1-1-2012

Towards formalizing a reputation system for cheating detection in peer-to-peer-based massively multiplayer online games

Willy Susilo
University of Wollongong, wsusilo@uow.edu.au

Yang-Wai Chow
University of Wollongong, caseyc@uow.edu.au

Rungrat Wiangsripanawan
rw26@uow.edu.au

Towards formalizing a reputation system for cheating detection in peer-to-peer-based massively multiplayer online games

Abstract

The rapidly growing popularity of Massively Multiplayer Online Games (MMOGs) has given rise to an increase in the number of players world wide. MMOGs enable many players interact together through a shared sense of presence created by the game. The Peer-to-Peer (P2P) network topology overcomes communication bottleneck problems associated with centralized client/server sys- tems. Thus, P2P-based MMOGs are seen as the way of the future, and many dierent P2P-based MMOG architectures have been proposed to date. However, many architectures are proposed in an ad hoc manner and enhancing the security of such systems is an elusive research problem. In this paper, we address this important issue by making the following contributions. Firstly, we formalize the notion of P2P-based MMOGs and demonstrate that existing P2P-based MMOG architectures can be unied using our model. To our knowledge, this is the rst time that this has been done in the literature. Secondly, we use our model to develop a real-time cheating detection mechanism to identify cheating players, which can be used to expose several MMOG cheating strategies. Finally, we propose a new reputation based system for P2P-based MMOGs to enhance the cheating detection process.

Keywords

era2014, towards, detection, formalizing, massively, multiplayer, online, games, reputation, system, peer, cheating

Disciplines

Engineering | Science and Technology Studies

Publication Details

Susilo, W., Chow, Y. & Wiangsripanawan, R. (2012). Towards formalizing a reputation system for cheating detection in peer-to-peer-based massively multiplayer online games. Lecture Notes in Computer Science, 7645 291-304.

Towards Formalizing a Reputation System for Cheating Detection in Peer-to-Peer-based Massively Multiplayer Online Games

Willy Susilo[1]*, Yang-Wai Chow[2], and Rungrat Wiangsripanawan**

[1] Centre for Computer and Information Security Research
[2] Centre for Multimedia and Information Processing
School of Computer Science and Software Engineering
University of Wollongong, Australia
{wsusilo, caseyc}@uow.edu.au
[3] Department of Computer Science, Faculty of Science
King Mongkut's Institute of Technology Ladkrabang, Bangkok, Thailand
kwrungra@kmitl.ac.th

Abstract. The rapidly growing popularity of Massively Multiplayer Online Games (MMOGs) has given rise to an increase in the number of players world wide. MMOGs enable many players interact together through a shared sense of presence created by the game. The Peer-to-Peer (P2P) network topology overcomes communication bottleneck problems associated with centralized client/server systems. Thus, P2P-based MMOGs are seen as the way of the future, and many different P2P-based MMOG architectures have been proposed to date. However, many architectures are proposed in an ad hoc manner and enhancing the security of such systems is an elusive research problem. In this paper, we address this important issue by making the following contributions. Firstly, we formalize the notion of P2P-based MMOGs and demonstrate that existing P2P-based MMOG architectures can be unified using our model. To our knowledge, this is the first time that this has been done in the literature. Secondly, we use our model to develop a real-time cheating detection mechanism to identify cheating players, which can be used to expose several MMOG cheating strategies. Finally, we propose a new reputation based system for P2P-based MMOGs to enhance the cheating detection process.

Keywords: Massively Multiplayer Online Games, Peer-to-Peer architecture, cheating detection, reputation system

1 Introduction

Massively Multiplayer Online Games (MMOGs) are online games which provide networked virtual environments where many players, typically ranging into the thousands, can interact with other players through a shared sense of presence created by the game. In MMOGs, players might physically be located all over the globe, but should be able to comfortably interact within the shared environment. The popularity and success of MMOGs has led to an increase in the number of users world wide. Consequently, the scalability of MMOG network architectures has become a key challenge that has to be addressed.

To date, many successful MMOGs are predominantly based on the Client/Server (C/S) network topology [8, 18]. In C/S systems, the centralized servers create a bottleneck as all communication must pass through the servers. This gives rise to a single point of failure, and expensive game servers have to be used to handle the large computational requirements of the system [12]. Furthermore, this centralized approach results in a huge amount of network traffic at the server-side, increases the communication latency between clients and inhibits the scalability of the system.

The Peer-to-Peer (P2P) network topology on the other hand overcomes the communication bottleneck problems associated with centralized servers by distributing computational load among

* This work is supported by ARC Future Fellowship FT0991397.
** This work was done when the author visited University of Wollongong, Australia.

the peers [12]. This allows for greater scalability, avoids the cost of expensive servers, and potentially reduces latency between interacting peers. As such, over the years researchers have proposed a variety of scalable P2P-based network architectures for MMOGs [1–4, 7, 9, 11, 14, 15, 20, 26, 27, 31].

However, the issue of security is a key concern that has to be dealt with before any P2P architecture can be used in the development of MMOGs, because cheating is rampant in MMOGs [32]. Many of the proposed P2P-based MMOG architectures are designed for scalability, but do not adequately handle the security of the system [1, 2, 4, 7, 9, 14, 15, 20, 27, 31]. In addition, several of the proposed mechanisms to address cheating in P2P MMOGs are ad hoc solutions that cannot be adopted in other systems [3, 11, 26].

Our Contributions. This paper addresses the important issue of security in diverse P2P-based MMOG network architectures. The aim of our work is as follows:

- to encapsulate different P2P-based MMOG architectures using a single unifying model,
- to provide a generic security mechanism which can be used to identify cheating players, and
- to develop a new reputation system that can be adopted by diverse architectures to enhance the cheating detection process.

We do this by formalizing the notion of P2P-based MMOG architectures, and show that our model can be instantiated on different P2P-based MMOG architectures. This will then be used as the basis for developing a real-time cheating detection mechanism, and we identify several MMOG cheating strategies that our security mechanism will be able to detect. Finally we propose the design of a new reputation-based system for P2P MMOGs which can be used in conjunction with our cheating detection mechanism. We stress that our main goal is to develop a new model for existing P2P-based MMOG architectures so that we can analyze their effectiveness based on a single unifying model.

2 Background

2.1 P2P-based MMOG Network Architectures

C/S architectures are currently the dominant approach adopted by MMOG developers. One of the primary reasons for its widespread use lies in the fact that it is easier to maintain security and to mitigate cheating in C/S systems as compared to P2P systems, because the server-side is a trusted system that is able to validate every action request sent by a client before carrying it out [8]. In addition, sensitive data is stored on the server and the clients are never given access to it [12]. Nevertheless, the downside is that this increases computational load at the servers and creates a communication bottleneck.

Security is much harder to maintain on P2P architectures as the data must be distributed and stored among the peers, which makes it a difficult but important problem to solve. While a variety of different P2P-based architectures for MMOGs have been proposed by the research community to overcome the various limitations of C/S architectures, these P2P-based architectures can generally be grouped into several broad categories. These will be described briefly in this section to provide the necessary background to our work. We adopt the terminology and definitions used in [8, 21] to represent the different types of P2P-based MMOG architectures. For a detailed survey, please refer to [8, 21].

ALM based Protocols
In the Application Layer Multicast (ALM) approach, game events and messages are distributed using standard ALM techniques. In many implementations, the virtual game world is partitioned

into subspaces or spatial regions. Each region is represented by a dedicated multicast group, and events within that region are sent to all relevant players in that region. Players only need to be informed of events happening within a certain range in the virtual environment. This range is known as the Area of Interest (AOI) and was a concept first introduced by Macedonia et al. [25]. In many cases, a player's AOI is fully inside a single region. However, if a player's AOI intersects the border between regions, he/she also has to subscribe to the other region's multicast group. Examples of the ALM based protocol can be found in [10, 11, 17, 20, 28, 30].

Supernode based Protocols
Similar to the ALM approach, the virtual game world is also divided into spatial regions. In some implementations, the region size is fixed [31], while in others, region sizes change dynamically based on player density in order to balance computational load [9]. For each region, a supernode, or superpeer, is selected and assigned as the coordinator for that region, effectively acting like a region server. The supernode is responsible for receiving all game event messages within the region and disseminating these to all players that are subscribed to that region. Supernode based protocols are used in [7, 16, 18, 31]. These supernode models are all based on the assumption that there is a way to choose a trustworthy node to act as the supernode for each region.

Mutual Notification based Protocols
This approach does not involve explicitly dividing the virtual game world into spatial regions. Instead, players send messages directly to other players within their AOI. Thus, message and event propagation delays are minimized. In mutual notification based protocols, players must be aware of all other players within their AOI. As such, players must depend on their neighbors for information regarding other players who have recently moved into their AOI. This protocol is used in [14, 15], where each player computes a Voronoi diagram based on all known neighbors. Whenever a player changes location, all neighbors must be notified so that they can updated their own local Voronoi diagrams. Neighbors are added to, or removed from, a player's notification list based on changes in Voronoi diagram information. Even though the virtual game world is not explicitly divided into spatial regions, in some sense the Voronoi diagrams still dynamically form non-uniform regions for mutual notification.

2.2 Existing P2P MMOG Reputation Systems

A number of researchers have proposed reputation systems for P2P MMOGs. Huang et al. [16] proposed REPS, a reputation management system for P2P MMOGs based on peer-rated reputations. In their approach, each user has a reputation value that is determined based on other users' subjective opinions formed during interaction between the peers. These reputation values are stored in trustworthy neighbours, akin to a supernode, that can be accessed distributively without the need of a server. Trustworthy nodes are chosen using a selection criteria based on the reputations. The problem with user assigned reputation systems is that they are subjective. This means that they can be abused by malicious or disgruntle players who can collude and assign negative ratings to their victims.

A non-subjective reputation system was proposed by Liu et al. [23]. In their approach, the reliability of peers were computed based on whether communication among peers were received timely and correctly in the process of synchronization. While this is a non-subjective approach where peer reputations were calculated based on their proposed algorithms, the aim of their approach was for quantifying the reliability of the peers in order to effectively distribute the computation and communication load among the peers, rather than for addressing cheating in MMOGs.

Xiang-bin et al. [30] proposed a cheating detection mechanism based on fuzzy reputation management for P2P MMOGs. In their work, a player's reputation is constantly updated based on the

player's game data history. To determine whether or not a player is a cheater, the player's reputation is compared against a certain threshold. Instead of using a fixed threshold that is common to all players, they propose a dynamically changing threshold for each individual player to minimize false detection ratio. However, their system must rely on central servers to store individual player reputations and thresholds.

3 Formal Model of P2P-based MMOG

In this section, we present a formal model of P2P-based MMOG systems. The formalization of a P2P-based MMOG system is essential in order for us to analyze the security of the system, and to add new functionality to the system. In the subsequent sections, we will demonstrate how to equip a P2P-based MMOG system with a cheating detection mechanism based on the formal model, as well as how to add a reputation-based system to determine cheating players.

3.1 High Level Description

The system described in this section will be a generic system that can be applied to the different P2P-based MMOG architectures previously discussed. While MMOG architectures are extremely complex systems that are made up of an amalgamation of diverse factors, many of these factors do not directly relate to cheating detection. As such, we will only focus on factors that will aid us in the task of detecting cheating peers. For example, P2P-based MMOGs may maintain a login server, whose main tasks include checking player subscriptions before allowing players to join the game, assigning a player to a region upon joining and providing initial information for the player to link with his/her peers. We will not consider such factors in our model, as our focus is on the P2P architecture that underlies the running of the in-game environment.

Typical P2P-based MMOG architectures are composed of a number of regions, whether fixed sized, dynamically changing with respect to player density or determined based on player AOIs. In many cases, a player's AOI is fully contained within a single region. At any given time, a player mainly resides in one of the defined regions. In which case, we say that the player is 'subscribed' to that region. If a player's AOI intersects the border of a neighboring region, then he/she has to also subscribe to the other region. On the other hand, the player 'unsubscribes' from a certain region if that region is no longer relevant to the player. Each region is identified with an ID, which may be implemented by simply using a collision resistant hash function.

3.2 Formal Definition and Model

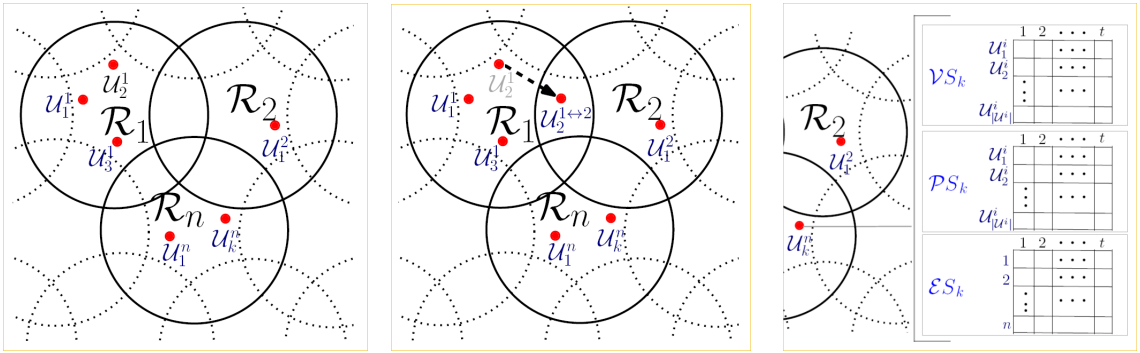
Setup

A P2P-based MMOG system comprises of n regions denoted as $\mathcal{R} = \{R_1, \cdots, R_n\}$. Each region, R_i, is identified by an identity ID_{R_i}. When a user $\mathcal{U}_i$ resides in a region R_j, we denote it as $\mathcal{U}_i^j$. Each region R_i contains m users at some stage, and therefore we denote it as $\mathcal{U}^i = \{\mathcal{U}_1^i, \mathcal{U}_2^i, \cdots, \mathcal{U}_m^i\}$. The total number of users in $\mathcal{U}^i$ is denoted as $|\mathcal{U}^i|$, which is equal to m in the above case. Each user in $\mathcal{U}^i$ is said to have 'subscribed' to R_i. This is illustrated in Fig. 1(a). When a user $\mathcal{U}_k^i \subset \mathcal{U}^i$ moves and interacts with R_j, we denote it as $\mathcal{U}_k^{i \leftrightarrow j}$, which implies[1] $\mathcal{U}_{|\mathcal{U}^j|+1}^j := \mathcal{U}_k^i$ and $\mathcal{U}^j := \mathcal{U}^j \cup \left\{\mathcal{U}_{|\mathcal{U}^j|+1}^j\right\}$. We note that this also means $|\mathcal{U}^j| := |\mathcal{U}^j| + 1$, since $\mathcal{U}_k^i$ has subscribed to R_j. When $\mathcal{U}_k^{i \leftrightarrow j_1 \leftrightarrow \cdots \leftrightarrow j_h}$, this implies that for each $\alpha \in \{j_1, \cdots, j_h\}$, $\mathcal{U}_{|\mathcal{U}_\alpha|+1}^\alpha := \mathcal{U}_k^i$ and $\mathcal{U}^\alpha := \mathcal{U}^\alpha \cup \left\{\mathcal{U}_{|\mathcal{U}_\alpha|+1}^\alpha\right\}$. This reflects

[1] This means that the last user in the region R_j, which is $\mathcal{U}_{|\mathcal{U}^j|+1}^j$, is set to be the new incoming user $\mathcal{U}_k^i$, and the size of the set $|\mathcal{U}^j|$ is increased by one.

the situation in which a player's position overlaps the borders of other regions. Fig. 1(b) shows a depiction of this. After the completion of $\mathcal{U}_k^{i\leftrightarrow j}$, meaning that the previous region is no longer relevant, $\mathcal{U}^i := \mathcal{U}^i \setminus \{\mathcal{U}_k^i\}$ must occur, and consequently $|\mathcal{U}^i| := |\mathcal{U}^i| - 1$. This represents the situation where $\mathcal{U}_k^i$ leaves R_i, and so $\mathcal{U}_k^i$ must 'unsubscribe' from R_i.

Let δ_{min} denote the minimum number of users required for a region to be formed. Where $\delta_{min} := 0$, means that there is *no* restriction on the minimum number of users.



(a) Users in the Regions (b) Movement of $\mathcal{U}_2^1$ to $\mathcal{R}_2$ (c) Data Storage of each peer

Fig. 1. Region, Nodes and Data Storage

Data Storage

In order to maintain the state of the game in P2P MMOGs, the peers need to store information about the state of the virtual game environment (e.g. Non-Player Characters (NPCs)), as well as the states of the other peers including his/her own. For supernode architectures, most of this information is stored in a single peer for each region. Three different lists need to be maintained by $\mathcal{U}_k^i$ to record information about the environment and the other peers in that region. These are to be implemented as queues. Let:

- $\mathcal{VS}_k := \{\mathcal{VS}_k^1, \cdots, \mathcal{VS}_k^t\}$ be the list of 'virtual player states'. These states contain the data of all player in the region. Player data might include Health Points (HP), experience points, level, money, items, attributes, etc.
- $\mathcal{PS}_k := \{\mathcal{PS}_k^1, \cdots, \mathcal{PS}_k^t\}$ be the list of 'physical states'. This records the players real world information such as connection speed, average message transmission time, latency, etc.
- $\mathcal{ES}_k := \{\mathcal{ES}_k^1, \cdots, \mathcal{ES}_k^t\}$ be the list of 'virtual environment states'. This is used to store information about non-player entities in the game environment, for example the state of NPCs.

The number of states maintained in the queue is represented by t. This means that if t is set to 10, the last ten states will be stored. As mentioned, this information is stored by $\mathcal{U}_k^i$, which might be a single peer in the case of a supernode architecture, or $|\mathcal{U}^i|$ peers otherwise. Since $\mathcal{VS}_k$ and $\mathcal{PS}_k$ encompasses the data of all players in the region, both virtual and physical, this is denoted as

$$\left(\mathcal{VS}_k^l, \mathcal{PS}_k^l\right) := \left(\{\mathcal{VS}_k^1, \cdots, \mathcal{VS}_k^{|\mathcal{U}^i|}\}, \{\mathcal{PS}_k^1, \cdots, \mathcal{PS}_k^{|\mathcal{U}^i|}\}\right)$$

we abuse the notation as $\mathcal{VS}_k^l[j]$, where $l = 1, \cdots, t$ and $j = 1, \cdots, |\mathcal{U}^i|$ to denote

$$\mathcal{VS}_k^l[j] := \mathcal{VS}_k^j$$

and $\mathcal{P}S_k^l[j]$, where $l = 1, \cdots, t$ and $j = 1, \cdots, |\mathcal{U}^i|$ to denote

$$\mathcal{P}S_k^l[j] := \mathcal{P}S_k^j$$

respectively.

Additionally, $\mathcal{U}_k^i$ needs to maintain the state of the virtual environment $\mathcal{E}S_k := \{\mathcal{E}S_k^1, \cdots, \mathcal{E}S_k^t\}$. For a region that has n non-player entities

$$\mathcal{E}S_k^l := \left(\mathcal{E}S_k^1, \cdots, \mathcal{E}S_k^n\right)$$

Therefore, in total user $\mathcal{U}_k^i$ in region R_i needs to store

$$t(2|\mathcal{U}^i| + n)$$

information to record the game environment states, as well as the virtual and physical states of all players in the region. Fig. 1(c) shows the information that is stored by the peers, or superpeers. Note that to deter cheating, in certain P2P MMOG implementations the user is not allowed to store his/her own state [24].

Communication
Let δ_{send} denote the set of peers that each user needs to report its states to for every single update cycle. For $\mathcal{U}_k^i$, δ_{send} is defined as $\delta_{send} := \{1, \cdots, |\mathcal{U}^i|\} \neq \left\{\mathcal{U}_k^i\right\}$ for R_i. At times the choice of the peers may be defined by proximity gathered from the physical states of other peers $\mathcal{P}S_k$. A special case happens when $\delta_{send} := \{1\}$, since each user needs to report its states to a designated user, $\mathfrak{R}_i$, in R_i. This designated user is often known as the supernode, which is selected using a selection scheme from among the users in R_i.

3.3 Instantiating the Model

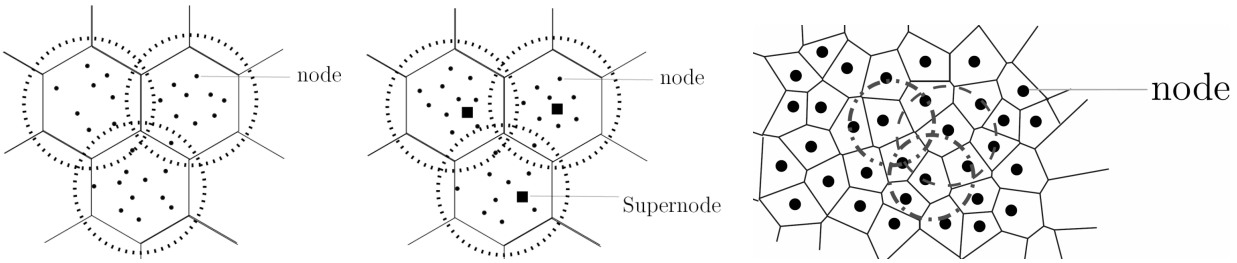
Here, we show how our formal model can be instantiated and applied to the existing types of P2P-based MMOG architectures that were described in section 2.1.

ALM based Protocols
In this architecture, the game world is typically divided into subspaces, and hence will be represented as a collection of R_i's in our model. A collection of players U^i reside in R_i and maintain their respective Area of Interests (AOIs). The way the user subscribes and unsubscribes to a region, based on their AOIs, is as per our model. Typical MMOGs divided the game world into square or hexagon based subspaces. Hence, Fig. 2(a) depicts how R_i in our model can be applied to hexagon subspaces. Similarly, our model can easily be applied to square based subspaces.

Supernode based Protocols
Supernode based protocols are similar to ALM based protocols in that they are region based. The difference being the existence of a responsible node, called the supernode in each subspace. This follows our model where $\delta_{send} = 1$, and hence, $\mathfrak{R}$ is the supernode. Refer to Fig. 2(b).

Mutual Notification based Protocols
Unlike the ALM and supernode based protocols, mutual notification based protocols do not explicitly divide the game world into rigid subspaces. Each player interacts with other peers in the system, when their proximities are closed to each other. Since they compute proximities using some method, for example by constructing a Voronoi diagram, these Voronoi regions can be clustered into the R_i regions represented in our model. Peers within R_i indicate the neighbors in which a node directly communicates with. This is illustrated in Fig. 2(c), the circles are examples of how

R_i would be formed around the Voronoi regions (note that only a few are shown to avoid over cluttering the Fig.). The difference between mutual notification based protocols as compared to the previous two protocols, is that this approach is based on dynamically changing regions, which are non-uniform. Hence, in Fig. 2(c) the circle sizes are non-uniform. There must be a minimum number of users required in order to define a region, namely δ_{min}.



(a) ALM based Protocols (b) Supernode based Protocols (c) Mutual Notification based Protocols

Fig. 2. Instantiations of Our Model

4 Cheating Detection

In this section, we present a cheating detection mechanism based on the formal model defined in the preceding section. Our method utilizes the existing data, required to run the P2P-based MMOG, that is already stored by the peers, or supernodes. The main principle adopted in the development of this approach is that a suspicious player's data will differ from the norm, which can be determined based on the past states that are stored and from a consensus among the other peers. In addition, once off cheats are rare because they do not give significant advantage to the cheater. As such, typical cheating strategies are continuously executed over many cycles, hence, cheaters can be identified by our cheating detection approach.

4.1 Detecting Suspicious Behavior

Before defining the cheating action, let

$$\beta \in \{1, \cdots t-1\}, \gamma_{\mathcal{ES}} := \texttt{Comp}_{\mathcal{ES}}(\mathcal{ES}_k^l, \mathcal{ES}_k^{l+\beta})$$

$$\beta \in \{1, \cdots t-1\}, j \in \{1, \cdots |\mathcal{U}^i|, j \neq k\}, \gamma_{\mathcal{VS}} := \texttt{Comp}_{\mathcal{VS}}(\mathcal{VS}_k^l[j], \mathcal{VS}_k^{l+\beta}[j])$$

$$\beta \in \{1, \cdots t-1\}, j \in \{1, \cdots |\mathcal{U}^i|, j \neq k\}, \gamma_{\mathcal{PS}} := \texttt{Comp}_{\mathcal{PS}}(\mathcal{PS}_k^l[j], \mathcal{PS}_k^{l+\beta}[j])$$

where $\texttt{Comp}_{\mathcal{ES}}, \texttt{Comp}_{\mathcal{VS}}, \texttt{Comp}_{\mathcal{PS}}$ define the comparison functions for virtual environment states, virtual player states and physical states, respectively. Depending on the nature of the game, these functions could be as simple as a substraction function, an XOR operation, or something more complex. To reduce a peer's computational load, these functions do not have to be executed every single cycle. Instead, they can be executed sporadically at random intervals, and only increasing the number of executions when a potentially suspicious player is detected.

These functions are used to identify in-game cheating behavior. The cheating action that can be identified by $\texttt{Comp}_{\mathcal{ES}}$ are those where the cheater tries to propagate false game environment states, for example, killing a strong NPC in a single blow, or falsifying the type or amount of an item

in the environment which is not currently owned by any of the peers. $\texttt{Comp}_{\mathcal{VS}}$ is used to identify whether a cheater tries to maliciously modify his/her own player status, for instance, inappropriately increasing his/her HP, moving through walls or moving at impossible speeds, duplicating an item that he/she owns, etc. $\texttt{Comp}_{\mathcal{PS}}$ on the other hand is used for detecting network cheats like trying to delaying event propagation to other peers, or changing an update messages' timestamp.

We define a function $\texttt{CheatDetect}(\mathsf{ID})$ that invokes the following:

- A user $\mathcal{U}_k^i$ in $\mathcal{R}_i$ suspects that there is a user, $\mathcal{U}_\ell^i$, in $\mathcal{R}_i$ that is cheating.
- $\mathcal{U}_k^i$ will multicast ID_ℓ to all other peers in $\mathcal{R}_i$, in order for them to check and verify this suspicion.
- All other peers in $\mathcal{R}_i$ will check and determine for themselves whether or not the user with ID_ℓ is cheating and multicast the same ID to all peers if this is found to be true. If the same ID is received more than once within a certain timeframe, it will be ignored to avoid network congestion due to message flooding.
- If a certain number of 'votes' given by the peers is obtained, a consensus is reached and the suspected user will be marked as a cheater. If no consensus is reached, the data will be ignored, and the situation will return to the status quo.

The function $\texttt{CheatDetect}(\mathsf{ID})$ accepts ID of the suspected user as its input, and it outputs either $\top$ or $\bot$, to indicate whether or not ID_ℓ^i is a cheater. Note that the voting system can be enhanced by using a reputation system. In other words, a vote from a peer with a higher reputation will have a greater weight in the overall decision. This will be elaborated in section 5, which describes our reputation system.

Specifically, a user $\mathcal{U}_k^i$ may suspect that $\mathcal{U}_\ell^i$ is cheating if $\gamma_{\mathcal{ES}} > \tilde{t}$ and/or $\gamma_{\mathcal{VS}} > \tilde{t}$ and/or $\gamma_{\mathcal{PS}} > \tilde{t}$, for a defined threshold $\tilde{t}$. When this condition occurs, $\mathcal{U}_k^i$ multicasts ID_ℓ to $\mathcal{U}_m^i \in \mathcal{U}^i$, where $m = \{1, \cdots, |\mathcal{U}^i|\}$, $m \neq \{k, \ell\}$. Upon receiving ID_ℓ, each $\mathcal{U}_m^i \in \mathcal{U}^i$ will check $\mathcal{ES}_m^j$ and $(\mathcal{VS}_m^l, \mathcal{PS}_m^l)$

$$\beta \in \{1, \cdots t-1\}, \gamma_{\mathcal{ES}} := \texttt{Comp}_{\mathcal{ES}}(\mathcal{ES}_m^j, \mathcal{ES}_m^{j+\beta})$$

$$\beta \in \{1, \cdots t-1\}, \gamma_{\mathcal{VS}} := \texttt{Comp}_{\mathcal{VS}}(\mathcal{VS}_m^l[\ell], \mathcal{VS}_m^{l+\beta}[\ell])$$

$$\beta \in \{1, \cdots t-1\}, \gamma_{\mathcal{PS}} := \texttt{Comp}_{\mathcal{PS}}(\mathcal{PS}_m^l[\ell], \mathcal{PS}_m^{l+\beta}[\ell])$$

and subsequently, if any of these comparisons indicate that U_ℓ^i has interfered with the update messages, tampered with the data or performed a suspicious action, ID_ℓ will be multicasted to $\mathcal{U}_m^i \in \mathcal{U}^i$, where $m = \{1, \cdots, |\mathcal{U}^i|\}$, $m \neq \{k, \ell\}$. Let $\bar{t}$ denote the threshold required to judge whether or not a user is a cheater. The value of $\bar{t}$ must be based on the total number of users in the region, i.e. $|U^i|$. If the number of ID_ℓ received is greater than $\bar{t}$, U_ℓ^i is identified as a cheater, and the output of $\texttt{CheatDetect}(\mathsf{ID}_k^i)$ will be $\top$. Otherwise, if when the number of ID_ℓ received given a number of cycles is less than $\bar{t}$, then its output will be $\bot$. Depending on the design of the game, a cheater may immediately be kicked from the game, and have his/her account suspended or banned from joining the game in future.

In a supernode architecture, $\mathcal{U}_k^i$ is a trusted peer and is responsible for identifying cheaters without the help of other peers. In this case, $\mathcal{ES}$, $\mathcal{VS}$ and $\mathcal{PS}$ are stored in the supernode. Thus, the supernode itself will run $\texttt{Comp}_{\mathcal{ES}}, \texttt{Comp}_{\mathcal{VS}}, \texttt{Comp}_{\mathcal{PS}}$, and collect this information over a number of cycles before determining the output of $\texttt{CheatDetect}(\mathsf{ID})$.

4.2 Cheating Techniques in P2P-based MMOG

Here, we elaborate several cheating strategies that can be launched by malicious players. We limit our discussion to cheating mechanisms that are related to P2P MMOG architectures. In addition, our method deals with in-game type cheating techniques. Therefore, we do not examine cheating

methods like login cheats or system administrator abuse, as they are not relevant to our discussion. We adapt the terminology of cheating strategies from [32].

Cheating by Exploiting Misplaced Trust
This is a common MMOG cheating mechanism that involves tampering with game code, configuration data, or both, and hence, requires reverse engineering on the client/peer's side. The malicious user, the cheater, can then modify the game client data to whatever value he/she wants. Alternatively, the cheater can also modify the game client in order to alter sensitive game states on the fly. This type of cheating can be detected in our architecture by observing past and present data, which may be stored in a supernode or shared among multiple peers. Specifically, if the cheater attempts to modify the current state of the game, the system can identify this suspicious behavior by comparing this with t previous states using the set of `Comp` functions. This detects cheats like increasing a player's attributes, "speed hacks", "duping items", etc.

Cheating by Modifying Client Infrastructure
The cheater can modify the client infrastructure such as device drivers in his/her operating system. By doing this, for instance, the cheater can make walls transparent (this is known as "wall hack"). In previous work by Laurens et al. [22], they detected "wall hacks" by incorporating the concept of a *trace*. Essentially, the *virtual states* of the each player has to be observed to determine precisely what the player is looking at. The frequency of illegal traces can identify the case where a player keeps looking at objects that the player cannot actually see. Alternatively, suspicious behavior can be determined if a player continually 'stares at a wall', because this is invisible to him/her. Our detection mechanism can handle this type of cheats by storing and observing the frequency of suspicious behavior over a number of cycles.

Timing Cheating
In this type of cheating mechanism, the cheating player choose to delay his/her own move until he/she knows all the opponents' moves, and hence, gaining a huge advantage. This type of cheating strategy can be detected by using our cheating detection mechanism as information about the *physical states* for each player is recorded. By observing the physical states of each peer, our system can determine artificially induced delays or when message timestamps are modified.

Cheating by Exploiting Lack of Secrecy
This type of cheating strategy is performed by illegally accessing game data (or states). This situation can arise in our model if the cheater can somehow obtain the contents of the queues. In general, combating this is straightforward as the state information for each player can be encrypted with a symmetric algorithm, such as AES. Assuming the security of the algorithm is hard (which is the case for the state-of-the-art AES algorithm), this cheating strategy will be rendered ineffective.

5 Adding a Reputation System to P2P-based MMOGs

This section presents our reputation system that is to be embedded into the P2P-based MMOG architecture. We employ a reputation system that is inspired by EigenTrust [19], X^2Rep [6] and X^{2BT}Rep [33] which have been designed for use in P2P networks. Nevertheless, we should stress that the reputation systems proposed in P2P networks cannot directly be used in P2P-based MMOGs. This is because in many large-scale cutting-edge MMOGs, the game itself requires tremendous computational resources to run and all this computation has to be performed in real-time. Any delays, due to network latency or processing load, can severely impact the players' in-game experience.

Therefore, the aim of any cheating detection mechanism or reputation system in MMOGs is to provide a decent level of security without over burdening the system. On the other hand, the

main goal of reputation systems in traditional file-sharing P2P networks is to measure the validity of the download resources offered by determining the level of trust of the other peers; Real-time performance and computational load are not the main driving factors.

Hence, we need to build a new reputation system that is suitable for P2P-based MMOG systems. One of the main principles that we employ in the development of our reputation system, is that in MMOGs the main purpose of each individual peer is to protect the player himself/herself, as opposed to trying to protect the entire system.

5.1 High Level Idea

The main idea underlying the reputation system is as follows. Each user is equipped with a list of reputations of all peers in the region. The user will not store his/her own reputation. When a new user joins a region R_i, the user is given a default reputation Δ. Note that Δ cannot be zero, since a reputation of zero will prevent the user from join the MMOG game in the first place (this is known as "cold start" in P2P-based reputation system). Upon joining R_i, the user contacts the peers in R_i to obtain the reputation of other peers. When used in cheating detection, a peer's voting weight can be adjusted based on the value of the peer's reputation. Hence, a user with higher reputation will contribute more weight towards determining whether or not another peer is a cheater. In a similar manner, once a user moves into another region, the user needs to contact the peers in that region to acquire the reputation values of the other peers. The user's reputation in that new region will be calculated based on the reputation given by the peers in the region that he/she just left.

This can easily be adapted to a supernode system, where the reputation list would be computed and stored at the supernode. When a user move to a different region, the new region's supernode needs to get the user's reputation from the previous region's supernode. If a supernode changes region, a new supernode is selected and the list is transferred to the new supernode.

5.2 System Design

Each user $\mathcal{U}_k^i$ in R_i is equipped with a list of reputations:

$$\mathsf{Rep}_k^i := \forall_{\ell=1,\cdots,|\mathcal{U}^i|,\ell\neq k}\{\mathsf{Rep}_\ell^i\}$$

We use Rep_ℓ^i to denote the reputation of user $\mathcal{U}_\ell^i$ who resides in region R^i. The list is kept by user $\mathcal{U}_k^i$ to represent user $\mathcal{U}_k^i$'s view on the other peers.

User Joining a Region
When user $\mathcal{U}_k$ enters region R_i (hence, $\mathcal{U}_k^i$), $\mathcal{U}_k^i$ queries δ_{send} peers to acquire the reputation of all other peers in order to fill Rep_k^i. When there is more than one response received, Rep_k^i is filled with the average of the responses. If there exists $\forall_{\ell=1,\cdots,|\mathcal{U}^i|,\ell\neq k}\{\mathsf{Rep}_\ell^i\} = \emptyset$, then the peer needs to query other peers in R_i. At the end of this process, $\mathcal{U}_k^i$ acquires a complete Rep_k^i. Other peers $\mathcal{U}_j^i \in \mathcal{U}^i$ will assign to $\mathcal{U}_k^i$ with a reputation of Δ.

Moving to Another Region
When $\mathcal{U}_k^{i\leftrightarrow j}$, $\mathcal{U}_k^i$ queries δ_{send} peers in R_j to acquire all other peer reputations and gather Rep_k^j. Additionally, since $\mathcal{U}_k^i$ has now moved to R_j (hence, $\mathcal{U}_k^j$), the peers in R_j will have to update their records to contain $\mathcal{U}_k^j$'s reputation. However, since $\mathcal{U}_k^j$ does not store his/her own reputation, the peers in R_j will have to acquire $\mathcal{U}_k^j$'s reputation from δ_{send} users in R_i. Similarly, the same process is done when $\mathcal{U}_k^{i\leftrightarrow j_1\cdots j_h}$.

Using the Reputation

The reputation information is used to enhance the quality of the votes given by the peers. When $\mathcal{U}_k^i$ suspects that $\mathcal{U}_d^i$ is cheating, $\mathcal{U}_k^i$ invokes $\texttt{CheatDetect}(\mathsf{ID}_d)$. Upon receiving the ID-s from $\mathcal{U}_\ell^i$, where $\mathcal{U}_\ell^i \in \mathcal{U}^i, \ell \neq \{d, k\}$ computes

$$Res = \sum_{\forall \ell \in \{1,\cdots,|\mathcal{U}^i|\}, \ell \neq \{d,k\}} \mathsf{Rep}_\ell^i \times \texttt{IsID}(\mathsf{ID_d}, \mathsf{ID}_\ell).$$

The function $\texttt{IsID}(\mathsf{ID}_d, \mathsf{ID}_\ell)$ will return 1, if ID_d has been returned by user ID_ℓ, or 0 otherwise. If $Res > \bar{t}$, for a threshold $\bar{t}$ as defined earlier in section 4.1, then the user $\mathcal{U}_d^i$ will be marked as a cheater. We call this system a credibility algorithm in P2P-based MMOGs. When a user is identified as a cheater, then his/her reputation is marked as 0.

Reputation Update
If a user $\mathcal{U}_\ell^i$ submits ID_d during a cheating detection phase, and ID_d is eventually marked as a cheater (which means, that ID_d's reputation is marked as 0), then $\mathcal{U}_\ell^i$'s reputation that is stored on the other peers should be updated as $\mathsf{Rep}_\ell^i := \mathsf{Rep}_\ell^i + \xi$. The value ξ is used to increase the reputation of $\mathcal{U}_\ell^i$, because the user correctly identified a cheater $\mathcal{U}_\ell^i$ should now be seen as a more trustworthy peer. A typical value that is used during implementation is 0.05, which is similar to the reputation system used in BitTorrent (X^2BT-Rep [33]).

Conversely, if a user $\mathcal{U}_\ell^i$ submits ID_d during a cheating detection phase, but ID_d is eventually declared to be a non-cheater (i.e. the threshold $\bar{t}$ criteria is not met), then $\mathsf{Rep}_\ell^i := \mathsf{Rep}_\ell^i - c\xi$, where $c \in \{2, \cdots\}$. This essentially means that if a user votes wrongly, then the 'penalty' given is linear to ξ. In contrast, if a correct vote is cast, then the reputation is increased by ξ. This is because a user will normally only vote incorrectly if his/her data diverges significantly from the norm, which indicates that the user is less trustworthy, possibly indicating that he/she has tampered with the system.

While it is conceivable that a player could hack his/her system to avoid submitting votes entirely, thereby never having other peers increase or decrease his/her reputation, this in no way benefits the player as there is nothing that the player gains from withholding votes. Moreover, in a MMOG system which hosts thousands of players, the chances of having many players hacking their system to withhold votes for no apparent benefit, is extremely remote to have any significant impact on the reputation system. However, it is possible that a player, or a group of players, might be able to reverse engineer their system to maliciously vote against other players in order to kick them out of the game. This is discussed in the section below.

5.3 Security Considerations

A number of security issues that occur in P2P-based MMOG systems are discussed here. In particular, we focus our discussion on security issues faced by P2P MMOG architectures with our reputation system in place.

Pseudospoofing
The idea of this attack is as follows. A malicious user registers with the system, behaves in a corrupt manner for a while and then re-register with the system (by quitting and rejoining the game) to re-initialize his/her reputation. With a re-initialized reputation the user can now falsely accuse another user of being malicious during a cheating detection phase.

Specifically, let $\mathcal{U}_{\texttt{corrupt}}^i$ join R^i. The initial reputation assigned to $\mathcal{U}_{\texttt{corrupt}}^i$ is Δ. After behaving maliciously, Δ will be significantly reduced. Now when $\mathcal{U}_{\texttt{corrupt}}^i$ re-registers as $\mathcal{U}_c^i$, the user's reputation will be re-initialized to Δ. With this reputation, $\mathcal{U}_c^i$ can now start to vote maliciously and accuse others of cheating. However, $\mathcal{U}_c^i$ will not gain any significant benefit from this action,

because assuming the value Δ that has been chosen is sufficiently small and $|\mathcal{U}^i|$ is sufficiently large, then this malicious activity will be ineffective in our reputation system algorithm, as the sum of the other votes will out-weight $\mathcal{U}_c^i$'s vote.

Reputation Spoofing
In this type of attack, the malicious user attempts to find some vulnerabilities in the reputation algorithm and spoof the reputation values. This may be achieved by conducting reverse engineering on the software. Using our reputation system, this attack is ineffective as the reputation for a user is *not* determined nor stored by the user himself/herself, but rather is determined based on the other peers' view of this particular user. The peers in the system gain reputation when voting correctly, as their reputation will increase. Hence, by providing this mechanism, only the peers that vote correctly will benefit from this reputation system.

Whitewashing Attack
This is the common attack on the eBay online transaction system. Essentially, this means that a malicious user actively participates in the system by providing genuine items, but sometimes provides a small number of inferior goods to be sold to others [13, 29]. In our scenario, consider a user $\mathcal{U}_k^i$ who is actively involved in the system by voting correctly whenever asked. Nevertheless, occasionally, this user also deliberately votes incorrectly. Note that our reputation system uses the formula $\mathsf{Rep}_\ell^i := \mathsf{Rep}_\ell^i + \xi$ to increase the value of the reputation, whilst the formula $\mathsf{Rep}_\ell^i := \mathsf{Rep}_\ell^i - c\xi$, where $c \in \{2, \cdots\}$ is used to decrease the value of the reputation. The range of c starts from 2, which means that the 'penalty' is more severe for incorrect votes as compared to the reward given by voting correctly. For example, if $c = 10$, this refers to the case the reputation gained from 10 correct votes will completely be negated by a single incorrect vote. This way, whitewashing attacks will be ineffective.

Reputation Attacks by Collectives
This attack is achieved when malicious users know each other and they collaboratively seek to harm the system by acting as a group. An example of this kind of attacks in the P2P system is known as *shilling*. Instead of creating multiple identities as in the pseudospoofing attack, the attackers maintain several true identities to influence the voting process. This is protected by the parameter $\bar{t}$ that controls the value of the threshold in the system. Unless all users are malicious, which will make the system totally ineffective, this attack is prevented by our reputation system.

6 Conclusion

The increasing number of MMOG players world wide has exposed the problem of scalability in C/S based MMOG architectures. As such, many researchers have proposed scalable P2P-based MMOG architectures. However, cheating is a common occurrence in MMOGs and this is not adequately handled in many of these proposed P2P systems. The security of a MMOG network architecture is an important issue that has to be addressed before the architecture can be used in the development of a game. Many proposed security mechanisms for P2P MMOG systems are specialized solutions and cannot be realized on other systems.

This paper addresses this vital issue by formalizing the notion of diverse P2P-based MMOG architectures into a single unifying model. We demonstrated that our formal model can be used to instantiate different P2P-based MMOG architectures. Based on this model, this paper presents a generic cheating detection mechanism that can be used to detect a number of different MMOG cheating strategies. In addition, we described a reputation system that can be used to further enhance the cheating detection process and describe its robustness against a number of attacks.

References

1. M. Assiotis and V. Tzanov. A distributed architecture for mmorpg. In *Netgames'06*, 2006.
2. A. Bharambe, J. Pang, and S. Seshan. Colyseus: A distributed architecture for online multiplayer games. In *ACM/USENIX NSDI 2006*, 2006.
3. F. R. Cecin, R. Real, R. D. O. Jannone, C. Fern, and O. R. Geyer. Freemmg: A scalable and cheatresistant distribution model for internet games. dsrt. In *in Proc. of International Symposium on Distributed Simulation and Real-Time Applications*, pages 83–90, 2004.
4. L. Chan, J. Yong, J. Bai, B. Leong, and R. Tan. Hydra: A massively-multiplayer peer-to-peer architecture for the game developer. In *Netgames'07*, 2007.
5. W. chang Feng, editor. *Proceedings of the 3rd Workshop on Network and System Support for Games, NETGAMES 2004, Portland, Oregon, USA, August 30, 2004*. ACM, 2004.
6. N. Curtis, R. Safavi-Naini, and W. Susilo. X2rep: Enhanched trust semantics for the xrep protocol. *The Second Applied Cryptography and Network Security (ACNS04), Lecture Notes in Computer Science 3089, Springer-Verlag*, pages 205 – 219, 2004.
7. L. Fan, H. Taylor, and P. Trinder. Mediator: A design framework for p2p mmogs. In *Netgames'07*, 2007.
8. L. Fan, P. Trinder, and H. Taylor. Design issues for peer-to-peer massively multiplayer online games. 2009.
9. C. GauthierDickey, V. M. Lo, and D. Zappala. Using n-trees for scalable event ordering in peer-to-peer games. In W. chi Feng and K. Mayer-Patel, editors, *NOSSDAV*, pages 87–92. ACM, 2005.
10. C. GauthierDickey, D. Zappala, and V. M. Lo. A fully distributed architecture for massively multiplayer online games. In chang Feng [5], page 171.
11. T. Hampel, T. Bopp, and R. Hinn. A peer-to-peer architecture for massive multiplayer online games. In *Netgames'06*, 2006.
12. A. Harwood and S. Kulkani. Delay sensitive identity protection in peer-to-peer online gaming environments. In *Proceedings of the 13th International Conference on Parallel and Distributed Systems - Volume 02*, pages 1–6, Washington, DC, USA, 2007. IEEE Computer Society.
13. K. Hoffman, D. Zage, and C. Nita-Rotaru. A survey of attack and defense techniques for reputation systems. *ACM Comput. Surv.*, 42:1:1–1:31, December 2009.
14. S.-Y. Hu, S.-C. Chang, and J.-R. Jiang. Voronoi state management for peer-to-peer massively multiplayer online games. In *New Interfaces for Musical Expression*, 2008.
15. S. Y. Hu and G. M. Liao. Scalable peer-to-peer networked virtual environment. In *Network and System Support for Games*, pages 129–133, 2004.
16. G.-Y. Huang, S.-Y. Hu, and J.-R. Jiang. Scalable reputation management for p2p mmogs,. In *Proceedings First International Workshop on Massively Multiuser Virtual Environments*, pages 4–8, 2008.
17. T. Iimura, H. Hazeyama, and Y. Kadobayashi. Zoned federation of game servers: a peer-to-peer approach to scalable multi-player online games. In chang Feng [5], pages 116–120.
18. P. Kabus and A. P. Buchmann. Design of a cheat-resistant p2p online gaming system. In *Proceedings of the 2nd international conference on Digital interactive media in entertainment and arts*, DIMEA '07, pages 113–120, New York, NY, USA, 2007. ACM.
19. S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina. The eigentrust algorithm for reputation management in p2p networks. In *WWW*, pages 640–651, 2003.
20. B. Knutsson, H. Lu, W. Xu, and B. Hopkins. Peer-to-peer support for massively multiplayer games. In *Infocom*, 2004.
21. S. Krause. A case for mutual notification : A survey of p2p protocols for massively multiplayer online games. In *Netgames'08*, 2008.
22. P. Laurens, R. F. Paige, P. Brooke, and H. Chivers. A novel approach to the detection of cheating in multiplayer online games. In *Engineering Complex Computer Systems, 2007. 12th IEEE International Conference on*, pages 97 –106, july 2007.
23. S. Liu, J. Li, and X. Wang. Local reputation for p2p mmog design. In *PDCAT'07*, pages 523–528, 2007.
24. M. Merabti and A. E. Rhalibi. Peer-to-peer architecture and protocol for a massively multiplayer online game. In *Globecom 2004 Workshops*, 2004.
25. M. J. Z. M.R. Macedonia, D. R. Pratt, D. P. Brutzman, and P. Barham. Exploiting reality with multicast groups. *IEEE Computer Graphics and Applications*, 15:38–45, 1995.
26. M. Picone, S. Sebastio, S. Cagnoni, and M. Amoretti. Peer-to-peer architecture for real-time strategy mmogs with intelligent cheater detection. In *Proceedings of the 3rd International ICST Conference on Simulation Tools and Techniques*, SIMUTools '10, pages 8:1–8:8, ICST, Brussels, Belgium, Belgium, 2010. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).
27. A. E. Rhalibi, M. Merabti, and Y. Shen. Aoim in peer-to-peer multiplayer online games. In *ACE'06*, 2006.

28. K.-H. Vik. Game state and event distribution using proxy technology and application layer multicast. In H. Zhang, T.-S. Chua, R. Steinmetz, M. S. Kankanhalli, and L. Wilcox, editors, *ACM Multimedia*, pages 1041–1042. ACM, 2005.
29. K. Walsh and E. G. Sirer. Fighting peer-to-peer spam and decoys with object reputation. In *Proceedings of the 2005 ACM SIGCOMM workshop on Economics of peer-to-peer systems*, P2PECON '05, pages 138–143, 2005.
30. S. Xiang-bin, L. Fang, D. Ling, C. X. hong, and X. Yuan-sheng. A cheating detection mechanism based on fuzzy reputation management of p2p mmogs. In *Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, 2007. SNPD 2007. Eighth ACIS International Conference on*, volume 2, pages 75 –80, 30 2007-aug. 1 2007.
31. S. Yamamoto, Y. Murata, K. Yasumoto, and M. Ito. A distributed event delivery method with load balancing for mmorpg. In *NETGAMES*, pages 1–8. ACM, 2005.
32. J. Yan and B. Randell. A systematic classification of cheating in online games. In *Proceedings of 4th ACM SIGCOMM workshop on Network and system support for games*, NetGames '05, pages 1–9, New York, NY, USA, 2005. ACM.
33. L. Yu, W. Susilo, and R. Safavi-Naini. X2bt trusted reputation system: A robust mechanism for p2p networks. *The 5th International Conference on Cryptology and Network Security (CANS 2006), Lecture Notes in Computer Science 4301*, pages 364 – 380, 2006.