Joint Update Rate Adaptation in Multiplayer Cloud-Edge Gaming Services: Spatial Geometry and Performance Tradeoffs

Saadallah Kassir
The University of Texas at Austin
Austin, TX, USA

Gustavo de Veciana
The University of Texas at Austin
Austin, TX, USA

Nannan Wang
Fujitsu Network Communications
Richardson, TX, USA

Xi Wang
Fujitsu Network Communications
Richardson, TX, USA

Paparao Palacharla
Fujitsu Network Communications
Richardson, TX, USA

ABSTRACT

In this paper, we analyze the performance of Multiplayer Cloud Gaming (MCG) systems. To that end, we introduce a model and new MCG-Quality of Service (QoS) metric that captures the freshness of the players' updates and fairness in their gaming experience. We introduce an efficient measurement-based Joint Multiplayer Rate Adaptation (JMRA) algorithm that optimizes the MCG-QoS by overcoming large (possibly varying) network transport delays by increasing the associated players' update rates. The resulting MCG-QoS is shown to be Schur-concave in the network delays, leading to natural characterizations and performance comparisons associated with the players' spatial geometry and network congestion. In particular, joint rate adaptation enables service providers to combat variability in network delays and players' geographic spread to achieve high service coverage. This, in turn, allows us to explore the spatial density and capacity of compute resources that need to be provisioned. Finally, we leverage tools from majorization theory, to show how service placement decisions can be made to improve the robustness of the MCG-QoS to stochastic network delays.

CCS CONCEPTS

• **Networks → Network performance modeling**; **Network performance analysis**.

KEYWORDS

Multiplayer Cloud Gaming, Rate Adaptation, Network Resource Provisioning, Service Placement, Edge Computing

ACM Reference Format:
Saadallah Kassir, Gustavo de Veciana, Nannan Wang, Xi Wang, and Paparao Palacharla. 2021. Joint Update Rate Adaptation in Multiplayer Cloud-Edge Gaming Services: Spatial Geometry and Performance Tradeoffs. In *The Twenty-second International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing (MobiHoc '21), July 26–29, 2021, Shanghai, China.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3466772.3467048

1 INTRODUCTION

Multiplayer Cloud Gaming (MCG) is emerging as one of the possible future dominant applications. Benefiting from the latest technological advances in communication networks, GPU virtualization, and high-performance computing systems in the cloud and/or edge network, this new *Gaming as a Service* business model is appealing to the three parties involved: gamers, game developers and game service providers (GSPs) [8]. Unlike traditional online gaming frameworks, the game is hosted at a remote server, allowing the players to interact by sending regular updates and receiving a video stream without the need for dedicated hardware or need to own the game license. Multiple platforms are already being commercialized, such as GeForce Now by Nvidia [3], Project xCloud by Xbox [5], Google Stadia [2], Playstation Now [4], and Amazon Luna [1].

While such platforms are becoming increasingly popular, many network design questions associated with delivering the best MCG-Quality of Service (MCG-QoS) across players remain unexplored. These include the choice of effective network architectures, resource allocation/provisioning strategies, performance guarantees under stochastic network congestion, or ensuring fairness amongst heterogeneous players. These technical challenges are likely to be exacerbated by the tight performance guarantees required by Extended Reality (XR) enhanced collaborative applications/games [14].

Contributions. A key challenge associated with supporting multiplayer games is that they may involve a (possibly large) number of geographically dispersed players increasing the games' exposure to congestion variations across a large set of network regions/resources. This makes the MCG-QoS potentially volatile making it particularly difficult to provide stable guarantees. In this paper, we address this general challenge through four key contributions.

The first contribution is the introduction of a novel multiplayer game model and MCG-QoS metric which captures the joint impact of the network delays/congestion experienced over different timescales by the all the players participating in the game.

The second contribution is the development of a measurement-based Joint Multiplayer Rate adaptation Algorithm (JMRA) geared at ensuring that the information transmitted by all the players is delivered and processed in a timely manner at the game server. We show how players can overcome large network delays through increased update rates to improve the MCG-QoS.

The third contribution lies in showcasing how GSPs can leverage the benefits of JMRA to cost effectively provision network resources so as to guarantee high service coverage. Leveraging tools from multivariate majorization theory, we relate the MCG-QoS to the player's spatial configuration and identify the worst-case geometry for a given "geographical spread". We show how the spread impacts the MCG-QoS, and how GSPs might envision provisioning network resources to meet service coverage requirements, e.g., by exploiting edge computing to deliver services with tight timeliness constraints.

The fourth contribution consists in providing a basis to study the MCG service placement problem using majorization theory. We propose a strategy that can be adopted by GSPs to make the MCG-QoS robust to stochastic variations in the network delays/congestion.

We note that the framework we present in this work is not limited to gaming applications. It is indeed also relevant to the general setting of provisioning real-time collaborative cloud-based services to geographically scattered participants/contributors, e.g., collaborative document editing, source code version control, etc.

Related Work. Multiple researchers have proposed approaches to optimize the performance of MCG networks, in a wide variety of settings, e.g., proposing energy-aware solutions [10] or cost-effective resource allocation strategies, Virtual Machine (VM) placement and network architectures subject to QoS constraints [12, 13, 16, 18]. However, this body of work does not place emphasis on the impact that each individual player has on the overall gaming experience. Other studies have focused on the interaction among players. In [9], the authors distinguish the notions of absolute response delay and inter-player delay which in turn allows them to study the fairness among the players, and place the VM resources accordingly. In [15], the authors solve a multi-objective optimization problem to solve the network resource provisioning problem, by minimizing both the worst inter-player delay and the network operating cost. However, none of these works consider the effect of adapting the players' update rates to improve on the freshness of the game state data received/computed at the server side. This notion of freshness has been studied in the context of Cloud Gaming in [22]. While the game server update rate is fixed, the authors analyze the effect of synchronizing the game server's and the players' phases under stochastic network delays, to minimize the mean *Age-of-Information* at the player side. Unlike this study, our focus is on the player-to-server traffic, characterizing the timeliness of the information processed at the game server side.

Paper Organization. This paper is organized as follows. In Section 2, we present our system model and MCG-QoS metric. We then introduce and study the properties of the JMRA algorithm towards optimizing the MCG-QoS in Section 3. In Section 4, we model and study the impact of the players' geographical locations on the MCG-QoS, and we deduce some approaches that can be used by GSPs to provision the network resources. We then capitalize on the models and performance characterization in Section 5 to provide additional insights about service placement strategies to face network delays variability. We conclude the paper in Section 6.

2 SYSTEM MODEL

2.1 Network Architecture

We consider a multiplayer cloud gaming system composed of three entities, consistent with MCG network architectures studied in the literature, see [12, 15, 23]:

(1) A set $\mathcal{P}$ of n players in a geographic configuration $\mathbf{x} = (\mathbf{x}_i, i \in \mathcal{P})$, where $\mathbf{x}_i \in \mathbb{R}^2$ corresponds to the location of player i.
(2) A Game-server (G-server) running on a VM in a compute node at location $\mathbf{g} \in \mathbb{R}^2$ that hosts the game, i.e., that keeps track of the state of the game by receiving and processing updates from the players.
(3) Rendering servers (R-servers) that receive aggregate state information from the G-server, render the video feed and stream it to the players. R-servers are typically placed closer to the players than the G-server to reduce network congestion, but they can also be colocated in the same datacenter.

2.2 Network Delay Variation Model

We model network delay variations happening on two different time scales: (1) slow time-scale variations, happening on the order of seconds or minutes, modeling overall network congestion level, and (2) fast time-scale packet delay variations, happening on the order of milliseconds or microseconds, modeling jitter and instantaneous bottlenecks in the network. In this work, we investigate the effect of slow time-scale variations due to network congestion, while abstracting out the fast time-scale variations, using a point estimate, e.g., the mean, median, or 90th percentile of this stochastic process over a small time window, and over which the slow time-scale delay variations are assumed to be constant. Hence, this point statistic is itself slowly varying over time. We model the slow-variations of this point statistic due to network congestion delays between player i and the G-server via a random variable D_i^t. In the sequel, we loosely refer to it as the *typical transport delay* experienced by player i. In addition, we assume that the impact of the players' updates on the network congestion they see is negligible.

2.3 Game Operation Model

The game's operation model, depicted in Figure 1, can be summarized in three stages.

(1) Player i sends periodic updates at a rate ρ_i updates per second to the G-server, containing instructions from the game controller (e.g., character's movement, camera/headset rotation) and experiences a typical transport delay D_i^t seconds.
(2) The G-server, in turn, forms periodic update batches by aggregating and processing everything it has received from the players every τ seconds. This model can be characterized by two types of delays (1) the individual random waiting time

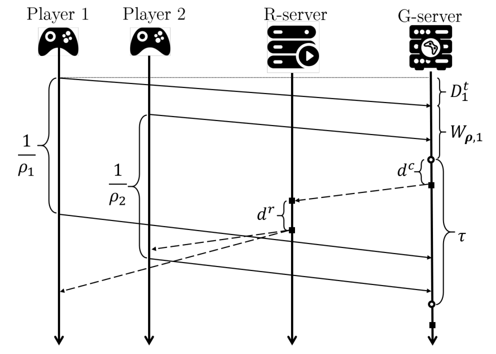


Figure 1: Figure of the game operation timeline

$W_{\boldsymbol{\rho},i}$ capturing the time between the latest update received from player i and the beginning of a typical compute cycle, and (2) a shared batch compute delay to process the updates aggregated in a batch, denoted by d^c. We model this batch compute delay $d^c(s)$ as a deterministic differentiable convex function of the game's overall update load $s = \sum_j \rho_j$ at the G-server side. While the load is stochastic (as it depends on the relative phases of the players' update cycles and the server's compute cycles), we assume that the load variations are negligible, and s can be seen as the mean batch load. In addition, we impose the constraint $d^c(s) \leq \tau$ to ensure that the G-server completes the processing of a batch before the end of a compute cycle of period τ, see Figure 1.

(3) Once the aggregated game's state is computed, the G-server sends it to the R-servers, that will render the video feed in d^r seconds and stream it back to the associated players.

In this paper we focus on the timeliness of the player to G-server traffic, leaving the analysis of the round-trip loop as future work.

2.4 Game Timeliness Model

In our framework, we seek to ensure that the state of the game is updated in a timely fashion. We use as our timeliness metric the *age of the game* $A_{\mathbf{d}^t,\boldsymbol{\rho}}$, conditioned on a given delay vector $\mathbf{d}^t$ (a realization of the random vector $\mathbf{D}^t = (D_i^t, \forall i \in \mathcal{P})$) and a choice of update rate vector $\boldsymbol{\rho} = (\rho_i, \forall i \in \mathcal{P})$, and defined as:

$$A_{\mathbf{d}^t,\boldsymbol{\rho}} = \max_{i=1,\ldots,n} [d_i^t + W_{\boldsymbol{\rho},i}] + d^c(\sum_{j=1}^{n} \rho_j) \tag{1}$$

where $W_{\boldsymbol{\rho},i} \sim \text{Uniform}([0, \frac{1}{\rho_i}]), \forall i \in \mathcal{P}$ as we assume that the transmissions are asynchronous with the G-server's batch processing regular schedule, hence a typical compute batch starts to get processed at a uniform random time in an interval of $1/\rho_i$ seconds.

In this formulation, the age of each player's updates processed at the G-server is the sum of the transport delay, waiting time at the G-server, and batch compute delay. The age of the game is then modeled as the maximum age of the each player's updates to capture the fact that stragglers considerably impact the quality of experience of all the other players interacting on a common virtual space. This formulation also hints at the need to ensure that network resources are "fairly" shared among the players involved in the game. This model also recognizes that the game's age depends jointly on the players' update rates and the network delays. As the update rates can be controlled, we identify a basic tradeoff involving this decision. While larger update rates lead to smaller waiting times, and help reduce the age of the game's state, this also results in larger computation delays, increasing the game's age.

2.5 The JMRA Problem

Our objective is to solve the Joint Multiplayer Rate Adaptation (JMRA) problem that ensures a small game age, defined as follows:

Problem 1: (Joint Multiplayer Rate Adaptation). *Given the current transport delay vector* $\mathbf{D^t} = \mathbf{d^t}$*, and a desired age level* a_0*, the JMRA problem consists in finding the update rate vector that maximizes the probability that the age of the game does not exceed some desired level* a_0*, i.e., solving:*

$$\boldsymbol{\rho}^*(\mathbf{d}^t) = \arg\max_{\boldsymbol{\rho}} \left\{\mathbb{P}(A_{\mathbf{d}^t,\boldsymbol{\rho}} \leq a_0) : d^c(\sum_j \rho_j) \leq \tau\right\} \tag{2}$$

$$= \arg\max_{\boldsymbol{\rho}} \mathbb{P}(\max_i [d_i^t + W_{\boldsymbol{\rho},i}] + d^c(\sum_j \rho_j) \leq a_0)$$

$$s.t.\ d^c(\sum_j \rho_j) \leq \tau \tag{3}$$

$$= \arg\max_{\boldsymbol{\rho}} \prod_{i=1}^{n} \mathbb{P}(W_{\boldsymbol{\rho},i} \leq a_0 - d_i^t - d^c(\sum_j \rho_j))$$

$$s.t.\ d^c(\sum_j \rho_j) \leq \tau \tag{4}$$

$$= \arg\max_{\boldsymbol{\rho}} \sum_{i=1}^{n} \left[\log\left(\rho_i \cdot (a_0 - d_i^t - d^c(\sum_j \rho_j))\right)\right]_-$$

$$s.t.\ d^c(\sum_j \rho_j) \leq \tau \tag{5}$$

where we define $x_- = \min[x, 0]$. We can now define the MCG Quality of Service as follows:

Definition 1: (MCG-QoS). *For a given transport delay vector* $\mathbf{d}^t \in \mathbb{R}_+^n$*, we define the MCG-QoS* $q(\mathbf{d}^t)$ *as:*

$$q(\mathbf{d}^t) = \begin{cases} \mathbb{P}(A_{\mathbf{d}^t,\boldsymbol{\rho}^*(\mathbf{d}^t)} \leq a_0), & \text{if } \boldsymbol{\rho}^*(\mathbf{d}^t) \text{ exists,} \\ 0, & \text{otherwise,} \end{cases} \tag{6}$$

i.e., the probability that the age constraint is met under JMRA.

In the sequel, we will present an efficient algorithm that solves for $\boldsymbol{\rho}^*(\mathbf{d}^t)$, hence computing $q(\mathbf{d}^t)$, under slowly varying delays.

3 THE RATE ADAPTATION ALGORITHM

In this section, we derive the Joint Multiplayer Rate Adaptation algorithm (JMRA), which jointly uses measured network delays to solve the JMRA problem described in Problem 1. We then study some of its properties, before evaluating its performance.

3.1 Algorithm Description

We envision an algorithm to be executed periodically at the G-server side. At each iteration, the G-server characterizes the transport delays to each player in the game. This can be done, for instance, by estimating the distribution of the (fast time-scale) packet delays experienced by previous updates in a sliding window, and computing the desired point estimate, e.g., the mean, median or 90th percentile. We assume that the players and G-server can measure time using synchronized clocks, hence the packet delays can be estimated with reasonable accuracy. Based on this information, the G-server re-optimizes the players' update rates accordingly, hence adapting to slow-time variations in the delays/congestion. Naturally, the frequency of execution of the algorithm would depend on the time scale for (slow) variations in the network delays' statistics. We describe below the procedure to optimize for the players' update rates given the vector of measured network delays $\mathbf{d}^t$.

Observe that Problem 1 is convex but it has a non-differentiable cost function leading to slow convergence of a numerical solver. We propose an alternative algorithm to compute the optimal update rate vector, by decomposing the optimization problem into a set of simpler (smooth) sub-problems. In addition, this analysis allows us to extract some basic properties of the JMRA policy.

Our approach is to first consider the case where the aggregate game server load $s = \sum_j \rho_j$ is fixed and known. Under this assumption, one can use the Lagrange multiplier procedure to solve for the optimal $\boldsymbol{\rho}^*(\mathbf{d}^t, s)$ in Problem 1 as a function of s. We find:

$$\rho_i^*(\mathbf{d}^t, s) = \min\Big[\frac{1}{\mu(\mathbf{d}^t, s)}, \frac{1}{a_0 - d_i^t - d^c(s)}\Big], \forall i \tag{7}$$
$$\text{where } \mu(\mathbf{d}^t, s) \text{ is s.t. } \sum_j \rho_j^*(\mathbf{d}^t, s) = s.$$

Observe that for any delay vector $\mathbf{d}^t$, and any choice of the sum-rate s, two types of players can be distinguished. We have on one hand a set $\mathcal{S}(\mathbf{d}^t, s)$ of *support players*, that will all pick the same update rate of $\frac{1}{\mu(\mathbf{d}^t,s)}$. These players see a transport delay to the G-server that is too large to be able to fully adapt their update rate to compensate for the large transport delays. We have from Equation 7, player $i \in \mathcal{S}(\mathbf{d}^t, s)$ if $d_i^t \geq a_0 - \mu(\mathbf{d}^t, s) - d^c(s)$. We have on the other hand a set $\overline{\mathcal{S}}(\mathbf{d}^t, s)$ of *non-support players* that can flexibly trade-off delay for update rate, where $i \in \overline{\mathcal{S}}(\mathbf{d}^t, s)$ if $d_i^t < a_0 - \mu(\mathbf{d}^t, s) - d^c(s)$. The players in this set do not impact directly the MCG-QoS, except by contributing to the total server congestion, as their respective terms in the sum vanish after substituting $\rho_i^*(\mathbf{d}^t, s)$. Equipped with the notion of support set, we can state the following property:

Property 1: (Support Set Monotonicity in Transport Delays). *For a given delay vector* $\mathbf{d}^t \in \mathbb{R}_+^n$, *and sum update rate* $s \in \mathbb{R}_+$, *if* $d_i^t \geq d_j^t$, *then* $j \in \mathcal{S}(\mathbf{d}^t, s) \implies i \in \mathcal{S}(\mathbf{d}^t, s)$. *Thus, there are only* $n + 1$ *possible support sets.*

We now use this property along with the solution of Equation 7 to solve for the optimal load s. After substituting $\boldsymbol{\rho}^*(\mathbf{d}^t, s)$, Problem 1 reduces to solving the one dimensional problem:

$$s^*(\mathbf{d}^t) = \arg\max_s \sum_{i=1}^n [\log \frac{a_0 - d_i^t - d^c(s)}{\mu(\mathbf{d}^t, s)}]_- \tag{8}$$
$$\text{s.t.} \begin{cases} \sum_{i=1}^n \min(\frac{1}{\mu(\mathbf{d}^t,s)}, \frac{1}{a_0 - d_i^t - d^c(s)}) = s \\ d^c(s) \leq \tau \end{cases}$$

This problem is still not easily solved as the cost function is non-differentiable and the constraint set is non-convex. However, by integrating more information about the support set, one can simplify this problem further. Invoking Property 1, we can distinguish $n + 1$ cases corresponding to the $n + 1$ possible support sets. Let $\mathcal{S}_m(\mathbf{d}^t)$ be the possible support set that contains m players with the largest measured delays, for $0 \leq m \leq n$. Each sub-problem reduces to solving for the optimal game server load $s_m^*(\mathbf{d}^t)$ assuming that $\mathcal{S}_m(\mathbf{d}^t)$ is the support set. For each sub-problem m, the Lagrange multiplier $\mu(s, \mathbf{d}^t)$ becomes $\mu_m(\mathbf{d}^t, s)$ and is dictated by the first constraint in Equation 8. Without loss of generality, we index the players in descending order of transport delays, and we get that:

$$\mu_m(\mathbf{d}^t, s)^{-1} = \frac{1}{m}\left(s - \sum_{i=m+1}^n \frac{1}{a_0 - d_i^t - d^c(s)}\right) \tag{9}$$

After substituting this constraint into the cost function of the mth sub-problem $\mathcal{U}_m(\mathbf{d}^t, s)$, the latter reduces to:

$$\mathcal{U}_m(\mathbf{d}^t, s) = \sum_{i=1}^m \log(a_0 - d_i^t - d^c(s)) \tag{10}$$
$$+ m \cdot \log(s - \sum_{i=m+1}^n \frac{1}{a_0 - d_i^t - d^c(s)}) - m \cdot \log(m)$$

and the new sub-problem becomes:

$$s_m^*(\mathbf{d}^t) = \arg\max_s \left\{\mathcal{U}_m(\mathbf{d}^t, s) : \mu_m(\mathbf{d}^t, s) > 0,\ d^c(s) \leq \tau\right\} \tag{11}$$

where $\mathcal{U}_m(\mathbf{d}^t, s)$ is smooth and concave in s over the range of values where $\mu_m(\mathbf{d}^t, s) > 0$ and the constraint set is convex. We note that the range of s values where $\mu_m(\mathbf{d}^t, s) > 0$ can be found by solving for the roots of the non-linear Equation 9. Hence, the solution to this problem can be found using a convex optimization solver, e.g., Gradient Ascent, along with a non-linear equation solver. Finally, the optimal $s^*(\mathbf{d}^t)$ is such that $s^*(\mathbf{d}^t) = s_{m^*}(\mathbf{d}^t)$ where $m^* = \arg\max_m \mathcal{U}_m(\mathbf{d}^t, s_m^*)$. We summarize JMRA in Algorithm 1.

Algorithm 1: Joint Multiplayer Rate Adaptation (JMRA)

```
Result: Solves for $\boldsymbol{\rho}^*(\mathbf{d}^t)$
1  Estimate $d_i^t$, $\forall i$;
2  Solve $s_0 = \sum_{i=1}^n \frac{1}{a_0 - d_i^t - d^c(s_0)}$;
3  if $s_0$ exists AND $d^c(s_0) < \tau$ then
4      $\rho_i^* = \frac{1}{a_0 - d_i^t - d^c(s_0)}, \forall i$;
5  else
6      for m=1 to n do
7          Find range of feasible $s$ s.t. $\mu_m(\mathbf{d}^t, s) > 0$;
8          Solve for $s_m^*(\mathbf{d}^t)$ in Eq. 11 using Gradient Ascent;
9      end
10     Pick $s^*(\mathbf{d}^t) = s_{m^*}(\mathbf{d}^t)$, $m^* = \arg\max_m \mathcal{U}_m(\mathbf{d}^t, s_m^*(\mathbf{d}^t))$;
11     Compute $\boldsymbol{\rho}^*(\mathbf{d}^t) = \boldsymbol{\rho}^*(\mathbf{d}^t, s^*(\mathbf{d}^t))$ using Equation 7
12 end
```

3.2 Algorithm Analysis

In Section 3.1, we proposed and established the optimality of the JMRA algorithm. We now study some interesting properties thereof, and compare its performance to a baseline where all the players share the same update rate. We start by stating a theorem that relates the players' update rates to their delays to the G-server.

Proposition 1: (Rate-Proximity Tradeoff). *Given a delay vector* $\mathbf{d}^t \in \mathbb{R}_+^n$ *and any players* $i, j \in \mathcal{P}$, *if* $d_i^t \geq d_j^t$ *then* $\rho_i^*(\mathbf{d}^t) \geq \rho_j^*(\mathbf{d}^t)$.

This proposition follows directly from Equation 7 and Property 1. Intuitively, this property states that players experiencing large transport delays, can compensate for this by increasing the rate at which they send information to the game server. While a unilateral increase in a player's update rate increases the G-server load, and hence all the players' compute delays, the algorithm finds the appropriate congestion level for the given player's configuration. An alternate interpretation would be that the players that experience the smallest delays are willing to reduce their update rates to allow players with larger transport delays to benefit from increased communication/compute resources.

We now compare the performance of the JMRA algorithm with one that distributes equal communication/compute resources among the players. We introduce the Best Static Rate algorithm (BSR), that solves Problem 1 subject to the additional constraint that all the players' update rates are the same. We note that this algorithm outperforms any algorithm that uses a fixed update rate imposed by the game designer, as the static update rate is still optimized based on the delay vector. Hence, the reported performance of the BSR algorithm should be viewed as as an upper-bound on what can be achieved when players used fixed update rates. We compare the two algorithms using the following performance metric:

Definition 2: (ϵ-Playable Games). *A game is said to be ϵ-playable for a given transport delay vector $\mathbf{d}^t \in \mathbb{R}_+^n$, and $\epsilon \in [0, 1]$, if its MCG-QoS function satisfies the condition $q(\mathbf{d}^t) > 1 - \epsilon$.*

To compare the performance of the JMRA and BSR update rate selection algorithms, we characterize the probability of a game being ϵ-playable under randomly generated player configurations. Towards evaluating the impact of the game's spatial geometry, i.e., the relative placement of the players and the G-server, we consider a setup where n players are placed randomly and uniformly in a disk of radius r meters forming a player configuration $\mathbf{X} \in \mathbb{R}^{n\times 2}$, while the G-server is placed in the center of the disk assumed to be the origin, i.e., $\mathbf{g} = \mathbf{0}$. In addition, we model the network delay experienced by player i to simply be a deterministic increasing linear function of its distance to the G-server, hence:

$$d_i^t(\mathbf{x}, \mathbf{g}) = \beta_1 + \beta_2 \cdot \|\mathbf{x}_i - \mathbf{g}\|_2 \tag{12}$$

where β_1 captures the communication delay components that are independent of the distance between the player and the G-server, e.g., contention delay on the wireless interface, and β_2 is a coefficient inversely proportional to the speed of light in fiber. We pick values of $\beta_1 = 3 \times 10^{-3}$ seconds and $\beta_2 = 1 \times 10^{-8}$ seconds/meter, consistently with empirical studies in WANs, see [17].

We also assume a specific model for the batch compute delay d^c:

$$d^c(s) = \frac{s \cdot \tau}{n \cdot k_G} \tag{13}$$

where k_G corresponds to the compute capacity reserved per player resulting from the compute resources allocated to the G-server's VM. This functional form is motivated by the fact that the expected number of updates received in a batch, i.e., in a window of τ seconds, is $s \cdot \tau$ updates, while the overall compute capacity allocated for the G-server's VM is such that it can process $n \cdot k_G$ updates/second.

Equipped with these models, we compare in Figure 2 the likelihood that a randomly generated game configuration is ϵ-playable, under the JMRA and BSR algorithms.

The figure clearly shows that JMRA outperforms the BSR algorithm, as a random player configuration is more likely to lead to a playable game configuration. The performance gap is most significant for small ϵ, i.e., in the regime we expect to operate. For instance, if a GSP seeks to offer high-quality games by picking $\epsilon = 0$, then around 79% of the randomly generated game configurations can be supported under JMRA, while none can be supported under BSR. The main takeaway of these results lies in the observation that allowing flexibility in the choice of the players' update rates considerably improves the MCG-QoS, or equivalently, allows the GSPs to deploy reduced network resources for a desired MCG-QoS.

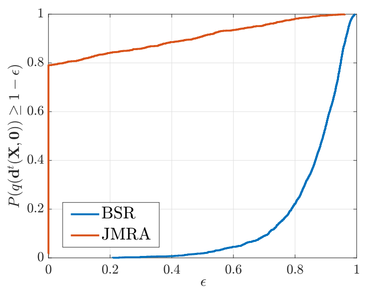


Figure 2: Comparison of the probability of the game being ϵ-playable under JMRA and BSR algorithms as a function of ϵ, under random players' configurations; $n = 20$, $r = 3 \times 10^6$**m,** $a_0 = 50$**ms,** $\tau = 20$**ms,** $k_G = 150$ **updates/s/player.**

4 SERVICE COVERAGE ANALYSIS AND NETWORK RESOURCE PROVISIONING

So far we have presented an efficient algorithm to maximize the MCG-QoS through joint multiplayer rate adaptation for a given network congestion regime as captured by the network delay vector. A GSP will, however, want to guarantee that newly instantiated games are playable for configurations that are not "too spread-out" and/or experiencing network congestion outliers. In this section, we further introduce a model tying geometry to network congestion and analyze the large-scale compute resources a GSP would need to deploy (e.g., rent from spatially distributed cloud service providers) so as to ensure that MCGs with a given geographical spread will be playable under JMRA – i.e., ensure MCG service coverage.

4.1 Linking Players' Spatial Geometry to Network Congestion

To capture the relationship between players' configuration geometry and large-scale network congestion, we model the *typical* transport delay vector $\mathbf{D}^t$ experienced by the players, via a random vector $\mathbf{D}^t_{\boldsymbol{\delta}} = (D^t_{\boldsymbol{\delta},i}, i \in \mathcal{P})$. It is parametrized by a distance vector $\boldsymbol{\delta} = (\delta_i, i \in \mathcal{P})$, where δ_i is the distance from player i to the G-server and $D^t_{\boldsymbol{\delta},i}$ models the typical slow variations in transport delay experienced by player i a distance δ_i from the G-server.

Remark. We emphasize that this model is not intended to capture the specific characteristics of congestion/delay variations as seen at a particular location, but instead what would be typically experienced by players in a large scale network to enable a study of the large-scale resources the GSP would require.

Assumption 1: (Network Congestion Model). *We assume for any player configuration with associated distance vector $\boldsymbol{\delta} \in \mathbb{R}^n$ that $\mathbf{D}^t \sim \mathbf{D}^t_{\boldsymbol{\delta}}$ where the entries of $\mathbf{D}^t_{\boldsymbol{\delta}}$ are mutually independent and furthermore for $\mathbf{z} \in \mathbb{R}^n_+$ and $i \in \mathcal{P}$ we have that*[1] $\mathbb{E}[D^t_{\boldsymbol{\delta},i}] \leq \mathbb{E}[D^t_{\boldsymbol{\delta}+\mathbf{z},i}]$ *and* $\mathbf{D}^t_{\boldsymbol{\delta}} \leq^{icx} \mathbf{D}^t_{\boldsymbol{\delta}+\mathbf{z}}$.

[1]As in[21], we define *increasing convex dominance* as $\mathbf{X} \leq^{icx} \mathbf{Y} \iff \mathbb{E}[\phi(\mathbf{X})] \leq \mathbb{E}[\phi(\mathbf{Y})]$, for all increasing convex $\phi : \mathbb{R}^n \to \mathbb{R}$.

Intuitively players that are further away from the G-server would on average experience higher transport delays, e.g., due to increased propagation delay and number of hops traversed. Perhaps more importantly with higher distances one might expect a higher variability due to congestion on intervening network resources. The multivariate *increasing convex (icx) ordering*, assumed above, see [21], partially captures an ordering in delay "variability" with distance.

As previously mentioned, the distribution of $\mathbf{D}^t_\delta$ reflects slowly varying network congestion an MCG game will need to overcome through rate adaptation. For simplicity, the GSP might provision network resources respect to a point estimate of the transport delay distributions, e.g., the mean, median or 90th percentile of the delay distribution for each user, depending on the GSP's risk tolerance. In Section 5, we discuss how the risk of under-provisioning the network due to variability in the network delays/congestion can be curtailed by the GSP during network operation. We denote as $\bar{\mathbf{d}}^t(\mathbf{x}, \mathbf{g})$ the transport delay vector associated with the desired network delay statistic, as a function of the player configuration $\mathbf{x}$ and the G-server location $\mathbf{g}$. For instance, $\bar{d}^t_i(\mathbf{x}, \mathbf{g}) = \mathbb{E}[D^t_{\delta(\mathbf{x},\mathbf{g}),i}]$, where $\delta_i(\mathbf{x}, \mathbf{g}) = \|\mathbf{x}_i - \mathbf{g}\|_2$, if the GSP decides to provision the network resources for the mean transport delays.

4.2 Characterization of Geographical Spread

We characterize the players' geographical spread as follows:

Definition 3: (Geographical Spread). *Let $\mathbf{x} \in \mathbb{R}^{n\times 2}$ be an n-player configuration, we define the configuration's geographical spread $\sigma(\mathbf{x})$ as the radius of the smallest disk containing all the players, i.e.,:*

$$\sigma(\mathbf{x}) = \min_{r\in\mathbb{R}_+, \mathbf{c}\in\mathbb{R}^2} \left\{ r : \|\mathbf{c} - \mathbf{x}_i\|_2 \le r, \ \forall i \right\} \tag{14}$$

Clearly, the larger $\sigma(\mathbf{x})$ is, the more "spread out" the players are, and hence, the harder it will be to find a server location that can ensure the game is ϵ-playable. We also introduce the notion of a regular configuration, which will be useful to characterize the class of configurations with n players with a given geographical spread.

Definition 4: (Regular Configuration). *An n player configuration of radius r $\boldsymbol{\omega}(n, r) \in \mathbb{R}^{n\times 2}$ is said to be regular iff all the players are equispaced on a circle of radius r.*

For instance, the configuration $\boldsymbol{\omega}(n, r)$ such that $\boldsymbol{\omega}_i(n, r) = (r \cdot \cos\frac{2\pi(i-1)}{n}, r \cdot \sin\frac{2\pi(i-1)}{n}), 1 \le i \le n$ is a regular configuration.

4.3 Service Coverage Analysis

Next we focus our attention on how the players' geographical spread impacts the service coverage and the GSP's network resource provisioning strategies. We assume GSPs whose goal is to ensure that games involving players with a given geographical spread will find a G-server (with high probability) such that the game is ϵ-playable. To that end, a GSP can control the density of compute nodes, as well as the compute capacity allocated to the G-servers' VMs. We study how these decisions are coupled and impacted by the target games' geographical spread to be supported.

We shall define first a useful service coverage metric that we adopt in this framework, as a function of the delay vector $\bar{\mathbf{d}}^t(\mathbf{x}, \mathbf{g})$.

Definition 5: (ϵ-Feasible Region). *For a given player configuration $\mathbf{x} \in \mathbb{R}^{n\times 2}$, we define the ϵ-feasible region $\mathcal{F}_\epsilon(\mathbf{x})$ to include all G-server locations for which the game would be ϵ-playable as:*

$$\mathcal{F}_\epsilon(\mathbf{x}) = \{\mathbf{g} \in \mathbb{R}^2 : q(\bar{\mathbf{d}}^t(\mathbf{x}, \mathbf{g})) > 1 - \epsilon\}, \tag{15}$$

Moreover, the area of region $\mathcal{F}_\epsilon(\mathbf{x})$ can be expressed as:

$$|\mathcal{F}_\epsilon(\mathbf{x})| = \iint_{\mathbb{R}^2} \mathbb{1}\{q(\bar{\mathbf{d}}^t(\mathbf{x}, \mathbf{g})) > 1 - \epsilon\} d\mathbf{g} \tag{16}$$

Large feasible areas $|\mathcal{F}_\epsilon(\mathbf{x})|$ are preferred as they are associated with large likelihood to find a G-server that can support the MCG service given the player configuration $\mathbf{x}$, see Figure 3.

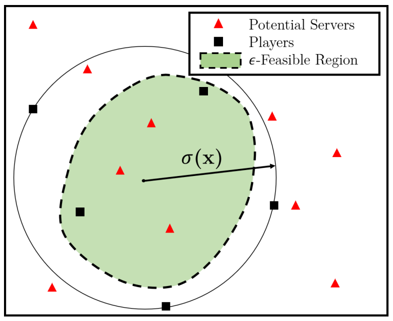


Figure 3: Example of an ϵ-feasible region, with $n = 5$ players.

To formally characterize the network provisioning problem, we model the compute nodes to be deployed according to a homogeneous spatial Poisson Point Process (PPP) $\Phi(\lambda)$ of intensity λ compute nodes/m^2. The GSP aims to provision the network resources so as to ensure an $(n, \sigma, \epsilon, \alpha, \nu)$-service coverage, defined as:

Definition 6: ($(n, \sigma, \epsilon, \alpha, \nu)$-Service Coverage). *An MCG network is said to ensure an $(n, \sigma, \epsilon, \alpha, \nu)$-service coverage, for $n \in \mathbb{N}, \sigma \in \mathbb{R}_+, \epsilon \in [0, 1], \alpha \in [0, 1], \nu \in \mathbb{N}$, if for any n-player configuration $\mathbf{x}$ with a geographical spread less than σ:*

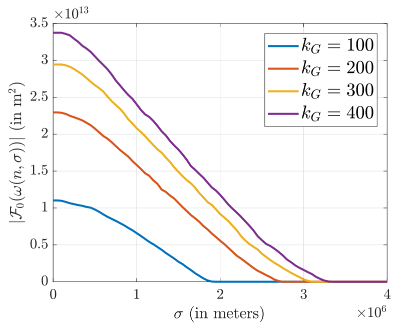
$$\mathbb{P}(|\Phi \cap \mathcal{F}_\epsilon(\mathbf{x})| \ge \nu) \ge 1 - \alpha, \tag{17}$$

i.e., if the probability that a randomly located game involving n players with a geographical spread less than σ finds at least ν compute nodes that would make the game ϵ-feasible exceeds $1 - \alpha$.

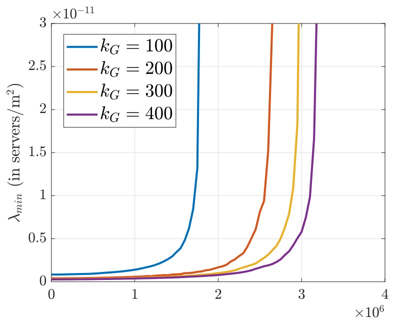
More formally, the GSP wishes to solve the network resource provisioning problem described as follows:

Problem 2: (Network Resource Provisioning). *The network resource provisioning problem consists in finding the smallest density of compute nodes λ guaranteeing an $(n, \sigma, \epsilon, \alpha, \nu)$-service coverage.*

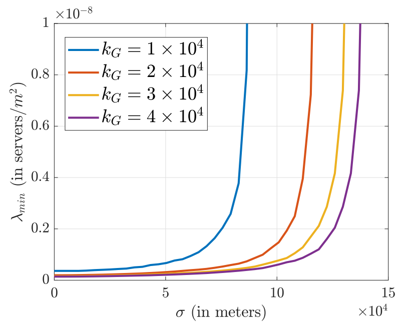
We note that while a single compute node falling inside an ϵ-feasible region may be enough to guarantee that the game is ϵ-playable, the GSPs may want to provision the network resources so as to ensure that multiple compute node ($\nu > 1$) fall within the region for three reasons. First, such a choice increases load-balancing flexibility across compute nodes in the network, which has been shown to improve the service availability in large networks [18]. Second, it provides additional guarantees in case some feasible servers are unable to support additional MCG instances due to limited resources. Third, it improves the robustness to variability in the transport delays experienced by the players, that may trigger costly migrations of the G-server's VM, more on this in Section 5.



(a) Area of the ϵ-feasible region vs. the players' geographical spread σ; $n = 20$, $a_0 = 50$ms, $\tau = 20$ms, $\epsilon = 0$.



(b) Smallest feasible server density $\lambda_{\min}$ vs. the players' geographical spread σ; $n = 20$, $a_0 = 50$ms, $\tau = 20$ms, $\epsilon = 0$, $\alpha = 1 \times 10^{-4}$.



(c) Smallest feasible server density $\lambda_{\min}$ vs. the players' geographical spread σ; $n = 20$, $a_0 = 5$ms, $\tau = 3$ms, $\epsilon = 0$, $\alpha = 1 \times 10^{-4}$.

Figure 4: Figures of the impact of the geographical spread σ and the per-user compute capacity k_G on the area of the ϵ-feasible region of an n player regular configuration, and the induced minimum density $\lambda_{\min}$ of compute nodes required to guarantee $(n, \sigma, \epsilon, \alpha, 1)$-service coverage, in traditional MCG and XR-MCG settings. For scale comparison, the distance from New York City, NY to Los Angeles, CA is on the order of 4×10^6 meters, while the area of the USA is on the order of 1×10^{13} square meters.

We now state an important result, enabling the GSPs to solve the network provisioning problem, defined in Problem 2, giving a lower bound on $|\mathcal{F}_\epsilon(\mathbf{x})|$ for any configuration of a given spread.

Theorem 1: (Lower-Bound on the ϵ-Feasible Area). *Let $\mathbf{x} \in \mathbb{R}^{n\times 2}$ be any configuration of n players with geographic spread $\sigma(\mathbf{x})$. Under the JMRA algorithm, we have $\forall \epsilon \in [0,1]$:*

$$|\mathcal{F}_\epsilon(\mathbf{x})| \geq |\mathcal{F}_\epsilon(\boldsymbol{\omega}(n, \sigma(\mathbf{x})))|. \tag{18}$$

A proof of this theorem is found in Appendix A. An intuitive interpretation is that regular configurations are the most "spread-out" among the class of n players configurations with a geographical spread equal to $\sigma(\mathbf{x})$, hence leading to the smallest ϵ-feasible region.

We now present a corollary of Theorem 1, allowing the GSPs to solve the network resource provisioning problem. For tractability, we shall present the case where $\nu = 1$.

Corollary 1: (Smallest Server Density). *Let $\Phi(\lambda)$ be a homogeneous PPP of intensity λ compute nodes/m^2. The smallest compute node density λ_{min} required to guarantee an $(n, \sigma, \epsilon, \alpha, 1)$-service coverage, for $n \in \mathbb{N}, \sigma \in \mathbb{R}_+, \epsilon \in [0,1], \alpha \in [0,1]$ is given by:*

$$\lambda_{min}(n, \sigma, \epsilon, \alpha) = \frac{-\ln(\alpha)}{|\mathcal{F}_\epsilon(\boldsymbol{\omega}(n,\sigma))|} \tag{19}$$

The proof directly follows from Theorem 1 and the fact that for a PPP, see [6]: $\mathbb{P}(|\Phi(\lambda) \cap \mathcal{F}_\epsilon(\boldsymbol{\omega}(n,\sigma))| \geq 1) = 1 - e^{-\lambda|\mathcal{F}_\epsilon(\boldsymbol{\omega}(n,\sigma))|}$.

Now that we established the optimal strategy for GSPs to densify the network, we study through numerical simulations how they should dimension the G-server VMs' compute capacity. Specifically, Figure 4 exhibits results capturing the effects of the geographical spread σ and the per-player compute capacity k_G on the area of the ϵ-feasible region corresponding to a regular configuration, and ultimately, on the required $\lambda_{\min}$. The results presented correspond to two different scenarios. In Figures 4a and 4b, we use parameters relevant to classical MCG instances, e.g., involving players interacting on a common virtual first-person shooter game, requiring a somewhat loose timeliness constraints (on the order of 100 milliseconds end-to-end [11], or around 50 milliseconds for the player-to-server leg). However, Figure 4 corresponds to an XR-MCG game setting, where players are equipped with XR headsets, requiring much tighter timeliness guarantees (on the order of 10 millisecond end-to-end [14], or around 5 milliseconds for the player-to-server leg). We study both scenarios separately. In these experiments, we adopt the functional forms introduced in Equations 12 and 13 to model the transport and batch computation delays.

The General MCG Setting. One can first observe in Figure 4a that the area of the ϵ-feasible region decreases with the players' geographical spread σ. This effect leads in turn to a sharp increase in the required density $\lambda_{\min}$, see Figure 4b, to compensate for the reduced area. This rapid increase is explained by the fact that $|\mathcal{F}_\epsilon(\boldsymbol{\omega}(n,\sigma))|$ eventually vanishes as the players become too widely spread, leading to the vertical asymptotes shown in Figure 4b. Therefore, for a fixed capacity per G-server's VM instance, we witness a geographical spread limit after which densification can no longer help in guaranteeing $(n, \sigma, \epsilon, \alpha, \nu)$-service coverage. Supporting larger spreads can then only be achieved by increasing the servers' compute capacity. One direct implication of this observation is that GSPs that can perform efficient matchmaking, i.e., match players in close proximity of each other, can afford to reduce the servers' compute capacity k_G, while keeping the server density reasonably low. In addition, we recognize in Figure 4b a law of diminishing returns on feasible σ with increasing k_G, pointing to the existence of a fundamental limit on the maximum geographical spread that can be supported for any n-player game, regardless of the network resources deployed and rate adaptation policy, due to the sole impact of the transport delay on the age of the game, see Equation 1.

We note that while the initial model proposed in this paper does not capture this effect, servers and players are in reality likely to be more densely located in cities. When the player's geographical spread is small enough, e.g., games involving players in the same city, then the GSPs can afford to provision compute nodes mostly in cities as per Figure 4, and the G-server would be placed nearby the players' city. If, however, GSPs want to support games with higher geographic spread, e.g., involving players across different cities, then they may need to densify the compute nodes between the cities, in the associated ϵ-feasible regions that is intuitively close to "center" of the players' configuration.

The XR-MCG Setting. Comparing Figure 4b to Figure 4c, one can highlight three key challenges faced by GSPs with extremely tight timeliness constraints, e.g., supporting XR-MCG. The first challenge is the need to ensure that the compute delay is as small as possible. To this end, the compute capacity per player k_G needs to be large enough to guarantee that the constraint in Problem 1 can be satisfied. Hence, XR-MCG instances require additional compute capacity compared to traditional MCG games.

The second challenge is the need to ensure that the players' geographical spread is small such that all the players are close enough to the G-server, leading to low transport delays. This is reflected by the scale of the horizontal axis, showing that XR-MCG instances can only be supported by connecting local players, e.g., in the same neighborhood/city, as opposed to the country/continent scale for traditional MCG instances.

The third challenge is the need to heavily densify the network to ensure small transport delays (and hence low variability under Assumption 1) so as to meet the service coverage requirement with a tight game age. The required density is on the order of $10^{-9} - 10^{-8}$ compute nodes per square meter, which clearly calls for leveraging the edge computing infrastructure to host the G-servers, in addition to the (potentially colocated) R-servers. Hefty network resource provisioning costs stemming from allocating considerable compute power in densely deployed edge compute nodes are unavoidable for XR-MCG GSPs to meet the tight game age constraint associated with such types of applications.

5 MCG NETWORK MANAGEMENT

In Section 4, we showed how GSPs can ensure high service coverage by appropriately provisioning the network resources. We now assume that these resources have been provisioned, and we use insights extracted from our previous analysis to investigate strategies that can be adopted by GSPs to improve the MCG-QoS.

We study the particular problem of G-server placement, consisting in selecting the best compute node to host the G-server's VM among a set of feasible options given a players' configuration, see Figure 3. Previously, we showed how the JMRA algorithm can help to ensure that the spatial region that may contain feasible servers has the largest area. This region is likely to contain multiple compute nodes, all of them satisfying the game QoS requirement. While this may imply that all of the options are equivalent in the framework formulated in this paper, additional considerations such as robustness to network delay variability may motivate the GSPs to prefer some options over others. We now investigate how the GSPs might go about selecting the G-servers' VMs locations to improve the robustness of the MCG-QoS under varying network delays/congestion statistics. This is an important consideration, as one can expect MCG sessions to potentially last several hours.

We first observe that JMRA can increase robustness to slow variations in network/congestion delays over time as the optimal choice of update rate can adapt to such variations. However, this reactivity feature of JMRA may not be sufficient to keep the game ϵ-playable under significant variations, or if the players are mobile. Indeed, a change in the joint delay statistics experienced by the players may induce a substantial change in the shape of the region $\mathcal{F}_\epsilon(\mathbf{x})$ causing a potential need to trigger a costly G-server VM migration. Therefore, given the opportunity to select a server among multiple feasible options, a simple strategy would be to select the one that maximizes the expected value of the MCG-QoS, as it would keep the game ϵ-playable under the largest delay variations. Hence, the GSPs might maximize a new service placement MCG-QoS:

Definition 7: (MCG-QoS for Service Placement). *Given a player configuration $\mathbf{x} \in \mathbb{R}^{n\times 2}$, and a feasible G-server location $\mathbf{g} \in \mathbb{R}^2$, inducing a distance vector $\boldsymbol{\delta} \in \mathbb{R}^n_+$, s.t. $\delta_i = \|\mathbf{x}_i - \mathbf{g}\|_2$, we define the MCG-QoS $\bar{q}(\boldsymbol{\delta})$ for service placement, for a given $\epsilon \in [0,1]$, as:*

$$\bar{q}(\boldsymbol{\delta}) = \mathbb{P}(q(\mathbf{D}^t_{\boldsymbol{\delta}}) > 1 - \epsilon) \tag{20}$$

i.e., the probability that the game remains ϵ-playable under JMRA and variable network congestion statistics.

Based on this MCG-QoS, we formally define the service placement problem as follows:

Problem 3: (Service Placement). *Given a player configuration $\mathbf{x} \in \mathbb{R}^{n\times 2}$ and a realization ϕ of the spatial server deployment Φ, inducing a set $\mathcal{G}(\mathbf{x},\phi) = \{\mathbf{g}_1, \ldots, \mathbf{g}_l\}$ of l ϵ-feasible server locations, the service placement problem consists in finding server $\mathbf{g}^*(\mathbf{x},\phi)$, s.t.:*

$$\mathbf{g}^*(\mathbf{x},\phi) = \arg \max_{\mathbf{g}_k \in \mathcal{G}(\mathbf{x},\phi)} \left\{ \bar{q}(\boldsymbol{\delta}) \; : \delta_i = \|\mathbf{x}_i - \mathbf{g}_k\|_2, \forall i \right\} \tag{21}$$

A straightforward way to solve Problem 3 would be to compute the MCG-QoS for service placement assuming that each of the candidate servers is selected to host the G-server, and choose the one that maximizes it. However, computing the MCG-QoS function may be impractical and computationally expensive as it involves solving numerous optimization problems, and laboriously estimate the distribution of $\bar{q}(\boldsymbol{\delta})$ through advanced sampling techniques.

To overcome this issue, we envision a 3-step algorithm, that can run in a centralized server, and that is aware of all the players' locations and the map of compute nodes:

Step 1: Exploration. First, one needs to identify the search space $\mathcal{G}(\mathbf{x},\phi)$ of candidate servers. This can be performed either by considering all the compute nodes within a vast region containing all the servers "within reach" of any player, i.e., such that the transport delay does not exceed the age constraint. As this solution would likely lead to an excessively large search space, heuristics can be leveraged to restrict the set to candidate servers, e.g., considering the l closest servers to the center of mass of configuration $\mathbf{x}$.

Step 2: Elimination. Second, one can considerably simplify the search space by only using the geometry of the players' configuration, as presented in Theorem 2.

Theorem 2: (Preferred G-Server Location). *Given a player configuration $\mathbf{x} \in \mathbb{R}^{n\times 2}$ and a compute node deployment ϕ, let $\mathbf{g}$ and $\mathbf{g}' \in \mathcal{G}(\mathbf{x},\phi)$ be the coordinates of two servers inducing distance vectors $\boldsymbol{\delta}$ and $\boldsymbol{\delta}'$, respectively. We have under Assumption 1:*

$$\boldsymbol{\delta} \prec_w \boldsymbol{\delta}' \implies \bar{q}(\boldsymbol{\delta}) \geq \bar{q}(\boldsymbol{\delta}') \tag{22}$$

hence the server at location $\mathbf{g}$ is to be preferred over the one at $\mathbf{g}'$.

where $\prec_w$ denotes the weak majorization ordering, see [19]. The proof of this theorem can be found in Appendix B.

Using this result, some of the candidate servers can be eliminated in $O(l^2)$ time only by inspecting the distance vectors induced by the players' configuration $\mathbf{x}$ and each potential server in $\mathcal{G}(\mathbf{x}, \phi)$. We note however that weak majorization is merely a partial order, hence not any pair of distance vectors can be compared and this procedure does not guarantee to single out the best candidate server. In such a case, the algorithm needs to execute Step 3.

Step 3: Approximation. Third, once the number of candidate servers has been reduced to only a few candidates, additional heuristics can be exploited to select the final server. For instance, $q(\mathbf{d}^t)$ can be used as a surrogate for $\bar{q}(\boldsymbol{\delta})$, where $\mathbf{d}^t$ can be measured/estimated as discussed in Section 3. Finally the best compute node is confirmed if its MCG-QoS function exceeds the desired level ϵ.

We assess the performance of the elimination step by studying the effect of the size of the search space l and the number of players n on the average number of survivors (i.e., options that were not eliminated in step 2), for a fixed players' geographical spread and density of servers. The average is taken over random player configurations of given spread, and over realizations of Φ. Clearly, values close to 1 are associated with an effective elimination. In this experiment, we initialize $\mathcal{G}(\mathbf{x}, \phi)$ to contain the l closest compute nodes in ϕ to the center of gravity of the configuration $\mathbf{x}$, as suggested in Step 1. We present the results of this experiment in Figure 5.

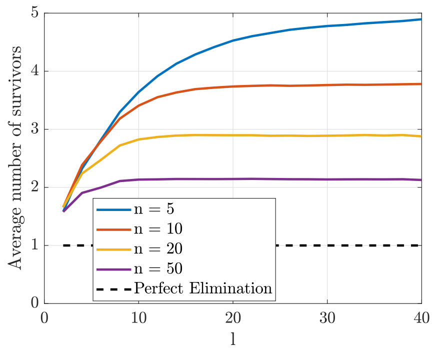


Figure 5: Effect of the size of the search space l and the number of players n on the average number of survivors; $\sigma = 2 \times 10^6$m, $\lambda = 4 \times 10^{-12}$ servers/m^2.

One can observe that the average number of survivors increases slowly with l, as additional options are increasingly more likely to be eliminated. This confirms that proximity to the center of gravity of the player's configuration is a valid criterion to initialize the search space. In addition, the elimination step appears to be the most effective in games involving a large number of players. Indeed, larger values of n lead to a *hardening* of the spatial distribution of players, homogenizing it over a disk of radius σ, and bringing the center of gravity closer to the center of this disk. This in turn increases the likelihood for any sub-optimal server to find at least one of the players being prohibitively far, hence making it more likely to be eliminated as its associated distance vector will weakly majorize the ones of servers that are closer to the center of gravity.

6 CONCLUSION

In this paper, we studied fundamental questions that arise in the design of MCG systems. We introduced an MCG-QoS capturing the freshness of the information processed by the G-server, as well as the joint impact of the variable delays experienced by the players. We proposed JMRA, an efficient measurement-based joint update rate adaptation algorithm maximizing the MCG-QoS. We then related the game's geometry to the network delays experienced by the players, and showed how GSPs can benefit from JMRA to combat the effect of geographical spread and slow-variability in the network delays/congestion, through effective network resource provisioning and service placement. We note that MCG player matchmaking, i.e., finding the best set of players to match on the same G-server, is a network operation problem that is complementary to service placement, and is also worth studying. Given the analytical framework presented in this paper, the matchmaking problem might be reduced to a clustering problem aiming at finding the partition of players that minimizes the geographical spread of the induced configurations. We intend to study this problem in future work.

APPENDIX

A Proof Theorem 1:

Proof. In this proof, we restrict our attention to a region $\mathcal{R}(\mathbf{x}, a_0)$, where $\mathcal{R}(\mathbf{x}, a_0) = \{\mathbf{y} \in \mathbb{R}^2 : \|\mathbf{y}\|_2 \leq \eta + \sigma(\mathbf{x})\}$, and η is such that $d^t(\eta) = a_0$. Defining $\mathcal{R}(\mathbf{x}, a_0)$ in this way leads to the following observation: $\mathbf{g} \notin \mathcal{R}(\mathbf{x}, a_0) \implies q(\mathbf{x}, \mathbf{g}) < 1 - \epsilon, \forall \mathbf{x} \in \mathbb{R}^{n\times 2}, \forall \epsilon \in [0, 1]$. Therefore, $\mathcal{F}_\epsilon(\mathbf{x}) \subset \mathcal{R}(\mathbf{x}, a_0)$, allowing us to study it by only considering points in $\mathcal{R}(\mathbf{x}, a_0)$. We now prove a useful lemma.

Lemma 1: (Stochastic Majorization of Max Distance). *Let* $\mathbf{x}$ *and* $\mathbf{x}' \in \mathbb{R}^{n\times 2}$ *be any two configurations of* n *players, where* $\sigma(\mathbf{x}) \geq \sigma(\mathbf{x}')$*, and let* $\mathbf{G}$ *be a random G-server coordinate vector uniformly distributed on* $\mathcal{R}(\mathbf{x}, a_0)$*. Define* $\boldsymbol{\Delta}, \boldsymbol{\Delta}' \in \mathbb{R}_+^n$ *to be the random vectors of induced distances between* $\mathbf{G}$ *and each point in* $\mathbf{x}$ *and* $\mathbf{x}'$*, respectively.*

If $\max_i \Delta_i \leq^{st} \max_i \Delta_i'$*, then under the JMRA algorithm*

$$|\mathcal{F}_\epsilon(\mathbf{x})| \geq |\mathcal{F}_\epsilon(\mathbf{x}')|, \forall \epsilon \in [0, 1]. \tag{23}$$

Proof. We start this proof by noting that

$$\begin{aligned} |\mathcal{F}_\epsilon(\mathbf{x})| &= \iint_{\mathcal{R}(a_0,\mathbf{x})} \mathbb{1}\{q(\bar{\mathbf{d}}^t(\mathbf{x}, \mathbf{g})) > 1 - \epsilon\} d\mathbf{g} \\ &= |\mathcal{R}(a_0, \mathbf{x})| \cdot \mathbb{E}_\mathbf{G}[\mathbb{1}\{q(\bar{\mathbf{d}}^t(\mathbf{x}, \mathbf{G})) > 1 - \epsilon\}] \end{aligned}$$

Similarly, $|\mathcal{F}_\epsilon(\mathbf{x}')| = \mathbb{E}_\mathbf{G}[\mathbb{1}\{q(\bar{\mathbf{d}}^t(\mathbf{x}', \mathbf{G})) > 1-\epsilon\}]$. Hence $|\mathcal{F}_\epsilon(\mathbf{x})| \geq |\mathcal{F}_\epsilon(\mathbf{x}')|, \forall \epsilon \in [0, 1] \iff \mathbb{E}_\mathbf{G}[\mathbb{1}\{q(\bar{\mathbf{d}}^t(\mathbf{x}, \mathbf{G})) > 1 - \epsilon\}] \geq \mathbb{E}_\mathbf{G}[\mathbb{1}\{q(\bar{\mathbf{d}}^t(\mathbf{x}, \mathbf{G})) > 1 - \epsilon\}], \forall \epsilon \in [0, 1]$.

Furthermore, we note that $\mathbb{1}\{q(\mathbf{d}^t) > 1 - \epsilon\} = \mathbb{1}\{\max_\rho \left\{\mathbb{P}(A_{\mathbf{D}^t_\delta, \rho} \leq a_0 \mid \mathbf{D}^t_\delta = \mathbf{d}^t) \ : \ d^c(\sum_j \rho_j) \leq \tau\right\} > 1 - \epsilon\}$ is a symmetric function of the delay vector $\mathbf{d}^t$, see Equation 5, and decreasing in each of the components of this random vector. Besides, the indicator function returns non-negative values, less than or equal to 1. Therefore, $\mathbb{1}\{q(\mathbf{D}^t_\Delta) > 1 - \epsilon\}$ is a symmetric joint survival function of the random delay vector $\mathbf{D}^t_\Delta$, hence of the random distance vector $\boldsymbol{\Delta}$. Now we have:

$$\begin{aligned} &\max_i \Delta_i \leq^{st} \max_i \Delta_i' \\ &\iff \mathbb{P}(\max_i \Delta_i \leq t) \geq \mathbb{P}(\max_i \Delta_i' \leq t), \forall t \in \mathbb{R} \\ &\iff \mathbb{P}(\Delta_1 \leq t, \cdots, \Delta_n \leq t) \geq \mathbb{P}(\Delta_1' \leq t, \cdots, \Delta_n' \leq t), \forall t \in \mathbb{R} \\ &\iff \max_i \Delta_i \leq^{slo} \max_i \Delta_i' \end{aligned}$$

$\iff \mathbb{E}[\psi(\mathbf{\Delta})] \geq \mathbb{E}[\psi(\mathbf{\Delta}')], \forall \psi \in C.$

where C is the class of symmetric joint survival functions. The definition of the *symmetric lower orthant* ordering and its properties can be found in [20, 21]. The result follows from the fact that $\mathbb{1}\{q(\bar{\mathbf{d}}^t(\mathbf{x}, \mathbf{G})) > 1-\epsilon\} \in C, \forall \epsilon \in [0,1]$. □

We now proceed to prove the theorem. The proof is subdivided in two parts: we first show that for any player configuration $\mathbf{x}$ in a disk of radius $\sigma(\mathbf{x})$ moving the players to the boundary of the disk reduces $|\mathcal{F}_\epsilon(\mathbf{x})|$; we then show that equispacing the players on the boundary of the disk minimizes this area.

Part 1: Equalizing the radial coordinate components. In this part, we construct a coupling between any configuration of players $\mathbf{x}$, of geographical spread $\sigma(\mathbf{x})$ and the configuration $\mathbf{x}'$ of players having the same polar angular coordinates, as in $\mathbf{x}$, but all the polar radial coordinate components equal to $\sigma(\mathbf{x})$, i.e., all the players are located on the boundary of the circle centered at the origin and of radius $\sigma(\mathbf{x})$. We observe that under configuration $\mathbf{x}'$ region $\mathcal{R}(\mathbf{x}, a_0)$ can be partitioned into n sectors, where sector $\mathcal{R}'_k(\mathbf{x}, a_0)$ is defined to be the region of points such that player k is the furthest player, or equivalently, $\mathcal{R}'_k(\mathbf{x}, a_0) = \{\mathbf{g} \in \mathcal{R}(\mathbf{x}, a_0) : \arg\max_i \Delta'_i = k\}$. Similarly, we define $\mathcal{R}_k(\mathbf{x}, a_0) = \{\mathbf{g} \in \mathcal{R}(\mathbf{x}, a_0) : \arg\max_i \Delta_i = k\}$. Since no adjacent players are separated by an angle larger than π, by construction of the circle of radius $\sigma(\mathbf{x})$ to be the circle of smallest radius encompassing all the players, it is clear that $\langle \mathbf{x}_k, \mathbf{g}\rangle \leq 0, \forall \mathbf{g} \in \mathcal{R}_k(\mathbf{x}, a_0), \forall k$ and $\langle \mathbf{x}'_k, \mathbf{g}\rangle \leq 0, \forall \mathbf{g} \in \mathcal{R}'_k(\mathbf{x}, a_0), \forall k$. Now we have: $\max_i \delta_i = \delta_k = \|\mathbf{x}_k - \mathbf{g}\|_2 = \sqrt{\|\mathbf{x}_k\|_2^2 + \|\mathbf{g}\|_2^2 - 2\langle \mathbf{x}_k, \mathbf{g}\rangle} \leq \sqrt{\|\mathbf{x}'_j\|_2^2 + \|\mathbf{g}\|_2^2 - 2\langle \mathbf{x}'_j, \mathbf{g}\rangle} = \delta'_j = \max_i \delta'_i$, where the inequality follows from the facts that $\|\mathbf{x}_k\|_2 \leq \|\mathbf{x}'_j\|_2$, $\langle \mathbf{x}_k, \mathbf{g}\rangle \leq 0$, $\langle \mathbf{x}'_j, \mathbf{g}\rangle \leq 0$, and the angle between $\mathbf{x}_k$ and $\mathbf{g}$ being equal to the one between $\mathbf{x}'_j$ and $\mathbf{g}$, by construction. Therefore, for any realization $\mathbf{g} \in \mathcal{R}(\mathbf{x}, a_0)$ we have $\max_i \delta_i \leq \max_i \delta'_i$, thus $\max_i \Delta_i \leq \max_i \Delta'_i$, almost surely. It follows that $\max_i \Delta_i \leq^{\text{st}} \max_i \Delta'_i$, hence we get from Lemma 1, $|\mathcal{F}_\epsilon(\mathbf{x})| \geq |\mathcal{F}_\epsilon(\mathbf{x}')|, \forall \epsilon \in [0,1]$.

Part 2: Equalizing the angular coordinate components. In this part, we prove that for any configuration $\mathbf{x}$ such that all the players are on the boundary of a circle of radius $\sigma(\mathbf{x})$, spacing the players regularly on the boundary minimizes $|\mathcal{F}_\epsilon(\mathbf{x})|$. In this setting, the player configuration can be parametrized by $\boldsymbol{\theta}$ the vector of differential angles between adjacent players on the circle.

We start by deriving an expression for $\mathbb{P}(\max_i \Delta_i \geq t | \|\mathbf{G}\|_2, \boldsymbol{\theta})$, the conditional c.d.f. of $\max_i \Delta_i$, given $\|\mathbf{G}\|_2 = r$ and parametrized by $\boldsymbol{\theta} \in [0, 2\pi]^n$, where $\sum_i \theta_i = 2\pi$. One can show that:

$$\mathbb{P}(\max_i \Delta_i \geq t | \|\mathbf{G}\|_2 = r, \boldsymbol{\theta}) = \frac{\sum_k \min[\gamma(t,r), \theta_k/2]}{\pi} \tag{24}$$

where $\gamma(t,r) = \pi - \cos^{-1}(\max[\min[\frac{\sigma(\mathbf{x})^2 + r^2 - t^2}{2r\sigma(\mathbf{x})}, 1], -1])$.

We observe that $\mathbb{P}(\max_i \Delta_i \geq t | \|\mathbf{G}\|_2 = r, \boldsymbol{\theta})$ is symmetric and concave in $\boldsymbol{\theta}$, it is therefore Schur-concave in $\boldsymbol{\theta}$. Let $\boldsymbol{\theta}'$ parametrize the equispaced configuration, i.e., $\theta'_i = \frac{2\pi}{n}, \forall i$, then clearly $\boldsymbol{\theta}' \prec \boldsymbol{\theta}, \forall \boldsymbol{\theta} \in [0, 2\pi]^n$, where $\sum_i \theta_i = 2\pi$. We say that $\boldsymbol{\theta}'$ is *majorized* by $\boldsymbol{\theta}$. Therefore, from Schur-concavity, we have $\mathbb{P}(\max_i \Delta'_i \geq t | \|\mathbf{G}\|_2 = r, \boldsymbol{\theta}') \geq \mathbb{P}(\max_i \Delta_i \geq t | \|\mathbf{G}\|_2 = r, \boldsymbol{\theta}), \forall r$, which implies that $\mathbb{P}(\max_i \Delta'_i \geq t) \geq \mathbb{P}(\max_i \Delta_i \geq t), \forall \boldsymbol{\theta} \in [0, 2\pi]^n$, by integrating over all values of r so as to span $\mathcal{R}(\mathbf{x}, a_0)$. It follows that $\max_i \Delta_i \leq^{\text{st}} \max_i \Delta'_i$, hence we get from Lemma 1, $|\mathcal{F}_\epsilon(\mathbf{x})| \geq |\mathcal{F}_\epsilon(\mathbf{x}')|, \forall \epsilon \in [0,1]$. □

B Proof of Theorem 2

Proof. We know that $\bar{q}(\boldsymbol{\delta}) = \mathbb{P}(q(\mathbf{D}^t_{\boldsymbol{\delta}}) > \epsilon)$ is a Schur-concave function in $\boldsymbol{\delta}$ as $q(\mathbf{d}^t)$ is Schur-concave in $\mathbf{d}^t$, see [19]. The Schur-concavity property of $q(\mathbf{d}^t)$ directly follows from the fact that the function is symmetric in the entries of $\mathbf{d}^t$, and concave in $\mathbf{d}^t$, see section 3.2.5 in [7]. In addition, from Assumption 1, we know that $\forall \mathbf{z} \in \mathbb{R}^n_+$, $\mathbf{D}^t_{\boldsymbol{\delta}} \leq^{\text{icx}} \mathbf{D}^t_{\boldsymbol{\delta}+\mathbf{z}}$. Since $q(\mathbf{d}^t)$ is decreasing and concave in $\mathbf{d}^t$, we have $\bar{q}(\boldsymbol{\delta}) = \mathbb{P}(q(\mathbf{D}^t_{\boldsymbol{\delta}}) > 1-\epsilon) \geq \mathbb{P}(q(\mathbf{D}^t_{\boldsymbol{\delta}+\mathbf{z}}) > 1-\epsilon) = \bar{q}(\boldsymbol{\delta}+\mathbf{z})$, i.e., $\bar{q}(\boldsymbol{\delta})$ is decreasing in $\boldsymbol{\delta}$. Therefore, $\bar{q}(\boldsymbol{\delta})$ is a Schur-concave decreasing function in $\boldsymbol{\delta}$, thus given $\boldsymbol{\delta}$ and $\boldsymbol{\delta}' \in \mathbb{R}^n_+$ be two distance vectors induced by two feasible game servers, $\boldsymbol{\delta} \prec_w \boldsymbol{\delta}' \implies \bar{q}(\boldsymbol{\delta}) \geq \bar{q}(\boldsymbol{\delta}')$, as argued in [19]. □

ACKNOWLEDGMENTS

This work was made possible by the support of Fujitsu Networks Communications, and the NSF Awards CNS-1731658 and ECC-1809327.

REFERENCES

[1] 2020. Amazon Luna. Retrieved Nov. 15, 2020 from https://www.amazon.com/luna/landing-page Accessed Nov. 15, 2020.
[2] 2020. Google Stadia. Retrieved Sep. 15, 2020 from https://stadia.google.com/
[3] 2020. Nvidia GeForce. Retrieved Sep. 15, 2020 from https://www.nvidia.com/en-us/geforce-now/
[4] 2020. PlayStation Now. Retrieved Sep. 15, 2020 from https://www.playstation.com/en-us/explore/playstation-now/
[5] 2020. Project xCloud. Retrieved Sep. 15, 2020 from https://www.xbox.com/en-US/xbox-game-pass/cloud-gaming/home
[6] F. Baccelli and B. Błaszczyszyn. 2010. *Stochastic geometry and wireless networks.* Vol. 1. Now Publishers Inc.
[7] S. Boyd and L. Vandenberghe. 2004. *Convex optimization.* Cambridge university press.
[8] W. Cai et al. 2016. A Survey on Cloud Gaming: Future of Computer Games. *IEEE Access* (2016).
[9] Y. Chen, J. Liu, and Y. Cui. 2016. Inter-player Delay Optimization in Multiplayer Cloud Gaming. In *IEEE CLOUD 2016.*
[10] S. Chuah, C. Yuen, and N. Cheung. 2014. Cloud gaming: a green solution to massive multiplayer online games. *IEEE Wireless Communications* (2014).
[11] M. Claypool and K. Claypool. 2006. Latency and player actions in online games. *Commun. ACM* (2006).
[12] Y. Deng et al. 2018. The Server Allocation Problem for Session-Based Multiplayer Cloud Gaming. *IEEE Transactions on Multimedia* (2018).
[13] E. Dhib et al. 2016. Modeling Cloud gaming experience for Massively Multiplayer Online Games. In *2016 13th IEEE Annual CCNC.*
[14] M. S. Elbamby et al. 2018. Toward Low-Latency and Ultra-Reliable Virtual Reality. *IEEE Network* (2018).
[15] Y. Gao, L. Wang, and J. Zhou. 2019. Cost-Efficient and Quality of Experience-Aware Provisioning of Virtual Machines for Multiplayer Cloud Gaming in Geographically Distributed Data Centers. *IEEE Access* (2019).
[16] H. Hong et al. 2015. Placing Virtual Machines to Optimize Cloud Gaming Experience. *IEEE Transactions on Cloud Computing* (2015).
[17] S. P. Kasiviswanathan, S. Eidenbenz, and G. Yan. 2011. Geography-based analysis of the internet infrastructure. In *2011 Proceedings IEEE INFOCOM.*
[18] S. Kassir et al. 2020. Service Placement for Real-Time Applications: Rate-Adaptation and Load-Balancing at the Network Edge. In *2020 7th IEEE CSCloud/2020 6th IEEE EdgeCom.*
[19] A. Marshall, I. Olkin, and B. Arnold. 1979. *Inequalities: theory of majorization and its applications.* Springer.
[20] M. Shaked and G. Shanthikumar. 1997. Supermodular stochastic orders and positive dependence of random vectors. *Journal of Multivariate Analysis* (1997).
[21] M. Shaked and G. Shanthikumar. 2007. *Stochastic orders.* Springer Science & Business Media.
[22] R. D. Yates et al. 2017. Timely cloud gaming. In *IEEE INFOCOM 2017 - IEEE Conference on Computer Communications.*
[23] Y.Lin and H.Shen. 2016. CloudFog: Leveraging fog to extend cloud gaming for thin-client MMOG with high quality of service. *IEEE TPDS* (2016).