# Secure Referee Selection for Fair and Responsive Peer-to-Peer Gaming

Steven Daniel Webb and Sieteng Soh
*Department of Computing*
*Curtin University of Technology*
*{steven.webb@postgrad.,S.Soh@}curtin.edu.au*

Jerry L. Trahan*
*Dept. of Electrical & Computer Engineering*
*Louisiana State University*
*trahan@ece.lsu.edu*

## Abstract

*Peer-to-peer (P2P) architectures for Massively Multiplayer Online Games (MMOG) provide better scalability than Client/Server (C/S); however, they increase the possibility of cheating. Recently proposed P2P protocols use trusted referees that simulate/validate the game to provide security equivalent to C/S. When selecting referees from un-trusted peers, selecting non-colluding referees becomes critical. Further, referees should be selected such that the range and length of delays to players is minimised (maximising game fairness and responsiveness). In this paper we formally define the referee selection problem and propose two secure referee selection algorithms, SRS-1 and SRS-2, to solve it. Both algorithms ensure the probability of corrupt referees controlling a zone/region is below a pre-defined limit, while attempting to maximise responsiveness and fairness. The trade-off between responsiveness and fairness is adjustable for both algorithms. Simulations show the effectiveness of our algorithms in two different scenarios.*

## 1. Introduction

Network games are computer games played amongst multiple players on different hosts across a network, often the Internet. Massively Multiplayer Online Games (MMOG) differ from traditional network games as they present a single universe in which thousands or tens of thousands of players participate simultaneously. Furthermore, these worlds are persistent; hence, the game world evolves even when the player is offline. Therefore, in addition to addressing game consistency, responsiveness, fairness, and cheat-free requirements, one must also address game persistency, system scalability, and system reliability when developing an MMOG [1], [2], [3].

The vast majority of networked games use a Client/Server (C/S) architecture, in which the server is the game authority. With only one centralised trusted server, keeping the game consistent, persistent, and cheat free in C/S is straightforward [4]. Unfortunately, C/S suffers from the following limitations: bandwidth scalability - the server's incoming and outgoing bandwidth is a bottleneck as the publisher must provision sufficient bandwidth at one location, which is an expensive re-occurring cost [5]; processing scalability - the server's processing power is a bottleneck, as it must simulate the entire virtual world and perform Area of Interest (AoI) filtering for all players [1], [6]; responsiveness - redirecting updates through the server increases game delay; reliability - the server is a single point of failure for the system; and fairness - players geographically close to the server have an unfair advantage, as they will have better responsiveness than those situated further away [2].

Several peer-to-peer (P2P) architectures [4], [7], [8] have been proposed to address the C/S limitations. P2P is scalable as the bandwidth and processing requirements are entirely handled by the clients; hence, there is no central bottleneck. Furthermore, P2P systems are resource growing; as the number of clients increases so does the overall bandwidth and processing power of the system. Unfortunately, keeping the game consistent and cheat-free in P2P is significantly harder and more costly than in C/S, as the latter utilises trusted servers to store the world state and to validate and authenticate all player updates [8].

Cheating is a major concern in network games as it degrades the experience of the majority of players who are honest [5]. This is catastrophic for games using subscription models to generate revenue [4]. Several P2P protocols [7] prevent protocol-level cheats. However, as these protocols do not use a trusted third party to store secret information and validate player actions, these protocols are vulnerable to information exposure and invalid command cheats which are prevalent in MMOG, while introducing new forms of cheating not possible in C/S [4]. In addition, these

* This work was supported in part by the National Science Foundation under grant number CCR-0310916.

1087-4097/08 $25.00 © 2008 Crown Copyright
DOI 10.1109/PADS.2008.16

solutions require costly distributed validation algorithms that increase game delay and bandwidth. See [9] for a review of possible cheats and their solutions for P2P architectures.

The Referee Anti-Cheat Scheme (RACS) [4] is a hybrid C/S and P2P architecture that allows players to exchange updates directly, minimising delay. RACS uses a trusted referee combined with cryptographic techniques to provide cheat prevention equivalent to that in C/S. Since the referee sends updates only in the event of inconsistencies or when peers cannot communicate directly, its outgoing bandwidth is minimised. However, the referee in RACS receives, simulates, and validates all updates, and therefore its incoming bandwidth or processing power may create a bottleneck. Furthermore, all of the bandwidth must be provisioned at one location. Finally, the referee in RACS is a single point of failure.

To provide security equivalent to C/S, a P2P architecture requires selecting peers to act as referees [4] or Region Controllers (RCs) [8], which are trusted to simulate the game fairly. Note, we assume the game world is divided into discrete sections called zones/regions. Referees/RCs are responsible for ensuring fair game play within the zone to which they are assigned. To remove any incentive for cheating, a referee should not supervise the zone in which the player's avatar is located [8]. However, this does not prevent a group of colluding players from cheating (a colluding player selected as a referee biases the outcome for another player) or griefers from disrupting the game (griefers intentionally damage other's experience for entertainment without gaining an advantage). Mutual checking - "you may not trust a single client, but you trust the consensus of multiple unaffiliated clients" [8] - can prevent these attacks. By using the consensus of multiple referees for the game state, it becomes far more difficult (but not impossible) for a group of colluding cheaters or griefers to influence the game unfairly.

Selecting multiple referees for each zone is the focus of this paper. Multiple, conflicting goals are relevant. In RACS [4], the referee's game state is authoritative; therefore, it is beneficial for peers to have low delay to the referee, since in many multiplayer computer games a player's delay has a significant impact on their performance [10]. Selecting referees located close (in terms of delay) to the players in a zone would be beneficial. Selecting referees close to players, however, increases the likelihood of a referee-player collusion, hence weakening security. Furthermore, in games where delay has an impact on the outcome, fairness is also an issue [3]. To be fair, all players should receive all updates from the referee simultaneously (to prevent one peer responding to an event before others have received the update), and the referee should process all received updates simultaneously (if two players perform conflicting actions simultaneously, they should both have an equal chance of getting the action accepted as valid). The selected referees should both minimise the delay and maximise fairness.

In this paper we define the Referee Selection Problem (RSP) and two secure referee selection algorithms, SRS-1 and SRS-2, to solve it. Both algorithms ensure the probability of corrupt referees controlling a zone is below a pre-defined limit, while attempting to maximise responsiveness and fairness. The trade-off between responsiveness and fairness is adjustable for both algorithms.

The remainder of the paper is organised as follows. In Section 2 we discuss related work. Section 3 describes the system model and formally defines the Referee Selection Problem (RSP). Section 4 presents our solutions to the RSP: SRS-1 and SRS-2. Section 5 uses simulation to evaluate both algorithms, and Section 6 concludes our paper. Note, "he" should be read as "he or she" throughout this paper.

## 2. Related Work

### 2.1. Referee Anti-Cheat Scheme (RACS)

In RACS each pair of interacting peers uses one of two communication modes: Peer-Referee-Peer (PRP) or Peer-Peer (PP) [4]. In PRP mode, updates are routed through the referee, similar to C/S, to prevent cheaters dropping updates to their opponents. PRP mode, however, increases delay and the referee's bandwidth. In PP mode, updates are sent directly between peers with a copy to the referee, which verifies the simulation and resolves conflicts between peers. As updates are not routed through the referee, responsiveness is maximised and the referee's outgoing bandwidth is minimised; therefore, PP mode is preferable for both the referee and peers. PRP mode should only be used in the event of poor connectivity or when cheating is suspected [4].

To prevent cheating, RACS divides time into rounds. Every peer generates one update per round. The game publisher defines QoS requirements between peers, and if a pair of peers cannot meet those requirements, then they will revert to PRP mode (poor connectivity and cheating are indistinguishable). Further, every update is digitally signed and includes round numbers to prevent traditional security attacks such as spoofing and replay attacks. For an in-depth discussion of the security features of RACS, see reference [4].

While RACS reduces the outgoing bandwidth of C/S, the referee must still receive all updates and simulate the entire virtual world, possibly causing a bottleneck. Furthermore, the referee is a single point of failure. Webb *et al.* [1] proposed using multiple mirrored referees to reduce the bandwidth bottleneck and remove the single point of failure; the referee, however, is still responsible for simulating the entire virtual world.

In this paper, we propose selecting peers to act as referees to increase the bandwidth and processing scalability of RACS. We assume the bandwidth and processing power of peers is below that of the referee in RACS; therefore, the virtual world is divided into discrete regions called zones, and each referee is only responsible for the zone to which it is assigned. Using multiple referees running on peer machines increases the scalability of RACS, but the assumption that the referee is trusted is no longer true. To overcome this, we propose using multiple referees per zone, and the authoritative game state is agreed among referees. As the sole referee in RACS is fixed, no optimisation problem arises. In contrast, in this work we attempt to optimise the selection of referees to maximise responsiveness and fairness while maintaining security.

## 2.2. Peer Vote Tallying

Referees may vote among themselves to form a consensus about the game state and then notify the peers (each peer is notified by one referee), but this introduces considerable delay that is not acceptable for many genres of games. Further, it requires a mechanism to prevent a malicious referee returning invalid results to peers. Kabus and Buchmann [8] proposed a solution for their scheme that uses multiple region controllers (RC) per zone. An RC is a peer elected to validate the game state for a zone, similar to our referee concept. An RC processes all updates for each round and sends the resulting game state (a vote) to all peers. Each peer receives and tallies the votes, and the majority vote is used as the current game state. This solution minimises delay at the cost of extra bandwidth for both peers and RCs.

This system also prevents a single RC from attacking peers by sending incorrect results, as they will be discarded in favour of the majority. If a group of colluding griefers were selected to be RCs for a zone, then they could potentially disrupt the game. Therefore, a secure random RC selection mechanism is required to minimise this risk. Their work does not address RC selection, however [8].

## 2.3. Secure Group Agreement

Corman *et al.* [11] proposed the Secure Group Agreement (SGA) protocol, a distributed protocol to securely select a set of peers randomly without requiring a trusted central authority. This set forms a verification group, and the majority is trusted to simulate the game fairly and correctly. This selection method is combined with the Secure Event Agreement (SEA) protocol [7] to prevent protocol-level cheats. The distributed group selection problem is solved by a complex cryptographic protocol that relies on a distributed hash table [11]. While this scheme is secure, the peer selection algorithm ignores the underlying network topology. Therefore, the verification group will frequently include peers located in distant parts of the network, dramatically reducing game QoS (*i.e.,* responsiveness and fairness). Our Referee Selection Problem (discussed in Section 3) addresses the QoS, in addition to the security issue.

## 2.4. Fairness

In fast paced games such as first person shooters (FPS), player delay has a significant impact on the outcome. If one player receives updates earlier than his opponents, he can react faster, giving him an advantage. For example, consider a game with two players A and B having 50ms and 200ms delays from the server, respectively. When the server sends out an update, player A will receive it and respond before player B has even received it.

Aggarwal *et al.* [3] measured game state error as the difference between the game state on different players caused by the server-to-client delay. Their measure of fairness is the standard deviation of the game state error. Decreasing the standard deviation improves fairness and is achieved by either delaying sending updates such that all players receive updates simultaneously or adjusting the frequency of updates sent to each player. Both of these schemes, however, ignore the player-to-referee delay. If two players perform conflicting events simultaneously, then they should each have equal chance of being successful. Furthermore, if updates are delayed/reduced to that of the slowest player, then the game may become unplayable. Finally, a single griefing player may artificially inflate their delay to damage the experience of others.

In this paper we use the range of peer delays as the measure of fairness. To be completely fair, updates that are both sent and received by the referees must be delayed to that of the slowest player. Further, our algorithms give the developer control to adjust the balance between fairness and responsiveness, preventing a single griefer from damaging the game state.

# 3. Optimal Referee Selection Problem

## 3.1. System Model

We assume the game world is divided into discrete regions called *zones*, either dynamically or statically, and that a mechanism exists to transfer players between zones. Further, a player perceives only a small portion of the virtual world, his *area of interest* (AoI), and AoI filtering is used to reduce the update size for each peer; zones are considerably larger than a player's AoI.

Let P = {$P_i$ | *i* is the unique identifier (ID) of each player} be the set of peers/players in the game, R = {$R_f$ | *f* is the unique identifier (ID) of each referee}$\subseteq$ P be the set of referees in the game, $Z_P \subseteq P$ be the set of players located within zone *Z*, and $Z_R \subseteq R$ be the set of referees controlling zone Z. Multiple referees are used to prevent a single griefer (becoming a referee) disrupting the game. As in [11], we distinguish between corrupt nodes (nodes that wish to disrupt the game) from colluding nodes (corrupt nodes that work together to disrupt the game). Further, we assume the number of corrupt nodes far exceeds the number of colluding nodes. Let C = {$C_i$ | $C_i$ is a set of colluding players}. We assume there is no method for corrupt nodes to identify each other (unless they are already colluding) as this same method could be used by the publisher to detect them [11]. Thus, all sets of colluders are disjoint, *i.e.,* $C_i \cap C_j = \varnothing$ for $i \neq j$. The size of the largest group of colluding peers, *max* ($|C_i|$), is unknown, however, we assume that the developer can estimate *maxC* ≈ *max* ($|C_i|$). Fig. 1 illustrates the relationships among players, referees, and cheaters.

We assume the publisher runs a trusted Authentication Server (AS), responsible for authenticating and validating joining players (subscripting, banning, *etc.*), and selecting peers to act as referees; therefore, the referee selection process is trusted. Further, the publisher may provision a small number of dedicated trusted referees that can bootstrap the system when the number of players is low. Note, acting as a referee requires additional bandwidth and processing requirements; therefore, only peers with sufficient resources should be considered for referees. The bandwidth and processing resources for each peer should be transmitted to the AS as part of the authentication process to achieve this.

Each peer can play and referee at the same time; therefore, $Z_R \subseteq P$. A referee should not control the zone in which his avatar is located. In other words, each $R_f$ in $Z_R$ is not in $Z_P$, *i.e.,* $Z_R \cap Z_P = \varnothing$. If a referee's avatar moves into a zone it is controlling, then the AS will select a new referee as a replacement. The following additional checks may be performed to improve security: (i) a referee must not share the IP address of a player whose avatar is located within the zone (multiple players sharing one Internet connection), and/or (ii) a referee must not control a zone containing players where game mechanics indicate bias (*e.g.,* if they are both members of the same team/guild/clan).

The number of referees per zone, $|Z_R|$, should be set by the developer. Each player in $Z_P$ receives the game state from all $R_f$ in $Z_R$ (three lines joining each $P_i$ to $Z_R$ in Fig. 1, $|Z_R|$=3) and takes the majority result; therefore, it requires at least $\lceil |Z_R|/2 \rceil$ colluding referees controlling one zone to tamper with the game state.

Finally, we model game fairness as the range of the average delay for each peer in $Z_P$ to all referees in $Z_R$. If all peers receive updates simultaneously from the referees, then the game is completely fair (delay range 0), whereas, the higher the range of delays, the greater the unfairness. Note that a game with very high delay may be fair, but unplayable. To be fun a game should be both fair and playable.

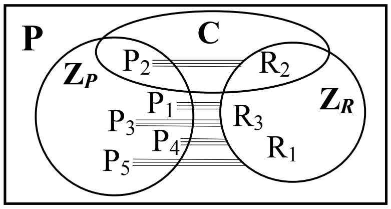

**Figure 1. Peer membership**

## 3.2. Problem Statement

Let $d_{i,f}$ be the delay from a player $P_i$ to/from a referee $R_f$; we assume symmetric delay, *i.e.,* $d_{i,f} = d_{f,i}$. Given $|Z_R|$ and $Z_P$ for zone Z and the publisher's predefined $0 \leq S_{max} \leq 1$ and *max*C, the *Referee Selection Problem* (RSP) is to select a referee set $Z_R$ such that:

(1) the probability $S_Z$ that there are $\lceil |Z_R|/2 \rceil$ or more colluding referees is not larger than $S_{max}$;

(2) the average peer-to-referee delay, $\overline{d_{i,Z_R}} = (\sum_{P_i \in Z_P, R_f \in Z_R} d_{i,f}) / (|Z_P| \times |Z_R|)$, is minimized; and

(3) the difference, Δ, between the maximum and the minimum of the player-referee delays (averaged across all referees in $Z_R$) for all peers in $Z_P$ is minimized, *i.e.*, minimize $\Delta = max(\overline{D_{i,Z_R}}) - min(\overline{D_{j,Z_R}})$, where

$\overline{D_{i,Z_R}} = \sum_{R_f \in Z_R} d_{i,f}/|Z_R|$ is the average delay from a player $P_i$ in $Z_P$ to all referees in $Z_R$.

The game developer should set the probability $S_{max}$ based on the security requirements of the game; lower $S_{max}$ improves security, as it reduces the chance that the majority of referees in $Z_R$ are colluding. Further, since colluders in one group cannot locate members of another group (see Section 3.1), the RSP considers security only against groups of up to *max*C colluders, not the union of all groups of colluders. Note that if *max* ($|C_i|$) > *maxC*, then we may not be able to select $Z_R$ that meets $S_Z \le S_{max}$. Corman *et al.* [11] proposed a distributed solution to solve criterion (1) but, as discussed in Section 2.3, the solution does not address criteria (2) and (3).

Criteria (2) and (3) are used to improve the game's QoS; criterion (2) deals with improving game responsiveness, while criterion (3) addresses game fairness. Since a valid game state is decided by the majority of referees (not the consensus of all referees), using the average value (not the maximum delay) in criterion (3) is sufficient to achieve fairness. To illustrate the criteria, consider Fig. 2 that includes $Z_P=\{P_1,P_2,P_3,P_4\}$ and assume we want to select one referee from two candidate referees, $R_A$ and $R_B$. Consider the following player-to-referee delays (in *ms*): $d_{1,A}=10$, $d_{1,B}=40$, $d_{2,A}=20$, $d_{2,B}=60$, $d_{3,A}=10$, $d_{3,B}=40$, $d_{4,A}=100$, $d_{4,B}=60$. If $R_A$ is selected, $\overline{d_{i,f}} = 35$ and $\Delta = 90$. However, selecting $R_B$ will give $\overline{d_{i,f}} = 50$ and $\Delta = 20$, which is better in terms of criterion (3) but worse for criterion (2). Note that selecting the peers closest to the players in $Z_P$ as referees will obviously optimise criterion (2), but could compromise criterion (1) as it is probable that colluding peers will be located within the same part of the network. In general, as all three criteria are conflicting it is not possible to simultaneously optimise all of them.

One may use a solution to the matching problem to solve criterion (2) or (3), although existing algorithms [12] do not address the security issues typical to MMOG. Our proposed secure referee selection algorithms, SRS-1 and SRS-2, attempt to address criteria (2) and (3) to the extent possible while meeting criterion (1).

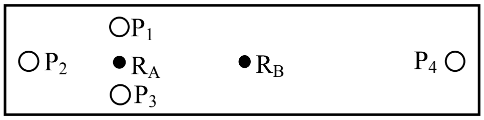

**Figure 2. Referee selection example**

# 4. Secure Referee Selection Algorithms

The Authentication Server (AS) holds responsibility for selecting referees. It will run one of the referee selection algorithms for each zone of the game world.

## 4.1. Estimating Delay between Peers

To address criteria (2) and (3), a solution to the RSP requires knowing all peer-to-peer delays. Each $d_{i,j}$ can be measured using echo packets between peers *i* and *j*. One could keep the delays in a $|P|\times|P|$ delay matrix since any peer may potentially act as a referee. Creating this matrix requires $O(|P|^2)$ measurements and space, which becomes infeasible for large |P|; the time cost is even worse if we consider maintaining the matrix given the dynamic nature of players and the Internet.

In this paper, we propose the use of network coordinates [13] to estimate peer-to-peer delay. Network coordinates provide a good estimation of the delay between any two peers *i* and *j*, even if no direct measurements between *i* and *j* have been made. In contrast to using a delay matrix, in this approach we estimate delay only when it is needed. Therefore, it is much more bandwidth- and space-efficient. Note that one can use other methods (*e.g.,* landmarks [13] or geographic location [14]) to estimate peer delays. We assume that each peer calculates its own network coordinates and transmits them to the AS.

## 4.2. Size of the Candidate Referee Set

Let $Z_{RP} = P - Z_P$ be the referee pool from which each $R_f$ in $Z_R$ is selected. As shown in Fig. 3, $Z_{RP}$ may include players from more than one $C_i$.

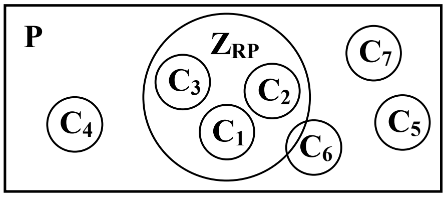

**Figure 3. Colluding peer membership**

The probability $S_Z$ of selecting at least $\alpha = \lceil |Z_R|/2 \rceil$ colluding referees is given by

$$S_Z = \frac{\sum_{i=\alpha}^{|Z_R|} \binom{maxC}{i} \binom{\Psi - maxC}{|Z_R| - i}}{\binom{\Psi}{|Z_R|}}, \qquad (1)$$

where $\Psi$ is the size of the candidate referee set, $Z_{CR} \subseteq Z_{RP}$. We want to find the minimum value of $\Psi$ such that randomly selecting $|Z_R|$ peers from $Z_{CR}$ will limit $S_Z$ to no more than $S_{max}$. A simple brute force approach suffices to find this minimum value, since the values of all other variables in Eq (1) are known.

Randomly selecting $Z_{CR}$ from $Z_{RP}$, as in [11], will likely result in obtaining $Z_R$ with large peer-to-referee delays and a large range of delays, which fails to satisfy criteria (2) and (3). In the following subsections, we propose two algorithms, SRS-1 and SRS-2, that meet criterion (1) and balance criteria (2) and (3).

### 4.3. SRS-1

Algorithm SRS-1 emphasizes responsiveness (criterion (2)) while satisfying security (criterion (1)). Given the pre-computed minimum size $\Psi = |Z_{CR}|$ from Eq. (1), SRS-1 performs three steps: (i) select the candidate referee set $Z_{CR} \subseteq Z_{RP}$, such that selecting any subset $Z_R \subseteq Z_{CR}$ will give a small value of $\overline{D_{i,Z_R}}$; (ii) select $|Z_R|$ referees randomly from $Z_{CR}$; and (iii) artificially inflate peer delays to $d_{F\%}$ (defined later) such that F% of peers have equal delay.

For Step (i), SRS-1 selects referees close to the majority of players. Let $(x_i,y_i)$ be the coordinates of $P_i$. Informally, we define the *major* as the point in the coordinate space that is closest to the *majority* of players in $Z_P$. As an illustration, consider Fig. 4 that shows an example $Z_P$ in 2D coordinate space. The total distance, TD(x,y), from a coordinate (x, y) to all players is:

$$TD(x,y) = \sum_{P_i \in Z_P} dist((x,y),(x_i,y_i)), \qquad (2)$$

where *dist*(A,B) is the Euclidean distance between points A and B. Formally, the major is the point that minimises Eq. (2). SRS-1 will form $Z_{CR}$ in Step (i) by selecting the $\Psi$ peers not in $Z_P$ that are closest to the major coordinate. Thus, selecting referees close to the major addresses criterion (2). The random referee selection in Step (ii) addresses criterion (1).

Since this selection does not consider the range of delays, a large range may result, so Step (iii) addresses criterion (3). The mechanism to decrease the range of delays is that a referee can send updates late to a peer with small delay so that it receives updates at the same time as a peer with large delay, artificially giving these peers the same delay. We define the *fairness weight* $0 \leq F \leq 100\%$ in Step (iii) as the minimum percentage of peers who must have equal average delay to the referees. Specifically, for each peer we calculate the average delay to all referees, find $d_{F\%}$ (the maximum delay among the fastest F% of players), and inflate the delay of the fastest F% of peers to $d_{F\%}$. The developer may set the weight of F between 0 and 100% to balance responsiveness and fairness (criteria (2) and (3)). If SRS-1 inflated delay for all peers (F = 100%), as proposed in [3], it would inflate all peer delays to that of the slowest peer, possibly undermining the purpose of Step (i). Using SRS-1 with F=0% is analogous to current commercial games [10] which attempt to provide the fastest service possible to each individual player, ignoring fairness. As an alternative to setting a fixed weight for F in Step (iii), one may use outlier detection (*e.g.,* the box-plot method [15]) to ignore peers with very high delay when calculating the inflation value.

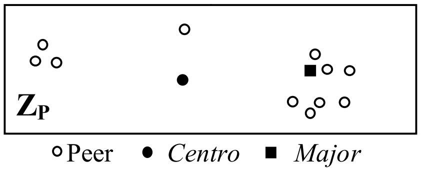

**Figure 4. Example 2D network coordinates**

```
Algorithm: SRS-1
Imports: $\Psi$: $|Z_{CR}|$
         r: The required size of $Z_R$
         F: The fairness weight
(x,y) = major($Z_P$)      //centro($Z_P$) for SRS-2
$Z_{CR}$ = find_CR((x, y), $\Psi$, P - $Z_P$)
while $|Z_R|$ < r do
         $R_i$ = random($Z_{CR}$)
         $Z_R = Z_R \cup R_i$
         $Z_{CR} = Z_{CR} - R_i$
end
$d_{F\%}$ = slowest_peer($Z_P$, $Z_R$, F)
inflate_peers($Z_R$, $Z_P$, $d_{F\%}$)
```

**Figure 5. SRS-1 Algorithm**

The referee selection algorithm SRS-1 is shown in Fig. 5. The major($Z_P$) function returns the *major* for the set of peers $Z_P$, calculated as the median x and median y value of all players in $Z_P$. The find_CR((x, y), $\Psi$, P - $Z_P$) function returns $\Psi$ peers $P_i \notin Z_P$ closest to the major coordinates (x,y). Function slowest_peer($Z_P$, $Z_R$, F) returns the average peer-to-referee delay of the F*$|Z_P|$th slowest peer. Finally, function inflate_peers($Z_R$, $Z_P$, $d_{F\%}$) notifies all referees in $Z_R$ to artificially inflate the delay to peers in $Z_P$ to $d_{F\%}$. Note that if $Z_R$ is already partially populated the algorithm will select only the number of referees required to fill the region. For example, the algorithm can select a replacement referee when one leaves the game.

### 4.4. SRS-2

Algorithm SRS-2 emphasizes fairness (criterion (3)) while satisfying security (criterion (1)). As for SRS-1, SRS-2 comprises three main steps: (i) select the candidate referee set $Z_{CR} \subseteq Z_{RP}$ such that selecting any subset $Z_R \subseteq Z_{CR}$, will incur a small inflation value $d_{F\%}$; (ii) select $|Z_R|$ referees randomly from $Z_{CR}$; and (iii) artificially inflate peer delays to $d_{F\%}$ such that the closest F% of peers have equal delay. Note that Steps (ii) and (iii) of SRS-1 and SRS-2 are identical.

We can directly address criterion (3) by selecting referees that minimise the range of peer delays. However, this approach would unlikely be able to optimise criterion (2), resulting in a larger $d_{F\%}$. Therefore, SRS-2 selects referees close to the center of $Z_P$, selecting referees such that the average delay from all referees to the furthest player is minimised. Let the *centro* be the point in the coordinate space such that the maximum distance to all players in $Z_P$ is minimised. Let the maximum distance, MD(x,y), from a coordinate (x, y) to all players be:

$$MD(x,y) = max(\forall P_i \in Z_P, dist((x,y),(x_i,y_i)), \quad (3)$$

The *centro* (illustrated in Fig. 4) is the point that minimises Eq. (3). However, if there are some outlying peers located in distant parts of the network they will have a significant impact on the *centro*. To prevent this SRS-2 may use outlier detection to ignore distant peers when calculating the *centro*. For each peer the delay to all other peers is calculated, and boxplot outlier detection [15] is used to identify distant peers. Similar to SRS-1, SRS-2 populates $Z_{CR}$ with the closest $\Psi$ peers, $P_i \notin Z_P$, to the *centro*.

As some players may be located very close to the *centro*, there may still be a significant delay range. As in SRS-1, to achieve criterion (3), in Step (iii) the referees artificially inflate peer delays to $d_{F\%}$. Note that SRS-2 with outlier detection is not effective when F=100% as it results in generating a large inflated delay, and hence reduces responsiveness. Therefore, we suggest using SRS-2 with outlier detection only for F<100%.

SRS-2 is shown in Fig. 5 by replacing function major($Z_P$) with centro($Z_P$). In this paper, we use gradient descent [16] in function centro to find the coordinates.

## 5. Simulation and Discussion

We use simulation to compare the effectiveness of SRS-1 and SRS-2 in addressing criteria (2) and (3) against random referee selection, which is equivalent to SGA [11]. The simulation requires knowing the peer-to-peer delays, $d_{i,j}$, and their avatar locations. As no trace data from a real MMOG is available, we synthesized two representative topologies, one from a popular MMOG [17] and Internet delays [14], and the other is a simple topology, and used them in Simulations 1 and 2, respectively.

### 5.1. Simulation 1

Table 1 shows the percentage of World of Warcraft (WoW) players located in each geographical region [17]; WoW is one of the most popular MMOG to-date, with over 9 million subscribers globally. We assume the **Other** players are located in Australia as it has significant delay to other regions [14], and a significant WoW player base [18].

As shown in the table, for Simulation 1, we generated a network game topology with 5000 peers distributed following the WoW player distribution. We used reference [14] to approximate the delay between these regions, and as most game players have broadband access [19], we added a last hop delay of 20ms [20] to all peers. We used the *Vivaldi* [13] simulator in matrix mode for $3 \times 10^6$ rounds to construct 2D network coordinates for all peers from the topology.

**Table 1. Player distribution for Simulation 1**

| Region | Peers | Region | Peers |
|---|---|---|---|
| **China: 44%** | 2200 | **Europe: 19%** | |
| **USA: 25%** | | UK: 46% | 437 |
| Boston: 25% | 313 | Germany: 35% | 333 |
| Dallas: 33% | 413 | France: 16% | 152 |
| LA: 28% | 350 | Spain: 3% | 28 |
| Seattle: 14% | 175 | **Other: 12%** | 599 |

Following WoW that allows groups of up to 40 players [18], our simulation populates $Z_P$ with 40 peers and selects $|Z_R|=3$ referees using random referee selection (SGA), SRS-1, and SRS-2. To show the impact of the distribution of players in $Z_P$ on each algorithm, we generated 41 different player distributions for $0 \leq W \leq 40$, where W is the minimum number of players located in the US. For each W value, we selected the other 40-W players randomly from around the world (including the US). We assumed $maxC = 10$ and $S_Z = 0.1$; therefore, $\Psi = 291$. The experiment is repeated 100 times for each of the 41 player distributions and the results are averaged, for inflated values $d_{100\%}$, $d_{80\%}$, and $d_{60\%}$. To evaluate the performance of our solutions when delay is not inflated, the figure includes the average peer-to-referee delay. Note that the current industry standard attempts to maximize responsiveness for every player individually, ignoring fairness, and therefore the average delay measure reflects the current standard.

As shown in Fig. 6, the average delay for SRS-1 outperforms random selection, even when only 25% of players are in the US (W=10). As interacting players for a MMOG are often located in the same region of the network [21] (*e.g.,* above 50% in the US), SRS-1 should be very effective in practice.

The figure also shows the performances of the algorithms when the publisher sets the weight of F. As shown in the figure, for F=100%, SRS-1 is better than random selection only when $W \in [40,37]$ (*i.e.,* at least 92.5% of the players are in the US). This result shows that it is not possible to achieve fairness and responsiveness to all players when even a small number of peers are located in distant parts of the network. Decreasing F trades fairness for responsiveness. The figure shows that when F=80%, the maximum delay remains the same even when 8/40=20% of the players are not in the US. Reducing F to 60% further reduces the effects of distant players (*i.e.,* tolerating almost 20/40=50% of peers outside the US) on the maximum response time, shifting the curve in the figure to the right. The results for SRS-2 for this topology are comparable to SRS-1, and therefore are excluded from the figure.

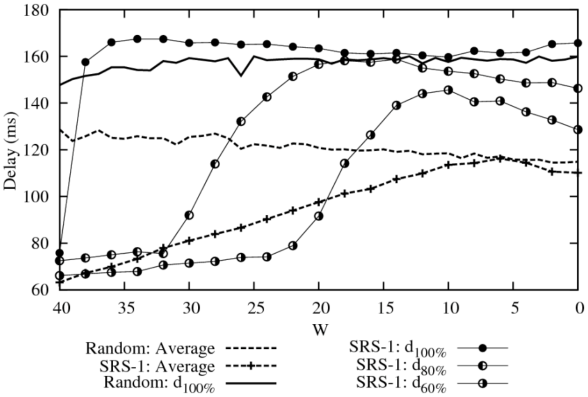

**Figure 6. Simulation 1 results.**

### 5.2. Simulation 2

The difference between SRS-1 and SRS-2 is not apparent from Simulation 1 due to the structure of the topology. For Simulation 2 we generated a topology with three locations, East Coast (East), Central, and West Coast (West) of the US, with delays between East/West to Central of 50ms, and East to West of 100ms. From Table 1, 25% of players are located in the US, and therefore we generated |P|=25% * 5000 =1250 players. We distributed them evenly among the three locations, and generated network coordinates similar to Simulation 1. We assumed $20 \le E \le 40$ players are located in the East, and the remaining 40 – E players were located in West; the referees are selected from any of the three regions.

As shown in Fig. 7, when 100% fairness is guaranteed, SRS-2 is significantly better than SRS-1. In contrast, when fairness is not guaranteed, the average delay for SRS-1 is significantly better than SRS-2. Note that the average and $d_{100\%}$ delays are identical for SRS-2. Provided P is distributed across many centres in the network, and not confined to a small number of locations as in Simulation 1, we believe the results will be comparable to Fig. 7.

Simulation 2 indicates that SRS-1 succeeds in its emphasis on responsiveness and SRS-2 succeeds in its emphasis on fairness. Consequently, a developer can choose either algorithm depending on which criterion is more important.

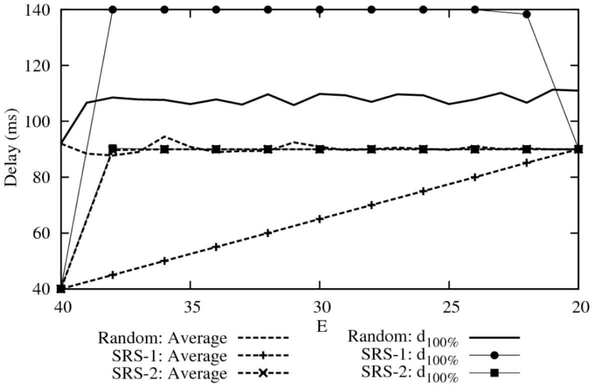

**Figure 7. Simulation 2 results.**

## 6. Conclusion

In this paper we have formally defined the Referee Selection Problem (RSP), the solution being critical for improving the performance of P2P network games that use referees to identify cheaters. The RSP raises three criteria in selecting an optimal referee set: security, responsiveness, and fairness. We have argued that the three requirements are conflicting, and therefore proposed two heuristic algorithms, SRS-1 and SRS-2 to solve the RSP.

SRS-1 solves the RSP by selecting referees such that the average peer-to-referee delay is minimised (emphasizing responsiveness), while SRS-2 selects referees such that the maximum distance to all players is minimised (emphasizing fairness). We have evaluated our algorithms using simulation, and discussed the merits of the solutions. We suggest game developers, first, use Eq. (1) to calculate the minimum size of the candidate referee selection pool to meet their required level of security. Then, use either SRS-1 or SRS-2 to select referees, depending on the game requirements for responsiveness and fairness.

For future work, since the game round length may be dynamic (*e.g.,* for RACS [4]), we wish to study the interaction of the round length adjustment algorithm, with that of the secure referee selection algorithm, in terms of responsiveness and fairness.

## 7. References

[1] S.D. Webb, S. Soh, and W. Lau, "Enhanced Mirrored Servers for Network Games," in Proc. Netgames, pp. 117-122, 2007.

[2] E.Cronin, A.R. Kurc, B. Filstrup, and S. Jamin, "An efficient synchronization mechanism for mirrored game architectures," Multimedia Tools and App., vol 23, 7–30, May 2004.

[3] S. Aggarwal, H. Banavar, S. Mukherjee, and S. Rangarajan, "Fairness in dead-reckoning based distributed multi-player games," in Proc. Netgames, pp. 1-10, 2005.

[4] S.D. Webb, S. Soh, and W. Lau, "RACS: a Referee Anti-Cheat Scheme for P2P gaming," in Proc. NOSSDAV, pp. 37–42, 2007.

[5] J. Mulligan, and B. Patrovsky, "Developing Online Games: An Insider's Guide," New Riders Publishing, 2003, Ch. 7.

[6] D. Ta, S. Zhou, and H. Shen, "Greedy algorithms for client assignment in large-scale distributed virtual environments," in Proc. PADS, pp. 103-110, 2006.

[7] A.B. Corman, S. Douglas, P. Schachte, and V. Teague, "A Secure Event Agreement (SEA) protocol for peer-to-peer games," in Proc. ARES, pp. 34–41, 2006.

[8] P. Kabus, and A.P. Buchmann, "Design of a cheat-resistant P2P online gaming system," in Proc. Dimea, pp. 113–120, 2007.

[9] S.D. Webb, and S. Soh, "Cheating in networked computer games - A review," in Proc. DIMEA, pp. 105–112, 2007.

[10] Valve, "Source Multiplayer Networking," http://developer.valvesoftware.com/wiki/Source_Multiplayer_Networking, Dec. 2006.

[11] A.B. Corman, P. Schachte, and V. Teague, "A Secure Group Agreement (SGA) protocol for peer-to-peer applications," in Proc. AINAW, pp. 24–29, 2007.

[12] S.D. Webb, and S. Soh, "Adaptive Client to Mirrored-Server Assignment for Massively Multiplayer Online Games," in Proc. MMCN, (to appear), 2008.

[13] F. Dabek, R. Cox, F. Kaashoek, and R. Morris, "Vivaldi: a decentralized network coordinate system," in Proc. SIGCOMM, pp. 15-26, 2004.

[14] W. Matthews, and L. Cottrell, "The PingER project: active Internet performance monitoring for the HENP community," IEEE Comm. Mag., vol 38, May 2000.

[15] J.W. Tukey, "Exploratory data analysis", Addison-Wesley, 1977.

[16] G. Arfken, "Mathematical Methods for Physicists," 3rd ed., Orlando, FL: Academic Press, 1985, pp 428-436.

[17] Blizzard, "World of Warcraft Surpasses 9 million subscribers worldwide," http://www.blizzard.com/press/070724.shtml, Jan 2008.

[18] WoWWiki, http://www.wowwiki.com/, Jan 2008.

[19] Valve, "Survey Summary Data," http://www.steampowered.com/status/survey.html, Jan 2008.

[20] K. Lakshminarayanan and V.N. Padmanabhan, "Some findings on the network performance of broadband hosts," in Proc SIGCOMM, pp. 45-50, 2003.

[21] K.T. Chen, and C.L. Lei, "Network game design: hints and implications of player interaction," in Proc. Netgames, no. 17, 2006.