks about trust based social network decision making approaches and gives the general composition of GDM (group decision making) processes; Agents preference elicitation (experts provide opinions), opinion aggregation (fusion of opinions from different agents), consensus calculation (level of similarity between opinions is calculated so we can accept or not that a group agreement has been reached) and feedback (feedback is returned to the agents so that opinions get closer and thus a consensus is reached).

In the end the survey talks about opinion dynamics and presents two main mathematical models.

The DeGroot [15] model which considers the opinion evolution of the individuals as a weighted average of their opinions at a previous time state and the Friedkin and Johnsen [16] model which generalizes the DeGroot model and considers as well how individual opinions evolve with respect to their previous state opinions.

5.1.2 Formulation of CrowdBED Trust and Reputation Management Systems from the Summary of this Survey

Reputation calculation mechanisms: We will combine ideas from both the counting technique and fuzzy models. That is, we will store information for how many times a user has been rated, so that this information is not lost because of its importance (the more ratings received the more chance that the trust value is accurate), and we will also use real number values to represent trust. We could also use a member function and thus use fuzzy numbers for trust, but this can be decided in the future and if we decide that such change would mean significant increase in the accuracy of our system.

Propagation approaches: Since we are using a ledger and consensus is needed for every change in the ledger then we do not need to worry about how trust will be propagated since the propagation of the ledger would also mean the propagation on trust. So, we leave this issue to the protocol that is used for the propagation of the ledger.

Distributed approaches of the paper:

RateWeb [12]: Nothing mentioned assists us to formulate our approach. For this scheme it is even mentioned that for an entity to find a suitable service it needs to query all the other entities while in our case it would just use the ledger and find out the answer faster without the communication overheads.

R^{2}Trust [3]: We already studied this scheme. Again, the time periods are mentioned here and that this helps the scheme to detect irregularities faster, but we already said that we can use check-ups every K interactions and achieve the same goal. We will also do so because storing every past interaction's rating and fading its effect out with time (as R^{2}Trust does) could make the ledger need more memory than what we aim for.

Graft [13]: It would not make sense in our case to use implicit reputation information like in-degree and out-degree of a user (in our case the count of users that served him and the count of users that he served) and that is because we are not in the same situation as a social network. If a user here decides to use certain users to serve him, because of his own preference, we should not consider this as something that would suggest maliciousness since it is his right to do so. We already addressed the issue of groups working together to elevate each other maliciously and that is the auditing of the tasks, if the auditors (who will not be picked by the users involved in the transaction but will be picked randomly) decide that maliciousness exists the system will be notified.

Then the approach [14] that uses social context is mentioned. For this we already said that we choose to keep a separate trust value for each user for his master trust and his worker trust and discussed the issue further before.

Group decision making processes: Here we can see that our system as it is designed until now follows all the parts of the GDM process; we use task auditors as expert opinion, we aggregate the opinions of all agents involved in the task, we use ledger for consensus and users can use the ledger for the feedback they need.

DeGroot [15] and Friedkin-Johnsen [16] models: It is true that the idea of watching and considering how one's opinions evolve through time would improve the accuracy of a system but that means that several previous trust and reputation values for each user must be stored in the ledger for us to decide how the opinions evolve and perform calculations based on this evolution to improve the system's accuracy, which we most probably would not afford considering that we need the ledger to be light so that all users have it.

Following are the mapping of the systems and schemes mentioned in the survey separated by their applications.

5.1.3 Social Networks

For the social networks (facebook.com and twitter.com) research was made for technical papers about reputation mechanisms that might exist to govern the network but since these mechanisms are proprietary, we just mention them here.

5.1.4 Online Market-Places

All the marketplaces that were mentioned and are studied here are using trust and reputation management systems that work on centralized architectures and are rather simple compared to systems used in other applications.

ebay.com

To map the trust mechanism of eBay to the framework wanted, in the best possible way while staying relevant to the survey, reference was made to the details provided by the survey [10] itself in the part where the counting technique is discussed for reputation calculation, and the cited papers [17,18].

The variable v_i:

Structure and Scale of the variable:

The main parameter:

• v_i.ratings : which holds for each seller (servicing peer in our case) his ratings and is a simple integer.

We also consider two sub-parameters:

• v_i.positive_ratings and v_i.negative_ratings : simple integers initial value 0.

and their usage is shown below. From the empirical analysis a seller's feedback profile will include a breakdown of positives, neutrals and negatives (ratings) also a breakdown of the most recent week, month and year is included, so from that we can decide to include v_i.neutral_ratings, v_i.last_week_bd, v_i.last_month_bd and v_i.last_year_bd as parameters as well.

Perception of the variable:

- v_i.ratings : global perspective for all sellers.

Visibility and sharing of the variable:

From the empirical analysis we know that users have the option to choose if their feedback ratings are displayed or not but what they choose will be enforced on their whole feedback profile so they can't choose to share certain cases and others not. By default, the feedback profile is publicly visible.

The method for calculation/updating of the variables:

Rating:

As stated in the empirical analysis paper buyers and sellers (users) after each transaction can leave a rating but are not required to do so. Also, for every positive or negative rating a certain transaction must be associated with it.

Ratings aggregation:

As mentioned in the survey eBay is using the Counting technique when calculating reputation which consists in the summation of positive ratings minus the summation of the negative ones. Also, in the empirical analysis is stated that the feedback summary is computed by taking the number of unique users who left positive feedback and subtracting the number of unique users who left negative feedback.

The topology of the network in which the scheme/system works:

A central authority (eBay) exists and so this system works on a centralized architecture.

<u>Strengths and Weaknesses of the scheme/system:</u>

<u>From a critical point of view:</u> eBay is using the counting technique to calculate reputation and it allows users to query for recent time periods regarding one's ratings. Both these properties of eBay increase the feedback users see and thus allows for greater accuracy. eBay's trust and reputation management system is a simple and straight-forward one (i.e. no complex techniques are used to calculate and propagate trust) and the main reason for this is that there is a central authority that governs the system. So, what a member of this system is expecting to happen is that all the ratings are to be trusted and thus reflect the truth because malicious members of this system are detected and punished by the central authority. This trust to the central authority to be just and pure is crucial for one to be able to admit that this system works well and fights effectively malicious users and eliminates corruption in general.

<u>Formulation of CrowdBED trust and reputation management systems from this system/scheme:</u>

We already studied and found interesting, from previous research that we did and from this survey as well, the ideas of both the counting technique and the use of time windows for trust, which are used in the eBay's reputation system as well.

amazon.com

<u>The variable v_i:</u>

<u>Structure and Scale of the variable:</u>
From what was stated in the survey about how Amazon [19, 20] is calculating ratings 4 parameters are to be considered;

- v_i.overall
- v_i.rater_trustworthiness
- v_i.distance_between_ratings

• v_i.current_score

<u>Perception of the variable:</u>

We are not explicitly informed about the perception of each parameter of the variable in the survey, but for some of them it can be inferred:

• The v_i.overall and v_i.current_score of a seller will be displayed the same to all buyers.

• The v_i.distance_between_ratings between a seller and two different members of the network will not always be the same since members don't always have the same rating obviously, so this parameter has a local to the peer (member of the network) perception.

<u>Visibility and sharing of the variable:</u>

• We can infer that v_i.overall and v_i.current_score of a seller will be public knowledge.

• Also, the v_i.distance_between_ratings parameter is public to all members of the network since one can find his distance between a seller's rating and his own.

<u>The method for calculation/updating of the variables:</u>

<u>Rating:</u>

Not mentioned but we can infer that sellers receive ratings from their customers with every transaction.

<u>Ratings aggregation:</u>

As mentioned in the survey Amazon is using an approach that is related to the counting technique when calculating reputation which is based on a weighted average of the ratings while considering factors such as rater trustworthiness, distance between ratings and current scores.

<u>The topology of the network in which the scheme/system works:</u>

Centralized architecture with a central authority.

<u>Strengths and Weaknesses of the scheme/system:</u>

<u>From a critical point of view:</u> Amazon uses something that we have not seen before and that is the distance between ratings parameter of the variable. So, it considers this when calculating reputation which provides a better inside between the relation between rater and ratee, because the gravity given to a rating, received from a rater that has a very different overall rating from the ratee, should not be the same as to a rating received from a rater with similar overall rating with the ratee. The counting technique is used which makes the reputation calculation simple and no complex parameters are used when representing trust and reputation values. We can attribute this simplicity to the fact that a central authority governs the system so there is no need for complexity in calculating and propagating trust and reputation. Just as with eBay's system trusting the central authority to be pure is the crucial requirement before one is able to admit that this system will result to decreasing corruption.

<u>Formulation of CrowdBED trust and reputation management systems from this system/scheme:</u>

We can see that centralized systems tend to use simple ways of calculating and propagating reputation. This might be a hint that in our centralized approaches we should do the same since the central authority is trusted by the users and thus the use of simplicity would be better since that would lead to smaller overheads. But on the other hand, we can see that this simplicity entails greater centralization, so we have to choose between greater decentralization and less (or less complex) computation.

5.1.5 Crowd-sourcing Platforms

wikipedia.com

There aren't any mentions of Wikipedia other than the one in the introduction and no paper relative to Wikipedia was cited in the survey, so a search was made in order to find any

technical papers or empirical analysis online. A paper was [21] but an IEEE account was needed to view it, so a decision was made to study the reputation system empirically.
From surfing the site what was realized was that administration of content exists only for certain projects of Wikipedia where an administrative body exists to check the content that users provide for that specific area that the project studies (e.g. wikispecies focused on listing the species of animals). Also, a try was made to edit content on a random article of the site and what warning one gets when trying to edit is the one in the figure 5.1 below:

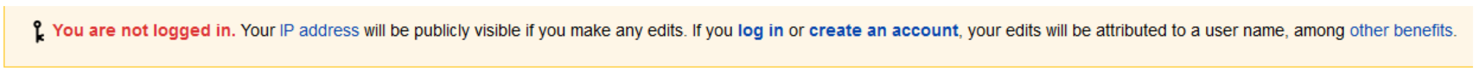
You are not logged in. Your IP address will be publicly visible if you make any edits. If you log in or create an account, your edits will be attributed to a user name, among other benefits.

Figure 5.1: The editing warning of Wikipedia

So, since we know that one can use a VPN and keep changing IP addresses then no matter how good of a reputation system Wikipedia has one can keep changing IP addresses and thus changing his identity in the network and keep being malicious without anyone being able to identify him and restore the changes he made and this renders the reputation system (any reputation system) useless. If account creation was mandatory, then it would make sense for us to continue studying the Wikipedia reputation system, but since it is not, we stop here.

quora.com

There aren't any mentions of quora.com other than the one in the introduction and no paper relative to quora.com was cited in the survey [10], so a search was made in order to find any technical papers or empirical analysis online. The article [22] was found and was used to map the reputation and trust management system of quora.com to our framework.

<u>The variable v_j:</u>

<u>Structure and Scale of the variable:</u>
It is not explicitly mentioned in the scheme what parameters does the variable have but we can infer them. We know that each answer to some question has its own rating which consists of upvotes and downvotes. Along with quality of answerer's past contributions the rating is

used to rank that answer and by that ranking the answer is presented to the user who searched or made the question, so for a user A and answer Y we declare:

• v_A.answer_Y_rating : which consists of the upvotes and the downvotes as mentioned before and v_Y.ranking.

Each user of the site has a profile but from all the information that this profile has the only part that makes sense to a reputation scheme is the bio, recent activity and the social graph, so we declare:

• v_i.recent_activity and v_i.social_graph,

we are not going to declare a parameter for the bio because of its ambiguous nature.

Also, the paper mentions that people that create a profile and provide answers must provide their real identities again we will not declare a parameter for that part of information. Additionally, it is mentioned that the system for determining a user's authority on a given question is called the PeopleRank of a user, so for a user A and a question X we declare:

• v_X.user_A_peoplerank

<u>Perception of the variable:</u>

• v_A.answer_Y_rating : the ratings of each answer are displayed the same to all users (i.e. all users see the same ranking and votes for the same answers at a given time).

• v_X.user_A_peoplerank : Each question will assign different people rankings to users than some other question (e.g. an engineering question will assign a better people ranking to some engineer than a question on mathematics will do) so v_X.user_A_peoplerank will have a local to the question perception.

<u>Visibility and sharing of the variable:</u>

• v_A.answer_Y_rating and v_Y.ranking is public since every user can see the ratings an answer has received and its ranking.

• The v_X.user_A_peoplerank is not mentioned anywhere to be public and it doesn't have to be since it is used from the site to automatically and in the background calculate the authority of a user to answering a question so we infer that it is hidden for all the other entities in the network.

<u>The method for calculation/updating of the variables:</u>

<u>Rating:</u>

Social rating of answers: users that read an answer can choose to rate it or not. So, everybody can rate (except the user that gave the answer of course) and doing so is not mandatory.

<u>Ratings aggregation:</u>

- In the paper it is mentioned "Votes from those who have been "detected to have been gaming the system" are ignored."
- v_Y.ranking is calculated using a combination of metrics (rating, answerer's past contributions quality etc.)
- It is mentioned that the system to extract the value for the parameter v_X.user_A_peoplerank is designed to provide a more sophisticated mechanism for ranking answers than merely vote count.

<u>The topology of the network in which the scheme/system works:</u>

All the data is stored in the servers of quora.com so there is a central authority that has access to modifying everything, so this system is implemented on a centralized architecture.

<u>Strengths and Weaknesses of the scheme/system:</u>

<u>From the author of studied paper point of view:</u> In the paper is mentioned "participants mentioned that once an answer received a lot of up-votes and was ranked highly, it was hard for other answers to get visibility, even if they added new or valuable information." and "There is a concern among Quora users that clique-voting is prevalent on the site". The authors of the paper asked users to provide their beliefs which are not necessarily true.
<u>From a critical point of view:</u> Quora's system provides the users with rich feedback to make their decisions (e.g. whether to trust some answer or not). Also, this rich feedback is used by the system itself to rank answers, example is how PeopleRank is used to determine how well can a user answer a question based on his past activities (what kinds of questions he answered or posed). This reputation system is a simple one as the centralized systems we have seen

before were and we can guess again that the reason for this is the existence of a central authority. For us to be able to decide that this system will not only rank answers correctly but also will detect malicious users (or users “detected to have been gaming the system” as the paper says) we need more details, about how the central authority governs the community, that we can’t find in this paper and maybe Quora is not releasing to the general public which is the most possible case.

Formulation of CrowdBED trust and reputation management systems from this system/scheme:

The idea that Quora’s system uses to rank users on their expertise on answering certain categories of questions, although a smart one, can’t be adapted to crowdsourcing platforms that require users to just provide their computational power. That is unless, there is a way to classify what kind of computational power a user provides (e.g. GPUs work better for certain types of calculations than CPUs, or we can detect that a user does better when providing parallel processing power rather than sequential processing power which of course depends on the nature of the task etc.). For us to decide if this classification is possible, we have to question what the tasks will ask from the users (e.g. if we just give the problem to the user to solve it we won’t know if he used a parallel approach or not so we don’t know what kind of processing power he provided). If we conclude that in our case this is possible (classifying users’ abilities) then this idea can be adapted in our approach, but it seems that this classification of computational resources and also keeping track what kind the worker uses is not ideas that can be developed easily and require rather a lot of different study of the field of computation.

5.1.6 Online Facilities Sharing Networks

uber.com

There aren’t any mentions of uber.com other than the one in the introduction and no paper relative to uber.com was cited in the survey [10], so a research was made in order to find any technical papers or empirical analysis online. No such papers were found so a visit to the service's website was done. From visiting the service's official website (www.uber.com) an article on how star ratings work was found [23] from this article we tried to collect as much information to map the star rating system of Uber to our framework.

<u>The variable v_i:</u>

<u>Structure and Scale of the variable:</u>

Both riders and drivers have a rating, so we declare:

- v_i.driver_rating and v_i.rider_rating.

For a certain trip Y, we can declare:

- v_i.trip_Y_driver_rating and v_i.trip_Y_rider_rating.

<u>Perception of the variable:</u>

All the riders will see the same rating for a driver and vice versa. So, users of Uber have a global perspective of ratings v_i.driver_rating and v_i.rider_rating.

<u>Visibility and sharing of the variable:</u>

It is stated in the site that ratings are anonymous, and you won't see individual ratings tied to a trip or person, here we understand that this means that ones vote to another is secret. So, v_i.trip_Y_driver_rating and v_i.trip_Y_rider_rating have a private visibility to the one giving the rating. In the video provided with the article it is also stated that in the beginning a driver will not have his rating displayed to him up until several trips have been completed so he can't infer rating given to him from his first riders. As soon as a driver finally gets his rating (has completed several trips) then the rating is visible to riders who might be interest for a trip that can be completed by the driver, so v_i.driver_rating is visible to all the users of the app (of course these users must be searching for a trip that this driver can complete at the certain time of the search in order for him to be listed in the results and thus his rating displayed).

<u>The method for calculation/updating of the variables:</u>

<u>Rating:</u>

Drivers and riders give each other ratings based on their trip experience; After each trip, riders and drivers can rate each other from 1 to 5 stars based on their trip experience. So, everybody involved in a transaction can vote the other user.

<u>*Ratings aggregation:*</u>

• The driver rating equals the average of the last 500 ratings from drivers.

• If a rider gives a low rating that he attributes to the price, traffic or other issues that the driver could not help then the rating is removed from the driver's average (the v_i.driver_rating that is).

<u>The topology of the network in which the scheme/system works:</u>

A rider can interact with drivers near his location and that can make the trip he needs but here we can expect a centralized topology since everybody interacts with the servers of uber to retrieve information.

<u>Strengths and Weaknesses of the scheme/system:</u>

<u>From a critical point of view:</u> Uber uses a very simple trust and reputation management system. We can see that from the scale of the votes (1 to 5 stars) and how the rating is calculated (average of the last 500 ratings). Also, there is no need to design a complex way of propagating trust since all users communicate with the central authority to retrieve all the information they need. Again, we will attribute this simplicity to the existence of a central authority. Also, it is important to notice that a lot of information, related to a trip, is received by Uber in real-time from the driver's device. So, the central authority here does not depend just on the ratings riders provide when detecting malicious drivers and since that is true, we can also say that the central authority depends just partly on the trust and reputation system when it comes to fighting maliciousness. So, we can't go ahead and say Uber's trust and reputation management system is weak, when it comes to detecting malicious users, because is not its primary goal. Its primary goal seems to be to provide feedback to the community for other members so they can choose who to make the trip with, and that it achieves.

<u>Formulation of CrowdBED trust and reputation management systems from this system/scheme:</u>

Both users involved in a transaction (trip here) receive a rating from the other user and thus Uber allows both the driver and the rider (the worker and the master in our case) to have feedback for one another when they are about to decide for a trip (task assignment and solving in our case) to take place. We have already mentioned previously that we will choose to include separate ratings for each user for the roles of master and worker. We adapted this idea from schemes we have seen earlier but we highlight that it used by Uber as well.
Uber chooses to hide ratings associated with a certain trip. That is, neither the driver nor the rider will know what ratings they have receive from one another. We can say that this decision, that Uber did for its system, is associated mostly with the nature of the application that it is used for. In our case, we choose to be fully transparent since that is the goal we wanted to achieve, the moment we decided to use a ledger.
Also, Uber hides ratings for drivers that have not completed a certain number of trips. We will not adopt this idea since our users require this feedback from the beginning so they can make their choices and also because a transaction in our case must have a greater effect on the trust values than a trip does to the ratings in Uber because of the difference of the nature of our task.

blablacar.com

There aren't any mentions of blablacar.com other than the one in the introduction and no paper relative to blablacar.com was cited in the survey[10], so a search was made in order to find any technical papers or empirical analysis online. No such papers were found. So, we visited the service's official website where an explanation [24] of how trust works in this service was found which we used to map all the information regarding the trust and reputation management system to our framework.

<u>The variable v_i:</u>

<u>Structure and Scale of the variable:</u>

• v_Y.group_X_trust_level : the respondents have a trust level towards the crew they are carrying (for respondent Y and group X parameter v_Y.group_X_trust_level)
also, it is mentioned that votes are out of 5 (from 1 to 5).
• v_i.rating : An ambassador has a rating which has the same scale as trust level of groups.

Perception of the variable:
• v_i.rating : global perspective since every member of the network will see the same value.

Visibility and sharing of the variable:
• v_i.rating : is displayed in the profile page of the ambassador so it is public.

The method for calculation/updating of the variables:

Nothing was found regarding the method of updating the variable in the website, but we can guess that is approximately like uber.com.

The topology of the network in which the scheme/system works:

Not mentioned. We can guess a centralized architecture since one needs to contact the servers to retrieve any information.

Strengths and Weaknesses of the scheme/system:

From a critical point of view: The same as Uber's system. The only difference here is that the feedback is not as rich as Uber so we can say that this system is weaker when it comes to this part.

airbnb.com

There aren't any mentions of airbnb.com other than the one in the introduction and no paper relative to airbnb.com was cited in the survey [10], so a research was made in order to find any technical papers or empirical analysis online. Two papers were found [25, 26] and after studying these two papers every detail related to trust and reputation and makes sense to our framework was mapped to it below:

__**The variable v_i:**__

__Structure and Scale of the variable:__

- v_i.property_rating : each property has a rating out of five stars (lowest 1 and maximum 5).
- v_i.guest_rating : each guest has a rating.

Perception of the variable:

From the mention in the introduction of the survey paper it is stated the rating score v_i.property_rating has a global perspective. The v_i.guest_rating is hidden so there is no perspective a host can form (hosts can only read reviews but can't see average rating for guests).

__Visibility and sharing of the variable:__

From what was mentioned in the introduction of the survey paper it is stated the reputation score for properties (hosts) is publicly available to all users but votes for individual reviews (voting for a certain transaction) is disclosed, so the votes are hidden but the overall rating v_i.property_rating is visible. The guest rating v_i.guest_rating is hidden.

__**The method for calculation/updating of the variables:**__

__Rating:__

- Both host and guest can rate each other at the end of the stay (end of the transaction) up until 14 days after. After 14 days ratings is disabled.
- Ratings are displayed only when both parties of a transaction have rated or when 14 days pass after the end of the transaction.

__Ratings aggregation:__

After 3 reviews average rating is rounded to the nearest half start (e.g. 4.3->4.5)

The topology of the network in which the scheme/system works:

Centralized.

Strengths and Weaknesses of the scheme/system:

From a critical point of view: A simple system is used here; ratings are on a small scale (1 to 5), global perspective for hosts ratings, average of ratings to produce overall rating. We will again attribute this to the central authority and how users of the system have to trust it to be pure.

Formulation of CrowdBED trust and reputation management systems from this system/scheme:

Here we can't mention any property that we might adapt to our system and we have not already met it when studying Uber.

5.2 Study of Survey Paper: "A survey of Trust and Reputation Systems for Online Service Provision"

This article [20] gives an overview of existing and proposed systems that can be used to derive measures of trust and reputation for Internet transactions, analyze the current trends and developments in the area and to propose a research agenda for trust and reputation systems. We will try to map the mentioned and analyzed reputation schemes in this survey to our framework as well as use ideas from this survey to formulate the mechanisms for CrowdBED.

5.2.1 Summary of the Survey

We will first summarize approaches analyzed for certain parts of trust and reputation systems by the survey first (an effort to trim down details as much as possible was made without missing the main parts of information that we need to make the formulation).

First off, the article distinguishes trust between reliability trust and decision trust. The first is the subjective probability by which an individual, A, expects that another individual B performs a given action on which its welfare depends, the latter is the extent to which on party is willing to depend on something or somebody in a given situation with a feeling of relative security, even though negative consequences are possible. Then the definition for reputation is given as well; reputation is what is generally believed about a person's or thing's character or standing.

Then it lists the purposes of research in trust and reputation systems based on the observations it highlights regarding the differences of trust and reputation systems in the physical world and online environments; first finding adequate online substitutes for traditional cues to trust and reputation that we are used to in the physical world and identify new information elements for measures and secondly utilize the IT to collect this information so decision making is supported and the quality of online markets is improved.

The three points that are needed for a reputation system to even operate at all: Longevity of entities so with every interaction there is always an expectation for future interactions. Ratings about current interactions are captured and distributed. Ratings about past interactions must guide decisions about the current ones. Then the article proceeds to present these points in detail.

Afterwards the article talks about collaborative filtering systems (CF) and collaborative sanctioning. In short collaborative filtering systems are systems that collect ratings in a community (just like reputation systems) but also take in account the fact that different entities in the network might have different ways of rating things whereas collaborative sanctioning describes the reputation systems because they sanction low quality service peers.

Then the five trust classes are listed and explained in the article:

1) Provision trust; trust between a party and a service or resource provider. Used when the user is searching for a reliable (not malicious) service provider.

2) Access trust; trust for the purpose of accessing resources that the relying party is responsible or owns.

3) Delegation trust; trust in an agent that acts and decides for the relying party.

4) Identity trust; how much an entity believes another entity has the identity it claims to have.

5) Context trust; how much a relying party believes that a transaction will be supported well, and safety is guaranteed if something goes wrong.

These trust classes are attached to the application for which a reputation system is developed of course while we are studying a general framework but still knowing what classes of trust a system might need to deal with is important.

In the following section the article presents the categories of trust semantics. Understanding these is important so that participants understand trust and/or reputation scores. Two dimensions are used here to categorize the systems based on their trust semantics specificity-generality and subjectivity-objectivity. Here a table is presented with examples of systems and where they belong but apart from eBay no other system is mentioned that can be studied and thus mapped to the framework (e.g. survey questionnaires, D&B rating etc.) which are routines that companies usually run and not specific systems (eBay's reputation system was studied from the previous survey [10]).

In the next section the reputation network architectures are presented which we had seen in the previous survey as well; centralized and distributed. Fundamental for a centralized architecture are a centralized communication protocol and an engine that calculates reputation that the central authority uses. Fundamental for a distributed system are a distributed communication protocol and a reputation computation method that each agent can used to calculate reputation based on the input he received from the network.

Then the article talks about P2P networks. Examples P2P networks are mentioned, for which we will later try and see if we can find any reputation mechanisms that they might have used.

Then reference is made to the threats in P2P networks. Finally, the purpose of a reputation system in P2P networks is explained; First to determine who is trusted to offer the best quality resources and who is trusted to provide information for the later.

Then reference is made to reputation computation engines. Private and public information are separated; the first comes from firsthand interactions while the second is the one obtained from third parties. Then the various techniques used by reputation computation engines are listed.

1) Simple summation or average of ratings: What this describes is identical to the counting technique we read about in the last survey. Systems/schemes mentioned here are eBay, Epinions (www.epinions.com) and Amazon, the last two use a more advanced version of this technique.

2) Bayesian systems: they take binary ratings and use statistical updating of beta probability density functions.

3) Discrete trust models: Here trust values are not of continuous measure but are of verbal nature. Discrete trust models are then referenced which are cited by the survey and will be analyzed later, and these are [27, 28, 29, 30].

4) Belief Models: From [20] "Belief theory is a framework related to probability theory, but where the sum of probabilities over all possible outcomes not necessarily add up to 1, and the remaining probability is interpreted as uncertainty". Here two examples of work are noted which use the opinion metric which incorporates uncertainty, and these are [31, 32]. Also, the model [33] is mentioned which uses belief theory to represent reputation scores.

5) Fuzzy models: Here trust and reputation are represented as linguistically fuzzy concepts. Fuzzy logic provides rules for reasoning with fuzzy measures of a type like "trustworthy" and "not trustworthy". Again, the scheme [30] is mentioned here along with the REGRET reputation system [34, 35, 36].

6) Flow models: Here three models are mentioned Google's PageRank [37], the Appleseed algorithm [38] and Advogato's reputation scheme [39]. EigenTrust [4] is also mentioned to showcase that flow models don't need a constant sum of reputation/trust scores.

In the next section the article describes commercial and live reputation systems all of which have a centralized network architecture. Systems mentioned here are listed and mapped to our framework afterwards.

Then the survey talks about problems in practical and academic reputation systems and mentions proposed solutions. These are summarized in the list below:

1) Low incentive for providing rating:

• When participants want to keep a resource to themselves and thus will not share it.

• Avoiding negative ratings because of attitude or fear.

• Not benefiting from providing rating.

Solution: The survey cites two schemes that provide financial rewards and payments as incentives.

2) Bias towards positive rating:

• The survey mentions an empirical analysis [17] on the eBay's reputation scheme that found that very small percentages of ratings are negative.

• Possible explanation for this bias is that positive ratings simply represents an exchange of courtesies, or the hope that the favor will be returned, or again the fear of giving a negative rating.

Solution: Anonymous voting, using a cryptographic scheme.

3) Unfair Ratings: The survey presents two groups of approaches that try to counter this problem.

• Endogenous Discounting of Unfair Ratings: methods that exclude or give low weight to presumed unfair ratings based on analysis and comparing the rating values themselves. So, for these methods to work it is required that these unfair ratings can be detected using statistics.

• Exogenous Discounting of Unfair Ratings: using the externally determined reputation of the rater to weight the rating.

4) Change of identities: The idea of identity is fundamental for reputation systems to work. Parties that have received a lot of negative ratings or are in a worse position than what they would have been if they started fresh might find interest in changing identity.

Solution: An approach that penalized newcomers is mentioned but the flaw of it is that you can’t infer if a newcomer is good or bad beforehand.

5) Quality variations over time: variations in quality are possible and can be deliberate or uncontrolled.

Solution: Reducing the effect of a rating the older it gets or completely eliminating it after a certain time passes. The survey also mentions that reinforcement learning can be used here.

6) Discrimination: When a user has a certain stance against a certain other user (good or bad).

Solution: What was proposed in the exogenous discounting of unfair ratings, but it is pointed out that no simulations have proved that this will work properly.

7) Ballot Box Stuffing: providing more than the legitimate number of ratings and specifically unfair ratings (both positive and negative ratings).

Solution: eBay’s system addresses this by requiring that a transaction is associated with each rating. Also, systems that require a user to be a registered member before providing a rating provide some degree of protection against this problem.

The survey ends with discussion and conclusion. It lists the criteria for judging the quality and soundness of reputation engines which are: Accuracy of long-term performance, weighting towards current behavior, robustness against attacks and smoothness (a single rating should not have a significant effect by itself).

5.2.2 Formulation of CrowdBED Trust and Reputation Management Systems from the Summary of this Survey

Three points that are needed for a reputation system to even operate at all in our case:

1) Longevity of entities: Our scheme will offer to the system admin two choices. To have a permissioned system where users will need to provide ID to join or a permissionless one where users will not be required to provide some ID to join. In the first case longevity of entities is assured because of the ID they provide. In the second case we will make sure that newcomers will have some kind of punishment so that users with low trust values that decide to rejoin the system with a different user profile will not have a trust value that is much better than they did in the past and thus what characterized them in the past (their reputation) will not be changed much. In the second case we could also suggest another approach which is

having little tolerance to maliciousness and thus kicking users immediately after a certain strict threshold is breached and thus if they join with a new profile, they will be approximately in the same position that they used to be the moment they were kicked. Both solutions in the second case will be provided to the system admin.
2) Capturing and distribution of ratings: All ratings given will be stored in the ledger we will use and thus will also be distributed to all users.
3) Previous ratings influencing future decisions: Yes, we will allow a master to pick with which worker he will work with based on the reputation values he will find available in the ledger.

Collaborative filtering and collaborative sanctioning: Since we are using more than one auditor and not only rely on ratings from the master and the worker these characteristics are found in our system.

Five trust classes: We could say that we are using provision trust since the workers provide their services to the masters and the trust values are meant to guide the masters to choosing the worker. We are also using context trust since that is what an auditor's trust represent.

Reputation calculation techniques: We already decided to adopt the ideas of simple summation of ratings (counting technique) from the previous survey.

Problems in reputation systems:
1) Low incentive for providing ratings: The user that will provide a correct rating will be rewarded by having his trust value increased.
2) Bias towards positive rating: In our case false positive ratings will receive punishment. For example, if a master gives a positive rating for a worker and the auditors find that the task was not as difficult as the master advertised it to be or that the worker did not actually work then the master will receive a punishment for providing that false positive rating.
3) Unfair ratings: To tackle this we will use both the trust value of the user that provides the rating to weight it and also multiple auditors, so we are also statistically sure.
4) Change of identities: we already addressed this above when we talked about the permissioned and permissionless approaches.

5) Quality variations over time: We discussed that already; we will use time windows to detect sudden changes in one’s behavior.
6) Discrimination: No clear solution was found for this, but we could utilize the ledger to detect one’s actions recommending discrimination. For example, a master selecting the same worker even if he has a lesser trust value than the others available, or a master selecting the same worker and providing negative ratings for him while all the other masters he had provided positive ratings etc. Especially in a centralized approach we could use artificial intelligence to process the ledger in each iteration and detect patterns that may suggest discrimination.
7) Ballot stuffing: With the permissioned approach this issue is obviously addressed. Also, the fact that we require a transaction to take place for a rating to take place as well and only the involved parties may rate (master, worker and auditors) will completely eliminate the issue of ballot stuffing.

In this survey, we came across issues related to whether a user account will be associated with some real world ID or not, and thus we talked about the ability we will provide to the system’s admin to choose whether to use a permissioned system or a permissionless one. So, we also have to analyze the advantages and the disadvantages of each approach to make them clear and thus the admin being able to make the right choice regarding what he tries to achieve:

With a permissioned system the positive is that because users have provided some ID it will be impossible for them to create a new account and have the status of a newcomer again (in the case that they have a negative overall rating, so they are incentivized to do so). Positive to this is also the fact that because their ID is known it might incentivize them to avoid being malicious so that their public image is maintained. The negative with a user’s ID being known is that this might cause discrimination among users so the equal opportunity for growth within the system’s community is lost. Also, the fact that anonymity will no longer be offered by the system will cause that some users might not join the system just because of the fact that they will not be anonymous in the system’s community.

With a permissionless system the positive is that anonymity is offered and thus all the negatives of the previous approach are eliminated. The negative is that users might rejoin the system with different profiles and thus the feedback we had from their previous history is completely lost because we no longer know who they were and thus we can’t infer their past.

Now we will research for every system mentioned by the survey and check if it deploys a trust and reputation management system that we can study and then we will map this system to our framework (in the best way with the information given) and at the same time try to formulate our approaches for CrowdBED.

5.2.3 Systems Mentioned when Reference was made to P2P Networks

Napster

Napster is just mentioned as a P2P network without any other reference made to it later in the survey. An effort was made to find scientific articles or technical papers related to Napster’s reputation scheme online, but nothing was found that would guarantee to give us something to map to our framework.

Gnutella

Gnutella is mentioned as a P2P network. A paper is cited that talks about free-riding on Gnutella [40] in the survey [20] which we studied. Also, a research was made online to find scientific or technical papers related to Gnutella’s reputation system, but nothing was found. So, since the online search did not yield any results we could use, we just relied on the paper [40] cited by the survey [20] but no details about a potential reputation system work on Gnutella were mentioned there.

Freenet, KaZaA, grokster and iMesh

For all these P2P networks we did not find any citations related to their reputation systems and no scientific or technical papers were found, in the online search that was done for each of them, that explained their reputation systems.

5.2.4 Systems that use Simple Summation or Average of Ratings

Epinions

From the survey [20] we learn that Epinions is a product shop review site with a business model mainly based on so-called cost-per-click online marketing. We try to map the trust scheme of Epinions based only on information given by the survey [20] in the commercial and live reputation systems section since it is sufficient, so we don't have to search for empirical analysis or technical papers online.

The variable v_i:

Structure and Scale of the variable:

• Products and Shops get reviews and receive ratings (from 1 to 5 stars) for various aspects (Ease of Use, Batter Life etc.) so, we can consider for each of these aspects a parameter in the variable that represents a product v_i.ease_of_use_rating, v_i.battery_life_rating etc.

We have different parameters when it comes to shops and different when it comes to products.

Each review can receive a rating from other members in the scale of 4 values expressed literally (Not Helpful, Somewhat helpful, Helpful and Very Helpful). Reviewers can have a certain status:

• v_i.status which can be one of the three values (from lowest to highest: Advisor, Top Reviewer or Category Lead).

Reviewers without any status are just named members. Each member has a web of trust which is a list of other members that he/she trusts.

Perception of the variable:

We are not told anything about the perception of the various parameters of the variable, but we can assume that all parameters are global since users are able to see all the details of a review and a reviewer's status. The web of trust of a member doesn't have to be public so its perception is not something that would matter, of course if a member blocks or trust another member then that would be visible to that user and thus a local perspective would exist (i.e. a member knows if another member blocks or trust him).

<u>Visibility and sharing of the variable:</u>

As we said above, the parameters that involve the review and its reviewer are publicly visible from what the survey says. On the other hand, a member's web of trust doesn't have to be publicly visible since it contributes to an automatic procedure where a reviewer's status changes and thus it might be visible only in between the two members that the relation involves (i.e. have a web of trust relation).

<u>The method for calculation/updating of the variables:</u>

<u>Rating:</u>

- Members write product and shop reviews. A review consists of text and quantitative ratings for all aspects of the product or shop. So, everybody can rate a product or a shop.
- Members can provide a rating for some other member's review. So, everybody can rate a review.
- A member can trust or block another member.

<u>Ratings aggregation:</u>

- The system takes into consideration how the community rated a review and that determines how big will be the difference that this certain review will make on the rating of the product or shop it was written for.
- Also, how the community rates a review will have effect on the reviewer's status. The status of a member is a function of the accumulated ratings on all his reviews over a period.
- Category Leaders are selected at the discretion of Epinions staff. So, for one to elevate to the Category Leader status the central authority's approval is needed. This happens once each quarter.
- Top Reviewers are automatically selected every month based on how well their reviews are rated and on the member's web of trust (The number of members and their status who trust a given member, will contribute to that member getting higher status and the complete

opposite applies to the members who block that member). Advisors are selected in the same way but with a lower threshold. These thresholds are only known by the central authority.

The topology of the network in which the scheme/system works:

Centralized.

Strengths and Weaknesses of the scheme/system:

From the survey's authors' view: "The reputation system can be characterized as highly sophisticated because of the revenue based incentive mechanism."
From a critical point of view: The fact that all member opinions are considered (i.e. the web of trust) for a member to get a certain status and that the central authority needs to examine a user before elevating him to the Category Leader status make this scheme robust. Also, the fact that products and shops get ratings for various aspects also highlights how this scheme provide rich feedback to its users. Also, we can see this system trying to be user friendly by providing words or phrases that can describe what rating a user might want to provide to make it easier for users to provide a rating for reviews of other users. In addition, the revenue based incentive showcases that this scheme tries to achieve that the community will generate as much feedback as possible and thus make the system more accurate. Overall, this scheme is a very well designed one especially for the application that it is used for, but we have to remember that a central authority is needed for it to work.

Formulation of CrowdBED trust and reputation management systems from this system/scheme:

Just as this scheme does, ours could also use multiple aspects when a service is rated. But this depends on the nature of the services that will be provided in the platform, that our reputation system will be used for. If we want to create a general scheme for crowdsourcing platforms where there is a variety of tasks solved there and there are no specific aspects that all these task share, then we can't adopt this idea. What the solutions of the tasks are essentially evaluated for is, in our case is, correctness. We could add a rating for the speed aspect (how fast does the worker solve the task), but since the master sets a certain time window where he wants the task to be completed then the worker should be responsible for just that.

Here we will open a parenthesis and talk about how the time that it takes for a worker to solve the task should be a concern when we are designing such a reputation scheme that deals with a platform where workers provide services that rely on the use of the computational power of their machines.

First of all, let's list the roles of each entity in a transaction when it comes to this time setting:

The master when advertising a task to the network will be responsible for providing some metrics to showcase the difficulties of solving that particular task. That is important because a worker must know what he will deal with, so he is not accused of being malicious when the truth is that he was unable to solve such task. We already used a general term until now "size of task". But now, we can see that this general term alone is not enough. So, we will need the master to define some basic amounts of each resource of computation that will be needed. That is, the amount of memory that will be needed (i.e. size of the data to be processed) and the approximate time that should be needed to solve the task (here the master can be tolerant and allow for a bigger window if he wishes to) from which a user can infer the computational speed that he will need to solve it.

The worker will, after deciding he can solve the task with the information that was given to him, try to solve it (that is run the code sent to him by the master on the given data).

The auditors (if we decide to use auditors that is) will, after deciding that they can solve the task with the information that was given to them, try to solve the task (or part of the task if that is possible – in order to reduce auditing overheads) and then check first that the information given by the master was not malicious (i.e. false time window or false amount of data advertised) and after assuring that the master was not malicious (and provide a rating for the master) then go ahead and provide their rating on the worker, based on the correctness of his solution and whether he provided the answer in the time he was asked to do so. If, the worker did not provide an answer before the deadline the auditors must address this as they would address an incorrect solution.

So, since the correctness of a solution and provision of the solution in time are closely related, we will choose not to include a separate aspect for which a service must be rated for.

In this scheme users provide ratings for other users' reviews. We will not do that since, in our case, for a rating to take place a transaction must be associated with it. So, a user can't provide his opinion on what the correct answer is unless he actually solves the task, or he is the one setting it (i.e. have a role in the transaction).

Also, the idea of statuses is used here. We will not use this idea since the trust values are enough for what we want to achieve in our case and thus we will not add the overhead of something that will be of no actual use.

Amazon (review reputation scheme)

Details from this survey [20] for Amazon's reputation system are given in the section of the survey where commercial and live reputation systems are analyzed. We have already studied Amazon's system before from the previous survey [10]. In the previous survey we were not given as much detail about how reviews and reviewers are rated.

So, the survey begins by mentioning what we already know which is that reviews consist of text and a rating in the range of 1 to 5 stars and that these ratings are averaged to give the final rating. Then it continues with what we did not learned from the previous survey and that we will map to our framework (how the trust and reputation system works for reviews and reviewers).

<u>The variable v_i:</u>

<u>Structure and Scale of the variable:</u>

• v_i.helpful_votes and v_i.not_helpful_votes : reviews receive votes (they are stated as votes in the survey so that's the word we will use) that can take one of the two values, "helpful" or "not helpful". The number of "helpful" and "not helpful" votes is what represents the trust value for a review.

• v_i.reviewer_rank : reviewers have a rank.

• v_{yx}.favourite_people : Reviewers can have a relation with another member as to be one of their *Favourite People*, lets declare v_{yx}.favourite_people that is a boolean and has value equal to 1, if and only if y belongs to the *Favourite People* of x, and 0, otherwise.

Perception of the variable:

Global perception for all parameters of the variable.

Visibility and sharing of the variable:

- v_i.helpful_votes and v_i.not_helpful_votes are publicly visible
- We are not explicitly informed if the v_i.reviewer_rank is publicly visible, but we know that reviewers in the top 1000, top 500, top 100 etc. receive a status that is visible.
- Also, we are not informed about the visibility of the *Favourite People* each user has.

The method for calculation/updating of the variables:

Rating:

Members and non-members can vote (again that is the word used by the survey) for a review.

Ratings aggregation:

Each reviewer's rank is determined by the number of helpful votes received, as well as other parameters not publicly revealed. Also, the survey mentions that the number of people that list a reviewer as one of their *Favourite People* has an effect on the reviewer's rank.

The topology of the network in which the scheme/system works:

Centralized.

Strengths and Weaknesses of the scheme/system:

From the survey: The survey highlights that, this scheme has been reported to receive a lot of attacks both for malicious elevation and demotion of reviewers, and that is not unexpected since users can vote without becoming members. To strengthen the point it tries to make, the survey also mentions that people in the top 100 reviewers have mentioned spikes of negative votes. In the end it states that Amazon needs to address these issues and due to this vulnerability of the review scheme it can't be described as a robust scheme.

From a critical point of view: We will have to agree with the survey on the fact that this scheme is not a robust one because this scheme does not ask for users to register before providing a vote for a review which in turn will lead to malicious voting.

Formulation of CrowdBED trust and reputation management systems from this system/scheme:

There is nothing notable for us to utilize in our case (since we don't have such a concept in our application that maps to the concept of reviews that can exist on top of the main reputation system).

5.2.5 Systems that Apply Discrete Trust Models

Supporting trust in virtual communities

Paper [27] proposes a trust model based on sociological characteristics of trust.

The variable v_i:

Structure and Scale of the variable:

- v_i.trustworthiness : an agent's trustworthiness might be believed to negative or positive but on a short range of values (four-value scale). The trust degrees are: vt, t, u and vu with the respective meanings ("Very Trustworthy, Trustworthy, Untrustworthy, Very Trustworthy").

The paper afterwards separates the v_i.trustworthiness parameter to two kinds of trust that have the same scale as v_i.trustworthiness;

- direct trust and recommender trust. So, we declare two functions to get either from the parameter; direct trust = D(v_i.trustworthiness) and recommender trust = R(v_i.trustworthiness).

Perception of the variable:

Different observers may have different perceptions of the same agent's trustworthiness, so v_i.trustworthiness follows a local perceptive. D(v_i.trustworthiness) may be perceived different within certain context to a certain degree. R(v_i.trustworthiness) may have a different degree with respect to a context.

Visibility and sharing of the variable:

Not mentioned but we infer that every agent can shape an image for agents that he interacted with or that he received a recommendation for.

The method for calculation/updating of the variables:

Rating:

• Agents are able to exchange reputational information through recommendations. A recommendation might not necessarily represent the belief of the recommending agent (A recommender can lie or give out contradictory recommendations to different agents).

Ratings aggregation:

• All evaluations of recommendations consider the source of the recommendation.

• Further experiences and recommendations increase or decrease the level of trust of another agent v_i.trustworthiness. Experience is the result of evaluating and experience with an agent or relying on a recommendation from him.

• At any given time, the v_i.trustworthiness of an agent is obtained by the summary of the relevant subset of recorded experiences.

The topology of the network in which the scheme/system works:

Not defined. This is a trust model for virtual communities in general.

Strengths and Weaknesses of the scheme/system:

From the authors' view: From the paper [27] "… we looked at the issues of trust in society and outlined a model for supporting trust in virtual communities…" and "We acknowledged the ad-hoc nature of certain aspects of the model, namely the trust degrees and the weightings".

Also, no simulations of the model were run since it is mentioned “Finally, it would be interesting to look into simulating artificial societies that implement the trust model presented in this paper.”.

From a critical point of view: This scheme recommends an outline model while incorporating sociology. We can’t criticize it as we would with another fully developed scheme, but we can say that the application where it would be used will dedicate how well will it work as we as how the ad-hoc aspects of the trust degrees and weightings will work.

A formal model for trust in dynamic networks

A formal model of trust [29] informed by the Global Computing scenario and focusing on the aspects of trust formation, evolution, and propagation.

The variable v_i:

Structure and Scale of the variable:

• v_i.trust : Each entity (referred to as principal in this model) has a trust measure. The scale of the trust values is not strictly defined but rather various examples are given.

• function Degree(v_i.trust) : Each trust value has a degree.

• function uncertainty(v_i.trust) : Also, in this model the notion of uncertainty is used for each trust value. Uncertainty is used because principals may have only a partial knowledge of their surroundings.

• v_i.local_policy : each principal has a policy. This model does not specify the way a policy is actually defined.

Perception of the variable:

• Each agent can have a different belief for the trust value of another peer (that we can infer since the model mentions a function m(*a*)(*b*) that takes agent *a* and *b* as parameters and returns the v_b.trust from the perspective of *a*.

• For each principal his policy is local.

Visibility and sharing of the variable:

The model does not specify this, but a lot of times mentions situations where principals use trust values of other principals.

The method for calculation/updating of the variables:

Rating:

- Trust v_i.trust is based on observations. But an entity A can choose to enforce that its trust in C is the same as B's trust in C.
- Each principal's policy contributes by way of delegation to form the global trust.

Ratings aggregation:

- The principal uses a "trust-box" as it is named in the paper to complete two functions which are the one to send the trust value to another principal and the other to update trust based on an observation.
- Decisions made by the principals are based on the ordering of the various kinds of trust (trust value ordering and trust information ordering).

This is a general outline model so the rest of the framework mapping for it is omitted (as with other models afterwards).

5.2.6 Systems that Apply Belief Models

Trust-based decision making for electronic transactions

Paper [31] proposed a scheme for propagating trust through computer networks based on public key certificates and trust relationships.

The variable v_i:

Structure and Scale of the variable:

This model uses trust based on belief and declares three parameters *b* (belief that the other agent will cooperate), *d* (disbelief that the other agent will cooperate) and *u* (uncertainty) so we declare v_i.b, v_i.d and v_i.u where v_i.b + v_i.d + v_i.u = 1 for all agents. The opinion is composed by the aforementioned parameters and an atomicity parameter, so we declare v_i.atomicity and for the opinion we declare v_i.opinion(v_i.b, v_i.d, v_i.u, v_i.a). Also, there is a function on the opinion that represents probability expectation, so we declare E(v_i.opinion).

Perception of the variable:

Not stated.

Visibility and sharing of the variable:

Nothing stated about visibility. It is stated that propagating trust will be based on authentication by public key certificates combined with the trust relationships between users, so we know that agents will share trust. It is pointed out that first-hand evidence must be shared whenever a certificate is sent which means that an agent shares his opinions that are about key authenticity and cooperation trustworthiness with another agent but will not share second-hand evidence (i.e. opinions based on recommendations from other agents) never.

The method for calculation/updating of the variables:

Rating:

• The agents have opinions that provide as recommendations.

Ratings aggregation:

• The calculation of the probability expectation is given by the formula: $\mathrm{E}(\mathrm{v}_i.\mathrm{opinion}) = \mathrm{v}_i.\mathrm{b} + \mathrm{v}_i.\mathrm{a} \cdot \mathrm{v}_i.\mathrm{u}$

• Two opinions may be combined to yield an opinion about another agent. Example A's opinion of B might be combined with B's opinion of x to yield the A's opinion of x.

• The consensus of two opinions reflects both opinions in a fair and equal way and the Bayesian approach is used in this model to achieve that.

An Evidential Model of Distributed Reputation Management

Paper [33] proposes an evidential model of distributed reputation management.

The variable v_i:

Structure and Scale of the variable:

v_i.trust_belief : trust here is an enhancement of the mathematical view where a (scalar) metric is used to model a subjective probability with which an agent will perform an action.

Perception of the variable:

The two kinds of beliefs are distinguished: local belief and total belief. So, we can say that v_i.trust_belief can be seen from both perspectives from each agent.

Visibility and sharing of the variable:

An agent's local belief can be propagated to others upon request. Agents are restricted from propagating their total beliefs, however the necessary information underlying a total belief can be obtained by the requesting agent from original witnesses.

The method for calculation/updating of the variables:

When an agent A is evaluating the trustworthiness of an agent B there are two components as evidence. The first is the services offered by agent B and the second is the testimonies of other agents in case that A has had no transactions with B. To evaluate the trustworthiness of agent B, agent A will check if B is one of its acquaintances. If so, A will use its existing local belief to evaluate the trustworthiness of B, otherwise A will query its neighbors about B. In turn, when an agent receives a query about B it will check if B is one of its acquaintances, if yes it will return the information about B, otherwise it will return referrals to A, and A can then query these referrals and so forth. There is a limit on how deep this action will keep going (how many references to other agents will be made). If the depth limit has been reached and no ratings have been received by A then the process terminates in failure.

Rating:
An agent's local belief about a correspondent is from direct interactions with it.

Ratings aggregation:
An agent's total belief about a correspondent combines the local belief (if any) with testimonies received from any witnesses. (The agents will use their total beliefs to decide if a correspondent is trustworthy or not).

5.2.7 Systems that Apply Fuzzy Models

ReGreT

ReGreT [34, 35, 36] is a reputation model for gregarious societies which is based on social relations. This model is a natural extension of other models currently being used in the area of electronic commerce.

The variable v_i:

Structure and Scale of the variable:
This model uses three types of reputation measures: outcomes, impressions and subjective reputation.

• v_i.outcome : The outcome is defined as "both an initial contract to take a particular course of action or to establish the terms and conditions of a transaction and the actual result of the actions taken or the actual values of the terms of the transaction". An outcome is represented as a conjunction of equalities between variables (outcome features) and constants (values of the features). Variables can be common (reflect aspects of the contract agreed by both parts) or expected (reflect aspects implicitly supposed to happen by one of the two parts).

○ v_i.outcome_D : where *D* the certain dialogue (so for a dialogue D between two agents A and B there will be two different outcomes v_A.outcome_D and v_B.outcome_D since expected variables are linked to the subjectivity of agents).

• v_i.impression_O : where O is the outcome in question. The impression is defined as the subjective evaluation made by an agent on a certain aspect of an outcome.

• v_i.subjective_reputation : which also has a reliability measure.

In the second paper [35] when talking about the social dimension three types of reputation are mentioned: witness, neighborhood and system reputation.

• So, we declare v_i.social_witness_reputation, v_i.social_neighborhood_reputation and v_i.social_system_reputation.

<u>Perception of the variable:</u>

The paper [34] talks about individual and social dimension so we can guess that both perceptions exist for each agent the social where groups share common way of thinking and the individual which is defined only from the agent's own experiences. Also, the ontological dimension is mentioned where the reputation associated with just a single aspect (of the dialogues) can be combined with reputation associated with other aspects to calculate a more complex aspect's reputation.

<u>Visibility and sharing of the variable:</u>

It is stated that beliefs of the reputation of others can be shared and communicated by the members of a society. We know that common variables from outcomes and impressions on an outcome are shared between the two parties involved in the dialogue.

<u>The method(s) that will be used to calculate/update the variables:</u>

<u>Rating:</u>

• When considering the group relation three new sources of information are added (except the direct interaction) when calculating a reputation value: the interaction with group's members, the information that our group related to that agent, and the information that our group has related to its group.

• Witness reputation is calculated based on the information about the target agent coming from other agents.

<u>Ratings aggregation:</u>

• Subjective reputation is calculated directly from an agent's impressions database. To calculate a subjective reputation, we use a weighted mean of the impressions' rating factors, giving more relevance to recent impressions. The reliability of the subjective reputation is calculated using two elements: the number of impressions used to calculate the reputation value and the variability of its rating values (the deviation of the impressions' ratings).
• Neighborhood reputation uses the social environment of the target agent, that is, the neighbors of the target agent and their relations with it.
• System reputation is a default value based on the role played by the target agent.
• Finally, the system puts all the types of reputation (outcome, witness, neighborhood and system – in decreasing order of reliability) in respect to all the dimensions (individual, social and ontological) together.

In the last paper [36] a section is dedicated on integrating the ReGreT model with a negotiation model where the agents try to reach an agreement about the provision service.

Formulation of CrowdBED trust and reputation management systems from this system/scheme:

From all the types of reputation that are used in this scheme in our case it would only make sense to use only outcome reputation. That is because all the ratings given in our case must be related to a transaction and only parties involved in that transaction are asked to provide their opinion. We don't have neighborhoods and there are no entities in the system that have roles that need to be faced differently from others). For example, there is no such thing as a bias towards a master in a transaction.

Also, we will not use different dimensions of reputation and that is because all the trust values will be global. Of course, users will be able to form their own opinions if they want to do so, but that is not enough for us to say that an individual dimension exists (since the formation of individual opinions is up to the users and not predefined or enforced by the system).

The idea of using negotiation before a transaction is good to follow in the case that the task that the master needs to be solved can be separated into parts (e.g. distributed algorithm where a separate part can be solved regardless of the other parts) and thus the worker might offer to solve a certain part of the task instead of the whole task. This could lead to reduction of transactions not going well because the worker could not fully complete the task he was assigned to complete (because without the negotiation phase he could not express his ability to solve just a part of the task but also did not want to let his resources unused so he took the task).

5.2.8 Systems that Apply Flow Models

Appleseed algorithm

Appleseed [38] borrows ideas form spreading activation models in psychology and relates their concepts to trust evaluation in an intuitive fashion.

The variable v_i:

Structure and Scale of the variable:

This model defines one directed trust graph where nodes are the agents and edges are the trust statements between them with weights. So, we declare v_{AB}.trust_statement_weight the weight for the trust statement B makes for A.

Perception of the variable:

Not mentioned how the graph is interpreted by the agents so we can’t infer this.

Visibility and sharing of the variable:

From the paper “we assume that all trust information is publicly accessible for any agent in the system through machine-readable personal homepages distributed over the network”.

The method for calculation/updating of the variables:

This model talks about how the trust will be propagated mainly so we don’t have enough evidence to decide how exactly are the values calculated/updated. Although it is mentioned that Appleseed works with partial trust graph information and adds virtual edges in the graph for backpropagation from the node under question to the source that queries. From what we can infer the system runs the calculations for the trust values and not the agents individually.

Advogato's reputation scheme

This reputation scheme is described in the commercial and live reputation systems section of the survey [20]. Advogato [39] is a community for opensource programmers. It is also highlighted that "The Advogato reputation system does not have any direct purpose other than to boost the ego of members…"

The variable v_i:

Structure and Scale of the variable:

Each member has a status with which some other member refers to it, so we declare v_i.status, which can be one of the three values (apprentice-lowest, journeyer-medium, master-highest).

Perception of the variable:

The status of a node is seen the same from all other nodes, so we have a global perception of the variable.

Visibility and sharing of the variable:

The statues of a node is public.

The method for calculation/updating of the variables:

Rating:

Members rank each other on how skilled they perceive each other to be. So, everybody can vote everybody.

Ratings aggregation:

A member will get the highest status for which there is a positive flow to his or her node. Example from survey “For example, if the flow graph of Master referrals and the flow graph of Apprentice referrals both reach member x then that member will have Master status, but if only the flow graph of Apprentice referrals reaches members x then that member will have Apprentice status.”

The topology of the network in which the scheme/system works:

Centralized but this architecture is further analyzed by the survey as follows: “… members constitute the nodes and the edges constitute referrals between nodes. Each member node is assigned a capacity between 800 and 1 depending on the distance from the source node that is owned by Raph Levin who is the creator of Advogato. The source node has a capacity of 800 and the further away from the node, the smaller the capacity.”

5.2.9 Commercial and Live Reputation Systems Analyzed by the Survey

eBay’s Feedback Forum

We already studied eBay’s system and this survey does not give any new or different information than the empirical analysis [17] and the details of the previous survey[10], that we already studied, gave. The only part of the framework that this survey provides information for and is worth mentioning here is:

Strengths and Weaknesses of the scheme/system:

From the survey authors’ point of view: The eBay reputation system is very primitive and can be quite misleading. Despite the drawbacks and primitive nature, the eBay reputation system seems to have a strong positive impact on the eBay marketplace.

AllExperts

AllExperts is an expert site. The mapping of this system is done based on the details given by the survey [20].

<u>The variable v_i:</u>

<u>Structure and Scale of the variable:</u>

- v_i.knowledgeable_ratings_score, v_i.clarity_ratings_score and v_i.politeness_ratings_score : As stated in the survey ratings are given in the aspects “Knowledgeable”,” Clarity of Response”, ”Timeliness and Politeness” and are given in the interval (1,10).
- v_i.questions : The number of questions an expert has received is also displayed.
- v_i.general_prestige : each expert has a General Prestige.

Also, the ratings can be displayed for a certain time frame (2 months to 1 year).

<u>Perception of the variable:</u>

All the entities are displayed with the same ratings for an expert, so we have a global perception for all parameters of the variable.

<u>Visibility and sharing of the variable:</u>

All the entities can see ratings of an expert, so we have a public visibility for all parameters of the variable.

<u>The method for calculation/updating of the variables:</u>

The score in each aspect is simply the numerical average of ratings received. Also, v_i.general_prestige is simply the sum of all average ratings an expert has received.

<u>The topology of the network in which the scheme/system works:</u>

Centralized.

AskMe

This expert site is just mentioned in the survey in the Expert Sites part of the Commercial and Liver Reputation Systems and as it is stated in the survey "AskMe does not publicly provide any details of how the system works."

BizRate

We will try to map this system to our framework with information given from the survey [20]. BizRate runs a Customer Certified Merchant scheme.

The variable v_i:

Structure and Scale of the variable:

• v_i.navigation_rating, v_i.selction_rating, v_i.prices_rating, v_i.options_rating and v_i.satisfaction_rating : each BizRate listed store receives ratings for site navigation, selection, prices, shopping options and for satisfaction of the shopping experience.

• v_i.customer_cert : a merchant might or might not have a Customer Certificate. This takes the value 1 or 0 if the merchant has a customer certificate or not respectively.

• v_i.product_rating : products also are rated with reviews for certain aspects just as in the Epinions scheme.

• v_i.product_review_rating : a review can receive ratings in the range of the three values (helpful, not helpful or off topic – highest to lowest).

Perception of the variable:

For all parameters if they are visible, we can infer that there is a global perspective since the survey never mentioned something different. Especially reviews are viewed the same from all the members since members must be able to study them and rate them so it would not make sense if the opposite was true.

Visibility and sharing of the variable:

If a merchant has a Customer Certificate, then he can display a seal of approval on his website thus it is publicly visible when a merchant has a Customer Certificate or not. Since members can vote for other reviews it means that reviews are visible thus v_i.product_rating is publicly visible. We are not told about the visibility of the rest of the parameters.

The method for calculation/updating of the variables:

Rating:

• Customers who buy at a BizRate listed store are asked to rate the store's site. When a customer participates in the scheme becomes registered BizRate member.

• Members can vote for products as well. So, every member can vote for v_i.product_rating of a product.

• Members can also vote for reviews of products. Thus, every member can vote for the parameter v_i.product_review_rating.

Ratings aggregation:

Merchants receive their Customer Certificate only if enough surveys over a given period are positive.

The topology of the network in which the scheme/system works:

Centralized.

Strengths and Weaknesses of the scheme/system:

From the survey: There is an incentive for members to fill out surveys which is discounts to listed stores. So, it is more likely that members will contribute. This scheme mostly fails to capture negative votes though, as it is stated in the survey "This scheme does not capture the frustrated customers who give up before they reach the check, and therefore tends to provide a positive bias of web stores." but this in turn is incentive for store to join the scheme. When it comes to product reviews since there is no incentive for members (i.e. they will not get any reward from writing product reviews) and since anyone can go ahead and leave a review (even non – members) this part of the reputation scheme is highly vulnerable to attacks.

Slashdot

Slashdot (http://slashdot.org/) is a site were articles are posted and works as a discussion forum. Slashdot was also mentioned in the previous survey [10] as well but no details, for the reputation system to be mapped to our framework, were given there so we used [20] to map its scheme to the framework.

The variable v_i:

Structure and Scale of the variable:

• v_i.karma : Logged in users have karma values that take one of the discrete values (Terrible, Bad, Neutral, Positive, Good, Excellent) and new logged in users start with neutral karma.

• v_i.M1_moderator : Logged in users can be M1 moderators at a certain time. This is a boolean with value 1 if the logged in user is a moderator and 0 if not.

○ v_i.points : Each M1 moderator has some moderation points he can spend (5 points from the moment he becomes a moderator is the initial value, but this might vary depending on the user's karma).

• v_i.comment_verbal_ratings : Comments on articles receive ratings from M1 moderators, positive ratings are can be one of the values (insightful, interesting, informative, funny and underrated) and negative ratings can be of the values (off-topic, flame-bait, troll, redundant, overrated). So, we declare this for each comment. This parameter will be an array containing all these ratings and how many times each one has been received.

• v_i.comment_score : Comments on articles also have an integer score in the range (-1,5), and this score is initially set to 1 but if the comment provider has a specific value of Karma then three other separate cases of initial values might be met namely very high Karma will give initial score 2, very low Karma will give initial score of 0 or even -1.

• v_i.M2_moderator : Any longstanding logged in user can also become an M2 layer moderator (meta-moderator) so this will be a boolean with value 1 if the user is a meta-moderator and 0 if not.

• v_i.moderation_ratings : Moderations (ratings of comments by M1 moderators) also receive ratings which are the distinct values (fair, unfair or neither).

Perception of the variable:

Not stated explicitly.

Visibility and sharing of the variable:

Not stated explicitly.

The method for calculation/updating of the variables:

A longstanding logged in user can become a meta-moderator several times per day if he wishes to.

Rating:

• Articles posted are selected by Slashdot staff.

• Anyone can give comments to an article. That means anyone from the users where users here means logged in users or anonymous people as well.

• The central authority (Slashdot) every 30 minutes automatically selects a group of M1 moderators among long time regular logged in users.

• M1 moderators spent 1 point to give a negative or positive rating to a comment. A moderator's positive rating to a comment will mean 1 point increase to the comment's score v_i.comment_score while a negative rating will mean 1 point decrease to the comment's score, but the score will always remain within the range (-1,5).

• A user that meta-moderates will be asked to moderate the M1 ratings on 10 randomly selected comment postings. A meta-moderator decides if a moderator's rating was fair, unfair or neither. So, these v_i.moderation_ratings will affect the v_i.karma of the M1 layer moderator who is associated with them (i.e. made the moderation). And the v_i.karma of that logged user will in turn influence his eligibility to become M1 moderator in the future.

• Slashdot staff (which we consider as the central authority) are able to spend arbitrary amounts of moderation points.

Ratings aggregation:

• M1 moderators in the group of the automatically selected group are assigned with 5 moderation points each.

• Comments scores can also be influenced by the comment's provider karma value.

• Positive moderation of a user's comments contributes to higher Karma whereas a negative moderation of a user's comments contributes to lower Karma of that user.

The topology of the network in which the scheme/system works:

Two moderation layers; M1 for moderating comments to articles and M2 for moderating M1 moderators (meta-moderation layer). But overall, we have a centralized architecture.

Strengths and Weaknesses of the scheme/system:

From the survey's author point of view: For the distinct values used in the rating of comments the survey states: "The Slashdot reputation system recognizes that a moderator's taste can influence how he or she rates a comment. Having one set of positive ratings and one set of negative ratings, each with different types of taste dependent rating choices, is aimed at solving this problem." and then goes on to explain that different positive ratings will be uniformly positive and the same will happen with different negative ratings. The survey also states that the reason for Slashdot stuff being able to spend any amount of moderation points is to give them the ability to stabilize the system in case of attacks of extreme volumes of spam and unfair ratings. Finally, the paper states that "The system is constantly being tuned and modified and can be described as ongoing experiment in search for the best practical way to promote quality postings, discourage noise and to make Slashdot as readable and useful as possible for a large community."

<u>Formulation of CrowdBED trust and reputation management systems from this system/scheme:</u>

The main idea that Slashdot showcases is the use of moderators which kind of already have used in our case using auditors and that is the maximum extend we will use it and that is because equal opportunity must be top priority in a system like ours where resources are provided. So, we will not use for example such thing as elevated auditors whose job will be to audit other auditors because the fact that certain users are elevated to the point that they will have that extra ability of influencing directly other auditors' reputation values would suggest imbalances in our system that did not exist before. But we do have a separate trust value for the auditing skills of a user.

Kuro5in (Mojo)

A website (http://www.kuro5hin.org/) for discussion of technology and culture. The reputation system of Kuro5in is called Mojo. Not much detail about how this system works is given in this survey [20] we will just highlight the points that are related to our study here. Mojo had some problems regarding noise from attackers that rated down comments maliciously so that the comment provider would lose Mojo (the reputation score) so changes were introduced some of which are to only let a comment's score influence a user's Mojo (i.e. reputation score) when there are at least six ratings contributing to it and to only let one rating count from any single IP address. Finally, it is highlighted that had Kuro5in used the Slashdot's moderation principle the issue above (as well as others regarding the reputation system) could have been avoided.

Google's Web Page Ranking System (PageRank)

PageRank [37] represents a way of ranking the best search results based on a page's reputation. The survey argues that because PageRank ranks a page based on the number of pages pointing at it, we can describe PageRank as a reputation system. Hyperlinks are like public information that can be combined to derive a reputation score.

The variable v_i:

Structure and Scale of the variable:

Each page has a ranking v_i.ranking which is a value in the range (0,1).

Perception of the variable:

v_i.ranking is perceived the same from all users (global perception) that make the same search in the same time, but Google (the central authority in this case) perceives it in a different scale but still with the same relation to the other pages (e.g. if Google sees a relation v_x.ranking > v_y.ranking for any x and y then the users will see the same relation).

Visibility and sharing of the variable:

Page rankings v_i.ranking are provided to the public but are scaled in the range of (0,10) and in increments of 0.25.

The method for calculation/updating of the variables:

Rating:

Pages can give a positive rating to another page (i.e. have a hyperlink to that page).

Ratings aggregation:

- The ranking of a page is calculated as: $v_i.ranking = c \cdot v_i.ranking + c \cdot \sum_{u \in N(i)} \frac{v_u.ranking}{|H(u)|}$, where N(x) is the set of pages pointing to x, H(x) is the set of pages x points to and c is a constant such that the sum of all ranking values of pages in the search equals to 1. So, the initial ranking and the hyperlinks are the factors that determine a page's ranking.
- Other elements are also taken into count not published by Google in order to make it expensive or difficult to influence PageRank. Regardless, a high public ranking is needed in addition to matching keywords so that a page is presented to the user.

The topology of the network in which the scheme/system works:

Centralized.

Strengths and Weaknesses of the scheme/system:

From the survey's authors' point of view: Google managed to reduce the problem of pages spamming keywords and metadata so that they appear on searches because it requires a page to have a high public ranking as well in order to be presented to the user at all. Also, it highlights that PageRank applies the trust transitivity principle to the extreme because rank values can flow through looped or arbitrarily long hyperlink chains.

Supplier Reputation Systems (specific case: Open Ratings)

The survey [20] makes a reference to these systems stating that reputation systems can tackle the problem of information asymmetry and uncertainty, about supplier reliability, that makes it risky to establish supply chains and agreements online, by providing the basis for making more informed decisions and commitments. Then the example of Open Ratings (http://openratings.com/) is presented. Open Ratings sells Past Performance reports based on ratings of past contract partners. Ratings are in the range 1-100 for 9 aspects (reliability, cost etc.) and the score is a function of recently received ratings. Also, some feedback is provided about the partners that provided the ratings.

Scientometrics

Scientometrics [41] is the study of measuring research output and impacts thereof based on the scientific literature.

The variable v_i:

Structure and Scale of the variable:

For each paper we know the number of times it has been cited so we declare v_i.times_cited. Also, each of the entities paper, author and journal a has a ranking so we declare v_i.ranking.

Perception of the variable:
Global.

Visibility and sharing of the variable:
Public.

The method for calculation/updating of the variables:

Rating:
Scientific papers can cite other papers therefore increasing the number of times that paper has been cited (v_i.times_cited).

Ratings aggregation:
The ranking of papers v_i.ranking is identical to v_i.times_cited. A journal's ranking is given from $\sum_{i \in J} v_i.times\_cited$, where J is the set of papers published in the journal.

Strengths and Weaknesses of the scheme/system:

From the survey's author point of view: The author notes the fact that this scheme allows only positive referrals just like Google's PageRank which secures that plagiarized content papers will not be easily sanctioned with Scientometrics (since the original version of that content will already have received more positive referrals and thus others will mostly use the original version an so it will receive more positive referrals than the plagiarized version would need so that it is discovered easier than the original version). Finally, it states that it would be interesting to use the ideas of PageRank to evolve Scientometrics.
From a critical point of view: This system applies a very simple way of calculating reputation (just counting the referrals an article has received). We will agree with the survey's author with the fact that ideas of the PageRank could be implemented in the case of Scientometrics in order to consider not only how many times a paper was cited but also by who.

Formulation of CrowdBED trust and reputation management systems from both Google PageRank and Scientometrics:

We will not use this idea of considering and allowing only positive referrals to be given and received by users and that is because in our case we need to penalize users that behave malicious (in the case of PageRank and Scientometrics the service that is provided which is the paper or the page itself can't be malicious in the sense of the content that it provides in the same way that a user in our case fails to provide the service and that's why this difference exists). Also, in our case we are not so much afraid of false negative ratings since for each transactions auditing will be provided by the community. In addition, there is no need for us to be afraid of plagiarism since answers will be given at the same by all users in a transaction. So, we allow both positive and negative ratings in our case.

5.3 Study of Paper: "Reputation-Based Mechanisms for Evolutionary Master-Worker Computing"

Paper [42] provides mechanisms from which we can formulate our approaches for the CrowdBED platform. In addition, what this paper presents is closely related to our approach since we are using master-worker computing model as well.

5.3.1 Summary of Paper and Mechanisms Mapping to Framework

We will firstly summarize the ideas that this paper provides, that we didn't come across in the previously done research and map the proposed scheme to our framework.

The introduction presents past work done where three types of workers were considered (altruistic, rational and malicious) and two separate approaches were followed: the first approach is the one of considering that workers are either malicious or altruistic and thus malicious tolerant protocols were developed, and the second type of approach was the one that was considering workers to be rational and thus a game theoretic approach was followed. Then the paper mentions that previously done work was not considering the fact that a master can use the history of the worker to make his decisions.

After the introduction the proposed model is described. This model assumes that a unique solution exists for each task. Then the paper goes on to talk more about the types of workers and how it perceives correctness. Afterwards it is mentioned how the master can use auditing, reward/punishment schemes to incentivize workers to be honest and the idea of auditing based on a probability (will not always audit the answer), which obviously aims to decrease the overhead imposed by the auditing process, is proposed. Also, here we have the assumption that a master can receive multiple answers (so a majority answer can be chosen when no auditing is done). Here we are also introduced with the aspiration metric that each worker can have at a certain round (which represents the minimum expected benefit he can take form the specific round). The workers are unaware that a reputation scheme exists (and to keep that so rewards are not based on reputation). We will try to map this model to our framework:

The variable v_i:

Structure and Scale of the variable:

Three types of reputation are used here, so we declare v_i.reputation_t1, v_i.reputation_t2 and v_i.reputation_t3 but once the system starts running the master will have to choose one of the three types. The initial reputation is the same for all the workers.

Perception of the variable:

The works have no idea that a reputation system exists so they can't form such perception.

Visibility and sharing of the variable:

Hidden and existence is unknown.

The method for calculation/updating of the variables:

Transactions and thus updates of the reputation values as well are taking place in rounds (but for updates of the reputation values auditing must also take place and that happens based on a certain probability initially set by the master).

Rating:

The reputation is measured by the master.

Ratings aggregation:

Reputation of a worker here is based on the number of times it was found truthful (which can happen only after auditing) and that is true for all the types of reputation used by the model.

The topology of the network in which the scheme/system works:

For reputation computation this model uses a centralized reputation mechanism.

5.3.2 Formulation from the Mechanism and Summary

We will now formulate our approach for the CrowdBED platform using this summary of the paper and the mapping of the proposed scheme.

Using a probability to whether decide to audit an answer is not that good of an idea cause from this the possibility that a worker that cheated 10 times is not discovered and another that cheated just one time is found which would lead to lack of consistency, but if we necessarily need to decrease the overhead of auditing this would be a good solution.

We are informed that a centralized reputation mechanism is used in this model so it should help us since the mechanisms we will propose are centralized. Here we will have to note that in this model there is apparently only one master that assigns tasks to all the users which will not be true in our case since we will have multiple masters that will assign the task just to one worker and auditors to do the auditing.

This reputation mechanism is based on the fact that workers don’t know about its existence. We can’t adopt this idea because one of our goals when we decided to use a ledger was to achieve full transparency which is the exact opposite of hiding the fact that such mechanism exists from the users of the system.

5.3.3 Study of Schemes Mentioned in Paper

Now a study of the relevant schemes/systems mentioned in the paper will be carried out to map them to our framework and formulate from them mechanisms for CrowdBED.

Amazon’s MTurk

The paper cited just the link to the MTurk website [43], where we could not find any complete description of the reputation system(s) used there, so individual research for this system had to be made in order to find any scientific or technical paper that would provide information about how the reputation system. Paper [44] was found that provided enough information for us to do the mapping.

The variable v_i:

Structure and Scale of the variable:

• v_i.productivity and v_i.approval : Two measures are used productivity (how many tasks a worker completed) and the approval rate (the percentage of overall number of tasks completed that have been approved by the requesters).

• v_i.master_stat_S : a worker can reach a master status in a specific area of service. Here S is the specific service, and this parameter is a boolean with value true if and only if the worker in question is a master at the specific type of service.

Perception of the variable:

Requesters (of services - that’s how they are named here) can form their own opinions as well, if they want to do so. So, each requester can choose to form a local perspective.

Visibility and sharing of the variable:

Not stated in the paper what the requesters can see.

The method for calculation/updating of the variables:

The paper mentions that how a worker is elevated to the master status, in a specific type of service, is unclear.

Rating:

• We can infer that the requester will rate the worker's service as approved or not and that will change his v_i.approval.

The topology of the network in which the scheme/system works:

Centralized.

Malicious tolerant protocols where master decides the correct result based on majority voting

Here the workers were considered to be either malicious or altruistic. We will just mention the papers related to this section:

(1) Reliable Internet-based computing in the presence of malicious workers [45].

(2) Robust network supercomputing with malicious processes [46].

Sabotage-tolerance mechanisms for volunteer computing systems

In paper [47] various mechanisms are mentioned for which we can take those properties that make sense to our approach and we can formulate it from them.

Majority voting: the same task or part of task is solved several times and we choose the answer of the majority. For the task to be marked as done at least m (where m is predefined) votes on what the correct answer is must be collected and this m votes must also be the same or a majority percentage of the answers must be the same. The mechanism as described above requires the other users to solve the task as well and provide some answer. In our case we can assume two cases; one where the task has a unique answer and one where it has not one. Also, we could assume the case where the task is such that the auditors would be better off verifying the worker's answer (verification costs less than solving the actual task and then comparing their answer with the user – e.g. NP problems). All these assumptions will lead each one of them to us deciding whether to use multiple workers or auditors and subsequently whether we will use majority voting to choose the answer or some other technique.

Spot checking: Master test the worker giving him just a spot of the task to solve (that the result is already known) and only after the master has seen that the worker gave a correct answer will assign the rest of the task to him. As expected, this is not as accurate as the majority voting mechanism but will provide for smaller overheads.
When designing our approaches, we must be flexible with what the systems administrator will want to achieve (minimize computation overheads VS maximize accuracy of detecting malicious users) so this mechanism would be good to consider but we have to note that for such mechanism to work then the task's nature must allow it (the computation of a part of the task must not require the computation of other parts so this mechanism can actually work).
Alternatively, the idea, of solving just a part of the task before the worker is given the responsibility of solving the entirety of it, can be adapted for the auditors if we will use any (because we could just go with the idea of using multiple workers). That is, the auditors might solve and or verify just parts of the task and do majority voting on these parts. But again, for this to be possible the nature of the task must allow it. This way while we can recruit more auditors to achieve better accuracy each individual auditor will have less work to do in order for his auditing to be completed and receive his reward.

Afterwards the paper mentions the fact that the two aforementioned mechanisms can be combined in various ways (which we already considered above when formulating as well) and finally goes ahead and presents the multiple ways that this can happen.

Game-theoretic approaches

Here the workers were considered to be rational.

Uncheatable distributed computations

One of the most notable subjects that paper [48] discusses is how participants (workers in our case) will be rewarded for finding a number (solving a task in our case) and how a set of values that act like milestones along the computation process can be expanded so that more participants will be able to receive rewards.

Afterwards discussion is made about the Ringers basic scheme where the assumption, that a supervisor that is trusted from the whole participant population exists, is made. Ringers here are values in a domain of a function that are chosen by the supervisor in order to do spot-checks for the work of participants.

After both the aforementioned mechanisms are used together in a hybrid scheme. Where ringers are hidden using the magic numbers so that they will be hidden from the participants.

Again, here the fact that information is hidden from the participants contradicts what we are trying to achieve with the ledger we are using.

5.4 Study of Paper: "Proof or Reputation: A Reputation-Based Consensus Protocol for Peer-to-Peer Network"

Here the authors present a reputation-based consensus protocol [49] (PoR), on top of a blockchain, that uses reputation to both incentivize for good behavior and block publication. So this makes sense to our study since the reputation mechanisms that will be proposed could as well as be used in combination with the ledger provided by CrowdBED in the same sense that reputation is used in this scheme and that is to influence how the blockchain will be updated and thus make it part of the reward scheme which will incentivize users to participate in block publication to gain trust.

5.4.1 Summary of Paper and Mapping of the Proposed Mechanism

The introduction talks about the two kinds of studies on reputation systems: centralized and distributed, and then analyzes they positives and negatives of each one of them. It is also mentioned that this approach chooses permissioned blockchains and attributes that to the fact that reputation is inherently tied to identity and needs time to accumulate. All of the participants will maintain the distributed ledger here and the safety of the reputation information is attributed to the cryptography used by the blockchain.

Afterwards the paper highlights, when talking about related work, how other approaches failed to be truly decentralized, left the computation of reputation to entities out of the network and require users to be online to verify a transaction.

Then the problem that this paper tries to solve is defined which is achieving reputation aggregation and maintenance of the blockchain without using a cryptocurrency as the incentive. Then it mentions how the consensus process that it uses is cost efficient and how the double spending problem doesn’t exist here. Public keys are stored by participants to authenticate identities and verify transactions.

Then the threat model is presented to which we will need to pay attention and keep in mind when designing our approach. We summarize the various attacks that the paper mentions below:
1) Bad mouthing attack: dishonest recommendation from a user to decrease reputation of other user(s) and elevate his own.
2) Replay attack: duplicate some transaction in order to increase the outcome it has (to reputation in our case).
3) On-off attack: When a malicious participant switches its behavior suddenly in order for him to be not detected.

4) Sybil attack: When multiple accounts are created by an attacker in order for him to use another account in the case that he is caught being malicious.

Then the paper talks about the proposed protocol which we will try to map to our framework.

The variable v_i:

Structure and Scale of the variable:

The blockchain itself will be considered here since the history and therefore are kept there and every user has it. We declare v_i.trust.

Perception of the variable:

Global (all users share the same blockchain eventually).

Visibility and sharing of the variable:

Public (distributed blockchain).

The method for calculation/updating of the variables:

The one who has the highest trust value in a group of transactions can package them into a block and publish it (how consensus is kept and thus the blockchain is updated and therefore the trust values). Publishing a block can increase the overall trust rank of the publisher. When a certain number of transactions are completed the node to publish stops receiving new transaction data and calculates the new trust ranking list based on these transactions.

(Ratings):

At the end of each interaction a piece of feedback will be generated by the service requestor, recording the rate of the service.

Ratings aggregation:

To decide whether a participant is malicious or not, the sigmoid function is used.

5.5 Tables of all Schemes by Application, by Topology and Attribute Analysis

Now we present two tables that categorize all the studied schemes (both from Chapter 4 and 5), Table 5.1 by application and Table 5.2 by topology. Also, a third table is given: Table 5.3 that gives an attribute analysis of all schemes. The goal of this categorizations and the attribute analysis is to showcase the diversity of the extensive bibliographic study and provide the reader with a summary.

Application	Schemes/Systems
Road information exchange between vehicles	VANETs reputation scheme [9]
Social networking	Facebook, Twitter reputation schemes
Online market-places	eBay [17,18], amazon [19,20]
Crowd-sourcing	Wikipedia, Quora[22], Amazon's MTurk [43,44]
Online facilities sharing	Uber [23], blablacar.com [24], Airbnb [25,26]
Product review sites	Epinions [20], BizRate [20]
Expert sites	AllExperts [20], AskMe
Discussion Fora	Slashdot [20], Kuro5in
Search Engines	Google's PageRank [37]
Scientometrics	Scientometrics [41]
Consensus Protocol	PoR [49]

Table 5.1: The studied schemes/system categorized by their application

Network Topology	Schemes/Systems
P2P Grid	H-trust [1], FR-Trust [7]
P2P over MANETs	M-trust[2]
Unstructured P2P	R^2-Trust [3], EigenTrust [4], RCert Trust[8]
Decentralized P2P - SOPSys	[6]
Distributed - VANETs	VANETs reputation scheme [9]
Centralized	Facebook, Twitter, eBay [17,18], Amazon [19,20], Quora[22], Uber [23], Airbnb [25,26], Epinions [20], BizRate [20], Slashdot [20], Google's PageRank [37], Amazon's MTurk [43,44]
Distributed	Scientometrics [41], PoR [49]

Table 5.2: The studied schemes/system categorized by the topology they work on

Reputation Scheme Identifier	Centralized	Uses Global Trust	Uses Local Trust	Uses Separate Credibility Factor	Public trust variable	Collective Opinions	Public votes	All aggregate	Permissioned Participation	Direct interaction voting only	Use of Distrust	Implicit reputation info.
H-Trust	-	-	+	+	+	+	?	+	?	-	-	-
M-Trust	-	-	+	+	+	-	-	+	-	-	-	-
R^2-Trust	-	-	+	+	-	-	-	+	-	+	+	-
EigenTrust	-	+	+	-	+	+	?	+	-	-	-	-
FIRE	-	-	+	-	Partially	-	Partially	?	-	+	-	-
[6]	-	+	-	-	-	+	-	-	-	-	-	-
FR-Trust	-	+	+	-	-	-	-	-	-	+	-	-
RCert Trust	-	+	-	-	+	-	+	+	-	+	-	-
VANETs [9]	-	-	+	-	+	+	+	+	-	-	-	-
ebay.com	+	+	-	-	Optional	+	?	-	+	+	-	-
amazon.com	+	+	+	+	+	-	?	-	+	+	-	-
quora.com	+	+	+	-	Partially	+	-	-	+	+	-	-
uber.com	+	+	-	-	+	+	-	-	+	+	-	-
blablacar.com	+	+	-	-	+	+	-	-	+	+	-	-

Reputation Scheme Identifier	Centralized	Uses Global Trust	Uses Local Trust	Uses Separate Credibility Factor	Public trust variable	Collective Opinions	Public votes	All aggregate	Permissioned Participation	Direct interaction voting only	Use of Distrust	Implicit reputation info.
airbnb.com	+	+	-	-	+ Host. – Quest.	+	-	-	+	+	-	-
Epinions	+	?	?	Not exactly. Status is used though.	Partially	+	-	-	+	-	-	-
Amazon (reviews)	+	+	-	-	Partially	+	-	-	- (non-members vote)	-	-	-
[27]	?	-	+	-	-	+	-	?	?	-	-	-
[29]	?	-	+	-	?	+	-	?	?	-	+	?
[31]	?	?	?	+	?	+	?	?	?	-	+	-
[33]	-	+	+	-	Partially	-	-	+	-	+	-	-
ReGreT	?	+	+	+	+	+	-	+	?	-	-	+
Appleseed	?	?	?	-	+	-	-	-	?	?	-	+

Reputation Scheme Identifier	Centralized	Uses Global Trust	Uses Local Trust	Uses Separate Credibility Factor	Public trust variable	Collective Opinions	Public votes	All aggregate	Permissioned Participation	Direct interaction voting only	Use of Distrust	Implicit reputation info.
Advogato	+	+	-	-	+	+	?	-	+	-	-	-
AllExperts	+	+	-	-	+	+	-	-	?	-	-	-
BizRate	+	+	-	-	-	+	+	-	+	-	-	-
Slashdot	+	?	?	-	?	-	-	-	?	-	-	-
PageRank	+	+	-	-	+	+	+	-	-	-	-	+
Scientometrics	-	+	-	-	+	+	+	-	+	-	-	+
[42]	?	Users don't know of existence.		-	-	-	-	-	?	+	-	-
Amazon's MTurk	+	-	+	-	?	-	-	-	+	+	-	-
PoR [49]	-	+	-	-	+	-	?	-	+	+	-	-

('+' : Has attribute, '-' : Has not attribute, '?' : Can't state with certainty or Unknown)

Table 5.3: The studied schemes/systems attributes analysis

Chapter 6

Proposing a Permissioned Centralized Reputation Scheme for CrowdBED

After the formulation that was done before, from studying the various schemes, now a permissioned centralized approach will be presented. Also, certain system parameters and assumptions were specified that guided this approach in a certain direction and which we mention when presenting this scheme below.

First off all, we will give the general image of what is the purpose of the system, where the reputation system is working, and then we will define what we try to achieve with this reputation scheme. We have a crowdsourcing platform where any user that seeks an answer for a certain task (that can be computed with a program and data that this user will provide) will be able to connect with users that are willing to utilize their computational resources, calculate the answer and return it to receive a reward.

The goals of our reputation scheme are, first to ensure that the possibility that the final answer, that the user that made the request of service will receive, is wrong is minimized as much as possible, secondly to minimize the possibility that the program and data, that the user that makes the request of service provides, contain viruses or anything malicious or that the user sets an unrealistic deadline. Also, we need our reputation scheme to contribute in the system's defense against various types of attacks, that are based on strategies that rational users that find in their best interest to be malicious at a certain time frame form, and we expect that is possible to be carried out.

6.1 System Specifications

In this section the system specifications will be analyzed, and these are true for both approaches that are suggested in this work except from the participation policy (permissioned and permissionless) which is the difference of the two approaches that are proposed and which we discuss further in this section.

6.1.1 Participation Policy: Permissioned VS Permissionless

This is a key decision to be made and although we propose approaches for both cases discussion will be made to further understand the positives and the negatives of each of the two. Also, this will guide the decisions that will be made in order for the schemes to offer the system administrator with solutions for both types of systems.

A permissioned system would provide us with a higher level of security than a permissionless one and that is because of the fact that an identity will be asked to be provided, from an interested to join entity, before it is granted with membership on the system and thus it becomes a user. The fact that the identities of users are known entails a lot of positive progress regarding the goal we want to achieve with the reputation scheme before we even use it. First of all, users will be incentivized to avoid being malicious since their identity's reputation will be at stake, so this will drastically reduce the number of malicious users by default. Secondly, all the kinds of attacks that are associated with creation and control of multiple accounts within the network to manipulate the reputation system maliciously will not be possible (or we can say at least carrying them out will be very hard because creating multiple fake identities is harder than creating multiple public keys which is what can happen with a permissionless approach). The negative of using a permissioned approach is the obvious loss of anonymity, which will discourage a certain number of potential users to join the system (and here we assume that users who did not plan to be malicious as well to be discouraged because if this only affected users that were predisposed to be malicious then this could only be seen as a positive towards our goal). This however (the loss of anonymity) is possible to be contained to a certain degree: suppose the central authority asks for the identity to be provided when one joins the system but that identity is not shared with the other users of the network by the central authority and instead the public key that will be given to that new user will be displayed in the transactions he is involved with. Also, one can argue that a permissioned system suggests more control by the central authority or certain nodes of the network and thus transparency is reduced, but if we give the same abilities to all users that is, we use a ledger for transparency and all users can make the same transactions (except introducing new users to the network – which is only done by the central authority) then this argument can't stand since transparency is achieved with the ledger where all entities have the same influence and it is not manipulated by the central authority.

A permissionless system has the exact opposite positives and negatives mentioned above for the permissioned approach. That is, it ensures anonymity since no identity is needed to join, it is transparent in the sense that it ensures that we will have no special nodes (based on their identity at least), but our system will need to be designed in such a way to defend against the attacks that are inherent with a permissionless approach that we mentioned before.

6.1.2 Publication of Answers

The system has been designed in such a way that it can check that answers have been calculated before a certain deadline, that the master has set, (using a proof of submission) and make the answers available only after the deadline time has passed. This is done with the aim to eliminate the issue of answer copying/stealing/sharing between users that have been assigned to the task.

6.2 Assumptions

Just as in Section 6.1 where the system specifications are described here the assumptions that are made before we proceed with proposing the reputation schemes for CrowdBED are described to form an outline and be able to take deterministic decisions when designing the reputation schemes.

6.2.1 Worker's Answer Errors and Communication Errors Tolerance

We assume that any errors that are related to communication or accuracy of the answer (e.g. decimal places of a number) or data corruption are all in the responsibility of the worker. The proposed mechanisms are not designed to deal with such issues (i.e. we don't propose a mechanism to detect such errors and act using a separate reward/punishment mechanism that is specific for this issues). So, we assume full responsibility of the worker for the provision of the correct result.

6.2.2 Master's Responsibility for Task's Answer Existence

For the master we assume that he has to make sure that the task he submits to the network has an answer (i.e. there is no case where the submitted task will not have a valid answer) and that the deadline he sets is a realistic one and any worker will be able to solve the task if he dedicates his resources for it.

6.2.3 Uniqueness of Correct Answer to Task

We assume that each task has only one answer that it is correct. This means that the program provided by the user that makes the request for service is a deterministic one and after it is executed with the given data can only produce one answer. With this assumption done we can make certain conclusions that we can later use when designing the reputation system:

If answer$_X$(t) is the answer of a user *X* for the task *t* and CORRECT(a) means that answer *a* is correct, then:

$$answer_X(t) = answer_Y(t)\ \Lambda\ CORRECT\big(answer_X(t)\big) \rightarrow\ CORRECT\big(answer_Y(t)\big)$$

and,

$$answer_X(t) \neq answer_Y(t)\ \Lambda\ CORRECT\big(answer_X(t)\big) \rightarrow\ \neg CORRECT\big(answer_Y(t)\big)$$

6.2.4 Auditing a Task Response

Here we assume that task answers can only be verified by executing the task's program with the task's data. Meaning, for a given answer for a task a user can't go ahead and verify it with some other program that will make it faster for him (e.g. The task has NP complexity and thus verifying the answer will cost less than recalculating the whole task). The only way of verifying an answer in our case is calculating what the task requires and then compare the given answer with the answer calculated. So, since we have made this assumption, there is no reason for us to use auditors (users that have the job to just verify a task's answer) but instead multiple workers will be used.

6.2.5 Partial Calculation of Task Answer

We assume that tasks cannot be broken down to parts. So, a user can't offer to solve just a part of a task. From this assumption we can conclude that there will be no negotiation phase prior to task assignment because there will be nothing to negotiate. A user who is interested in working with a task will be assigned and calculate the entirety of it.

6.2.6 Minimum Computational Power Threshold

For users that want to join the system to solve tasks we will specify a minimum computational power threshold so that we avoid problems associated with deadlines being inaccurate. We will provide users that will assign the tasks as well with this specification, so they are responsible to adjust their deadlines accordingly.

6.2.7 Task Data Size

A minimum requirement of memory capacity will be defined when a task is advertised. The user that will choose to solve the task will be responsible about possessing that memory capacity (i.e. he has to deny getting the task assigned to him if he doesn't possess that memory capacity).

6.3 Network Entities

Users: We are using two roles:

Master: The user that needs a task to be solved and will provide a reward for the answer to the task.

Worker: The user that uses his computational power to try and solve a task for a reward.

We allow a user to be able to have both roles within the system (but not in the same transaction of course).

Central Authority:

Since the mechanisms are centralized a central authority exists by default.

6.4 Identities within a Permissioned Network

We could also design the system in such a way that an identity will be provided by the user but only the central authority will know of it. When interacting with the network the user will be using a public key assigned to him and not the identity it provided thus the anonymity is not lost within the network.

After these specifications and assumptions, we will now present the approach for a permissioned centralized network.

6.5 Permissioned Centralized Approach

The central authority in this approach only intervenes in the joining process, so an effort is made achieve as much decentralization of the system.

6.5.1 Reputation Variable v_i

Structure and Scale

For each user we declare the following parameters.

v_i.master_trust : represents the probability that the user as a master will cooperate (will not be malicious). Real number in the range (0,1).
v_i.worker_trust : represents the probability that the user as a worker will cooperate. Real number in the range (0,1).
v_i.master_reputation : represents the reputation of a user to be a master that will cooperate. Real number in the range (0,1).
v_i.worker_reputation : represents the reputation of a user to be a worker that will cooperate. Real number in the range (0,1).
v_i.credibility : represents the probability that a user's opinion is the correct one (will lead us to trusting the correct users that will cooperate). Real number in the range (0,1). Here we decided to use a separate parameter so that we separate the safety we feel for a user to be a good worker or a good master and the safety we feel that he will provide good opinions on other users, just as schemes [1,2,3] do. The importance of a separate value for this concept is also highlighted in [10].
v_i.opinions : integer number, the opinions that the user have given.
v_i.correct_opinions : integer number, the correct opinions (those that led to trusting a user that did not acted maliciously) that the user have given.

Perception of the variable
v_i.master_trust : Global.
v_i.worker_trust : Global.
v_i.master_reputation : Local to the user.
v_i.worker_reputation : Local to the user.
v_i.credibility, v_i.opinions, v_i.correct_opinions : Global.

Visibility and sharing of the variable
All visible in the ledger.

6.5.2 Joining the Network

The moment a user joins the network his variable parameter values will be set as follows:
v_i.master_trust = 0.5, v_i.worker_trust = 0.5

v_i.master_reputation = v_i.worker_reputation = unknown (for all users since they have had no evidence to form an opinion for this user yet)

v_i.credibility = 0.5, v_i.opinions = v_i.correct_opinions = 0

6.5.3 Transaction Process

We suppose that the master for the particular transaction is a user labeled X.

6.5.3.1 Advertisement

The master uploads the task to the platform Figure 6.1 and then it is advertised to the network Figure 6.2. This advertisement will include the minimum memory capacity required, for the program and the data that it will work with, the deadline (the latest time of answer submission) and the reward that will be provided to each user that calculates the task. Users that have their clients online will then receive that advertisement.

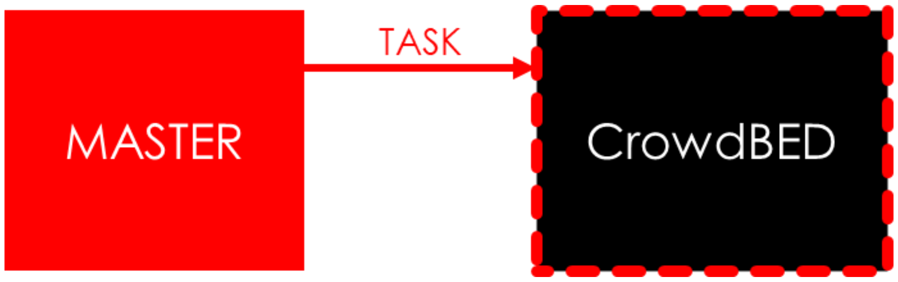

Figure 6.1: The master uploads the task to the platform

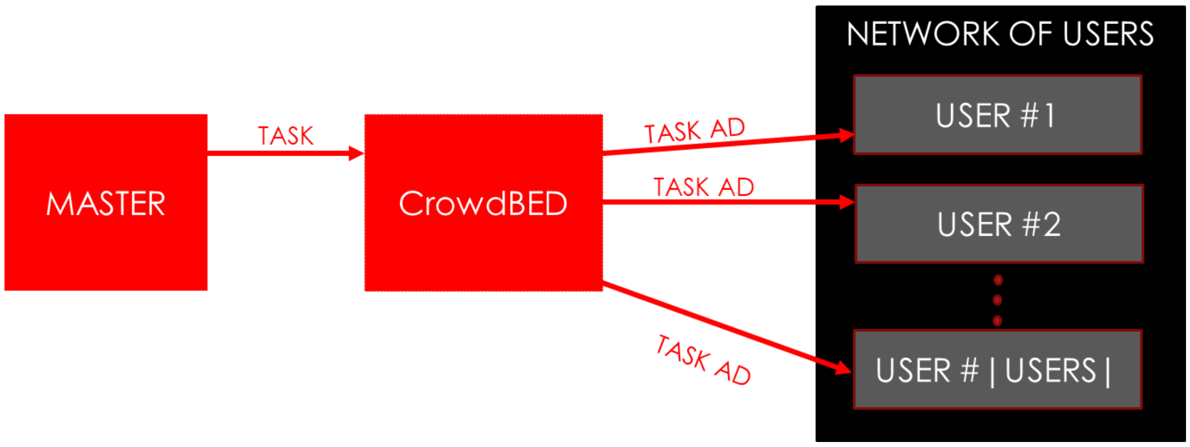

Figure 6.2: The task is advertised to the network

6.5.3.2 Assignment

The user after receiving the advertisement will have some time to decide, by consulting her knowledge of the value of v_X.master_reputation and in the case that this value is unknown to her the v_x.master_trust value, if she wants to solve the task. Afterwards and if she decided that she wants to solve the task, the user, by comparing her memory capacity and the required memory capacity to solve the task, will decide whether she can solve the task or not. All the users that have decided that they want to, and can, solve the task will return an acknowledgment back to the master as soon as they make this decision. The moment the acknowledgment is receive the user will enter the pool of potential workers for the task and this process can be seen drawn in Figure 6.3. After the decision time window is over, we will proceed to the ranking phase. Now for each user a selection pool rank will be calculated. We declare set USERS the set of all users and V(Y) the set of users that have a knowledge of v_Y.worker_reputation (they have an opinion for user Y as a worker) and are willing to provide it as an opinion (i.e. they are willing to put their credibility at stake), at the time of the assignment process. Also, for a user k and a user Y we declare v_{Yk}.worker_reputation to be the value that user k knows for v_Y.worker_reputation. In addition, ack_time represents the time in milliseconds that the user needed to return the acknowledgment. The selection pool rank for a user Y will be calculated as follows:

$$
\begin{aligned}
&selection\_pool\_rank(Y) \\
&\quad = \sum_{k \notin V(Y) \wedge k \neq Y} \left(\frac{v_k.credibility}{\sum_{p \in USERS \wedge p \neq Y} v_p.credibility} \cdot v_Y.worker\_trust \right) \\
&\quad + \sum_{k \in V(Y) \wedge k \neq Y} \left(\frac{v_k.credibility}{\sum_{p \in USERS \wedge p \neq Y} v_p.credibility} \cdot v_{Yk}.worker\_reputation \right) \\
&\quad - \left(\frac{ack\_time}{10^9}\right)
\end{aligned}
$$

In short, a weighted average (with credibility being the weight) of the collective opinions is calculated for each user minus a fraction of the acknowledgment time which is used as a tiebreaker. Schemes such as [1,2,5,19,20,34,35,36] used the weighted average of the opinions and this method of aggregation makes sense to the parameters we chose to use. Also, collective opinions is common among schemes that aimed to achieve high accuracy [1,4,6] and we are using this idea as well since we already have the communication overhead of the ledger consensus protocol and it would be a waste not to use the data that are available in the ledger to achieve better accuracy (i.e. assign the task to users that are more likely to solve it correctly).

Then the users are sorted in descending order of their selection pool rank and the first *N* users are selected to act as workers in the transaction (the *N* users with the greatest selection pool ranks) the whole process is described in Figure 6.4 diagrammatically. The value of *N* is selected by the system admin as we discussed in the previous approach. We define the set of the selected users to act as workers for task t as *workers(t)*.

For the later evaluation of the opinions that users provided we store the opinion given by a user *U* for a worker *W* for task *t* as *opinion(U,W,t)* which equals to v_{WU}*.worker_reputation* at the time of the assignment process.

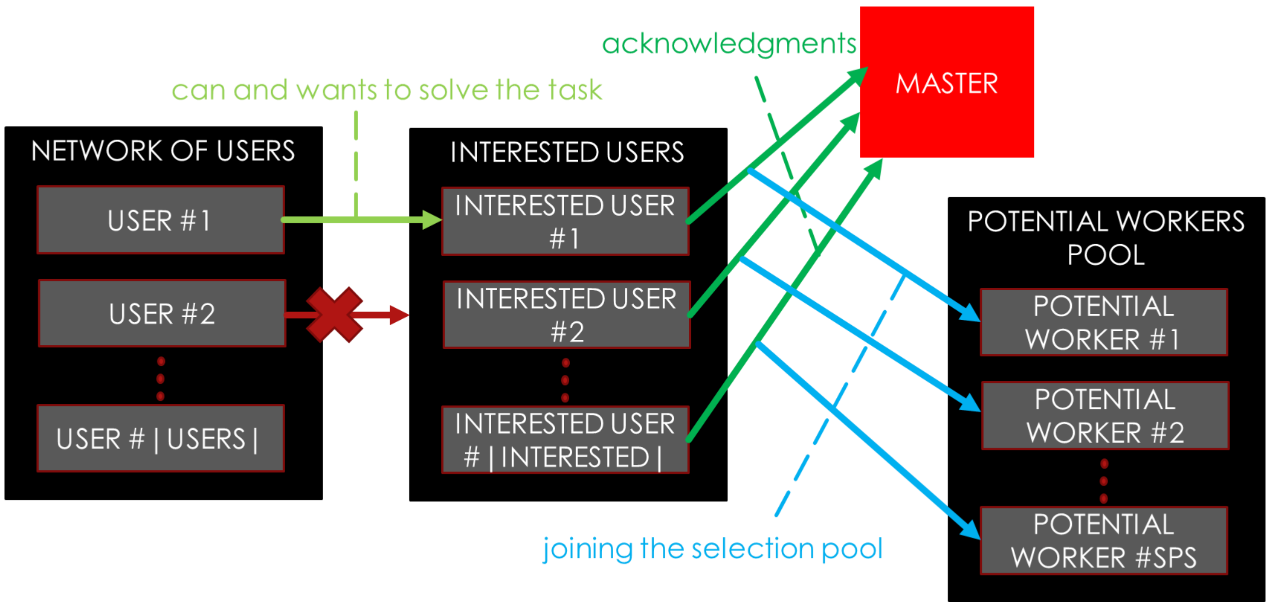

Figure 6.3: The user joins the potential workers pool (SPS = selection pool size)

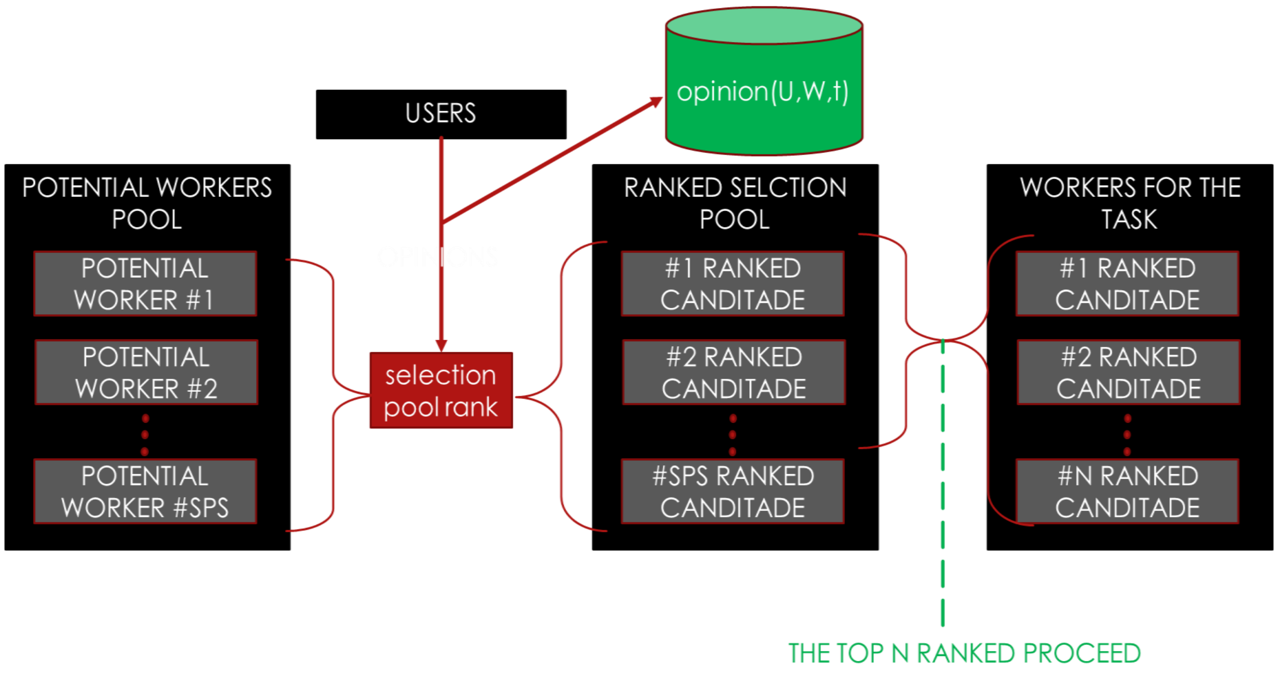

Figure 6.4: Ranking the workers and selecting the N with the highest rank. Saving opinions for future evaluation.

6.5.3.3 Answering

When the deadline is over all the workers will provide their answers. If no answer has been calculated by a worker, then that worker's answer will be empty, and a time limit exceeded flag will be raised for that worker.

6.5.3.4 Answer/Result Selection

Here we will utilize the idea of majority voting that is mentioned in [42] and is used in [45,46,47]. We follow the conditions bellow in the order given to find the correct answer/result. Once a condition is met then the correct answer/result is found as well, and we will proceed to updating the trust and reputation variables based to that answer/result. We define result$_W$(t) to be the result that worker *W* gave for task *t*. The set of users that gave result *R* to task *t* is defined by the following equation:

$$\forall W \in workers(t), (W \in G_t(R) \leftrightarrow result_W(t) = R)$$

(1) Answers are results returned by users that completed the calculation of the task in time (i.e. provided a result before the deadline was over). We declare the set of such results as ANS(t). If the following is true:

$$\exists a \in ANS(t), \left(\frac{|G_t(a)|}{workers(t)} \geq \frac{51}{100}\right)$$

Then that answer *a* that has been given by the 51% (or more) of the worker population will be deemed as the correct one.

2) Deadline time limit exceeded flag (DTLE) is the result that is returned by workers that failed to calculate an answer in the time they were given. DTLEs are not answers (i.e. do not belong to ANS(t)). If the following is true:

$$DTLE \notin ANS(t), \left(\frac{|G_t(DTLE)|}{workers(t)} \geq \frac{51}{100}\right)$$

Then we will assume that the master was malicious and has provided a task that could not be solved in the time frame that he provided.

3) If none of the above conditions was met, we will turn to another ranking system of the different results (we say results here to include both answers and time limit exceeded flags) to choose what we will deem as the correct one. For a result *R* and a task *t*, we declare the equation below to calculate the trust rank of *R*:

$$trust\_rank(R) = \sum_{W \in G_t(R)} v_W.\, worker\_trust$$

and the result with the greatest trust rank will be deemed as the correct one.

4) In the case that we had a draw when calculating trust ranks of results and we did not have a clear winner we calculate the following ranking for results that is based on the selection pool ranks and thus will be a tiebreaker:

$$result\_sp\_rank(R) = \sum_{W \in G_t(R)} selection\_pool\_rank(W)$$

and the result with the greatest $result\_sp\_rank$ will be deemed as the correct one.

We label the result that was chosen to be the correct one with the method we just described above, for task *t*, as *correct_result(t)*. So, the master will receive the *correct_result(t)* and we will update credibility and trust based on this *correct_result(t)*.

The mechanism we have provided for the result selection will be difficult to defeat because of the main obstacle the worker that orchestrates an attack on this mechanism (i.e. gives a malicious answer and tries to make the mechanism deem that answer as the correct one) will have to overcome for his attack to be successful. The main obstacle is that he has to control 51% or more of the worker population, so that it returns the same result as he does, which is something that he will have to achieve in the assignment process (i.e. the workers he controls have high trust and reputation values so that they are selected from the potential workers pool). This is hard because he needs to time the attack and because other users will have to provided good opinion for his workers which subsequently means that these workers need to be elevated in advance which means that resources have been used to elevate them (completing tasks and providing correct answers) which is of great cost and expensive for the malicious worker. If the malicious worker does not overcome this obstacle he will have to deal with many risks when deploying his attack; First of all he does not know if the other workers will all be honest and provide the same correct answer and thus he gets punished along with the workers he controls, secondly even in the case that the other workers are not all honest so they don’t win the correct result by majority he does not know if they will win it by average trust. So, we can see that this mechanism is a robust one since it makes it really expensive for the attacker to succeed and requires that he makes a great effort on timing the attack.

6.5.4 Updating of Variable’s Parameters after Selecting Correct Result

The credibility values of all users will be updated after the correct result has been selected. Algorithm 6.1 describes the way of this updating:

```
1: for each worker $W \in workers(t)$ and each user $U$ do
2:     if $opinion(U, W, t) \neq unknown$ then
3:         $v_U.opinions \leftarrow v_U.opinions + 1$
4:         if $(opinion(U, W, t) \geq 0.5) \wedge (result_W(t) = correct\_result(t))$ then
5:             $v_U.correct\_opinions \leftarrow v_U.correct\_opinions + 1$
6:         end if
7:         if $(opinion(U, W, t) < 0.5) \wedge (result_W(t) \neq correct\_result(t))$ then
8:             $v_U.correct\_opinions \leftarrow v_U.correct\_opinions + 1$
9:         end if
10:    end if
11: end for
12: for each user $U$ do
13:     if $v_U.opinions \neq 0$ then
14:         $v_U.credibility \leftarrow \frac{v_U.correct\_opinions}{v_U.opinions}$
15:     else
16:         $v_U.credibility \leftarrow 0.5$
17:     end if
18: end for
```

Algorithm 6.1: Updating credibility values

Now a scenario will be presented where the functionality of the mechanism suggested by Algorithm 6.1 will be showcased.

We consider a worker W_1 and users U_1 and U_2. Also let’s assume that both have given their opinions for worker W_1 for task t. We also assume that opinion(U_1,W_1,t) < 0.5 and opinion(U_2,W_1,t) >= 0.5.

Now we have two possible cases:

(1) Worker W_1 has returned the correct result for task t.

In this case the credibility of U_1 will be decreased since the number of opinions provided by him will increase but the number of correct opinions provided by him will remain the same.

At the same time the credibility of U_2 will increase since the number of correct opinions provide by him will increase.

(2) Worker W_1 has returned a wrong result for task *t*.

The opposite of what happened in the first case will happen now and that is credibility of U_1 will increase and credibility of U_2 will decrease.

We can see that in both cases the mechanism rewards/punishes the credibility of each user fairly depending on the opinion that they gave. A binary approach when considering the opinion of a user is followed (<0.5 or >= 0.5) and that is to incentivize users to actively adjust the reputation values that they perceive or follow the global trust values.

The trust values of the involved users (i.e. the master and the workers) will also be updated after the correct result has been selected. The following algorithms describe this update. The constants for increasing/decreasing the values that are used in the algorithms are indicative and the system admin is free to change them to adjust the rewards/punishments as he wills.

```
1: for master $X$ do
2:     if $correct\_result(t) = DTLE$ then
3:         $avg\_wtrust \leftarrow \frac{\sum_{W \in G_t(DTLE)} v_W.worker\_trust}{|G_t(DTLE)|}$    ▷ average trust of workers that raised a DLTE flag
4:         if $v_X.master\_trust \neq 0$ then
5:             $wVm \leftarrow \frac{avg\_wtrust}{v_X.master\_trust}$    ▷ DTLE workers' vs master's trust
6:         else
7:             $wVm \leftarrow 2$
8:         end if
9:         $v_X.master\_trust \leftarrow max(0, v_X.master\_trust - min(0.2, 0.1 * wVm))$
10:    else
11:        $CR \leftarrow correct\_result(t)$
12:        $avg\_crwtrust \leftarrow \frac{\sum_{W \in G_t(CR)} v_W.worker\_trust}{|G_t(CR)|}$    ▷ average trust of workers that returned the correct result
13:        $avg\_wrwtrust \leftarrow \frac{\sum_{W \in workers(t) \wedge W \notin G_t(CR)} v_W.worker\_trust}{max(1, |workers(t)| - |G_t(CR)|)}$    ▷ average trust of workers that returned a wrong result
14:        $crwVwrw \leftarrow \frac{avg\_crwtrust}{max(2 * avg\_wrwtrust, avg\_crwtrust)}$    ▷ correct result workers vs wrong result workers
15:        $v_X.master\_trust \leftarrow min(1, v_X.master\_trust + 0.1 * crwVwrw)$
16:    end if
17: end for
```

Algorithm 6.2: Updating the master's trust

Now we will present a scenario to showcase how the mechanism described by Algorithm 6.2 is fair.

First of all, we punish the master if he has provided a task for which the result that was deemed to be correct was the deadline time limit exceeded from the result selection process. The punishment is based on the average trust of the workers that raised that flag. So, if less trustworthy workers raised this flag then the less the punishment will be for the master. Also the value *wVm* which dictates the punishment is based on the master's trust so a more trustworthy master will receive a smaller punishment than a master that has already low trust which means that he was spamming tasks to the system that had a deadline that was impossible to for the workers to keep up to and thus now has a lower trust so the greater the punishment.

In the case that we don't have a DTLE then the master's trust will be rewarded for providing a legitimate task to the community (i.e. a task that had an answer and a deadline that was realistic for all participating workers). Just as with the punishment, when giving a reward to a master the average trust of workers that gave the correct result will be compared to the average trust of workers that provided a different result than the one that was deemed correct so that the reward will be proportional to these trusts.

We can consider a case where the workers of a task t are {W_1 … W_k , W_{k+1} … W_{k+z}} so we have k+z workers in total, let's consider that $k \geq \frac{51}{100} * (k+z)$ thus if the k workers return the same result then that result will be deemed as the correct one (the first rule for choosing the correct answer is 51% majority). Now if the workers {W_1 … W_k} were maliciously teaming up to elevate the master and demote the other workers then the mechanism since it compares their average trust to the average trust of the other workers {W_{k+1} … W_{k+z}}, then the higher the average trust of the other workers is the more it will contribute to minimizing the reward the master will receive. So, a master will receive a greater reward if less workers that are trustworthy oppose the deemed correct result and vice-versa. So, the mechanism utilizes the trust values collectively when rewarding the master to counter this kind of attack (teaming up to gain majority and elevate master in the expense of the rest of the worker population), by minimizing the reward the master will receive.

```
1: $CR \leftarrow correct\_result(t)$
2: $avg\_CRtrust \leftarrow \frac{\sum_{W \in G_t(CR)} v_W.worker\_trust}{|G_t(CR)|}$   ▷ average trust of correct result's workers
3: for worker $U \in workers(t)$ do
4:     $UR \leftarrow result_U(t)$
5:     if $UR \neq CR$ then
6:         $avg\_URtrust \leftarrow \frac{\sum_{W \in G_t(UR)} v_W.worker\_trust}{|G_t(UR)|}$   ▷ average trust of workers that gave the result worker $U$ gave
7:         $crVur \leftarrow \frac{avg\_CRtrust}{avg\_URtrust}$   ▷ correct result vs U's result workers' trust
8:         $v_U.worker\_trust \leftarrow max(0, v_U.worker\_trust - min(0.2, 0.1 * crVur))$
9:     else
10:        $avg\_WRtrust \leftarrow \frac{\sum_{W \in workers(t) \wedge W \notin G_t(CR)} v_W.worker\_trust}{max(1, |workers(t)| - |G_t(CR)|)}$   ▷ average trust of workers that returned a wrong result
11:        $crVwr \leftarrow \frac{avg\_CRtrust}{max(2 * avg\_WRtrust, avg\_CRtrust)}$   ▷ correct result vs not correct result workers' trust
12:        $v_U.worker\_trust \leftarrow min(1, v_U.worker\_trust + 0.1 * crVwr)$
13:    end if
14: end for
```

Algorithm 6.3: Updating worker's trust

Now we will provide a scenario to showcase how the mechanism provided by Algorithm 6.3 will work.

Here we can consider the same attack we mentioned before when proving the functionality of Algorithm 6.2. Again, we consider the same set of workers {$W_{1 \dots} W_k$, $W_{k+1 \dots} W_{k+z}$} and the same populations so that the relation $k \geq \frac{51}{100} * (k + z)$ is true. Now for each worker we have two cases:

(1) The worker (let's label him *Wq*) belongs to the malicious group that managed to get their result be deemed as the correct one ($i.e. W_q \in \{W_1 \dots W_k\}$):

Then the reward he will receive will be based upon the comparison of the average trust of the malicious group against the average trust of the rest of the worker population and thus the greater that trust is the lowest the reward of this malicious worker will be.

(2) The worker does not belong to the malicious group that managed to get their result be deemed as the correct one ($i.e. W_q \notin \{W_1 \dots W_k\} \rightarrow W_q \in \{W_{k+1} \dots W_{k+z}\}$) :

Then the punishment he will receive will be based upon the comparison of the average trust of the malicious group against the average trust of the workers that provided the same result as *Wq*, so the greater the trust of these workers that provided the same result as *Wq* the less the punishment he will receive.

So, we can see that in both cases the effect of the rewards/punishments will be adjusted by the trust differences between parties which means that higher trust will allow honest workers to defend their trust values and lower trust attackers will not achieve a significant change with a single successful attack.

For the reputation parameters v_{YU}.worker_reputation and v_{YU}.master_reputation we allow each user U to change their values, for each other user Y, in the way he wants, but only after a transaction involving Y having the respective role (i.e. worker or master) has taken place. A user U has to change the values of parameters v_{YU}.worker_reputation and v_{YU}.master_reputation from the initial state of 'unknown' to some real number in the range (0,1) the moment that a transaction, that involved both U and Y having roles in it (both are workers, or one is master and one is worker), has taken place. As a default way of updating the reputation values we recommend the same way with which trust values were updated above. A user U can reset the value of the parameters v_i.worker_reputation and v_i.master_reputation to 'unknown' when a predefined, by the system admin, amount of time RCD (reset cooldown) has passed since he last interacted with user Y and Y had the respective role. We allow this freedom to the users to set their opinions the way they want to (with the limitations that would make this functional of course and that we described above) because we have assumed anonymity within the network and with the ledger these values are transparent so the users will have to set these reputation values responsibly because if they don't the other users will have knowledge of this, since these values are in the ledger, and they will adjust their behaviors accordingly.

Chapter 7

Proposing a Centralized Reputation Scheme with Permissionless Participation for CrowdBED

In this approach we will assume that joining the network will not require the provision of an identity. Also, this scheme will allow all the users to execute the same set of actions and there will not be any form of any essential central authority intervention thus a permissionless approach will be proposed.

In [20] we have seen how important identity is for a reputation system to function. If a reputation system does not use identities then the longevity of the entities, in the network which it governs, will be affected. That is, users with low reputation scores might decide (if they are rational) to abandon their current account and create a new one and thus the entity that is described by the data of the abandoned account will never be active again. So, all the opinions that were formed for that entity will be of no use. Another attack that can take place against a permissionless system is the Sybil attack where a user creates multiple account in order to manipulate the network. So, we need to focus specifically on the issue of identity absence, that comes with a permissionless approach, when designing the reputation system, to deal as effectively as possible with attacks associated with it.

The purpose of our scheme is the same as before; create a reputation scheme that will work with the CrowdBED platform and we are assuming that no central authority exists but the ledger and the rest of the functionalities of CrowdBED are still in play. Again, the scheme will aim to complete the same goals as before, plus this time we will need to face the extra attacks that are inherent to the permissionless nature of the scheme. The assumptions, system specifications and network entities are identical to the previous approach except that we have a permissionless system now and no central authority is present.

7.1 Permissionless centralized approach

7.1.2 Reputation Variable v_i

Structure and Scale:

For each user we declare the following parameters:

v_i.master_trust : represents the probability that the user as a master will cooperate (will not be malicious). Real number in the range (0,1).

v_i.worker_trust : represents the probability that the user as a worker will cooperate. Real number in the range (0,1).

v_i.master_reputation : represents the reputation of a user to be a master that will cooperate. Real number in the range (0,1).

v_i.worker_reputation : represents the reputation of a user to be a worker that will cooperate. Real number in the range (0,1).

v_i.credibility : represents the probability that a user's opinion is the correct one (will lead us to trusting the correct users that will cooperate). Real number in the range (0,1). Again, we use this because of its importance [1,2,3,10].

v_i.opinions : integer number, the opinions that the user have given.

v_i.correct_opinions : real number, the correct opinions (those that led to trusting a user that did not acted maliciously) that the user have given. We will use a real number because of special cases of updating that we will see below (e.g. joining rate punishment).

v_i.inactivity : describes how much participation was missed by the user from what the system needed him to have, real number in range (0,1).

v_i.master_transactions : the transactions in which the user had the role of the master

v_i.worker_transactions : the transactions in which the user had the role of the worker

Perception of the variable:

v_i.master_trust : Global.

v_i.worker_trust : Global.

v_i.master_reputation : Local to the user.

v_i.worker_reputation : Local to the user.

v_i.credibility, v_i.opinions, v_i.correct_opinions : Global.

v_i.inactivity: Global.

Visibility and sharing of the variable:
All visible in the ledger.

7.1.3 Joining the Network

The moment a user joins the network his variable parameter values will be set as follows:
v_i.master_trust = 0.25, v_i.worker_trust = 0.25. We set these values lower than what we did with the permissioned approach. That is because we want newcomers to be punished to counter change of identity attacks in the expense of punishing potentially non-malicious users as well, just as it was mentioned in [20].
v_i.master_reputation = v_i.worker_reputation = unknown (for all users since they have had no evidence to form an opinion for this user yet)
v_i.credibility = 0.25. Again, lower than the permissioned approach to counter change of identity attacks. v_i.opinions = v_i.correct_opinions = 0.

After a certain number *T* of transaction rounds (concept which we explain later), we will have a count of the newcomers *T* rounds before the user joined and *T* rounds after the user joined. So, if a user U joined at transaction round 0, we have the number of all newcomers in the range of transaction rounds (-T,T). Now to make Sybil attacks more difficult, we will design the scheme is such way that when the number of users joining the network in a period of time peaks (is more than P), we will punish all the users that have joined in that period. The constants T and P are left for the system developer to set. We declare newcomers(X,Y) to be equal to the number of users that joined the system from transaction round X to Y, inclusive. Algorithm 7.1 will describe how parameter values will be updated after T transaction rounds from the moment a user joined have passed (constant values used are indicative and can be changed by the system developer):

```
1: $punishment \leftarrow 0.05$    ▷ indicative value and can be changed by the system developer
2: for user $U$ that joined before $T$ transaction rounds do
3:     $jtr\_U \leftarrow current\_transaction\_round - T$    ▷ the joining transaction round of U
4:     if $newcomers(jtr\_U - T, jtr\_U + T) > P$ then
5:         $v_U.master\_trust \leftarrow v_U.worker\_trust \leftarrow v_U.credibility \leftarrow max(0.25 - (punishment * \frac{newcomers(jtr\_U-T,jtr\_U+T)}{P}), 0.15)$
6:         $v_U.correct\_opinions \leftarrow (v_U.credibility * v_U.opinions)$    ▷ resetting the correct opinions value to negate effects that took place in the current lifetime of U
7:     end if
8: end for
```

Algorithm 7.1: Punishing newcomers based on the current joining rate

Now a scenario will be presented to showcase the functionality of the mechanism Algorithm 7.1 suggests.

Let us assume that the attacker *A* plans to spawn a set of users {U_1 … U_N} in the range of rounds T_{start} to T_{end}. From the algorithm we can see that the average punishment each of these users will receive upon joining the system will be greater for smaller T_{end}-T_{start}. Let's consider three cases for comparison and further proof. In all cases we assume T = 5, P = 2 and {U_1, U_2, U_3} the be the set of users attacker *A* tries to spawn, we also consider that no other users join the network in the time frame (T_{start} - T , T_{end} + T) where *A* is spawning his puppet users. The spawn times of {U_1, U_2, U_3} are T_1, T_2 and T_3 respectively.

(1) $T_1 = T_2 = T_3$. Here the punishment will be multiplied by $\frac{|\{U_1,U_2,U_3\}|}{P} = \frac{3}{2}$. So, the initial values for all three users (for trust and credibility) will be set to $0.25 - \left(0.05 * \frac{3}{2}\right) = 0.175$.

(2) $T_1 = T_2 - 5$, $T_1 = T_3 - 10$. So, for user U_1 we have two users spawn in (T_1-T, T_1+T) thus his initial values will be set to $0.25 - \left(0.05 * \frac{2}{2}\right) = 0.20$, for user U_2 we have three users spawn in (T_2-T, T_2+T) thus his initial values will be set to $0.25 - \left(0.05 * \frac{3}{2}\right) = 0.175$ and for user U_3 we have two users spawn in (T_3-T, T_3+T) thus his initial values will be set to $0.25 - \left(0.05 * \frac{2}{2}\right) = 0.20$. So, the average initial value for the users will be $\frac{0.20+0.20+0.175}{3} = 0.1916$.

(3) $T_1 = T_2 - 6$, $T_1 = T_3 - 12$. So, for all users the initial values will be set to $0.25 - \left(0.05 * \frac{1}{2}\right) = 0.225$.

From the three cases presented we can see that the greater the spawning time difference each user has from the others the higher the average initial values of the users will be. So no matter which strategy the attacker will choose to follow he will have to give something up and that is either the puppet users will have lower average trust and credibility values or a lot of time will need to be spent in order to finally spawn a puppet user population with good initial trust values (which will still have to be improved since newcomers are punished anyway). This mechanism will be used in the discretion of the system administrator (for example in the beginning of the lifetime of the system it is more possible that the users joining rate will pick so it would be good to disable this mechanism then, or he might choose to not use it at all).

7.1.4 Transaction Process

Transactions will take place in rounds; all transaction rounds will have the same time length which will be predefined by the system developer. A transaction can take from 1 to several transaction rounds. We suppose that the master for the particular transaction, that we describe as example, is a user labeled X and the task that will need to be solved is labeled *t*.

7.1.4.1 Advertisement

The master advertises to the network the task he wants to be completed. This advertisement will include the minimum memory capacity required, for the program and the data that it will work with, the deadline (the latest time of answer submission), the number of transaction rounds and the reward that will be provided to each user that calculates the task. Users that have their clients online will then receive that advertisement.

7.1.4.2 Assignment

After a user has received the advertisement and to her judgment, as we described in the permissioned approach, decides to join the network she will need to return a positive answer, in the time window that will be set for receiving interest of work acknowledgments, and then enter the selection pool for the task *t*. *N* of the users that were interested will be assigned with the task after the selection process that we describe below is finished and *N* will be a constant number (the same will be used in all transactions) and will be set by the system developer.

To make Sybil attacks even more ineffective at this point of the transaction a mechanism will be described, that the system administrator will be free to use or not depending on the policies he wants to follow and in his own discretion. This mechanism will provide a defense mechanism against the efficiency of Sybil attacks in the expense of forcing the expansion of the group of trust a user will have (it will not deem a user unable to form a group of trust it will just force him to expand it in some degree and we will see why when the mechanism is presented below).

Two relationships are introduced here that will exist between users:
$M\_W(U_1,U_2)$ = In how many transactions user U_1 had the role of the master and user U_2 had the role of a worker.
$W\_W(U_1,U_2)$ = In how many transactions both users U_1 and U_2 had the role of workers.
These relationships will be used to monitor that users do not favor specific users to work with them. Namely, we do not want a user having more than MTPL (mutual transactions percentage limit) of his transactions mutual with another user and with the same roles. So, to make sure that this happens we set Algorithm 7.2 to check for a user's eligibility to be assigned a task. We declare the set of users interested to be workers for task t to be labeled as *selection_pool(t)*.

```
1: $MTPL \leftarrow 0.3$                                              ▷ indicative value
2: for user $U_1 \in selection\_pool(t)$ do
3:     if $|selection\_pool(t)| <= N$ then
4:         BREAK
5:     end if
6:     if $MTPL < \frac{M\_W(X,U_1)}{max(1,min(v_x.master\_transactions,v_{U_1}.worker\_transactions))}$ then
7:         $selection\_pool(t) \leftarrow selection\_pool(t) - U_1$
8:     else
9:         for user $U_2(U_2 \neq U_1 \wedge U_2 \in selection\_pool(t))$ do
10:            if $MTPL < \frac{W\_W(U_2,U_1)}{max(1,min(v_{U_2}.worker\_transactions,v_{U_1}.worker\_transactions))}$ then
11:                if $v_{U_2}.worker\_transactions < v_{U_1}.worker\_transactions)$ then
12:                    $selection\_pool(t) \leftarrow selection\_pool(t) - U_1$
13:                else
14:                    $selection\_pool(t) \leftarrow selection\_pool(t) - U_2$
15:                end if
16:            end if
17:            if $|selection\_pool(t)| <= N$ then
18:                BREAK
19:            end if
20:        end for
21:    end if
22: end for
```

Algorithm 7.2: Ensuring a MTPL between users

So, what this mechanism essentially does is not allowing a user to interact with another user in a degree that could suggest a relation that is built upon a team effort that started off with a Sybil attack.

Now that we know who the eligible users are, we need a way to rank them. After the ranking is done, we will pick the top *N* ranked users and assign them with the task. To make this ranking we need opinions from other users.

Again, at this point a mechanism will be proposed to reduce the effectiveness of Sybil attacks. The system administrator again in his discretion will be able to use or not use this mechanism for the same reason as with the mechanism above.

We will first of all check which users are eligible to provide their opinions (have not offered their opinions, when the master of the transaction was X, too many times compared to all the other users). These users will belong to the set *eligibleOP(t)*, eligible opinion providers for task t. The size of this set will be at least MOPL (minimum opinion providers limit) where MOPL is a constant (same for all transactions) and is set by the system developer. The following relationship and Algorithm 7.3 will be used to do so:

M_OP(U_1,U_2) = In how many transactions user U_1 had the role of the master and user U_2 had the role of an opinion provider.

For this relationship TOPL (transaction opinion provider limit) will be used in the same sense that MTPL was used before.

```
$TOPL \leftarrow 0.3$
$eligibleOP \leftarrow USERS - X$
for $U_1 (U_1 \in eligibleOP)$ do
    if $|eligibleOP| <= MOPL$ then
        BREAK
    end if
    if $TOPL < \frac{M\_OP(X,U_1)}{max(1,v_X.master\_transactions)}$ then
        $eligibleOP \leftarrow eligibleOP - U_1$
    end if
end for
```

Algorithm 7.3: Ensuring not hearing opinions from same users repetitively

The set eligibleOP will contain all the opinion providers for the worker selection process. The selection pool rank for each user that is eligible and interested to be a worker will be calculated in the same way that it did with the permissioned approach; weighted average of the opinions (with the weight being the credibility of the user providing the opinion) and the acknowledgment time as a tiebreaker. The *N* users with the top selection pool rankings will then be assigned the task. Here we do not use collective opinions since not all users are eligible to provide their opinions, but we do take all the opinions of the eligible users.

7.1.4.3 Answering

When the deadline is over all the workers will provide their answers. If no answer has been calculated by a worker, then that worker's answer will be empty, and a time limit exceeded flag will be raised for that worker.

7.1.4.4 Answer Selection

This will be done in the same way that it was done in the permissioned approach; majority voting alternatively trust ranks with selection pool ranks as tiebreakers.

7.1.5 Updating of Variable's Parameters after the Transaction:

When it comes to updating of the variable's parameters, based on the outcome of the transaction, we will use the same algorithms that were described for the permissioned approach. What will be added to the updating of variable's in this approach will be not based on the outcome of the transaction (will not be part of the transaction process) but on the transaction rounds and is presented below.

7.1.6 Updating of Variable's Parameters after each Transaction Round:

Finally to incentivize the longevity of accounts in the network which is very important as mentioned in [20], eliminate the effect of opinions from abandoned accounts and to make it expensive for Sybil attackers to maintain and elevate their puppet user accounts, we will add a reward/punishment mechanism based on the transaction rounds that a user will participate in a transaction (i.e. is active). This mechanism will treat user differently based on the value of the parameter that will be updated (trust or credibility – reputation values will not be updated using this mechanism since what they describe is essentially the opinion of a user which the system allows it to be dedicated by the user himself). Since the punishment/reward will be based on the activity of the user in the interval of some transaction rounds we will need to know how often (i.e. after how many transaction rounds) the mechanism will work, so we use a constant TRAE (transaction rounds before activity evaluation) which the system developer will predefine. In the previous study, scheme [1] also used a periodical mechanism for updating trust and credibility values. The values for the demand of participation and reward punishment will also be left to the system developer to set (in the example below literals high, medium and low will be used). The mechanism is described with Algorithm 7.4.

Now the choice of reward/punishment values will be explained to showcase their correctness and the incentive that the users will receive.

We assume the case where a user's parameter under question is lower than 0.25:
Here the user is either a newcomer or he has been malicious in the greater part of his lifetime in the system and he is a situation of, so he has started from the beginning. So, we need to incentivize the user to show a bit of effort to get a reward and increase his activity within the community and thus when its inactivity is not high in the timeframe that the mechanism works with, he will receive a medium reward to start climbing. If he is highly inactive then the possibility is that the account is inactive so we need to reduce its effect on the platform (the opinions it provides will not be up-to-date) and thus a medium punishment is received.

We assume the case where a user's parameter under question is in the range (0.25,0.5):
Here the user is past the newcomer phase and his effect on the community starts growing. If the user has a low measure of inactivity which means it is highly active the he will keep climbing faster than before, so a high reward is received to incentivize that the high activity continues. We can see that it will be expensive for accounts of users control by a Sybil attacker to continue being highly active in order to get to the next level as fast as the average user will with the bonus that this mechanism will provide. If the user has greater than medium inactivity, he will receive a low punishment with the intend of keeping the user in its current level but still incentivizing higher activity. Again, we make it hard for the Sybil attacker to keep up all the accounts he controls active.

In the following two cases where the parameter is in the range (0.5 and 1) the same strategy as before is followed in order to make it even harder for Sybil attackers to elevate and maintain the puppet users' credibility and trust values in order for their effect on the system to be of actual concern. Also, the effort that is made with the value choices is to also reduce the effects that this mechanism will have to the rest of the user population that will be able to keep a certain degree of activity.

```
1: After each transaction round for user $U_1$:
2: $current\_rounds(U_1) \leftarrow current\_rounds(U_1) + 1$
3: if $U_1$ is involved in at least 1 running transaction then
4:     $activity\_rounds(U_1) \leftarrow activity\_rounds(U_1) + 1$
5: end if
6: if $TRAE <= current\_rounds(U_1)$ then
7:     $v_{U_1}.inactivity \leftarrow \frac{current\_rounds(U_1) - activity\_rounds(U_1)}{current\_rounds(U_1)}$
8:     for $PARAM \in v_{U_1}.master\_trust, v_{U_1}.worker\_trust, v_{U_1}.credibility$ do
9:         switch $PARAM$ do
10:            case $0.0 < PARAM < 0.25$
11:                if $v_{U_1}.inactivity >= high\_inactivity$ then
12:                    $PARAM \leftarrow PARAM - medium\_punishment$
13:                else
14:                    $PARAM \leftarrow PARAM + medium\_reward$
15:                end if
16:            case $0.25 < PARAM < 0.5$
17:                if $v_{U_1}.inactivity >= medium\_inactivity$ then
18:                    $PARAM \leftarrow PARAM - low\_punishment$
19:                end if
20:                if $v_{U_1}.inactivity < low\_inactivity$ then
21:                    $PARAM \leftarrow PARAM + high\_reward$
22:                end if
23:            case $0.5 < PARAM < 0.75$
24:                if $v_{U_1}.inactivity >= medium\_inactivity$ then
25:                    $PARAM \leftarrow PARAM - medium\_punishment$
26:                end if
27:                if $v_{U_1}.inactivity < medium\_inactivity$ then
28:                    $PARAM \leftarrow PARAM + low\_reward$
29:                end if
30:            case $0.75 < PARAM < 1$
31:                if $v_{U_1}.inactivity >= low\_inactivity$ then
32:                    $PARAM \leftarrow PARAM - high\_punishment$
33:                else
34:                    $PARAM \leftarrow PARAM + low\_reward$
35:                end if
36:    end for
37:    $activity\_rounds(U_1) \leftarrow 0$
38:    $current\_rounds(U_1) \leftarrow 0$
39: end if
```

Algorithm 7.4: Punishment/Reward based on recent inactivity

Chapter 8

Discussion and Conclusions

8.1 General Discussion and Conclusions

We can review our approaches and discuss what was achieved and failed with each of them. The permissioned centralized approach in Section 6.5, provides a degree of safety with the expense of the privilege given to the central authority, which is asking for an identity in order for the user to join, thus increasing the centralization degree of the system, but with the usage of the ledger the system remains transparent so the central authority can't work maliciously without the rest of the network finding out.

The permissionless centralized approach in Section 7.1, is also a simple one and follows the same ideas as the approach in Section 6.5, but for this approach extra mechanisms were added in the attempt to tackle attacks that are inherent to a permissionless system like the change of identity and Sybil attacks. The truth is not much information was found from our study on permissionless systems. [6] uses anonymity but works for a special decentralized approach. The only schemes that were working on a permissionless distributed network were the reputation schemes for VANETs [9], Scientometrics [41] and PoR [49]. All of them, apart from the reputation scheme for VANETs [9], used identities. For the PoR it is explicitly stated in [49] in the enrolment control section: "Our consensus protocol requires an access control layer built into the blockchain nodes, which is known as "permissioned blockchain"." and for Scientometrics we know that scientific papers have authors so that's a form of identity. On the other side the reputation scheme for VANETs works for a very specific type of network and also utilizes RSUs, to achieve reliability, which we can’t have in our case.

8.2 Future Work

For future work the proposed schemes can be implemented and tested with high populations of users and against various attacks and combinations of them to expose potential weaknesses and thus room for improvement.

Another direction that future work can follow is the implementation of the two identity layers and the framework between them for the permissioned centralized approach as described in Section 6.4, so that the benefits of asking for identity provision from user will exist when at the same time anonymity will exist within the network.

Future work can also be done on expanding the bibliographic study that was done in Chapters 4 and 5. Although this study is extensive there is still room for expanding it since reputations schemes are born faster than before and the contribution of keeping track of their characteristics and mapping them to a framework would be great for the scientific community that deals with them.

Also, the framework with which the mapping of schemes is done can also be improved to add more characteristics and thus provide better feedback for every scheme that it is mapped to it.

Finally work on the mechanics of a crowdsourcing platforms can be done to introduce new dimensions with which a reputation system can work (e.g. negotiation for partial solving etc.) which would require new assumptions (just like we did in Section 6.1 and Section 6.2) to be made for the functionality of the system.

Bibliography:

[1] Zhao HY, Li XL. H-Trust: A group trust management system for peer-to-peer desktop grid. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 24(5): 833{843 Sept. 2009

[2] Basit Qureshi, G. Min, D. Kouvatsos. A distributed reputation and trust management scheme for mobile peer-to-peer networks. COMPUTER COMMUNICATIONS 35(5): 608-618 March 2012

[3] Chunqi Tian, Baijian Yang. R2Trust, a reputation and risk based trust management framework for large-scale, fully decentralized overlay networks. FUTURE GENERATION COMPUTER SYSTEMS 27 (2011): 1135-1141

[4] S.D. Kamvar , M.T. Schlosser , H. Garcia-Molina , The eigentrust algorithm for reputation management in P2P networks, in: Proceedings of the Twelfth International Conference on World Wide Web, ACM, 2003, pp. 640–651.

[5] Trung Dong Huynh, N. R. Jennings, N. R. Shadbolt. An integrated trust and reputation model for open multi-agent systems. Auton Agent Multi-Agent Sys (2006) 13: 119–154

[6] Andreea Vişan, F. Pop, V. Cristea. Decentralized Trust Management in Peer-to-Peer Systems. 10th International Symposium on Parallel and Distributed Computing (2011)

[7] Saeed Javanmardi, M. Shojafar, S. Shariatmadari, S. S. Ahrabi. FR TRUST: A Fuzzy Reputation Based Model for Trust Management in Semantic P2P Grids. Int. J. Grid Utility Comput. 6(1): 57–66 (2015)

[8] Liau C.Y., Zhou X., Bressan S., Tan KL. (2003) Efficient Distributed Reputation Scheme for Peer-to-Peer Systems. In: Chung CW., Kim CK., Kim W., Ling TW., Song KH. (eds) Web and Communication Technologies and Internet-Related Social Issues — HSI 2003. HSI 2003. Lecture Notes in Computer Science, vol 2713. Springer, Berlin, Heidelberg

[9] Oluoch, Jared. ”A distributed reputation scheme for situation awareness in Vehicular Ad Hoc Networks (VANETs).” In 2016 IEEE International Multi-Disciplinary Conference on Cognitive Methods in Situation Awareness and Decision Support (CogSIMA), pp. 63-67. IEEE, 2016.

[10] R. Ureña, G. Kou, Y. Dong, F. Chiclana, E. Herrera-Viedma. A review on trust propagation and opinion dynamics in social networks and group decision making frameworks. Information Sciences 478 (2019): 461-475

[11] R. Guha , R. Kumar , P. Raghavan , A. Tomkins , Propagation of trust and distrust, in: Proceedings of the Thirteenth International Conference on World Wide Web, 2004, pp. 403–412 .

[12] Z. Malik , A. Bouguettaya , RATEWeb: reputation assessment for trust establishment among web services, VLDB J. 18 (2009) 885–911 .

[13] F. Hendrikx , K. Bubendorfer , R. Chard , Reputation systems: a survey and taxonomy, J. Parallel Distrib. Comput. 75 (2015) 184–197 .

[14] S.R. Yan , X.L. Zheng , Y. Wang , W.W. Song , W.Y. Zhang , A graph-based comprehensive reputation model: exploiting the social context of opinions to enhance trust in social commerce, Inf. Sci. (NY) 318 (2015) 51–72 .

[15] M. DeGroot , Reaching a consensus, J. Am. Stat. Assoc. 69 (1974) 118–121 .

[16] N.E. Friedkin , E.C. Johnsen , Social Influence Network Theory: A Sociological Examination of Small Group Dynamics, Cambridge University Press, New York, 2014 .

[17] P. Resnick , R. Zeckhauser , M.R. Baye, Trust among strangers in internet transactions: empirical analysis of eBays reputation system, in: The Economics of the Internet and E-

commerce (Advances in Applied Microeconomics Volume 11), Emerald Group Publishing Limited, 2002, pp. 127–157 .

[18] D. Houser , J. Wooders , Reputation in auctions: theory, and evidence from eBay, Econ. Manag. Strat. 15 (2005) 353–369 .

[19] J. Schneider , G. Kortuem , J. Jager , S. Fickas , Z. Segall , Disseminating trust information in wearable communities, Pers. Ubiquitous Comput. 4 (20 0 0) 245–248 .

[20] A. Jøsang , R. Ismail , C. Boyd , A survey of trust and reputation systems for online service provision, Decis. Support Syst. 43 (2007) 618–644 .

[21] S. Javanmardi, Y. Ganjisaffar, C. Lopes and P. Baldi, "User contribution and trust in Wikipedia," 2009 5th International Conference on Collaborative Computing: Networking, Applications and Worksharing, Washington, DC, 2009, pp. 1-6, doi: 10.4108/ICST.COLLABORATECOM2009.8376.

[22] S. A. Paul, L. Hong, Ed. H, Chi, "WHO IS AUTHORITATIVE? UNDERSTANDING REPUTATION MECHANISMS IN QUORA", PROCEEDINGS, CI (2012)

[23] https://www.uber.com/us/en/drive/basics/how-ratings-work/

[24] https://blog.blablacar.com/trust

[25] Zervas, Georgios and Proserpio, Davide and Byers, John, A First Look at Online Reputation on Airbnb, Where Every Stay is Above Average (January 28, 2015). Available at SSRN: https://ssrn.com/abstract=2554500 or http://dx.doi.org/10.2139/ssrn.2554500

[26] Eyal Ert, Aliza Fleischer, Nathan Magen. "Trust and reputation in the sharing economy: The role of personal photos in Airbnb". Tourism Management 55 (2016) 62-73.

[27] A. Abdul-Rahman and S. Hailes. Supporting Trust in Virtual Communities. In Proceedings of the Hawaii International Conference on System Sciences, Maui, Hawaii, 4-7 January 2000.

[28] V. Cahill, B. Shand, E. Gray, et al. Using Trust for Secure Collaboration in Uncertain Environments. Pervasive Computing, 2(3):52–61, July-September 2003

[29] M. Carbone, M. Nielsen, and V. Sassone. A Formal Model for Trust in Dynamic Networks. In Proc. of International Conference on Software Engineering and Formal Methods (SEFM'03), Brisbane, September 2003.

[30] D.W. Manchala. Trust Metrics, Models and Protocols for Electronic Commerce Transactions. In Proceedings of the 18th International Conference on Distributed Computing Systems, 1998.

[31] A. Jøsang. Trust-Based Decision Making for Electronic Transactions. In L. Yngström and T. Svensson, editors, Proceedings of the 4th Nordic Workshop on Secure Computer Systems (NORDSEC'99). Stockholm University, Sweden, 1999.

[32] A. Jøsang. A Logic for Uncertain Probabilities. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, 9(3):279-311, June 2001.

[33] B. Yu and M.P. Singh. An Evidential Model of Distributed Reputation Management. In Proceedings of the First Int. Joint Conference on Autonomous Agents & Multiagent Systems (AAMAS). ACM, July 2002.

[34] J. Sabater and C. Sierra. REGRET: A reputation model for gregarious societies. In Proceedings of the 4th Int. Workshop on Deception, Fraud and Trust in Agent Societies, in the 5th Int. Conference on Autonomous Agents (AGENTS'01), pages 61-69, Montreal, Canada, 2001.

[35] J. Sabater and C. Sierra. Reputation and Social Network Analysis in Multi-Agent Systems. In Proceedings of the First Int. Joint Conference on Autonomous Agents & Multiagent Systems (AAMAS), July 2002.

[36] J. Sabater and C. Sierra. Social ReGreT, a reputation model based on social relations. SIGecom Exchanges, 3.1:44-56, 2002.

[37] L. Page, S. Brin, R. Motwani, and T. Winograd. The PageRank Citation Ranking: Bringing Order to the Web. Technical report, Stanford Digital Library Technologies Project, 1998.

[38] C.-N. Ziegler and G. Lausen. Spreading Activation Models for Trust Propagation. In Proceedings of the IEEE International Conference on e-Technology, e-Commerce, and e-Service (EEE '04), Taipei, March 2004.

[39] R. Levien. Attack Resistant Trust Metrics. PhD thesis, University of California at Berkeley, 2004.

[40] E. Adar and B.A. Huberman. Free Riding on Gnutella. First Monday (Peer-reviewed Journal on the Internet), 5(10):8, October 2000.

[41] W. Hood and C.S. Wilson. The Literature of Bibliometrics, Scientometrics, and Informetrics. Scientometrics, 52(2):291-314, 2001.

[42] Christoforou E., Anta A.F., Georgiou C., Mosteiro M.A., Sánchez A. (2013) Reputation-Based Mechanisms for Evolutionary Master-Worker Computing. In: Baldoni R., Nisse N., van Steen M. (eds) Principles of Distributed Systems. OPODIS 2013. Lecture Notes in Computer Science, vol 8304. Springer, Cham

[43] Amazon's MTurk, https://www.mturk.com

[44] Ted Matherly. "A Panel For Lemons? Positivity bias, reputation systems and data quality on MTurk". Forthcoming, European Journal of Marketing, 12 July 2018.

[45] Fernández, A., Georgiou, C., Lopez, L., Santos, A.: Reliable Internet-based computing in the presence of malicious workers. Parallel Processing Letters 22(1) (2012)

[46] Konwar, K.M., Rajasekaran, S., Shvartsman, M.M.A.A.: Robust network supercomputing with malicious processes. In: Dolev, S. (ed.) DISC 2006. LNCS, vol. 4167, pp. 474–488. Springer, Heidelberg (2006)

[47] Sarmenta, L.: Sabotage-tolerance mechanisms for volunteer computing systems. Future Generation Computer Systems 18(4), 561–572 (2002)

[48] Golle, P., Mironov, I.: Uncheatable distributed computations. In: Naccache, D. (ed.) CT-RSA 2001. LNCS, vol. 2020, pp. 425–440. Springer, Heidelberg (2001)

[49] Gai F, Wang B, Deng W, Peng W. Proof of reputation: a reputation-based consensus protocol for peer-to-peer network. Springer 2018:666–681.

[50] Adamos Ttofari, CrowdBED: A Crowdsourcing system that leverage on Blockchain technology for rEliability and Decentralization, Bachelor Thesis, Dept. of Computer Science, University of Cyprus, 2020

[51] Djellel Eddine Difallah, Michele Catasta, Gianluca Demartini, Panagiotis G. Ipeirotis, and Philippe Cudré-Mauroux. 2015. The Dynamics of Micro-Task Crowdsourcing: The Case of Amazon MTurk. In Proceedings of the 24th International Conference on World Wide Web (WWW ’15). International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 238–247.