Delft University of Technology
Parallel and Distributed Systems Report Series

The Game Trace Archive: A Technical Report

Yong Guo, Alexandru Iosup
{Yong.Guo, A.Iosup}@tudelft.nl

Report number PDS-2012-005

ISSN 1387-2109

Published and produced by:
Parallel and Distributed Systems Group
Department of Software and Computer Technology
Faculty of Electrical Engineering, Mathematics, and Computer Science
Delft University of Technology
Mekelweg 4
2628 CD Delft
The Netherlands

Information about Parallel and Distributed Systems Report Series:
reports@pds.ewi.tudelft.nl

Information about Parallel and Distributed Systems Group:
http://www.pds.ewi.tudelft.nl/

Abstract

Spurred by the rapid development of the gaming industry and the expansion of Online Meta-Gaming Networks (OMGNs), many gaming studies and measurements have been conducted in recent years. However, few or no traces of games and OMGNs are publicly available to game researchers and practitioners. Moreover, the few traces that are available are shared using diverse formats. This situation is an obstacle in exchanging, studying, and using game traces. To address this problem, we design the Game Trace Archive (GTA) to be a virtual meeting space for the game community. We identify five main requirements to build an archive for game traces, and address them with the GTA. We propose a unified format for game traces, and introduce a number of tools associated with the format. With these tools, we collect, process, and analyze 9 traces of both games and OMGNs. We collect in the GTA traces corresponding to more than 8 million real players and more than 200 million information items, spanning over 14 operational years. We also show that the GTA can be extended to include a variety of real-game trace types. Finally, we discuss possible applications of the GTA in gaming area such as game resource management, Quality of Experience for players, and advertisement.

Contents

List of Figures

List of Tables

1 Introduction

Over the last decade, video and computer gaming have become increasingly popular branches of the entertainment industry. Understanding the characteristics of games and OMGNs such as player behavior [1], [2], traffic analysis [3], [4], resource management [5], [6], etc., is essential for the design, operation, and tuning of game systems. However, only a small number of game traces exist; even fewer can be accessed publicly. As a consequence, previous studies used at most a few traces and no comprehensive comparative analysis exists. Moreover, the few available game traces have diverse formats, which makes it difficult to exchange and use these game traces among game researchers. To address this situation, in this work we propose the Game Trace Archive.

There are thousands of successful games in the world, which attract a large number of players. For example, World of Warcraft, one of the most popular Massively Multiplayer Online Games (MMOGs), and CityVille, a widely spread Facebook game, have each tens of millions of players. A number of online communities have been constructed by game operators and third-parties around single games or even entire collections of games. These communities through the relationships between entities such as players and games, form Online Meta-Gaming Networks [7]. Tens of millions of players currently participate in OMGNs, such as Valve's Steam, XFire, and Sony's PlayStation Network, to obtain game-related information (game tutorials, statistic information, etc.) and use non-gaming functionalities (voice chat, user product sharing, etc.).

We design in this work the GTA as a virtual meeting space for the game community, in particular to exchange and use game traces. Game traces, which can be collected at one timepoint, several or series of timepoints, or through a continuous period of observation, contain many types of game-related data about both games and OMGNs. With the GTA, we propose a unified Game Trace Format (GTF) to store game traces, and provide a number of tools to analyze and process game traces in GTF format. Our goal is to make both the game traces and the tools in the GTA publicly available. Mainly because of the diversity and big size of game traces, there are three main challenges in building the GTA. Firstly, game traces can be collected from many sources. They can focus on any of the multiple levels of the operation of gaming systems, from OMGNs to single game traces, from players to player relationships, and to packets transferred between the players and the game servers. The trace content at different levels is significantly different; moreover, even at the same level the traces may be very different. Secondly, the content of each trace can be complex. Traces may include many kinds of relationships between the game entities (players, guilds, etc.) and detailed information of entities (player name, player date of birth, in-game and meta-game information, etc.). Thirdly, it is difficult to process the large-scale and complex game traces, and to choose from tens of or hundreds of game traces depending on specific scenarios.

This work is further motivated by our ongoing project @larGe, which aims at designing new systems that support gaming at large-scale. Trace archives support several other computing areas including the Parallel Workloads Archive (PWA) [8] for the parallel systems, the Grid Workloads Archive (GWA) [9] for the grid systems, etc. However, non of these previous archives can include the complex game traces we target with the GTA. **Our research is the first work in establishing a comprehensive Game Trace Archive to benefit gaming researchers and practitioners.** The main content of this work is structured as follows:

1. We synthesize the requirements for building an archive in the gaming domain (Section 2).
2. We design the Game Trace Archive, including a unified format and a toolbox to collect and share game and OMGN traces (Section 3).
3. We conduct a comparative analysis using many real game traces (Section 4).
4. We identify several potential applications for the Game Trace Archive (Section 5).

2 Requirements for a Game Trace Archive

Starting from the three main challenges we mentioned in the introduction, in this section we identify five main requirements to build an archive for game traces. Although these requirements are similar to those of archives in other areas, our ambitious goal of building a single format for *all* game data makes it challenging to achieve these requirements.

Requirement 1: Trace collecting. To improve game trace readability and facilitate game trace exchange, a unified game trace format should be provided to collect game traces from diverse sources. Firstly, the game trace format must be able to include many types of game information. Secondly, formats already exist for specific game information; for example, packets sent between game servers and clients. The unified game trace format needs to use these other formats. Finally, due to the rapid evolution of the gaming industry, many new games and game traces may emerge. The format should be extensible to collect traces of future games, while not affecting the usage of old game traces that have already been stored in the archive.

Requirement 2: Trace converting and anonymizing. The raw content of game traces is complex. However, a small part of the content may not be game-related, because of where the game traces are collected. For example, when crawling OMGN traces from their websites, advertisement information in each website may also be collected into the trace. Thus, during the procedure of converting game traces to the unified game trace format, this useless content should be filtered out. Another important issue in converting traces is to provide a privacy guarantee. Many studies [10], [11] have shown that just anonymizing user names is not sufficient to ensure privacy.

Requirement 3: Trace processing. The archive should provide a toolbox to process the converted game traces and generate reports including commonly used characteristics of game traces (e.g., trace size, the number of information items). Furthermore, the toolbox could also be used by archive users to build comprehensive analysis tools.

Requirement 4: Trace sharing. The game traces and the trace processing toolbox must be shared publicly. The archive users may face the problem of selecting proper traces according to their own requirements, especially when the archive expands to store tens of or hundreds of traces. A trace-selection mechanism is needed to address this problem. Allowing archive users to rank and comment on the traces would be a good component in the selection mechanism.

Requirement 5: Community building. The main goal of building a game trace archive is to establish a virtual meeting space for game researchers. Besides the exchange of game traces, a list of research, projects, people, and applications of the archive should be maintained to facilitate further communication of game community.

3 The Game Trace Archive

In this section, we introduce the Game Trace Archive. We first discuss the design of the GTA and then describe the design of the Game Trace Format (GTF).

3.1 The Design of the GTA

Figure 1 illustrates the overall design of the Game Trace Archive, including the GTA members and how the game traces been processed in the GTA. The circles in Figure 1 represent the five requirements we formulated in Section 2.

We envision three main roles for the GTA members. The *contributor*, who is the legal owner of game traces, offers their traces to the GTA and allows public access to the traces. The *GTA Admin* helps contributors to add and convert game traces to the GTA, and manage traces processing and sharing. Our game research team may act as the GTA Admin. The *user* accesses the archived game traces, uses the processing toolbox, and obtains the relevant research information through the GTA. Most users may be game researchers and practitioners, but

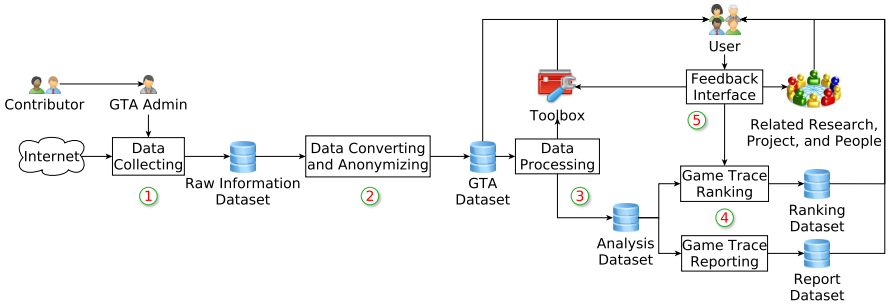


Figure 1: The design of the Game Trace Archive.

we believe that people in other areas (e.g., biologists, economists, social network researchers) can also benefit from the GTA.

In the design of the GTA, the *Data Collecting* module is for collecting game traces from multiple sources: contributors, publicly shared game data repositories, game websites, etc. These game traces have their own formats and some of them may include sensitive information. Firstly, we store these traces in a raw information dataset without additional processing. Then, we map the raw content to the unified GTF (for requirement 1).

The *Data Converting and Anonymizing* module is responsible for converting the game traces from their own formats to the unified GTF, while anonymizing the sensitive information (for requirement 2). The anonymization process, which is a topic of research in itself [10], [11], is outside the scope of this work. The map from the original information and the anonymized information will not be distributed, only the corresponding trace contributor has the authority to read it.

In the *Data Processing* module, a toolbox is provided for comprehensive trace analysis: overview information, such as trace size, period, the number of relationships, and the number of players; in-game characteristics, such as active playing time, average session duration, and the number of played games; relationship graph metrics, such as diameter, link diversity, clustering coefficient. The basic tools in the toolbox can be used to build other processing tools and can also be used to process large scale graphs in other areas, such as social network and viral marketing (for requirement 3).

Two modules are designed for trace sharing (for requirement 4). The *Game Trace Reporting* module receives the trace analysis results from Data Processing module and formulate them into more visible reports. The *Game Trace Ranking* module considers both the overview information from Data Processing module and feedback from trace users to rank the game traces. The Game Trace Reporting and Ranking modules help the GTA users to select a game trace based on a quick knowledge of the basic game trace characteristics.

The *Feedback Interface* is for supporting trace sharing and community building (for requirements 4 and 5). It is the interface for users to submit their feedback after using game traces in the archive. There are four types of feedback: rank of traces, comment on traces, updated or newly designed tools for traces, and research information (e.g., research direction, project, applications for traces). For each game trace in the GTA, we maintain a list of research information derived from feedback. Through these lists, users can know the game community better and users in similar research directions may establish further communication.

3.2 The Design of the GTF

We propose the Game Trace Format to facilitate the exchange and ease the usage of game traces. It is a unified format to cover many different types of game traces and to include complex and detailed gaming information in each trace. In this subsection, we introduce the main elements of the design of the GTF.

Figure 2 shows the structure of the Game Trace Format. Briefly, the GTF consists of three datasets:

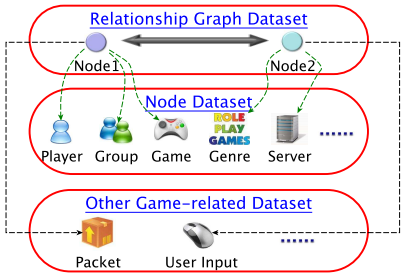


Figure 2: The structure of the Game Trace Format.

the *Relationship Graph Dataset*, the *Node Dataset*, and the *Other Game-related Dataset*. These datasets are responsible for storing different kinds of content in game traces respectively.

3.2.1 Relationship Graph Dataset

From our observation, in many game traces the relationships (e.g., play with, send message to, member of) between many kinds of game entities (e.g., player, group, game in OMGN, genre in OMGN) are significant for the operation of these games and OMGNs. Thus, we model these relationships as *edges* in a *graph* where *nodes* are game entities. We include in the GTF the Relationship Graph Dataset to include the relationships presented in game traces. The Relationship Graph Dataset has three sub-datasets: with *basic edge information*, with *detailed edge information*, and with *meta-information*.

Table 1: Format for basic edge information.

ID	Column	Description
1	RowID	*Int.* A count field. The lines in this file should be sorted by ascending RowID.
2	SrcType	*Int.* The type of *source node*, including player, community, team, faction, guild, etc. Each node type has a unique integer number assigned to it, the correspondence of type and number can be found in meta-information *Format for node type*. Source node: the starting node of directed edge. For undirected edge, it is the node appearing first in the raw data of the edge.
3	SrcID	*Int.* ID of source node. To protect the source node privacy, their names are anonymized by assigning each unique source node a unique ID.
4	DstType	*Int.* The type of *destination node*, including player, community, team, faction, guild, etc. Each node type has a unique integer number assigned to it, the correspondence of type and number can be found in meta-information *Format for node type*. Destination node: the ending node of directed edge. For undirected edge, it is the node appearing last in the raw data of the edge.
5	DstID	*Int.* ID of destionation node. To protect the destination node privacy, their names are anonymized by assigning each unique destination node a unique ID.
6	EdgeType	*Int.* The type of edge between source node and destination node, indicating the relationship between source node and destination node. Each type is represented by a unique integer number, the correspondence of type and number can be found in meta-information *Format for edge type*.

The basic edge information (Table 1) includes the essential or must-have elements for relationships (e.g.,

edge/node type, edge/node identifier). By using different edge types and node types, various relationships in game traces can be presented in our format.

Table 2: Format for the fixed part in detailed edge information.

ID	Column	Description
1	RowID	*Int.* Same to RowID in the *basic edge information*.
2	TimeStamp	*UnixTimeStamp*, in millisecond. The beginning time of the recorded event. Can be Null.
3	EdgeLifetime	*Float*, in millisecond. Duration of the edge exsiting. Can be Null.
4	SrcScore	*String.* Source node payoff. The meaning of SrcScore can be found in meta-information *Format for edge type*. Can be Null.
5	DstScore	*String.* Destination node payoff. The meaning of DstScore can be found in meta-information *Format for edge type*. Can be Null.
6	ExtEdgePath	*String.* The path set to access the *extended part* of detailed edge information.

To store other diverse edge-related information, we use the detailed edge information sub-dataset, which includes two parts, the *fixed part* and the *extended part*. The fixed part (Table 2) stores typical attributes of edges (e.g., timestamp, edgelifetime). The extended part stores extended edge attributes that are not common for all relationships. Since these extended edge attributes differ per trace, there is no exact format for the extended part. For each attribute, we use one column to store its value. The design of extended part makes it possible to cover new types of edge-related information.

Table 3: Format for edge type in meta-information.

ID	Column	Description
1	EdgeType	*Int.* The integer number assigned to edge type. The lines in this file should be sorted by ascending EdgeType.
2	EdgeDirectivity	*String.* Directed or Undirected edge.
3	EdgeTypeTerm	*String.* A short term describing edge type.
4	EdgeTypeDef	*String.* A detailed definition or description of edge type.
5	SrcScoreDef	*String.* The meaning of *SrcScore*.
6	DstScoreDef	*String.* The meaning of *DstScore*.

Table 4: Format for node type in meta-information.

ID	Column	Description
1	NodeType	*Int.* The integer number assigned to node type. The lines in this file should be sorted by ascending NodeType.
2	NodeTypeTerm	*String.* A short term describing node type.
3	NodeTypeDef	*String.* A detailed definition or description of node type.

The meta-information includes the overview of the Relationship Graph Dataset and the definitions of all the edge (Table 3) and node types (Table 4).

3.2.2 Node Dataset

The Node Dataset is designed to include detailed node information. Since the information of different types of nodes can be diverse, to store this information in a unified format, we categorize the information first.

We divide the complex node information into *static* (keep constant with time, e.g., player gender, date of birth) and *dynamic information* (change with time, e.g., player rank, level).Each type of node has its own *node sub-dataset*. For each type of node, we further divide their static and dynamic information into *typical static information*, *typical dynamic information*, *extended static information*, and *extended dynamic information*.

Table 5: Format for the fixed static part in Node Dataset.

ID	Column	Description
1	RowID	*Int.* A count field. The lines in this file should be sorted by ascending RowID.
2	NodeID	*Int.* The integer number assigned to node.
3	TypStatic1	*String.* Typical static attribute of node.
4	TypStatic2	*String.* Another typical static attribute of node.
...	...	*String.* Another typical static attribute of node.
N+2	TypStaticN	*String.* The Nth typical static attribute of node.
N+3	ExtStaPath	*String.* The path set to access the *extended static part*.
N+4	TypDynPath	*String.* The path set to access the *fixed dynamic part*.
N+5	ExtDynPath	*String.* The path set to access the *extended dynamic part*.

Table 6: Format for the fixed dynamic part in Node Dataset.

ID	Column	Description
1	RowID	*Int.* A count field. The lines in this file should be sorted by ascending RowID.
2	NodeID	*Int.* The integer number assigned to node.
3	TimeStamp	*UnixTimeStamp*, in millisecond. The timestamp for dynamic attribute value.
4	TypDyn1	*String.* Typical dynamic attribute of node.
5	TypDyn2	*String.* Another typical dynamic attribute of node.
...	...	*String.* Another typical dynamic attribute of node.
N+3	TypDynN	*String.* The Nth typical dynamic attribute of node.

These four kinds of information are stored in *fixed static part* (Table 5), *fixed dynamic part* (Table 6), *extended static part*, and *extended dynamic part*, respectively in the node sub-dataset. We use the same method (as we do for the extended part in detailed edge information, Section 3.2.1) to store extended static and dynamic attributes. Through its design, specifically through its extended static and extended dynamic parts, our Node Dataset is the first to store the information of many kinds of nodes in a unified format.

3.2.3 Other Game-related Dataset

For more traditional gaming data, such as packets between game clients and servers, match replays, player click streams, etc., we use when possible the de-facto standard formats and store the data in the Other Game-related Dataset. For example, for match replays, we keep their own formats derived from games. The formatted replays, such as StarCraft replays, are used by Weber et al. [12] and Hsieh et al. [13] to study player in-game behavior.

Moreover, we provide detailed introduction files to guide the archive users how to process the formats. Meta-information is also provided, to link the game-related information to its corresponding nodes; for example, the links between players and the packets they have sent.

4 Game Traces and Trace Analysis

In this section, we present and analyze the game traces collected by the GTA. Using our GTA toolbox, we conduct an analysis that focus on four main aspects. Some of this work, specifically, Sections 4.1.3, 4.1.4, 4.1.5, 4.2.2, 4.2.3, 4.2.4, 4.3.2, 4.3.3, has been published in our previous work [14]. The other sections, specifically, Sections 4.1.2, 4.2.1, 4.3.1, are new.

Table 7: Summary of game traces in the GTA.

Trace	Period	Size (GB)	# Nodes (K)	# Links (M)	Genre
KGS[1]	2000/02-2009/03	2	832	27.4	board
FICS[2]	1997/11-2011/09	62	362	142.6	board
BBO[3]	2009/11-2009/12	2	206	13.9	card
XFire[4]	2008/05-2011/12	58	7,734	34.7	OMGN
Dota-League[5]	2006/07-2011/03	23	61	3.7	RTS
DotAlicious[6]	2010/04-2012/02	1	64	0.6	RTS
Dota Garena[7]	2009/09-2010/05	5.2	310	0.1	RTS
WoWAH [15]	2006/01-2009/10	1	91	N/A	MMORPG
RS [16]	2007/08-2008/02	1	0.1	N/A	MMORPG

Table 7 summarizes the nine game traces currently formatted in the GTF. These traces have been collected from five types of games or OMGNs, including board games, card games, RTS games, MMORPG games, and OMGNs. Our GTA can cover game traces collected by ourselves. The KGS and FICS traces include a large number of matches in two popular board games, Go, and chess. The BBO trace is collected from one of the largest bridge sites in the world, where people can play bridge online for free. The dataset of the OMGN XFire contains the information of thousands of games and millions of players, as well as complex relationships between those games and players. Defense of the Ancients (DotA) is mod build on the RTS game Warcraft III. A match of DotA is played by two competing teams, allowing at most 5 players in one team. The main content in the Dota-League, DotAlicious, and DotA Garena traces is plenty of DotA matches played on three different DotA playing platforms. We are also able to include in the GTA existing game traces. WoWAH is a public dataset consists of online session records and attributes of World of Warcraft avatars collected from game servers in Taiwan. RS collects the online player counts of more than 100 servers of the MMORPG game RuneScape. The size in Table 7 is the raw data size of each trace. The node may be player, avatar, or game server, corresponding to each trace. In XFire, we use the link as friendship. For the other traces, we define the link as playing in a same match. With these game traces, we can conduct a comparative analysis of games, inter- and intra-genre.

[1] http://www.gokgs.com/
[2] http://www.freechess.org/
[3] http://www.bridgebase.com/
[4] http://beta.xfire.com/
[5] http://www.dota-league.com/
[6] http://www.dotalicious-gaming.com/
[7] http://replay.garena.com

4.1 Analysis of Workload Characteristics

Provisioning resources is an essential task for online gaming operators. Using the least possible resources to support the workloads generated by players brings maximum profits. However, inadequate provisioning may result in idle resources or the departure of players. In this subsection, we discuss five characteristics of the workloads of online games.

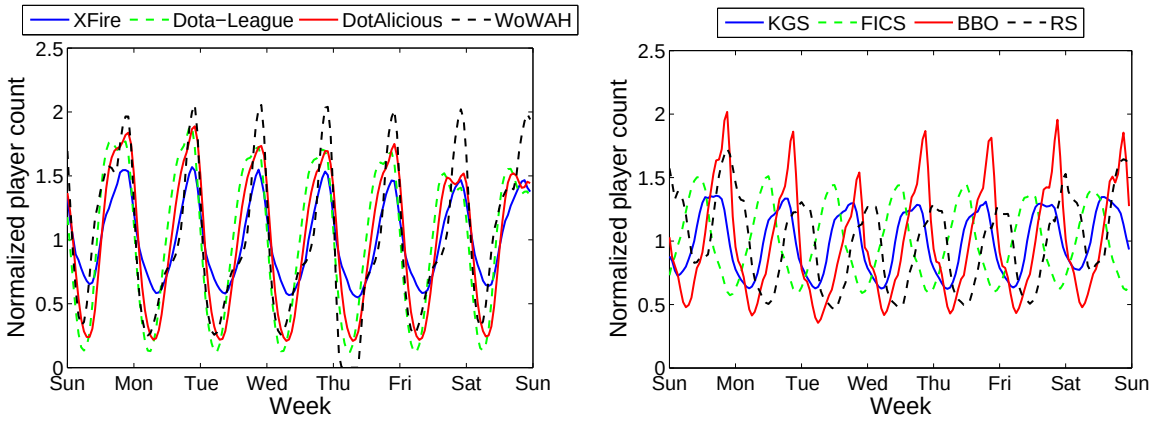


Figure 3: Weekly pattern of normalized online player count.

4.1.1 Weekly pattern of online player count

The online player count is an important metric of the usage and workload of game severs. In this subsection, we study how the online player count fluctuates during the week.

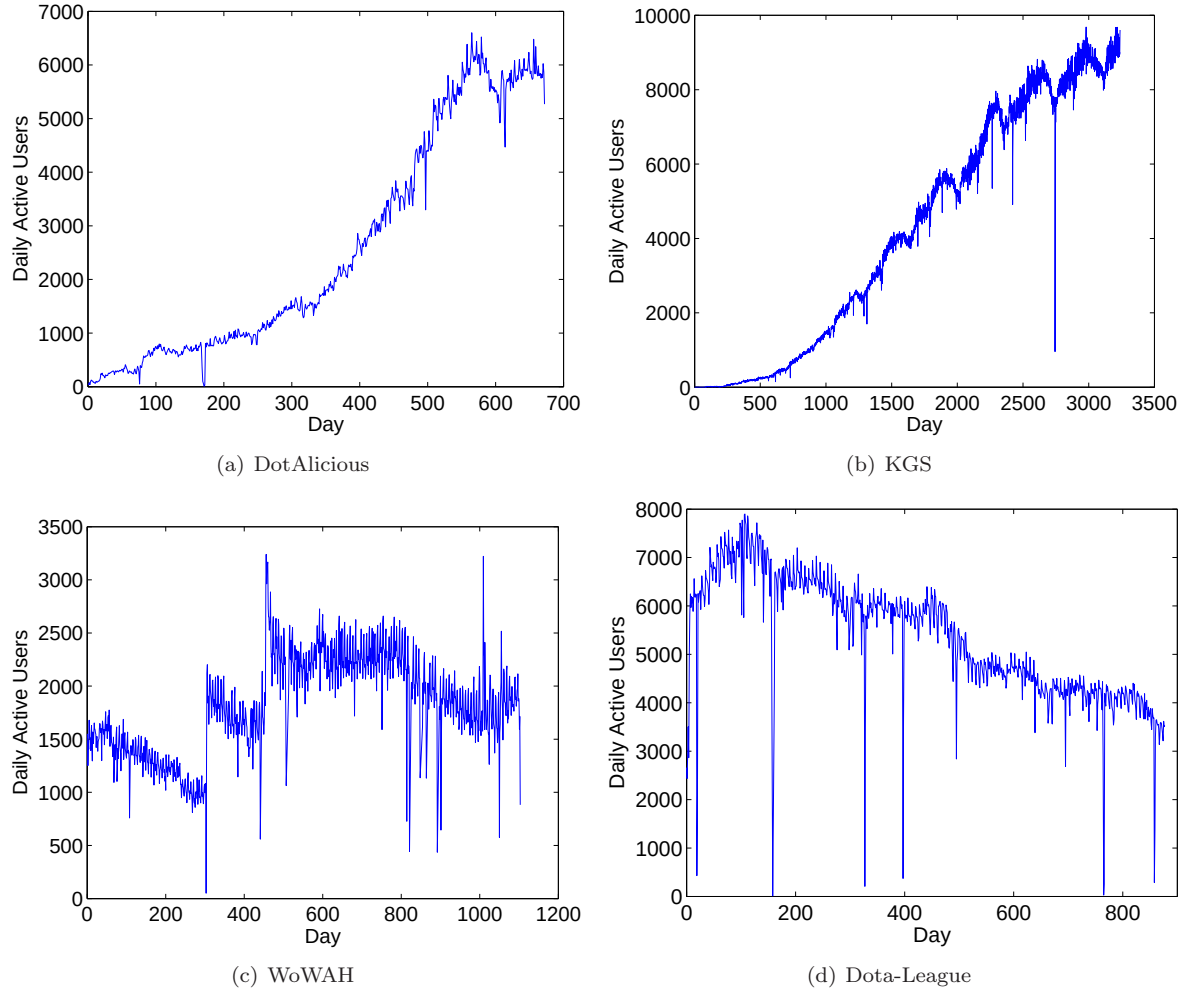
We define the *normalized online player count* at any moment of time, as the ratio between the player count at that moment and the trace-long average player count. Figure 3 illustrates the normalized online player count for eight game traces. The ninth trace (Dota Garena) currently included in the GTA does not have player count information over time. The figure shows that, for each game trace, the online player counts have obvious diurnal patterns; the peak and bottom counts occur at nearly the same time for each day. However, the occurring time of both peak and bottom differ per game. For example, for chess the peak occurs during the day, whereas midnight is the busiest period for all DotA platforms. Un-expectedly and unlike the workloads of web servers, for our traces the player counts do not differ significantly between week days and week ends. Moreover, for DotAlicious the player presence is even lower during week ends. Due to the scheduled weekly maintenance, there is an outage period on Thursday morning in WoWAH; the normalized player count drop to 0, see Figure 3 (left).

Lee et al. [15] investigate the daily, weekly, and monthly patterns of avatar counts in World of Warcraft. Chambers et al. [5] study the online player count over a 4-week period in FPS, casual, and MMORPG games. Both of their results show a similar weekly pattern as our result, but for fewer games.

4.1.2 Daily Active Users

Daily Active Users (DAU) is another important metric of the workloads of online games. On understanding the evolution of DAU, game operators can deploy game resources better in long term.

Figure 4 shows the evolution of DAU for four game traces: DotAlicious, KGS, WoWAH, and Dota-League. We collected the datasets of DotAlicious and KGS from their launch, but the WoWAH and Dota-League datasets start at the date later than their setup. Both DotAlicious and KGS attracted more and more player significantly



(a) DotAlicious (b) KGS

(c) WoWAH (d) Dota-League

Figure 4: Daily Active Users.

from their establishment. By contrast, the DAU of Dota-League dropped gradually until the end of the trace, which might be one of the reasons why the Dota-League platform shut down in November, 2011. In WoWAH, the release of new game contents or expansions can result in surges of the DAU. For example, on the day of around 450, the beginning of April, 2007, when an important expansion of WoW - The Burning Crusade was released(in Taiwan). However, the previous study [17] on another MMORPG EVE Online shows that updates slightly impact player growth.

4.1.3 Match count per player

Figure 5 depicts the match count (total number of matches played) per player of Dota-League, KGS, and FICS. The Cumulative Distribution Function (CDF) is depicted against the left vertical axis. The Probability Distribution Function (PDF) is depicted against the right axis and only for the Dota-League dataset in Figure 5 (left). The right vertical axis and the horizontal axis are log-scale. Since there is a number of computer players (bots) existing in KGS and FICS servers, we filter out the top 0.5% "players" in terms of their match count,

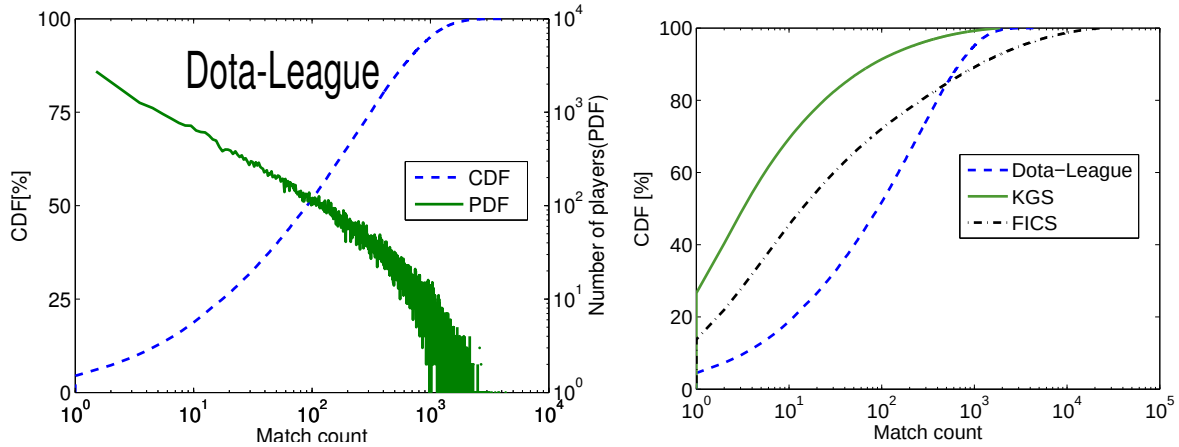


Figure 5: Match count per player for Dota-League, KGS, and FICS.

assuming these are all bots.

Although the number of board game players is larger, a significant portion of them play only a few matches: about 25% of KGS players and about 15% of FICS players participated in only one match. However, this value is much lower for Dota-League (3%). The match count per player of match-based games is heterogeneous. The median values of the match count in Dota-League, KGS, and FICS are 91, 4, and 15, while the 99.5% quartiles are 1,945, 1,908, and 23,396, respectively. Nearly half of the online board game players participate in less than 15 matches.

The match count per player follows a long tail distribution, where the maximum value can be over a hundred times larger than the median value. We fit the match count per player against the power-law, log-normal, weibull, exponential, normal, and gamma distributions using the maximum likelihood estimation technique. The best-fitting distribution has the smallest Akaike information criterion with correction (AICc) [18]. The match count per player can be best fitted, for KGS and FICS, using the power-law distribution. For Dota-League, the log-normal distribution is the best fit.

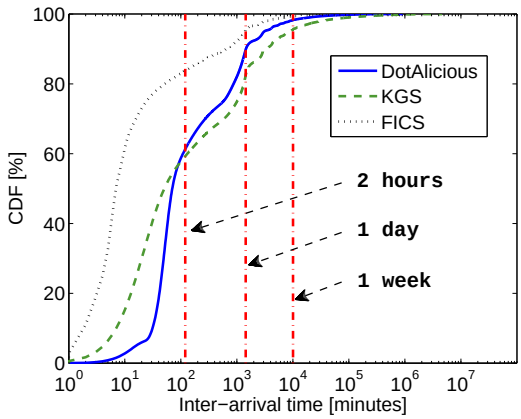
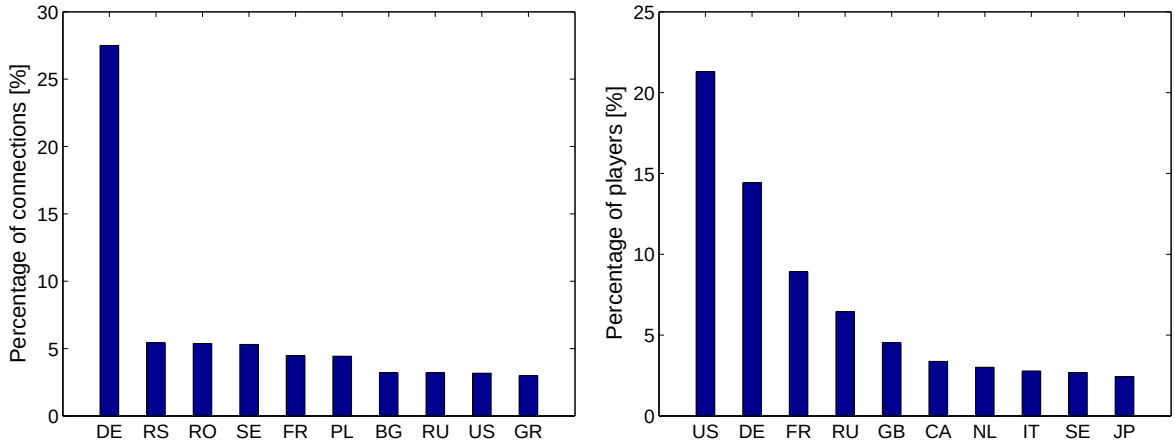


Figure 6: CDF of inter-arrival time.

4.1.4 Inter-arrival time distribution

We define the (match) inter-arrival time as the duration between the start times of two consecutive matches of a player. The inter-arrival times of a player represent his frequency of playing matches. Figure 6 shows the CDF of inter-arrival times across all the players of DotAlicious, KGS, and FICS. Over 80% of the inter-arrival times are less than one day and over 95% of the inter-arrival times are less than one week in these datasets. This indicates that a large percentage of players comes back to play shortly after their last match; this is similar to MMORPG EVE Online [5]. The distribution of the inter-arrival times peaks at 47 minutes. Given the average duration of DotA matches (41 minutes), it means players tend to play two consecutive DotA matches. The inter-arrival time of DotA matches is longer than for World of Warcraft (median 20 minutes) [19]. As players are very likely to be continuously playing in successive matches, it could be beneficial to build a highly-efficient P2P-based MMVE using the DotA players' own computers as servers.



(a) DotAlicious (b) DGS

Figure 7: Geographic distribution of connections and players.

4.1.5 Geographic distribution of connections and players

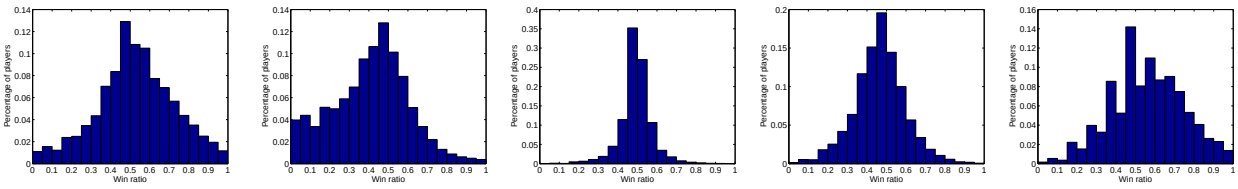
International use can be a metric to measure the success of online games. Many of the popular games, such as World of Warcraft, Starcraft, etc., are catering to subscribers from all over the world. One of the problems in serving players from different countries is how to deploy geographically distributed game servers while keeping a reasonable quality of experience for all players (see [16] and references within). Investigating the geographic distribution of game workloads can support addressing this problem.

For each match in DotAlicious, the countries where the players connect from are recorded. We count connections from each country, over all matches. Figure 7 shows the geographic distribution of connections in DotAlicious and that of the players of DGS[1]. The workloads of games are not equally distributed, since the top 10 countries account for a majority of the connections or the players. For DotAlicious, probably because nearly half of the European game servers are located in Germany, there are significantly more connections from Germany than from other countries. For both of these games, the top 10 countries are located in North America and Europe, which should therefore be key areas in resource deployment. For comparison, Feng et al. [20] analyzed the distribution of online FPS players, and also found that most players are located in North America, Europe, and Asia.

[1]Data from http://www.dragongoserver.net/statistics.php?stats=1, 2012-06-27

4.2 Analysis of Win Ratio

A big skill gap between players in a match can result in a disappointing experience. The more skillful players may lack challenge; the less skillful players may give up. Most of the match-based games have implemented a rating system to help players recognize their skill level and find proper opponents. The quality of rating system affects an elementary metric for match-based game players, the win ratio, defined as the percentage of wins of the total of wins and losses. In this subsection, we study the distribution of win ratios and the correlations between winning and several other characteristics, including the amount of played matches, friendship, and current rating systems.



(a) KGS (b) FICS (c) Dota-League (d) DotAlicious (e) DotA Garena

Figure 8: The distribution of player win ratios.

4.2.1 The distribution of win ratios

First of all, we investigate the overview of win ratios of players. Figure 8 depicts the distributions of the win ratios for two board game traces and three DotA traces. To improve the accuracy of the analysis, players who play less than 10 matches are filtered out. The win ratio distributions of KGS, DotAlicious, and Dota-League almost follow the Gaussian distribution, with the average win ratios shift around 0.5. There are more players whose win ratios are less than 0.5 in FICS, which indicates chess beginners may find proper opponents easier. For Dota Garena, larger fraction of players own higher win ratios (more than 0.5) compared with that of the other two DotA platforms. The reason might be that matches collected from Dota Garena are uploaded by players, who may tend to show their victories.

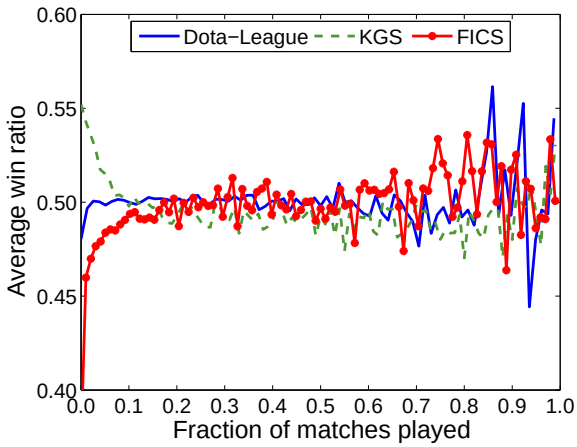


Figure 9: Average win ratio over fraction of matches played.

4.2.2 Win ratio vs. match count

Intuitively, playing more matches should lead to better gaming skills and thus higher win ratios. Figure 9 shows the average win ratio versus the match count for Dota-League, KGS, and FICS. We normalize the match count based on the maximum values observed for each game. We then slice the normalized match count range into 100 bins and calculate the average win ratio for each bin.

For beginning players (range [0.0-0.1] in the horizontal axis), the evolution trends of the win ratios of KGS and FICS are opposite. The reason may be that, when registering in these games, players are suggested to fill in their skill levels. However, the default value of KGS is low, while that of FICS is a median value. Thus, beginners in KGS whose actual skill level may be higher, will play with less skillful players and gain a higher win ratio, and vice-versa for FICS. Beyond this starting zone, the win ratio fluctuates around 0.5. Thus, there is no direct correlation between win ratio and match count (with the correlation coefficient $R = 0.1108$, p-value $P = 0.3342$). For advanced players (range [0.7-1.0]), the fluctuation is larger, which might be caused by the different types of players. People with longer player lifetime may be professional players or hardcore players with varying skill levels.

4.2.3 Win ratio vs. friendship

In many types of competitions, team-spirit and cooperation with friends have great effect on the success. In the gaming field, we can also find friendship and cooperation between players in many different types of games, for example in online bridge [21]. In this subsection, we analyze the impact of friendship on win ratios in DotA.

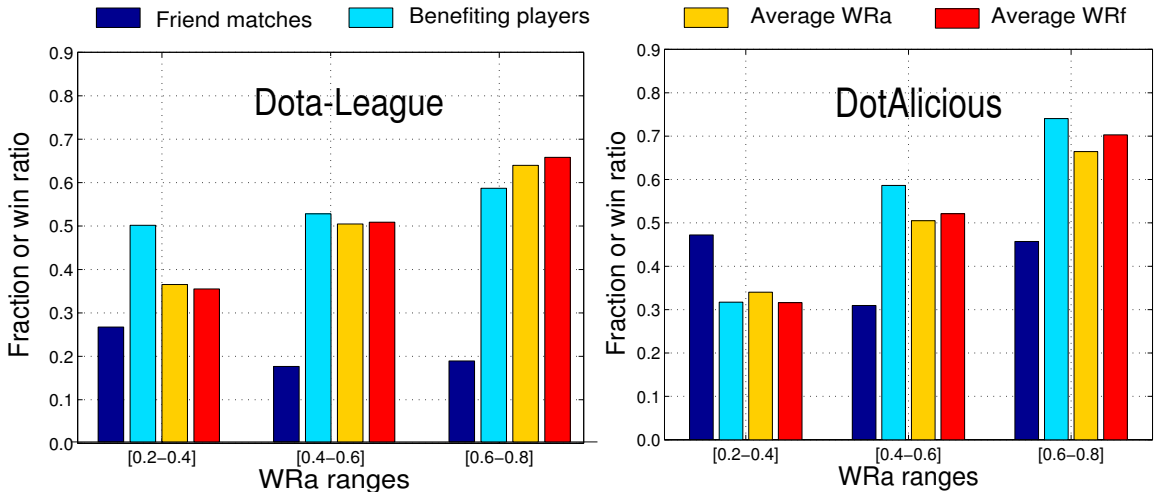


Figure 10: Performance of players for different win ratio ranges.

We find two kinds of relationships between players in Dota-League and DotAlicious. In Dota-League, players have a friend list called "buddy link", whereas in DotAlicious players can be a member of a clan with maximum 8 members. We consider both the buddy link and the clan membership as friendship. Figure 10 illustrates the performance of players, for 3 equally sized win ratio ranges. In this figure, friend matches refer to the matches players play with their friends in the same team. WRa is the win ratio of all matches; WRf is the win ratio of matches played with a friend. Benefiting players are players whose WRf is higher than their WRa. Since the fraction of players whose WRa is less than 0.2 or more than 0.8 is very small, we eliminate these players as outliers and focus on the WRa range from 0.2 to 0.8. The vertical axis is used to represent the fraction of friend matches, the fraction of benefiting players, average WRa and average WRf.

According to the "Friend matches" bars, more than half of the players play individually; also, players in Dota-League play less games with friends than players in DotAlicious. The reason is probably that the Dota-League matchmaking assigns players randomly, without guarantee to be in the same team with a friend. With the increase of win ratio, more players perform better in matches with their friends and improve their win ratios ("Benefiting players" bars). On average (bars "Average WRa" and "Average WRf"), the players with higher WRa (ranges [0.4-0.6] and [0.6-0.8]) win more matches when they play with friends in a team. However, the increase of the average win ratio is small (under 0.05). Surprisingly, the players with lower WRa (range [0.2-0.4]) lose more matches when they cooperate with friends. To some extent, it implies that DotA is not a beginner-friendly game: beginners can't help each other to higher win ratios. For comparison, Ducheneaut et al. [22] found that cooperation helps players to level up in World of Warcraft; Mason and Clauset [23] found that in Halo: Reach, teams composed of friends, on average, win more games than teams composed of strangers.

4.2.4 Winning prediction of current rating systems

As we mentioned at the beginning of this section, game operators may design or implement their own rating systems. In this subsection, we discuss the existing rating systems in DGS and FICS, and analyze the winning probability (WP) of a player in a match according to the skill level given by the rating systems. The DGS and FICS servers implement the EGF[1] and Glicko[2] systems, which are both based on the Elo[3] rating system, to measure the skill of players, respectively.

Table 8: The winning probability by skill gaps.

DGS				FICS			
Gap	# Matches	%	WP	Gap	# Matches	%	WP
2	389	1.0	0.499	2	1,675,560	1.2	0.502
5	588	1.5	0.524	4	1,821,258	1.3	0.505
11	1,300	3.2	0.518	10	5,094,868	3.7	0.509
25	2,970	7.4	0.484	22	10,019,375	7.4	0.521
57	6,283	15.6	0.534	49	20,980,110	15.4	0.545
129	11,950	29.6	0.550	108	35,523,732	26.1	0.597
290	9,130	22.6	0.617	238	38,386,954	28.2	0.696
652	5,349	13.3	0.710	520	19,009,291	14.0	0.838
1,467	1,962	4.9	0.787	1,137	3,365,828	2.5	0.932
3,299	448	1.1	0.821	2,487	227,260	0.2	0.994
Total	40,369	100.0	0.591	Total	136,104,236	100.0	0.648

We define the winning probability for a specific skill gap as the fraction, from the matches between players whose skill rating differs by the gap, of the matches where the winner is the player with higher skill rating, based on the rating before the match. In general, if the skill gap is less than 100 in DGS or less than 200 in FICS, the skill levels of the two players are very similar. We logarithmically assign the skill gap value into 10 bins based on the maximum skill gaps of DGS and FICS (3,299 and 2,487, respectively). Table 8 presents the change of winning probability from the lowest to the highest skill gap. After filtering out abnormal matches (such as draws, matches with handicap[4] in DGS, etc.), we obtain 40,369 cleaned DGS matches and 136,104,236

[1] http://senseis.xmp.net/?FIDETitlesAndEGFGoRatings
[2] http://senseis.xmp.net/?GlickoRating
[3] http://senseis.xmp.net/?EloRating
[4] http://senseis.xmp.net/?Handicap

FICS matches.

When the skill gap is small (under 100), the winning probability of the higher skill player is around the expected value of 0.5. As the skill gap increases, the higher skill players have higher probability to win matches. When the skill gap is high enough, there is still a probability for the lower skill player to win the match, especially in DGS. The reason may be that a number of players play casually in online board games, and the amount of DGS matches is not as large as that of FICS. The percentage of matches with large skill gaps (over 500) is over 16%, which indicates that players in those matches may not have a good gaming experience and that the board game operators should improve the quality of their matchmaking systems.

4.3 Analysis of Player Behavior and Evolution

Due to the large variety of available games, players have many choices. The selection can be influenced by both the content of games and the quality of the offered service. Retaining players with longer player lifetime (from the first time till the last time a player has been seen) can yield more revenue for game companies. In this subsection, we study how many matches played by departure players over their player lifetimes and analyze how the player lifetime interacts with the number of in-game friends, and in-game play strategy.

4.3.1 Normalized match count of departure players.

In this subsection, we analyze the player departure behavior in Go, chess, and DotA. We assume that a departure player is a people who did not play any match in the last month of the game trace. We count the number of played matches every week for each departure player. After that, we map the weekly match count into the percentage of player lifetime. Next, we calculate the average match count of all players at the same percentage of their lifetime. Finally, as shown in Figure 11, we plot the *normalized match count* as the ratio between the match count at one percentage and the average match count of the whole player lifetime. We can find that departure players play mostly at the beginning, and then, they play less matches gradually towards the ends, regardless of the type of games and platforms.

Similarly, Feng et al. [17] shows the EVE Online players spend less time on the game before quitting, with shorter session time and longer intersession time. Tarng et al. [24] try to predict ShenZhou Online gamer's departure on their average daily playtime and playing density.

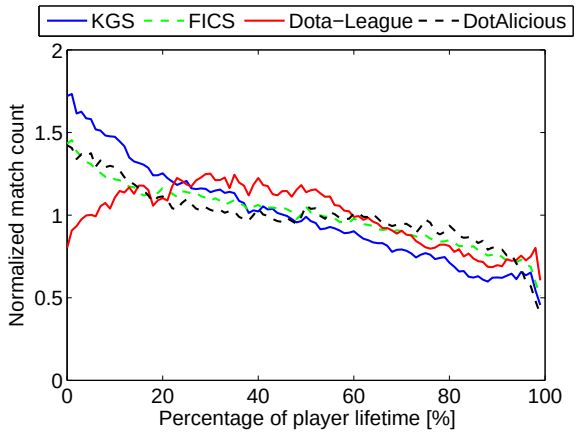


Figure 11: Normalized match count over player lifetime.

4.3.2 Player lifetime and match count vs. number of friends

We have discussed the influence of friendship on the performance of players in matches (Section 4.2.3). We now study how the friendship affects player lifetime in DotA-League.

Figure 12 illustrates that players with more friends generally stick to the game longer (left vertical axis) and play more matches (right vertical axis). The friendship does have a strong correlation with player lifetime ($R = 0.8412$, $P < 0.01$). Thus, it would be a good idea for the game operators to maintain players by reminding players to make more friends and by providing convenient services to support social interaction. Unlike Facebook, where users have on average of about 130 friends, most players here have at most 60 friends.

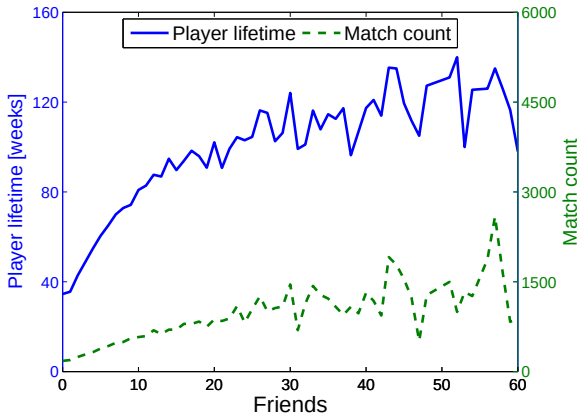
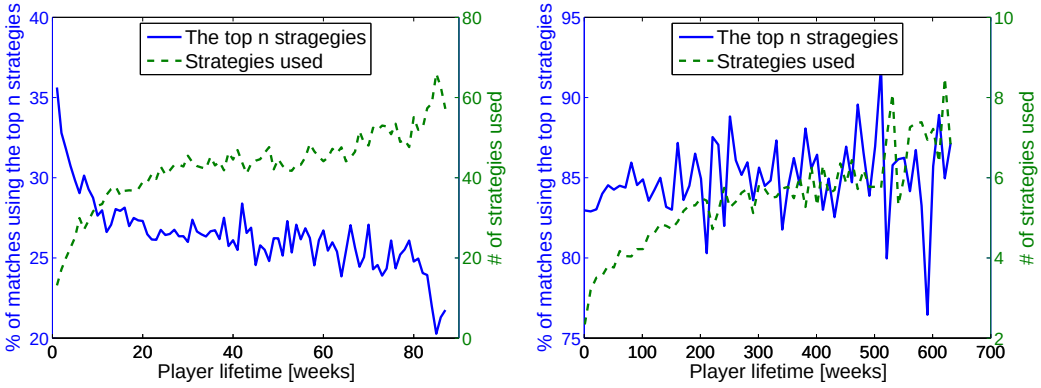


Figure 12: Number of friends with player lifetime and match count.

4.3.3 Player lifetime vs. play strategy

Predicting player lifetime is an important task for a game company, because if a company can predict how long the player's game lifetime will be, it can both leverage some methods to prolong player lifetime and better market in-game or side products. Although lifetime prediction has been researched for the past 5 years, only a small fraction of these studies have taken into account the players' in-game behavior.



(a) DotAlicious (b) FICS

Figure 13: The evolution of play strategy with player lifetime.

We study the number of strategies players use in DotAlicious and FICS. In DotAlicious, the in-game characters of players are called "heroes". Different heroes represent different types of strategies. In FICS, for simplicity, the first move of a player in a match represents a strategy. There are over 100 available strategies in our DotAlicious dataset and 20 available strategies in FICS.

Figure 13 shows the average number of strategies used by players with different lifetimes. The horizontal axis shows the lifetime of players, and the percentage of matches using the top n strategies--by match count--is depicted against the left vertical axis, while the right vertical axis shows the number of strategies used. The value of n is 3 and 1 in DotAlicious and FICS, respectively. In general, there is a positive correlation between player lifetime and the number of used strategies ($R = 0.8789$, $P < 0.01$): the longer the player lifetime is, the more strategies he will have used. The top n strategies account for a large majority of matches, which indicates that players are conservative. According to Figure 13, if the amount of available strategies is larger, players may spend more time in exploring all the strategies. As for the game operators, they may need to provide (better) awards to encourage players to try new strategies.

4.4 Analysis of Gaming graph

The relationships between entities in game traces can form gaming graphs. As we will argue in Section 5, the structure and evolution of gaming graphs may become essential for game operation. The formatted Relationship Graph Dataset in the GTA makes it easier to investigate the graph characteristics of various games. In this subsection, we analyze the gaming graph for seven traces, except WoWAH and RS, which do not have the graph information. We form the graphs as follow: for all traces, each node represents a player. Then, for XFire each edge expresses a player-to-player friendship; for all the other traces, each edge represents a played game match.

Table 9: Graph metrics of the largest connected component.

Trace	**# Comps**	**# Nodes**	**# Edges**	**d** ($\times 10^{-4}$)	$\bar{\mathbf{D}}$
KGS	6,099	819,249	17,884,783	0.53	44
BBO	11	206,333	13,654,906	6.41	132
FICS	2,574	356,244	50,460,185	7.95	283
XFire	198	7,733,276	29,257,283	0.01	8
Dota-League	1	61,171	50,870,316	272	1663
DotAlicious	1	64,083	20,006,143	97.4	624
Dota Garena	1,544	291,706	2,767,594	0.65	19

 # Comps is the number of connected components in the graph.
 # Nodes, **# Edges**, **d**, and $\bar{\mathbf{D}}$ are the number of nodes, the number of edges, the link density, and the average degree of the largest connected component, respectively.

Table 9 shows, for each trace, typical graph metrics of the largest connected component of the gaming graph. A connected component is a sub-graph in which all pairs of nodes are reachable. The largest connected component contains almost all the nodes of the whole graph in each game trace, which means that nearly every player can reach any other player in the trace. From the link density, all these largest connected components are *not* dense. For XFire, the average friend count per user ($\bar{\mathbf{D}}$ in Table 9) is much lower than for Facebook: 8 vs. 130. For the other traces, the relationship counts range from 19 to 1663; we leave for future work a study of the strength and meaning of such game relationships.

5 Applications of the Game Trace Archive

Many research directions may benefit from the comprehensive game information stored in the GTA. In this section, we discuss the potential applications of the GTA. We look at three main application categories, game resource management, Quality of Experience (QoE) for players, and advertisement.

5.1 Game resource management

Resource management is critical for game and OMGN operators and players. Inadequate management can lead to service shut-down, player departure, etc. We identify four main uses of GTA traces for game resource management.

1. *Provisioning the overall resources needed for game and OMGN operation* [7]. Knowledge about the evolution of the player graph (e.g., a graph formed as in Section 4.4), from the simple player count to the complex guild information, can enable accurate prediction of the needed resources.

2. *Deploying resources.* The GTA can help in understanding the change of game workloads with different time patterns (diurnal, weekly, etc.). Idle game resources of one game may be used to support other games or applications during a specific period in a day or week [5].

3. *Reducing resource consumption.* The use of positional update messages, which account for a large portion of network consumption, can be tuned to play characteristics [25]. We believe that the GTA can further help with this tuning process. Firstly, we may identify groups and core group players from player graphs. Then, using the core players as proxies for their groups, the game may transfer only the core players' characters positional updates and significantly reduce the network consumption.

4. *Assigning relevant resources to different types of players.* The types or importance of players in a group can be identified from their playing behavior. For example, when supporting a group voice-communication [26], the group leaders may need more upload bandwidth than other group members for sending their commands, because they have multiple voice channels, one for each other group member.

5.2 Quality of Experience for players

The Quality of Experience for players covers a wide range of aspects. In the following, we list four applications that could use the GTA to improve the QoE.

1. *Improving match-making systems* [27]. Considering more player information stored in the GTA such as skill, rank, win ratio, and the friendship graph may improve the quality of match-making systems and player gaming experience. For example, considering friendships and assigning friends in a same team may allow better match-making.

2. *Detecting cheaters* [28]. The behavior of cheaters may be different from that of normal players, especially different from that of the cheaters' graph neighbors. Identifying graph neighbors and extracting their behavior pattern may be helpful for distinguishing and confirming cheaters.

3. *Building reputation systems* [29]. The reputation of a player (ratee) is calculated from ratings given by the other players (raters). During the calculation, each rating should have its own weight, which may be assigned according to many metrics derived from the GTA data, such as the level and online time of rater, the relationship of rater and ratee in the gaming graph, etc.

4. *Recommending in OMGNs: games, friends, videos, and screenshots.* Better recommendation should be made based on players' interests and their relationship graph.

5.3 Advertisement

Advertisement is an important source of revenue for game operators. Successful game advertisements should meet at least the following requirement: for advertisers, advertisements should attract more users than their expectation; for game and OMGN operators, they should obtain more income from advertisement than the revenue lose due to the effect of user experience. To achieve this requirement, the content of advertisements should be well designed and the placement of advertisements should be well selected. We focus in this work on potential application of the GTA data to select the location and moment to integrate advertisement.

1. *Integrating advertisements into games with less influence to gaming experience* [30]. If an advertisement interrupts player immersion, especially during important and time-limited tasks, the advertisement may fail. What is worse, this may results in player departure. Analyzing the type and frequency of player in-game operations may help advertisers and game operators to integrate seamless advertisements.

2. *Finding places for advertising in OMGNs.* Delivering advertisements in homepages of OMGNs is a common way to prompt potential users. However, there may be other good places for displaying advertisements, for example, popular discussion topics. These popular topics can be identified from the GTA data, such as the number of responses, the influence of participants, and the links between participants in OMGNs.

6 Related Work

In the game community, many researchers have published their work by analyzing their own game traces. However, the type and number of game traces used in the previous work are few, and comprehensive comparative experiments are seldom. Most of those research are based on individual game trace [1], [2], [3], [4]. In [5], the authors study the characteristics of online games by four game traces in three different game genres. However, the traces used in this work only cover a small part of the online games and the analysis of game traces from same game genre is not sufficient.

A number of archives have been build in the other computer science areas. The Internet Traffic Archive (ITA) [31] archives internet network traffic traces for the research of network characteristics. The Parallel Workloads Archive (PWA) [8] collects workloads from parallel environments. The Grid Workloads Archive (GWA) [9] is a grid trace repository and a community center for the grid area. For the peer-to-peer community and wireless network community, the Peer-to-Peer Trace Archive (P2PTA) [32] and CRAWDAD [33] are designed, respectively. These archives are limited to their specific areas, they can not cover the game traces.

Social network and large-scale graph analysis are popular research topic in recent years. Large-scale graphs also exist in the gaming area, forming by the game entities and their relationships. We already store more complex information, such as more types of nodes and relationships, in our gaming graphs. Many formats have been designed to enable the exchange of social networks and graphs, but none of them is able to cover the gaming graphs. The SNAP [34] project is a high efficient library to analyze large scale networks and graph, however, the format in this project can not be extended to express multi relationships. The DyNetML [35] is an XML-based format which is extensible to express abundant social network data. The disadvantage of this format is that plenty of markups and delimiter tags are needed to store each data item, which consumes more storage space and require more memory in data processing. This disadvantage will become truly visible when this format is used to store large-scale gaming graphs.

7 Conclusion and Future Work

We have designed the Game Trace Archive to address the lack of game traces for the game community. We have proposed a unified Game Trace Format that underlies a generic game and OMGN data repository. Our

format includes the Relationship Graph Dataset, the Node Dataset, and the Other Game-related Dataset. We have also provided a toolbox and mechanisms to facilitate trace sharing and community building.

The currently archived traces have shown that the GTA can already store different types of game traces, including board game, card game, RTS, MMORPG, and OMGN. Our example trace analysis has revealed that the GTA can support comparative analysis of gaming traces on the workloads of online games, winning of match-based games, player behavior and evolution, and gaming graph.

We have discussed potential applications for using the GTA in game resource management, Quality of Experience for players, and advertisement.

For the future, we plan to improve the Game Trace Archive in three ways: find more contributors and game traces to enrich the dataset, process game traces in more depth to generate more comprehensive reports, extend the functionality of GTA by building game trace generators.

Data Availability and Contributions

The Game Trace Archive will be available online at: *http://gta.st.ewi.tudelft.nl*. We are looking for contributors who would like to share their game traces through the GTA.

Acknowledgments

We would like to thank Marcin Biczak and Boxun Zhang for useful discussions about the design of the GTA, Siqi Shen and Otto Visser for trace processing. We thank Stephen van der Laan and Paulo Anita for support with this technical report. The first author is supported by a joint China Scholarship Council and TU Delft grant, and by the NSFC No.61272483. This work was supported by the STW/NWO Veni grant @larGe (11881). This work was also supported by the EIT ICT Labs project EUROPA (12015).

References

[1] M. Suznjevic, I. Stupar, and M. Matijasevic. MMORPG player behavior model based on player action categories. In *NetGames*, 2011. 4, 22

[2] N. Ducheneaut, N. Yee, E. Nickell, and R.J. Moore. The life and death of online gaming communities: a look at guilds in world of warcraft. In *SIGCHI*, pages 839–848. ACM, 2007. 4, 22

[3] K.T. Chen, P. Huang, and C.L. Lei. Game traffic analysis: an MMORPG perspective. *Computer Networks*, 50(16):3002–3023, 2006. 4, 22

[4] J. Kinicki and M. Claypool. Traffic analysis of avatars in Second Life. In *NOSSDAV*, 2008. 4, 22

[5] C. Chambers, W.C. Feng, S. Sahu, D. Saha, and D. Brandt. Characterizing online games. *TON*, 18(3):899–910, 2010. 4, 11, 14, 21, 22

[6] A. Denault, C. Canas, J. Kienzle, and B. Kemme. Triangle-based obstacle-aware load balancing for massively multiplayer games. In *NetGames*, 2011. 4

[7] Siqi Shen and A. Iosup. The XFire online meta-gaming network: observation and high-level analysis. In *MMVE*, 2011. 4, 21

[8] The Parallel Workloads Archive Team. Parallel Workloads Archive, July 2007. http://www.cs.huji.ac.il/labs/parallel/workload/. 4, 22

[9] A. Iosup, H. Li, M. Jan, S. Anoep, C. Dumitrescu, L. Wolters, and D.H.J. Epema. The Grid Workloads Archive. *FGCS*, 2008. 4, 22

[10] A. Narayanan and V. Shmatikov. De-anonymizing social networks. In *Security and Privacy*, pages 173–187, 2009. 5, 6

[11] A. Sala, X. Zhao, C. Wilson, H. Zheng, and B.Y. Zhao. Sharing graphs using differentially private graph models. In *IMC*, pages 81–98, 2011. 5, 6

[12] B.G. Weber and M. Mateas. A data mining approach to strategy prediction. In *CIG*, 2009. 9

[13] J.L. Hsieh and C.T. Sun. Building a player strategy model by analyzing replays of real-time strategy games. In *Neural Networks*, 2008. 9

[14] Y. Guo, S. Shen, O.W. Visser, and A. Iosup. An Analysis of Online Match-Based Games. In *MMVE*, 2012. 10

[15] Yeng-Ting Lee, Kuan-Ta Chen, Yun-Maw Cheng, and Chin-Laung Lei. World of Warcraft Avatar History Dataset. In *ACM Multimedia Systems*, 2011. 10, 11

[16] Vlad Nae, Alexandru Iosup, and Radu Prodan. Dynamic Resource Provisioning in Massively Multiplayer Online Games. *TPDS*, 2010. 10, 14

[17] W. Feng, D. Brandt, and D. Saha. A long-term study of a popular MMORPG. In *NetGames*, pages 19–24, 2007. 12, 18

[18] K. P. Burnham. Multimodel Inference: Understanding AIC and BIC in Model Selection. *Sociol. Methods & Research*, 33(2), 2004. 13

[19] Xinyu Zhuang, Ashwin Bharambe, Jeffrey Pang, and Srinivasan Seshan. Player Dynamics in Massively Multiplayer Online Games. *CMU-CS-07-158*, 2007. 14

[20] W. Feng and W. Feng. On the geographic distribution of on-line game servers and players. In *NetGames*, pages 173–179, 2003. 14

[21] Mihaela Balint, Vlad Posea, Alexandru Dimitriu, and Alexandru Iosup. User behavior, social networking, and playing style in online and face to face bridge communities. NetGames, 2010. 16

[22] Nicolas Ducheneaut, Nicholas Yee, Eric Nickell, and R. J. Moore. Alone together?: exploring the social dynamics of massively multiplayer online games. In *SIGCHI conference*, pages 407–416, 2006. 17

[23] Winter A. Mason and Aaron Clauset. Friends FTW! Friendship and competition in Halo: Reach. *CoRR*, abs/1203.2268, 2012. 17

[24] Pin-Yun Tarng, Kuan-Ta Chen, and Polly Huang. On prophesying online gamer departure. In *NetGames*, 2009. 18

[25] D.T. Ahmed and S. Shirmohammadi. Improving online gaming experience using location awareness and interaction details. *Multimedia Tools and Applications*, pages 1–18, 2011. 21

[26] T. Hildebrandt, S. Bergsträßer, C. Rensing, and R. Steinmetz. Dynamic voice communication support for multiplayer online games. In *NetGames*, 2008. 21

[27] J.S.T. Fritsch, B. Voigt, and J. Schiller. The next generation of competitive online game organization. NetGames, 2008. 21

[28] Hsing-Kuo Pao, Kuan-Ta Chen, and Hong-Chung Chang. Game Bot Detection via Avatar Trajectory Analysis. *IEEE Transactions on Computational Intelligence and AI in Games*, 2010. 21

[29] E. Kaiser and W. Feng. PlayerRating: a reputation system for multiplayer online games. In *NetGames*, 2009. 21

[30] B. Lewis and L. Porter. In-game advertising effects: Examining player perceptions of advertising schema congruity in a massively multiplayer online role-playing game. *Journal of Interactive Advertising*, 2010. 22

[31] Peter Danzig, Jeff Mogul, Vern Paxson, and Mike Schwartz. The Internet Traffic Archive, April 2008. http://ita.ee.lbl.gov/. 22

[32] Boxun Zhang, Alexandru Iosup, and Dick Epema. The Peer-to-Peer Trace Archive: Design and Comparative Trace Analysis. Technical Report PDS-2010-003, Delft University of Technology, 2010. 22

[33] J. Yeo, D. Kotz, and T. Henderson. CRAWDAD: a community resource for archiving wireless data at Dartmouth. *ACM SIGCOMM Computer Communication Review*, 36(2):21–22, 2006. 22

[34] J. Leskovec. Stanford Network Analysis Project (SNAP), July 2009. http://snap.stanford.edu. 22

[35] M. Tsvetovat, J. Reminga, and K.M. Carley. DyNetML: Interchange format for rich social network data. Technical Report CMU-ISRI-04-105, Carnegie Mellon University, 2004. 22