OPEN ACCESS Freely available online

PLOS ONE

Quid Pro Quo: A Mechanism for Fair Collaboration in Networked Systems

Agustín Santos[1]*, Antonio Fernández Anta[1], Luis López Fernández[2]

1 Institute IMDEA Networks, Madrid, Spain, **2** FUN-LAB, Universidad Rey Juan Carlos, Madrid, Spain

Abstract

Collaboration may be understood as the execution of coordinated tasks (in the most general sense) by groups of users, who cooperate for achieving a common goal. Collaboration is a fundamental assumption and requirement for the correct operation of many communication systems. The main challenge when creating collaborative systems in a decentralized manner is dealing with the fact that users may behave in selfish ways, trying to obtain the benefits of the tasks but without participating in their execution. In this context, Game Theory has been instrumental to model collaborative systems and the task allocation problem, and to design mechanisms for optimal allocation of tasks. In this paper, we revise the classical assumptions of these models and propose a new approach to this problem. First, we establish a system model based on heterogenous nodes (users, players), and propose a basic distributed mechanism so that, when a new task appears, it is assigned to the most suitable node. The classical technique for compensating a node that executes a task is the use of payments (which in most networks are hard or impossible to implement). Instead, we propose a distributed mechanism for the optimal allocation of tasks without payments. We prove this mechanism to be robust evenevent in the presence of independent selfish or rationally limited players. Additionally, our model is based on very weak assumptions, which makes the proposed mechanisms susceptible to be implemented in networked systems (e.g., the Internet).

Citation: Santos A, Fernández Anta A, López Fernández L (2013) Quid Pro Quo: A Mechanism for Fair Collaboration in Networked Systems. PLoS ONE 8(9): e66575. doi:10.1371/journal.pone.0066575

Editor: Sergio Gómez, Universitat Rovira i Virgili, Spain

Received January 4, 2013; **Accepted** May 7, 2013; **Published** September 5, 2013

Copyright: © 2013 Santos et al. This is an open-access article distributed under the terms of the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

Funding: This research was supported in part by the Comunidad de Madrid grant S-2009/TIC-1692, the Spanish MICINN grant TEC2011-29688-C02, and the National Natural Science Foundation of China grant 61020106002. The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Competing Interests: The authors have declared that no competing interests exist.

* E-mail: agustin.santos@imdea.org

Introduction

Selfish behavior is becoming a subject of great concern and practical importance to network designers [1]. Game Theory is the approach of preference to face the design of communication systems with (potentially) selfish entities. This has lead to the proposal of a number of interesting protocols and mechanisms for networks based on Game Theory concepts [2,3]. However, in the study of networks under conventional models, a collection of simplifying assumptions are typically made. For instance, it is assumed that selfish users are rational, that they are homogeneous, that they can compute a Nash equilibrium, that their utility function is known, etc. However, there are many systems in which these assumptions are not very realistic.

In this paper we revisit the study of communication systems with selfish users (or players), reevaluating and relaxing the above-mentioned common assumptions. In particular, we propose the problem of analyzing and designing a fair collaborative system under a very weak set of game theoretic assumptions. In this general context, we propose mechanisms to be used to implement this collaborative system with provable properties, like the fairness of the system and the truthfulness of its users. The mechanisms proposed can be applied to such varied technologies as social and crowd computing, Web 2.0, P2P, opportunistic networks, and cloud systems.

As mentioned, we abstract the problem to be solved as the fair execution of tasks in a decentralized collaborative system. The main challenge when creating collaborative systems in a decentralized manner is dealing with the fact that system nodes may behave in selfish ways, trying to obtain the benefits of the tasks but without participating in their execution. (This is the realm of Game Theory, which has been instrumental to model collaborative systems and the task allocation problem, and to design mechanisms for optimal allocation of tasks.) We assume that all nodes have an interest on having the tasks done. However, establishing fair mechanisms for sharing the generated work-load is not immediate. (E.g., in current P2P systems, usually a low fraction of peers assume most of the required effort, and this causes reduced performance, lack of reliability, low incentive to participate for fair users, etc.) It would, therefore, be desirable that each node could take the responsibility of the execution of a balanced fraction of the tasks.

The objective is to establish some kind of protocol to share the task execution costs. For this, we need to consider the concept of ability or *opportunity of execution*. Let us assume that each node has some capacity for timely execution of a given task. This capacity may vary over time and with the type of task. For example, at a given time, a node may have free bandwidth but have full utilization of its CPU, while theits situation could be the opposite at another time. Hence, at a particular moment, a node may have greater ability to perform tasks involving communication, while at a later time itits situation may change to prefer tasks that are more intensive in CPU computation.

This opportunity or ability is related with the notion of task execution cost. In other words, we define the cost as some kind of metric measuring the capability of executing a particular task at a given time. Hence, the cost varies from one task to another (even when the task is the same, but at a later time). In Game Theory, closely related to cost, there is the notion of utility. We define the utility as the cost savings associated with a work not done. Hence, given that all nodes are interested in the execution of the tasks, a node gets more utility whenever it avoids running tasks, by letting other nodes running them.

Clearly, when trying to formalize a model based on these notions, a number of problems arise. First, node's costs are only known by the node itself. For external entities it would be difficult to audit or check if a given particular node has more or less CPU capacity. In Game Theory, this concept is called *private information*. To obtainFor obtaining the private information of a node, the basic mechanism is to directly ask for it and expect the node to declare its value correctly.

For us, each node is a computing elementnode that belongs to a user who can alter her node's behavior for her own benefit (i.e., may declare false costs trying to avoid the execution of tasks). Whenever this happens, we claim that the user acts in a selfish way. This selfishness is one of the factors that may distort the internal workings of a distributed application. The loss of system performance produced by selfish nodes is a parameter to consider and it is called price of anarchy [4,5].

Therefore, the problem we face consists of designing a system capable of assigning tasks to nodes so that all the tasks are executed, and the total cost incurred is minimal. When the behavior of nodes is guaranteed to be fair, this is just a simple optimization exercise. However, when nodes may choose whether to be selfish the problem becomes much more complex. In this paper we propose an algorithm that, based on Game Theorybasing on game theory principles, solves this problem. We have called this algorithm *Quid Pro Quo Mechanism* (QPQ). The name comes from a Latin expression commonly used by lawyers and which may be translated as *"This for that"* or *"A thing for another"*. This expression is often used when someone does a job and waits for anand equivalent compensation in exchange. We used this expression since it reflects the spirit of the algorithm: due to the lack of payments in our model, the nodes work for others with the hope that others will work for them in the future.

1.1 State of the Art

As described above, the problem addressed in this paper is the allocation of task executions to potentially selfish users. This problem has been extensively studied in the literature. One important related work was carried out by Rosenschein et al. [6], where they define a "Task Oriented Domain". Even though they obtain fairly relevant conclusions, they do not shed any light on the specific problem considered here, since their model makes strong assumptions, such as knowledge of the task costs or a bargaining power over time. Recently, the use of Game Theory to model selfish behavior in the design of distributed systems has been proposed. Some works have appeared using mechanism design, a branch of mathematics derived from Game Theory, which provides the required background for the study and design of distributed systems under the action of selfish nodes (see, e.g., [7–9]).

Our work falls naturally into the large area of mechanism design without money (see, e.g., [10]). In this direction, our QPQ algorithm is similar to the mechanism proposed by Jackson and Sonnenscheinet al. [11,12]. In that work, they present a new interesting type of mechanism (called *linking mechanism*) which, instead of offering incentives or payments to players, limits the spectrum of players' responses to a probability distribution known by the game designer. In that paper, the authors proved that a linking mechanism is valid when the players' possible decisions are distributed following discrete probabilities. Additionally, the authors show that a linking mechanism can also be used for repeated games. Even though the work of Jackson et al. is very relevant to the problem we consider, it does not offer a method for the construction of mechanisms when the game is based on unknown continuous probability distributions, as assumed here. A second work that explores the idea of linking mechanism is due to Ferenc [13]. In that paper, he proposes a mechanism which limits player responses by restricting the first two moments (mean and variance) of the probability distribution, being that distribution known to the designer. Both works reflect the main idea behind the concept of linking mechanism: when a game consists of multiple instances of the same basic decision problem (e.g., saying yes or no, choosing among a number of discrete options), it is possible to define selfishness-resistant algorithms by restricting the players' responses to a given distribution. Hence, in that case, the frequency with which a player declares a particular decision is known beforehand.

In the specific areas of computing and communications, it is important to remark that most mechanisms proposed for dealing with selfish agents make unrealistic assumptions [14]. In this direction, Bauer et al. [15] criticize many of these hypotheses, reviewing well-known works [16–18] to show that they are not applicable in real environments. Specifically, they identify two common strong artificial assumptions:

1. The assumption that the designer of the algorithms has some knowledge about the preferences of the nodes.
2. The assumption that the interaction among players is limited to a single round (while it is well known in the literature that a solution for a single round does not necessarily apply when the game is repeated).

1.2 Results

In this paper, we face the problem of task allocation relaxing these (and other) common hypotheses, so that the obtained results can be applied in real environments. Hence, the contributions of this paper are twofold. First, to the best of our knowledge, this is the first work proposing a linking mechanism solution without prior knowledge of the distribution of the players' decisions, and without a payment system among them. Second, we generalize and improve previous works in the area to provide algorithms which are susceptible of being applied in the context of repeated task execution allocation in real communication and computing systems, even in the presence of selfish or non-rational users.

As we previously claimed, we do not want to restrict our mechanism to a set of unrealistic hypotheses. Instead, we establish a number of requirements that our model must satisfy. These requirements should provide the appropriate flexibility to guarantee the applicability of our results in real environments.

- *Abstract utility metrics:* We assume, as an abstract notion, that the cost of executing a task for a node (user, player, and node will be used indistinctly in the rest of the paper) depends on its interest on the task, its opportunity or ability to execute it, or its degree of willingness to cooperate. We need to accept that each node may measure this parameter in its very own metric and units. Hence, for example, a node may decide on the cost of a task according to the occupation of its CPU, but another

one may prefer to make it depend on its available bandwidth. In a real scenario, the number of factors that can influence the execution cost of a task can be extremely large. In this direction, ourout model must enable each node to define, in a flexible way, how costs (and utilities) are measured.

- *No payment system:* Payments are, in its most basic interpretation, a way of exchanging costs. Many existing mechanisms base their incentive schemes on the existence of payments. For payments to be possible, it is necessary that all players manage a common currency reference (euro, dollar, etc.). However, given our previous requirement, it is not clear how we can find that shared currency reference in our model. If a node measures its costs in terms of, for example, reputation, it can hardly "pay" to another node that measures its costs on CPU units. Hence, in our work, we assume that payments are not possible.
- *Players' rationality:* In Game Theory, most of the existing algorithms require players to be perfectly rational. This means that a player, using the available information, should always be capable of selecting the best strategy (the one that maximizes her utility). However, this is a controversial hypothesis which is suffering much criticism. Accepting this assumption means that players are capable of mathematically calculating all alternatives, which in some cases requires solving complex (NP-hard) problems. Clearly, this is not always feasible for all players. Hence, we commit ourselves to proposing mechanisms suitable for finding quasi-optimal task allocation, even in the presence of rationally-limited players.
- *Incentive to participate:* In relation to players rationality, even in the case in which we are able to find global quasi-optimal task allocation, it is possible that the behavior of rationally-limited users may harm the benefit of other players. In this direction, we add a stronger requirement. We force to ensure an incentive to participate in the game to all nodes, independently of whether they are rational or not.
- *No central entity:* A final requirement we impose is the capability of the system to work without the existence of any kind of central entity. This means that the proposed mechanisms must be susceptible of being implemented following completely distributed schemes.

1.3 Structure

The rest of the paper is structured as follows. First we provide a formal definition of the problem and define basic terminology. Then, we present a basic linking mechanism, and evaluate the issues that need to be faced to make it suitable for our problem. Next, we present the QPQ mechanism, and formally prove its properties. Finally, we describe how QPQ could be used in real environments and present some conclusions.

Analysis

2.1 Definitions

To establish a formal framework for the problem, let us provide some definitions.

Definition 1 (Problem). The problem of the assignment of tasks is a tuple $\langle T,N,C\rangle$ where:

1. $T=\{t_1,t_2,\ldots\}$ is the (not necessarily finite) set of tasks that are issued to the system over time (t_k is the task issued at time step k). We assume tasks to be atomic, independent, and of fixed duration. (For simplicity, we will assume the durationit equals i.e, each task takes one time step to be executed.)
2. $N=\{1,2,\ldots,n\}$ is an ordered list of nodes or players, where N is assumed to be finite,
3. $(C(t))_{i\in N}$ is a vector of costs (or utilities) where $C_i(t)$ is the cost of executing task $t\in T$ by node i. This information is private (only known by node i).

It is important to remark some aspects of the above definitions. First, we assume that the set of tasks is not known beforehand. Tasks appear one by one in a sequence of time steps, which drives our discrete time evolution. Hence, the arrival of a new task dictates the start of new a round of our repeated game. We assume that tasks are independent among them and that the execution of a task does not influence the cost of the subsequent ones. Moreover, we force that one task must be completely executed by the time the next task is issued. For simplicity, we assume that the mechanisms take negligible time to coordinate the allocation of the tasksto coordinate the allocation of the tasks take negligible time (with respect to the time step). Finally, we assume that every node that is assigned a task by the allocation mechanism actually executes the task.

Hence, as tasks are issued, each node $i\in N$ estimates a sequence of costs $C_i(t_1),C_i(t_2),\ldots,C_i(t_k),\ldots$, which we assume as independent samples of a probability distribution $\sigma_i\in\Delta(S_i)$ characterizing node i's behavior. In this context, we denote S_i as the distribution support (i.e., the range of values for which the probability is different than zero) and $\Delta(S_i)$ as the set of all possible probability distributions over S_i. From now on, we will consider that C_i is a real-valued random variable. To simplify the notation, we define realizations of this random variable as $c_i(t)=C_i(t)$, $t\in T$. When clear from the context, we may remove the task t from the notation $c_i(t)$, as c_i.

Given that all players enjoy the result of any task executed in the system, we can define the utility of a player as the savings obtained by not executing some tasks (i.e. the benefit obtained from participating in the cooperative computing scheme and not making all the work by itself). That is, the utility $u_i(t)$ of node i corresponding to a given task t is given by

$$u_i(t)=\begin{cases}0 & \text{if node } i \text{ executes the task,}\\ c_i(t) & \text{otherwise,}\end{cases} \quad (1)$$

and the total utility of node i is $u_i=\sum_{t\in T}u_i(t)$. Note that, in Game Theory it is common to add a discount factor (δ) in time. We have assumed it to be equal to $\delta=1$.

We define U_i as the random variable associated to the total utility of node i. In a similar way, we denote by W_i the real-valued random variable associated to the actual player i's executed cost and by $w_i(t)$ its concrete realization for task t. Note that each task is either executed or not by a particular player. Hence,

$$E[C_i]=E[U_i+W_i]=E[U_i]+E[W_i]. \quad (2)$$

Finally, we assume that communication between players is *reliable* and *concurrent*. In particular, in the mechanisms we propose all players exchange their values $c_i(t)$. We assume that these values are correctly received by the players in a time that is negligible with respect to the time step (hence the reliability property). Additionally, we assume that each player sends its value before receiving the value of any of the other players (hence the concurrency property).

2.2 Basic Linking Mechanism

As mentioned above, a linking mechanism is applicable to repeated games where the decision (also knownknow as message) of players is restricted to a particular known set. In our problem, the decision is the cost $c_i(t)$ of the task. With this concept in mind, let us define our first algorithmic attempt to solve the problem by applying a linking mechanism, presented in Algorithm at table 1.

As it can be observed, for each task, each player estimates the cost of computing the task and publishes it. Publication means broadcasting a message with the cost to all players (although any other means of distribution, like shared memory, can be used). By assumption, a player send its costs before it receives any of the others (concurrency, which implies that coststhey do not depend on each other), and all of the costs are correctly received at each player (reliability). Then, the algorithm assigns the task to the player that publishes the lowest cost. If players publish their real costs, this will produce that the total utility is maximized. However, this kind of approach could drive selfish users to publish fake costs in order to avoid executing tasks. For this reason, we add an acceptance test. When a published cost is not considered acceptable, then the system generates a random value for the cost of that node on the round. The implementation of this acceptance test will be discussed later, however it is important to remark that it contains the linking part of the mechanism (it depends on the historical values published by that particular node). Just as an example, we can imagine that if we force that nodes must publish costs between 0 and 1 following a uniform distribution, then we could consider unacceptable values deviating from that distribution. It is also important to note that all nodes use the same acceptance test with the same history. Then, they all accept or reject. Then, if players reject a value c_j, the value $Random(c_{-j})$ generated is in fact a value deterministically generated from the set of values $c_{-j} = \bigcup_{k \neq j}\{c_k\}$, so that all players re-generate the same value for j.

Algorithm at table 1 has the objective of providing intuition on how we build our mechanism, but it clearly has several issues that contradict our previously stated requirements. In particular, fair allocation is not guaranteed. For instance, there is not a way of defining a notion of fairness within this algorithm, given that costs may have different meanings for different players. Additionally, given that costs are abstract notions, we cannot have any a-priori information on the shape of their corresponding distributions. So, it is not clear how to implement the acceptance test.

Table 1. Simple linking mechanism (code for node i, and a generic task t, omitted).

1: Estimate and publish the cost c_i of the task
2: Wait to receive the costs c_j from the other players
3: **for all** $j \in N$ **do**
4: **if** not $Accepted(c_j, Historic_j)$ **then**
5: $c_j \leftarrow Random(c_{-j})$
6: **end if**
7: $Historic_j \leftarrow Historic_j \cup \{c_j\}$
8: **end for**
9: $d \leftarrow \underset{j \in N}{\operatorname{argmin}}\ c_j$
10: **if** $d = i$ **then**
11: execute the task
12: **else**
13: do nothing (node d will execute the task)
14: **end if**

doi:10.1371/journal.pone.0066575.t001

Digging into these problems, it is easy to understand that one of their causes is the fact that, given our requirements, each player has the right of measuring her costs on her preferred metric. (Hence, each player may have different distributions with different support.) For this reason, cost comparisons cannot be easily made. Additionally, there is a second aspect that must be addressed. In the literature about linking mechanisms, authors assume that instances of the game (rounds) are simultaneous in time. In this case, defining the acceptance function over the set of values is easier. However, in our case, tasks are issued (,and hence players generate their costs), over time. Then, from the point of view of the designer, it is not clear how to determine the acceptance of a value by comparing with a certain probability distribution. The issue is even worse given the fact that this distribution is not known by the designer. To solve all these problems we propose a novel solution based on applying a transformation over the utility function.

2.2.1 Utility normalization. Given that the utility is defined as the work not done by a node, we may use as utility function of a node its probability distribution of costs. Once this is done, we may modify Algorithm at table 1 and normalize players' utilities so that they may be compared among each other. To normalize we use a transformation called *Probability Integral Transformation* (PIT). Our idea is to use the known fact that any cumulative probability distribution function has in itself a uniform distribution [19]. More formally, the PIT is defined as follows

Definition 2 (Probability Integral Transformation) Let X be a continuous random variable with a Cumulative Distribution Function (CDF) F; that is $X \sim F$. Then, *the Probability Integral Transformation (PIT)* defines a new random variable Y as: $Y = F(X)$.

As mentioned above, our interest in the PIT is due to the following lemma.

Lemma 1 (PIT follows uniform distribution) *Let X be a continuous random variable with CDF F, then F follows a uniform distribution on interval* [0,1]. *That is, the random variable Y defined by the probability integral transformation $Y = F(X)$ is a normalized uniform distribution.*

Moreover,Note that X does not need to be a continuous random variable. In the case that the player's costs follow a discrete distribution, it is still possible to perform a similar transformation called *Generalized Distributional Transform* [20], whose properties are equivalent to those of the PIT.

Definition 3 (Generalized Distributional Transform) Let X be a random variable (not necessarily continuous) with a cumulative distribution functionprobability F and let $V \sim U(0,1)$ be a random variable with uniform distribution in [0,1] independent of X. *The modified distribution function $F(x,\lambda)$ is defined as*

$$F(x,\lambda) = Pr(X < x) + \lambda Pr(X = x).$$

From this, we can define the general distributional transform of X as $Y = F(X,V)$, which can be proved to be a uniform distribution on the unit interval.

Proofs of these properties can be found in [20]. Many studies in economics use this definition and its properties, such as [21] or [22]. In our case, to simplify the notation, we just call PIT to both transformations independently of whether the base distribution is continuous or discrete.

Coming back to Algorithm at table 1, our idea is to modify it by applying the PIT on the players' declared costs. Hence, instead of

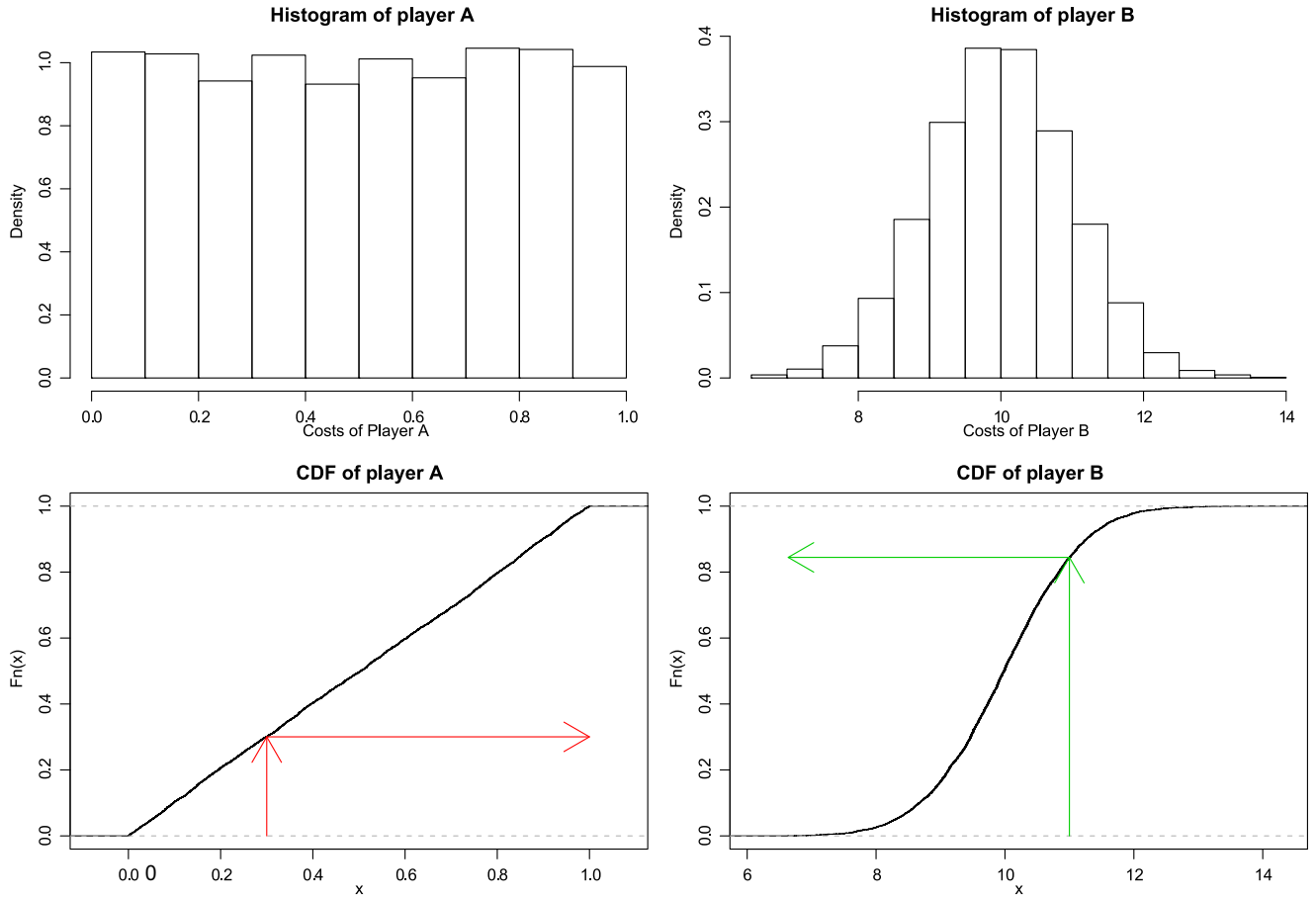

Figure 1. Two different players. At the top, we can see the execution task cost histograms of two different players. Note that they follow different probability distributions. At the bottom, we depict the Cumulative Distribution Function (CDF) for both. As it can be noticed through the depicted arrows, the fact that player A has a smallerminor cost than player B (0.3 versus 11) does not mean that player A will be assigned the task. Instead, when applying the PIT, player B is the one publishing the lowest normalized cost.
doi:10.1371/journal.pone.0066575.g001

publishing the values from herits real probability distribution, a player must publish the normalized ones, so that the new algorithm chooses for running the tasks the player minimizing the normalized cost values instead of the original costs. Fig. 1 illustrates this process.

Based on these arguments, it is clear that the PIT provides a mechanism for comparing (normalized) node costs. However, we may wonder if the proposed transformation is valid, in the sense that it may not preserve the preferences of the player. To solve this issue, it suffices to notice that, what we are doing is changing the space of preferences. Therefore, the PIT somehow means that, instead of asking the user "How much does it cost to execute the task?", we inquire for something like "What percentage of tasks do you prefer to this one?" At the end of the day, and for our objectives, these questions are requesting the same information, but the latter is normalized in the interval [0,1], which is a great advantage.

Although from an analytic point of view we assume that players can compute the PIT perfectly, in a practical set up players do not need to consider any a priori distribution of probability. They can simply generate costs using their particular distribution and apply the PIT using the successive generated samples. This process uses what in statistics is know as the Empirical Cumulative Distribution Function (ECDF). We will review this concept later, when we analyze the practical formulation of QPQ.

2.2.2 Acceptance test. Once we know the properties of the PIT, it is clear how we can implement the acceptance test for the linking mechanism. The idea is that any player applying correctly the PIT on her real cost distribution, must generate a uniform distribution on the unit interval on her published normalized cost values. Hence, from the point of view of the mechanism designer, the problem consists inon determining whether these published values follow or not that uniform distribution. There are a wide range of tests that allow checking that. These tests are called *Goodness-of-Fit* (GoF) tests.

Continuing with this argument, we propose to implement the acceptance test of our algorithm by using some GoF test on the declared transformed sequence of costs published by the player. Whenever a player is honest and she declares the values by applying the PIT transformation on her own distribution, these values will be uniformly distributed in the unit interval. In that case (with high probability) the GoF tests will accept the samples. More important, this process has an error which tends to zero when the number of samples (rounds) increases for any reasonable value of the threshold. For the study of our analytic results, we assume that GoF tests are perfect and this error is zero. (We will review this concept again in our practical implementation of QPQ.)

2.2.3 Punishment. In the case that a dishonest player tries to avoid the execution of tasks, one possible strategy is to generate

increasing cost values, so that the PIT transformed values are close to the unit. However, this type of behavior is quickly detected by the test. An open question is how to establish a punishment to this and any other player whose GoF test comes out negative. One possibility is to force the node to execute the task. Unfortunately, this policy would force fair players to execute tasks in cases of false negatives.

Another possibility, inspired byon previous works on linking mechanisms, is to reject the value declared by the player and generate a new random value according to the normalized uniform distribution. Additionally, we require that no central entity existsexist on the system. For these reasons, we propose to use a deterministic (repeatable) random generator that any of the remaining nodes can use to calculate the new value. (We deal with the practical aspects of this approach below.) At a first sight, this strategy may seem as a very poor punishment, given that there is always a chance that a player emerges victorious of a lie. However, later in this paper we will prove that this is not only enough to discourage dishonest players, but also a crucial ingredient to guarantee that our mechanism is strategy-proof.

2.3 The Quid Pro Quo Mechanism

After describing the different ingredients of our solution, we are able to propose the final algorithm, which we call the *Quid Pro Quo* (QPQ) mechanism. The details can be observed in Algorithm at table 2.

Note that we use $\bar{c}_i$ to denote the PIT-normalized cost to bethe published, while c_i is the actual cost. We also hold in $\bar{c}_i$ the pseudorandom value that replaces the value published by i when it does not pass the acceptance test. (Hopefully context will allow disambiguation.) It is important to notice that the algorithm is the same for all participants, and that it is based on information known by all of them. Therefore, no central entity is required. When a task is issued, each node can estimate its own cost and publish its PIT-normalized value. This value is then received by all other players. When a player has all the values, she checks whether any player published a dishonest value by applying the GoF test. If the value does not pass the test, it is regenerated as described above, by using a pseudorandom generator (that allows all players to generate the same value) of uniformly distributed values in [0,1]. With these reviewed values, the player proceeds to determine if herits own value is the minimum, in which case it executes the task, publishing the results to the rest of the nodes if necessary.

Table 2. Quid Pro Quo mechanism (code for node i, and a generic task t, omitted).

1: Estimate the cost c_i of the task
2: Publish the normalized cost $\bar{c}_i = PIT(c_i)$
3: Wait to receive the normalized costs $\bar{c}_j$ from the other players
4: **For all** $j \in N$ **do**
5: **if** not $GoF\_Test(\bar{c}_j, Historic_j)$ **then**
6: $\bar{c}_j \leftarrow Random(\bar{c}_{-j})$
7: **end if**
8: $Historic_j \leftarrow Historic_j \cup \{\bar{c}_j\}$
9: **end for**
10: Let $d = \underset{j \in N}{\operatorname{argmin}}\ \bar{c}_j$
11: **if** $d = i$ **then**
12: execute the task
13: **else**
14: do nothing (node d will execute the task)
15: **end if**

doi:10.1371/journal.pone.0066575.t002

In the following sections, we formally study the expected harm (or reduction of benefit) that dishonest behavior causes on QPQ. Intuition says that the loss due to a dishonest player should be comparable to having that player executing tasks at random. Indeed, we show below that, independently of their behavior, nodes may never expect a profit of less than the one obtained through a mechanism in which tasks are randomly assigned. This property is very useful in case the node is not capable of accurately evaluating theits costs (it is non-rational).

Another important aspect is that QPQ guarantees a minimum benefit to the entire system, even if one or more players are non-rational or rationally limited. In this sense, we will show that the best strategy for any player is to act as if the rest of the players were rational and fair. That is, incorrect behaviorbehaviors of some players does not alter the strategy of correct players. In the next section, we prove all these claims in a formal way.

2.4 Formal Analysis of QPQ

Our QPQ algorithm is strongly inspired by the work of Jackson and Sonnenscheinet al. [11,12]. Hence, some of our proofs have been adapted from the ones provided there. We review now the most relevant properties of the QPQ mechanism presented in Algorithm at table 2. Assuming that the number of rounds (tasks) is large enough, and that the players' costs are independent to each other, we prove the following properties.

1. QPQ is optimal in the sense that it minimizes the total work done when all players are honest.
2. For any player, the rest of the players can be seen as a single aggregated player. For each task, the aggregated player's cost is the smallest of its members'. These costs follow a Beta distribution.
3. The best strategy of a player is independent of the behavior of the rest.
4. The strategy that optimizes the utility of a player is being honest. In Game Theorygame theory terminology, this means that QPQ is strategy-proof.
5. Each player always obtains a positive expected utility, which is determined by the number of players.
6. An irrational or rationally-limited player always obtains a positive profit.
7. The system is fair in the allocation of tasks and in normalized effort. That is, all the players will run the same number of tasks and perform a similar normalized effort (in expectation).
8. When the number of player is high enough, QPQ ensures very attractive performance.

To address the mathematical analysis of the algorithm we will assume that the PIT and GoF steps are perfect. In fact, with a large number of samples, these processes have errors close to zero. Another aspect that will simplify our analysis, is the idea of *aggregated player*. We evaluate the performance of a node playing against a "fictitious" node that aggregates the responses of all the other nodes. This aggregated player behaves by publishing at each round the minimum of all the normalized costs of the players in the aggregation. This approach is compatible with all the assumptions of the model and is helpful because it significantly simplifies the analysis.

To make our notation clear,clearer, given a task, we use x to denote the true normalized cost of player i for that task, while X (or X_i) is the random variable for that value. When executing QPQ, players may publish the true value x or a$x=\bar{c}_i$ or another false value. When convenient,In that case, we use z to denote that dishonest value $\hat{c}_i$ and also, overloading the notation, the re-generated random value replacing it when the GoF test fails. We assume that the z values are realizations of some random variable Z. Given a task with cost c_i, the player obtains a normalized utility $\bar{u}_i=\bar{c}_i$ when she does not execute the task (independently on what she published) and makes a normalized work of $\bar{w}_i=\bar{c}_i$ when she executes the task (where $\bar{W}_i$ denotes the random variable). Additionally, we use y to denote the value $\min \bar{c}_{-i}$ published by an aggregated player. Following mechanism design notation, we say that the (social) decision function d of QPQ is

$$d=\underset{i\in N}{\operatorname{argmin}}\ \bar{c}_i.$$

Then, we define $Pr[d=i]$ as the probability that player i declares the minimum value and executes the task. When working with the aggregated player, Y is a vector of random variables, and we use $Pr[Y\leq y]$ to denote the probability that at least one element of Y, say j, validates $Y_j\leq y$.

With this notation in mind, we can prove that, for any player i, the expectation of the declared costs is equal to the expected utility plus the expected work. Additionally, this quantity is a constant. I.e.,

$$E[\bar{C}_i]=E[\bar{U}_i+\bar{W}_i]=E[\bar{U}_i]+E[\bar{W}_i]=\int_0^1 \bar{c}_i d\,\bar{c}_i=\frac{1}{2}. \quad (3)$$

This means that a player maximizes her utility when she minimizes her work, and vice-versa. In the following propositions, we will use this fact.

2.4.1 Players' normalized costs distributions. We argue here that all players' normalized costs follow an independent uniform distribution on [0,1]. When players are honest, their reportedreport values follow a uniform distribution on [0,1]. This follows from the properties of the PIT transformation introduced above. On the other hand, when a player is dishonest, it may change the distribution of its normalized costs trying to obtain extra benefit. However, we assume that in this case the GoF test fails. Then, her attempt will be detected, and the value will be replaced by a pseudorandom value drawn from an independent uniform distribution on [0,1]. A final case is that the dishonest player may generate fake normalized costscost that follow a uniform distribution on [0,1], hence passing the GoF test. In this case the normalized cost $\bar{c}_i(t)$ for a task t is independent from the values $\bar{c}_j(t)$ of the other players since, from concurrency, the value has to be sent before the others are received. Hence, the following result.

Proposition 2 *The set of final normalized costs $\bar{c}_j$ considered in Line 10 of Algorithm at table 2 are drawn from independent and identical distributed (iid) random variables, with uniform distributions on* [0,1].

2.4.2 Optimality. The QPQ algorithm is optimal in the sense that, if all players are honest, it minimizes the total normalized work done, as shown in the following proposition.

Proposition 3 *Assume that all players are honest. For a given set T of tasks, there is no mechanism M such that $E[\sum_{i=1}^{n}\bar{W}_i^M]<E[\sum_{i=1}^{n}\bar{W}_i]$, where $\bar{W}_i^M$ is the random variable associated with the normalized work done by player i when using mechanism M.*

Proof. The proof is straightforward using contradiction. Assuming that such mechanism M existsexist, there must be, at least, one task for which $\bar{w}^M<\bar{w}$, however, the social decision function of QPQ always selects the player publishing the minimum of the normalized costs, so it is not possible that M is able to select another player capable of executing with less cost. So, we conclude that M cannot exist.

2.4.3 Aggregated player. It is assumed that players' normalized costs have independent uniform distributions on [0,1]. Hence, the probability density function of each player i is $f_i(\bar{c}_i)=1$ on that interval. Thus, the costs of an aggregate player for $n-1$ nodes follow a probability distribution $Beta(y;1,n-1)$ as shown.

Proposition 4 *The costs Y of the aggregated player of $n-1$ i.i.d. players (with uniform distribution on* [0,1]*) follow a $Beta(y;1,n-1)$ distribution, with probability density probability function $f(y)=(n-1)(1-y)^{(n-2)}$ and CDF $F[y]=Pr[Y\leq y]=1-(1-y)^{n-1}$.*

Proof. Recall that the cost of an aggregated player is the minimum of the normalized costs of the players in the aggregation. The CDF $F(\cdot)$ of that cost can be obtained as follows. Let us assume that the players in the aggregation are 1 to $n-1$.

$$F[y]=Pr[Y\leq y]=1-Pr[Y>y] \quad (4)$$

$$=1-\prod_{j=1}^{n-1} Pr[Y_j>y] \quad (5)$$

$$=1-\left(\int_y^1 1d\varphi\right)^{n-1} \quad (6)$$

$$=1-(1-y)^{n-1} \quad (7)$$

Where Y_j is the random variable associated with the normalized cost of node j. Hence, the probability densitydensity probability distribution is

$$f(y)=(n-1)(1-y)^{(n-2)}.$$

The *Beta* distribution is defined as follows [23].

$$Beta(y;1,n-1)=\frac{1}{B(1,n-1)}y^{1-1}(1-y)^{(n-1)-1} \quad (8)$$

$$=\frac{(1+(n-1)-1)!}{((n-1)-1)!}(1-y)^{n-2}, \quad (9)$$

where $B(\cdot,\cdot)$ is the Beta function. Now, it is easy to check that $f(y)=Beta(y;1,n-1)$.

2.4.4 Players' strategies. Every rational player knows that the rest of players follow uniform and independent distributions. The question a selfish rational player makes is which is the best strategy for obtaining the greatest possible benefit. If a player uses a distribution other than the uniform, her values will be rejected by the GoF, and will be re-generated from a uniform distribution. However, a player could lie following a uniform distribution that is not independent of her actual values. Note that QPQ does not know about true normalized costs (they are private) and uses for

the assignment decision the declared value or the random value assigned by the system if a lie is detected. In both cases, the aggregated player observes a random variable Z that must follow a uniform distribution. We show now that either case drives the player to worse results than her own honest distribution, so that the player will notthen that player will no have any incentive to cheat.

Let us first quantify the expected work done by honest players.

Proposition 5 *The expected normalized work $E[\bar{W}_i]$ done by an honest player i is $\frac{1}{n+n^2}$.*

Proof. Recall that we assume that player i is in the system with an aggregated player of $n-1$ nodes. Then, $Pr[d=i]$ is the probability that player i publishes a normalized cost smaller than the one of the aggregated player.

$$E[\bar{W}_i] = \int_0^1 xPr[d=i]dx \tag{10}$$

$$= \int_0^1 x \int_x^1 (n-1)(1-y)^{(n-2)} dydx \tag{11}$$

$$= \frac{1}{n+n^2}. \tag{12}$$

Notice that we use the probability distribution of the aggregated player derived in Proposition 4.

However, publishing dishonest values independent ofon the real distribution drives to making an amount of normalized work of $\frac{1}{2n}$. The following proposition proves this statement.

Proposition 6 *When a player i publishes dishonest non uniform values or values independent of her true normalized uniform distribution, it performs in expectation $E[\hat{W}_i] = \frac{1}{2n}$ work.*

Proof. The values z used to decide whether to assign a task to player i follow a uniform distribution that is independent of the actual costs for i. Hence,

$$E[\hat{W}_i] = \int_0^1 xPr[d=i]dx \tag{13}$$

$$= \int_0^1 x \int_0^1 \int_z^1 (n-1)(1-y)^{(n-2)} dydzdx \tag{14}$$

$$= \frac{1}{2n} \tag{15}$$

From this result, Proposition 5, and Eq. 3, we directly derive the following corollary.

Corollary 7 *Assume a player i publishes dishonest non uniform values or values independent of her true normalized uniform distribution. Given that $n \geq 2$, it holds that $E[\bar{W}_i] = \frac{1}{n+n^2} < \frac{1}{2n} = E[\hat{W}_i]$. Hence, since the sum of the expected work and expected utility is $\frac{1}{2}$, players obtain higher expected utilities by being honest than by publishing dishonest normalized costs.*

In essence, the previous result shows that, unless the player cheats declaring values z that follow a uniform distribution and depend on the true values x, the system will randomly assign tasks to that player. Hence, the player is better off being honest in that case. However, it is still pending to study the case in which it follows a more general dishonest strategy, modeled by a bi-variate probability density function $f_{x,z}(x,z)$ that relates both values x and z. As mentioned, the marginal distribution for z must be uniform. The first step is to show that the expected work assigned to the aggregate player does not depend on this function.

Proposition 8 *The total expected normalized work assigned to an aggregate player j (aggregating $n-1$ nodes), with costs $x = \bar{c}_j$, does not change when a player i (not in the aggregation) declares dishonest values $z = \hat{c}_i$.*

Proof. Given that $f_{x,z}(x,z)$ has uniform marginal distribution for z, we have that,

$$f_z(z) = \int_0^1 f_{x,z}(x,z)dx = 1. \tag{16}$$

Hence, the expected work assigned to by j is

$$E[\hat{W}_j] = \int_0^1 yPr[d=j]dy \tag{17}$$

$$= \int_0^1 y(n-1)(1-y)^{(n-2)} \int_0^1 \int_y^1 f_{x,z}(x,z)dzdxdy \tag{18}$$

$$= \int_0^1 y(n-1)(1-y)^{(n-2)} \int_y^1 \int_0^1 f_{x,z}(x,z)dxdzdy, \tag{19}$$

where $E[\hat{W}_j]$ is the expected work assigned todone by the aggregated player j when player i lies. But, as we have uniform marginals, from Eq(16) the above expression becomes

$$E[\hat{W}_j] = \int_0^1 y(n-1)(1-y)^{(n-2)} \int_y^1 1dzdy \tag{20}$$

$$= \int_0^1 y(n-1)(1-y)^{(n-2)}(1-y)dy \tag{21}$$

$$= \int_0^1 y(n-1)(1-y)^{(n-1)}dy \tag{22}$$

$$= \frac{n-1}{n+n^2}. \tag{23}$$

Which is equal to the total work assigned todone by the aggregated player j when i is honest, that can be computed as follows.

$$\int_0^1 y(n-1)(1-y)^{(n-2)} \int_y^1 1dxdy = \int_0^1 y(n-1)(1-y)^{(n-1)}dy \tag{24}$$

$$= \frac{n-1}{n+n^2}. \tag{25}$$

□

In summary, an aggregate player j expects to be assigned the same amount of work, independently of the behavior of a given player i not in the aggregation. I.e., its expected work is not affected by whether i is honest or dishonest. A similar argument can be used to prove that the same expected work will be assigned to player i independently of the behavior of the aggregate player j.

Proposition 9 *The total expected normalized work assigned to player i, does not depend on the strategies of the other $n-1$ players, and therefore on the perceived strategy of the aggregate player j.*

This allows us to prove that the optimal strategy for a player is to be honest.

Theorem 10 *A player i never does more normalized work (in expectation) by being honest. That is,*

$$E[\bar{W}_i] \leq E[\hat{W}_i], \tag{26}$$

where $E[\hat{W}_i]$ is the expected work performed by player i when it is dishonest.

Proof. For the sake of contradiction, let us suppose this proposition is false. Hence, there is some set of tasks for which, if i is not honest, it performs less work in expectation. I.e., $E[\bar{W}_i] > E[\hat{W}_i]$. From Proposition 9, this holds for any strategy of the aggregate player j, and in particular when all its players are honest. Hence, we can consider in the rest of the proof that the rest of $n-1$ players behave honestly. Additionally, using Proposition 8, we know that the aggregated player, j, will do the same expected work, i.e., $E[\bar{W}_j] = E[\hat{W}_j]$. Now we can define a new mechanism M that assigns a task to player i (when i is honest and declares x) with the same probability as QPQ assigns the task to the player i when she declares a false value z. Then, $E[\bar{W}_i^M] = E[\hat{W}_i]$ and $E[\bar{W}_j^M] = E[\hat{W}_j]$. Hence, it follows that

$$E[\bar{W}_i] + E[\bar{W}_j] > E[\hat{W}_i] + E[\hat{W}_j] \tag{27}$$

$$= E[\hat{W}_i] + E[\bar{W}_j] \tag{28}$$

$$= E[\bar{W}_i^M] + E[\bar{W}_j^M] \tag{29}$$

However, in this case QPQ would not be optimal, since mechanism M a mechanism that reproduces the same task assignments done under i lying (in presence of honest players) would have less expected work. Clearly, this is in contradiction of Proposition 3. Therefore, the best strategy for a player (the one minimizing her normalized work done) is to be honest. □

2.4.5 Real expected utility. Note that the normalized work done by ana honest player, as calculated above, is equal to $\frac{1}{n+n^2}$. But we may wonder what is the real (not normalized) work done. We can easily calculate it in terms of real utility as follows.

Theorem 11 *For each player i, the real expected utility is*

$$E[U_i] = \int_{\Omega} x f_i(x)(1-(1-F_i(x))^{n-1})dx.$$

where the real cost of player i is a continuous random variable with support Ω, probability density function $f_i(\cdot)$, and CDF $F_i(\cdot)$.

Proof. Let $Y_j = PIT(X_j) = F_j(X_j)$ be the uniform random variable that gives the normalized cost for player $j \neq i$ at the time of assigning the tasks (Line 10), and $Y = \min_{j \neq i}\{Y_j\}$. Then, the expected (real) utility of player i is the following:

$$\begin{aligned} E[U_i] &= \int_{\Omega} x f_i(x) Pr(Y \leq F_i(x))dx \\ &= \int_{\Omega} x f_i(x)(1-(1-F_i(x))^{n-1})dx, \end{aligned}$$

where we have used that $Pr[Y \leq y] = 1-(1-y)^{n-1}$ (Proposition 4). □

2.4.6 Fairness. The following result, combined with Proposition 2, will be used to show that all players execute, on expectation, the same number of tasks, even when some players are non-rational or dishonest.

Proposition 12 *Let $X_1, X_2, \ldots, X_n$ be n continuous and i.i.d. random variables, then:*

$$Pr(X_i \leq \min_{j \neq i}\{X_j\}) = \frac{1}{n}.$$

Proof.

$$\begin{aligned} Pr(X_i \leq \min_{j \neq i}\{X_j\}) &= \int_{-\infty}^{\infty} f(y)\left(\int_y^{\infty} f(x)dx\right)^{(n-1)} dy \\ &= \int_{-\infty}^{\infty} f(y)(1-F(y))^{(n-1)}dy \\ &= \int_{-\infty}^{\infty} (1-F(y))^{(n-1)}dF(y) \\ &= -\left[\frac{(1-F(y))^n}{n}\right]_{-\infty}^{\infty} \\ &= \frac{1}{n} \end{aligned}$$

Hence, QPQ not only offers best utility guarantee to honest rational players, but it also offers good properties in environments where nodes have difficulty in estimating costs. This is because, even in environments where the nodes are non-rational, QPQ divides the work fairly and optimaly with respect to the declared normalized costs. Clearly, non-rational players run major efforts, but it is always under completely random task assignments. In other words, the extra cost of non-rational players is caused by their own ignorance, not by the wickedness of the other players. Then, given that players are assigned tasks by choosing the smallest value from a set of i.i.d. random variables (Proposition 2), QPQ ensures that the expectation of the number of tasks executed by each node is $|T|/n$, where recall that T is the set of tasks and n the number of players.

Corollary 13 *In QPQ, players will execute in expectation a proportion of $\frac{1}{n}$ of the tasks, and thus a total of $\frac{|T|}{n}$ of tasks.*

Proof. Declared values follow continuous and independent identically distributed (uniform) random variables in [0,1], from Proposition 2, and therefore applying Proposition 12, each player will execute in expectation a proportion of $\frac{1}{n}$ of the tasks.

2.4.7 Bounds. Finally, we think that it could be interesting to define some ratio that measures the efficiency of the QPQ mechanism. Following concepts similar to the "price of anarchy" [4], we define the measure of *efficiency* as the ratio between the utility of an equilibrium (usually the "worst equilibrium") and the utility of some optimal solution.

Obviously, the player's normalized utility must be between 0, when the node runs all tasks, and $\frac{1}{2}$ when the node has not executed any task. But there are two levels that may be considered as references to establish the goodness of the algorithm. On one side, when a node runs completely random $\frac{1}{n}$ tasks, the expected effort is $\frac{1}{2n}$. On the other hand, the maximum benefit a player i could get occurs when its tasks correspond exactly to her cheapest tasks. In this case, the expected utility would be

$$E[\bar{U}_i^*] = \frac{1}{2} - E[\bar{W}_i^*] \tag{30}$$

$$= \frac{1}{2} - \int_0^{\frac{1}{n}} x dx \tag{31}$$

$$= \frac{1}{2} - \frac{1}{2n^2}. \tag{32}$$

Although this case has null probability, we propose to use this value for our definiton of measure of efficiency.

Definition 4 (Measure of efficiency) We define the measure of efficiency of an algorithm M for tasks assignment under selfish behavior as the ratio between the expected normalized utility obtained under some equilibrium and $E[\bar{U}_i^*] = \frac{1}{2} - \frac{1}{2n^2} = \frac{n^2-1}{2n^2}$. I.e.,

$$\text{Efficiency} = \frac{E[\bar{U}^M]}{E[\bar{U}_i^*]} = \frac{2n^2 E[\bar{U}^M]}{n^2-1}.$$

Hence, we can compute the efficiency of QPQ as

$$\text{Efficiency} = \frac{2n^2(\frac{1}{2} - \frac{1}{n+n^2})}{n^2-1} = \frac{n^2 - (\frac{2n}{n+1})}{n^2-1} > \frac{n^2-2}{n^2-1}.$$

Note that the efficiency of QPQ is close to 1 when the number of participants is high. For instance, with just 10 nodes the efficiency of QPQ is 0.991.

Experiments

3.1 Implementing QPQ in Rreal Eenvironments

In this section, our objective is analyzing what are the restrictions for QPQ to be implemented in real environments. From the above sections, we may claim that the computation and communication capabilities required by the algorithm are affordable with current technology. We do not claim that implementing such capabilities would be an easy task, since there are many technological challenges that should be addressed to do it. Other previous works solve some of them [24]. Thus, our only claim is that it would be feasible.

However, going beyond the required communication and computation capabilities, we may see that a number of issues arise. The first of all is on the definition of selfishness itself. This paper is mainly focused on detecting and neutralizing users publishing values not coming from the PIT of their real costs. However, one can claim that other non-cooperative harmful behaviors are possible such as, for example, not executing tasks at all, or executing them incorrectly. Hopefully, most of these evil conducts can be easily avoided using a two step scheme. First, by detecting such behaviors (previous works on the area show that it is possible [25,26]). Second, by establishing a strong enough punishment to discourage misbehaving players from repeating them. For example, we may adopt the radical solution of just sending off misbehaving users. In order to guarantee that reoffending players do not participate again, all that is needed is that user identities are unique and cannot change on different game instances. Note that QPQ does not discard misbehaving users, because it assumes that the publication of dishonest values cannot be distinguished from the publication of values generated from rationally-limited players, and it would not be reasonable to send off the latter from the game given that, in a realistic scenario, all players would have some rationally limitations (i.e. it is not possible to estimate costs with total accuracy). Hence, QPQ's approach of keeping them in the system is one of the most difficult ways of dealing with selfish users.

Coming back to the subtleties of QPQ, another point to consider is how to re-generate the random value when the system detects a lie. As we said before, we require a deterministic (repeatable) random generator that any of the remaining nodes can use to calculate the new value. One possibility for generating the random value is to use a hash function over the published normalized cost of other nodes. Alternatively, it is possible to request a random value to each player (except the value of player in question) and apply the hash function on them. Even another possibility is to use techniques similar to the procedures proposed by Aumann et al. [27] to generate jointly controlled lotteries. For example, for two players, we can request random values to both of them, and replace the value of the liar's by the sum of these numbers, if the sum is less than 1, or with the sum minus oneone minus the sum otherwise. With this scheme, it is easy to show that when one of the players declares random values according to a uniform distribution, then this process generates random values also uniformly over [0,1], regardless of what the other player does. As a conclusion, we may claim that there are several mechanisms suitable for the generation of the punishment random value independently on the behavior of a dishonest player.

Another obstacle that stands inon the way of a potential implementation of the mechanism is the acceptance test. We have assumed that we have a perfect GoF test function. This is somewhat similar to assuming that we have a set of samples whose number is very large (ideally infinite) for detecting lies with the usual tests. In a real system, this solution is impractical since nodes would require to store all the historical values of the rest of players, and initially the number of samples is necessarily limited. As mentioned before, we propose that players simply generate costs using their particular distribution and apply the PIT using the successive generated samples. The CDF used for the PIT is synthesized from the existing samples y_i. This CDF obtained from samples is known in statistics as the Empirical Cumulative Distribution Function (ECDF).

Definition 5 (Empirical Cumulative Distribution Function) The empirical cumulative distribution function (ECDF) F_n for n observations y_i is defined as

$$F_n(x) = \frac{1}{n}\sum_{i=1}^{n} 1\{y_i \le x\},$$

where $1\{A\}$ is the indicator function or the characteristic function of event A. In our case, it is defined as

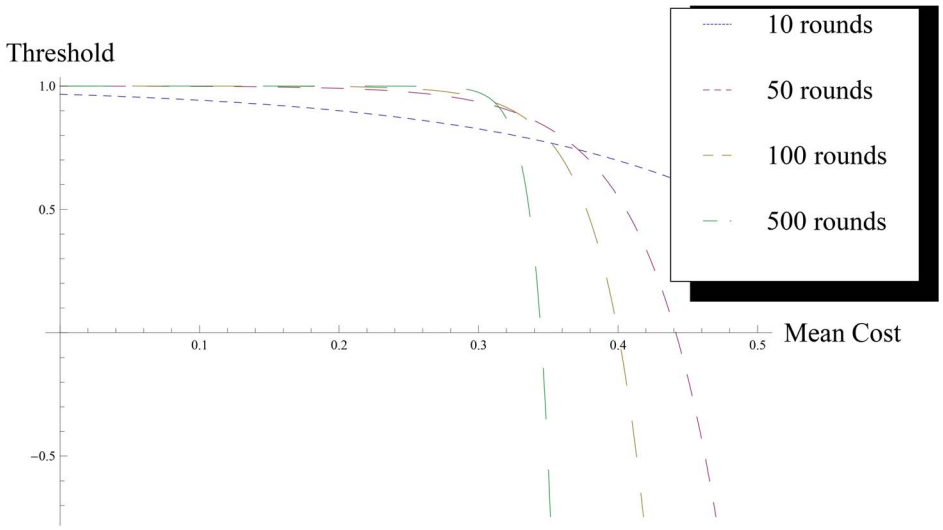

Figure 2. Shape of the threshold curve. This picture depicts the curves of our elastic *p-value* thresholds as function of the normalized utility of a player. When we have a small number of rounds (blue line for 10 rounds) our system is quite tough, but if the number of rounds increases (yellow line correspond to 100 rounds and green line is for 500 rounds), our proposal is more relaxed, and accepts values if the player's utility is within a reasonable range.
doi:10.1371/journal.pone.0066575.g002

$$1\{y_i \le x\} = \begin{cases} 1 & \text{if } y_i \le x, \\ 0 & \text{otherwise.} \end{cases}$$

Obviously, this process has an error which tends to zero when the number of samples (rounds) increases as, by the it is proved by *Glivenko-Cantelli theorem* [28].

Regarding the GoF used, a tremendous number of GoF tests have been proposed in the scientific literature. Some of them may be applied over discrete distributions and others require continuous distributions. The Kolmogorov-Smirnov (KS) test [29,30] is probably the best-known test for continuous distributions, basically due to its simplicity. The KS test calculates the greatest distance between the ECDF associated to a sequence of samples and the CDF we want to check. It may be defined by the following expression:

$$D = \max_{1 \le i \le n} \left\{ F(x_i) - \frac{i-1}{n}, \frac{i}{n} - F(x_i) \right\}$$

where $F(\cdot)$ is the CDF to check, n is the number samples, and $(x_1, x_2, \cdots, x_n)$ is the set of samples arranged in increasing order. What makes the KS test so versatile is that the distribution of the distance D does not depend on the theoretical probability distribution (null hypothesis). Several authors, such as Smirnov [30], Birnbaum and Tingey [31], have obtained exact and approximate expressions of the distribution of the variable D as a random function of the number of available samples. Due the complexity of such expressions, the KS test is often used through tables containing the most common percentiles.

We propose to use the KS test as the GoF test of QPQ. Hence, whenever a new normalized cost is issued, we check the KS test of it, together with the historical sequence of that player, so that we obtain the corresponding (*p-value*). Note that, in statistical significance testing, the *p-value* is the probability of obtaining a particular test statistic on the model at least as extreme as the one that was actually observed. Now, the value is accepted by the test when that *p-value* is over a particular acceptance threshold, $p-th$.

For practical reasons, we need to reduce the history of a user to a relatively small number of samples. Hence, we propose a slight modification to the acceptance test of Algorithm at table 2 to make it implementable in real systems. With this modification, each node applies the KS test using only a small number of the latest published values. However, this makes the KS test susceptible of

Table 3. Implementable Quid Pro Quo mechanism (code for node i, and a generic task t, omitted).

1: Estimate the cost c_i of the task
2: Publish the normalized cost $\bar{c}_i = PIT(c_i)$
3: Wait to receive the normalized costs $\bar{c}_j$ from the other players
4: **for all** $j \in N$ **do**
5: Let $p-th_j \leftarrow \frac{1}{\log(k+1)^{\delta(1-(\mu_{j,k}-\mu)\cdot\sqrt{k})}}$
6: **if** not $KSTest(\bar{c}_j, Historic_j, p-th_j)$ **then**
7: $c_j \leftarrow Random(c_{-j})$
8: **end if**
9: $Historic_j \leftarrow Historic_j \cup \{\bar{c}_j\}$
10: **end for**
11: Let $d = \underset{j \in N}{\operatorname{argmin}}\ \bar{c}_j$
12: **if** $d = i$ **then**
13: execute the task
14: **else**
15: do nothing (node d will execute the task)
16: end if
17: **for all** $j \in N$ **do**
18: recompute $\mu_{j,k+1}$
19: **end for**

doi:10.1371/journal.pone.0066575.t003

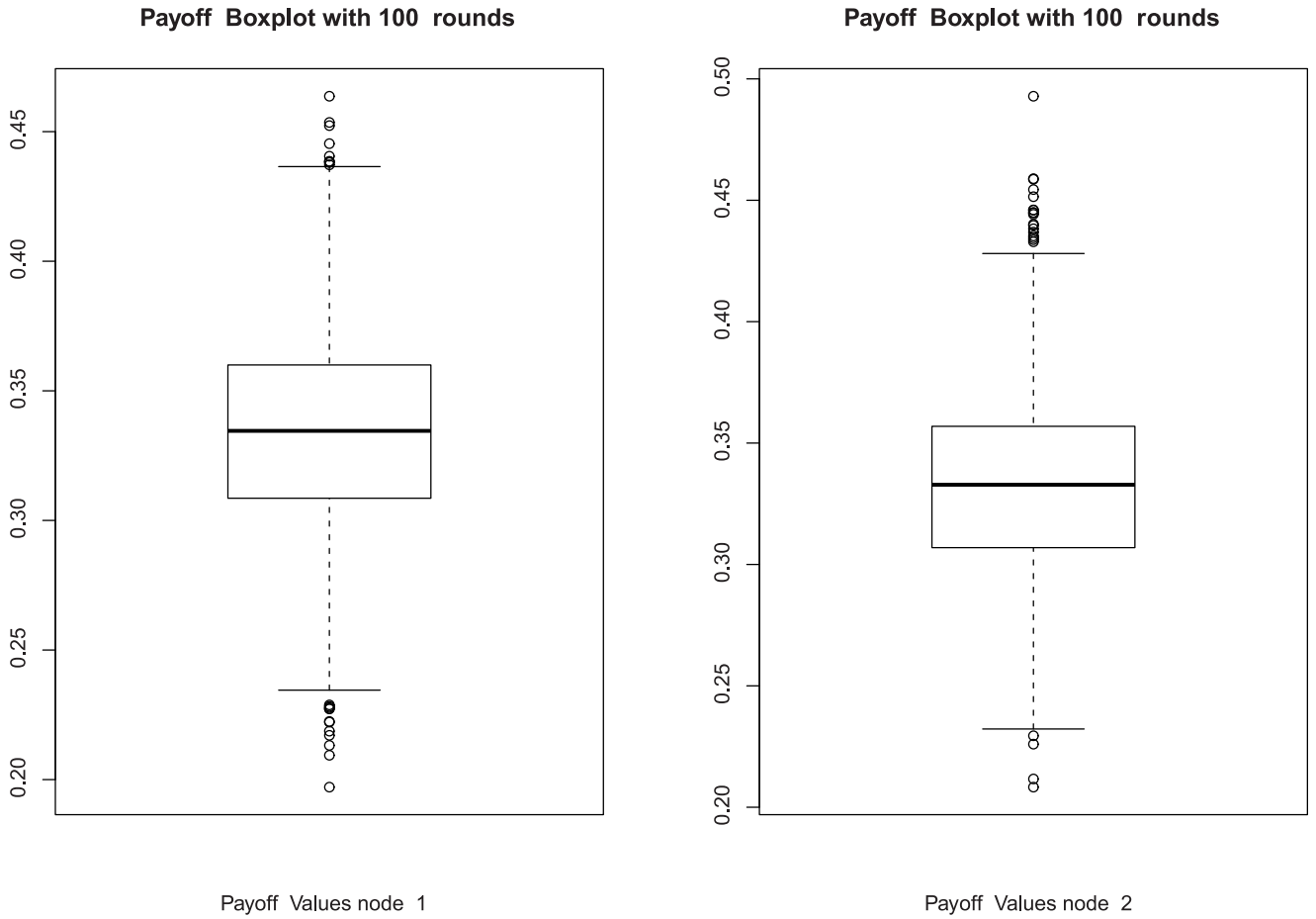

Figure 3. Two honest players with no control (100 rounds).
doi:10.1371/journal.pone.0066575.g003

generating inaccurate estimations. For example, a selfish node could publish values following a Beta distribution (1,0.9). With high probability, this situation could not be detected with sample sequences of small length. In addition toof choosing a large enough sample size (our simulations show that 50 samples are enough), we play with the threshold to refine the test. The idea is to modify the acceptance threshold so that it is hardened when the actual normalized utility of the player is higher than the theoretical expectation, and it is relaxed when players are losing more than expected. There are many ways of implementing this idea, but we propose the following expression

$$p-th_k=\frac{1}{\log(k+1)^{\delta\cdot(1-(\mu_k-\mu)\cdot\sqrt{k})}}, \qquad (33)$$

where δ is a tuning parameter, μ is the expected normalized utility of each playerall players and μ_k is the actual normalized utility of the player at round k. To illustrate this idea we depict Fig. 2, which represents the value of this threshold as a function of the total normalized utility of the player. Clearly, the above formula is entirely empirical, although the simulations below in this paper show that it fits well our requirements. The intuition behind Equation (33) is that, in the initial rounds, a high number of values are rejected by the GOF test, and thus the assignment of tasks uses mostly randomly generated values. This property has two nice features. Firstly, it allows to collect samples to be used in the statistical tests without allowing the players to take advantage of a period of incomplete knowledge. Secondly, it implements the One of the reasons that has led to the development of this proposal has been the idea that a new player must "pay" some kind of "fee" when she enters into the system. In this way, we want to avoid, or at least reduce, the problem of low-cost identities or cheap pseudonyms. With our proposal, at the beginning QPQ assigns tasks almost randomly, while later, when we have more information about players, QPQ assigns tasks optimally. Each player has to "pay" at the beginning working in random assignments and thus, she has no incentive to exit and reenter into the system.

The final implementable QPQ algorithm we propose may be written as presented in Algorithm at table 3.

3.2 Simulations

By performing simulations, we have checked various aspects of the implementable QPQ. First of all, we wondered if the new GoF test may punish fair players by generating false negatives. In this direction, Fig. 3 represents the boxplot of the expectation of the normalized work done in 100 rounds when all players are honest and no GoF test is applied. This picture serves as control, and allows us to compare it with the same game bybut introducing the GoF test of Algorithm at table 3, using a history of 50 samples for the KS test and $\delta=2$. The results are depicted in Fig. 4. As it can

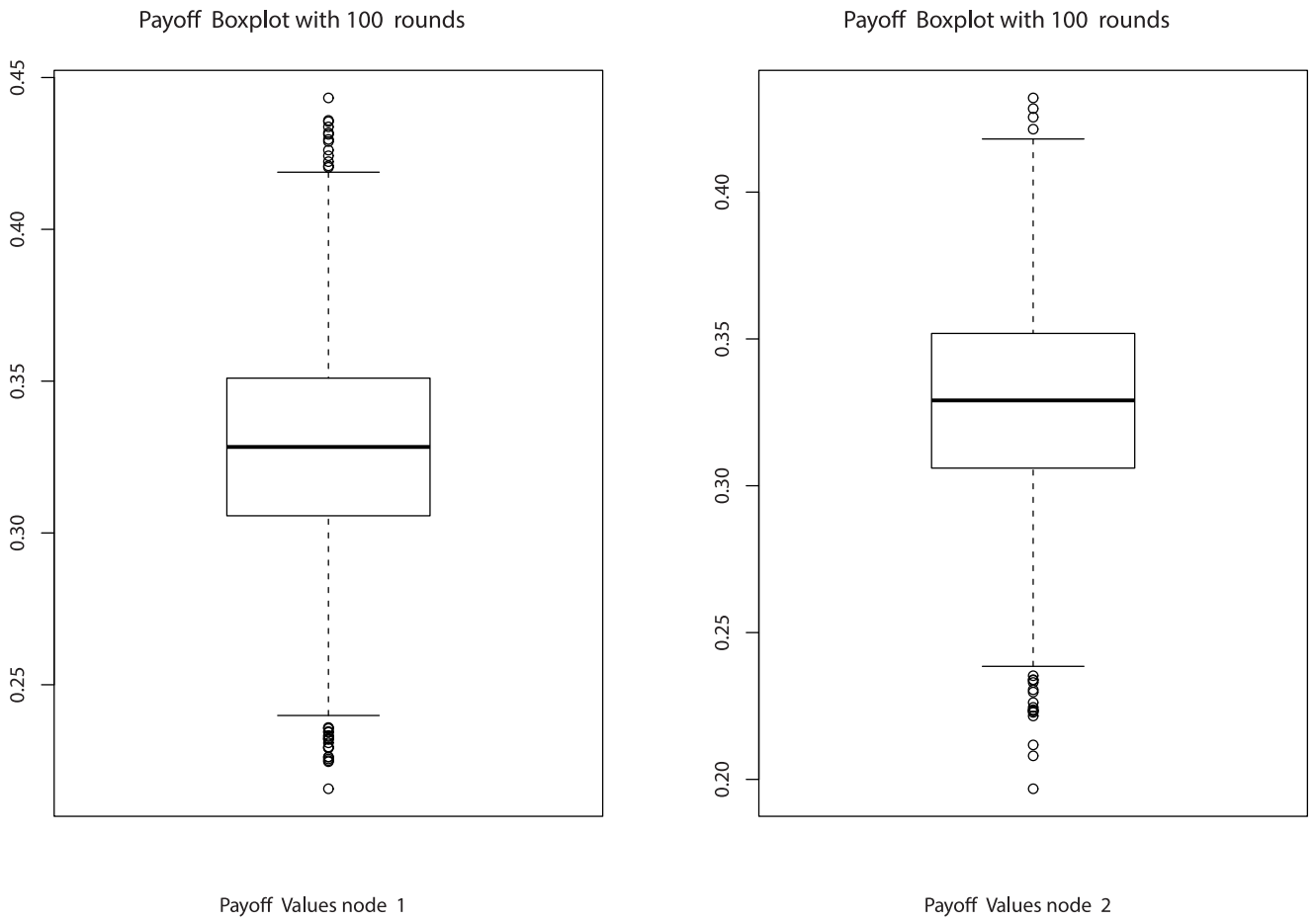

Figure 4. Two honest players with KS control (100 rounds).
doi:10.1371/journal.pone.0066575.g004

be seen, the performance loss caused by false negatives is minimal and barely noticeable in these scenarios.

The next question is to which extend selfish users can fool the algorithm and achieve improvements in their utility. We have simulated dishonest behavior by using several distributions close to the uniform but with higher mean by taking advantage of the properties of the Beta function. T, so that these distributions try to pass the implementable KS test and, at the same time, obtain some profit inon the long run. Again, we have run simulations considering a game with two nodes, one honest (uniform) and one dishonest, for a set of 1,000 rounds, with historical lengh of 50 samples for the implementable KS test and with $\delta = 2$. The results can be seen in Table 4, which depicts the normalized player utilities for different scenarios. In the table, the name *Uniform* represents honest nodes, *Random* is used for non-rational players generating random costs and finally, *"Beta"* and *"Normal"* are used for dishonest players following those distributions. As it can be observed, honest utilities remain quite constant, while non-rational and dishonest utilities decrease, although never under a given limit. Interestingly, note that this behavior is maintained even in the extreme case of a $Beta(1,0.9)$ distribution. Observe that, when the number of samples is small (around 50), a $Beta(1,0.9)$ is so similar to a uniform distribution that it is hardly distinguishable to the eye.

Finally, for the same simulation scenario, in Fig. 5 we compare the behavior of the implementable KS test of Algorithm at table 3 for fair (uniform) users playing against a node with several manipulative profiles (Beta distribution variants) as the number of rounds increase. As it can be observed, the honest player rapidly gets her values to pass the test, while the dishonest gets into trouble rapidly because her values are rejected, even with distributions very similar to the uniform.

Conclusions

Throughout this paper, we have presented QPQ, an algorithm for optimal allocation and execution of tasks in a distributed environment with selfish behavior. Unlike many of the preexisting works, this algorithm proposes a mechanism that does not use

Table 4. Honest vs. dishonest utility/cost.

Distributions	$\bar{U}_1$	$\bar{U}_2$
Uniform vs. Uniform	0.332	0.332
Uniform vs. Random	0.331	0.250
Uniform vs. Beta(1, 0.9)	0.321	0.258
Uniform vs. Beta(1, 0.7)	0.315	0.264
Uniform vs. Normal	0.352	0.250

doi:10.1371/journal.pone.0066575.t004

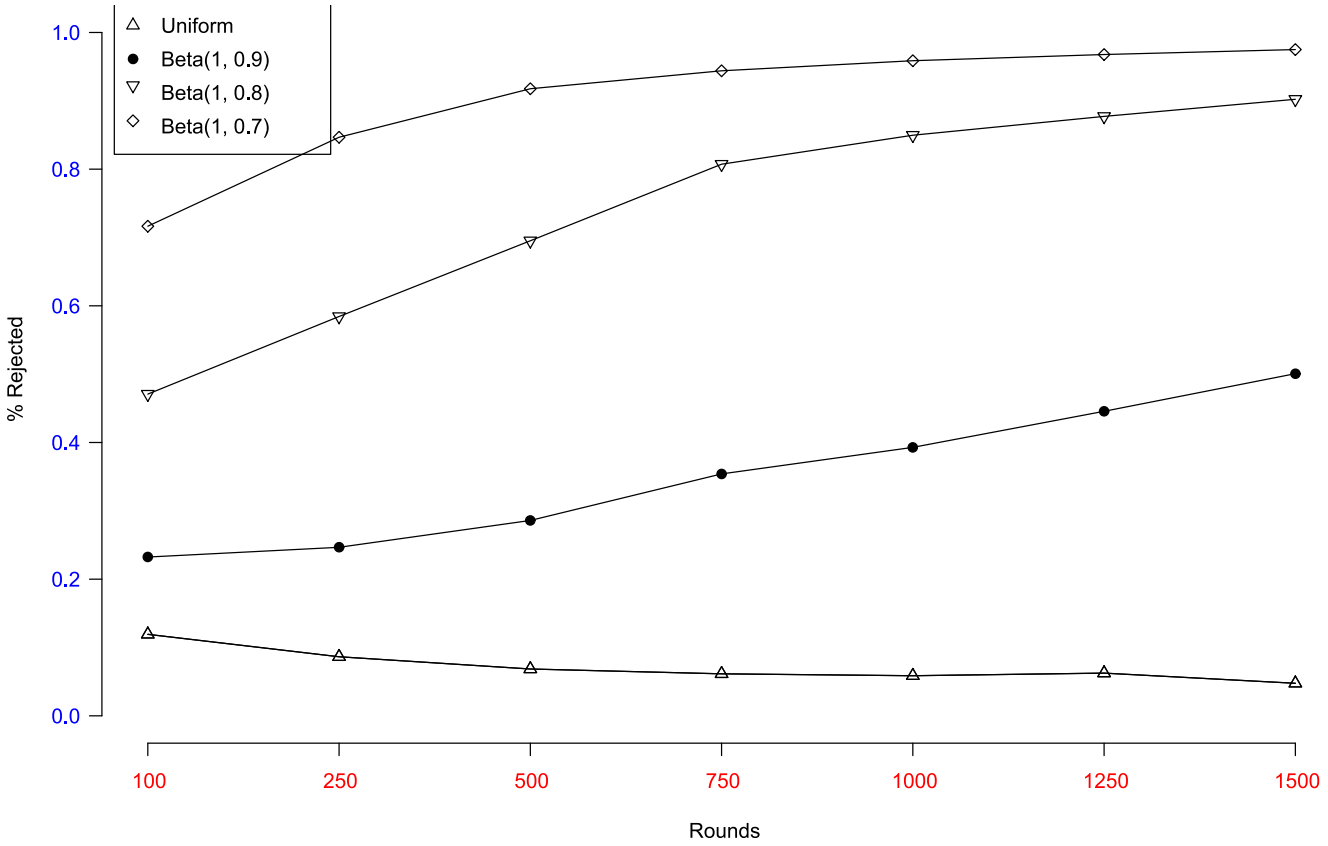

Figure 5. Percentage of rejected values. Simulation results using a control with historical lengh of 50 samples and with $\delta = 2$.
doi:10.1371/journal.pone.0066575.g005

payment or prior information on the behavior of the players. We have demonstrated that the algorithm is tolerant to dishonest, non rational or rationally limited behaviors without punishing fair users and rewarding players proportionally to their degree of truthfulness. The proposed algorithm may be adapted using reasonable approximations so that it can be implemented in real networks with affordable computational and communication complexity. For all these reasons, we claim that this algorithm opens new horizons for the creation of novel computing frameworks where users can openly and effectively cooperate to achieve a common goal, based on the collaborative execution of simple atomic independent tasks.

Despite this, the authors consider necessary to carry out further research to make QPQ robust to more sophisticated selfishness scenarios. For example, it would be necessary to consider cases in which players are not independent, and associate in groups trying to break the system's fairness. QPQ can be also made more robust by tolerating the unreliability of players or the communication between them. Observe that this can be modeled as a random process or as a possible strategy of the players. Another aspect that should be extended is related to the notion of task utility. We have assumed that all nodes have an interest in having all tasks done. However, in a real environment, it is possible that only a subset of tasks are relevant for a given node. Hence, further work should be developed to relax some of the QPQ hypotheses, and deal with this aspect. To conclude, another aspect that may be improved is investigating GoF tests other than the KS to analyze if they can provide advantages for real implementations of the algorithm (for instance, using just a smaller set of samples to implement the acceptance test).

Acknowledgments

We would like to thank Quim Gabarró for useful discussions.

Author Contributions

Wrote the paper: AS AFA LLF.

References

1. Han Z, Niyato D, Saad W, Başar T, Hjørungnes A (2012) Game Theory in Wireless and Communication Networks: Theory, Models, and Applications. Cambridge, UK: Cambridge University Press.
2. Bell MG (2000) A game theory approach to measuring the performance reliability of transport networks. Transportation Research Part B: Methodological 34: 533–545.
3. Srivastava V, Neel J, Mackenzie AB, Menon R, Dasilva LA, et al. (2005) Using game theory to analyze wireless ad hoc networks. Communications Surveys & Tutorials, IEEE 7: 46–56.
4. Koutsoupias E, Papadimitriou CH (2009) Worst-case equilibria. Computer Science Review 3: 65–69.
5. Roughgarden T (2005) Selfish Routing and the Price of Anarchy. The MIT Press.
6. Rosenschein JS, Zlotkin G (1994) Rules of encounter - designing conventions for automated negotiation among computers. In: Rules of Encounter - Designing Conventions for Automated Negotiation among Computers.
7. Papadimitriou CH (2011) Games, algorithms, and the internet. In: Srinivasan S, Ramamritham K, Kumar A, Ravindra MP, Bertino E, et al., editors, WWW. ACM, pp. 5–6.

8. Jackson MO (2003) Mechanism theory. In: Derigs U, editor, Encyclopedia of Life Support Systems, Oxford UK: EOLSS Publishers.
9. Jackson MO (1999) A crash course in implementation theory. Working Papers 1076, California Institute of Technology, Division of the Humanities and Social Sciences.
10. Procaccia AD, Tennenholtz M (2009) Approximate mechanism design without money. In: Proceedings of the 10th ACM conference on Electronic commerce. New York, NY, USA: ACM, EC '09, pp. 177–186. doi:10.1145/1566374.1566401.
11. Jackson MO, Sonnenschein HF (2003) The linking of collective decisions and efficiency. EconWPA Microeconomics : 0303007.
12. Jackson MO, Sonnenschein HF (2007) Overcoming incentive constraints by linking decisions. Econometrica 75: 241–257.
13. Veszteg RF (2005) Linking decisions with moments. Faculty Working Papers 10/05, School of Economics and Business Administration, University of Navarra.
14. Czumaj A, Ronen A (2004) On the expected payment of mechanisms for task allocation. In: Chaudhuri S, Kutten S, editors, Proceedings of the Twenty-Third Annual ACM Symposium on Principles of Distributed Computing, PODC 2004. pp. 98–106.
15. Bauer S, Faratin P, Beverly R (2006) Assessing the assumptions underlying mechanism design for the internet. In: Economics of Networked Systems.
16. Anderegg L, Eidenbenz S (2003) Ad hoc-VCG: a truthful and cost-efficient routing protocol for mobile ad hoc networks with selfish agents. In: MobiCom '03: Proceedings of the 9th annual international conference on Mobile computing and networking. pp. 245–259.
17. Feigenbaum J, Papadimitriou CH, Sami R, Shenker S (2002) A BGP-based mechanism for lowest-cost routing. In: Proceedings of the Twenty-First Annual ACM Symposium on Principles of Distributed Computing, PODC 2002. pp. 173–182.
18. Feigenbaum J, Shenker S (2002) Distributed algorithmic mechanism design: Recent results and future directions. In: Proceedings of the 6th International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications. ACM Press, New York, pp. 1–13.
19. Angus JE (1994) The probability integral transform and related results. SIAM Rev 36: 652–654.
20. Ferguson TS (1967) Mathematical Statistics: A Decision Theoretic Approach. Academic Press, 1st edition.
21. Burgert C, Rüschendorf L (2006) On the optimal risk allocation problem. Statistics and Decisions 24: 153–171.
22. Rüschendorf L (2009) On the distributional transform, Sklar's theorem, and the empirical copula process. Journal of Statistical Planning and Inference 139: 3921–3927.
23. Evans M, Hastings N, Peacock B (2000) Statistical Distributions. New York: John Wiley & Sons.
24. Murray DG, Yoneki E, Crowcroft J, Hand S (2010) The case for crowd computing. In: Proceedings of the second ACM SIGCOMM workshop on Networking, systems, and applications on mobile handhelds. MobiHeld '10, pp. 39–44.
25. Fernández Anta A, Georgiou C, Mosteiro MA (2010) Algorithmic mechanisms for internet-based master18 worker computing with untrusted and selfish workers. In: 24th IEEE International Symposium on Parallel and Distributed Processing, IPDPS 2010. IEEE, pp. 1–11.
26. Fernández Anta A, Georgiou C, López L, Santos A (2012) Reliable internet-based master-worker computing in the presence of malicious workers. Parallel Processing Letters 22.
27. Aumann R, Maschler M, Stearns R (1995) Repeated Games with Incomplete Information. MIT Press.
28. Dehardt J (1971) Generalizations of the Glivenko-Cantelli Theorem. The Annals of Mathematical Statistics 42: pp. 2050–2055.
29. Kolmogorov AN (1933) Sulla determinazione empirica di una legge di distribuzione. Giornale dell'Istituto Italiano degli Attuari 4: 83–91.
30. Smirnov NV (1939) On the estimation of the discrepancy between empirical curves of distribution for two independent samples. Bulletin of Moscow University 2: 3–16.
31. Birnbaum ZW, Tingey FH (1951) One-sided confidence contours for probability distribution functions. The Annals of Mathematical Statistics 4: 592–596.