Load Dynamics of a Multiplayer Online Battle Arena and Simulative Assessment of Edge Server Placements

Valentin Burger[†], Jane Frances Pajo[‡], Odnan Ref Sanchez[‡], Michael Seufert[†], Christian Schwartz[†], Florian Wamser[†], Franco Davoli[‡], Phuoc Tran-Gia[†]

[†]University of Würzburg, Institute of Computer Science, Würzburg, Germany
{burger | seufert | christian.schwartz | wamser | trangia}@informatik.uni-wuerzburg.de

[‡]University of Genoa, Telecommunication Networks and Telematics Lab, Genoa, Italy
{jane.pajo | odnan.sanchez}@tnt-lab.unige.it, franco.davoli@unige.it

ABSTRACT

Free-to-play models, streaming of games and eSports are reasons for online gaming to grow in popularity recently. On the forefront are multiplayer online battle arenas, which gain high popularity by introducing a competitive format that is easy to access and requires cooperation and team play. These games highly rely on fast reaction of the players, which makes latency the key performance indicator of such applications. To obtain low latency, this paper proposes moving game servers close to players towards the edge of the network. The performance of such mechanism highly depends on the geographic distribution of players. By analyzing match histories and statistics, we develop models for the arrival process and location of game requests. This allows us to evaluate the performance of edge server resource migration policies in an event based simulation. Our results show that a high number of edge servers is preferable compared to few larger edge servers to reduce the latency of players. This supports approaches that allow deploying virtual server instances in the back-haul.

CCS Concepts

•**Networks** → **Network performance modeling; Network simulations; Cloud computing;** *Location based services;*

Keywords

Online Gaming; Load Dynamics; Cloud Computing; Resource Allocation

1. INTRODUCTION

Online gaming is becoming increasingly popular and accounts for a growing proportion of today's Internet traffic. It has an expected compound annual growth rate of more than 40% from 2014 to 2019 [6]. As more and more traditional games are provided as cloud games, gaming could become one of the largest Internet traffic categories in the next years [7].

Currently, especially multiplayer online games such as Massively Multiplayer Online Games (MMOGs) or Multiplayer Online Battle Arena (MOBA) games attract a large number of players. In the beginning of 2014, Riot Game's League of Legends[1] accounted for 67 million monthly players with up to 7.5 million playing at the same time [26]. Valve's Dota 2[2] is currently responsible for more than half a million concurrent users on average a month with a peak of 1.2 million concurrent players in February 2015 [29].

To meet the huge demand, game publishers, like Riot Games and Valve, allocate a considerable amount of resources for hosting these games. They are commonly hosted in large data centers and are accessed with client software that renders the video games locally. User interactions are uploaded to the game server that maintains the global match states and distributes pending rendering instructions to the clients. Those interactive online games are largely affected by network delay, jitter, and packet loss (e.g., [2, 33]), which considerably influence the game play and the users' Quality of Experience (QoE) [13, 18]. Thus, together with the emerging trends of service virtualization, network virtualization, and edge computing, there are intentions to bring virtual game servers closer to the users (e.g., [5]). Thereby, multiple, typically smaller dimensioned, game servers can be distributed to servers or small data centers at the edge of the network near the end users. Hosting game servers at the network edge is expected to decrease latency and loss and consequently increase the gaming QoE.

In this work, we look in detail at the possibilities for hosting the popular MOBA games at the network edge. First, we investigate single gameplay statistics of Dota 2 in order to develop realistic user and traffic models. These models serve as input for a comprehensive simulative evaluation of edge-supported online gaming. In particular, we look at placement strategies for virtual game servers and evaluate their impact on network traffic and game performance.

The contribution of the work comprises the following items:

- Common traffic characteristics of the Dota 2 MOBA online game are presented, such as access patterns or

[1]http://leagueoflegends.com
[2]http://dota2.com

MMSys'16, May 10-13, 2016, Klagenfurt, Austria

DOI: http://dx.doi.org/10.1145/2910017.2910601

match duration distribution.

- A system model is specified, which can be used as a basis for simulative assessment of game server placement and migration at the network edge.
- Based on the traffic characteristics and simulation, first results on the impact of edge placement of game servers are derived.

The remainder of this paper is structured as follows. Section 2 introduces background and related work on MOBA, models for gaming, and placement strategies. Section 3 discusses the general system model, components, and investigated strategies. The used models and underlying data are presented in Section 4, and details of the simulation are described in Section 5. The obtained results are shown in Section 6, and Section 7 concludes.

2. BACKGROUND AND RELATED WORK

This section provides background information about the considered game type and presents relevant literature related to real-time strategy games.

2.1 Game Concept of Multiplayer Online Battle Arena Games (MOBA)

This paper considers the *Multiplayer Online Battle Arena (MOBA)* computer game genre. Examples of this genre include Dota 2, considered as the main example in this paper, as well as League of Legends and Heroes of the Storm. The MOBA genre has an estimated monthly player count of more than 80 million [3], highlighting the impact of consumed network and compute resources. MOBA games are are historically derived from the Real Time Strategy (RTS) genre and share the same hardware and network demands. In a MOBA game, typically two teams of 5 players compete on a game *map* with the ultimate goal of destroying the enemy base.

A MOBA requires teams to cooperate with each other and exploit the capabilities of their game character chosen from a fixed set, to defeat the opponent. Team work and strategy are key to winning, highlighting the importance of fast reaction times and the corrosponding network requirements.

2.2 Existing Literature on Online Gaming and Related Fields

Online gaming has been an active research area in the past decade, and several studies have been conducted. We summarize in the following findings according to different directions such as statistics about particular games or findings about migration in cloud computing in general.

A study on the player performance for the First Person Shooter (FPS) game Unreal Tournament 2003 (UT2003) is conducted in [2] under varying amounts of packet loss and latency. The authors state that UT2003, and possibly FPS games in general, can tolerate a modest amount of packet loss, but shooting is greatly affected by latency. In addition to these two QoS metrics, Wattimena, et al. [32] examined the impact of jitter on the gaming experience for the FPS game Quake IV via subjective and objective measures. The authors reached similar conclusions as [2], and noted that the introduction of jitter has a large negative effect on the gaming QoE. Other studies [8, 25, 19] have also investigated other classes of online games. The characteristics of a mobile MMORPG are studied in [22] by a large-scale and long-term measurement, investigating its population, playersâĂŹ game usage behavior, the players' interest and money spent in game. In the field of traffic modeling, there is especially work on FPSs [9, 4] and MMORPGs [14, 30]. In [24] the virtual populations of two MMORPGs World of Warcraft (WoW) and Warhammer Online are characterized.

Furthermore, research is conducted in [30] for World of Warcraft. The session times of MMOGs are typically higher than those of round based action games, like FPSs. The server to client session times of MMORPG can be fitted with a Weibull distribution.

Multiple studies have been conducted in the field of virtualization in cloud environments in general. A particular research area covers the placement of virtual machines inside the cloud infrastructure that is the basis for migrations [23, 17, 11, 27].

The concept of migrating an ongoing game emerged over a decade ago. A paper [10], published in 2004, presents a migration algorithm that "can be adopted on a generic multiplayer, multi-server online gaming architecture", which allows a player to choose a better server and migrate the game state in the middle of a game. Moreover, the authors indicate that a good migration heuristic considers at least two factors: the network latency and server load. The latter mainly accounts for the possibility that many players may choose a well-connected server, which may lead to the saturation of the server capacity. Considering the world-spanning MOBAs, Beskow, et al. [3] used core selection to find an optimal node in the system for placing a virtual region, and correspondingly the players interacting in that region. Conversely, Jalaparti [12] considered FPS games and proposed a platform for Seamless Migration of Online Games (SMOG), which determines not just where, but also when, to move game servers. For the server placement problem, SMOG focused on minimizing the number of players in a game server whose latency is greater than a certain threshold, initiating a migration when the new optimal location results to a gain better than a pre-defined threshold.

3. SYSTEM MODEL

To define the system model, we describe the machine environment and the job characteristics.

As machine environment we consider a set of server re-

Table 1: Notation of the paper.

Parameter	Description	Default
C_{DS}	dedicated server capacity	3000
C_{ES}	edge server capacity	1000
λ	arrival rate of requests	
k	number of players per match	10
μ	match service rate	
ρ	throughput of edge link	
σ	memory footprint	
d_{rnd}	distance from city center	5 km
d_{party}	distance from party leader	100 km

[3]http://venturebeat.com/2015/07/15/comparing-mobas-league-of-legends-vs-dota-2-vs-smite-vs-heroes-of-the-storm/, accessed: November, 27^{th}2016

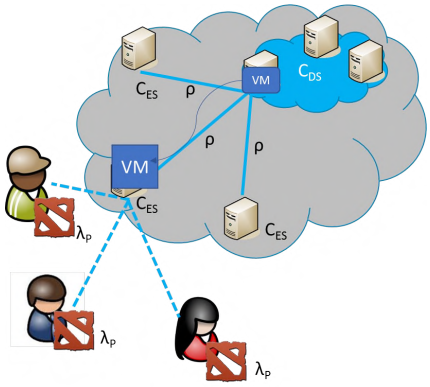


Figure 1: System model.

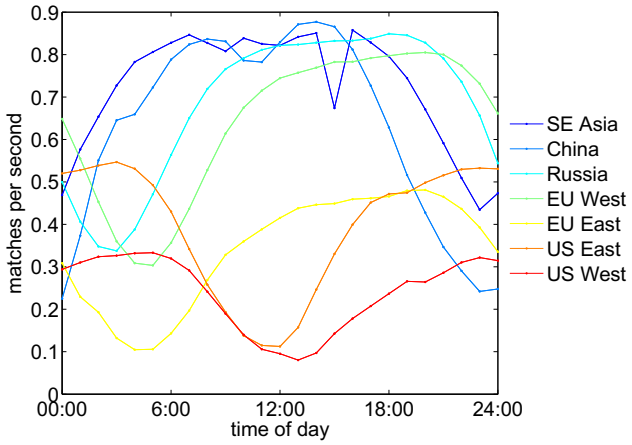


Figure 2: Arrival rate of game requests depending on the time of day.

sources V. For each server resource there is a positive integer size that specifies the capacity to host games. Here we distinguish between dedicated servers and edge servers, with capacities C_{DS} and C_{ES} respectively. Capacity is determined by resources of server as ram, cpu, storage and throughput. The total capacity of a server is divided in fractions where a capacity fraction is occupied for each match hosted on the server. Each resource in V has coordinates describing its location. The resources are organized in a graph $G = (V, E)$ with a set of links E. Each link $(u, v) \in E$ connects two resources $u, v \in V$. Each link has a throughput ρ.

The job characteristics are defined by the matches played and hosted on the server resources. Users generate demands for the resources if they want to play a game. The game requests are generated with rate $\lambda(t) = \frac{1}{\beta}(t)$, where $\beta(t)$ is the average inter-arrival time of game requests at time t. For each request i the location ξ_i of the request is determined randomly. A match is made out of k game requests within a certain region.

We distinguish between party game requests with rate λ_p and single player game requests with rate λ_s. For each match a job is generated that occupies a fraction of server resources for a certain game duration that is $\frac{1}{\mu}$ on average. The jobs generated depend on the match making of the game, which depends on the player experience and other factors. For the sake of simplicity, we do not consider player experience.

The migration policy determines which match is hosted on which server. If M_t is the set of active matches at a given time t, then it can be considered as a function that maps each match $m \in M_t$ to a server resource $v \in V$.

$$f : M_t \mapsto V \tag{1}$$

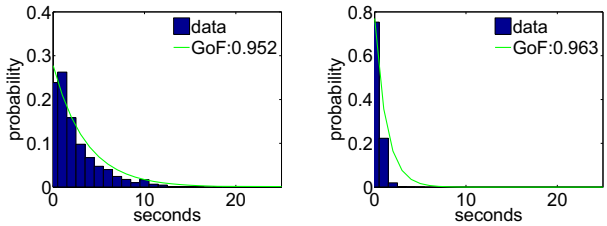
A migration policy is feasible with respect to the resource constraints if at any time t the index set M_t, of jobs being executed at t satisfies $\sum_{j \in M_t} r_{hj} \leq C_h (h = 1, \ldots, l)$, where C_h is the capacity of each resource $v_h \in V$.

4. MOBA LOAD CHARACTERISTICS

To realistically model the load on the MOBA game servers, it is necessary to understand when and where new matches are created and how long a match strains the server. In order to obtain these characteristics for Dota 2, the match histories in the Steam database were crawled, and the Steam Web API [28] was used to query the game database. This API allows licensed individuals to retrieve data available via Steam by using their respective API keys. The data returned by successful API calls consist of Dota 2 match histories, particularly, the game start time and date, game duration, and the server location. A total of 8,470,933 Dota 2 public matches and 1,786,148 unique public-profiled players' data have been crawled. These public matches are worth of exactly one week (from March 18 to March 25, 2015) with more than 1 million games per day and a peak of almost 1.4 million matches during the weekend.

4.1 Game Requests

Figure 2 depicts the arrival process on Saturday, March 21, 2015 for the different dedicated server regions. The x-axis depicts the time of day in Central European Time (CET, UTC+01:00), and the y-axis shows the number of match requests per second. The arrival process follows typical dynamics according to day and night time in each region, which result in periods of high arrival rates during the night times of the respective region. Moreover, it can be observed that the peak rates differ. In certain regions, such as SE Asia or China, peak loads are higher. The least load is on EU East, US East, and US West.



(a) non-busy hour (04:00) (b) busy hour (18:00)

Figure 3: EU West server inter-arrival times

To model the inter-arrival time of match requests, we use discrete time steps of one second. For each hour, we compute the empirical probability distributions of the inter-arrival time. We approximate the empirical distributions with an

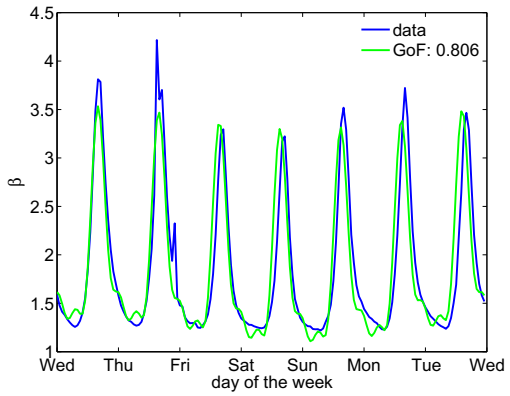


Figure 4: β dynamics in EU West server.

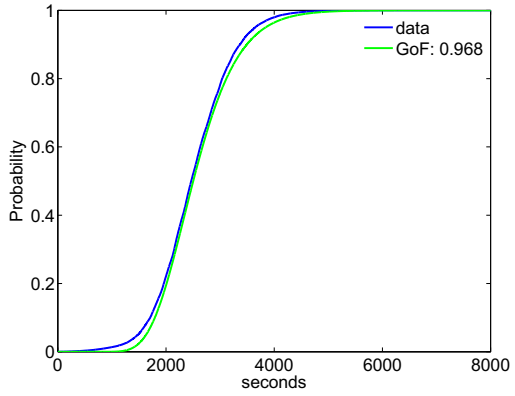


Figure 5: CDF of match durations in Valve's EU West server.

exponential distribution

$$f(x,\beta) = \frac{1}{\beta}\exp(\frac{-x}{\beta}) \quad (2)$$

by fitting the mean parameter β in a least-squares sense. Figure 3 shows the probability distribution of the inter-arrival time for the EU West server at a non-busy hour (04:00) and a busy hour (18:00). The non-busy hour distribution can be well approximated by an exponential distribution with $\beta = 3.60$, and the busy hour shows a smaller mean inter-arrival time $\beta = 1.30$. The goodness of fit (GoF) is determined by using the coefficient of determination R^2, which gives the amount of variation of the dependent variable that can be predicted by the independent variable. The average goodnesses of fit $\overline{R^2}$ over all 168 hours of the week is larger than 0.91 for all server regions. Although slightly better approximations can be reached with Weibull ($\overline{R^2} > 0.95$) or log-normal ($\overline{R^2} > 0.97$), we restrict ourselves to the exponential fitting to use a simple Poisson process in the simulation.

Looking at the whole crawled period of one week, the server load shows a daily repetition of the day patterns. This pattern can also be observed for the parameters of the fitted distributions over the course of the week. Figure 4 illustrates the mean parameter β of the exponential fitting of the EU West server region as a blue line. Due to its periodic characteristic this function can be well decomposed by Fourier analysis (DFT). The green line depicts an approximation by the five most significant Fourier terms (sines), which captures the daily periodic patterns and also the transition of increasing rates and smaller means from the weekdays to the week-end, and shows a high similarity. For all server regions, the approximation with the five most significant Fourier terms resulted in $R^2 > 0.74$, which can be increased by adding more Fourier terms.

4.2 Match Duration

Fig. 5 shows the cumulative distribution function of the match durations of the EU West region server ($1,368,703$ regular matches from March 18 to March 25). A Dota 2 match has an average match duration of 2590 seconds (ca. 43 minutes) and a standard deviation of 685 seconds. The distribution of the match duration can be approximated using a lognormal distribution

$$f(x,\mu,\sigma) = \frac{1}{x\sigma\sqrt{2\pi}}\exp(-\frac{(\log(x)-\mu)^2}{2\sigma^2}) \; . \quad (3)$$

The lognormal distribution's location parameter μ and the scale parameter σ can be computed from the empirical mean and standard deviation. The fitted lognormal distribution is a good approximation for all region servers and reaches a high $R^2 > 0.94$ for each server.

Having analyzed the game request arrival process and the duration of each match, the server load can then be generated in simulations according to the fitted distributions.

Table 2: Top 5 countries in the EU West server.

Country	Players	Probability
Russia	115210	0.355
Ukraine	39605	0.122
Great Britain	15078	0.046
Germany	12565	0.039
Belarus	12322	0.038

4.3 Player Locations

In order to determine where game servers need to be hosted, it is necessary to know how players are distributed over the Internet. For that purpose, we consider the player counts per country from public Steam profiles to estimate the country probabilities in a Dota 2 server. In total, there were 757,172 public-profiled accounts with a unique player ID that had set their locations. 324,511 of these played on the EU West server. Table 2 lists the top 5 countries in the EU West server by player count and the resulting empirical probabilities. It can be seen that this server is not only frequented by players from the populous western European countries (Great Britain, Germany), but also by eastern European (Ukraine, Belarus) and Russian players, although there are separate EU East and Russia region servers.

Additionally, European cities are extracted from Maxmind's data on countries by continent [16] and world cities, which include country, population, latitude and longitude details [15], to estimate the city probabilities, as well as the player coordinates. Based on the population distribution of the cities per country, player locations, which obey the crawled distribution of players per country, can be gener-

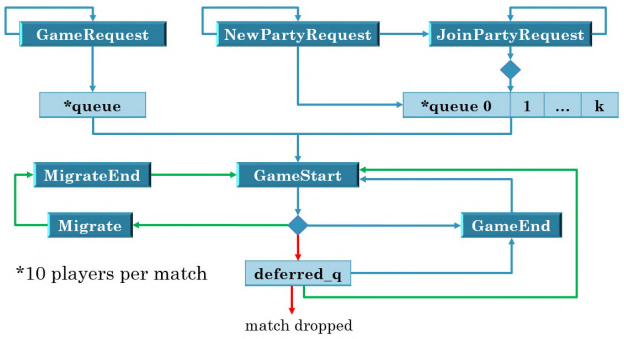


Figure 6: Simulation framework overview.

ated on a per-city granularity by Bayesian inference. The details will be described in Section 5.1 below.

5. SIMULATION DESCRIPTION

In order to evaluate the proposed migration of Dota 2 matches towards the edge, an event-based simulation framework was implemented in Java using the JSimLib [20] Discrete-Event Simulation (DES) library. The emphasis is on how the game server placement impacts on the gaming QoE, hence a detailed simulation on flow or packet level is not necessary. In this respect, the framework is developed at application level, where capacities and loads are defined in terms of the number of games. Moreover, networks between each player and the servers are not considered in the simulation.

We consider Valve's EU West dedicated server (DS) located in Luxembourg, LU [31]. Instead of limiting the virtual game servers to the DS, edge servers (ESs) are also considered to potentially host games through migration. Nevertheless, a match is always set up in the DS (i.e. the server chosen by the players) in order to take into account other server selection criteria (e.g., social, popularity, skill level, etc. [12]). Schemes for geographically distributing user bases and ESs were implemented in the framework. Furthermore, the framework implements a naïve migration model in which migration times are obtained by simply dividing the VM memory size by the available link bandwidth. For the sake of simplicity, the star topology is considered, assuming a direct link between the DS and each ES.

An overview of the framework, featuring the events, queues and their relationships, is shown in Fig. 6. There are seven events defined in the simulation framework — namely, *GameRequest*, *NewPartyRequest*, *JoinPartyRequest*, *GameStart*, *Migrate*, *MigrateEnd*, and *GameEnd*. The *GameRequest*, *NewPartyRequest* and *JoinPartyRequest* events correspond to the three sources of game requests. One queue is dedicated for all random players generated by the *GameRequest* event, while a new queue is created for each party started by the *NewPartyRequest* event; players can join a specific party via the *JoinPartyRequest* event. Once a queue contains $k = 10$ players (i.e. a typical Dota 2 match), the *GameStart* event is triggered. In this event, a match can either be: (1) migrated; (2) started; or, (3) deferred/dropped, depending on the mean distances to servers and server loads.

Case (1): An ES closer to the players is available, to which the game server can be migrated. Hence, two resources are allocated for the match — one in the DS and another in the optimum/suboptimal ES. The *Migrate* event is triggered to start the game server migration from the DS to the specified ES. In this event, the *MigrateEnd* event is scheduled at a time given by the migration duration, which updates every time a flow enters/leaves the link. When the migration is complete, the *GameStart* event is again triggered, in which the resource in the DS is freed and the *GameEnd* event is scheduled at a time given by the match duration. When the match has ended, the resource in the ES is also freed.

Case (2): The DS is either the optimum server for the match or all closer ESs currently have no resources available. Hence, a resource is allocated in the DS for hosting the match. The *GameEnd* event is scheduled at a time given by the match duration. When the match has ended, the resource in the DS is freed.

Case (3): The DS is fully loaded. Up to 100 new matches can be placed in the *deferred_q* to wait for an available resource in the DS, after which, succeeding matches will be dropped.

Note that every time a resource in the DS is freed, the system checks if there are matches waiting in the *deferred_q*. If so, a new *GameStart* event is triggered for the first match in the queue, and the process repeats.

5.1 Player Location Model

Based on the player count per country, the empirical probability f_x that a player originates from country x is

$$f_x = \frac{n_x}{\sum_j n_j} \ , \tag{4}$$

where n_x is the number of players from country x determined from the player locations in Section 4.3.

Given that a player originates from country x with population $\mathbb{P}_x = \sum_y \mathbb{P}_x^y$, the empirical probability f_x^y that the player resides in city y with population $\mathbb{P}_x^y$ is

$$f_x^y = \frac{n_x^y}{\mathbb{P}_x} = \frac{n_x^y}{\sum_y \mathbb{P}_x^y} \ . \tag{5}$$

We approximate the probability f^y that a player resides in city y according to the Bayes theorem, by

$$f^y \approx f_x \cdot f_x^y \ . \tag{6}$$

Based on this, the locations of players in a match are generated according to two schemes: *(a) random* and *(b) party*, in both of which, player coordinates are computed, given an initial point, angle and geographical distance.

Random: This scheme refers to solo queuing, where a single player looks for other random players. As the name implies, this scheme generates player locations randomly, according to the country and city probabilities. The location of player i is then calculated by determining a city y according to probability f^y. Given the latitude and the longitude of the center of city y, the exact coordinates ξ_i of player i are determined by adding an exponentially distributed distance with parameter d_{rnd} in a uniformly distributed angle.

Party: With this option, players in a match may tend to be more geographically concentrated. For instance, when friends play together in one location. This assumption relies on the fact that according to [1] the probability of friendship decreases exponentially with distance. In this scheme, the location of the first player i is generated randomly as in the *random* case. The distances of the remaining $k-1$ players from player i is exponentially distributed with mean party

distance d_{party}. This ensures that the player distribution in a party is geographically concentrated around player i, while also covering the cases when one or two players are from distant locations.

5.2 Server Location Model

Similar to the player locations, the server locations are also based on the player count per country, as well as the basic statistics on European cities. The DS remains in the EU West location Luxembourg, while ESs can be distributed by ranking the cities according to $f^y \approx f_x \cdot f_x^y$, where city y is in country x.

The majority of countries have very small values for f_x. On the other hand, f_x^y, can have high values (i.e. in countries where population is concentrated in one city). In this case, the city rank is greatly influenced by f_x^y, putting cities from countries with lower f_x at high ranks.

5.3 Migration Policy

The goal is to improve the gaming experience of players. On one hand, this entails hosting games in the server corresponding to the minimum average latency of interacting players. On the other hand, it is also important to take into account that servers have limited capacities. If the optimum server is full, and new games are still migrated there, players may have to wait for a long time, or worse, they may leave, before their game is hosted. These two factors are considered in the migration policy.

The latter is mainly based on physical distance, which is one factor influencing latency. The geographical distances are first computed given the latitude and longitude of players and servers. Then, considering a typical Dota 2 match of 10 players, the mean distance of the players to each server is obtained. From this, the servers are sorted by increasing mean distance.

Besides physical distance, the server load is also considered. This ensures that a game will not be migrated to a busy server; instead, it will be migrated to the next available server in the sorted list, or started if the DS is next in line.

5.4 Parameters and Metrics

First, we introduce the five main parameters whose values can be varied in the command line, allowing game providers to conduct customized performance evaluations, according to their parameters of interest.

Mean Inter-arrival Time of Game Requests: By default, the mean inter-arrival time of game requests varies according to $\beta(i, j)$, where $i = 0, 1, \ldots, 6$ and $j = 0, 1, \ldots, 23$, corresponding to mean inter-arrival time of hour i and day j for one week, demonstrating the load dynamics.

Simulation Mode: The simulation mode determines the game requests arrival processes that will be activated in a simulation, which can be 1, 2 or 3 (default). The three types of game request arrival processes correspond to the following types of players:

1. single players looking for a random match;
2. players who want to set up a party and play together with friends; and
3. players who want to join a party.

In mode 1 (RANDOM), there is only one game requests arrival process, which corresponds to player type 1, while in mode 2 (PARTY), there are two game requests arrival process, which correspond to player types 2 and 3. The default mode 3 (MIXED) is the combination of the first two modes, in which there are three game requests arrival process, corresponding to all types of players.

Server Capacities: By default, the capacities C_{DS} and C_{ES} are equal to 3000 and 1000 matches, respectively. In this framework, these values are high enough to ensure that all games are served, allowing for load analysis. The capacities can be set to different values to study different resource allocations. For instance, there could be cases when the resources at the edge are limited and may not be able to allocate such huge capacities. In this respect, the framework supports different allocation schemes for ESs with uniform or non-uniform capacities, with or without constraints on the total capacity among ESs.

Simulation Duration: The simulation duration is set in terms of weeks, which by default is equal to 1. This can be varied to study daily and, possibly, weekly patterns.

Further on, we introduce the two performance metrics considered in the evaluations. With these metrics, the game providers are able to study the effect of setting up ESs to both gaming QoE and server infrastructure.

Mean Distance to Server: As previously mentioned, the network between each player and the servers is not considered in the simulation; hence, the proximity of interacting players to the game server is measured in terms of physical distance, instead of latency. According to [21] there is a significant correlation between network delay and geographical distance. In this respect, the metric for gaming QoE used in this work is the mean distance of the 10 players in a match to the server hosting their game. From this, the game provider will have a view on the improvement of the gaming QoE as other parameters are varied.

Server Load: Besides the gaming QoE, game providers may also want to understand how the load on the servers changes as ESs are deployed. Load analysis can be performed on the DS, as well as on each ESs. Knowing the average load in a server opens new opportunities for resource allocation optimization and saving.

6. NUMERICAL EXAMPLES

A variety of parameter studies is conducted to evaluate the performance of the proposed game server migration towards the edge. In order to understand how the player distribution affects the gaming QoE, simulations are run in RANDOM, PARTY and MIXED simulation modes, with fixed mean inter-arrival time β. To demonstrate the effect on the DS load dynamics when migrations are introduced, simulations are run in MIXED mode dynamically varying mean inter-arrival times $\beta(t)$. Up to this point, all evaluations are performed using the default server capacities (i.e., $C_{DS} = 3000$ and $C_{ES} = 1000$) which are highly over-provisioned. This ensures that all matches are served in the optimum server. Further on, different resource allocation schemes are applied and analyzed based on the performance metrics. The number of ESs and their respective C_{ES} are varied in parameter studies. The results can help game providers in deciding on the number and capacity of ESs to be deployed and where. In order to add significance, simulation runs are repeated 10 times using different random number seeds. The resulting 95% confidence intervals are computed from the obtained data.

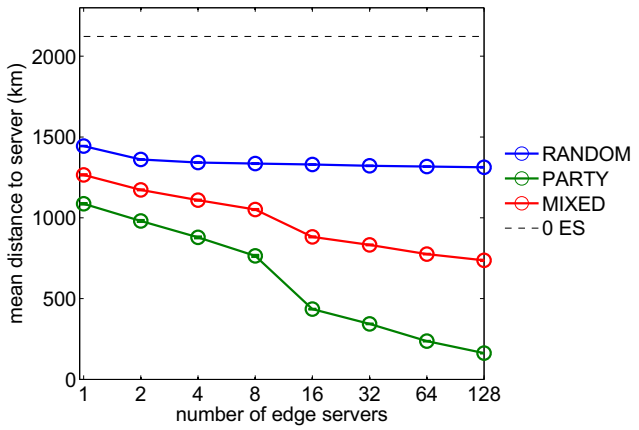


Figure 7: Effect of the player distribution on the mean distance to server.

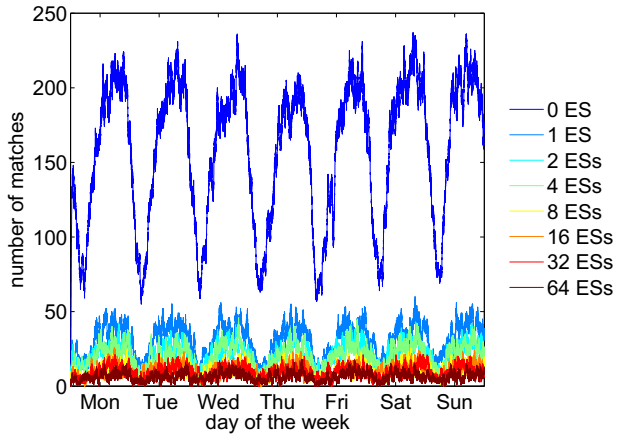


Figure 8: DS load dynamics with migrations.

6.1 Simulation Modes

The simulation mode determines how players are distributed in the matches of a simulation run (i.e., random, party or their combination). As shown in Fig. 7, game server migration improves the mean distances to server by at least 30%. Note that it particularly benefits players in parties, proposing 50% to up to over 90% improvements depending on the number of ESs deployed. Conversely, the improvement experienced by random players reaches an equilibrium at around 35% even when more ESs are added. The real-world gaming scenario (i.e. MIXED mode) depicts a trade-off between these two cases. These results imply that the proposed approach greatly improves the gaming QoE.

Although the approach is particularly beneficial to players in parties, the MIXED mode is considered in the succeeding evaluations, simulating a more realistic gaming scenario.

6.2 DS Load Dynamics

In order to investigate the impact of migrating to ES on the DS load, we evaluate the DS load dynamics. Fig. 8 shows the DS load dynamics when $0(\text{baseline}), 1, 2, 4, \ldots, 64$ ESs are deployed. The load on the DS server is given in number of matches dependent on the time of the day in the term of one week. The request process in the simulation framework fits the daily dynamics of the game request inter-arrival time β. Deploying 1 ES with decent capacity already reduces the peak load on the DS by around 75%. This implies that the location of the DS is not optimal for most of the players connecting to it. Placing an ES in Moscow, would highly improve the system performance because of the high number of players from Russia.

6.3 Resource Allocation Schemes

In the previous evaluations, the server capacities were configured high enough to ensure that all matches are served in the optimal server. However, there are cases where the resources at the edge are limited and may not be able to allocate sufficient capacities. In this case, the optimal server may be unavailable at some point, and the system must migrate the game server to the next best server available or host the game in the DS.

To understand their effect on the performance metrics, different resource allocation schemes are analyzed in this section. Particularly, ESs were allocated with uniform and non-uniform capacities, with and without constraints on the total capacity among ESs.

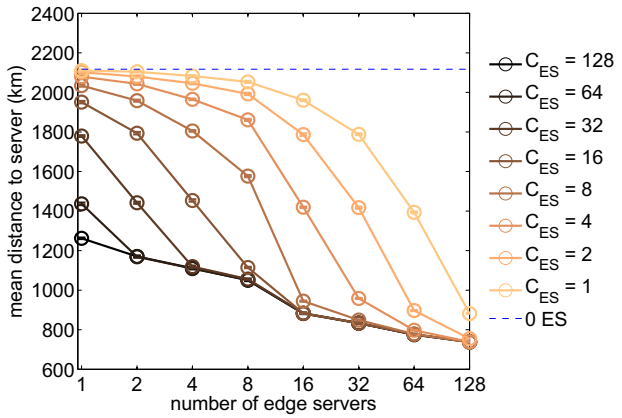


Figure 9: Effect of C_{ES} on the mean distance to server.

Without $C_{ES,tot}$ constraints: In this resource allocation scheme, ESs were allocated with uniform capacities without constraints on the $C_{ES,tot}$ (i.e. regardless of the number of ESs deployed, each is allocated with the specified C_{ES}). Simulations were run with $C_{ES} = 1, 2, 4, \ldots, 128$ match(es), and $1, 2, 4, \ldots, 128$ ESs deployed.

Fig. 9 illustrates the behavior of the mean distances to server as the number of ESs and C_{ES} are varied. These results help game providers visualize the effect of the allocation on the expected latency and the response time of games, allowing them to find the best compromise between resources and user satisfaction. For instance, deploying 16 ESs with at least $C_{ES} = 16$ matches is a good option, since adding more ESs only offers little improvements. Of course, the decision still depends on the availability of resources at the edge.

The same trend is observed in the behavior of the average DS load, as shown in Fig. 10. Understanding how the DS load changes with the allocation can help game providers in optimizing the resources in the DS. For instance, deploying 8 ESs with $C_{ES} = 16$ matches reduces the average DS load by about 90%. This means that most of the resources originally

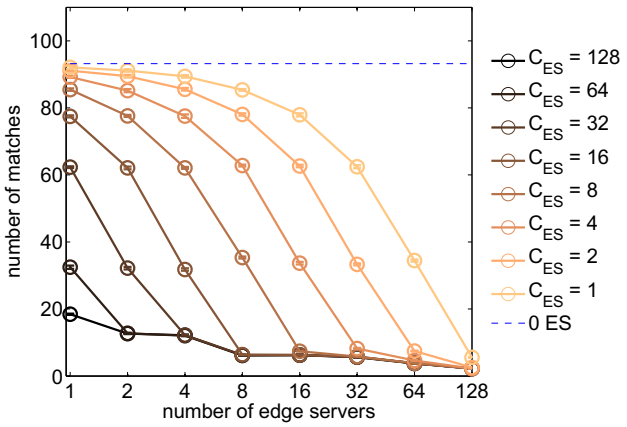


Figure 10: Effect of C_{ES} on the DS load.

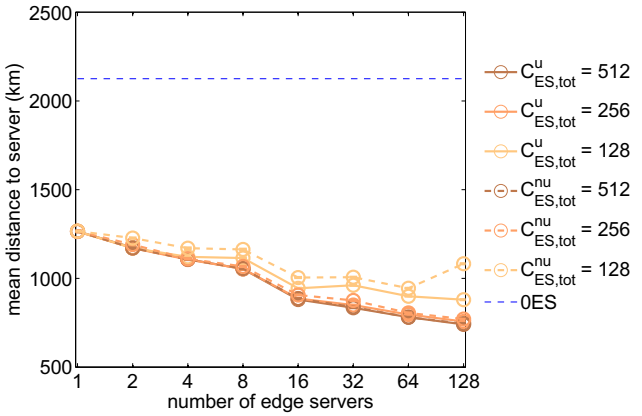


Figure 11: Effect of fixing the $C_{ES,tot}$ on the mean distance to server.

allocated for hosting the game can now be allocated for other applications.

With $C_{ES,tot}$ constraints: In the following resource allocation schemes, $C_{ES,tot}$ is fixed and distributed among ESs in two different ways: *(a) uniform* and *(b) non-uniform*. As the name implies, the former distributes the $C_{ES,tot}$, among the ESs equally, by simply dividing it by the number of ESs; conversely, the latter distributes the $C_{ES,tot}$ among the ESs according to the population in the ES locations. These can help game providers find out which is better, if any, between having few big ESs or many small ESs. Simulations were run with $C_{ES,tot} = 128, 256, 512$ matches and $1, 2, 4, \ldots, 128$ ESs deployed.

Fig. 11 shows similar performance of the uniform and non-uniform allocation, except when $C_{ES,tot} = 128$ matches, where the former yields better results. The mean distances to server are considerably increased if 128 ESs are deployed with non-uniform allocation, representing its drawback — it cannot support the deployment of more ESs for smaller $C_{ES,tot}$ values, as it tends to allocate $C_{ES} = 0$ to ESs located in cities with relatively small population. This means that at some point the effective number of ESs deployed is less than what is intended, and the ES(s) allocated with $C_{ES} = 0$ is/are actually those that would have been the optimal choice for a significant number of matches.

7. CONCLUSION

Multiplayer online battle arenas are rising online gaming services that have high requirements on the network parameters. The performance of the player and the gaming service highly depend on the distance and latency to the game server. To reduce latency, servers can be migrated to the edge of the network where they are in close proximity to the players.

In order to evaluate the impact of migration policies on the latency and load of game servers, we determined the load dynamics of the multiplayer online battle arena Dota 2, by evaluating match histories from the provided API. Based on the load dynamics we developed generic statistic models for game request processes with daily dynamics and match duration. We use the models to evaluate migration policies in an event-based simulation framework under realistic server load conditions.

Our results show that deploying one additional edge server can already reduce the mean distance to the server by one half and highly reduce the load on the dedicated server. For a fixed total capacity of edge resources, a high number of edge servers with smaller capacities is beneficial for the distance to the server, but also results in a higher operational overhead.

Acknowledgment

This work was funded in the framework of the EU ICT Project INPUT (H2020-2014-ICT-644672). The authors thank all project partners for their valuable contributions.

8. REFERENCES

[1] L. Backstrom, E. Sun, and C. Marlow. Find me if you can: improving geographical prediction with social and spatial proximity. In *Proceedings of the 19th international conference on World wide web*, pages 61–70. ACM, 2010.

[2] T. Beigbeder, R. Coughlan, C. Lusher, J. Plunkett, E. Agu, and M. Claypool. The Effects of Loss and Latency on User Performance in Unreal Tournament 2003®. In *Proceedings of 3rd ACM SIGCOMM Workshop on Network and System Support for Games*, Portland, OR, USA, 2004.

[3] P. B. Beskow, K.-H. Vik, P. Halvorsen, and C. Griwodz. The partial migration of game state and dynamic server selection to reduce latency. *Multimedia Tools and Applications*, 45(1-3):83–107, 2009.

[4] F. Chang and W. Feng. Modeling player session times of on-line games. In *Proceedings of the 2nd Workshop on Network and System Support for Games*, pages 23–26. ACM, 2003.

[5] S. Choy, B. Wong, G. Simon, and C. Rosenberg. A Hybrid Edge-cloud Architecture for Reducing On-demand Gaming Latency. *Multimedia Systems*, 20(5):503–519, 2014.

[6] Cisco. Cisco Visual Networking Index: Forecast and Methodology, 2014–2019. Technical report, Cisco, 2015.

[7] Cisco. The Zettabyte Era – Trends and Analysis. Technical report, Cisco, 2015.

[8] M. Dick, O. Wellnitz, and L. Wolf. Analysis of factors affecting players' performance and perception in multiplayer games. In *Proceedings of 4th ACM SIGCOMM Workshop on Network and System Support for Games*, pages 1–7. ACM, 2005.

[9] J. Färber. Network game traffic modelling. In *Proceedings of the 1st Workshop on Network and System Support for Games*, pages 53–57. ACM, 2002.

[10] L. Gardenghi, S. Pifferi, G. D'Angelo, and L. Bononi. Design and simulation of a migration-based architecture for massively populated internet games. In *Global Telecommunications Conference Workshops, 2004. GlobeCom Workshops 2004. IEEE*, pages 166–175. IEEE, 2004.

[11] C. Hyser, B. Mckee, R. Gardner, and B. J. Watson. Autonomic virtual machine placement in the data center. *HP Laboratories*, 2008.

[12] V. Jalaparti. Enabling Seamless Wide Area Migration of Online Games. Master's thesis, University of Illinois, 2013.

[13] M. Jarschel, D. Schlosser, S. Scheuring, and T. Hoßfeld. An Evaluation of QoE in Cloud Gaming Based on Subjective Tests. In *Proceedings of the 5th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS)*, Seoul, Korea, 2011.

[14] J. Kim, J. Choi, D. Chang, T. Kwon, Y. Choi, and E. Yuk. Traffic characteristics of a massively multi-player online role playing game. In *Proceedings of 4th ACM SIGCOMM Workshop on Network and System Support for Games*, pages 1–8. ACM, 2005.

[15] Maxmind. Free World Cities Database, 2015. Last accessed: 25-10-2015.

[16] Maxmind. ISO 3166 Country Codes with Associated Continent, 2015. Last accessed: 25-10-2015.

[17] E. Mohammadi, M. Karimi, and S. R. Heikalabad. A novel virtual machine placement in cloud computing. *Australian Journal of Basic and Applied Sciences*, 5(10):1549–1555, 2011.

[18] S. Möller, S. Schmidt, and J. Beyer. Gaming Taxonomy: An Overview of Concepts and Evaluation Methods for Computer Gaming QoE. In *Proceedings of the 5th International Workshop on Quality of Multimedia Experience (QoMEX)*, Klagenfurt, Austria, 2013.

[19] A. Normoyle, G. Guerrero, and S. Jörg. Player perception of delays and jitter in character responsiveness. In *Proceedings of the ACM Symposium on Applied Perception*, pages 117–124. ACM, 2014.

[20] S. Oechsner. JSimLib - A Simulation Framework for Java. Student project thesis, University of Würzburg, 2005.

[21] V. N. Padmanabhan and L. Subramanian. An investigation of geographic mapping techniques for internet hosts. In *ACM SIGCOMM Computer Communication Review*, volume 31, pages 173–185. ACM, 2001.

[22] A. Patro, S. Rayanchu, M. Griepentrog, Y. Ma, and S. Banerjee. The anatomy of a large mobile massively multiplayer online game. *ACM SIGCOMM Computer Communication Review*, 42(4):479–484, 2012.

[23] J. T. Piao and J. Yan. A network-aware virtual machine placement and migration approach in cloud computing. In *Grid and Cooperative Computing (GCC), 2010 9th International Conference on*, pages 87–92. IEEE, 2010.

[24] D. Pittman and C. GauthierDickey. Characterizing virtual populations in massively multiplayer online role-playing games. In *Advances in Multimedia Modeling*, pages 87–97. Springer, 2010.

[25] M. Ries, P. Svoboda, and M. Rupp. Empirical study of subjective quality for massive multiplayer games. In *Systems, Signals and Image Processing, 2008. IWSSIP 2008. 15th International Conference on*, pages 181–184. IEEE, 2008.

[26] Riot Games. League Players Reach New Heights in 2014, 2014. Last accessed: 22-10-2015.

[27] P. Rygielski and A. Gonczarek. Migration-aware optimization of virtualized computational resources allocation in complex systems. In *21st International Conference on Systems Engineering (ICSEng)*, pages 212–216. IEEE, 2011.

[28] Steam. Steam Web API Terms of Use, 2015. Last accessed: 16-11-2015.

[29] Steam Charts. Dota 2, 2015. Last accessed: 22-10-2015.

[30] P. Svoboda, W. Karner, and M. Rupp. Traffic analysis and modeling for world of warcraft. In *Communications, 2007. ICC'07. IEEE International Conference on*, pages 1612–1617. IEEE, 2007.

[31] Valve Corporation. Dota 2, 2013. Last accessed: 25-10-2015.

[32] A. Wattimena, R. E. Kooij, J. Van Vugt, and O. Ahmed. Predicting the perceived quality of a first person shooter: the quake iv g-model. In *Proceedings of 5th ACM SIGCOMM Workshop on Network and System Support for Games*, page 42. ACM, 2006.

[33] S. Zander and G. Armitage. Empirically Measuring the QoS Sensitivity of Interactive Online Game Players. In *Proceddings of Australian Telecommunication Networks and Applications Conference (ATNAC)*, Sydney, Australia, 2004.