Gossip-Based Aggregation of Trust in Decentralized Reputation Systems*

Yoram Bachrach Ariel Parnes Ariel D. Procaccia
Jeffrey S. Rosenschein
School of Engineering and Computer Science
The Hebrew University of Jerusalem
{yori,arielp02,arielpro,jeff}@cs.huji.ac.il

Abstract

Decentralized Reputation Systems have recently emerged as a prominent method of establishing trust among self-interested agents in online environments. A key issue is the efficient aggregation of data in the system; several approaches have been proposed, but they are plagued by major shortcomings.

We put forward a novel, decentralized data management scheme grounded in gossip-based algorithms. *Rumor mongering* is known to possess algorithmic advantages, and indeed, our framework inherits many of its salient features: scalability, robustness, globality, and simplicity.

We demonstrate that our scheme motivates agents to maintain a sparkling clean reputation, by showing that the higher an agent's reputation is above the threshold set by her peers, the more transactions she would be able to complete within a certain time unit. We analyze the relation between the amount by which an agent's average reputation exceeds the threshold and the time required to close a deal. This analysis is carried out both theoretically, and empirically through a simulation system called *GossipTrustSim*. Finally, we show that our approach is inherently impervious to certain kinds of attacks.

1 Introduction

In open multiagent environments, self-interested agents are often tempted to employ deceit as they interact with others. Fortunately, dishonest agents can expect their victims to retaliate in future encounters. This "shadow of the future", as Axelrod termed it [6], motivates cooperation and trustworthiness.

However, as the size of the system grows, agents have an increasingly small chance of dealing with another agent they already know; as a consequence, building trust in domains teeming with numerous agents becomes much harder. One prominent example of such an environment is the Internet, which has a key benefit of allowing interaction

*A preliminary version of this paper appeared in the proceedings of IJCAI 2007.

between entities who are not familiar with each other. Since the value of a transaction depends heavily on the reliability of the parties involved in it, designing mechanisms that allow the obtaining of information regarding the parties involved in a possible transaction is very important in such environments.

Reputation systems address this problem by collecting, maintaining, and disseminating reports among agents, so that agents may learn from others' experience. To put it differently, agents are intimidated by the "shadow of the future" today, even though tomorrow they are most likely to meet total strangers. Also, such information may allow an agent to *choose* the most suitable party with whom to carry out a transaction. Such systems have been used in a wide variety of applications, in electronic commerce, auctions, and many peer-to-peer systems.

Reputation systems can be decomposed into two major components: 1) the trust model, which describes whether an agent is trustworthy, and 2) the data management scheme. The latter component poses some interesting problems, since it is imperative to efficiently aggregate trust-related information in the system. A simple solution is to maintain a central database that contains the feedback gathered from past transactions.

Unfortunately, the centralized solution is inappropriate in distributed environments where scalability is a major concern, as the database soon becomes a bottleneck of the system. Moreover, this approach is not robust to failures. Previous work on *decentralized* reputation schemes suffered from their own major problems: agents have to maintain complex data structures, evaluation of trust is based only on local information, or there are restrictive assumptions on the trust model.[1]

We approach this hornets' nest by designing a novel method of trust aggregation (i.e., a reputation system's data management scheme). That is the focus of this article. The method is demonstrated in this paper for a simple trust model, but it can be extended to more complex models.

The roots of our *gossip-based* approach to reputation system data management can be traced to a seminal paper by Frieze and Grimmett [16]: a rumor starts with one agent; at each stage, each agent that knows the rumor spreads it to another agent chosen uniformly at random. The authors show that the rumor reaches all agents quickly (a result that roughly coincides with situations in real life).

We directly rely on more recent results, surveyed in the next section. It has been shown that aggregate information, such as averages and sums of agents' inputs, can be calculated using similar methods of uniform gossip in a way that scales gracefully as the number of agents increases. Furthermore, the approach is robust to failures, and the results hold even when one cannot assume a point-to-point connection between any two agents (as is the case in peer-to-peer [P2P] networks).

In our setting, each agent merely keeps its private evaluation of the trustworthiness of other agents, based on its own interactions.[2] When an agent wishes to perform a transaction with another, it obtains the *average* evaluation of the other's reputation from all agents in the system, using a gossip-based technique.

Although the algorithms that we present estimate the *average* reputation, they can be easily adapted to estimating whether a certain agent has a high reputation in the eyes

[1] The "or" is not exclusive.

[2] The question of how agents set this valuation is outside the scope of this paper.

of the *majority* of the agents, or certain other similar metrics. Thus, the framework we advocate for aggregating reputation information accommodates more sophisticated trust models.

Some advantages are immediately self-evident. Each agent stores very little information, which can be simply and efficiently organized, and evaluation of trust is based on global information. Additionally, this framework inherits the advantages of gossip-based algorithms: scalability, robustness to failure, decentralization, and as a consequence, applicability in peer-to-peer networks.

We have implemented a system called *GossipTrustSim*, for simulating transactions among agents who use our suggested procedure to decide whether or not to carry out a transaction. This simulation is used to investigate some of the properties of our method.

An important desideratum one would like a reputation system to satisfy is motivating agents to maintain an untarnished reputation, i.e., to be *absolutely trustworthy* (as opposed to, say, being generally trustworthy but occasionally cheating). We claim that our data management scheme, together with an extremely simple trust model, satisfies this property, by showing that the higher an agent's reputation is above the threshold set by the peers with whom she wishes to interact, the more transactions she would be able to complete within a certain time unit. This claim is supported by both theoretical results, as well as by empirical results obtained using *GossipTrustSim*.

We also demonstrate that our trust data management scheme has another advantage: it is *inherently resistant to some attacks* (with no assumptions on the trust model). This is a positive side effect of the exponential convergence rates of the algorithms we use.

In this paper we do *not* address the problem of designing a trust model. Rather, we suggest an approach for agents to aggregate distributed trust information so as to decide with whom to carry out transactions.

The article is organized as follows. Section 2 discusses gossip-based methods for aggregating information, which lie at the core of the method we suggest. Section 3 introduces our method of gossip-based aggregation of trust information for decentralized reputation systems. In Section 4, we theoretically analyze how keeping a very high reputation allows an agent to shorten the time she requires to achieve a successful transaction. Section 5 discusses simulation results that also support this claim. In Section 6, we consider certain kinds of attacks that can be used by agents who attempt to manipulate reputation systems, and show that our method is resistant to such attacks. Section 7 discusses important related work; we conclude in Section 8.

2 Gossip-Based Information Aggregation

In this section, we survey the relevant results of Kempe, Dobra and Gehrke [20]. These algorithms allow us to estimate the average of values held at network nodes (in our case, these values will be the reputation values concerning a particular agent). [20] also shows how to calculate other functions over these values, such as the majority function and sum. Thus our algorithms can be adapted for other, more sophisticated models of trust.

2.1 Push-Sum

We begin by describing a simple algorithm, PUSH-SUM, to compute the average of values at nodes in a network. There are n nodes in the system, and each node i holds an input $x_i \geq 0$. At time t, each node i maintains a *sum* $s_{t,i}$ and a *weight* $w_{t,i}$. The values are initialized as follows: $s_{0,i} = x_i$, $w_{0,i} = 1$. At time 0, each node i sends the pair $s_{0,i}, w_{0,i}$ to itself; at every time $t > 0$, the nodes follow the protocol given as Algorithm 1.

Algorithm 1

procedure PUSH-SUM
  Let $\{(\hat{s}_l, \hat{w}_l)\}_l$ be all the pairs sent to i at time $t-1$
  $s_{t,i} \leftarrow \sum_l \hat{s}_l$
  $w_{t,i} \leftarrow \sum_l \hat{w}_l$
  Choose a target $f_t(i)$ uniformly at random
  Send the pair $(\frac{1}{2}s_{t,i}, \frac{1}{2}w_{t,i})$ to i and to $f_t(i)$
  $\frac{s_{t,i}}{w_{t,i}}$ is the estimate of the average at time t
end procedure

2.2 Convergence and the Diffusion Speed of Uniform Gossip

Let $U(n, \delta, \epsilon)$ (the *diffusion speed* of uniform gossip) be an upper bound on the number of turns PUSH-SUM requires so that for all $t \geq U(n, \delta, \epsilon)$ and all nodes i,

$$\frac{1}{\sum_k x_k} \cdot \left| \frac{s_{t,i}}{w_{t,i}} - \frac{1}{n} \sum_k x_k \right| \leq \epsilon$$

(the relative error is at most ϵ) with probability at least $1 - \delta$.

Theorem 1 ([20]).

1. *$U(n, \delta, \epsilon) = O(\log n + \log \frac{1}{\delta} + \log \frac{1}{\epsilon})$.*
2. *The size of all messages sent at time t by* PUSH-SUM *is $O(t + \max_i \mathit{bits}(x_i))$, where $\mathit{bits}(x_i)$ is the number of bits in the binary representation of x_i.*

2.3 Advantages of Push-Sum

A major advantage of gossip-based algorithms is their robustness to failures: the aggregation persists in the face of failed nodes, permanent communication failures, and other unfortunate events. Further, no recovery action is required. The assumption is that nodes can detect whether their message has reached its destination; PUSH-SUM is modified so that if a node detects its target failed, it sends its message to itself.

Theorem 2 ([20]). *Let $\mu < 1$ be an upper bound on the probability of message loss at each time step, and let U' be the diffusion speed of uniform gossip with faults. Then:*

$$U'(n, \delta, \epsilon) = \frac{2}{(1-\mu)^2} U(n, \delta, \epsilon).$$

In several types of decentralized networks, such as P2P networks, point-to-point communication may not be possible. In these networks, it is assumed that at each stage nodes send messages to all their neighbors (*flooding*). When the underlying graph is an expander, or at least expected to have good expansion, results similar to the above can be obtained. Fortunately, it is known that several peer-to-peer topologies induce expander graphs [25, 21].

2.4 Push-Sum Notes

In the rest of the paper, we have $x_i \leq 1$, and in particular $\sum_i x_i \leq n$. Therefore, it is possible to redefine U to be an upper bound on the number of turns required so that for all $t \geq U$ and all nodes i, the *absolute error* $\left|\frac{s_{t,i}}{w_{t,i}} - \frac{1}{n}\sum_k x_k\right|$ is at most ϵ with confidence $1-\delta$, and it still holds that $U(n, \delta, \epsilon) = O(\log n + \log \frac{1}{\delta} + \log \frac{1}{\epsilon})$. Hereinafter, when we refer to U we have this definition in mind.

Remark 1. The protocol PUSH-SUM is presented in terms of a synchronized starting point, but this assumption is not necessary. A node that poses the query may use the underlying communication mechanism to inform all other nodes of the query; convergence times are asymptotically identical.

3 Our Framework

Let the set of agents be $N = \{1, \ldots, n\}$. Each agent $i \in N$ holds a number $r_i^j \in [0, 1]$ for each agent $j \in N$ (including itself). This number represents j's reputation with respect to i, or to put it differently, the degree to which i is willing to trust j. As agents interact, these assessments are repeatedly updated. We do not in general concern ourselves with how agents set these values.

When an agent i is deliberating whether to deal with another agent j, i wishes to make an informed evaluation of the other's reputation. Let $\bar{r}^j = \frac{\sum_k r_k^j}{n}$ be the average of j's reputation with respect to all agents. Knowledge of $\bar{r}^j$ would give i a good idea of how trustworthy j is (this is, of course, a simple model of trust).

3.1 Gossip Based Trust

We show that in this scenario, agents can use gossip-based algorithms to decide with whom to carry out transactions. Also, in such a setting, agents are encouraged to keep a completely untarnished reputation. Similar results can be obtained for more complex trust models.

A simple way to compute the average trust is via PUSH-SUM.

The protocol EVAL-TRUST is given as Algorithm 2. PUSH-SUM is executed for $U = U(n, \delta, \epsilon)$ stages. At time U, it holds for all $k \in N$, and in particular for agent i, that $\left|\frac{s_{t,i}}{w_{t,i}} - \bar{r}^j\right| \leq \epsilon$, with probability $1-\delta$. In other words, the algorithm returns a very good approximation of j's average reputation.

Algorithm 2

```
1: procedure EVAL-TRUST(i, j, δ, ε)        ▷ i evaluates r̄^j with accuracy ε, confidence 1 − δ
2:     for all k ∈ N do
3:         x_k ← r_k^j             ▷ Inputs to PUSH-SUM are j's reputation w.r.t. agents
4:     end for
5:     run PUSH-SUM for U = U(n, δ, ε) stages
6:     return s_{U,i} / w_{U,i}
7: end procedure
```

3.2 Use of the Technique

In practice, when two agents i and j interact, i may evaluate j's reputation (and vice versa) by calling EVAL-TRUST. The protocol quickly returns the approximation of $\bar{r}^j$, *based on the values* r_k^j *at the time* EVAL-TRUST *was called*. Each agent i keeps different values $s_{t,i}$ and $w_{t,i}$ for every different query that was issued by some other agent in the system, and updates these values repeatedly according to PUSH-SUM. Thus, at any stage every agent participates in many parallel executions of PUSH-SUM.

A possible cause for concern is the amount of communication each agent has to handle at every turn. However, the quick convergence of PUSH-SUM implies that often the burden would not be too great. Indeed, assume that the number of new interactions at each turn is bounded by a constant c (or at worst is very small compared to n). Each such new interaction results in at most two new executions of EVAL-TRUST, but the execution lasts at most U turns. To conclude the point, under the foregoing assumption, each agent sends at most $c \cdot U = O(\log n)$ messages per turn.

Remark 2. The size of messages depends on how the r_i^j are calculated, and as mentioned above, this issue is outside the scope of this paper. Nevertheless, there would usually be a constant number of reputation levels (say, for instance, that r_j^i can assume the values $r_j^i \in \{0, 0.1, 0.2, \ldots, 1\}$), so the message size would normally be constant.

As the above method of aggregating an agent's average reputation relies on the gossip-based algorithm PUSH-SUM, it inherits all the latter's benefits, in particular robustness to failure and applicability in peer-to-peer networks.

4 The Benefit of an Unstained Reputation

It is very desirable (indeed, crucial) that a reputation system be able to induce truthfulness in agents. Naturally, an agent with a stained reputation would be shunned by its peers, while an agent with a good reputation would easily solicit deals and transactions. A further step in this direction is motivating agents *never* to cheat.

4.1 Reputation and Speed of Transaction

An agent with a *generally* good reputation, one that is slightly above the required threshold of her peers—an agent that only occasionally cheats—would probably be able to win the confidence of her peers; there is seemingly no reason why an agent should not "play false" now and again. Nevertheless, we consider in this section an extremely simple and general trust model, and show that with the data management scheme that we have presented, there is a social benefit to having a very high reputation: the higher the agent's reputation, the shorter the time required to close deals.

We consider a model in which each agent i has a reputation threshold r_i^{thr} (similar to [27]) and a confidence level δ_i: agent i is willing to deal with an agent j iff i knows that j's average reputation is at least r_i^{thr}, with confidence $1 - \delta_i$. i evaluates j's reputation as above, using EVAL-TRUST. Recall that when the algorithm terminates, agent i only has an ϵ-close approximation of $\bar{r}^j$. If $\frac{s_{t,i}}{w_{t,i}}$ is very close to r_i^{thr}, i would have to increase the accuracy.

Remark 3. We still do not commit to the way the values r_j^i are determined and updated, so the above trust model is quite general.

Algorithm 3

procedure DECIDE-TRUST(i, j) ▷ i decides if it wants to deal with j
    $\epsilon \leftarrow 1/2$ ▷ Initialization
    $k_1 \leftarrow 0$
    loop
        $k_2 \leftarrow U(n, \delta_i, \epsilon)$
        run EVAL-TRUST(j) for another $k_2 - k_1$ stages ▷ A total of k_2 stages
        if $s_{t,i}/w_{t,i} < r_i^{thr} - \epsilon$ **then**
            return false
        else if $s_{t,i}/w_{t,i} > r_i^{thr} + \epsilon$ **then**
            return true
        end if
        $k_1 \leftarrow k_2$
        $\epsilon \leftarrow \epsilon/2$
    end loop
end procedure

The procedure DECIDE-TRUST, given as Algorithm 3, is a straightforward method of determining whether $\bar{r}^j \geq r_i^{thr}$. Agent i increases the accuracy of the evaluation by repeatedly halving ϵ, until it is certain of the result. In this context, a stage of EVAL-TRUST corresponds to a stage of PUSH-SUM.

4.2 Theoretical Analysis of Speed of Transaction

Proposition 3. *Let $i, j \in N$, and $\Delta_{ij} = |\bar{r}^j - r_i^{thr}|$. With probability at least $1 - \delta_i$,* DECIDE-TRUST *correctly decides whether agent j's reputation is at least r_i^{thr} after*

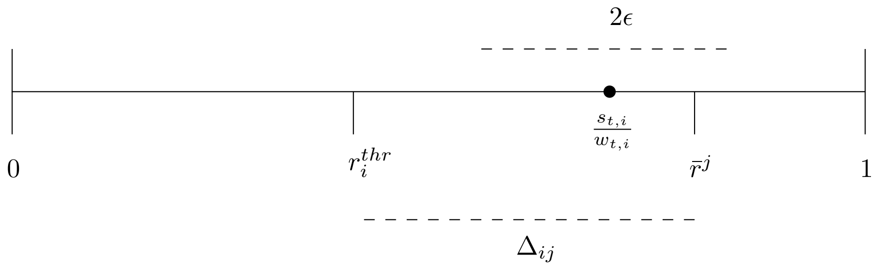

Figure 1: Illustration for the proof of Proposition 3. Once $\epsilon < \Delta_{ij}/2$ and $s_{t,i}/w_{t,i} > r_i^{thr} + \epsilon$, it holds with probability $1 - \delta_i$ that $\bar{r}^j \geq r_i^{thr}$.

$O(\log n + \log \frac{1}{\delta_i} + \log \frac{1}{\Delta_{ij}})$ *stages of* EVAL-TRUST.[3]

Proof. Assume w.l.o.g. that $r_i^{thr} < \bar{r}^j$, and that the algorithm reached a stage t_0 where $\epsilon < \Delta_{ij}/2$. At this stage, it holds that $|\frac{s_{t,i}}{w_{t,i}} - \bar{r}^j| \leq \epsilon$ (with probability $1 - \delta_i$), and therefore:

$$\begin{aligned} \frac{s_{t,i}}{w_{t,i}} &\geq \bar{r}^j - \epsilon \\ &= r_i^{thr} + \Delta_{ij} - \epsilon \\ &> r_i^{thr} + \epsilon. \end{aligned}$$

Hence, the algorithm surely terminates when $\epsilon < \Delta_{ij}/2$. Now the proposition follows directly from the fact that $U(n, \delta_i, \Delta_{ij}) = O(\log n + \log \frac{1}{\delta_i} + \log \frac{1}{\Delta_{ij}})$. $\square$

To conclude, Proposition 3 implies that there is a benefit for agent j in maintaining a high reputation: for any agent i with a reasonable threshold, Δ_{ij} is significant, and this directly affects the running time of DECIDE-TRUST.

Remark 4. The result is limited, though, when the number of agents n is large, as the time to evaluate an agent's reputation is also proportional to $\log n$.

5 Reputation and Speed of Transactions: Simulation Results Using *GossipTrustSim*

Section 4 discussed how high reputation has a social benefit for an agent, as it shortens the time that the agent requires to close deals. In that model, agent i is willing to interact with agent j iff i believes j's average reputation is above a certain threshold value r_i^{thr} with confidence level of δ_i. Section 4 presented a simple algorithm, *Decide-Trust*, that agents can use in this model. The algorithm enables an agent to bound

[3]The probability is the chance that the algorithm will answer incorrectly; the bound on the number of stages is always true.

her probability of accepting a transaction by mistake (i.e., the probability of the agent deciding that $\bar{r}^j > r_i^{thr}$, when in fact it is not) by a desired confidence level, δ_i. The algorithm automatically chooses the number of Push-Sum steps to perform, based on the Push-Sum results obtained along the way.

We have denoted the amount by which an agent j's average reputation truly is above the desired threshold for transactions required by agent i as $\Delta_{ij} = |\bar{r}^j - r_i^{thr}|$. In Section 4, we showed that the higher an agent's reputation is above the required threshold (that is, the higher Δ_{ij} is), the fewer Push-Sum steps are required. More precisely, the number of *Eval-Trust* calls is $O(\log n + \log \frac{1}{\delta_i} + \log \frac{1}{\Delta_{ij}})$. We now provide the results of several simulations, which exactly characterize how the time required to close a deal depends on Δij.

Although our empirical results only concern certain trust models, they indicate that agents who are significantly above the required reputation threshold can indeed significantly shorten the time they need to close a single deal, and are thus able to complete more deals within a given time unit. These simulation results complement our theoretical analysis in the previous section.

We constructed a system called *GossipTrustSim*, for simulating transactions among agents who use *Decide-Trust* (Algorithm 3), to decide whether or not to perform a transaction. *GossipTrustSim* has been implemented in Java, and includes an implementation of *Push-Sum* and of our suggested *Eval-Trust* and *Decide-Trust* (Algorithms 2 and 3).

5.1 Empirical Analysis of Speed of Transaction

The simulations deal with a certain agent, x, who is interested in a transaction with agent j. In each such simulation, for each agent i the system randomly chooses r_i^j, j's reputation in the eyes of i—the degree to which i is willing to trust j—according to a certain model. In all the simulations presented in this paper we use a straightforward model, where r_i^j is drawn from the same distribution for any agent i. Once the r_i^j values are set for all agents i, we then simulate what happens when agent x attempts to decide whether to interact with agent j, using our suggested *Decide-Trust* algorithm. Each such simulation runs *Decide-Trust* until agent x decides whether to trust agent j or not, and we count the total number of Push-Sum steps performed.

There are 2500 agents in our simulations. The initial value for ϵ is set to be $\epsilon_1 = 0.5$. Since this is a very large value, Eval-Trust is typically called more than once. The threshold value in our simulation is $r_x^{thr} = 0.5$, so agent x wants to accept the transaction only if j's average reputation is $\bar{r}^j > 0.5$. Our confidence level is $\delta_x = 0.1$, so agent x would accept the transaction only if $\bar{r}^j > r_x^{thr}$ with a probability of at least $1 - \delta_x = 0.9$.

In our first simulation, each r_i^j is distributed uniformly between a minimal value a and a maximal value $b = 1$; thus, its expectation is $\mathbb{E}\left[r_i^j\right] = \frac{1+a}{2}$. Since there are many agents, the average reputation of agent j is an approximately normal distribution. We simulate the *Decide-Trust* run when in fact $\bar{r}^j > r_x^{thr}$ (and a is chosen so that this is indeed the case), so agent x should eventually accept the transaction. It is easy to see that the higher a is, the higher $\mathbb{E}\left[r_i^j\right]$ for all agents i, and the higher $\mathbb{E}\left[\Delta_{xj}\right]$ is.

We used *GossipTrustSim* to find the required number of Push-Sum steps for agent x to approve the transaction with agent j. We did this for several values of a, and for each value we repeated the simulation 100 times, then calculated the average number of required Push-Sum steps. The results are presented in Figure 2, which shows the relation found between Δ_{xj}, the amount by which agent's j's reputation exceeds the threshold required by agent x, and the number of Push-Sum steps agent x has to perform before approving the transaction.

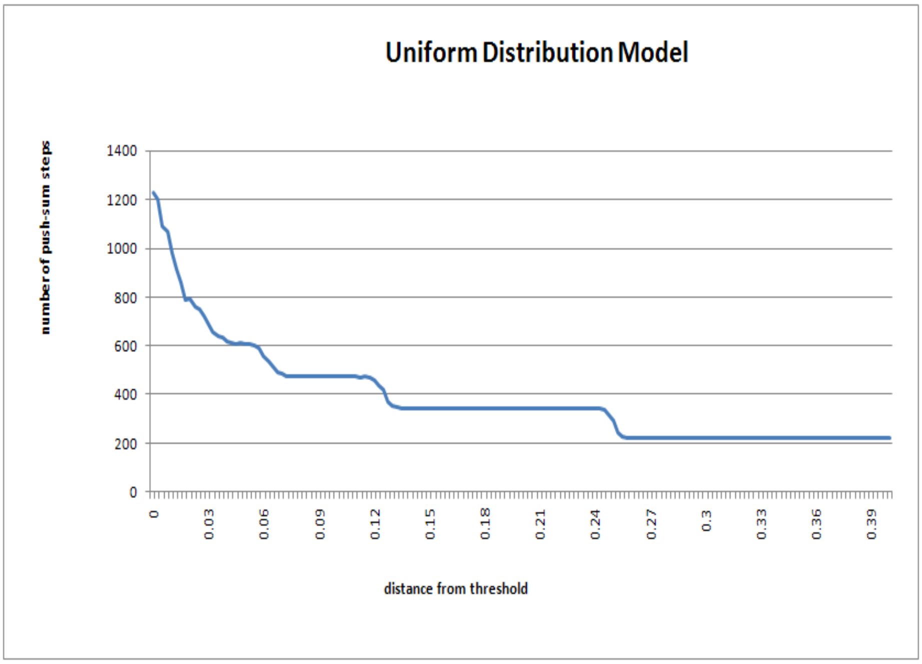

Figure 2: Uniform Distribution Model: the relation between the distance from the threshold, Δ_{xj}, and the required number of Push-Sum steps.

It is easy to see from Figure 2 that indeed agent j's situation is better when her average reputation is well above agent x's threshold. The higher her reputation is, the fewer Push-Sum steps are required for agent x to accept the transaction. This means that although an agent can successfully complete transactions with another agent when her reputation is only slightly above the other agent's required threshold, she would be able to complete more transactions within a time unit by making sure her reputation is much higher than that threshold value. However, the speedup achieved by increasing the reputation diminishes when the reputation is well above the threshold.

Another thing to note is that there are "bumps" in the graph, where suddenly a small increase in the distance from the threshold, Δ_{xj}, results in a big decrease in the required number of Push-Sum steps. This occurs since Algorithm 3, Decide-Trust, calls Algorithm 2, Eval-Trust, several times, and halves the value of ϵ each time. Since each such call to Eval-Trust requires several Push-Sum steps, when typically one less call to Eval-Trust is required, there is a significant decrease in the required number of Push-Sum steps.

In our second simulation, each r_i^j has a normal distribution, with expectation μ and variance $\sigma^2 = 0.0001$. As before, since there are many agents, the average reputation of agent j is also an approximately normal distribution, but with a much smaller variance. Again, μ is chosen so that in fact $\bar{r}^j > r_x^{thr}$ (so we only simulate for values $\mu > r_x^{thr}$), so agent x should eventually accept the transaction. The higher μ is, the higher $\mathbb{E}[\Delta_{xj}]$ is. As before, we tested the number of Push-Sum steps required for agent x to approve the transaction with agent j for several values of μ. For each value, we repeated the simulation 100 times, and calculated the average number of required Push-Sum steps. The results are presented in Figure 3, which shows the relation found between Δ_{xj} and the number of required Push-Sum steps.

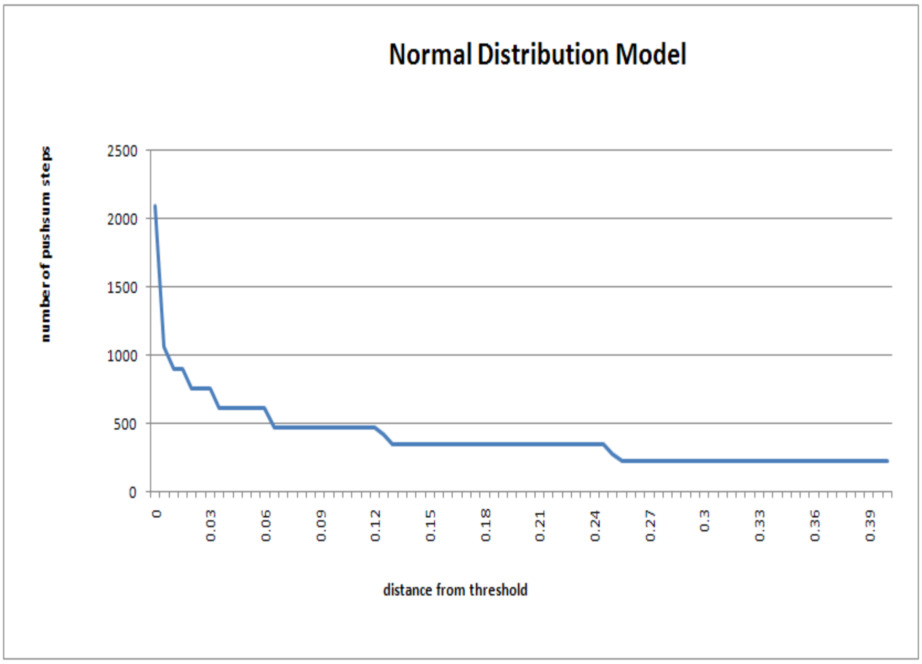

Figure 3: Normal Distribution Model

Similarly to the uniform distribution model, Figure 3 shows that agent j's situation is better when her average reputation is well above agent x's threshold. In this model as well, the higher her reputation is, the fewer Push-Sum steps are required for agent x to accept the transaction, so she can complete more transactions per time unit.

To conclude, the simulations support the claim that when using the gossip-based approach, agents are incentivized to make sure their reputation is well above the threshold value they need in order to complete transactions. Agents who occasionally cheat may still be able to complete transactions, as long as their reputation does not drop below the threshold chosen by the agents with whom they wish to interact, but they would be able to perform fewer transactions in each time unit.

6 Resistance to Attacks

We have seen that information about an agent's reputation can be efficiently propagated, as long as all agents consistently follow EVAL-TRUST. However, with reputation systems we are usually dealing with self-interested agents. In our context, a manipulative agent may artificially increase or decrease the overall evaluation of some agent's reputation by deviating from the protocol.

In the framework we have presented, trust is evaluated on the basis of global knowledge, i.e., the average of all reputation values in the system. Therefore, any small coalition cannot significantly change the average reputation of some agent j by setting their own valuations r_i^j to legal values in $[0, 1]$, and then following the protocol EVAL-TRUST.[4]

6.1 Arbitrarily Setting Reputation

An interesting setting which should be considered arises when a manipulator is allowed to set its reputation value arbitrarily. As a simple motivating example, consider a situation where agents propagate agent j's average reputation ($x_i = r_i^j$ for all i), and a manipulator i^m wants to ensure that for all i, $\frac{s_{t,i}}{w_{t,i}}$ converges to a high value as the time t increases. At some stage t_0, the manipulator updates s_{t_0,i^m} to be n, but except for this harsh deviation follows the protocol to the letter. In particular, the manipulator might initially set $r_{i^m}^j = x_{i^m} = n$. We refer to this strategy as *Strategy 1*. Clearly, for all i, $\frac{s_{t,i}}{w_{t,i}}$ eventually converges to a value that is at least 1.

Despite the apparent effectiveness of Strategy 1, it is easily detected. Indeed, unless for all $i \neq i^m$ it holds that $s_{t_0,i} = 0$ at the time t_0 when the manipulator deviated by assigning $s_{t_0,i^m} = n$, the expressions $\frac{s_{t,i}}{w_{t,i}}$ would eventually converge to a value that is strictly greater than 1; this would clearly unmask the deceit. It is of course possible to update s_{t_0,i^m} to be less than n, but it is difficult to determine *a priori* which value to set without pushing the average reputation above 1.

6.2 A More Subtle Approach to Manipulating Reputation

We now consider a more subtle way to increase the values $\frac{s_{t,i}}{w_{t,i}}$, a deceit that is indeed difficult to detect; we call this strategy *Strategy 2*. For the first T stages of the algorithm, the manipulator i^m follows PUSH-SUM as usual, with the exception of the updates of s_{t,i^m}: after updating $w_{t,i^m} = \sum_l \hat{w}_l$ (as usual), i^m updates: $s_{t,i^m} = w_{t,i^m}$. In other words, the manipulator sets its personal evaluation of the average $\frac{s_{t,i^m}}{w_{t,i^m}}$ to be 1 at every stage $t = 1, \ldots, T$. For time $t > T$, the manipulator abides by the protocol.

Using this strategy, it always holds that $\frac{s_{t,i}}{w_{t,i}} \leq 1$ for all i. In addition, for all t, it still holds that $\sum_i w_{t,i} = n$. Therefore, without augmenting the system with additional security measures, this manipulation is difficult to detect. We shall presently

[4] In fact, this holds for every coalition that does not constitute a sizable portion of the entire set of agents.

demonstrate formally that the manipulation is effective in the long run: $\frac{s_{t,i}}{w_{t,i}}$ converges to 1 for all i.

Proposition 4. *Under Strategy 2, for all $i \in N$, $\frac{s_{2T,i}}{w_{2T,i}} \overset{T\to\infty}{\longrightarrow} 1$ in probability.*

Proof. We first notice that $\sum_i s_{t,i}$ is monotonically increasing in stage t. Moreover, as noted above, it holds that at every stage, $\sum_i w_{t,i} = n$, as for all $i \in N$: $\frac{s_{t,i}}{w_{t,i}} \leq 1$, and thus:

$$\sum_i s_{t,i} \leq \sum_i w_{t,i} = n.$$

Let $\epsilon, \delta > 0$. We must show that it is possible to choose T large enough such that for all $t \geq 2T$ and all $i \in N$, $\Pr[\frac{s_{t,i}}{w_{t,i}} \geq 1-\epsilon] \geq 1-\delta$.

Assume that at time t it holds that:

$$\frac{\sum_i s_{t,i}}{n} < 1 - \epsilon/2. \tag{1}$$

Let $I_t = \{i \in N : \frac{s_{t,i}}{w_{t,i}} \geq 1-\epsilon/4\}$, $w(I_t) = \sum_{i\in I_t} w_{t,i_t}$. It holds that:

$$\begin{aligned} n(1-\epsilon/2) &\geq \sum_{i\in N} s_{t,i} \\ &\geq \sum_{i\in I_t} s_{t,i} \\ &\geq \sum_{i\in I_t} w_{t,i}\cdot(1-\epsilon/4) \\ &= w(I_t)(1-\epsilon/4). \end{aligned}$$

It follows that $w(I_t) \leq n \cdot \frac{1-\epsilon/2}{1-\epsilon/4}$. The total weight of agents in $N \setminus I_t$ is at least $n - w(I_t)$. There must be an agent $i_t \in N \setminus I_t$ with at least a $1/n$-fraction of this weight:

$$w_{t,i_t} \geq \frac{n - w(I_t)}{n} \geq \frac{\epsilon}{4-\epsilon}. \tag{2}$$

In order for the choice of i_t to be well-defined, assume i_t is the minimal index that satisfies Equation (2).

Now, let s'_{t,i^m} be the manipulator's sum had he updated it according to the protocol, i.e., $s'_{t,i^m} = \sum_l \hat{s}_l$ for all messages l sent to i^m. With probability $1/n$ (and independently of other stages), $f_t(i_t) = i^m$; if this happens, it holds that:

$$\begin{aligned} s'_{t+1,i^m} &\leq (w_{t+1,i^m} - 1/2\cdot w_{t,i_t}) + 1/2\cdot s_{t,i_t} \\ &\leq (w_{t+1,i^m} - 1/2\cdot w_{t,i_t}) \\ &\quad + 1/2\cdot w_{t,i_t}\cdot(1-\epsilon/4). \end{aligned} \tag{3}$$

For all stages t it holds that $\sum_i s_{t+1,i} - \sum_i s_{t,i} = s_{t+1,i^m} - s'_{t+1,i^m}$, as the manipulator is the only agent that might change $\sum_i s_{t,i}$. Therefore, in the conditions of

Equation (3),

$$\begin{aligned}\sum_i s_{t+1,i} - \sum_i s_{t,i} &= s_{t+1,i^m} - s'_{t+1,i^m} \\ &= w_{t+1,i^m} - s_{t+1,i^m} \\ &\geq 1/2 \cdot w_{t,i_t} \cdot \frac{\epsilon}{4} \\ &\geq \frac{\epsilon^2}{32 - 8\epsilon} \\ &= \Delta(w).\end{aligned}$$

So far, we have shown that for each stage t where Equation (1) holds and $f_t(i_t) = i^m$, it is the case that $\sum_i s_{t+1,i} - \sum_i s_{t,i} \geq \Delta(w)$. This can happen at most $\frac{n(1-\epsilon/2)}{\Delta(w)}$ times before Equation (1) no longer holds, or to put it differently, before $\frac{\sum_i s_{t,i}}{n} \geq 1 - \epsilon/2$.

Let X_t be i.i.d. binary random variables, that are 1 iff $f_t(i_t) = i^m$. It holds that for all t where Equation (1) is true, $\mathbb{E}[X_t] = 1/n$. By Chernoff's inequality, it holds that:

$$\Pr[\frac{1}{T_1}\sum_{t=1}^{T_1} X_t \leq \frac{1}{2n}] \leq e^{-\frac{T_1}{2n^2}}.$$

It is possible to choose T_1 to be large enough such that this expression is at most $\delta/2$, and in addition $\frac{1}{2n} \cdot T_1 \geq \frac{n(1-\epsilon/2)}{\Delta(w)}$. Therefore, at time T_1, the average $\frac{\sum_i S_{T_1,i}}{n} \geq 1 - \epsilon/2$ with probability $1 - \delta/2$.

Recall that after T stages (where i^m deviated from the protocol), it still holds that $\sum_i w_{T,i} = n$. Assume that indeed $\frac{\sum_i S_{T_1,i}}{n} \geq 1 - \epsilon/2$. By modifying the proof of Theorem 3.1 from [20], it is possible to show that after another $T_2 = T_2(n, \delta, \epsilon)$ stages where all agents observe the protocol, it holds with probability $1 - \delta/2$ that for all i, $\left|\frac{s_{T_1+T_2,i}}{w_{T_1+T_2,i}} - \frac{\sum_i S_{T_1,i}}{n}\right| < \epsilon/2$, and thus for all i and $t \geq T_1 + T_2$, $\frac{s_{t,i}}{w_{t,i}} > 1 - \epsilon$ with probability $1 - \delta$.

The proof is completed by simply choosing $T = \max\{T_1, T_2\}$. □

6.3 Resistance to Subtle Reputation Manipulation

Proposition 4 implies that Strategy 2 poses a provably acute problem, when PUSH-SUM is run a large number of turns. Fortunately, PUSH-SUM converges exponentially fast, and thus it is usually the case that the manipulator is not able to significantly affect the average reputation, as the following proposition demonstrates.

Proposition 5. *Let $T_1 \leq T$. Under Strategy 2 it holds that* $\mathbb{E}\left[\frac{\sum_i S_{T_1,i}}{n} - \bar{r}^j\right] \leq \frac{T_1}{2n}$.

Proof. Let $\{\hat{s}_l, \hat{w}_l\}$ be the messages that the manipulator received at time $t+1$. The manipulator sets $s_{t+1,i^m} = w_{t+1,i^m} = \sum_l \hat{w}_l$. Essentially, this is equivalent to setting for all l $\hat{s}_l = \hat{w}_l$, or in other words, raising each $\hat{s}_l$ by $\hat{w}_l - \hat{s}_l$. At turn t it was already true that $s_{t,i^m} = w_{t,i^m}$ (w.l.o.g. this is also true for $t = 0$), so it is enough to consider messages at time t from all $i \neq i^m$.

Therefore, for all stages t, it holds that:

$$\begin{aligned}
\mathbb{E}\left[\sum_i s_{t+1,i} - \sum_i s_{t,i}\right] &= \sum_{i \neq i^m} (\Pr[f_t(i) = i^m] \cdot (\frac{1}{2} w_{t,i} - \frac{1}{2} s_{t,i})) \\
&= \frac{1}{2n} \sum_{i \neq i^m} (w_{t,i} - s_{t,i}) \\
&\leq \frac{1}{2n} \sum_{i \neq i^m} w_{t,i} \\
&\leq \frac{1}{2n} \sum_{i \in N} w_{t,i} \\
&= \frac{1}{2}.
\end{aligned}$$

The last equality follows from the fact that for all t, $\sum_i w_{t,i} = n$.

As $\bar{r}^j = \frac{\sum_i s_{0,i}}{n}$, and from the linearity of expectation, we obtain that

$$\begin{aligned}
\mathbb{E}\left[\frac{\sum_i s_{T_1,i}}{n} - \bar{r}^j\right] &= \frac{1}{n}\mathbb{E}\left[\sum_{t=0}^{T_1-1}\left(\sum_i s_{t+1,i} - \sum_i s_{t,i}\right)\right] \\
&= \frac{1}{n}\sum_{t=0}^{T_1-1}\mathbb{E}\left[\sum_i s_{t+1,i} - \sum_i s_{t,i}\right] \\
&\leq \frac{1}{n} T_1 \cdot \frac{1}{2}.
\end{aligned}$$

□

In particular, since $U(n, \delta, \epsilon) = O(\log n + \log \frac{1}{\delta} + \log \frac{1}{\epsilon})$, PUSH-SUM is executed $O(\log n)$ stages, and thus the difference in the average is at most $O(\frac{\log n}{n})$, which is quite insubstantial.

Remark 5. It is not guaranteed at time T_1 that each $\frac{s_{t,i}}{w_{t,i}}$ is close to $\bar{r}^j$, because the inputs were dynamically changed during the execution of PUSH-SUM.

Remark 6. The above discussion focused on a setting where the manipulator attempts to increase the average reputation of an agent. It is likewise possible for a manipulator to decrease an agent's average reputation, or indeed to set that average reputation eventually to any value the manipulator wants.

Remark 7. Jelasity, Montreso and Babaoglu [18] propose a general method to completely prevent malicious agents from deviating in gossip-based algorithms, by augmenting the protocol with exchange of certificates. However, the authors describe their approach in a very general manner, so this approach was not implemented here.

7 Related Work

The main focus of research on trust and reputation systems has been on the semantic aspects of these systems, and their effect on social welfare. Previous work has highlighted the advantages of reputation systems in overcoming social pitfalls in several domains. Akerlof [4], for instance, considered markets where information asymmetry exists between buyers and sellers, in the sense that buyers can only guess the quality of goods; in such a setting, a reputation system can improve social welfare.

Several papers provide detailed background concerning reputation systems [22, 19, 17]. A number of such systems are used in practice. One prominent reputation system is that of eBay; the efficiency of eBay-like reputation systems has been studied in [12].

Other research has analyzed manipulations of general reputation mechanisms. Friedman and Resnick [15] have discussed the effects of *cheap pseudonyms*. When agents can enter the system using pseudonyms, and the cost of recreating an identity is cheap, agents who have a stained reputation may easily shed it. The authors considered several solutions to this problem: disallowing anonymity, entry fees (which make pseudonyms more expensive), and using a central authority for irreplaceable ("once-in-a-lifetime") pseudonyms. However, each approach has major drawbacks. [7] discusses Sybil attacks, where an agent creates many false identities in order to boost the reputation of its primary identity.

Dellarocas [11] studied a setting where agents manipulate a reputation system by providing unfair ratings to some of their peers, and suggests several solutions. A number of other mechanisms can make reputation systems resistant to manipulations; [24] discusses scoring rules that help obtain honest feedbacks, while [5] proposes a method for learning who the good raters are.

7.1 Distributed Reputation Systems

Early research on distributed reputation systems included that of Abdul-Rahman and Hailes [1], which relied on the results of Marsh [23] to design a model of trust in online environments. In this framework, each agent must maintain and update large data structures, which contain knowledge about the entire system. Updating this data may be inefficient, and in particular it is not certain that the scheme scales well when the number of agents grows.

P2PRep [8] and Xrep [10] are P2P reputation systems that can be piggybacked onto existing P2P protocols (such as Gnutella). P2PRep allows peers to estimate trustworthiness of other peers by polling. XRep takes another step forward; each peer keeps trust evaluations both of other peers and of resources. No guarantees are given with respect to computational efficiency and scalability.

Aberer and Despotovic [3] introduce a reputation system that consists of both a semantic model and a data management scheme. The latter relies on P-Grid [2], and uses distributed data structures for storing trust information; the associated algorithms scale gracefully as the number of agents increases. In addition, a limited resistance to manipulation and failure is achieved through replication of data. This approach suffers from several shortcomings compared to ours. Agents in this scheme assess others' reputation only on the basis of complaints filed in the past; the framework is generally limited to such binary trust information. In addition, trust is evaluated only according to referrals from neighbors, whereas in our scheme the evaluation is based on all the information in the system.

Xiong and Liu [27] introduced a sophisticated framework specifically applicable in peer-to-peer networks, where the decision whether to trust a peer is based on five metrics: satisfaction, number of transactions, credibility of feedback, transaction context, and community context. This work was extended in [26]. Both papers concentrate on the trust model, and generally do not elaborate on the data management scheme. Specifically, in [27] a P-Grid [2] is used. Therefore, this work is in a sense orthogonal but complementary to ours. Dewan and Dasgupta [14] propose self-certification and IP-Based safeguards as ways of inducing trust; that work also complements ours.

Finally, gossip-based algorithms[5] have many applications in other domains, for instance replicated database maintenance [13].

8 Conclusions and Future Research

We have presented a data management scheme which is based on gossip-based algorithms, and have demonstrated that it possesses the following features:

- Decentralization: no central database, and further, applicability in networks where point-to-point communication cannot be assumed.
- Scalability: the time to evaluate an agent's average reputation with confidence $1-\delta$ and accuracy ϵ is $O(\log n + \log \frac{1}{\delta} + \log \frac{1}{\epsilon})$.
- Robustness to failure.
- Global perspective: evaluation of trust is based on all relevant information in the system, rather than local information.
- Extremely simple data structures: each agent merely keeps an assessment of the agents with which it personally interacted.
- Motivates absolute truthfulness, as the time to close deals may decrease as reputation increases.
- Resistance to some attacks, such as carefully tampering with the updates performed by PUSH-SUM.

We have focused on the data management scheme, and have largely ignored the trust model (with the exception of Section 4). However, we believe that many existing trust models can be integrated with our framework. A very simple example is the binary trust model of [3], where agents can file complaints against other agents. In our framework, each agent i sets its value r_i^j to be 0 if it wishes to file a complaint against j; otherwise, the value is 1.

More sophisticated models may require tweaks in the framework. Consider the trust model presented in [27], where five factors are taken into account. Three of the

[5] They are also called *epidemic algorithms*.

factors mentioned simply determine the way an agent updates its own values r_j^i, and our framework of course supports any update formula. The "number of transactions" factor is already taken into account, as we compute the average reputation. The "credibility of feedback" factor requires a small change: given credibility ratings c_i for agents, a weighted average can be computed; the initial inputs of agents are $x_i = s_{0,i} = c_i \cdot r_i^j$, and their weights are $w_{0,i} = c_i$.

An interesting direction for future research is augmenting the framework with an option to efficiently choose among *service providers*:[6] agent i requires a specific service, and there are m other agents that offer to respond to i's request. Agents can be matched to service providers using a *matchmaking* service, but that problem has been handled [9]. We are concerned with the following question: once an agent is given a list M of m-service providers, which one should it choose? The obvious answer is, the one with the highest reputation: $\text{argmax}_{j \in M} \bar{r}^j$. However, as the size of M may approach n, it is difficult to estimate the reputation of all agents in M.

One possible solution is to hold an election: the voters are the n agents, and the candidates are the m service providers. It is possible, for instance, to determine the winner using the simple *plurality* rule: each agent votes for one candidate; the winner is the candidate that secured the largest number of votes. It is also possible to resolve this election in our framework, using PUSH-SUM. Denote $M = \{j_1, j_2, \ldots, j_m\}$. The input x_i of each agent for PUSH-SUM is (conceptually) a base $n+1$ number with m coordinates; the l'th coordinate of x_i is n if i votes for j_l, and 0 otherwise. The average is calculated by using PUSH-SUM with an *absolute* error $\epsilon = 1/3$, and some confidence $1 - \delta > 0$ (the x_i are translated to base 10). After the average $\frac{s_{t,i}}{w_{t,i}}$ is calculated, the result is rounded to the nearest integer and again translated to base $n+1$; the agent j_l such that the $l'th$ coordinate is largest wins the election.

Unfortunately, since in this case the inputs x_i are large, obtaining such a small absolute error requires a large number of iterations of PUSH-SUM, and furthermore, the message size is large. Is there a way of holding an election (using some other voting rule, perhaps) in our framework in a way that scales well with respect to both the running time and message size?

A different direction is using our framework to prevent attacks based on cheap pseudonyms [15]. This is made possible due to the fast aggregation of information and its global nature. If an agent cheats, *all* other agents will soon know. It is possible to restrict newcomers, with an unestablished reputation, to only one transaction in every period of length $O(\log n)$. This way, each identity would be good for only a single deceitful transaction, since after the period is over, the information could have already been obtained by any agent. Granted, it would still be possible to shed stained identities, but the flow of transactions a cheater would be able to complete would be severely diminished.

[6] In a sense, similar to [8].

9 Acknowledgments

This work was partially supported by Israel Science Foundation grant #898/05. Ariel Procaccia is supported by the Adams Fellowship Program of the Israel Academy of Sciences and Humanities.

References

[1] A. Abdul-Rahman and S. Hailes. Supporting trust in virtual communities. In *Proceedings of the 33rd Hawaii International Conference on System Sciences*, 2000.

[2] K. Aberer. P-grid: A self-organizing access structure for P2P information systems. In *Proceedings of the 9th International Conference on Cooperative Information Systems*, pages 179–194, 2001.

[3] K. Aberer and Z. Despotovic. Managing trust in a peer-2-peer information system. In *Proceedings of the Tenth International Conference on Information and Knowledge Management*, pages 310–317, 2001.

[4] G. A. Akerlof. The market for lemons: qualitative uncertainty and the market mechanism. *The Quarterly Journal of Economics*, 84:488–500, 1970.

[5] Awerbuch and Kleinberg. Competitive collaborative learning. In *COLT: Proceedings of the Workshop on Computational Learning Theory, Morgan Kaufmann Publishers*, 2005.

[6] R. Axelrod. *The Evolution of Cooperation*. Basic Books, 1984.

[7] A. Cheng and E. Friedman. Sybilproof reputation mechanisms. In *P2PECON '05: Proceeding of the 2005 ACM SIGCOMM Workshop on Economics of Peer-to-Peer Systems*, pages 128–132, New York, NY, USA, 2005. ACM.

[8] F. Cornelli, E. Damiani, S. De Capitani di Vimercati, S. Paraboschi, and P. Samarati. Choosing reputable servants in a P2P network. In *Proceedings of the 11th International Conference on the World Wide Web*, pages 376–386, 2002.

[9] G. Cybenko and G. Jiang. Matching conflicts: Functional validation of agents. In *Proceedings of the AAAI Workshop on Agent Conflicts*, pages 14–19, 1999.

[10] E. Damiani, S. De Capitani di Vimercati, S. Paraboschi, P. Samarati, and F. Violante. A reputation-based approach for choosing reliable resources in peer-to-peer networks. In *Proceedings of the 9th ACM Conference on Computer and Communications Security*, pages 207–216, 2002.

[11] C. Dellarocas. Immunizing online reputation reporting systems against unfair ratings and discriminatory behavior. In *Proceedings of the 2nd ACM Conference on Electronic Commerce*, pages 150–157, 2000.

[12] C. Dellarocas. Analyzing the economic efficiency of eBay-like online reputation mechanisms. In *ACM Conference on Electronic Commerce (EC-01)*, Tampa, Florida, 2001.

[13] A. J. Demers, D. H. Greene, C. Hauser, Wes Irish, and John Larson. Epidemic algorithms for replicated database maintenance. In *Proceedings of the Sixth Annual ACM Symposium on Principles of Distributed Computing*, pages 1–12, 1987.

[14] Prashant Dewan. Peer-to-peer reputations. In *Proceedings of the 18th International Parallel and Distributed Processing Symposium*, 2004.

[15] E. J. Friedman and P. Resnick. The social cost of cheap pseudonyms. *Journal of Economics and Management Strategy*, 10(2):173–199, 2001.

[16] A. M. Frieze and G. R. Grimmett. The shortest-path problem for graphs with random arc-lengths. *Discrete Applied Mathematics*, 10:57–77, 1985.

[17] D. Fudenberg and D. K. Levine. Reputation and equilibrium selection in games with a patient player. Levine's Working Paper Archive 103, UCLA Department of Economics, January 1995. Available at http://ideas.repec.org/p/cla/levarc/103.html.

[18] M. Jelasity, A. Montresor, and O. Babaoglu. Towards secure epidemics: Detection and removal of malicious peers in epidemic-style protocols. Technical Report UBLCS-2003-14, Department of Computer Science, University of Bologna, 2003.

[19] M. Kandori. Social norms and community enforcement. *The Review of Economic Studies*, pages 63–80, January 1992.

[20] D. Kempe, A. Dobra, and J. Gehrke. Gossip-based computation of aggregate information. In *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science*, pages 482–491, 2003.

[21] P. Keyani, B. Larson, and M. Senthil. Peer pressure: Distributed recovery from attacks in peer-to-peer systems. In *Proceedings of the International Workshop on Peer-to-Peer Computing*, pages 306–320, 2002.

[22] D. M. Kreps and R. Wilson. Reputation and imperfect information. *Journal of Economic Theory*, pages 305–15, August 1982.

[23] S. P. Marsh. *Formalising Trust as a Computational Concept*. PhD thesis, University of Stirling, Apr 1994.

[24] N., P. Resnick, and R. J. Zeckhauser. Eliciting honest feedback in electronic markets. KSG Working Paper Series RWP02-039, 2003.

[25] G. Pandurangan, P. Raghavan, and E. Upfal. Building low diameter P2P networks. In *Proceedings of the 42nd Annual IEEE Symposium on Foundations of Computer Science*, pages 492–499, 2001.

[26] M. Srivatsa, L. Xiong, and L. Liu. Trustguard: Countering vulnerabilities in reputation management for decentralized overlay networks. In *Proceedings of the 14th International World Wide Web Conference*, pages 422–431, 2005.

[27] L. Xiong and L. Liu. A reputation-based trust model for peer-to-peer ecommerce communities. In *Proceedings of the 4th ACM Conference on Electronic Commerce*, pages 228–229, 2003.