Scalable reputation management with trustworthy user selection for P2P MMOGs

Guan-Yu Huang, Shun-Yun Hu* and Jehn-Ruey Jiang

Department of Computer Science and Information Engineering,
National Central University,
Taiwan, ROC
E-mail: aby@acnlab.csie.ncu.edu.tw
E-mail: syhu@csie.ncu.edu.tw
E-mail: jrjiang@csie.ncu.edu.tw
*Corresponding author

Abstract: Recent research on Peer-to-Peer Massively Multiplayer Online Games (P2P MMOGs) has tried to find more scalable and affordable solutions to build virtual environments via the resource sharing of clients. However, P2P approaches face the problem of client misbehaviours where game rules are not processed properly, undermining a game's fairness and normal operations. In this paper, we present REPS, a distributed reputation management system for P2P MMOGs that allows trustworthy clients be identified to perform important tasks. REPS first considers the generation, storage, and query for certain *reputation factors* that could be the basis to evaluate user trustworthiness. We then propose Trustworthy user Selection (TuS) to adjust the weights of each factor to give preference to the more important ones. Based on the mutual rating among users and reputation queries, REPS provides a scalable and reliable mechanism to facilitate decision making, from choosing trustworthy superpeers, to deciding whether to interact with particular users.

Keywords: P2P; peer-to-peer; virtual environments; trustworthy user selection; distributed reputation management.

Reference to this paper should be made as follows: Huang, G-Y., Hu, S-Y. and Jiang, J-R. (2008) 'Scalable reputation management with trustworthy user selection for P2P MMOGs', *Int. J. Advanced Media and Communication*, Vol. 2, No. 4, pp.380–401.

Biographical notes: Guan-Yu Huang is currently a Master student at National Central University, Taiwan. His current research interest encompasses distributed systems.

Shun-Yun Hu is currently a PhD student at National Central University, Taiwan. His main research interests are networked virtual environments and peer-to-peer systems. He started the SourceForge project VAST (http://vast.sourceforge.net) in 2005, to provide open source libraries for creating scalable peer-to-peer-based virtual environments.

Jehn-Ruey Jiang received his PhD Degree in Computer Science in 1995 from National Tsing-Hua University, Taiwan. He is currently with the Department of Computer Science and Information Engineering, National Central University. He was a recipient of the Best Paper Award of the 32nd *International Conference on Parallel Processing*, 2003, and a recipient of Excellent Paper Award of Mobile Computing 2004. His research interests include distributed computing, mobile computing, pervasive computing and peer-to-peer computing.

1 Introduction

Massively Multiplayer Online Games (MMOGs) such as World of Warcraft and Second Life, where over hundreds of thousands of players assume virtual identities and engage in various interactions, have become very popular in recent years. These virtual worlds are very attractive as they provide immersive 3D environments that people can constantly explore together. As of 2008, there are more than 12 millions registered Second Life accounts and over 11 millions paying subscribers in World of Warcraft. As user population grows, the traditional client-server architecture will suffer from the server's limited bandwidth and processing power. To solve this problem, peer-to-peer networked virtual environments (P2P NVEs, e.g., Knutsson et al., 2004; Bharambe et al., 2006; Hu et al., 2006) have since been proposed.

In client-server architectures, the server receives and processes all the user-generated events. This ensures that the actions of every participant are monitored, and game rules are executed objectively as the designers have intended. Cheating is also restricted as all important processing is done by the servers. However, P2P NVEs do not have such fairness guarantee because most server functions are now assumed by some clients. A client may modify any information that it possesses and may even change to a new identity after it has cheated. Although, most players may not go to great lengths to cheat, as modifying the game code requires certain technical skills. But even if only a small portion of the users is successful at cheating, gameplay can still be disrupted seriously.

Fortunately, we observe that the nature of MMOGs is highly socialised, and users often invest large amounts of time and energy to build their in-game persona to ensure their status in the virtual world. Users often are also regulated by guilds or other social organisations, as opinions from other users affect one's reputation and social experiences even more than other in-game activities. In other words, there exists strong social forces in successful MMOGs where active users typically value highly their status and reputations among peers. Such reputations thus may be exploited to facilitate certain game operations, such as the selection of trustworthy clients for important functions (e.g., the group leader or manager for a region). We have seen similar mechanism in online marketplaces such as *eBay* or *Yahoo Auctions*, where online reputations based on mutual user ratings are used to estimate the trustworthiness of a user. If such reputation mechanism can be adopted in P2P MMOGs, it might help users make decisions on whether to interact with a particular peer, or to select the more trustworthy clients to hand over responsibilities.

One challenge for reputation schemes in a distributed environment is how to deal with the disruptions from malicious users. The reputation scores given by users might not be accurate enough to reflect true trustworthiness, because malicious users can give false reputation ratings. Also, ratings related to specific user behaviours may not be generalisable to judge the overall trustworthiness of a user. Therefore, considering more game parameters may help to increase the estimation accuracy and restrain the interruptions from malicious users. Reputations in a game can be affected by many factors (i.e., the *reputation factors*). Besides mutual ratings among peers, accumulated online time, the number of completed tasks or trades, etc., all can be the basis to judge trustworthiness. However, the importance of each factor can be different, so we need to give appropriate weights to each factors. Here we define *importance* as how well a given factor can discriminate among users (i.e., the users have a higher degree of variability in respect to the factor), because with higher discriminability, it indicates that the factor can better discern the differences among users. How to utilise various parameters to decide trustworthiness and assign weights for each relevant factors are thus the main problems for us.

In this paper, we propose REPS, a reputation management system for P2P MMOGs based on peer-rated reputations. Each user has a reputation value based on other users' subjective opinions during their interactions. The reputation data is stored distributively among all users for scalability and security reasons. We also propose the process of Trustworthy user Selection (TuS) to choose trustworthy users. TuS uses a statistical regression to combine all potential reputation factors and compute their importance weights, so that only users matching the strictest reputation criteria are chosen as trustworthy users.

The rest of this paper is organised as follows. Section 2 provides background on reputation management and P2P NVEs. Section 3 presents our problem formulation and challenges in distributed reputation management. We describe the design of REPS in Section 4 and the design of TuS in Section 5. Evaluations for REPS are performed in Section 6, while concluding remarks are given in Section 7.

2 Background

2.1 Reputation management

Recently, there have been a number of reputation systems proposed for P2P applications, often in the context of e-commerce (e.g., Atif, 2002; Dellarocas, 2001; Aberer and Despotovic, 2001; Xiong and Li, 2004; Ismail and Josang, 2002; Josang et al., 2007). The goal of these systems is to compute the reliability of a user and predict future behaviours in respect to a specific metric, and the P2P approach is to reduce the overhead for servers. Such predications are based on past experiences and interactions with the user, who is often a buyer or seller in an existing distributed or semi-distributed e-commerce environment. The reputation value represents a global view for the user's behaviour, and can be used as reference to warn of or convince other users. Users may also quickly identify whether another user in contact is trustworthy, and could thus avoid interactions with *malicious users* who cheat for private benefits. The reputation systems in these P2P applications calculate a peer's reputation value by collecting the local evaluations from other users. For example, in Kamvar et al.

(2003) and Dellarocas (2001), the sum of a user's rating from every transaction is used to compute each user's personal reputation value. To make reputation values more globally accessible and reliable PeerTrust (Xiong and Li, 2004) normalises the values by specific weights based on each user's global reputation value.

Some recent approaches like Ganeriwal and Srivastava (2004), Mui et al. (2001), Buchegger and Le Boudec (2004) and Zhou and Hwang (2007) use the Bayesian method that takes a binary input (i.e., positive or negative) to predict the cheating probability of the next transaction with a user based on past experiences. Zhang and Fang (2007) provides the QoS experience vectors to perform reputation evaluation on many levels to determine reputations more precisely.

When querying someone's reputation in P2P applications, a decentralised method is often used to aggregate reputation scores from various places to compute a global reputation value. Users thus not only evaluate each others but also learn of someone's reputation value by aggregating the evaluation records. In a client-server architecture, the server stores all the reputation data, and users just query the server for one's reputation. However, in a decentralised environment, often a P2P storage such as Chord (Stoica et al., 2001), CAN (Ratnasamy et al., 2001) or P-Grid (Aberer, 2001) is used to distributively store the reputation data on other peers. For example, Zhang and Fang (2007) uses Chord to find the successors of a user A, where A's reputation records (evaluated by other users) are stored on its successors. When other users need to know A's reputation, they can hash A's identifier to aggregate the reputation records from A's successors. Similarly, EigenTrust (Kamvar et al., 2003) uses the identifier hash to discover successors to store the reputation values by using CAN (Ratnasamy et al., 2001) or Chord (Stoica et al., 2001).

There are other issues in P2P reputation management. For example, TrustGuard (Srivatsa et al., 2005) describes how to distinguish honest persons from dishonest ones, or to detect the dishonest ones pretending to be honest; how to filter extreme (i.e., too positive or negative) or fake reputation evaluations to ensure the final reputation's correctness; and how to prove that a reputation management is reliable for a given application? There are many researches that discuss these problems for P2P and non-P2P applications (e.g., Atif, 2002; Yan et al., 2007; Dellarocas, 2001). We will discuss how REPS deals with these problems in P2P MMOG scenarios.

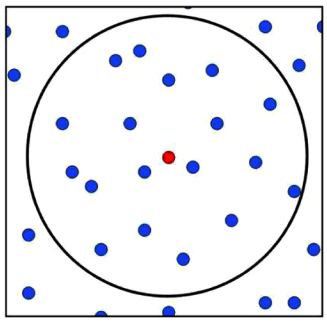
2.2 P2P NVEs

In NVEs, every participant has a visibility range called Area of Interest (or AOI, see Figure 1). The AOI is often circular, and other users within the AOI are called a user's *AOI neighbours*. Users can exchange messages to comprehend the environment around them, and see the dynamic updates from other AOI neighbours. The key to scalable P2P-based NVEs is based on the fact that users have limited views within their AOI and only need to know information within the AOI. The scalability of the whole environment thus can be extended if each user only exchanges messages with its AOI neighbours, without going through the server.

In some approaches (e.g., Knutsson et al., 2004; Yamamoto et al., 2005; Lee and Sun, 2006; Hu et al., 2008), the whole world is divided into serval disjoint regions in order to manage information updates effectively. Some participants with better capacities are chosen as *superpeers* to relay information (e.g., position updates and

event notifications) for other users. Lo et al. (2005) describe superpeers as having a special role that can provide services to non-superpeers.

Figure 1 Large circle is the AOI of the centre star user (see online version for colours)



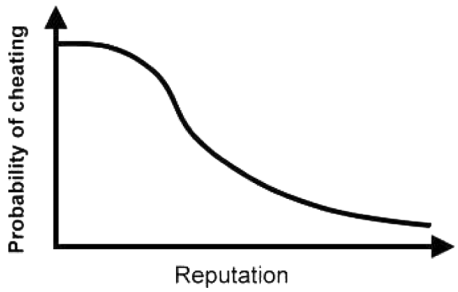
For many P2P NVE schemes that adopt superpeers, whether the selected clients are trustworthy is essential for the system's proper operations. One of the implications for our proposed distributed reputation management thus is to provide a reliable method for selecting trustworthy nodes that may assume important superpeer functions.

3 Problem formulation and challenges

Our goal is to build a scalable reputation management system that supports P2P MMOGs by developing a distributed method to rate, store, and query reputation values. Trustworthy users can then be selected based on these reputation evaluations. The main problem is how to store the reputation scores on reliable peers and query them effectively. We first make the following assumptions:

- Every user has a fixed AOI radius, where users see each other only when they are within each other's AOI. Between two mutually visible users, certain game-specific interactions can occur (e.g., talking, fighting, trading, etc.). The users within AOI, or *AOI neighbours*, change periodically due to users' positional changes as they move.
- We assume that a P2P NVE overlay exists to provide a list of AOI neighbours for each user (e.g., Knutsson et al., 2004; Bharambe et al., 2006; Hu et al., 2006). So any user may connect and exchange messages directly with its AOI neighbours.
- Two mutually visible users can rate each other multiple times with a score of positive, neutral, or negative $(+1, 0, -1)$ based on past interactions. A reputation record follows the form of (*rater, rated-user, evaluation*), where *rater* is the user making the rating, *rated-user* is the user being evaluated, and *evaluation* records the actual rating.
- We assume that the probability for a user to cheat decreases with a person's reputation value, especially if the reputation has exceeded certain threshold, as possibly a lot of effort has been spent to build the reputation (Figure 2).

Figure 2 Probability to cheat and reputation value



Based on the above scenario, some challenges for a reputation system in P2P MMOGs are outlined below:

Reputation evaluation: Building a reputation system requires the experiences and inputs from users as the basis for reputation values. How to efficiently and precisely represent user impression about each others thus is the first problem faced by any reputation schemes. Reputations are meaningless if most values are close to zero due to the lack of rating. In MMOGs, players often focus more on the game itself than on miscellaneous activity such as reputation evaluation, mechanisms to encourage user rating thus is needed. To provide incentives for peer evaluation, the evaluation method needs to be simple and efficient, so that the evaluation can be done conveniently, and the reputation values can be aggregated quickly.

Storage and query: How to store and query reputations in a fully distributed environment is the main challenge for a P2P reputation system. To ensure that the system would scale, we need to store the data distributively while avoiding any server or client overloads. For the purpose of efficiently querying reputation data, to find the users that store the reputations and to collect the data with minimal delays are two main considerations.

Security: Ensuring the reliability and trustworthiness of the information is another important aspect for a reputation system. In P2P environments, users may modify the reputation data they keep for private gains. This would disrupt the validity of the reputation data and possibly cause misunderstandings among users. Therefore, a system also needs to be able to prevent or recover from possible cheating behaviours.

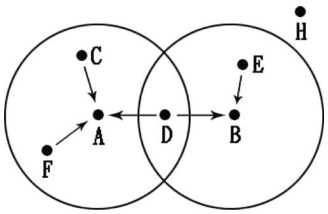
4 Basic architecture of REPS

We note that there are many game-related indicators, or *reputation factors*, related to someone's reputation in a MMOG. These factors are generated during game play, and can be based on certain game statistics (e.g., number of completed tasks or accumulated online time) or feedbacks from other users (e.g., reputation ratings). The game statistics can be collected at the nodes that manage game states (e.g., servers or superpeers), whereas reputation feedback is obtained from the evaluations of other users. Below we describe how REPS performs rating, and how the reputation values are stored and queried.

4.1 Localised reputation evaluation

In REPS, users perform mutual rating when they are within each others' AOI, because interactions can only occur with AOI neighbours. For example, in Figure 3, users C and F could rate A because they are within A's AOI. Rating may occur with a probability related to the intensities of interactions. To ensure that rating would only occur after user interactions, interacting users have to generate a *rating right* authorised by the rated user to the potential rater, so that the rater can give a rating at some later time, while preventing users to rate people whom they have never interacted with. The rating right contains the rated user's unique identifier and IP address, and is recorded at the rater so that rating may be performed at a later, more convenient time. Rating right can be generated via Proxy Signature (Das et al., 2006), which basically provides a method to authorise a user to act on behalf of the authoriser to perform certain tasks. In our case, the rated user authorises the rater to modify his or her reputation value at another third party node (called *reputation manager* that will be described later). However, the details of such authorisation is beyond the scope of this paper.

Figure 3 The rating condition in REPS



As an example, if user C rates user A with the score of 1, then a rating record of (C, A, 1) will be stored at A's reputation manager, which would update A's reputation based on A's existing reputation value.

4.2 Reputation storage and query

Similar to EigenTrust (Kamvar et al., 2003) and Powertrust (Zhou and Hwang, 2007), in order to scalably store the reputation records, a user chooses M users as its *reputation managers* to store and retrieve reputation data, where M is a system-wide parameter. The reason for having M reputation managers is to prevent the loss or corruption of reputation data due to the failure of malicious act of any single reputation manager. Reputation managers are chosen by hashing unique user identifers using M different Distributed Hash Table (DHT) functions such as Chord (Stoica et al., 2001) or CAN (Ratnasamy et al., 2001). DHT provides an unique mapping between a key (such as user identifer) and a user node located within a logical coordinate space, so by using M hash functions, M separate nodes can be selected to store the reputation data for any given user. As the hash functions are well-known and agreed upon in advanced, any other user can also easily locate the M reputation managers for a given user. Reputation managers are in charge of saving and computing the reputation score, while making sure that only users with the proper *rating right* can modify the respective reputation score. Potential raters thus send their ratings to a rated user's reputation managers by hashing the rated user's identifer via M different hash functions. Users can also

query a given user's reputation data from the respective reputation managers via the same way.

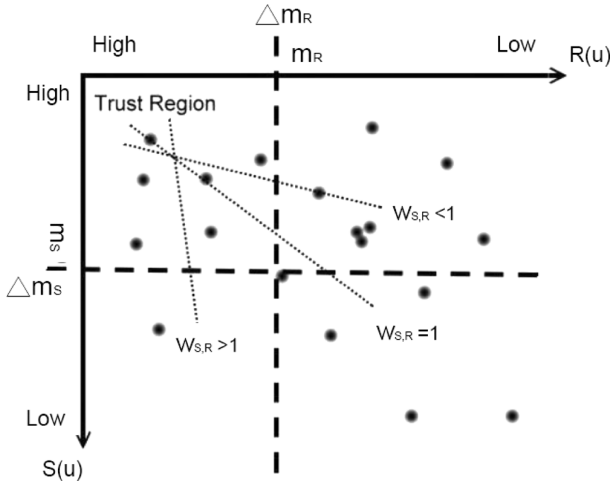
5 Trustworthy user Selection (TuS)

Based on the reputation scores collected from reputation managers and possibly some game statistics from game state managers (i.e., the server or some superpeers managing a region), the types of factors that are important and relevant in forming users' reputations could still differ for various MMOGs. In order to build a reputation system that can adapt to different game scenarios, REPS integrates all potential reputation factors to choose trustworthy users via a mechanism called Trustworthy user Selection (TuS). After collecting the the relevant reputation factors for some users, TuS can locally determine and adjust the importance of each reputation factors based on user behaviours, so that the more trustworthy users can be discovered for a given system.

5.1 Scenario description

In a typical scenario, there are r reputation factors that can affect users' reputation in the game world (e.g., reputation scores, number of completed tasks, accumulated online time, etc.). Each reputation factor has a weight w_i between 0 and ∞ that represents its importance. TuS also uses m_i to represent the *reputation threshold* of the ith reputation factor. For a user to be considered trustworthy, it must satisfy the thresholds for all reputation factors, where satisfaction means exceeding the thresholds. So the higher the value of a reputation factor, the better. If we plot the reputation factors of each user in consideration on an r-dimensional plane, then the set of points where all reputation thresholds are satisfied is called a *trust region* (as shown in Figure 4 for two reputation factors). Each threshold m_i would update with time and environment according to Δm_i, which indicates the magnitude of change for m_i. Each Δm_i would change as appropriate to adjust the size of the trust region according to the relative importance of each reputation factor.

Figure 4 Trust region in TuS



Take two reputation factors for example, we use the most popular reputation factors *total score*, $S(u)$, and *rating ratio*, $R(u)$, to explain how TuS chooses trustworthy users. Total score is the summation of every score $S(i,u)$ an user u receives from each rater i, and the rating ratio $R(u)$ indicates the proportion between the total score $S(u)$ and the total number of ratings $T(u)$ that user u receives. The higher $S(u)$ or $R(u)$, the more trustworthy user u is

$$S(u) = \sum S(i,u) \quad R(u) = \frac{S(u)}{T(u)}.$$

But which reputation factor is more important If user A scores 30 out of 100 ratings, and user B scores 9 out of 10 ratings. According to $S(u)$ alone, A is more trustworthy as its total is higher than B's. But the ratio $R(u)$ of B is higher than A's, making B more trustworthy. Yet since 100 people have rated A and only ten persons have rated B, the A's rating may be more significant. Some proportionality distortions thus exist (Table 1).

Table 1 Example of proportionality distortions

User	$S(u)$	$T(u)$	$R(u)$
A	30	100	0.3
B	9	10	0.9

Ideally, we would like to combine the effects of both the total score and the rating ratio, as they can both be meaningful. However, we do not know which is more important as it may differ across regions or MMOGs, where the willingness to rate can vary. So it is better for TuS to combine $S(u)$ and $R(u)$ in a flexible way.

Figure 4 illustrates the concpet of TuS by a two-factor example, where the x-axis represents all possible values for the rating ratio and the y-axis represents all possible values for the total score. A user u can be selected as a trustworthy user if its reputation point lies within the trust region (satisfying the conditions of $R(u) > m_R$ and $S(u) > m_S$, where m_R is between 0 and 1 and m_S is between the most negative rating and the most positive rating). If we want to select N trustworthy users, we can adjust the thresholds m_R and m_S so that there are exactly N points (i.e., user reputation values) in the trust region.

To adjust the thresholds of m_R or m_S, we define the value $w_{S,R}$ as the absolute value of the *regression coefficient* (i.e., the slope of the regression line for all points in the trust region). $w_{S,R}$ can be used as the relative importance weight for reputation factors from $S(u)$ to $R(u)$. We can also define $w_{R,S}$, the relative importance weight from $R(u)$ to $S(u)$, as the inverse of $w_{S,R}$. If $\overline{R}$ is the average $R(u)$ and $\overline{S}$ is the average $S(u)$ for all users within the trust region, then:

$$w_{R,S} = \frac{1}{w_{S,R}}$$

$$w_{S,R} = \left| \frac{\sum (R(u) - \overline{R})(S(u) - \overline{S})}{\sum (R(u) - \overline{R})^2} \right|.$$

The regression coefficient shows the distribution for all values, and taking absolute values means that TuS only cares about the direction of the distribution but not the shape of the regression line. If $w_{S,R} > 1$, the trend for points in the trust region is towards $S(u)$, the importance of $S(u)$ is thus relatively higher than $R(u)$ because the reputation points are more spread out (i.e., have greater variability) on the dimension of $S(u)$ than on $R(u)$. The weight of $S(u)$ thus should increase more. If $w_{S,R} < 1$, it means that the points are tilting towards $R(u)$ in the trust region, and instead the weight for $R(u)$ should increase more.

The actual adjustments Δm_R and Δm_S for m_R and m_S depend on the value of $w_{S,R}$, where $\Delta m_S / \Delta m_R = w_{S,R}$. TuS increases or decreases Δm_R and Δm_S simultaneously with the fixed ratio $w_{S,R}$ until the number of the candidate points matches the number of required users.

Likewise, Δm_R and Δm_S decrease with the ratio $w_{S,R}$ when the number of required trustworthy users is less. When TuS is first initialised, $w_{S,R}$, m_R and m_S are set to 1.0, 1.0 and the number of current online users (i.e., the maximum values for each threshold), which makes the area of the trust region null. We then reduce m_R and m_S to extend the trust region with $\Delta m_R / \Delta m_S = 1$ to find some initial trustworthy users.

5.2 TuS algorithm process

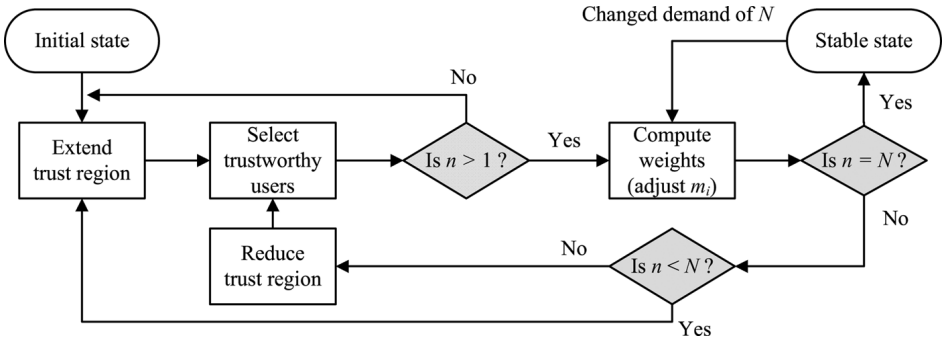
In order to choose trustworthy users, TuS adjusts the reputation thresholds according to the weights of each reputation factors. Assuming that a high value indicates good reputation, TuS adjusts the reputation threshold *more* for the more important reputation factors, and *less* for the less important reputation factors. As such, the change in the threshold's magnitude becomes closely related to the weight w_i of the reputation factor i (note that when $i = S$, $w_i = w_{S,R}$, as used in the last section).

When the desired number of trustworthy users, N, increases, TuS can select more users by extending the trust region. On the other hand, TuS increases the reputation threshold to reduce the trust region when the demand for trustworthy users is less. If the number of currently selected trustworthy users is n, w_i is the weight of the reputation factor i and adjustment fraction ρ is an adjustable system parameter between 0 and 1, then TuS adjusts threshold m_i of i by adding or subtracting a chunk Δm_i, where Δm_i is defined as follows:

$$\Delta m_i = \begin{cases} w_i \rho m_i & \text{if } n < N \\ \dfrac{1}{w_i} \rho m_i & \text{if } n \geq N \end{cases}.$$

Figure 5 shows the process of the TuS algorithm, where each reputation threshold is set as the highest value initially and the number of trustworthy users is 0. Then, each reputation threshold is decreased by the same rate $\Delta m_i = \rho$ to extend the trust region, so that users whose reputation factors higher then all the reputation thresholds can be selected as the trustworthy users. When there are at least two trustworthy users, TuS begins to calculate the weight w_i by multivariate analysis and adjust the threshold m_i by the formula above until N trustworthy users are found (the the calculation of w_i will be explained in the next section). At this point, and system goes into a stable state. When there are dynamic updates to the system (e.g., new AOI neighbours are encountered) or the number of required trustworthy users N has changed, the system would modify the weights for reputation factors, and adjust the size for the trust region.

Figure 5 Process flow of TuS. n = number of selected trustworthy users; N = number of required trustworthy users



5.3 *Multivariate analysis for weight adjustment*

In Multivariate Analysis, (Mardia et al., 1979) describe the statistical principles of multivariate statistics, which involves observations of more than one statistical variable. The method is used to perform tradeoff studies across multiple dimensions while taking into account the effects of all variables of interest. It analyses the principal components of all input variables to determine the component that is most discriminating. In mathematical terms, this means finding the distribution direction that will create the largest variations for weighted averages, and the weights for each variable guaranteed to generate the largest difference among all variables.

In TuS, we take r different reputation factors as the variables, s total users, and $x_{i,j}$ as the value for user *i's j*th reputation factor. In order to compute the weight of each reputation factor relative to the first factor, TuS finds each regression coefficients relative to the first reputation factor according to all current $x_{i,j}$ values by multivariate analysis. TuS then takes the regression coefficients as the reputation factor weights relative to the first reputation factor $w_{2,1}, w_{3,1}, \ldots, w_{r,1}$ (for brevity, we use $w_2, w_3, \ldots, w_r$ to represent them). The computations for the weights are shown as follows:

$$\begin{bmatrix} x_{1,1} \\ x_{2,1} \\ \cdots \\ x_{s,1} \end{bmatrix} = \begin{bmatrix} 1 & x_{1,2} & \cdots & x_{1,r} \\ 1 & x_{2,2} & \cdots & x_{2,r} \\ 1 & \cdots & \cdots & \cdots \\ 1 & x_{s,2} & \cdots & x_{s,r} \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \cdots \\ w_r \end{bmatrix} + \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \cdots \\ \varepsilon_r \end{bmatrix}$$

or equivalently:

$$\underset{(s*1)}{X_0} = \underset{(s*r)}{X} \; \underset{(r*1)}{W} + \underset{(s*1)}{\varepsilon}.$$

We ignore ε during computation because $\varepsilon = 0$ based on the assumption of Mardia et al. (1979) where the expected error for each reputation factor is minimum.

We can now use a matrix transformation to find the solution matrix W as follows:

$$W = (X'X)^{-1}X'X_0$$

where X' is the transformation matrix of X. Each w_i except w_1 in W represents the adjustment ratio of the trust region where X_i corresponds to X_1. Note that w_1 is not the weight of m_1 but the intercept of the distribution representing the absolute location of the distribution. In other words, if Δm_{X_i} and Δm_{X_1} are the respective adjustment ratios of X_i and X_1, then $w_i = \Delta m_{X_i}/\Delta m_{X_1}$ represents the adjustment ratio of the reputation factor i to the first reputation factor (i.e., the first reputation factor).

6 Performance evaluation

In this section, we simulate and compare the operations of TuS with other methods to select trustworthy neighbours. The main purpose here is to show and compare the accuracy of TuS under different conditions. We also evaluate the performance when different reputation factors are considered together. Our simulations are based on the simulator for VON (Hu et al., 2006), where each node in the system has a fixed AOI range and can exchange messages with its AOI neighbours. In the simulation, 2000 nodes are placed within a two-dimensional plane 1000- by 1000-unit, and each AOI radius is set to 100 units. As we are mainly interested in the accuracy of TuS's selection, we first assume that the values of various reputation factors are stored and retrieved from the reputation managers instantly without any delays. Note that on a real system, as DHT is used for the storage and retrieval of reputation values, to select N trustworthy users would incur N DHT queries, each with an average latency of $O(n \log n)$ (Stoica et al., 2001), where n is the number of users in the system. As the DHT queries can be performed concurrently, the overall additional latency is roughly $O(n \log n)$.

At the beginning of the simulation, each node is assigned a random location. They then move according to a random way-point (Hyytia et al., 2006) model for 1000 simulation *time-steps*. Each node also has its own *misbehaviour probability*, the frequency that misbehaviour occurs (e.g., a misbehaviour probability of 0.3 means that 30% of the node's interactions with others is bad and the other 70% is normal). Each node would rate each other whenever they are within each other's AOI, at a probability given by *rating frequency* (e.g., a 50% rating frequency indicates a node would on average, rate once for every two neighbour encounters). A score of 1 is given for encountering normal behaviours and a score of -1 is given for bad behaviours. We can then collect all the ratings to calculate the *total score* (i.e., summation of all ratings) and *rating ratio* (i.e., ratio of the total score to the number of ratings given).

Initial reputation thresholds for rating ratio and total score (i.e., m_R and m_S) are set to the maximal values (i.e., 1 and 2000, respectively), and the adjustment fraction ρ is 0.005. When the simulation starts, the trust region first extends according to the fixed weight $w_{S,R} = 1$ until the number of selected users exceeds one (because at least two points are required to calculate the regression line for the trust region). The area of the trust region would then extend according to how the weight $w_{S,R}$ is adjusted.

To evaluate the accuracy of selections, we require each user to identify a number of most trustworthy neighbours. The number is chosen to be 20, as this is roughly 25% of the average AOI neighbours in the most crowded scenario. The main accuracy metric is defined as whether the chosen trustworthy users are indeed the most trustworthy ones, based on their misbehaviour probability (i.e., the number of correctly identified users with the lowest misbehaviour probabilities). For example, if a method correctly identifies 9 out of 10 neighbours with the lowest misbehaviour probability, then the *reputation accuracy* is 90%. The *average reputation accuracy* represents the mean reputation accuracy for all users. Another useful matric is *convergence time*, defined as the average time for average reputation accuracy to exceed 95%. Convergence time shows how fast a given method can select the most trustworthy users.

6.1 Accuracy analysis

In the first set of simulations, we compare the average reputation accuracy for the following four methods: *TuS*, *Total score + Rating Ratio* (T + R), *Total Score* only and *Rating Ratio* only. T + R means that the reputation thresholds Δm_i are adjusted using the same initial ratio. Total Score and Rating Ratio choose trustworthy users simply based on the highest total score or rating ratio, respectively. In other words, T + R adjusts the trust region without considering the relative importance of a given metric, while Total Score, and Rating Ratio determines trustworthiness based on only a single metric.

If we set the rating frequency $f = 10\%$ (Figure 6(a)), the average relative weight of total score to rating ratio $w_{S,R}$ turns out to be 1.72 (i.e., total score is more important than rating ratio). Therefore, we expect that the average accuracy for the Total Score method should be higher than that of Rating Ratio, which is indeed the case. Because the Rating Ratio method considers the factor that may be less relevant for reputation (i.e., the total number of ratings) in this case, the consideration thus interferes with the accuracy to choose trustworthy users. By combining total score and rating ratio, TuS shows a much higher accuracy than other methods, including T + R (29.55%, i.e., 64.27–34.72%, see Table 2). This is because the relative weight of each metric may shift during the simulation, so an adaptive weighting scheme such as TuS would perform better than a non-adaptive scheme such as T + R.

When the rating frequency increases to $f = 50\%$, as shown in Figure 6(b), the difference between Total Score and Rating Ratio becomes smaller. By combining considering both total score and rating ratio, TuS and T + R now achieve better accuracy than the other two naive schemes. The weight $w_{S,R} = 1.07$ shows that the difference in accuracy between TuS and T + S gets smaller as their relative importance becomes more similar.

When $f = 90\%$, all the schemes have enough samples in reputation ratings to distinguish users' trustworthiness after 1000 steps, as shown in Figure 6(c). All four schemes are able to converge to the highest accuracy of 100% as time goes. However, TuS converges the fastest among the schemes, i.e., convergence time for TuS is only 76.66% of the next-best scheme, T + R, and only 62.31% and 55.91% of the convergence time of Total Score and Rating Ratio (see Table 3). These results show that better selection accuracy can be achieved with a properly chosen scheme, and that TuS adapts to different conditions with a generally shorter convergence time.

Figure 6 TuS accuracy analysis in different rating frequency: (a) $f = 10\%$; (b) $f = 50\%$ and (c) $f = 90\%$ (see online version for colours)

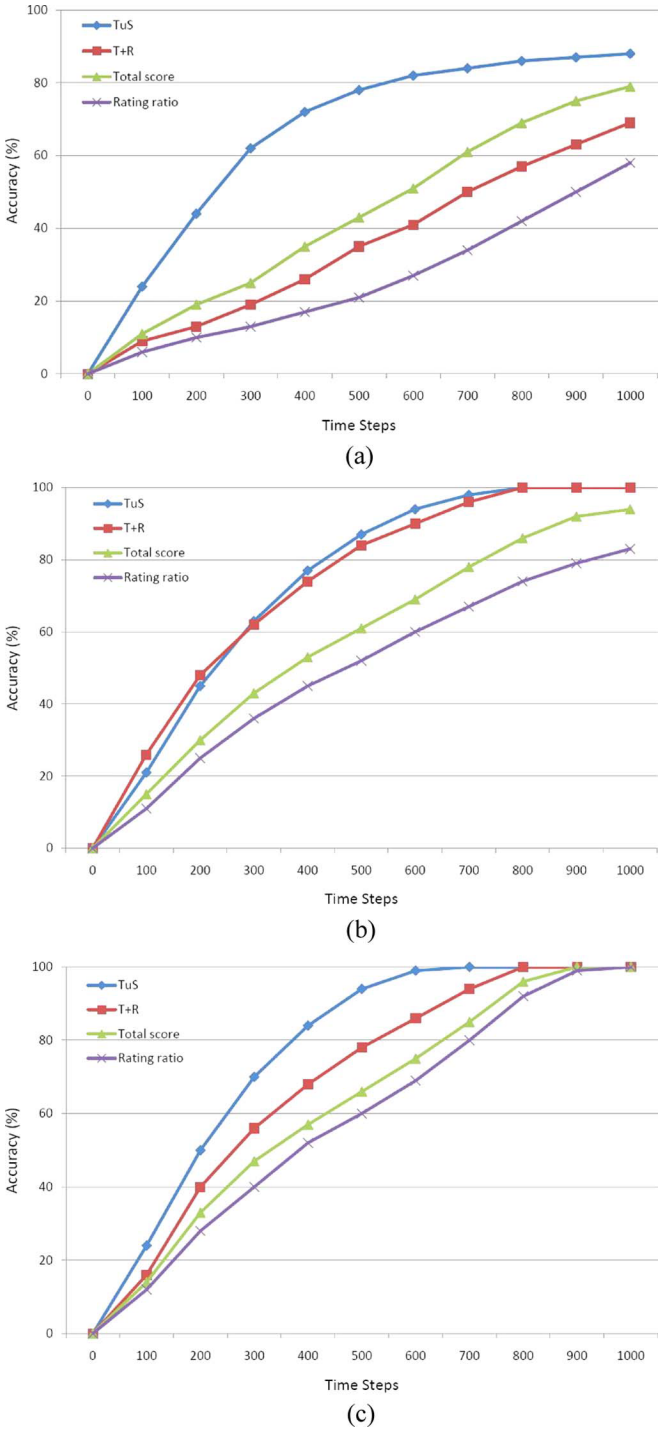


Table 2 Average reputation accuracy under different rating frequencies

Scenario		*Average reputation accuracy* (%)			
Rating frequency (f)	$w_{S,R}$	TuS	$T+R$	*Total score*	*Rating ratio*
10%	1.72	64.27	34.72	42.54	25.27
50%	1.07	71.36	70.9	56.45	48.36
90%	1.13	74.72	67.09	61.18	57.45

Table 3 Convergence time under different rating frequencies

Scenario	*Convergence time* (*time-steps*)			
Rating frequency (f)	TuS	$T+R$	*Total score*	*Rating ratio*
10%	–	–	–	–
50%	607	635	–	–
90%	496	647	796	887

'–' indicates when convergence is not achieved within 1000 steps.

6.2 *Effect of malicious behaviours*

In order to test the robustness of each scheme, we assume that some malicious users exist and act as follows:

- They give a score of -1 when meeting normal behaviour, and a score of 1 when seeing bad behaviour.
- Malicious users give a score of 1 to each other regardless of whether the encounter is normal or bad.

Malicious users are assumed to have the same misbehaviour probability between 0 and 1 as normal users and their activities are not detected (so that malicious acts can occur continuously to produce inaccurate ratings). In order to determine each scheme's robustness in face of malicious behaviours, we use Reputation Aggregation Error (RAE) to represent the departure of the chosen trustworthy users and the truly trustworthy ones under the interference from malicious users. RAE is defined as:

$$\mathrm{RAE} = \sqrt{\frac{\sum_{i=1}^{s}\left(\frac{r_i - \grave{r}_i}{r_i}\right)^2}{s}}$$

where r_i is the ranking of the chosen trustworthy user from all AOI neighbours, $\grave{r}_i$ is the true ranking based on misbehaviour probabilities, and s is the number of selected trustworthy users at the moment. RAE is a relative metric that reflects the difference between the rankings of all the selected users against the rankings of the truly trustworthy group of users. The smaller the RAE value, the closer the selected set is to the actual set of trustworthy users.

As the percentage of malicious users increases, Figure 7 shows the variation of average RAE from all trustworthy users under each scheme. We can find that although TuS sometimes provides lower accuracy than T + R (in Figure 6(b)), simulations show that the ranking of the chosen trustworthy users under TuS matches more closely to the true ranking. Figure 8 shows the change in average RAE with different total number of users when the percentage of malicious users is 10%. These simulations show that besides working under normal situations, TuS also has better performance than other schemes in face of malicious interference, even under different user sizes.

Figure 7 Effect of malicious users on RAE (see online version for colours)

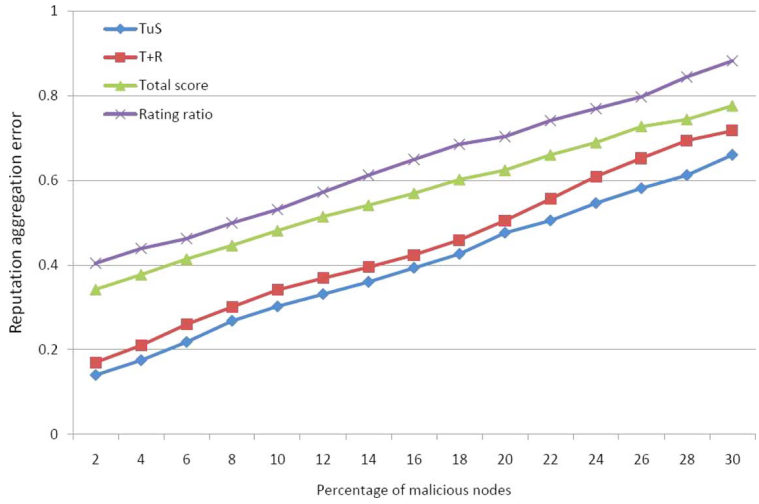
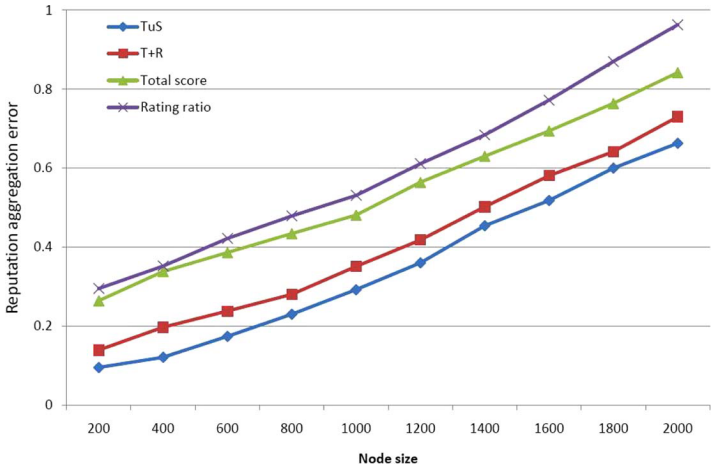


Figure 8 Effect of user size on RAE (see online version for colours)



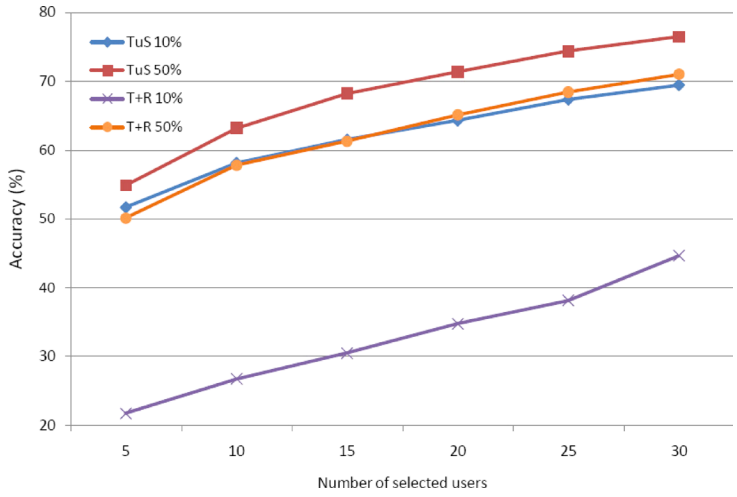
6.3 *Analysis on selectivity*

After evaluating the accuracy and robustness of TuS, another important aspect for a selection method is its *selectivity* – the ability of a scheme to discern qualified users

from a user pool with ever stricter criteria. For example, if a selection method is capable to identify only the best 30% in a group, then when the requirement is to identify the best 10%, the results returned may not be accurate enough for the stricter demand. We are thus also interested in how selective TuS can identify an ever-smaller group of trustworthy users. As the average AOI neighbour for 2000 nodes is about 80, we want to see how good TuS can choose 5, 10, 15, 20, 25, and 30 trustworthy users (i.e., 6.25, 12.5, 18.75, 25, 31.25 and 37.5% of AOI neighbours, respectively) from an average of 80 neighbours.

Figure 9 shows the accuracy of TuS and T + R to choose a specified number of trustworthy users under different rating frequencies (e.g., $f = 10\%$ and $f = 50\%$). We use T + R for comparison as it is the second best scheme from previous experiments. In general, we see that better average accuracy is achieved when more trustworthy users are required, because the penalty for incorrect selection is less (e.g., one missed selection produces a 20% inaccuracy when selecting 5 users, but only 3.3% for 30 users). We can also observe that TuS has better accuracy than T + R regardless of the number of selected users, especially when the rating frequency is low. That is, even with a small number of ratings, TuS is still good at identifying the top users.

Figure 9 Effect of different selectivity on TuS accuracy (see online version for colours)



We also see that reputation aggregation error decreases with increasing numbers of selected users from Figure 10, as less penalty for missed selection reduces the reputation aggregation error. Here, TuS also has less errors than T + R, which shows that TuS selects the real set of trustworthy users better than T + R.

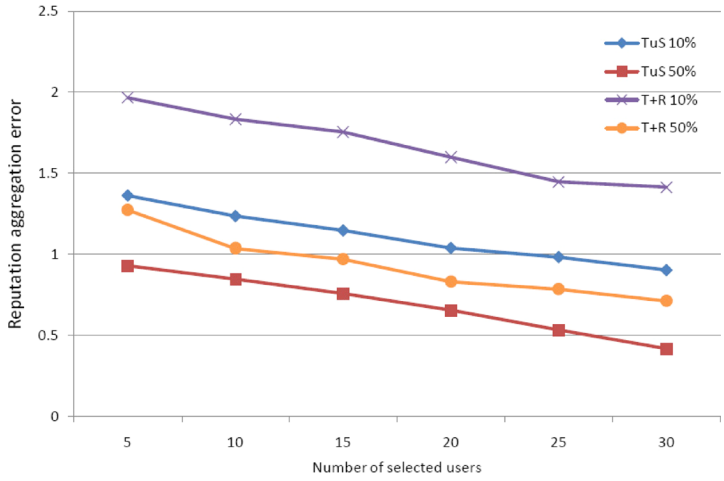
6.4 *Accuracy in higher dimensions*

In order to see how TuS performs under multiple dimensions (i.e., more than two reputation factors), besides total score and rating ratio, we consider one more reputation factor called *latest score*, $L(u)$, which is defined as follows:

$$L(u) = \sum_i LS(i, u)$$

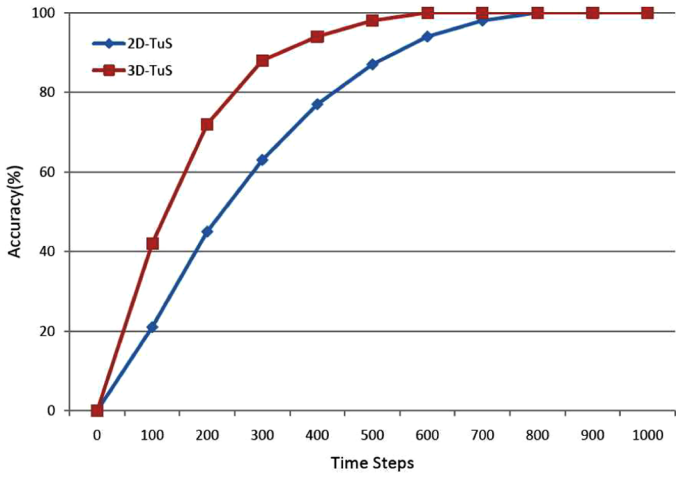
where $LS(i, u)$ represents the last (i.e., most recent) rating user i has given to user u. Unlike total score, latest score cannot accurately show the historically accumulated reputation, but it still shows the reputation most relevant to current user behaviour. Therefore, latest score is also an indicator for a user's trustworthiness.

Figure 10 Effect of different selectivity on TuS RAE (see online version for colours)



In Figure 11, we compare the accuracy of two dimensional (total score and rating ratio) and three dimensional (total score, rating ratio and latest score) reputation factors to choose trustworthy user by TuS. The simulation shows that 3D-TuS has 18% more average reputation accuracy than 2D-TuS, and also reduces 32% convergence time (see Table 4). Both accuracy and convergence time improve if we combine more relevant reputation factors for determining trustworthiness.

Figure 11 2D and 3D TuS accuracy with latest scores (see online version for colours)



Besides combining reputation factors directly related to user behaviour, we also try to see how factors unrelated to user behaviour would impact the evaluation results.

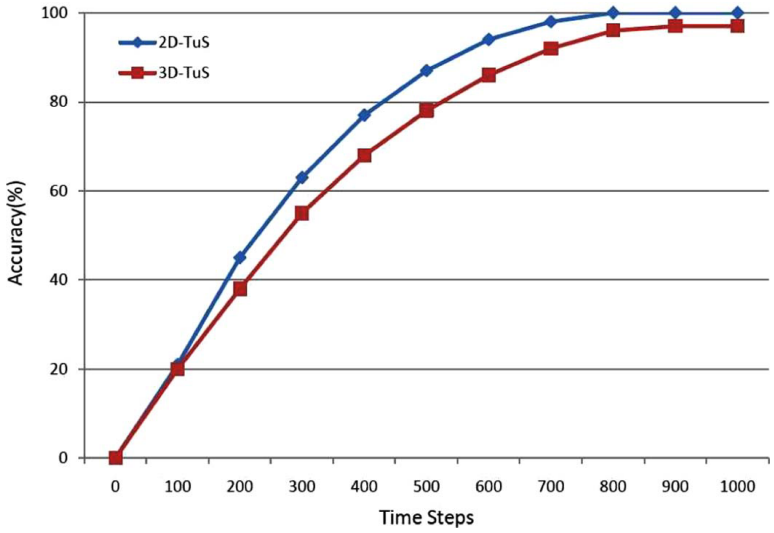
Table 4 Effect of three reputation factors on accuracy and convergence time

Scenario		$w_{S,R}$		$w_{(L,Ran),R}$	ARA (%)		*CT*	
Reputation factor	*f* (%)	2*D*	3*D*	3*D*	2*D*	3*D*	2*D*	3*D*
Latest score	50	1.07	1.11	1.24	71.36	81.27	607	413
Random score	50	1.07	0.88	0.14	71.36	65.81	607	679

*ARA: Average Reputation Accuracy; CT: Convergence Time.

Random score is a random fixed value between 0 and 1 assigned to each user. Its weight to other reputation factors is thus relatively lower than the weight of latest score. In Figure 12, we compare the accuracy of two dimensional and three dimensional TuS using random score as the third reputation factor. We can see that the effect of a random factor to both the accuracy and convergence time is low, and the average difference between 2D TuS and 3D TuS is below 5%, as the weight of the random factor is low under multivariate analysis. In order to filter less relevant reputation factors like random score, we can set a threshold ϵ. If the weight of a reputation factor less then ϵ, we can discard this factor to increase the accuracy for the whole system. Therefore, we can conclude that TuS can filter out less relevant reputation factors by lowering their weights, and by combing influential reputation factors, overall system performance would improve.

Figure 12 2D and 3D TuS accuracy with random scores (see online version for colours)



7 Concluding remarks

7.1 Discussions

Reputation evaluation: REPS uses direct rating between users as the main representation for reputations, where users give a simple score of $(-1, 0, 1)$ to indicate

their impressions for each other. It is thus very simple to perform rating and calculate one's reputation value. A user's reputation managers update reputation values directly and individually whenever they get a new reputation record from a rater. The rating right control allows users to identify which users can rate and ensures that only users who have interacted can rate each other. REPS thus provides a simple yet effective method to evaluate and compute reputation values.

Storage and query: Querying for reputation can be done efficiently as a querying user only needs to hash an user identifier, then it can ask some reputation managers directly. As the number of users increases in a system, the number of query overhead may also increase for a given user. However, the overhead of each reputation managers can be reduced by increasing the number of reputation managers M for each user, so that more reputation managers may share the querying load.

Security: The effect of malicious users on the system is reduced in REPS due to the mutual monitoring among users. As everyone can rate another user and update their scores when new situations occur, a cheating user will soon be rated very negatively if some misbehaviour is discovered. A cheater's reputation thus can be reduced rapidly and its privileges or responsibilities could be removed.

As reputation values are stored on multiple reputation managers, improper modifications by any single reputation manager is masked from the correctly maintained records in other reputation managers. Reputation manager misbehaviour thus will impact the system minimally. As reputations are stored and accessed at reputation managers instead of the rated user, users also cannot manipulate their own reputation values for unfair benefits.

7.2 Summary

REPS provides reputation management to support P2P MMOGs by allowing users to rate each other after some interactions, and select trustworthy nodes based on these ratings. Through the use of reputation managers, reputation records can be stored and accessed distributively without relying on a centralised server. Reputation values can thus be used in a scalable way. We also present TuS that chooses the trustworthy users by combining several reputation factors such as a user's total score and rating ratio, and adjusts each factor's weight to adapt for different scenarios by multivariate analysis. Dynamic adjustments of the trust region identify the minimum area that satisfies a given number of required trustworthy users, effectively selecting trustworthy nodes using the strictest criteria. Additional evaluations also show that TuS improves its performance if more relevant reputation factors are considered, yet it is unaffected by irrelevant, undistinguishing factors (e.g., malicious behaviours, random scores).

There are still some issues we have not yet fully explored in this paper, for example, the performance for the storage and query of reputation factors under DHT, and the integration of REPS to existing games or actual P2P MMOGs. Detection of cheating behaviours by malicious users is another potential issue. These future works would help to evaluate REPS better in real scenarios and potentially help to address the security issues hindering the realisation of P2P MMOGs.

References

Aberer, K. (2001) 'P-Grid: a self-organizing access structure for P2P information systems', *CoopIS 2001*, Vol. 2172, June, pp.179–194.

Aberer, K. and Despotovic, Z. (2001) 'Managing trust in a peer-to-peer information system', *Proc. ACM CIKM*, pp.310–317.

Atif, Y. (2002) 'Building trust in e-commerce', *IEEE Internet Computing*, Vol. 6, No. 1, pp.18–24.

Bharambe, A., Pang, J. and Seshan, S. (2006) 'AColyseus: a distributed architecture for online multiplayer games', *Proc. NSDI*, pp.155–168.

Buchegger, S. and Le Boudec, J-Y. (2004) 'A Robust reputation system for P2P and mobile ad-hoc networks', *Proceedings of SASN '04*, October.

Das, M.L., Saxena, A. and Phatak, D.B. (2006) 'Algorithms and approaches of proxy signature: a survey', *International Journal of Network Security*.

Dellarocas, C. (2001) 'Analyzing the economic efficiency of Ebay-like online reputation reporting mechanisms', *Proceedings of the 3rd ACM Conference on Electronic Commerce*, pp.171–179.

Ganeriwal, S. and Srivastava, M.B. (2004) 'Reputation-based framework for high integrity sensor networks', *Proc. Wksp Economics of P2P Systems*, June, pp.66–77.

Hu, S.Y., Chen, J.F. and Chen, T.H. (2006) 'VON: a scalable peer-to-peer network for virtual environments', *IEEE Network*, Vol. 20, No. 4, pp.22–31.

Hu, S-Y., Chang, S-C. and Jiang, J-R. (2008) 'Voronoi state management for peer-to-peer massively multiplayer online games', *Proc. 4th IEEE Intl. Workshop on Networking Issues in Multimedia Entertainment (NIME)*, pp.1134–1138.

Hyytia, E., Lassila, P. and Virtamo, J. (2006) 'A Markovian waypoint mobility model with application to hotspot modeling', *Proceedings of IEEE ICC 2006*, pp.979–986.

Ismail, R. and Josang, A. (2002) 'The beta reputation system', *Proc. 15th Bled Conf. on Electronic Commerce*, p.41.

Josang, A., Ismail, R. and Boyd, C. (2007) 'A survey of trust and reputation systems for online service provision', *Decision Support Systems*, Vol. 43, No. 2, June, pp.618–644.

Kamvar, S., Schlosser, M. and Garcia-Molina, H. (2003) 'The eigentrust algorithm for reputation management in P2P networks', *Proc. WWW*, May, pp.640–651.

Knutsson, B., Lu, H., Xu, W. and Hopkins, B. (2004) 'Peer-to-peer support for massively multiplayer games', *Proc. INFOCOM*, pp.96–107.

Lee, H.H. and Sun, C.H. (2006) 'Load-balancing for peer-to-peer networked virtual environment', *Proc. NetGames*, October, Article No. 4.

Lo, V., Zhou, D., Liu, Y., Dickey, C.G. and Li, J. (2005) 'Scalable supernode selection in peer-to-peer overlay networks', *Proc. HOT-P2P*, pp.18–27.

Mardia, K.V., Kent, J.T. and Bibby, J.M. (1979) *Multivariate Analysis*, Academic Press.

Mui, L., Mohtashemi, M., Ang, C., Szolovits, P. and Halberstadt, A. (2001) 'Ratings in distributed systems: a Bayesian approach', *Proc. Workshop on Information Technologies and Systems*.

Ratnasamy, S., Francis, P., Handley, M., Karp, R. and Shenker, S. (2001) 'Scalable content-addressable network', *Proc. ACM SIGCOMM*, pp.161–172.

Srivatsa, M., Xiong, L. and Liu, L. (2005) 'Trustguard: countering vulnerabilities in reputation management for decentralized overlay networks', *Proc. WWW*, pp.422–431.

Stoica, I., Morris, R., Karger, D., Kaashoek, F. and Balakrishnan, H. (2001) 'Chord: a scalable peer-to-peer lookup service for internet application', *Proc. ACM SIGCOMM*, pp.149–160.

Xiong, L. and Li, L. (2004) 'PeerTrust: supporting reputation based trust for peer-to-peer electronic communities', *IEEE TKDE*, Vol. 16, No. 7, pp.843–857.

Yamamoto, S., Murata, Y., Yasumoto, K. and inoru Ito, M. (2005) 'A distributed event delivery method with load balancing for MMORPG', *Proc. NetGames*, pp.1–8.

Yan, Y., Adel, E-A. and Ehab, A-S. (2007) 'Ranking-based optimal resource allocation in peer-to-peer networks', *Proc. INFOCOM*, pp.1100–1108.

Zhang, Y. and Fang, Y. (2007) 'A fine-grained reputation system for reliable service selection in peer-to-peer networks', *IEEE Transaction on Parallel and Distributed System*, Vol. 18, No. 8, pp.1134–1145.

Zhou, R. and Hwang, K. (2007) 'Apowertrust: a robust and scalable reputation system for trusted peer-to-peer computing', *IEEE Transaction on Parallel and Distributed Systems*, Vol. 18, No. 4, pp.460–473.

Websites

Second Life, http://secondlife.com/

World of Warcraft, http://www.worldofwarcraft.com/