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
gianni.costanzi@studenti.unitn.it

Abstract. In this paper, we present a feedback-based system for managing trust and detecting malicious behavior in autonomically behaving networks. Like other distributed trust management systems, nodes rate the interactions they have with other nodes and this information is stored in a distributed fashion.
Two crucial insights motivate our work. We recognize as separate entities the trust placed in a node, *reputation*, and the trust placed in the recommendations made by a node, *credibility*. We also introduce the concept of quality of a trust rating. Together, these two factors enhance the ability of each node to decide how much confidence it can place in a rating provided to it by a third party.
We implement our scheme on a structured P2P network, Pastry, though our results can be extended to generic autonomic communication systems. Experimental results considering different models for malicious behavior indicate the contexts in which the RQC scheme performs better than existing schemes.

Keywords: Trust management, reputation, quality, credibility, autonomic systems, peer-to-peer systems.

1 Introduction

Autonomic systems aim at incorporating methods to monitor their dynamic behavior and to react in automated ways, leading to self-awareness, self-management, self-healing and self-improvement. This work explores how ideas of automated reputation schemes that have originated in other contexts (e.g. e-commerce) can be modified and applied for the soft enforcement of rules of behavior in specific autonomic communication systems based on distributed control. In particular, this work proposes a novel scheme to self-manage trust in autonomic systems using Peer-to-Peer (P2P) environments as an example setting.

Many 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, breaking these guidelines can go undetected by the system as a whole

* This work was supported by the Provincia Autonoma di Trento through the WILMA Project.

and does not result in any significant penalties to misbehaving components. This can result in poorer quality of service and, in the worst case, outright service denial.

For any system that is open and anonymous, imposing barriers for entry is not an acceptable solution. A reputation-based trust management system offers a better solution. These systems rely on the dissemination, throughout the network, of trust information gathered through transactions between nodes. In this way nodes can build knowledge about nodes with whom they have never interacted before and use this information to decide whether to interact with new nodes. 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 helps "good" nodes to avoid interacting with "malicious" nodes. Along with node Reputation (R), we use Quality (Q) and Credibility (C) to provide a richer and more robust trust management system called the **RQC** system. In this way, the paper contributes to the understanding of how trust information should be aggregated and how much credence should be attached to reported trust values by other nodes in the network.

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 [1] 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 [2–4].

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 [5] 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 complaints made by and complaints about a given node, are stored at other nodes called agents. The mechanism is made robust by keeping multiple copies of reports at different agents. When a node wishes to interact with another node, it sends messages querying trustworthiness of the other node to random nodes in the network, which are 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 et. al. [6, 7] 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 its 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 node that possesses a given resource and they focus on vote aggregation on incorporating voter credibility and on ensuring the integrity of trust reports as they pass over the insecure network.

Kamvar et. al. [8] use a different approach and assume that trust is 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 by using a trust matrix at each node. Successive iterations involving exchange of trust values with neighbors and re-computation 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.

Buchegger et. al. [9] propose a modified Bayesian approach to trust. Like Damiani et. al. they separate a node's reputation (performance in the base system such as file-sharing, routing, distributed computing etc.) and its credibility[1] (performance in the reputation system). In their solution, only information on first-hand experiences is published by nodes. This information is used by other nodes to construct their own reputation and credibility data structures for other nodes. Reputation data is also aged giving less weight to evidence received in the past.

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

In this section, we describe the RQC algorithm. In RQC, each node is assigned M score managers and all transaction involving the node are reported to each of its score managers[2]. The score managers aggregate trust information for the node to construct its global trust value or *reputation*. They also respond to rating requests by nodes wishing to transact with that node. A quality value is attached to all trust ratings by the sender (i.e., a node reporting on a transaction or a score manager responding to a trust query). The recipient of these ratings (the score manager and the querying node respectively) weighs them using this attached quality value and the credibility of the sender. Sender credibility is calculating by comparing its reported trust rating with the average value computed at the recipient. A sender that reports values that diverge from the calculated average sees its credibility go down whereas a sender whose reported values agree with the average sees its credibility go up. In the following sections we explain this process in greater detail.

3.1 Local Opinion and Opinion Quality

Each node i maintains an average opinion O_{ij}^{avg} of the behavior of all nodes j with which it has had an interaction. O_{ij}^{avg} can be interpreted as node i's estimate of the probability that node j will behave honestly during a transaction. After each interaction with j, i updates O_{ij}^{avg} and sends the updated value to all the score managers responsible for j.

[1] They call it trust.

[2] Since the score managers are also nodes in the network, we use the term "node" to refer exclusively to a node when it is not acting in its capacity as a score manager.

Along with O_{ij}^{avg}, the node i also stores the number of interactions, N_{ij}, it has had with j and the variance s_{ij}^2 in the behavior of j. Thus, the average local opinion is computed as follows:

$$O_{ij}^{avg} = \frac{\sum_k O_{ij}^k}{N_{ij}} \tag{1}$$

where O_{ij}^k is node i's opinion of its k^{th} interaction with j. Each O_{ij}^k takes a value in $[0, 1]$ and represents node i's satisfaction with node j's behavior during the transaction.

When node i sends its updated average opinion about j (O_{ij}^{avg}) to the score managers, it also sends an associated quality value, Q_{ij}. Q_{ij} represents the quality node i attaches to the opinion information it is sending to the score managers and lies within $[0, 1]$. Q_{ij} enables a node to express the strength of its opinion. Its value could depend on the context of the interaction as well as on past transaction history. In our current implementation, quality values are computed solely on the basis of the number of interactions and the variance in the opinion.

We assume that j's trust behavior is a normally distributed random variable. Through interactions with j, node i makes observations of this random variable resulting in a sample. The sample mean and standard deviation are then simply O_{ij}^{avg} and s_{ij}.

The quality value of the opinion (Q_{ij}) is defined as the confidence level that the actual mean trust rating for a node lies within the confidence interval:

$$O_{ij}^{avg} \cdot (1 \pm \frac{r}{100}) \tag{2}$$

where r is a system parameter that denotes the size of the confidence interval as a percentage of the sample mean. We experimented with various values of r, ranging from 5 to 30 and found that a confidence interval of 10% of the sample mean (and thus $r = 10$) resulted in the best performance. Using too high a value for r produced useless quality values, as large variations in node behavior were allowed without any decrease in the quality. Similarly, too low a value of r resulted in excessively low quality values.

Since the actual mean and standard deviation are unknown, we used the *Student's t-distribution* to compute the confidence levels. Note that the usual idiom is inverted here in that we know the interval and wish to compute the probability that the actual mean lies within the interval as opposed to normal practice where confidence level is known and the required interval is computed.

The t-value for the *Student's t-distribution* is given by the following equation:

$$t = \frac{r}{100} \cdot \frac{O_{ij}^{avg} \cdot \sqrt{N_{ij}}}{s_{ij}} \tag{3}$$

And the quality value is computed as:

$$Q_{ij} = 1 - B\left(\frac{(N_{ij}-1)}{(N_{ij}-1)+t^2}; \frac{1}{2} \cdot (N_{ij}-1), \frac{1}{2}\right) \tag{4}$$

where B is the *Incomplete Beta Function* defined as $B(z; a, b) \equiv \int_0^z u^{a-1} \cdot (1-u)^{b-1}\, du$

Thus 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 (resulting in a smaller variance). When the number of observations is high but they do not agree with each other, the quality value is lower.

When a node has had only one interaction with the other node, equations 3 and 4 cannot be used since sample variance is undefined for a sample size of one. Instead, a default quality value of 1 is used in this case. If a lower value, such as 0.5, is used, the opinions sent by malicious nodes during the initial interactions (when there is little credibility information[3]) would overwhelm the opinions of the good nodes, as malicious nodes would always report a quality value of 1 for their opinions.

After each interaction, both participating nodes report their updated opinions along with the associated quality values to the score managers responsible for their counterpart. The inclusion of quality in the message sent to the score manager allows the score manager to gauge the how much confidence the node itself places in the rating it has sent.

3.2 Computation of Reputation at the Score Manager

The underlying DHT structure designates M score managers for each node in the network. Since the score managers are selected from nodes within the network itself, each score manager is responsible for M nodes on average. This provides the reputation system with redundancy so that the failure of a few score managers does not affect the trust management system.

A score manager receives an averaged opinion whenever a node it is responsible for is involved in a transaction. This opinion is sent by the other node involved in the transaction. The individual opinions from successive transactions are aggregated to form the *reputation* of a node. The score managers therefore represent the global, system-wide view of a given node's behavior, updated every time a new opinion is received.

Along with the aggregate reputation, R_{mj} for a node j, a score manager, m, also stores the number of opinions it has received about j, N_{mj} and the variance s^2_{mj} in the reported opinions. This information is used to compute the quality value that a score manager attaches to a reputation value. The computation of the quality of reputation values at a score manager is similar to that of the individual nodes' computation of quality values for their averaged opinions described in equations 3 and 4. So, the quality of a reputation value is simply the confidence level that the actual mean reputation of the node is within $r\%$ of the sample mean.

If the reputation value of a node at the score manager has been calculated using the opinion of a single voter only, a quality value of 1 is returned. The reason for this is the same as described in the previous section.

3.3 Retrieval of Trust Information

When a node wants to know the reputation of another node before interacting with it, it locates the M score managers for the node by using the DHT substrate and asks them for the reputation of the node in question. Each score manager responds with a reputation value and an associated quality value. The node then computes the average

[3] The role of credibility is explained in Section 3.4.

reputation for the node in question, R_{ij}^{avg} using the quality values and the credibility values of the score managers since a score manager itself may also be malicious and send the wrong reputation values. In this way, multiple score managers allow the system to cope with malicious behavior.

3.4 Credibility

C_{ij} or the credibility of node j in the eyes of node i is the confidence node i has in node j's opinions about other nodes and contributes to the weight node i gives to the opinions expressed by node j and to the reputation values furnished by node j in its capacity as a score manager. A single credibility value is used for a node for both of its roles as a reporting node and a score manager.

Every node stores the credibility rating for each node (or score manager) that has sent it an opinion value (or a reputation value). The credibility rating is updated every time a node reports an opinion or reputation. In addition, when the score manager 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 and/or a large variation in experience where this was explicitly stated by the reporting node through a low quality rating. Credibility values are not shared with other nodes and are used simply to weigh the responses received from other nodes and always lie within the range $[0, 1]$.

3.5 Putting it all Together

A score manager uses the quality value sent by a reporting node and the credibility of the reporting node to compute the average reputation of a node. The reputation of a node j is computed at the score manager m as follows:

$$R_{mj} = \frac{\sum_i O_{ij}^{avg} \cdot C_{mi} \cdot Q_{ij}}{\sum_i C_{mi} \cdot Q_{ij}} \tag{5}$$

where R_{mj} is the aggregated reputation of node j, C_{mi} is the credibility of node i according to the score manager m, O_{ij}^{avg} is the average opinion of j reported by i and Q_{ij} is the associated quality value reported by i. The score manager only keeps the latest value of O_{ij}^{avg} reported by each node i. Thus the score manager gives more weight to ratings that are considered to be of a high quality and that come from nodes who are more credible in the eyes of the score manager.

In the case of reputation retrieval, a node aggregates the responses from the reporting score managers using the reported quality value and the stored credibility value of the reporting score managers. The aggregation is performed in exactly the same way as shown in equation 5 except that the reputation values R_{mj} are aggregated instead of the opinions O_{ij}^{avg}.

$$R_{ij}^{avg} = \frac{\sum_y R_{yj} \cdot C_{iy} \cdot Q_{yj}}{\sum_y C_{iy} \cdot Q_{yj}} \tag{6}$$

where R_{yj} is a reputation value received from a score manager y about node j.

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

$$C_{mi}^{k+1} = \begin{cases} C_{mi}^{k} + \frac{(1-C_{mi}^{k})\cdot Q_{ij}}{2} & \text{if } |R_{mj} - O_{ij}^{avg}| < s_{mj} \\ C_{mi}^{k} - \frac{C_{mi}^{k}\cdot Q_{ij}}{2} & \text{if } |R_{mj} - O_{ij}^{avg}| > s_{mj} \end{cases} \tag{7}$$

where C_{mi}^{k} is the credibility of node i after k reports to score manager m, $O_{ij}^{a}vg$ is the opinion reported by node i, Q_{ij} is the associated quality value, R_{mj} is the aggregated reputation and s_{mj} is the standard deviation of all the reported opinions about node j. Thus, credibility updates take the reported quality value into account. Since an opinion with a smaller quality value does not count as much at the score manager, the change in credibility is proportionately lower. At the highest reported quality value of 1, a reported rating that falls within one standard deviation of the aggregated reputation, increments the credibility of the reporting node by half the amount required for credibility to reach 1. A reported rating outside this region results in the credibility rating dropping to half the previous value.

In this way, if a reporting node (or score manager) is malicious, its credibility rating is gradually reduced when its opinion does not match that of other nodes (or score managers). And a node with a lower credibility value therefore contributes less to the aggregated reputation at the score manager.

3.6 Resource Requirements

Since each node has M score managers associated with it, a transaction between two nodes results in $2M$ messages to the score managers. Similarly, a trust query from a node to the score managers of its potential transaction partner results in M messages to the score managers and M responses. As the number of score managers M does not depend on the number of nodes in the network, the network traffic increases by a constant factor due to the RQC scheme.

Assume that a node transacts with K nodes on average. Since each node acts as a score manager for M nodes and stores the last reported opinion from each of the K transaction partners for these M nodes, the storage requirements for being a score manager are $O(KM)$. Each node also stores its own average opinion for the K nodes it has interacted with, resulting in $O(K)$ storage. Hence the total storage requirements for a node are $O(KM + K) = O(KM)$.

The processing requirements at each node are relatively light. Each transaction (preceded by two trust queries) results in reputation and quality computations at $2M$ score managers (M for each transacting node), followed by weighted averages of the reputation being computed at each transacting node, followed by a single average opinion update at each of the $2M$ score managers.

4 Experimental Results

We simulated the RQC scheme using Pastry [10], a structured overlay network that uses distributed hash tables for routing. 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. Moreover, nodes do not leave or join the network in the during the simulation. Since Pastry is written in Java, we decided to implement our reputation system in Java as well.

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 in the key space 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.

In the experiments we describe, we simulated a network of 200 nodes unless specified otherwise. In each experiment, 50000 random transactions took place unless specified otherwise (i.e., each node has an average of 50 interactions). Both participants of each interaction are chosen randomly. The default number of score managers storing reputation ratings for each node was 6 unless specified otherwise. The reason we chose a network of this size had to do with the running time for the simulation. We simulated a network of 5000 nodes and $1,000,000$ interaction, but since we were using the Pastry substrate and the entire network was being simulated on just one machine with 1GB of RAM, this took several hours for each run. As the experiment in Figure 5 shows, the impact of the number of nodes was very small on the results of the simulation as long as a sufficient number of interactions took place. Hence, our results should be valid for larger networks as well.

We decided instead, to invest our CPU resources on running each experiment 10 times. In the figures that follow, we plot the average of the results obtained from the 10 experiments, along with a confidence interval of size $\frac{s}{\sqrt{n}}$ where s is the standard deviation and n is the number of samples (10).

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 former case, two good nodes (and two bad nodes) give each other a rating of 1 after interacting whereas if a good and a malicious node interact they both give each other a rating opinion of 0. In the latter case, the node behaves maliciously in its capacity as a score manager and sends incorrect reputation values to requesting nodes.

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.

The performance of our scheme is evaluated as the number of correct decisions made (i.e., interactions with good nodes that went ahead plus interactions with malicious nodes that were avoided) as a proportion of the total number of decisions made. So, an interaction with a malicious node 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 no information about the reputation of a node can be found are not counted.

4.1 Types of Maliciousness

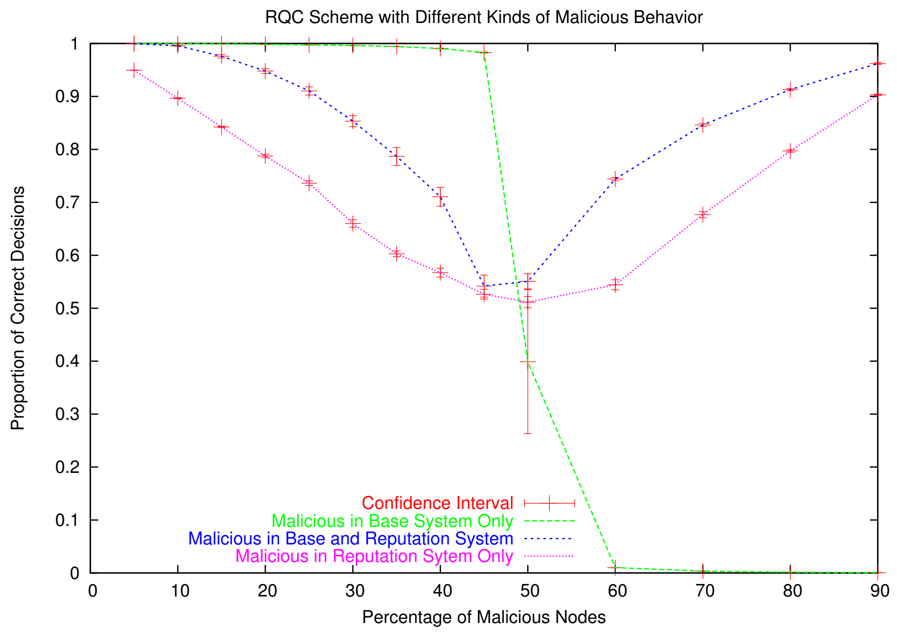

Fig. 1. Performance of the RQC Scheme with Different types of Malicious Nodes

In Figure 1, 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.

Malicious behavior in the reputation system is defined as the sending of false reputation values by 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 chose this model of maliciousness as it is the worst possible type of maliciousness from the perspective of a reputation system. Other models of maliciousness, such as sending an arbitrary reputation value in response, do not cause as much harm to the reputation system.

In the case when nodes act maliciously in the base system only, our scheme performs very well till the percentage of malicious nodes reaches 50%. Note that the confidence interval is very large at this point. This shows that our scheme is very sensitive to the proportion of malicious nodes in the system. When the proportion of malicious nodes exceeds half, the dominant ethic of the system becomes that of the malicious nodes. All good nodes are branded as malicious and vice-versa resulting in a precipitous drop in performance.

Another striking feature in Figure 1 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. 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.

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 50%. 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.

4.2 Comparison with the Aberer-Despotovic Scheme

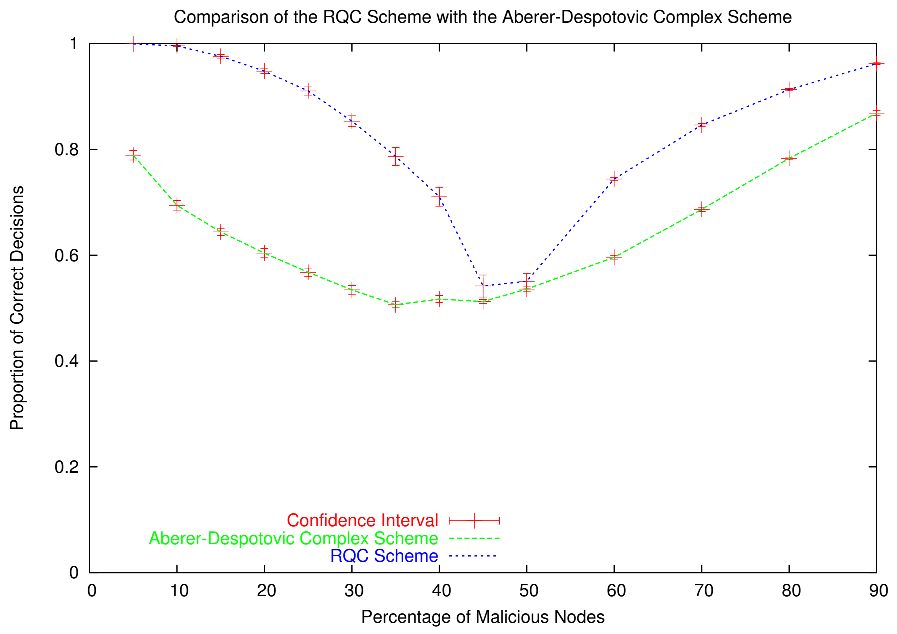

Fig. 2. Comparison of the RQC Scheme with the Aberer-Despotovic Scheme

In this experiment we compared the performance of our scheme against the trust management scheme proposed by Aberer and Despotovic in [5]. We implemented their scheme in Pastry and ran experiments with the same number of nodes and interactions as in our scheme (200 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 large number of interactions to finish. As we can see in Figure 2, our scheme performs consistently better than the Aberer-Despotovic scheme in terms of the proportion of correct decisions made.

Figure 2 compares the performance of the two schemes when there is maliciousness in both the base and the reputation system. While we do not show the corresponding graphs for other types of maliciousness, we would like to note that the RQC scheme also outperforms Aberer-Despotovic for other two models of maliciousness.

4.3 Probabilistic Cheating

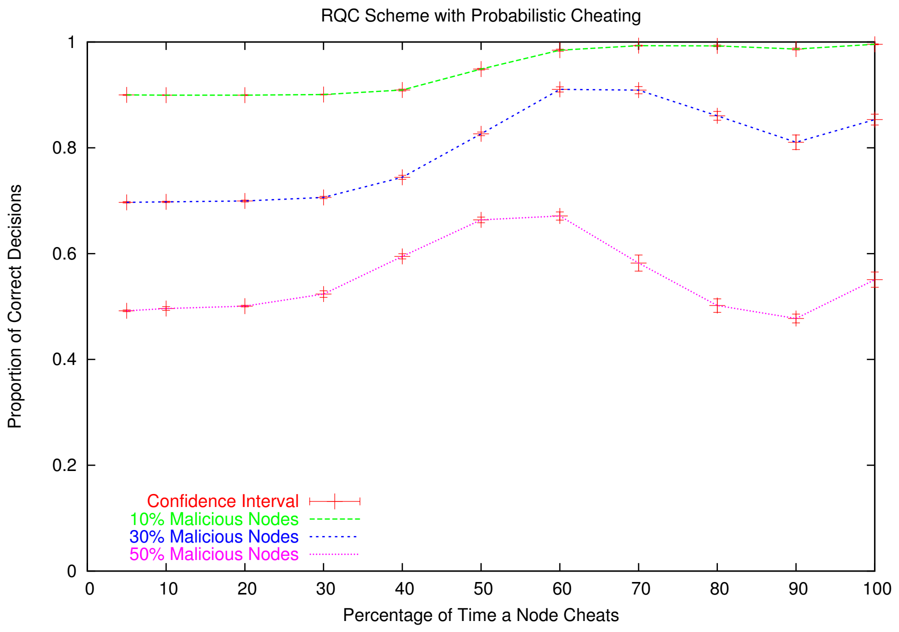

Fig. 3. Performance of the RQC Scheme with Probabilistic Cheating

In Figure 3 we examine the case when the malicious nodes do not cheat all the time but instead cheat with a certain probability. The three curves correspond to a total of $10\%, 30\%$ and 50% of the nodes being malicious. We see that the proportion of correct decisions is slightly affected by the probability of a node cheating with the RQC scheme generally performing better as nodes cheat more consistently.

4.4 Number of Score Managers

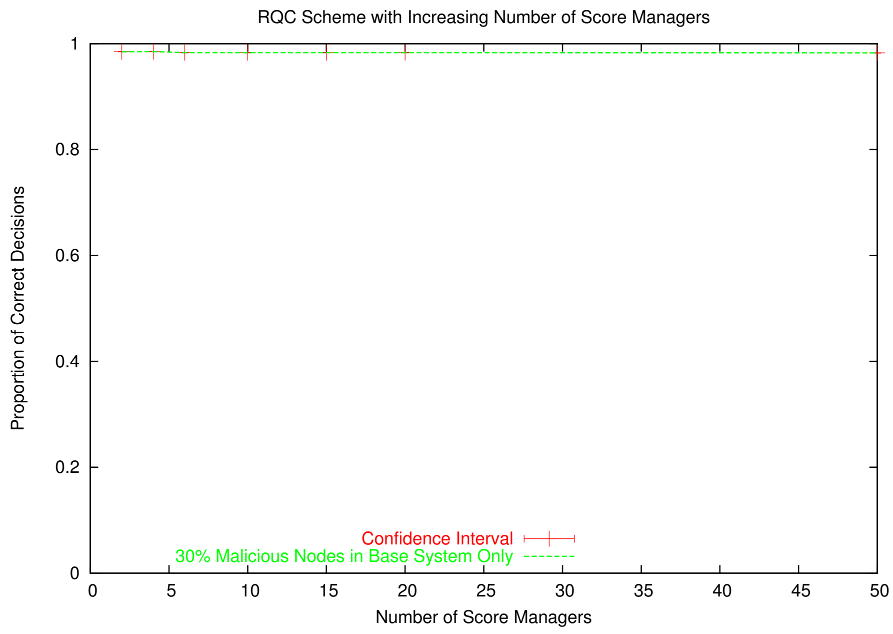

Fig. 4. Performance with Increasing Number of Score Managers

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 in the base system only. The number of score managers makes no difference to the performance.

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 30%, acting maliciously only in the base system. We ran the experiment with 100, 200, 400, 1000, 2000 and 5000 nodes. There are three curves in the figure corresponding to the total number of interactions being $10,000$, $40,000$ and $160,000$.

We find that an increase in the number of nodes results in a drop in performance if the number of transaction is kept fixed. As the number of nodes increases, the number of interactions per node decreases resulting in less reputation information per node being

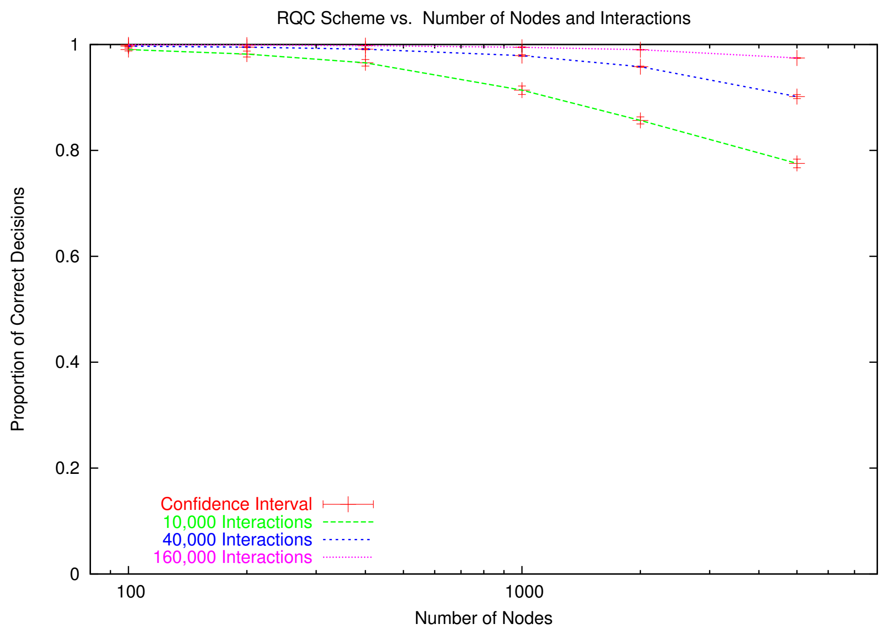

Fig. 5. Performance scaling with Number of Nodes in System

stored in the system. However, when the number of interaction is increased, the increase in the number of nodes does not have much impact on the performance of the scheme. Hence the scheme scales well with the size of the network if each node takes part in sufficient interactions.

5 Conclusions and Future Work

In this paper, we have presented the **RQC** scheme for trust management in autonomic systems. Our scheme computes Reputation(R), Quality(Q) of 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 its 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 [9] the credibility of a node is lowered if its 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 half. It significantly outperforms 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.

While we apply the RQC scheme to measure node honesty, the scheme also lends itself to other soft-management tasks. For instance, the RQC scheme can be used to measure the reliability or the QoS provided by nodes in the network.

There are still several steps that can be taken to improve the RQC scheme and we are currently working on several enhancements. We are working to incorporate "churn" in our experiments. "Churn" is the phenomenon of nodes joining and leaving a network and is an important component of autonomic and P2P systems. We are also looking at extending the RQC scheme for various network topologies. Finally, we are studying alternative decision-making strategies such as incorporating individual node opinions along with system-wide reputation values when deciding whether to interact with another node.

References

1. 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
2. Resnick, P., Zeckhauser, R., Swanson, J., Lockwood, K.: The value of reputation on ebay: A controlled experiment (2002)
3. 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
4. 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
5. Aberer, K., Despotovic, Z.: Managing trust in a peer-2-peer information system. In: CIKM. (2001) 310–317
6. 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)
7. 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
8. 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

9. 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)
10. 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