PlayerRating: A Reputation System for Multiplayer Online Games

Edward Kaiser
Department of Computer Science
Portland State University
Portland, Oregon
edkaiser@cs.pdx.edu

Wu-chang Feng
Department of Computer Science
Portland State University
Portland, Oregon
wuchang@cs.pdx.edu

***Abstract*— In multiplayer online games, players interact with each other using aliases which unfortunately enable antisocial behavior. Vague rules and limited policing mean that only the very worst offenders are ever disciplined. This paper presents PlayerRating, a distributed reputation system specifically designed for online games. It leverages the prior experiences of a player's peers to determine the reputability of all other peers, allowing well-behaved players to safely congregate and avoid interaction with antisocial peers. The system has been implemented as an interface add-on for the game World of Warcraft and is evaluated theoretically and experimentally.**

***Index Terms*— Online Games, Player Ratings, Reputation Systems.**

I. Introduction

A. The Problem with Online Behavior

Like the users of many other network applications, players of multiplayer online games interact with each other using aliases, graphically represented by customizable avatars. An alias hides the player's real-world identity so they may be immersed in the virtual world. Players invest considerable time, effort, and even money improving the noteworthiness of their in-game persona – taking pride in any renown that they achieve (colloquially referred to as "*e-fame*").

Unfortunately, the anonymity provided by aliases leads to a number of antisocial behaviors [1]. For example, players cheat to automate the acquisition of wealth and items, or to misrepresent their abilities and falsely receive recognition from their peers. Players scam their peers (often exploiting a poorly designed facet of the user-interface) to steal wealth or items. Sometimes powerful players "*grief*" weaker players by playing in a manner that makes the game unenjoyable for them (e.g., "*corpse-camping*" a victim who must resurrect at the same location). Disgruntled players harass their peers through in-game text or voice chat. Some players simply act selfishly with poor sportsmanship (e.g., "*rage quitting*" the game when it greatly inconveniences their peers).

Although these behaviors are against the spirit of the game and many of its rules, limited policing resources mean that only the most grievous infractions are ever investigated. Often the virtual community is so large that a malicious player may negatively affect many peers before sufficient complaints are raised to warrant developer attention. Even then, the game rules regarding social behavior are so vague that infringing players are given several warnings before incurring significant repercussions. Innocent players are thus posed with a dilemma: either they only play with peers they know and trust "*in real-life*" (i.e., from outside the game) or they must risk playing with peers they meet in-game who may behave poorly.

This material is based upon work supported by Intel Corporation under Grant 34557. Any findings, conclusions, and recommendations expressed in this material are those of the authors and do not necessarily reflect the views of Intel.

B. Our Solution

This paper presents *PlayerRating*, a distributed reputation system specifically designed for online games. It leverages the prior experiences of a player's peers to determine the reputability of other peers they meet in-game. A player-specific reputation is bound to in-game aliases (preserving real-world anonymity) and provides an indication of the likely behavior one could expect from that player. This allows well-behaved players to congregate and avoid antisocial players. The novel features of PlayerRating are as follows:

Experience Sharing. While some in-game mechanisms exist for recording social relationships (e.g., friend-list and ignore-list) that data is never propagated. PlayerRating transparently shares player relationship data using strictly authenticated in-game channels (avoiding Sybil attacks [2]). This allows the player to learn of their peers' behavior and enables them to congregate with friendly peers and avoid known malicious peers.

Personal Perspective. Using the shared relationships, PlayerRating locally learns the game's social network from each player's perspective: players determine where they fit into the network by rating peers that they like or dislike. The system propagates trust through positive ratings to predict the player's perceived reputation of peers that they have yet to meet. The system works well without knowing every rating, but improves as more ratings are observed.

Fine Granularity. Existing mechanisms are too coarse, limiting relationship data to: like, dislike, or ambivalence. In PlayerRating, player relationships are analog meaning a player may express their like or dislike of each peer to different degrees. The PlayerRating system could be used to sort and better match players for group play, or focus policing resources on the most disruptive players.

C. Related Work

Existing In-game Tools. Many games offer simple tools for tracking peers that a player likes (e.g., friend-list) or dislikes (e.g., ignore-list). Extensions like PlayerNotes [3] add annotation capabilities to those tools so that the player may be reminded why they befriended or ignored peers. While these tools are easy to implement and are present in many online games, they do not share such preferences and thus cannot predict whether or not a player will like a peer they meet in-game for the first time.

Reputation and Recommender Systems. A common approach to improving online social interaction is the use of reputation systems, such as eBay's Feedback system [4] or Slashdot's Karma system [5]. Related to reputation systems are recommender systems (where value is assessed for objects rather than people), the most notable of which is the PageRank system that powers the Google search engine [6].

The PageRank system determines the popularity or rank of a webpage as the sum of all supporting evidence (i.e., HTML hyperlinks pointing to it) weighted by the rank of the referring webpages and a decay factor. The algorithm iteratively updates each webpage's rank through random walks of the Web, eventually reaching Web-wide equilibrium. So long as a webpage has many references from popular pages, it will also have a high PageRank.

Deployed reputation and recommender systems evaluate the global perception of persons or objects, yielding rough predictions that do not account for personal preference. For personalization, some approaches (e.g., TrustRank [7] and Personal PageRank [8]) extend PageRank while others (e.g., EigenTrust [9] and email filtering [10]) implement systems that operate in distributed fashion. Personalized approaches allow people with similar interests (e.g., players who dislike profanity) to congregate and form cliques, shown in Figure 1. Such approaches record peers that one likes or dislikes and extrapolates personal and shared ratings to predict like and dislike for personally unmet peers.

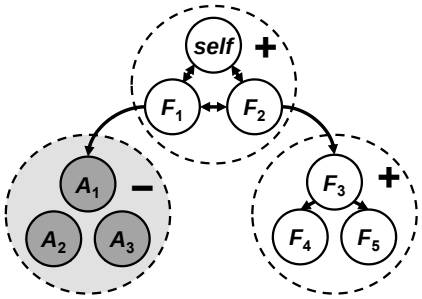

Fig. 1. A small social network where the observer ($self$) belongs to a clique with friends F_1 and F_2. Friend F_1 dislikes A_1 so without having met A_1 the observer will probably also dislike A_1 and her clique. Friend F_2 likes F_3 so without having met F_3 the observer will probably also like F_3. Furthermore, the observer will probably also like F_3's friends F_4 and F_5, although with less certainty.

Architecturally, our approach is similar to the PageRank system but is computed in a distributed fashion. PlayerRating calculations are tailored to better suit online games and direct exposure to players. PlayerRating leverages both positive and negative ratings, as well as bounding reputations so that they remain on a fixed scale and can be more easily interpreted by players.

II. Architecture

A. Definitions

This section defines the terminology used throughout this paper. The complete set of players is denoted by $Players$. A *rating* is the value corresponding to the opinion of one player (the *rater*, $i \in Players$) about another player (the *ratee*, $j \in Players$) and is denoted by $r_{i,j} \in [-1.0, 1.0]$, where a negative rating corresponds to dislike or distrust, and a positive rating corresponds to like or trust. A rating of 0.0 represents an unknown or neutral relationship.

The ratings for a given ratee j are combined to formulate their *reputation*, denoted by $R_j \in [-1.0, 1.0]$ with a similar interpretation of affinity. Intuitively, a rater's reputation dictates how influential their ratings are in determining the reputation of their peers: disliked raters will have little or no influence while liked raters will have more influence.

B. Assumptions

An assumption underlying all reputation systems is that past performance is a reliable indicator of future behavior, however, players may reform and improve their conduct. The game population forms a weighted directed graph where each vertex is an individual player (*uniquely identified* by a player ID) and edges represent ratings. The following assumptions are made:

- *Ratings are subjective.* Each player is allowed only one rating for each peer. A rating encompasses the rater's perception of every interaction with the ratee; how they judge interactions and base their ratings is up to them. To accommodate new interactions, a rating may be updated or withdrawn by the rater at any time.
- *Raters are authenticated.* Communication is authenticated with respect to the sender's (i.e., rater's) player ID by the server, which is true in practically all online games.
- *Players are self esteemed.* It is assumed that players always view themselves with highest regard (i.e., as absolutely likeable and trustworthy); ratings where $i = j$ are ignored and $R_{self} = 1.0$.
- *Ratings are asymmetrical.* It is likely that any two players will have different opinions of each other and their peers.
- *Positive ratings are transitive.* Insofar as positive ratings represent trust, positive ratings and reputations are transitive. Specifically, a peer trusted by a trusted peer becomes trusted, albeit with less certainty. The opposite may not be true as the ratings from distrusted peers are not necessarily useful.
- *Ratings follow a power-law distribution.* Players will self-organize following a power-law distribution [11], like other social systems [12], [13], [14], [15]. Much of the graph will be sparse as most players will create few ratings – even in systems with monetary incentive (e.g., eBay) only 60% of users ever generate ratings [16]. However, the power-law-based "*small-world*" model of Watts and Strogatz [17] indicates that a few well-connected players will result in a short average path length between any two players (i.e. only a few degrees of separation). This is fundamental for the system to yield good predictions.

C. System Goals

Players compete over recognition and limited in-game resources and will adopt tools that make the game more enjoyable or help them outperform their peers. Any reputation system should improve facilitate more positive player interactions yet prevent abuse:

- *Incremental Deployment.* As a new tool, the system must facilitate gradual adoption. Specifically, it must generate reputations as accurately as possible given only partial yet frequently updated graph information.
- *Encourage Participation.* Players cannot be expected to rate every player they interact with, yet reputation systems become more accurate with more ratings. The system must encourage participation by being easy to use and immediately beneficial.
- *Resistant to Abuse.* While encouraging participation is important, preventing abuse is equally important. Malicious players should not be able to increase their own reputation, even through collusion. Otherwise ill-gotten reputation could be used to slander legitimate players, or worse, lure trusting players into scams.
- *Minimal Overhead.* As the goal of the system is to improve the overall gameplay experience, it must not distract players at inopportune times or degrade game performance. This means the system should mostly operate in the background and be efficient in terms of computation, storage, and communication.

D. System Design

Figure 2 overviews the PlayerRating system which operates in a distributed fashion: each participating player runs a PlayerRating agent within their game client that determines the reputation of their peers, accounting for their personal perspective based on who they like and dislike. This section illustrates system features using a World of Warcraft [18] user-interface implementation [19], however, the system can easily be adapted to other games and game genres.

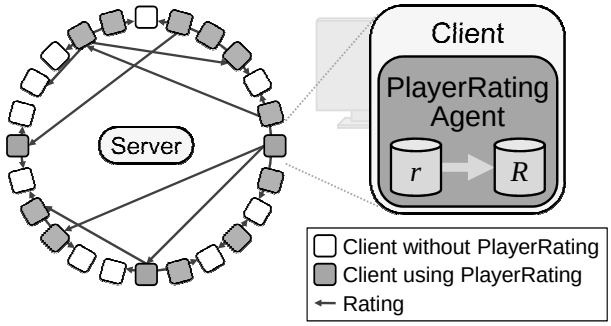

Fig. 2. Overview of the PlayerRating distributed reputation system. Agents calculate reputations from ratings (their own and those shared by peers).

Participation is optional in the sense that a player may choose not to rate their peers, however, they cannot prevent their peers from rating them. Furthermore, a player must accurately rate their peers before their agent can accurately calculate meaningful peer reputations.

Each player rates their peers when it is convenient to do so, presumably while socially interacting with the ratee or shortly thereafter. Ratings are shared with peers and expired periodically. Using all currently known ratings, the PlayerRating agent calculates each ratee's reputation as the average of every rating about them, weighted by the influence (a function of reputation) of the corresponding raters. Ratings are not absolute, but instead express a ratee's reputation relative to that of the rater. Thus a ratee cannot be more reputable than the most reputable rater who rated them, preventing collusion via positive feedback loops.

1) Initialization: The initialization of a PlayerRating agent simply involves zeroing all data (ratings, reputations, and related variables) and fixing their own reputation R_{self} to 1.0.

2) Recording Ratings: The PlayerRating agent unobtrusively extends the game's user interface (like the addition shown in Figure 3) to enable the player to easily rate their peers. As defined earlier, ratings run along a single dimension and represent the overall impression about the ratee. It is possible to extend ratings to other dimensions corresponding to criteria specific to player skills (e.g., $r_{i,j,healer}$) or behaviors (e.g., $r_{i,j,swearing}$) although this would increase system state and communication overheads. Some criteria may not be useful enough to justify such overheads and warrants further investigation.

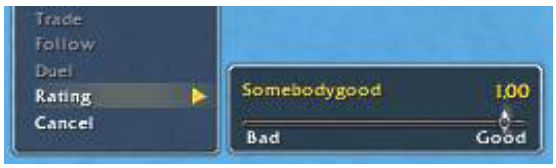

Fig. 3. The PlayerRating rating slide-bar added to the peer interaction menu in World of Warcraft. The menu is only displayed when the player chooses to interact socially with a peer.

The recording algorithm records both personally created ratings and ratings disseminated by one's peers. The freshness of a peer's rating is indicated by setting a corresponding time-to-live variable $ttl_{i,j}$ to the maximum value TTL_MAX so that it may expire if it later becomes irrelevant. Subsequent receipt of a peer's rating that had been previously recorded reaffirms the peer's commitment to that rating so the corresponding time-to-live value is reset to the maximum value. By doing so, relevant ratings survive the expiration process.

3) Sharing Ratings: Ratings are disseminated transparently via data channels which exist to support game modification ("*modding*") and operate similar to in-game chat channels. Only personally created ratings are broadcast on the PlayerRating channels. All peers listening to the channel may record ratings. A peer may record ratings from an unknown rater (i.e., $R_i = 0.0$) with the idea that in the future either they or someone they trust might determine that rater is also trustworthy.

Communication is strongly-authenticated since each message must go through the server which validates the sender's alias. To mitigate flooding, the server restricts message length to a couple hundred bytes and limits senders to a few messages every 10 seconds. PlayerRating agents may further limit ratings they accept to control the growth of their knowledgebase.

DISSEMINATION POLICY. A game-tailored policy must share ratings as quickly as possible yet minimize the required communication. The dissemination policy must also introduce some redundancy over time to ensure notification for peers who may have been offline during earlier broadcasts and to reaffirm the continued conviction in previously shared ratings otherwise they will be expired and discarded by one's peers. Strategies used by the World of Warcraft implementation include: broadcasting a rating as it is created or updated, broadcasting the rating after interacting with the ratee, and periodically broadcasting ratings sequentially. Other strategies may include broadcasting ratings randomly, or when joining a zone, small server, or group.

4) Expiring Stale Peer Ratings: The persistent nature of online games means that a lot can change while a player is offline. This affects the PlayerRating system in that one's peers may change or revoke previously shared ratings, or quit playing the game (obviating ratings created by or about them). To handle the possibility that the PlayerRating agent may obliviously possess stale peer ratings, the ratings collected from peers are slowly expired to ensure that the knowledgebase only contains relevant data.

EXPIRY POLICY. Periodically with period T_{dec}, ratings are aged by decrementing the associated time-to-live values. If a time-to-live value reaches zero, the agent has not received any reaffirmation of the peer's conviction behind the corresponding rating, so it is expired and removed from the knowledgebase. Players have accurate knowledge of their own ratings so those ratings do not need to be expired. The maximum lifetime of a peer's outdated rating is

$$\text{max\_lifetime} = T_{dec} \times \text{TTL\_MAX} \quad (1)$$

As a part of the game client, the PlayerRating agent only runs while the player is online so rating lifetime is measured in terms of *played time* (referred to as "*pTime*" with units "*pHours*", "*pDays*", etc.). While both T_{dec} and TTL_MAX are currently set as system settings, players could adjust them to change the maximum lifetime of ratings and thus control the amount of state kept by their PlayerRating agent. To be clear, a player that changes their own T_{dec} or TTL_MAX does not affect in any way how their peers store, use, or share ratings.

5) Calculating Reputations: The core algorithm of the PlayerRating system uses all known ratings to determine the reputations of one's peers, shown in Algorithm 1. The algorithm calculates the new reputations R' and updates R only once the entire graph has been processed, meaning only line 10 must be atomic. As the bulk of the algorithm is non-atomic, the algorithm may be periodically and incrementally executed with low priority to avoid sudden computation spikes that may result in degraded gameplay at inopportune times.

Algorithm 1 UpdateReputations()

```
1:  R', w ⇐ ∅
2:  for all i ∈ Players do
3:      w_Δ ⇐ Influence(R_i)
4:      for all j ∈ Players, r_{i,j} ≠ 0.0 and ¬IsSelf(j) do
5:          R'_Δ ⇐ (r_{i,j} × R_i × Decay(ttl_{i,j})) − R'_j
6:          w_j ⇐ w_j + w_Δ
7:          R'_j ⇐ R'_j + (R'_Δ × w_Δ / w_j)
8:      end for
9:  end for
10: R ⇐ R'
```

One iteration of the algorithm loops over all the players (lines 2-9) and their ratings (lines 4-8). Each rater's influence is calculated (line 3) by a monotonically non-decreasing function of their reputation: Influence(). In the World of Warcraft implementation, this function is the square of positive reputation and is zero otherwise:

$$\mathsf{Influence}(R_i) = \begin{cases} (R_i)^2 & \text{if } R_i > 0.0 \\ 0.0 & \text{otherwise} \end{cases} \quad (2)$$

Using this influence function, disliked and unknown peers have no influence (all their ratings may be skipped) while liked peers have quadratically more influence.

For each non-zero rating where the ratee j is not the actual player (i.e., is not $self$), *the relative rating is decayed, weighted and averaged* (lines 5-7) with all other ratings about the ratee to formulate the ratee's new reputation R'_j. Averaging relative ratings ($r_{i,j} \times R_i$) means that a ratee cannot be more reputable than their most reputable rater and that reputations will always fall on the same scale as ratings (i.e., [-1.0, 1.0]). The ratings are weighted by the rater's influence calculated earlier.

Intuitively, more recent interactions and ratings are more important than older ones. Since the rating's age is already maintained via its time-to-live value, the time relevance of the rating is logically calculated as a function of that value: Decay(). The World of Warcraft implementation uses a linear decay to avoid large changes when ratings are expired and removed from the knowledgebase:

$$\mathsf{Decay}(ttl_{i,j}) = \frac{ttl_{i,j}}{\text{TTL\_MAX}} \quad (3)$$

Figure 4 numerically illustrates the example system from Figure 1 after reputations calculated by $self$ have reached equilibrium. Before the first iteration, $R_{self} = 1.0$ and all other reputations are 0. After one iteration, the two peers closest to $self$ have $R_{F_1} = R_{F_2} = 0.5$ and all other peer reputations remain at 0. In the second iteration, the PlayerRating agent calculates the reputation for A_1 by $R_{A_1} = r_{F_1,A_1} \times R_{F_1} \times \frac{\mathsf{Influence}(F_1)}{\mathsf{Influence}(F_1)} = -0.5 \times 0.5 \times \frac{0.25}{0.25} = -0.25$ and the positive reputation for F_3 is calculated similarly through F_2. The reputations for F_1 and F_2 only remain unchanged because of the simple numeric formulation: $R_{F_1} = \frac{r_{self,F_1} \times R_{self} \times \mathsf{Influence}(self) \; + \; r_{F_2,F_1} \times R_{F_1} \times \mathsf{Influence}(F_1)}{\mathsf{Influence}(self) \; + \; \mathsf{Influence}(F_1)} = \frac{0.5 \times 1.0 \times 1.0 \; + \; 1.0 \times 0.5 \times 0.25}{1.0 \; + \; 0.25} = 0.5$. In the third iteration, F_4 and F_5 are discovered and none of the previously calculated reputations change. At this point the network is discovered as fully as possible given the two ratings $self$ created.

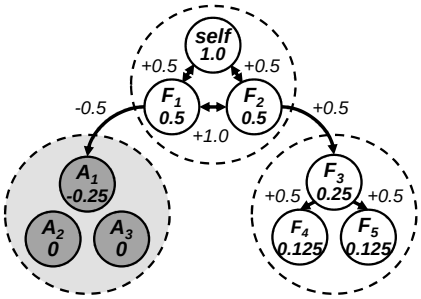

Fig. 4. The results of iterating UpdateReputations() over the small example social network shown in Figure 1. Labeled edges are ratings and labeled vertices are the reputations (from the point of view of $self$) achieved after reaching equilibrium in only three iterations.

6) Reputation Lookup: Looking up a peer reputation is as simple as returning R_j (or 0.0 if the peer is completely unknown). It is helpful to remind the player of their rating for that peer at the same time by returning $r_{self,j}$. Both lookups are inexpensive and the information can be presented in many graphical and numerical ways, such as on player tooltips shown in Figure 5.

Fig. 5. The PlayerRating reputation added to the World of Warcraft tooltip displayed when the player cursors over a peer's character.

III. Evaluation

Through its design, the PlayerRating system facilitates incremental deployment since one does not need to rate peers and choosing not to participate does not prevent peers from rating oneself. The system also encourages participation since one can only benefit from accurate peer predictions if they first accurately rate some peers. This section evaluates the system in terms of meeting the two measurable design requirements: resistance to abuse and low system overhead.

For evaluation, the PlayerRating system was implemented as an offline C++ application that was optimized for speed. This allowed experiments to be run without impacting real players and demonstrates how efficient the system might be if implemented directly within a game client. When it is instead implemented as a user-interface modification, it would be written in the game's scripting language. In the case of World of Warcraft that language is Lua which uses roughly the same memory footprint but executes at roughly $\frac{1}{30}$ of the speed of C++ for equivalent programs [20].

A. Resistance to Abuse

To demonstrate the PlayerRating system's resistance to abuse, experiments were performed on an emulated player population constructed using 30,000 identities from the Slashdot Zoo [14]. This is a reasonable number of player identities since World of Warcraft census indicate that realms support up to this many players [21]. Summarized in Table I, this subset preserves power-law characteristics of the original set.

CHARACTERISTIC	VALUE
Ratees	30,000
Raters	3,919
$1 \leq outlinks \leq 10$	2,402
$10 < outlinks \leq 100$	1,230
$100 < outlinks$	287
Ratings	101,842
positive	76,101
negative	25,741
Mean Ratings	3.4
positive	2.5
negative	0.9
Sparseness	0.000113

TABLE I
CHARACTERISTICS OF THE SLASHDOT ZOO SUBSET.

FUNCTION	THEORETICAL BOUNDS	EXPERIMENTAL CYCLES	EXPERIMENTAL TIME
Initialization()	$O(1)$	473	$0.2\mu s$
RecordRating()	$O(1)$	33,539	$14.0\mu s$
ShareRatings()	$O(outlinks)$	*policy dependent*	
ExpireRatings()	$O(\|r\|)$	30,267,923	$12.7ms$
UpdateRep′s()	$O(\|r\|)$	18,300,514	$7.7ms$
process rating	$O(1)$	300	$0.1\mu s$
process rater	$O(\|Players\|)$	26,089	$10.9\mu s$
commit R′	$O(\|Players\|)$	10,047,723	$4.2ms$
LookupRep()	$O(1)$	752	$0.3\mu s$

TABLE II
THE THEORETICAL BOUNDS AND EXPERIMENTAL COMPUTATION TO EXECUTE THE VARIOUS OPERATIONS OF A PLAYERRATING AGENT.

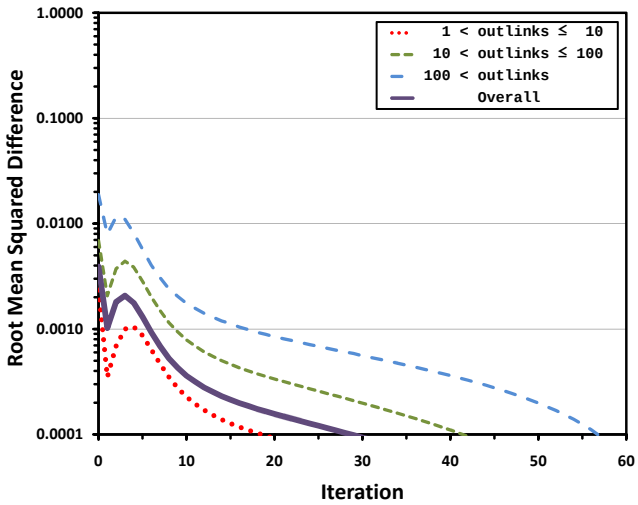

Fig. 6. The average $RMSD$ (i.e., reputation convergence) of PlayerRating agents as they iterate the UpdateReputation() function.

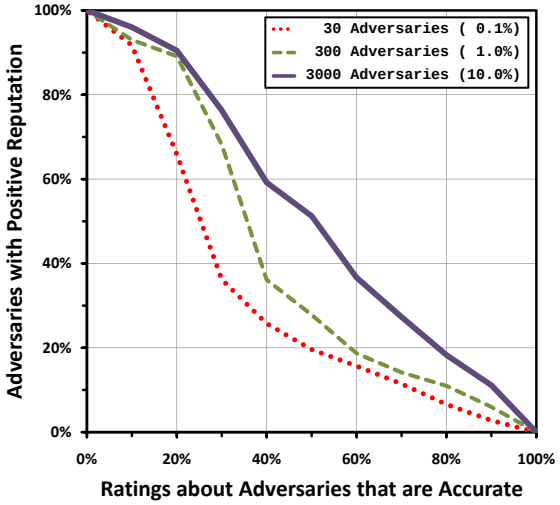

Fig. 7. The percentage of colluding adversaries with positive reputation (as viewed by good clients) vs. the percentage of accurate ratings about them.

1) Convergence to Equilibrium: This experiment shows how quickly PlayerRating agents converge to equilibrium in the worst case: when all ratings are discovered at the same time causing maximum disorder. Within one agent, reputation instability is measured as the root-mean-square-difference ($RMSD$) between iterations of the UpdateReputations() function calculated by

$$RMSD = \sqrt{\frac{\sum_{i \in Players} (R'_i - R_i)^2}{|Players|}} \quad (4)$$

where a value of zero means the system has completely reached equilibrium and any other value indicates that some reputations are still in flux.

Figure 6 shows the average $RMSD$ for various players with at least one rating. The results indicate that overall the average participant system converges quickly. Players who rate fewer peers converge faster since their connected graph is generally smaller while players who rate many more peers converge slower since their connected graph is larger. These results mean that malicious peers cannot generate a number of ratings that would cause instability. The small bump occurs at iteration 3 because that is the first iteration that can uncover a cycle in a completely new graph, possibly propagating trust back to a known peer.

2) Collusion Resistance: This experiment shows the ability of PlayerRating agents to accurately identify populations of colluding adversaries (i.e., assign them negative reputation). Initially, colluding adversaries join the game, establish a fully connected positive rating graph amongst themselves, and then briefly play legitimately to obtain positive ratings from the original player population (following the rating sparsity of the original set). Figure 7 shows that at this early stage, all the adversaries have positive reputation because they have not betrayed anyone's trust yet.

As adversaries commit acts that betray their previously earned positive reputation, they acquire negative ratings (which reverse some previously obtained positive ratings). As the percentage of positive ratings decreases, the percentage of negative (i.e., accurate) ratings increases and fewer adversary reputations remain positive. Eventually, the very last adversaries abuse their earned trust and are properly re-identified with negative ratings, leaving no adversaries with positive reputation.

The figure shows that larger colluding populations retain positive reputation longer than smaller populations. This occurs because the colluding population is fully connected meaning that the links attributed to them grows quadratically and vastly outnumbers links in the original data set (adversary outlinks $= 3,000^2 = 9,000,000 =$ 88.4 times the original outlinks).

B. System Overhead

1) Computation: As the PlayerRating agent runs in within the game client process which requires fast execution for playability, speed is more important than space. To measure the speed, benchmarks of the various PlayerRating functions were performed on an Intel® Core 2 Quad system (Q6600/2.4GHz). The results shown in Table II each represent an average of 10,000 executions. Most operations are on the order of μs. The only operations on the order of ms are ExpireRatings() and committing R' in UpdateReputations(). Fortunately these operations are infrequently performed ($\geq pHour$) and may be scheduled to occur with the next logon, loading screen, or idle time when they will imperceptibly impact gameplay.

2) Memory: While the memory footprint is less important than speed, PlayerRating state may be kept small. Specifically, neutral/unknown ratings and ratings from peers with negative reputation are not used and need not be stored. Thus an agent's state is proportional to the number of kept ratings and reputations, which is far less than the square of the number of players:

$$state = |r| + |R| \ll |Players|^2 \quad (5)$$

For example, the ratings and reputations throughout the evaluation merely require 2.4MB of application memory.

IV. Discussion

A. Applications

There are several possible PlayerRating applications. Currently the system is publicly available as a World of Warcraft mod [19] and may be adopted by any player willing to do so. The author as well as in-game and real-life friends used the system for a number of years before quitting the game. It is the author's experience that the system successfully warns players when first interacting with peers who have behaved badly in the past. If implemented within the game's algorithms matching players for group play, weight could be placed on the likelihood the group will get along together.

Further, developers with multiple game titles may build a recommender system on top of PlayerRating to focus online marketing to persons with friends who enjoy those titles.

PlayerRating can completely replace existing friends-list and ignore-list tools by simply treating peers with positive ratings as friends and ignoring peers with negative ratings.

B. Limitations

The current limitations of PlayerRating involve changes to player account information that is not public and therefore is not readily available to an implementation as a user-interface mod. Specifically, a player may quit the game, transfer their character to another server, or rename their character (although only to another unique name). While Sybil attacks are not possible, some system inaccuracy (i.e., rating duplication may exist until those ratings naturally expire). This inaccuracy could possibly be avoided (obviating the need for rating expiration) if the system was aware of relevant changes to peer accounts and peer ratings made while offline.

External to the game, peers sometimes sell their accounts for profit although it is often against the Terms of Service Agreement. Characters changing ownership in this fashion can have an abrupt change in behavior, making existing ratings about them obsolete. Players with positive ratings about those peers may be briefly misled until those ratings are corrected.

Finally, adversaries may attempt Sybil attacks [2] by creating additional game accounts. In general this is prohibitively expensive because creating a game account involves purchasing a copy of the game and paying a subscription fee. However, some games like World of Warcraft offer temporary trial accounts (10-day trial accounts are already abused by "*gold farmers*") which may facilitate short-term Sybil attacks. This may be addressed by incorporating the character's level in the Influence() function (i.e., ratings from high-level characters are more relevant than from low-level characters) and would immediately mitigate trial accounts since those accounts expire before an adversary could reach maximum character level (it takes roughly 300 $pHours$ to reach maximum level).

V. Conclusions

In multiplayer online games, players interact using aliases which unfortunately enable some antisocial behaviors. This paper presented PlayerRating, a distributed reputation system designed specifically for online games, which empowers like-minded players to congregate and avoid malicious peers. By design, the system facilitates incremental deployment and encourages participation. Experimentation shows that the system resists abuse and requires minimal overhead.

References

[1] M. Tresca, "The Impact of Anonymity on Disinhibitive Behavior through Computer-Mediated Communication," Master's thesis, Michigan State University, 1998.

[2] J. R. Douceur, "The Sybil Attack," in *International Workshop on Peer-to-Peer Systems*, March 2002.

[3] netherby, "Playernotes," http://wow.curseforge.com/addons/playernotes/.

[4] eBay Inc., "Feedback Forum," http://pages.ebay.com/services/forum/feedback.html.

[5] Slashdot.org, "Slashdot FAQ: Karma," http://slashdot.org/faq/com-mod.shtml#cm700.

[6] L. Page, S. Brin, R. Motwani, and T. Winograd, "The PageRank Citation Ranking: Bringing Order to the Web," Stanford University, Tech. Rep., Jan 1998.

[7] Z. Gyongyi, H. Garcia-Molina, and J. Pedersen, "Combating Web Spam with TrustRank," in *VLDB*, August 2004.

[8] G. Jeh and J. Widom, "Scaling Personalized Web Search," in *WWW*, May 2003.

[9] S. D. Kamvar, M. T. Schlosser, and H. Garcia-molina, "The EigenTrust Algorithm for Reputation Management in P2P Networks," in *WWW*, May 2003.

[10] J. Golbeck and J. Hendler, "Reputation Network Analysis for Email Filtering," in *CEAS*, July 2004.

[11] A. Adamic, "Zipf, Power-law, Pareto - a Ranking Tutorial," HP Labs, Tech. Rep., October 2000.

[12] P. Bak, *How Nature Works: the Science of Self-Organized Criticality*. Copernicus, 1996.

[13] M. Buchanan, *Ubiquity: The Science of History or Why the World is Simpler than We Think*. Crown Publishers, 2001.

[14] J. Kunegis, A. Lommatzsch, and C. Bauckhage, "The Slashdot Zoo: Mining a Social Network with Negative Edges," in *WWW*, April 2009.

[15] J. Rauch, "Seeing Around Corners: The New Science of Artificial Societies," *The Atlantic Monthly*, April 2002.

[16] C. Dellarocas, M. Fan, and C. A. Wood, "Self-Interest, Reciprocity, and Participation in Online Reputation Systems," MIT Sloan, Tech. Rep., Aug 2004.

[17] D. J. Watts and S. H. Strogatz, "Collective dynamics of 'small-world' networks," *Nature*, June 1998.

[18] Blizzard Entertainment, "World of Warcraft," http://www.worldofwarcraft.com/.

[19] E. Kaiser, "PlayerRating," http://www.curseforge.com/addons/playerrating/.

[20] Computer Language Benchmarks, "C++ vs. Lua," http://shootout.alioth.debian.org/u32q/benchmark.php?test=all&=gpp&lang2=lua.

[21] Warcraft Realms, "WoW Census," http://www.warcraftrealms.com/realmstats.php.