Scalable Reputation Management for P2P MMOGs

Guan-Yu Huang[1], Shun-Yun Hu[2], Jehn-Ruey Jiang[3]
Department of Computer Science and Information Engineering
National Central University, Taiwan, R.O.C.
[1]aby@acnlab.csie.ncu.edu.tw [2]syhu@yahoo.com [3]jrjiang@csie.ncu.edu.tw

***Abstract*—Networked virtual environments (VEs) such as Massively Multiplayer Online Games (MMOGs) have become very popular in recent years. However, existing client-server architectures suffer from resource constrains when the number of concurrent users increases. Research on peer-to-peer virtual environments (P2P-VEs) thus tries to create more scalable and affordable VEs via the resource sharing of mutually cooperating clients. However, P2P approaches face the problem of client misbehavior where clients may not properly process the game rules. Without the monitoring and control from servers, the misbehavior could negatively affect a game's normal operations. In this paper, we present REPS, a distributed reputation management system for P2P-based MMOGs that allows trustworthy clients be identified. Based on the mutual rating and reputation query among users, REPS provides a scalable and reliable reputation mechanism that helps users to estimate the trustworthiness of others, so that subsequent grouping, trading, or superpeer selection decisions are made more reliably.**

I. Introduction

Massively Multiplayer Online Games (MMOGs) such as *World of Warcraft* [1] and *Second Life* [2], 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 10 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 virtual environments (P2P-VEs, e.g., SimMud [3], Colyseus [4], and VON [5]) have thus been proposed.

In client-server architectures, the server receives and processes all the user-generated events. This ensures that the action of each participant is 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-VEs 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 assume new identities when it has cheated for private benefits. Although, most players may not go to great length to cheat, as modifying the game code requires certain technical skills. But even if only a small number of users are successful at cheating, gameplay can still be disrupted seriously.

Fortunately, we observe that the nature of MMOGs is highly social, and users often invest large amount of time and energy to build their in-game persona to ensure their standings in the virtual world. Users often are also bounded by guilds or other social organizations, 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 reputation thus may be exploited to facilitate certain game operations, such as the selection of trustworthy clients for important functions. We have seen similar mechanism in online marketplaces such as eBay and 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 to make decisions on whether to interact with a particular peer, or to select trustworthy clients for assigning more responsibilities.

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. Reputation values are stored at some trustworthy neighbors called *trust nodes*, so that they may be accessed distributively without requiring a server. To select the trust nodes, we use *Neighbor Trust node Selection (NTS)* to choose trustworthy peers. NTS uses statistical regression method to choose trustworthy users, so that only users matching the strictest reputation criteria are chosen.

The rest of this paper is organized as follows. Section II provides background on reputation management and P2P-VEs. Section III presents our problem formulation and challenges in distributed reputation management. We describe the design of REPS in Section IV, and discuss its characteristics in Section V. Conclusions are given in Section VI.

II. Background

A. Reputation management

Recently, many P2P-based reputation systems have been proposed, often in the context of e-commerce (e.g., [6], [7], [8], [9], PeerTrust [10], Beta [11]). The goal of these systems is to compute the reliability of a peer and predict its future behavior of a specific identifier based on past interactions with the peer. The users are often buyers and sellers in an existing distributed or semi-distributed e-commerce environment. The reputation value represents a summary view for the user's

behavior, and can be used as the reference to warn or convince other users. By quickly identifying whether an user is trustworthy, interactions with *malicious users* who would cheat for private benefits can be avoided.

A peer's reputation value in these systems is calculated by collecting the local evaluations from other users. For example, in [12] and [8], the sum of the rating scores from every transaction is used to compute each user's reputation value. To make reputation values globally accessible and reliable, PeerTrust [10] normalizes the values by specific weights computed from each user's global reputation.

Some recent approaches like [13], [14], [15] and [16] 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 from past experience. [17] provides the QoS experience vectors to perform reputation evaluation on many levels to determine more precise reputation values.

Besides evaluating other users, users will also need to know someone's reputation value for various tasks. To query reputations in P2P environments, a decentralized method is often used to aggregate reputation scores from various places to compute the global reputation value. In a client-server architecture, the server stores all the reputation data, and users just need to query the server. However, in decentralized environments, often a P2P storage such as Chord [18], CAN [19], or P-Grid [20] is used to distributively store the reputation data. For example, [17] uses Chord to find the successors of each user A, where A stores all reputation records evaluated by other users on its successors. When another user B needs to know A's reputation, it will hash A's identifier to aggregate the reputation evaluations from A's successors. Similarly, [12] uses the identifier hash to discover successors for storing the reputation values with CAN or Chord.

Some other problems also exist in P2P reputation management, for example: how to distinguish honest from dishonest people, or to detect dishonest ones pretending to be honest [21]? How to filter extreme (i.e. too positive or negative) or fake reputation evaluations in order to ensure correctness? And how to prove that a reputation management is reliable for a given application? There are many works that discuss these general problems for reputation management (e.g., [6], [22] and [8]). We will discuss how REPS deal with these problems in P2P MMOG scenarios.

B. P2P-VEs

In P2P-VEs, every user has a visibility range called *area of interest* (or AOI, see Fig. 1). The AOI is often circular, and other users within the AOI are called the *AOI neighbors*. Users can exchange messages to comprehend the environment around them, and see the dynamic updates from other AOI neighbors. The key to scalable P2P-VEs is based on the fact that users have limited view within their AOI and only need to observe changes within the AOI. The scalability of the whole environment thus can be extended if each user only exchanges messages with its AOI neighbors, without going through the server.

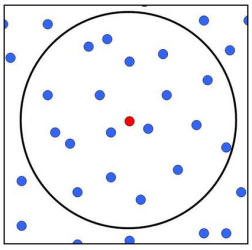

Fig. 1. Large circle is the AOI of the center user.

In some approaches (e.g., [3], [23], [24], [25]), the whole world is divided into serval disjoint sub-regions in order to manage information updates distributively. Some participants with better capacities are chosen as *superpeers* to relay information (e.g., position updates and the event notifications) for other users. Lo et al. [26] define a superpeer as a special role that can provide services to non-superpeers and describe several superpeer selection methods.

For the many P2P-VE schemes that adopt superpeers, whether the selected clients are trustworthy is essential for the system's. One of the implications for REPS thus is a reliable method to select trustworthy nodes that could assume important superpeer functions.

III. 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. The main problem is how to store the reputation scores on reliable peers and query them effectively. We first assume the following for our scenario:

1. Every user has a fixed AOI radius because most current MMOGs' use a fixed visual range, where users see each others 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 neighbors*, change periodically as users move around in the virtual world.

2. We assume that there exists some P2P-VE overlays that provide a list of AOI neighbors for each user (e.g., SimMud [3], Colyseus [4], VON [5]). So any user may connect and exchange messages directly with its AOI neighbors.

3. Two mutually visible users can rate each other with a score of positive, neutral, or negative (+1, 0, -1) based on their past interactions. A reputation record follows the form of (*rater, rated-user, evaluation*), where *rater* is the user who makes the rating, *rated-user* is the user evaluated by the rater and *evaluation* records the actual rating.

4. A user can only give a single rating to another user. However, the rating can be changed later at any time if the original rater wishes to.

5. We assume that if a user's reputation exceeds certain threshold, the probability of cheating is low, as more effort than others has been spent to build the reputation (Fig. 2).

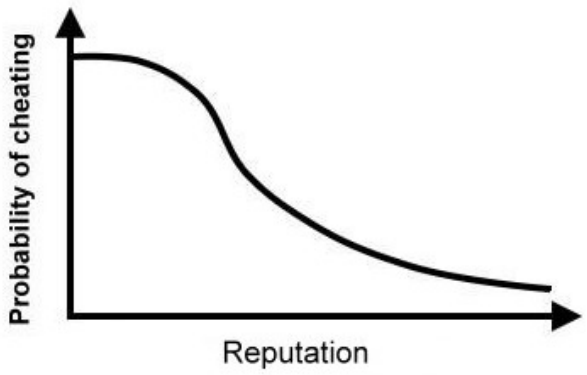

Fig. 2. Relation between probability of cheating and reputation value.

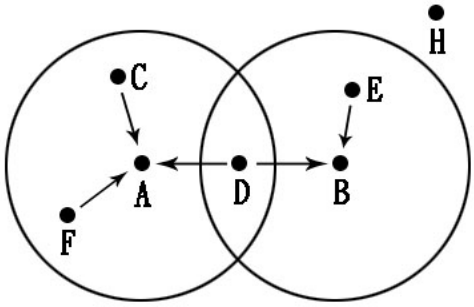

Fig. 3. The rating condition in REPS

To build a scalable reputation system for P2P MMOGs for the above scenario, we outline some challenges below:

Reputation evaluation User experiences are the basis for the reputation values in a reputation system. How to efficiently and precisely represent user perceptions thus is an important problem. Reputations are also meaningless if most users do not provide ratings. In MMOGs, players often focus more on the game itself than miscellaneous activity such as reputation evaluation, mechanisms to encourage user rating thus is needed. To provide suitable evaluations, we need a simple and efficient method that allows user evaluations be done conveniently, and reputation values be aggregated efficiently.

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 with a distributed method while avoiding any server or client overloads. To efficiently query reputation data from other users, two problems need to be addressed: how to find the users that store the reputation data, and how to collect the data with minimal delays.

Reliability There are more transactions like trading, talking and grouping [27] in MMOGs than in online auction sites, where over 90% of users have only transacted once [28]. Therefore, ensuring that a reputation system provides reliable and trustworthy information is very important. In P2P environments, users may modify the reputation data they keep for private benefits. This would cause misunderstandings among users and unbalanced gameplay. Prevention, detection, and recovery of cheating behavior thus are needed.

IV. Design of REPS

A. Local reputation evaluation

In REPS, users rate one another when they are within each others' AOI, because interactions can only occur with AOI neighbors. For example, in Fig 3, users C and F could rate A because they are within A's AOI. The rating should also occur with a probability related to the level of interactions. The more exciting the interactions and the more unfamiliar the users are to each other, the higher the probability for rating. The interacting users will generate a **rating right** authorized by the rated user to the potential rater, so that it can give a rating at some convenient time. The rating right is used to prevent some raters to rate users that they have never interacted with. The rating right contains the rated user's name and IP address and is recorded at the rater so that the rater can choose a time that is convenient to give ratings. For example, if user C rates user A with a score of 1, then a rating record of (C, A, 1) will be stored at A's trust nodes, who would update A's reputation based on A's last reputation value. Each user can have only one rating about another user, but can also change the rating when impression for the other user has changed. This way users may mutually monitor each other's behaviors.

B. Reputation storage and query

In order to scalably store and query the reputation records, a user identifies and chooses N trustworthy users as its **trust nodes** from its current and recent AOI neighbors (called the *potential neighbors*). Trust nodes are chosen from potential neighbors according to **Neighbor Trust node Selection (NTS)** that will be described next. Once reliable peers are found through NTS, they are recorded in a **trust list** containing the chosen trust nodes' identifiers and IP addresses. The trust list is stored at the user to allow easy inquiry by others. To give a rating to user A, raters first query A's trust list from A, and then send their ratings (i.e., reputation records) directly to all trust nodes to update A's reputation value.

When a user B is within user A's AOI, B can request for A's trust nodes in order to query for A's reputation value. User A would send its current trust list to B, where B randomly chooses n (where $1 \leq n \leq N$) trust nodes from the list to query. The chosen trust nodes will then respond the reputation value of A to B. The reason for asking reputation values from n trust nodes is to prevent any trust nodes from manipulating the stored reputation values. A reputation value is recognized only if it is returned by a majority of the trust nodes.

A user's potential neighbors would expire after a certain timeout, but may be renewed if the neighbors revisit a user's AOI. This has the effect that a trust node will also expire if it has not been a user's recent AOI neighbor. By limiting the time a node may be a trust node, users with high reputation

TABLE I
EXAMPLE OF PROPORTIONALITY MISREPRESENTATION

User	total score, TS(u)	number of ratings, V(u)	P(u)
A	30	100	0.3
B	9	10	0.9

can be saved from being always selected as trust nodes and bombarded with requests.

C. Neighbor Trust node Selection (NTS)

In REPS, each user requires N trust nodes that are chosen via *Neighbor Trust node Selection* (NTS). Many existing reputation systems use peer rating to devise the reputation (e.g. *PeerTrust* [10]), where an user i can give another user u a score $S(u,i)$ of either positive or negative, and the reputation is simply the summation of all scores, or a *total score* (TS). Other methods also exist in auction sites such as eBay or Yahoo Auctions, where a ratio $P(u)$ indicates the proportion between the total score $TS(u)$ and the total number of ratings $V(u)$. The higher the $P(u)$, the more trustworthy a given user u is.

$$TS(u) = \sum S(u,i)$$

$$P(u) = \frac{TS(u)}{V(u)}$$

But which is better? If A scores 30 out of 100 ratings, and B scores 9 out of 10 ratings. According to $TS(u)$, A is more trustworthy as its total rating is higher than B's. But the ratio $P(u)$ of B is higher than A's, which indicates that B may be more trustworthy. Yet since 100 people are willing to rate A and only 10 persons have rated B, the significance of A's rating may be higher. Some proportionality misrepresentations thus exist in existing approaches (Table I).

Ideally, we would like to combine the effects of both the total score and the ratio of positive rating, as they are both meaningful for a person's reputation. However, we do not know which is more important as it may differ across regions or MMOGs, where the willingness to rate can vary. We thus design *Neighbor Trust node Selection(NTS)* that combines both $TS(u)$ and $P(u)$ in a flexible way. A simple way to conceptualize NTS is in Fig. 4, where the x-axis represents all possible ratio values and the y-axis represents all possible total scores. There is also an area called *trust region* where a user u can be selected to become a trust node if its reputation point lies within the trust region (i.e. $P(u) > P_{bound}$ and $TS(u) > TS_{bound}$, where P_{bound} is between 0 and 1 and TS_{bound} is between the most negative and the most positive ratings). If we want to select N trust nodes, we can select N points (i.e. clients) from the trust region. If more trust nodes are needed, the area of the trust region is adjusted by changing P_{bound} and TS_{bound}. To adjust P_{bound} or TS_{bound}, we define the value m as the absolute value of the regression coefficient that represents the slope of the regression line for all points in the trust region, where $\bar{P}$ is the average $P(u)$ and $\bar{TS}$ is the average $TS(u)$ of all users within the trust region:

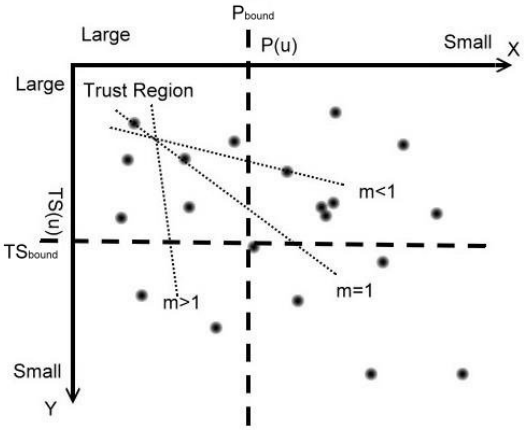

Fig. 4. Trust plane in REPS

$$m = \left| \frac{\sum (P(u) - \bar{P})(TS(u) - \bar{TS})}{\sum (P(u) - \bar{P})^2} \right|$$

The regression coefficient shows the pattern of distribution for all reputation points, and taking absolute values means that NTS only cares about the direction of the distribution but not the shape of the regression line. If $m > 1$, the trend for points in the trust region is towards $TS(u)$, its weight thus should be increased. If $m < 1$, it means that the point positions are tilting towards $P(u)$ in the trust plane, and NTS should increase the weight for $P(u)$. The actual adjustments ΔP_{bound} and ΔTS_{bound} for P_{bound} and TS_{bound} are adjustment ratios (i.e., they are percentages of the change in the values of P_{bound} and TS_{bound}), and depend on the value of m, where ΔP_{bound} / ΔTS_{bound} = m. NTS increases ΔP_{bound} and ΔTS_{bound} simultaneously with a fixed ratio m until the number of the candidate points matches the system demand. Likewise, ΔP_{bound} and ΔTS_{bound} could also decrease with the ratio m when the required trust nodes are less.

When the number of AOI neighbors is not large enough, trust nodes are chosen randomly until the number of users exceeds a threshold δ, then NTS is used again. When NTS is first used, we initialize m, P_{bound} and TS_{bound} to be 1.0, 1.0 and n where n is the number of current online users and the area of the trust region would be 0. We then reduce P_{bound} and TS_{bound} to extend the trust region by ΔP_{bound} / ΔTS_{bound} = 1 in order to find new trust nodes or remove old ones, as the set of potential neighbors change with time.

V. DISCUSSIONS

A. Reputation evaluation

REPS uses direct rating as the evaluation method, where users give a simple score (-1, 0, 1) to represent their impressions for each reputation evaluation. It is thus very simple to integrate one's reputation value. A user's trust nodes can update reputation values directly and individually whenever they get a new reputation record. The rating right control lets

users to recognize which users can rate them and ensures that only users who have interacted can rate each other. REPS thus provides a simple and effective method to represent and compute the reputation value.

B. Storage and query

Users in REPS store their reputation data on their trust nodes, this prevents a user from modifying its own reputations, as other users store and query reputation values directly with the trust nodes. A rated user provides only the trust list to a rater, and is not responsible to maintain his or her reputation value.

Querying reputation data can also be done efficiently as a querying user only needs to obtain the trust list, then it can ask some chosen trust nodes directly. As the number of users increases in a system, the number of queries may also increase for a given user. The overhead for each trust nodes can be reduced by increasing the number of trust nodes N for each user, so that more trust nodes may share the load of querying.

C. Reliability

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 misbehavior is discovered. The cheater's reputation will reduce rapidly and its privileges or responsibilities could be removed.

For the storage of reputation values, as they are stored on multiple trust nodes, improper modifications by any single trust node is masked from the correctly maintained records in other trust nodes. Trust node misbehavior thus will impact the system minimally and can be quickly detected. As reputations are stored and accessed at trust nodes instead of the rated user, users also cannot manipulate their own reputation values for unfair benefits. On the other hand, it is in a user's best interest to provide a list of trustworthy trust nodes to any potential rater, as it will surely wish that its reputation value is recorded and accessed objectively.

VI. Conclusion

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 trust nodes, reputation records can be stored and accessed distributively without relying on a centralized server. Reputation values can thus be built and used in a scalable way. We also present Neighbor Trust node Selection (NTS) that chooses the trust nodes by combining two intuitive factors (e.g., a user's total score and positive rating ratio) and adjusts each factor's weights to adapt for different scenarios. Dynamic adjustments of the trust region finds the minimum area that satisfies a given number of required trust nodes, effectively selecting trustworthy nodes using the strictest criteria. We plan to evaluate REPS's effectiveness via simulations as our future work.

References

[1] "World of warcraft," http://www.worldofwarcraft.com/.

[2] "Second life," http://secondlife.com/.

[3] B. Knutsson, H. Lu, W. Xu, and B. Hopkins, "Peer-to-peer support for massively multiplayer games," in *Proc. INFOCOM*, 2004, pp. 96–107.

[4] A. Bharambe, J. Pang, and S. Seshan, "Colyseus: A distributed architecture for online multiplayer games," in *Proc. NSDI*, 2006, pp. 155–168.

[5] S. Y. Hu, J. F. Chen, and T. H. Chen, "Von: A scalable peer-to-peer network for virtual environments," *IEEE Network*, vol. 20, no. 4, pp. 22–31, 2006.

[6] Y. Atif, "Building trust in e-commerce," *IEEE Internet Computing*, vol. 6, pp. 18 – 24, 2002.

[7] A. Josang, R. Ismail, and C. Boyd, "A survey of trust and reputation systems for online service provision," *Decision Support Systems*, vol. 43, no. 2, pp. 618–644, 2007.

[8] C. Dellarocas, "Analyzing the economic efficiency of ebay-like online reputation reporting mechanisms," in *Proc. 3rd ACM conference on Electronic Commerce*, 2001, pp. 171 – 179.

[9] K. Aberer and Z. Despotovic, "Managing trust in a peer-to-peer information system," in *In Proc. ACM CIKM*, 2001, pp. 310 – 317.

[10] L. Xiong and L. Li, "Peertrust: Supporting reputation based trust for peer-to-peer electronic communities," *IEEE TKDE*, vol. 16, pp. 843–857, 2004.

[11] R. Ismail and A. Josang, "The beta reputation system," in *Proc. 15th Bled Conference on Electronic Commerce*, 2002.

[12] S. Kamvar, M. Schlosser, and H. Garcia-Molina, "The eigentrust algorithm for reputation management in p2p networks," in *Proc. WWW*, 2003.

[13] S. Buchegger and J.-Y. L. Boudec, "A robust reputation system for p2p and mobile ad-hoc networks," in *Proc. Second Workshop on Economics of P2P Systems*, June 2004.

[14] L. Mui, M. Mohtashemi, C. Ang, P. Szolovits, and A. Halberstadt, "Ratings in distributed systems: A bayesian approach," 2001.

[15] S. Ganeriwal and M. B. Srivastava, "Reputation-based framework for high integrity sensor networks," in *Proc. SASN '04*, Washington, D.C., USA, October 2004.

[16] R. Zhou and K. Hwang, "Powertrust: 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, 2007.

[17] Y. Zhang and Y. Fang, "A fine-grained reputation system for reliable service selection in peer-to-peer networks," *IEEE Transaction on Parallel and Distributed Systems*, vol. 18, no. 8, pp. 1134–1145, 2007.

[18] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," in *Proc. ACM SIGCOMM*, 2001, pp. 149–160.

[19] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A scalable content-addressable network," in *Proc. ACM SIGCOMM, August 2001*.

[20] K. Aberer, "P-Grid: A self-organizing access structure for P2P information systems," *LNCS (CoopIS 2001)*, vol. 2172, pp. 179–194, 2001.

[21] M. Srivatsa, L. Xiong, and L. Liu, "Trustguard: Countering vulnerabilities in reputation management for decentralized overlay networks," in *Proc. 14th Intl World Wide Web Conf*, 2005, pp. 422–431.

[22] Y. Yan, A. El-Atawy, and E. Al-Shaer, "Ranking-based optimal resource allocation in peer-to-peer networks," in *Proc. INFOCOM*, 2007, pp. 1100–1108.

[23] S. Yamamoto, Y. Murata, K. Yasumoto, and M. Ito, "A distributed event delivery method with load balancing for mmorpg," in *Proc. Netgames*, 2005.

[24] H. Lee and C. Sun, "Load-balancing for peer-to-peer networked virtual environment," in *Proc. Netgames*, Oct. 2006.

[25] S.-Y. Hu, S.-C. Chang, and J.-R. Jiang, "Voronoi state management for peer-to-peer massively multiplayer online games," in *Proc. NIME*, 2008.

[26] V. Lo, D. Zhou, Y. Liu, C. GauthierDickey, and J. Li, "Scalable supernode selection in peer-to-peer overlay networks," in *Proc. of HOT-P2P*, 2005, pp. 18 – 27.

[27] K.-T. Chen, P. Huang, and C.-L. Lei, "Game traffic analysis: An MMORPG perspective," *Computer Networks*, vol. 51, no. 3, 2007.

[28] P. Resnick and R. Zeckhauser, "Trust among strangers in internet transactions: Empirical analysis of ebays reputation system," *The Economics of the Internet and E-Commerce,*, vol. 11, pp. 127–157, 2002.