Matchmaking in multi-player on-line games: studying user traces to improve the user experience

Maxime Véron, Olivier Marin, Sébastien Monnet
Sorbonne Universités, Université Pierre et Marie Curie (Paris 6), Paris, France
CNRS, UMR_7606, LIP6, Paris, France
Inria, Équipe REGAL, Paris, France
Email: *firstname.lastname*@lip6.fr

ABSTRACT

Designing and implementing a quality matchmaking service for Multiplayer Online Games requires an extensive knowledge of the habits, behaviors and expectations of the players. Gathering and analyzing traces of real games offers insight on these matters, but game server providers are very protective of such data in order to deter possible reuse by the competition and to prevent cheating. We circumvented this issue by gathering public data from a League of Legends server (information over more than 28 million game sessions). In this paper, we present our database which is freely available online, and we detail the analysis and conclusions we draw from this data regarding the expected requirements for the matchmaking service.

1. INTRODUCTION

In the digital gaming industry, software support architectures such as game engines and online services call for constant enhancement to keep up with technological advances and user expectations. In the case of *matchmaking* for Multiplayer Online Games (MOGs), the crucial service that allows players to find opponents, current solutions raise important issues regarding player experience. They often lead to mismatches where strong players face weak ones, a situation which satisfies none of the players involved. Furthermore, response times can be very long: ranging up to hours of waiting until a game session can begin.

However, most game production teams focus on improving graphics and playability to compete with one another. As a consequence, game supporting software such as engines and middleware services consists mainly of legacy components that are widely reused and often quite old. While this allows to cut game production costs and to build upon time-proven solutions, it prevents some much awaited improvements to the overall game experience. For instance most player communities would welcome better schemes towards the elimination of malicious player behaviors and truly efficient matchmaking.

The improvement of game supporting software requires an extensive study of how they behave when used by real players. To do so, it is crucial to obtain and analyze real data. Such data is very hard to get by because game developers want to prevent its reuse by their competition and by potential cheaters.

In this paper, we focus on acquiring real game data. We use data gathered from a popular online game server [1] to show that mismatches and response times are indeed critical issues for matchmaking. Our main contributions are the following.

1. We describe our freely available dataset which covers information about over 28 million game sessions [2].
2. We give a detailed analysis of the acquired data and highlight the main issues raised by one of the game's crucial services: namely its matchmaking.

2. OUR STUDY CASE: MATCHMAKING IN LEAGUE OF LEGENDS

This section describes our study case: the matchmaking service of League of Legends (LoL). LoL is a popular game with a community regrouping over 12 million players worldwide per day in 2012 ; this ensures an abundance of user traces from a genuinely successful online game. It is a competitive game where ranking is a central element; as such, ranking is a precious metric for studying player behaviours and extrapolating quality of service. Finally it is a session-based game with relatively short sessions, averaging around 34 minutes according to the game developers. Its matchmaking service is used often and plays an important role in the overall game experience.

2.1 League of Legends: a brief overview

League of Legends is a multiplayer online battle arena (MOBA) video game developed and published by Riot Games. LoL players are matched in two opposing sides comprising five teammates each. Both teams fight in an arena in order to destroy the enemy team's main building called *nexus*.

The LoL server definition of a *game* is the brawling session that pits players against each other inside the arena. We will therefore use this term to refer to the players' games history.

Three factors are crucial to the gaming experience: waiting times, matching accuracy, and server response times. With an average waiting time of 90 seconds in between sessions, players can spend a lot of time waiting for a session to begin (see Subsection 4.1). This is especially true for very skilled players: their scarcity makes it harder to match 10 such players together. Matching players with disparate skill levels reduces waiting times significantly, however it can reflect poorly on the experience. Unskilled players will feel helpless against much stronger opponents, while the latter will spend time on an unchallenging session with little or no reward to look forward to. Orthogonal to these issues, server response time is crucial in this game which requires extremely sharp reflexes. Lags caused by the servers often increase the ping by up to 300%, which severely impedes the gameplay.

LoL is a competitive game where ranking takes a significant part, both in terms of matchmaking and in terms of player status. Rank defines the skill level of a player, and rank improvement is the main goal of most LoL players. Once players are ranked, they get placed in competition categories called *Leagues*, and subcategorized into *Divisions*. This ranking, as it can evolve quickly over time, needs to be computed precisely and that's why most games use the Elo rating system. The Elo rating system [3] was invented as an improved method for calculating the relative skill levels of chess players. Today many other games, including multiplayer competitions, have adapted it for their own use. Thereafter, we use ranking, Elo, or MMR (short for *MatchMaking Rating*) to talk about player ranking values.

The formulas used in League of Legends for ranking calculations have not been disclosed publicly. However, most ranking implementations share the same bases inherited from the original Elo rating system. Riot Games developers do divulge interesting information about the matchmaking in LoL through its website and dedicated forums , though. Based on this information and on our own data analysis, we inferred the general rules that guide LoL player matching and describe them in the following subsection.

2.2 Matchmaking in League of Legends

League of Legends players can join several types of games, associated with different *queues* on the servers. A group of persons joining the same *queue* in order to play together in the same team is called a *premade*. A *premade* can comprise from one to five players. *Normal* games do not count towards official ranking, whereas *ranked* games do and are only open to seasoned players (above level 30). Our study focuses mainly on *ranked* games since they draw together players whose statistics are more likely to be representative of the game's core community. Also, fair matchmaking is more critical for *ranked* games since they induce real stakes for the players.

According to Riot Games developers, matchmaking in LoL is based on: player ranking, the experience of each player (number of games played), and the size of the premade.

Player ranking weighs most since it has the highest impact on the outcome of the game. LoL tries to match teams as fairly as possible: it computes the average of the player ranking values for each team, and then uses these averages to match teams similarly to one-versus-one fights. This solution speeds up the matching of multiple players, yet it may result in an unbalanced game if two players with very different rankings join in a premade since the least skilled player is likely to face a highly ranked opponent. In a competitive context, this kind of quick fix is very unsatisfactory.

Adding the personal number of played games as a matching criterion alleviates this issue, but it also increases the average waiting time drastically (roughly around 50%). This shows the direct impact on the average waiting time of adding a single preference to the matching system.

Premades also increase the waiting time, and can even induce unbalanced battles. If the matching system cannot find an opposing team with a premade of the same size and level in the allotted time (roughly 30 seconds for a normal game), it gathers single players with higher individual ranks to build an opponent team. The higher rank is supposed to compensate for lesser coordination in the opposing team, but usually leads to a victory of the higher ranked players.

3. GATHERING DATA ABOUT LEAGUE OF LEGENDS

In order to acquire data for our study, we gathered public information from a League of Legends server. The resulting data set covers information about more than 28 million game sessions obtained over a month of crawling, and is freely available as a database [2]. This section describes the nature of the data we acquired and our retrieval method.

3.1 The nature of the retrieved data

We distinguish three categories of data among the dataset we acquired:

The first category, *avatar information*, regroups data that characterizes a player's status inside the game. This category is common to many games such as World of Warcraft or Diablo 3, where the main purpose is avatar evolution and competitiveness. For example, the items that an avatar possesses or the damages dealt to others are fields that can be found in every role playing game and in every first-person shooter. This category is by far the largest, accounting for 43 data fields out of 77.

The second category, *company handlers*, gathers content that is specific to League of Legends. The company handlers are variables that exist only to help the company maintain the game, such as player identification numbers and timestamps. This 13 data fields of this category helped sorting our players in the database. Although it represents data specific to League of Legends, some of it allows analysis for business related concerns. Such is the case with data about *skins* that players apply to their avatars. *Skins* are add-ons that players buy in exchange for real money in LoL. Relating the *summoner* level of the players in the game with the first time they played with a skin provides insight about the moment players tend to buy their first skins, and which kind of skin they first buy.

The third and last category encompasses data related to the *matchmaking of the game*. It intersects with the previous categories: it includes avatar information and precise network latency monitoring. The *userServerPing* reveals the average latency incurred by a player during the game. The *timeInQueue* data field is also crucial for our analysis as it represents the waiting time before a player gets matched with other players. We also use some avatar data in our LoL matchmaking analysis, such as the number of kills dealt by the avatar (*kill*), and the number of times it died (*num_ death*), in order to detect imbalanced games.

3.2 Services used to retrieve data

To retrieve LoL information we used a free open source code [4] to pack/unpack the contents of the messages sent to the game server.

LoL servers follow a *function call* paradigm to handle requests. A message must be constructed in such a way that it identifies the right *function* from the right *service* and contains the function parameters.

An example of a typical request is : <"summonerService" "getSummonerByName" "darkKnight67">.

Our first task was to examine all LoL services (there are over a hundred) to identify the ones associated with game statistics. Among the latter, we identified two specific services: the first handles the recent games history of players, and the second provides statistics about any given game once its identifier is known.

More specifically in terms of LoL server requests, we used the *getSummonerByName* function of the *summonerService* to translate *accountId* values into what RiotGames stores internally as *summonerId* values. Once this *summonerId* is obtained we can then call *getRecentGames* on the *playerStatsService* to get an array of *games* from the LoL server.

Each player history contains 10 games or less, depending on the number of games played during the last seven days.

LoL servers process game data as a hashMap of (key, value) couples, but send it through the network as raw content. A big part of our work was to redefine a proper Java class describing game statistics, with which we could grab all the fields sent by the server when requesting for games histories. Once this was achieved we fed all possible *summonerId* values to the server to obtain games histories data, and stored it into a database for long-term usage. This allowed efficient searches and statistical computations for our analysis presented in section 4.

After thorough verification, we discovered that the LoL servers we queried fail to fill in some data fields and hand out invalid NULL values instead. Unfortunately, such is the case for data that we deemed really valuable for our analysis: in particular the predicted win percent, and the *KCoefficient* that is used to compute the Elo gain.

4. ANALYSIS FOR MATCHMAKING SYSTEMS

This section presents our analysis of the data concerning the LoL matchmaking service. We evaluate the impact of this service on the game experience in terms of waiting time, matching precision, and server response time.

4.1 Influence of ranking on waiting times

Our first assessment concerns the distribution of LoL players in terms of ranking, displayed as the solid line in Figure 1. Besides allowing correlations between player skills and quality of service, such information is very valuable for designing and evaluating new matchmaking solutions. The initial ranking value for LoL beginners is 1200, which explains why there is a large majority of players around and just above this value. As expected the number of players diminishes as the ranking increases, since it requires increasing skills to attain higher rankings. An obvious consequence of this distribution is that matching highly ranked players together is harder, and thus should take more time.

Figure 1 illustrates this issue through the dashed line which represents the average waiting time with respect to player ranking. Above rank 1700 the average waiting time increases exponentially. It actually reaches up to 45 minutes for the highest ranks, but we did not include this in the graph to preserve its readability, as most waiting times are below 100 seconds. In the LoL dedicated forums, Riot Games developers state that they aim for an average waiting time that will not exceed 30 seconds. Our observations show that the slowdown incurred by the vast majority of players brings the overall average waiting time closer to 90 seconds. One might consider 90 seconds an acceptable duration; however further calculations show that, out of 825000 ranked game requests, 65259 took more than 5 minutes to get matched, affecting all player ranks alike. This amounts to 7.9% of matches that fail to start within an acceptable time-frame.

We draw an important conclusion from these observations: the current LoL matchmaking system does not scale well. Both curves top out simultaneously around the Elo value of 1200. In our opinion, this demonstrates the bottleneck effect of the matchmaking service trying to deal with huge numbers of similarly skilled players. Considering our information covers more than a month of crawling, this conclusion is unlikely to result from temporary server issues.

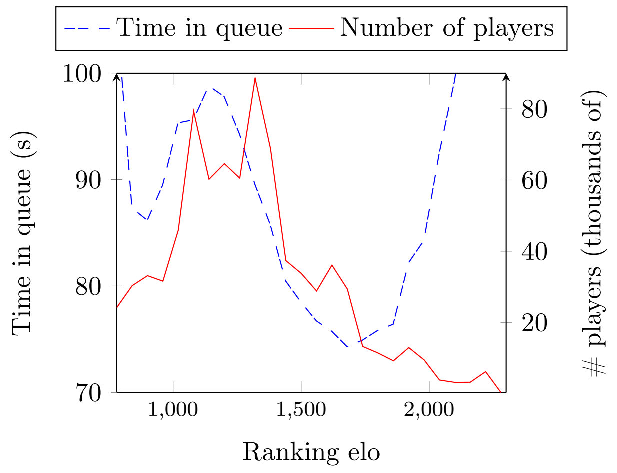

Figure 1: Distribution of the players ranks their waiting time to get matched

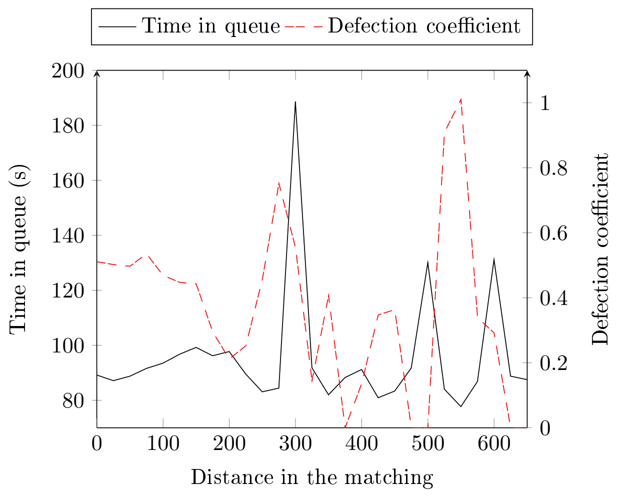

Figure 2: Matching distance and frequency of game defections

4.2 Impact of the matching distance on the game experience

Our next assessment studies the impact of the matching distance on the gaming experience. Getting matched against opponents with extremely different skill levels usually results in a boring session: an outclassed player will experience a half-hour of severe pounding from the opponents, while a very superior player will have a very unchallenging session. This sometimes results in a player simply quitting the game. It is important to keep in mind, however, that defections are rare in LoL as they often lead to banishment from the game.

For the purpose of our study, we first computed a coefficient associated with the frequency of game defections; a value of 1 corresponds to the highest number of defections encountered for any given matching distance. Figure 2 correlates these results with the average waiting time. Both curves are inversely proportional, which means that a quick matching resulting in a significant ranking difference often leads to a defection. The two defection peaks observed before 300 and before 600 correspond to the ranking distances between different *Ranked Leagues* in LoL. In other words, players pitted against opponents in a different category of competition are more likely to defect. This is a strong argument **against** opting for a bad match in order to speed up the process.

4.3 Impact of latency on the game experience

Even though Sheldon et al. [5] show that latency bears little importance in real time strategy games such as Warcraft 3, such might not be the case for MOBAs. Hence we evaluated the impact of latency on LoL gameplay.

To begin with, we studied the distribution of the latencies within the player population; Figure 3 presents our results. It shows that the ping remains between 30ms and 50ms for a vast majority of users. Since this is a very low value, our first intuition was that the results from [5] stand true for LoL. However on closer inspection, we observed that pings above 100ms were found in more than 1.2 million games, representing 7% of the total.

Ranking is not a good metric for evaluating the effect of latency on the gameplay, as lags will impede poorly skilled and highly skilled players in similar ways. Besides it affects single players instead of teams. Therefore, we used another metric associated with player performance: the *killed death assist ratio*, or *KDA*. It is possible to die, kill or assist in killing enemy avatars multiple times in a LoL session. Hence, the ratio between killing/assisting and dying gives a good insight on the in-session performance of a player. The formula used to compute the KDA is the following :

$$kda = \frac{assists+kills}{max(deaths,1)}$$

A KDA inferior to 1 reflects poorly on a player's general skills.

The dotted curve in Figure 4 illustrates the relation between the KDA and the ping. Since the KDA de-

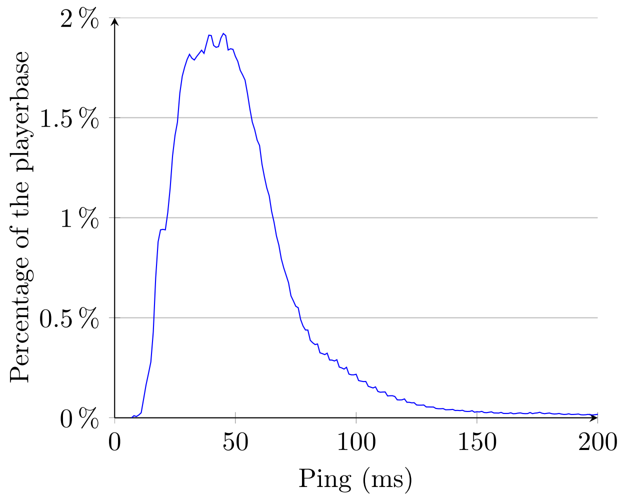

Figure 3: Distribution of the players' ping in League of Legends

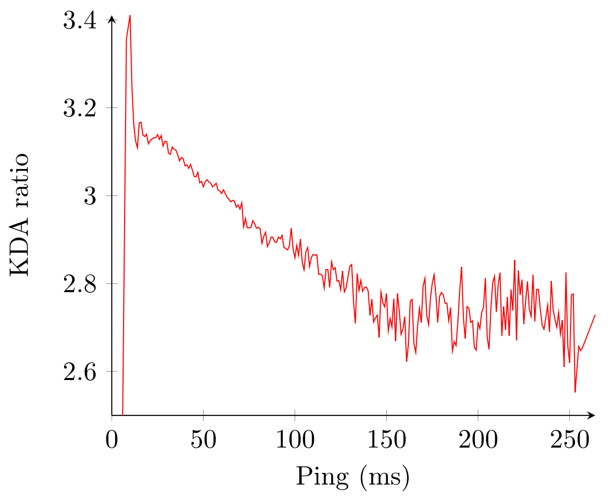

Figure 4: Impact of the latency on the player performance

creases steadily as the ping increases, ultimately dividing the maximum KDA by a factor of 8, we conclude that latency does indeed impede on playability.

To better understand the problem of latency, we also found that among players that leave games, 16% of them have a latency superior to 100ms. Considering that the amount of players, leavers or not, with a latency superior to 100ms is of 7%, we can correlate undergoing high latency and leaving games prematurely, as shown also in [6]. MOBAs games are close to FPS games in this characteristic that latency impacts strongly the gameplay.

Taking this into consideration, our conclusion is that matchmaking ought to include ping values as a criterion in order to improve the overall game experience for MOBAs. However, LoL's current matchmaking service fails to answer on time for the highest majority of ranked players without any extra criteria.

Our conclusion with respect to the experiments described in this section is that a new matchmaking architecture would lead to much needed improvements: reduced waiting times, better gaming experience, and extended matching criteria in order to satisfy players further.

5. RELATED WORK

The literature on distributed gaming is abundant. Yet we have found no other paper about gathering and analyzing such an amount of real user data to study matchmaking.

Most papers on distributed game traces focus either on crawling methods [7, 8] or on very specific aspects of games like positions and movements of player avatars over time [9, 10, 11]. Although our primary goal was to acquire and analyze data related to matchmaking, our traces can serve many other purposes.

Some studies propose approaches to improve gaming experience. Chen et al. [6] show that latency impacts the length of gaming sessions. The results of this study are consistent with our database analysis where we observe that increasing amounts of players leave the game early as latency deteriorates. As a consequence, several works propose to take latency into account in the matchmaking mechanism. Agarwal and Lorch propose a latency-aware matchmaking solution [12]; Zander et al. [13] also suggest to improve the matchmaking system by adding more properties (eg. latency). In [14], the authors propose an improved matchmaking, there again by adding more information to feed the matchmaking mechanism.

However, our study shows that the matchmaking system in a popular game like League of Legends fails to reply on time with few criteria. Our intuition is that operating multi-criteria matchmaking systems on a large scale requires a fully decentralized architecture.

Other papers, such as [15, 16, 17], focus on designing distributed matchmaking approaches. Yet they lack real game data/application traces to validate their approaches. These solutions could take advantage of our publicly available data.

6. CONCLUSION

User traces enable to improve game software through careful study and analysis. However, game companies seldom make such data available. In this paper, we show how to gather a huge quantity of publicly available information about players of a very popular online game. We also analyze this information to draw conclusions about how to improve a critical service for multiplayer online games, namely matchmaking.

As a consequence of the work presented in this paper, we are currently working on a fully distributed, peer to peer matchmaking system. The database we acquired constitutes a precious tool for testing our system with real application data. We aim to offer better precision and response times than current matchmaking service architectures. Our approach aims to make matchmaking more accurate and more effective, all the while allowing a wider range of matching criteria.

We strongly believe that our database can help design future software solutions for gaming. A considerable spectrum of studies could use this information, and we think that gathering this type of information automatically for recent games could greatly improve the future of gaming solutions.

7. REFERENCES

[1] "League of legends." [Online]. Available: http://euw.leagueoflegends.com/

[2] "League of legends game traces database." [Online]. Available: http://pagesperso-systeme.lip6.fr/maxime.veron/examples.html

[3] A. E. Elo, *The rating of chessplayers, past and present.* New York: Arco Pub., 1978.

[4] "lolrtmpsclient - a lightweight rtmp client in java for communicating with league of legends." [Online]. Available: https://code.google.com/p/lolrtmpsclient/

[5] N. Sheldon, E. Girard, S. Borg, M. Claypool, and E. Agu, "The effect of latency on user performance in warcraft iii," in *Proceedings of the 2nd workshop on Network and system support for games*, ser. NetGames '03. New York, NY, USA: ACM, 2003, pp. 3–14.

[6] K.-T. Chen, P. Huang, and C.-L. Lei, "How sensitive are online gamers to network quality?" *Commun. ACM*, vol. 49, no. 11, pp. 34–38, Nov. 2006.

[7] Y. Guo and A. Iosup, "The game trace archive," in *Proceedings of the 11th Annual Workshop on Network and Systems Support for Games*, ser. NetGames '12. Piscataway, NJ, USA: IEEE Press, 2012, pp. 4:1–4:6.

[8] B. Zhang, A. Iosup, P. Garbacki, and J. Pouwelse, "A unified format for traces of peer-to-peer systems," in *Proceedings of the 1st ACM workshop on Large-Scale system and application performance*, ser. LSAP '09. New York, NY, USA: ACM, 2009, pp. 27–34.

[9] S. A. Tan, W. Lau, and A. Loh, "Networked game mobility model for first-person-shooter games," in *Proceedings of 4th ACM SIGCOMM workshop on Network and system support for games*, ser. NetGames '05. New York, NY, USA: ACM, 2005, pp. 1–9.

[10] G. Armitage, "An experimental estimation of latency sensitivity in multiplayer quake 3," in *Networks, 2003. ICON2003. The 11th IEEE International Conference on*, 2003, pp. 137–141.

[11] J. Kinicki and M. Claypool, "Traffic analysis of avatars in second life," in *Proceedings of the 18th International Workshop on Network and Operating Systems Support for Digital Audio and Video*, ser. NOSSDAV '08. New York, NY, USA: ACM, 2008, pp. 69–74.

[12] S. Agarwal and J. R. Lorch, "Matchmaking for online games and other latency-sensitive p2p systems," in *Proceedings of the ACM SIGCOMM 2009 Conference on Data Communication*, ser. SIGCOMM '09. New York, NY, USA: ACM, 2009, pp. 315–326.

[13] S. Zander, I. Leeder, and G. Armitage, "Achieving fairness in multiplayer network games through automated latency balancing," in *Proceedings of the 2005 ACM SIGCHI International Conference on Advances in Computer Entertainment Technology*, ser. ACE '05. New York, NY, USA: ACM, 2005, pp. 117–124.

[14] O. Delalleau, E. Contal, E. Thibodeau-Laufer, R. Ferrari, Y. Bengio, and F. Zhang, "Beyond skill rating: Advanced matchmaking in ghost recon online," *Computational Intelligence and AI in Games, IEEE Transactions on*, vol. 4, no. 3, pp. 167–177, 2012.

[15] V. Shafran, G. Kaminka, S. Kraus, and C. V. Goldman, "Towards bidirectional distributed matchmaking," in *Proceedings of the 7th int. joint conference on Autonomous agents and multiagent systems*, ser. AAMAS'08, 2008, pp. 1437–1440.

[16] E. Ogston and S. Vassiliadis, "Local distributed agent matchmaking," in *In Proc. of the 9th International Conference on Cooperative Information Systems*, 2001, pp. 67–79.

[17] J. Manweiler, S. Agarwal, M. Zhang, R. Roy Choudhury, and P. Bahl, "Switchboard: A matchmaking system for multiplayer mobile games," in *Proceedings of the 9th International Conference on Mobile Systems, Applications, and Services*, ser. MobiSys '11. New York, NY, USA: ACM, 2011, pp. 71–84.