DYNAMIC SELF-MANAGEMENT OF AUTONOMIC SYSTEMS: THE REPUTATION, QUALITY AND CREDIBILITY (RQC) SCHEME

Anurag Garg, Roberto Battiti and Gianni Costanzi

2004

Technical Report # DIT-04-082

Dynamic Self-management of Autonomic Systems: The Reputation, Quality and Credibility (RQC) Scheme

Anurag Garg   Roberto Battiti   Gianni Costanzi

Dipartimento di Informatica e Telecomunicazioni,
Università di Trento,
Via Sommarive 14, 38050 Povo (TN), Italy.
{garo,battiti}@dit.unitn.it

Abstract. In this paper, we present a feedback-based system for managing trust and detecting malicious behavior in autonomically behaving networks. The two crucial insights that motivate our work are the notion of quality of a trust rating and the recognition as separate entities of the trust placed in a node and of the trust placed in the recommendations made by a node. These variables allow nodes to decide how much confidence they can place in the rating. We implement our scheme on a structured P2P network, Pastry, though our insights can be extended to unstructured systems and generic autonomic communication systems as well. Experimental results considering different models for malicious behavior indicate the contexts in which the RQC scheme has a better performance as compared to existing schemes in the literature.

Keywords: Trust management, reputation, structured overlays, peer-to-peer systems.

1 Introduction

Autonomic systems aim at incorporating methods to monitor their dynamic behavior and to react in automated ways, aiming at self-management and self-improvement. This work explores how ideas of automated reputation schemes originated in other contexts (e.g. e-commerce) can be modified and applied for the soft enforcement of rules of behavior in specific autonomic systems based on distributed control.

In particular, this work proposes a novel scheme to self-manage reputation values and presents experiments in the area of peer-to-peer (P2P) systems. It is known that in certain application areas P2P systems have benefits over the more traditional client-server architectures. With no centralized administration and no central database or server, there is no single point of failure. The peers themselves are autonomous and can join and leave the system at will without adversely affecting service availability. Therefore, the design of such systems requires them to be self-organizing, self-managing, self-healing and self-aware.

All autonomic systems implicitly place a certain trust in participants, assuming that they will follow certain guidelines for "fair-use". But due to the distributed nature of such systems systems, breaking these guidelines can go undetected by the system as a whole and not result in any significant penalties to misbehaving users. This can result in poorer quality of service and, in the worst case, outright service denial.

Examples of such “unfair” use range from the relatively benign “free-riders” that utilize the facilities offered by the network (such as download of files, use of processing power etc.) without contributing back to the network in reasonable measure such as sharing their own files, processing power or disk space, or forwarding routing requests from other peers, to more serious abuses such as distributing viruses, intentionally sharing inauthentic or malformed files that cause an unnecessary increase in network traffic, giving incorrect information in response to queries and manipulating trust management systems by giving false trust ratings for a peer.

Since P2P systems are usually open and anonymous, imposing barriers for entry into the system is not always an acceptable solution. Instead, several reputation-based trust management systems have been proposed. These systems rely on the dissemination, throughout the network, of trust information gathered through transactions between peers. In this way peers can build knowledge about peers with whom they have never interacted before and use this information to decide whether to interact with new peers. But relying on information from third parties also makes the system vulnerable to manipulation through false complaints or false praise.

We present a reputation-based system that is robust against false ratings and at the same time allows “good” nodes to avoid interacting with “malicious” nodes a large proportion of the time. We simulate our trust management scheme on Pastry [1], a structured overlay network that uses distributed hash tables for routing. Using Pastry, we assign one or more Score Managers (SM) to each node and all transaction involving that node are reported to each of these score managers. The score managers aggregate trust information and respond to rating requests by nodes wishing to transact with a node for whose trust ratings they are responsible.

Our system provides nodes with a mechanism to specify a quality value for trust ratings they submit to score managers, usually based on the number, variation and the scope of the interactions they have had with the node they are rating. Similarly, score managers also attach a quality value to the ratings they provide to other nodes. This allows nodes to set their own standards of rating quality which must be met for an interaction to happen.

We also distinguish between the reputation of a node and its credibility as a reporter of other nodes’ ratings. For example, consider a decentralized system for ranking restaurants. It is possible that a restaurant may serve very good food but it falsely rates other restaurants as bad to prevent prospective clients from going there. Therefore this restaurant should have a high reputation but low credibility.

The paper therefore contributes to the understanding of how trust information should be aggregated and how much credence should be attached to it by other nodes in the network. Together, the three variables Reputation (R), Quality (Q) and Credibility (C) provide a richer and more robust trust management system and we call our scheme the **RQC** scheme.

The rest of this paper is organized as follows. In the next section we review existing work on trust management in P2P systems. In Sec. 3 we present our solution. In Sec. 4 we present our experimental results and we conclude in Sec. 5.

2 Related Work

Initial efforts at trust management in electronic communities were based on centralized trust databases. The eBay rating system used for choosing trading partners where each participant in a transaction can vote $(-1, 0, 1)$ on their counterpart, the Amazon customer review system and the Slashdot self-moderation of posts [2] are all systems where the ratings are provided by nodes but are stored in a central database. Many such reputation systems have been studied in the context of online communities and marketplaces [3–5].

In true P2P environments, the storage of trust ratings also needs to be done in a distributed fashion. Aberer and Despotovic introduced such a scheme [6] using a decentralized storage system P-Grid to store and retrieve trust information. Peers can file complaints against each other if they feel the node has behaved maliciously. All interactions pertaining to a given node, i.e. complaints made by and complaints about that node, are stored at other nodes called agents. The mechanism is made robust by keeping copies of reports at several agents and by making sure that agents do not know the identity of the node whose information is being stored. A node then sends messages querying trustworthiness of the node with which it wishes to interact to random nodes in the network. These are then routed to appropriate agents. Two algorithms are described to compute the trustworthiness. The first relies on a simple majority of the reporting agents' decisions and the second checks the trustworthiness of the reporting agents themselves and disallows any reports coming from untrustworthy agents.

Cornelli, Damiani et. al. [7, 8] propose a mechanism built on the Gnutella network, where a node uses a secure polling mechanism to ask its neighbors about interactions they may have had with a specific node to gauge it's trustworthiness. The scope of the messages querying trust is limited by the Gnutella architecture design. Their work is directed at file-sharing networks and the objective is to find the most trusted peer that possesses a given resource and they focus on vote aggregation, incorporating voter credibility and on ensuring the integrity of trust reports as they pass over the insecure network.

Kamvar et. al. [9] use a different approach in that trust is assumed to be transitive. Therefore, a node weighs the trust ratings it receives from other nodes by the trust it places in the reporting nodes themselves. Global trust values are then computed in a distributed fashion using a trust matrix at each node with successive iterations involving exchange of trust values with neighbors and recomputation of the matrix. Trust values asymptotically approach the eigenvalue of the trust matrix, conditional on the presence of pre-trusted peers that are always trusted regardless of performance. Their objective is to reduce the number of inauthentic file downloads and therefore free-riding is in fact rewarded.

Buchegger et. al. [10] propose a modified Bayesian approach to trust. Like Damiani et. al. they separate a node's reputation (it's performance in the base system such as file-sharing, routing etc.) and it's credibility (it's performance in the reputation system). In their solution only information on first-hand experiences is published by nodes. This is used by nodes to construct their reputation and credibility (called trust by them) data structures for other nodes. They also age reputation data giving less weight to evidence received in the past.

3 The RQC (Reputation, Quality, and Credibility) Approach

In our approach we assume existence of a distributed routing mechanism that consistently maps an identifier to a logical key space. For our experiments, we use the Pastry P2P substrate that implements a structured overlay network using distributed hash tables (DHT). While such a mechanism is not essential, it greatly increases the robustness of our system against false reports and against certain kinds of attacks.

The trust information pertaining to a node i is stored at m score managers that are assigned using a DHT. A hashing function is used to map a persistent node identifier to a point in the key space. The m nodes that are closest to this point are then used as score managers for that node. A node j can then query all the m score managers in order to compute the reputation of i and decide whether to interact with i.

3.1 Node Opinions

Each node i maintains an opinion $O(i, j)$ of the behavior of all other nodes j with which it has interacted in a local data structure. $O(i, j)$ is node i's estimate, through fist-hand experience, of the probability that a node j will behave honestly in the base system and can take any value ranging from 0 to 1.

Along with the opinion on behavior, a node also stores the number of interactions $N_{i,j}$ it has had with a given node as well as the variance $\sigma^2_{i,j}$ in opinion over the entire history of interactions. These two values act as an indicator of the quality a node attaches to its opinion of a given node. This allows us to capture the notion that an opinion is of greater quality when the number of observations on which it is based is larger and when the interactions have been consistent (indicated by a smaller variance). When the number of observations is high but they do not agree with each other, the quality value is lower. Thus redundancy is inferred from consensus which in turn provides us with a higher level of confidence.

The quality value of an opinion is computed by using the Student's t distribution. We assume that a node behavior is normally distributed with some actual trust rating as the mean and some unknown variance in the node's behavior. The observations made through a node's interactions are then a random sample drawn from the behavior distribution and we only know the sample mean and sample variance. In reality, since the node's rating is restricted to the range $[0, 1]$, we make appropriate modifications in our calculations to take this into account. The quality value indicates the probability that the actual mean of the node's behavior lies within $k\%$ of the observed mean. The t-value is given by:

$$t = \frac{k * \bar{x} * \sqrt{n}}{100 * s}$$

where $\bar{x}$ is the mean observed rating, s is the sample standard deviation and n is the number of interactions node i has had with node j. The resulting t is used to compute the quality value such that

$$Q(i, j) = P((1 - \frac{k}{100})\bar{x} < \mu < (1 + \frac{k}{100})\bar{x})$$

After each interaction, both participating nodes report their updated opinions to the score managers responsible for their counterpart. Along with the opinion, the node also sends the associated quality value. The inclusion of quality indicator in the message sent to the score manager (SM) allows the SM to gauge the how much confidence the node itself places in the rating it has sent.

3.2 Score Managers

A score manager (SM) receives trust opinions from nodes that interact with all of the nodes for which it is responsible. The individual opinions are then aggregated to form the *reputation* of a node. The score managers therefore represent the global view of a given node's behavior.

On average, each SM stores the ratings for m nodes since there are m SMs per node. A SM uses these opinions to compute a global reputation rating for each node it is responsible for.

Like an ordinary node a SM also computes and stores an associated quality value for each of these reputations. However, unlike individual nodes that compute quality values using the number and variance of interactions, a SM computes quality taking into account number of reporting nodes, the quality values associated with the opinions and the variance in the reported opinions.

A SM uses the quality value sent by a reporting node as well as the credibility of the reporting node to compute the average reputation of a node.

$$R(SM, j) = \frac{\sum_i C(SM, i) * O(i, j) * Q(i, j)}{\sum_i C(SM, i) * Q(i, j)}$$

where $R(SM, j)$ is the aggregated reputation of node j according to the SM, $C(SM, i)$ is the credibility of node i according to the SM, $O(i, j)$ is the opinion about j reported by i and $Q(i, j)$ is the associated quality value. This allows a SM to give more weight to ratings that are considered high quality and that come from nodes whose judgment is more credible.

The credibility value indicates the trust the SM has in the judgment and/or honesty of the reporting node as a player in the reputation system. The SM stores the credibility rating for each node that sends it an opinion. The credibility rating is updated every time a node reports an opinion. In addition, when the SM updates the credibility of a node it uses the quality value furnished by the node to decide the amount of modification in the trust value. This is because a node should not be penalized for an incorrect opinion that was based on a small number of interactions where this was explicitly mentioned through a low quality rating.

When a node reports an opinion to a SM for the first time, it's credibility is set to 0.5. Thereafter, every time it reports an opinion on any of the nodes the score manager is responsible for, it's credibility is adjusted according to the following formula:

$$C(SM, i) = C(SM, i) + \frac{(1 - C(SM, i)) * Q(i, j)}{2} \text{ if } |R(SM, j) - O(i, j)| < \sigma_{SM,j}$$

and

$$C(SM,i) = C(SM,i) - C(SM,i) * Q(i,j)2 \text{ if } |R(SM,j) - O(i,j)| > \sigma_{SM,j}$$

where $\sigma(SM, j)$ is the standard deviation in the reported opinions about node j. In this way the credibility of a node remains between 0 and 1 and indicates the level of agreement of the nodes opinions with the aggregated consensus. A node with a lower credibility value therefore contributes less to the aggregated reputation at the SM than a node with high credibility.

3.3 Retrieval of Trust Information

When a node wishes to learn the reputation of another node (e.g. to interact with another node in the system that it has never interacted with), it locates the SMs for the node using the DHT substrate and asks them for the reputation of the node in question. Each SM responds with a reputation value and an associated quality value. The node then computes the average reputation for the node in question using the quality values and the credibility values of the SMs themselves since a SM may also be malicious and send the wrong reputation values.

As in the case of the reputation aggregation at the SMs, the node aggregates the responses and then updates the credibility values of the responding SMs. In this way, if a SM is malicious, it's credibility rating is gradually reduced when it's opinion does not match that of other SMs.

3.4 Node Classification

As compared to other trust management systems our approach gives a node more information on which to base its classification of another node as malicious or not. A querying node receives both the reputation values and the associated quality values from the respondent. While we outline a method in the previous section where reputation is aggregated using a weighted average of the reported reputations, it is possible to implement other strategies as well. For instance, SMs may decide to ignore opinions with associated quality values below a certain threshold or from nodes with credibility below a certain threshold.

Similarly, if a node wants stricter guarantees of another node's good intent, it can insist on a high reputation as well as a high associated quality value implying consensus among other nodes that the node in question indeed has a high reputation.

4 Experimental Results

As we mentioned in the introduction, we use the Pastry substrate to implement and test our reputation algorithm. We assume that the nodes are always online and full connectivity in the network. We also assume that no messages are dropped and that messages cannot be spoofed or altered in any way. Since Pastry is written in Java, we decided to implement our reputation system in Java as well.

In the experiments we describe, we simulated a network of 1000 nodes unless specified otherwise. In each experiment, 50000 random interactions were performed unless specified otherwise(i.e. each node has an average of 50 interactions). Both the participants of each interaction are chosen randomly. The default number of score managers storing reputation ratings for each node was 6 unless specified otherwise.

We simulate two different kinds of maliciousness. A node can be malicious in the base system, i.e., behave maliciously when interacting with other nodes and/or it may be malicious in the reputation system. In the latter case, the node behaves maliciously in its capacity as a SM and sends incorrect reputation values to requesting nodes. A node can therefore act maliciously in its capacity as a peer, as a SM or both.

We also simulate probabilistic maliciousness where a node does not act maliciously all the time and is malicious with a probability p_m which is a parameter to the simulation. Finally, we simulate several scenarios where the number of malicious nodes ranges from 5% of the total population to 90% of the total population.

We now evaluate the performance of our system for a variety of threat models. The metric of interest is the number of correct decisions made (i.e., interactions with good peers that went ahead plus interactions with malicious peers that were avoided) as a proportion of the total number of decisions made. So, an interaction with a malicious peer counts against the RQC scheme as much as when an interaction with a good node is prevented due to false ratings. Since, we are interested in the steady state performance of the system, the initial interactions that take place when the reputations of nodes have not been built yet are not counted. In our experiments with 50000 total interactions we found that this number ranged from 500 to 600. So after about 600 interactions, enough reputation information was generated for good peers to make decisions about interacting with other peers.

4.1 Maliciousness in Base System

In this experiment we examine how the fraction of malicious peers in the network affects the percentage of correct decisions. Maliciousness in this experiment is defined as malicious behavior during interactions with other peers. So if a good and a malicious peer interact they both give each other an opinion of 0. Two good peers or two malicious peers will give each other a value of 1 after interacting.

Figure 1 depicts the results. As we can see, our scheme performs very well until the percentage of malicious peers approaches 40%. At this point the dominant ethic of the system becomes that of the malicious peers and the the good peers are shut out of the system. It is worth noting that this switchover in the dominant ethic happens when malicious peers form 40% of the total peers and not 50%. We believe this is because we count interactions initiated by malicious peers as well and include interactions between two malicious peers in our count of decisions. As a result the percentage of correct decisions is lower than expected.

We compared the performance of our scheme against our implementation of the trust management scheme proposed by Aberer and Despotovic in [6]. We implemented their scheme in Pastry and ran experiments with the same number of nodes and interactions as in our scheme (1000 and 50000 respectively). However, unlike the simulations they performed, we did not make trust assessments at the end of the interaction period but instead made trust assessments before each interaction. We feel this model is closer to reality as nodes would want to know the nature of a node before interacting with it instead of waiting for a requisite number of interactions to finish. As we can see in Fig. 1, our scheme performs much better than the Aberer-Despotovic scheme in terms of the proportion of correct decisions made when the proportion of malicious nodes in the population is under 35%.

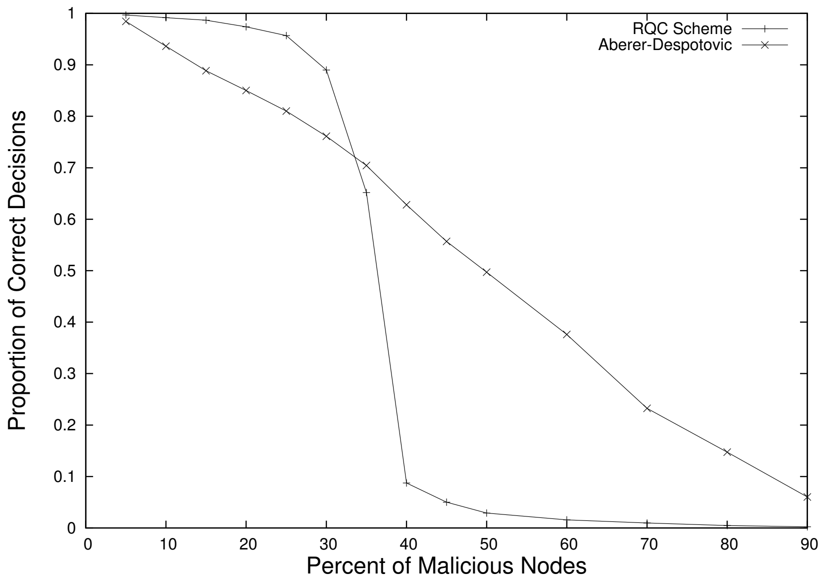

Fig. 1. Comparison of the RQC Scheme with the Aberer-Despotovic Scheme

4.2 Maliciousness in the Reputation System

In Fig. 2, we depict the performance of our scheme for different kinds of malicious behavior. The three lines represent the cases when malicious nodes behave maliciously as participants in the base system (i.e., when interacting with other nodes), as participants of the reputation system (i.e., sending false reputations in their capacity as score managers) and as participants in both the base and reputation system.

Since all nodes report opinions to and receive reputations from score managers, we define malicious behavior in reputation system only in the context of score managers. A malicious score manager sends a reputation value that is the inverse of the actual (1 minus the actual) reputation value of the nodes it is responsible for. We assume that malicious nodes are not aware of each others existence and therefore malicious score managers do not treat other malicious nodes any different from other nodes.

The most striking feature of Fig. 2 is that the performance of our scheme goes down and then rises again in both the cases where nodes act maliciously in the reputation system. This is because with malicious score managers, a large number of interactions good nodes are avoided contributing to poorer performance. However, as the proportion of good nodes decreases in the network, an ever increasing number of interactions that are avoided are with malicious nodes leading to a higher performance value. As an illustration, when the fraction of nodes behaving maliciously in the reputation system is 90%, only 103 of the 50000 interactions are executed and the rest are all avoided. And since 90% of these avoided interactions were with malicious nodes, the performance of the RQC scheme is almost 90%.

When the nodes act maliciously both as participants in the base system and in the reputation system a similar pattern is observed with the worst performance coming when the fraction of malicious nodes is 0.25. This is because as the number of malicious nodes increases, a larger proportion of interactions take place between two malicious nodes. These nodes give each other an opinion rating of 1 but when this opinion is reported to the score manager – which itself has a high likelihood of being malicious – it inverts this rating and the reputation of the malicious nodes is correctly reduced to 0. Therefore, a large number of interactions with malicious nodes are avoided, thus improving the performance of our system.

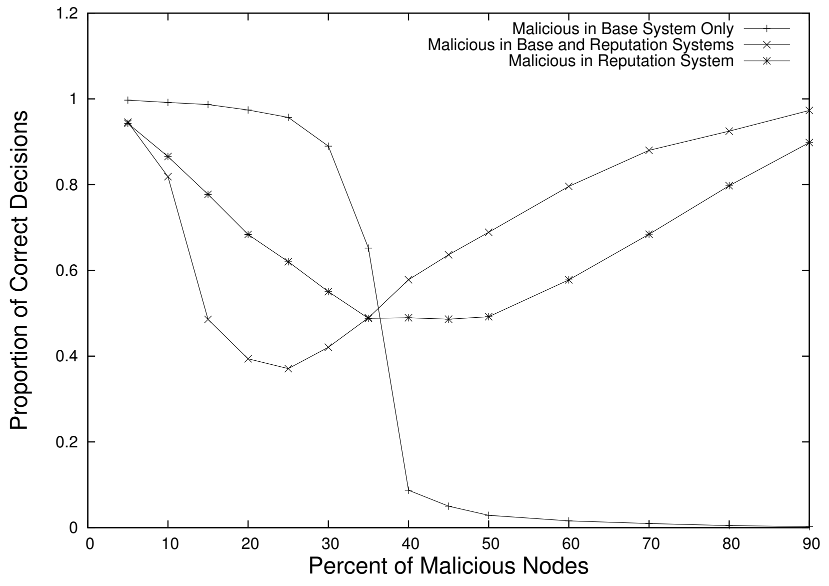

Fig. 2. Performance of the RQC Scheme with Different types of Malicious Nodes

4.3 Probabilistic Cheating

In Fig. 3 we examine the case when the malicious nodes do not cheat all the time but instead cheat with a certain probability. In this experiment, 10% of the nodes are

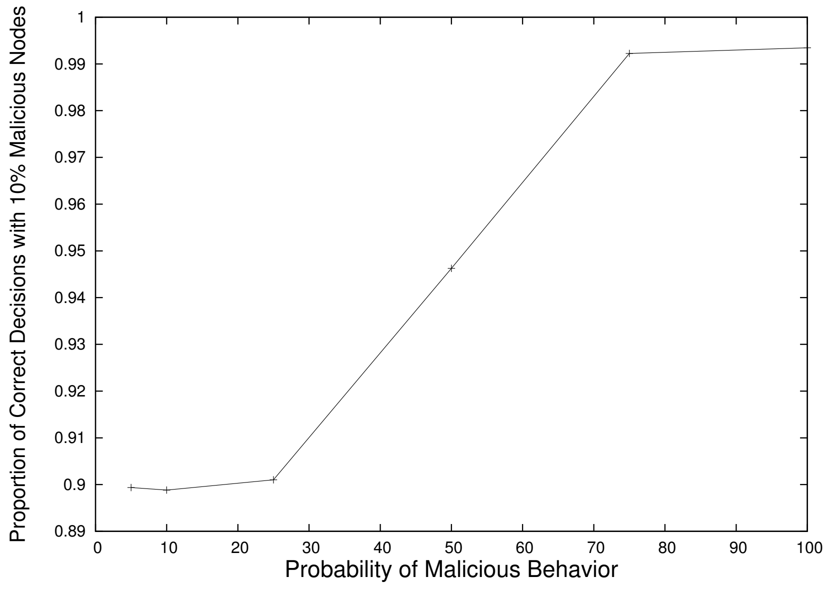

Fig. 3. Performance of the RQC Scheme with Probabilistic Cheating

malicious. We see that the proportion of correct decisions is not affected very much by the probabilistic cheat patterns of malicious nodes and the proportion of correct decisions ranges between 90% and 100%.

4.4 Number of Score Managers

In this experiment we study the impact that the number of score managers have on the decision making process. Figure 4 shows the performance of the RQC scheme when 30% of the nodes are malicious and they act maliciously both in the reputation and the base system. Surprisingly, the number of score managers makes no difference to the performance. This is perhaps due to the fact that the score managers themselves are selected from the population at random and the proportion of score managers that are malicious remains the same regardless of their actual number.

4.5 Number of Nodes

Figure 5 shows how the RQC scheme scales as the number of nodes in the network increases. We ran our simulation with the fraction of malicious nodes at 10%, acting maliciously only in the base system. We ran the experiment with 100, 200, 500, 1000 and 2000 nodes. In each case, the total number of interaction in the experiment was 50 times the total number of nodes.

We find that the scheme scales well and the performance of the RQC scheme remains more or less constant with the fraction of correct decisions being 0.99 as the number of nodes in the system increases.

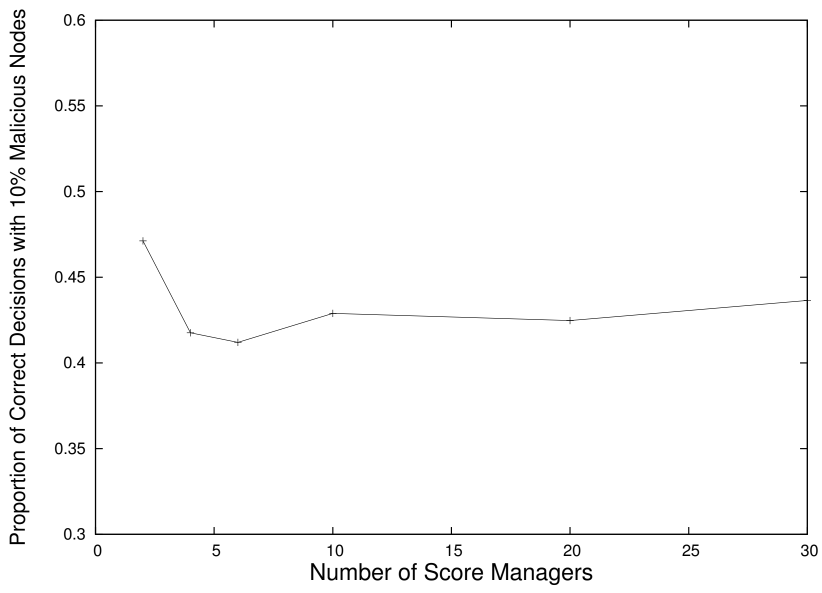

Fig. 4. Performance with Increasing Number of Score Managers

5 Conclusions

In this paper, we have presented the **RQC** scheme for trust management in autonomic systems. Our scheme computes node Reputations(R), Quality(Q) of rating and node Credibility(C) to provide a richer trust management system that lends itself to a wide variety of self-management tasks.

We simulate the RQC scheme using Pastry, a P2P substrate written in Java. However, the RQC scheme can be easily adapted to other environments. The RQC scheme is flexible enough to allow for trust ratings other than 0 or 1. Moreover, the scheme emphasizes consensus as an important indicator of the confidence that can be placed in a rating. This along with the number of interactions forms the basis of the quality of a rating.

The credibility of a node is dependent on the amount by which it's opinion of a node (or the reputation it furnishes in case of a score manager) deviates from the mean reputation of the node in question. In [10] the credibility of a node is lowered if it's opinion deviates from the mean reputation by more than a constant *d*. The RQC scheme lowers credibility if the opinion deviates from the mean reputation by more than the sample standard deviation. This does not penalize nodes by lowering their credibility when they report on a node that behaves erratically. The credibility is also predicated on the quality value of the opinion/reputation furnished by a node/score manager. A node should not be penalized as much for an opinion in which it does not have very much confidence.

Our simulation shows that the RQC method performs very well when the number of malicious nodes in the system is under 25%. It significantly outperforms our implemen-

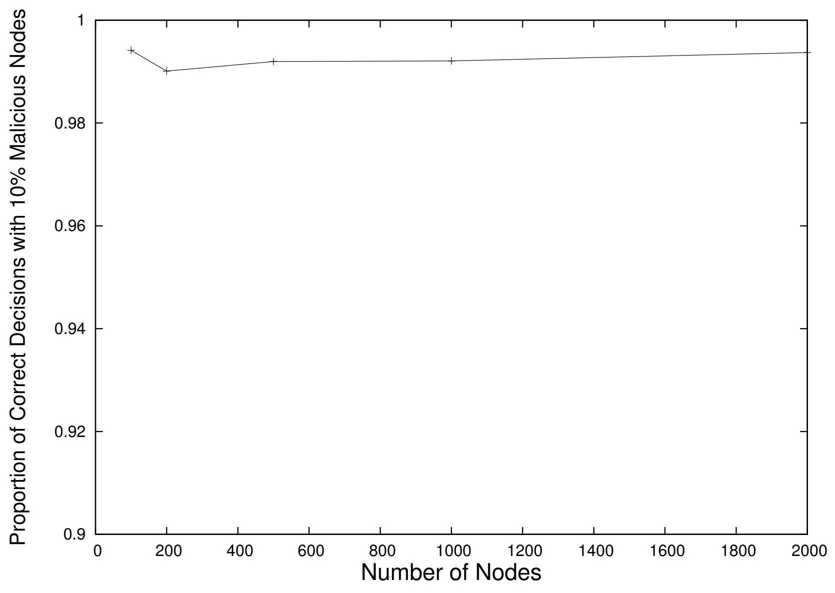

Fig. 5. Performance scaling with Number of Nodes in System

tation of the Aberer-Despotovic scheme. The scheme continues to work well when the malicious nodes cheat in a probabilistic fashion instead of cheating all the time. Finally, the simulation also shows that the scheme scales well with the number of nodes.

References

1. Rowstron, A., Druschel, P.: Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In: IFIP/ACM International Conference on Distributed Systems Platforms (Middleware). (2001) 329–350
2. Lampe, C., Resnick, P.: Slash(dot) and burn: distributed moderation in a large online conversation space. In: Proceedings of the 2004 conference on Human factors in computing systems, ACM Press (2004) 543–550
3. Resnick, P., Zeckhauser, R., Swanson, J., Lockwood, K.: The value of reputation on ebay: A controlled experiment (2002)
4. Dellarocas, C.: Immunizing online reputation reporting systems against unfair ratings and discriminatory behavior. In: Proceedings of the 2nd ACM conference on Electronic commerce, ACM Press (2000) 150–157
5. Zacharia, G., Moukas, A., Maes, P.: Collaborative reputation mechanisms in electronic marketplaces. In: Proceedings of the Thirty-second Annual Hawaii International Conference on System Sciences-Volume 8, IEEE Computer Society (1999) 8026
6. Aberer, K., Despotovic, Z.: Managing trust in a peer-2-peer information system. In: CIKM. (2001) 310–317
7. Cornelli, F., Damiani, E., di Vimercati, S.D.C., Paraboschi, S., Samarati, P.: Choosing reputable servents in a p2p network. In: Eleventh International World Wide Web Conference, Honolulu, Hawaii (2002)

8. Damiani, E., di Vimercati, S.D.C., Paraboschi, S., Samarati, P.: Managing and sharing servents' reputations in p2p systems. IEEE Transactions on Data and Knowledge Engineering **15** (2003) 840–854
9. Kamvar, S.D., Schlosser, M.T., Garcia-Molina, H.: The eigentrust algorithm for reputation management in p2p networks. In: Proceedings of the twelfth international conference on World Wide Web, ACM Press (2003) 640–651
10. Buchegger, S., Boudec, J.Y.L.: A robust reputation system for p2p and mobile ad-hoc networks. In: Proceedings of the Second Workshop on the Economics of Peer-to-Peer Systems. (2004)