Cloud Gaming: A QoE Study of Fast-paced Single-player and Multiplayer Gaming

Sebastian Flinck Lindström[§]
Linköping University
Sweden

Markus Wetterberg[§]
Linköping University
Sweden

Niklas Carlsson
Linköping University
Sweden

***Abstract*—Cloud computing offers an attractive solution for modern computer games. By moving the increasingly demanding graphical calculations (e.g., generation of real-time video streams) to the cloud, consumers can play games using small, cheap devices. While cloud gaming has many advantages and is increasingly deployed, not much work has been done to understand the underlying factors impacting players' user experience when moving the processing to the cloud. In this paper, we study the impact of the quality of service (QoS) factors most affecting the players' quality of experience (QoE) and in-game performance. In particular, these relationships are studied from multiple perspectives using complementing analysis methods applied on the data collected via instrumented user tests. During the tests, we manipulated the players' network conditions and collected low-level QoS metrics and in-game performance, and after each game, the users answered questions capturing their QoE. New insights are provided using different correlation/auto-correlation/cross-correlation statistics, regression models, and a thorough breakdown of the QoS metric most strongly correlated with the users' QoE. We find that the frame age is the most important QoS metric for predicting in-game performance and QoE, and that spikes in the frame age caused by large frame transfers can have extended negative impact as they can cause processing backlogs. The study emphasizes the need to carefully consider and optimize the parts making up the frame age, including dependencies between the processing steps. By lowering the frame age, more enjoyable gaming experiences can be provided.**

***Keywords*-Cloud computing; cloud gaming; QoE; frame age; single-player; multiplayer; gaming; fast-paced; performance**

I. Introduction

Modern computer games demand more and more computing, while consumers want smaller and cheaper devices. A solution to this mismatch is to offload heavy graphical calculations to the cloud. This enables services such as mobile gaming, where the client device only obtains user inputs and displays a real-time game video stream generated by one or more cloud servers.

Cloud computing offers an effective way to deliver high-performance services to players who lack the necessary computation resources. In the context of cloud gaming, this approach has positive effects for both players and developers. For example, the reduced need for high-performance computations on the client side allows players to use cheaper devices and improves battery life. Second, since players only need a "thin client", they can use regular home entertainment devices rather than custom-made gaming devices. This significantly extends the pool of customers that game developers can target. Third, the game-specific code would only need to be stored on the server, reducing piracy risks for the developers [1].

Driven by these advantages, there has been a significant increase in the use of cloud gaming services. The global cloud gaming market is set to generate \$3.2 Billion by 2023 [2]. However, although this approach has many advantages and is increasingly deployed, not much work has been done to understand the factors impacting the gamer's user experience when processing is moved to the cloud.

In this paper, we study what objective quality of service (QoS) factors best explain the players' quality of experience (QoE) and in-game performance, and how each impacts the player experience. To capture these relationships from multiple perspectives and to allow the use of selected, complementing analysis approaches (taking into account the impact of game, player, and win/loss, for example), we designed a series of user-based tests in which two gamers played a fast-paced single-player game and a fast-paced multiplayer game many times under different network conditions. We carefully manipulate the players' network conditions (e.g., end-to-end latencies and packet loss rates) and collect low-level QoS metrics and in-game performance during the different tests. After each game, the users answer questions regarding their QoE, as measured by the Mean Opinion Scores (MOS). In total, the study consists of 325 single-player games and 143 multiplayer games, resulting in over 2,340 (1,625+715) minutes of detailed per-frame measurement data.

New insights are provided using different correlation-based statistics, regression models, and a careful breakdown of the QoS metric most strongly correlated with the users' QoE. To provide context, we first present a brief scenario-based analysis that studies the impact of basic input parameters (e.g., end-to-end latencies and packet loss rates). Second, we characterize and identify the QoS metrics with the most significant impact on QoE metrics and the players' in-game performance. Here, we present an extensive correlation analysis using Pearson, Kendall, and maximal information-based correlation (MIC) [3]. Third, we use multi-factor regression models to provide insights into

[§]S. F. Lindström and M. Wetterberg contributed equally.

how well combined sets of factors explain whether users experience good/bad QoE. Fourth, we study the impact of individual player differences and whether a user won/lost a game. Fifth, we present a detailed analysis of the frame age, defined as the time it takes before the frame is displayed to the user, and the QoS metric that we found best explains QoE on its own. This analysis includes basic regression modeling and an in-depth frame-by-frame analysis in which we look at auto-correlations, cross-correlations, and other dependencies between the processing steps making up the frame age under different conditions. This analysis highlights dependencies and independencies that may be important for both QoE modeling and performance optimizations of future solutions.

Our analysis shows that frame age is the most important QoS metric for determining the players' in-game performance and their QoE; that end-to-end latencies typically impact QoE more than packet losses; that temporary spikes in frame age are often the results of long frame transfers; and that these spikes can have a significant negative impact on QoE long after they first took place as they can cause a backlog in the decoding process. These results emphasize developers' need to consider and optimize the different parts making up the frame age and reduce time dependencies between them to reduce long-drawn negative effects of potential spikes. Our results suggest that developers and providers can ensure that their users enjoy high-quality gaming experiences on basic mobile devices by providing users with low average frame age and minimizing the size, frequency, and duration of frame-age spikes.

Outline: Section II reviews related work. Section III presents our methodology. The following four sections present our scenario-based analysis (Section IV), QoE modeling using application-specific QoS metrics (Section V), an analysis of the impact of non-QoS factors (Section VI), and a frame-by-frame analysis of the different parts of the frame age (Section VII). Finally, Section VIII presents conclusions.

II. Related Work

The works most closely related to ours can broadly be classified into two categories: (i) user studies measuring the QoE under different network conditions, and (ii) system evaluations and optimizations targeting application-specific QoS metrics.

User studies measuring QoE: Several works have studied the effects of latency on gaming, either regular or cloud-based [4]–[7]. For example, Jarschel et al. [8] present a subjective user study of QoE showing that users are more tolerant to packet loss when playing fast-paced games than slower games such as role-playing games. Clincy and Wilgor [7] find that the games become almost unplayable at a packet loss of 1%. Claypool and Finkel [5] study the effects of latency on player performance in the context of arcade-type games like "Crazy Taxi". By measuring player scores and subjective opinion scores, they show that player performance degrades almost linearly with increasing latencies. Others have studied the impact of latencies in first-shooter games [6], with findings suggesting that a round-trip-time (RTT) less than 150 ms is required for a good gaming experience. Raeen and Petlund [4] studied the impact of delays within the local systems on the gaming experience, but did not measure players' in-game performance or the impact of network effects and/or distributed computing. Others have considered how performance relates to QoS [9] and QoE [10], and how performance degrades when delays are introduced [11].

None of the above works presents a comprehensive correlation and auto-correlation analysis of application-specific QoS metrics and the QoE or in-game performance. Instead, they typically only consider the impact of the network parameters under direct control (i.e., RTT and packet loss). Furthermore, we are not aware of any prior work on QoE that provides a breakdown of frame age and its impact on QoE. This is an important aspect, as it is the application-specific QoS metric that most strongly correlates with our QoE metrics and in-game performance. In contrast to most prior work, but similar to Hu et al. [12], we use three different QoE metrics, each providing a different perspective into the users' experience.

System evaluations and optimizations of application-specific QoS: Many works characterize or optimize the performance of these systems. Examples include video encoding optimizations for cloud gaming delivery [13], [14]; virtualized GPU resource scheduling [15], performance benchmarking [16], [17]; the development of a distributed game engine that allows the graphical rendering to be dynamically executed and moved across cloud VMs and client devices [18]; characterizations of the interaction delay and the responsiveness on the cloud-based platform when adding network latencies [19]; measurements to understand the impact of the latencies, packet losses, and bandwidths on frame rates and graphical quality of cloud-based platforms [20], or analyzing the impact that frame rates or graphical quality have on QoE [21]. However, these papers typically do not evaluate the actual QoE or in-game performance. One exception is the work by Hsu et al. [12], who use trace-driven simulations based on a user study in which QoE is measured to demonstrate the effectiveness of their optimized frame-rate and bitrate adaptation algorithms.

Frame age: We are not the first to highlight the importance of frame age. Yates et al. [1] use the recent modeling concept of "age of information" to optimize a cloud gaming system. Although their work is theoretic, they stress the importance of future work (empirically) determining "whether the average frame age is an effective measure of *user-perceived* QoE ...". Our work provides such empirical and data-driven support.

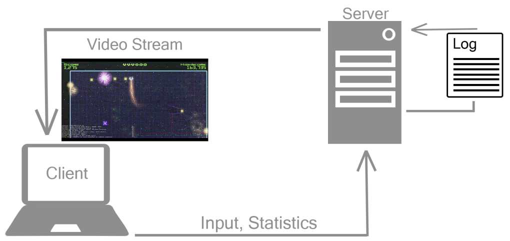

Figure 1: Streaming setup

III. Data Collection Methodology

High-level setup: Figure 1 shows our test setup. Steam Remote Play [22] is installed on both the server and the client. The client runs on a mid-power laptop with integrated graphics (Intel i5 8250U, 8 GB RAM), while the server runs on a high-powered desktop computer (Intel i7 8700K, Nvidia GTX 1080 Ti, 16 GB RAM). At a high level, the client machine captures user inputs, sends these to the server, and displays video streams rendered and delivered frame-by-frame by the server. In addition to rendering video streams, the server also collects and logs statistics (including some client-side statistics) and player performance. Finally, we use Clumsy [23] to introduce artificial network latencies and packet losses. These network parameters allows most practical network conditions to be emulated. In our multiplayer tests, we also use asymmetric (and adaptively assigned) latencies to penalize the better player and achieve more even winning probabilities.

Per-frame QoS metrics and per-game summary statistics: For each game scenario, Steam Remote Play summarizes data such as average bitrate for the client and server, average ping times, average bandwidth usage, and much more. Furthermore, we periodically (every 10 seconds) poll video trace information from the client through Steam Remote Play. These traces contain detailed frame-by-frame information (for the last 10 seconds) and per-second network statistics. The traces are compressed and saved on the server. Table I summarizes the per-frame metrics analyzed here. For our per-game analysis, we also calculate and analyze the mean, median, maximum, minimum, and standard deviation values for each of these metrics.

QoE and in-game performance metrics: At the end of each game, when a player either died or time run out, we measure the user's Mean Opinion Score (MOS). In particular, we ask the user for its subjective opinion on a 7-point scale (1 worst and 7 best) regarding (i) the graphical quality, (ii) the quality of the interactivity, and (iii) their overall opinion score. These three questions were inspired by Hsu et al. [12], although we opted to use seven levels rather than five. We note that both 5 and 7 fall within the criteria for the number of rating levels that a person generally can distinguish between [24] and empathize that a MOS score should not be interpreted as a precise number but as a statistical measurement [25]. We will therefore not compare the absolute values that we obtain against those obtained by Hsu et al. [12]. Finally, we record the in-game score, which the player obtained playing the game, and use as a measure of the player's in-game performance.

Table I: QoS metrics collected and stored by server

Per-second statistics: Ping (network latency), server bandwidth usage, client bandwidth usage, link bandwidth (maximum available), packet loss rate (%)
Per-frame statistics: Frame size, frame age (from creation to when displayed), capture time, encode time, transfer time, decode time, complete time (the wait time until the frame can be displayed)
Per-game statistics: Summary stats (i.e., mean, median, min, max, stdev) for all above metrics, as well as: "dropped slow network" (% frames dropped due to transfer took too long), "dropped lost network" (% lost during transfer), "dropped slow decode" (%), "dropped corrupt decode" (%), "dropped late" (% too long time to display), "dropped reset" (%), and "dropped total" (% frames dropped in total)

Single player tests: The single-player tests were performed using "Geometry Wars: Retro Evolved" [26]. This is a 2D game in which the player plays a small ship inside a larger box where enemies of different sizes and abilities are spawned. The player can move the ship inside the box, earn points by shooting enemies, and can clear the screen of enemies using bombs. (Use of bombs earn the player no points.) The player starts the game with three lives and three bombs. Figure 1 includes a (compressed) game screenshot.

During a test session, the user is first given a chance to familiarize itself with the game. Each user then plays 25 games per session, where each game captures a new set of network conditions.

The first and last game in each session are used as baseline tests. During these games, no additional delays or packet losses are introduced beyond those that occur naturally in our high-speed local area network (LAN). To measure the relative impact of added delays and packets losses on player performance, at the end of each session, the in-game score of the other 23 games of the session are normalized compared to the corresponding average baseline score.

Participants played each game until they ran out of lives or for at most 5 minutes. The time limit ensures that the players' experience is still relevant at the end of the test [27].

Between the two baseline tests, the users go through a series of test scenarios in random order. To capture the impact of end-to-end latencies, we include experiments with eight levels of end-to-end latencies (0, 25, 50, 75, 100, 125, 150, 175, 200 ms) for a series of low-loss scenarios without any additional packet losses and with five latency levels (0, 50, 100, 150, 200 ms) for a series of high-loss scenarios with 1% packet loss rate. Similarly, we include experiments with eight packet-loss levels (0, 0.25, 0.5, 0.75, 1, 1.25, 1.5, 1.75, and 2%) for a series of low-delay scenarios without any additional delay and with five levels of packet-loss rates (0, 0.5, 1, 1.5, and 2%) for a series of high-delay scenario with 100ms end-to-end delay. Except for the baseline scenario (first + last), we do not repeat any experiment scenario twice during a session.

Multiplayer tests: Our multiplayer tests were performed similarly to the single-player tests, but with a few differences. First, two gamers compete against one another. Second, we use the game Speedrunners [28]. Third, we only alter the latency; not the packet losses. Finally, and most importantly, we adapt the network delays for each player on a game-by-game basis so to compensate for skill differences and more fairly balance their win probabilities. We use a simple adaptive algorithm for this purpose. During the first game of a session, both players start with an added latency of 50 ms. Then, after each new match, the delay of the player that won the n ($n \geq 1$) most recent games is increased by a factor $f = x^n$, where x depends on the difference in the score at the end of the match (best of 5), and the losing player has its delay reduced by dividing its current delay with the same factor f. Here, we set $x = 1.2$ if the score was 3-0, $x = 1.15$ if it was 3-1, and $x = 1.1$ if it was 3-2.

Dataset limitations and reproducibility: Our data collection is only performed using a single platform (Steam Remote Play), we only use two example games (both fast-based: one single-player and one multiplayer), and we only performed experiments using two players (that played a lot of games). The choice to use Steam Remote Play is motivated by the desire to collect detailed per-frame information and its popularity as a service. The choices to only use two games and have two players play these games many times (rather than having many players each playing fewer times) were made so to allow us to build and compare more accurate models using four complementing approaches (i) per-player models, (ii) per-game models, (iii) as an aggregate across players, and (iv) using combined multivariate regression models capturing player or game-related differences using categorical variables to capture players, games, and win/loss, for example. This is important for our non-QoS related analysis in Section VI. For improved reproducibility, tools and datasets are made available.[1]In Section VII-A we use detailed system measurements (e.g., memory, disk, bandwidth, and CPU utilization) to validate that system constraints and data collection does not impact our results.

IV. Scenario-based QoE Analysis

To understand the impact of end-to-end latencies and packet losses on QoE and in-game performance, we first present a scenario-based analysis. These results are for the single-player sessions.

Figure 2 shows a box plot with the three QoE metrics (MOS, graphical, and interactive) for all scenarios of consideration. In our box plots, we show the minimum (bottom marker), 25-percentile (bottom of box), median (marker inside box), 75-percentile (top of box), maximum (top marker), outliers (dots), and determine outliers using the commonly used 1.5-IQR rule. If the results from a scenario are displayed a second time (from left-to-right), we use light-grey text in the scenario label. From left-to-right, we show the baseline scenarios (no added delay or losses), eight low-loss scenarios (no added losses) with varying latencies, eight low-latency scenarios (no added latency) with varying packet loss rates, five high-loss scenarios (1% losses) with varying latencies, and five high-delay scenarios (100 ms default latency) with varying loss rates. Figure 3 shows the corresponding plot for the normalized in-game performance score (i.e., the ratio between the in-game score of a game and the average score for the two baseline tests from the same session).

We observe downward trends in the QoE scores and in-game performance for all four sub-groups (for clarity, sub-groups are separated using colors in Figure 3 and separator lines in Figure 2) as the network conditions worsen (i.e., the end-to-end latencies or loss rates increase). However, in general, for the latency and loss rate ranges considered here, we found that the highest latencies (i.e., 200 ms) have a greater negative impact on player performance than the highest packet loss rates (i.e., 2%). This finding matches observations from fast-paced games for which users appear more tolerant to packet losses than to high latencies [29].

Having said that, the impact of packet losses is higher when the delays are high. For example, while we observe limited impact of packet loss for the low-delay scenarios (second group), the negative impact of packet losses is much greater for the high-latency scenarios (fourth group) with a default latency of 100 ms. This is captured by a much sharper decline in QoE scores and in-game performance for this group of scenarios.

In general, the increasing latencies and packet losses provide compounding negative effects. We note that increasing end-to-end latencies still appear to have a bigger negative effect than increasing packet losses also for the most challenging scenarios we consider. For example, the worst-case scenario is the one with 200 ms latency and 1% packet loss rate, rather than 100 ms latency and 2% loss rate. Both these scenarios see as doubling in one of the two parameters compared to the scenario with 100 ms delay and 1% packet loss. In addition to having the worst QoE scores, we can note that the players typically did not reach the 10,000 points limit needed to reach the first in-game milestone for this scenario.

Similar arguments about the relative impact of latencies vs. packet losses can be made by comparing the relative improvements when halving the latency (i.e., 50 ms and 1% loss rate) compared to halving the loss rate (i.e., 100 ms and 0.5% loss rate) relative to the shared high latency/loss case (i.e., 100ms and 1% loss rate). Also, in this case, the change in latency have significantly greater impact on the QoE and in-game performance than the change in loss rate.

[1]Scripts and datasets are made available here: https://www.ida.liu.se/~nikca89/papers/ucc2020.html

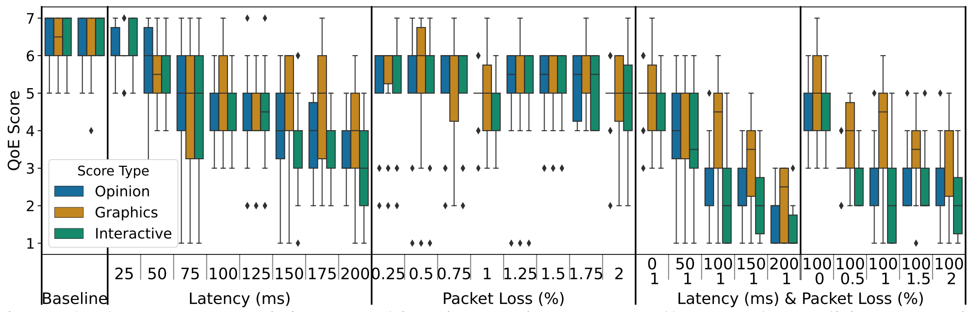

Figure 2: QoE-scores (opinion, graphics, interactive) across all network condition scenarios

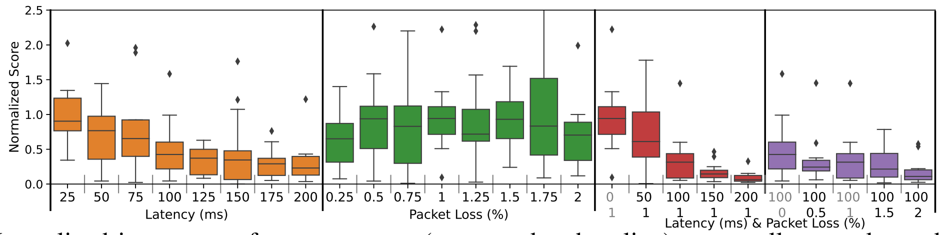

Figure 3: Normalized in-game performance score (compared to baseline) across all network condition scenarios

All of this indicates that the latency impacts the player more than packet loss for the game considered here.

Comparing the QoE metrics, it is also interesting to note that the interactive score deteriorates the most across the scenarios. This result is another indicator of the fast-paced nature of the game. The players appear to notice changes in the responsiveness of the game but do not appear to be as concerned about the visual aspects (i.e., the graphical score). This is in sharp contrast to role-playing games, for which packet losses appear to be a more important factor [8].

Another interesting observation is the drop in the players' scores for scenarios with 100ms or higher latencies. This observation is consistent with the findings by Claypool and Finkel [5], who observed a 25% downgrade when increasing the delay by 100ms. (We observed a 50% downgrade from 100 ms to 200ms.) Finally, when comparing with other works, it should be noted that packet losses appeared to have much smaller impact for us (and Suznjevic et al. [30]) than reported by Clincy and Wilgor [7]. For example, while they observed a significant degradation already for 1%, the QoE and in-game performance remained high even for a 2% loss rate (especially in low-latency scenarios) in our experiments. Again, this difference is most likely due to differences in the nature of the games and how the implementations handle packet losses.

V. QoE Modeling using QoS Metrics

This section presents correlation (Section V-A) and regression modeling (Section V-C) analysis for determining the application-specific QoS metrics best explaining QoE and in-game performance. In this section, we analyze all single-player sessions as an aggregate. Section VI extends our methodology and analysis to account for non-QoS factors such as skill/opinion differences, game differences (single-player vs multiplayer), and whether a player wins/loses.

A. Correlation Analysis

To compare how individual QoS metrics affect the players' QoE and in-game performance, we first performed a correlation analysis. For this analysis we used two traditional correlation metrics (Pearson's r and Kendall's τ) and a more recent technique, called the maximal information coefficient (MIC) [3]. Pearson's r is perhaps the most popular correlation metric. It assumes a linear relationship and Gaussian errors. The other two metrics are less restrictive (in their assumptions) and provide complementing perspectives. Kendall's τ is a rank-based metric that captures the ordinal association between two measures. Both Pearson's r and Kendall's τ use a range [-1,1]. A coefficient close to 1 (or -1) indicates a perfect linear relationship or a perfect monotone relationship, respectively, and random data would have a value of 0. In contrast, MIC uses a range [0,1], only measures how strong the relationship (linear or non-linear) between the two variables is (not the direction or type of relationship), and random data has a MIC score of 0.18.

Figure 4 presents correlation results for the QoS metrics of most interest. For each correlation metric, we include results for all three QoE metrics and the normalized in-game performance. When interpreting this figure, note that Pearson and Kendall can have values within the full range (red-to-blue) but MIC only can have positive values (white-to-blue).

Perhaps the most important finding is the importance of the average frame age. This metric has the strongest correlation of all QoS metrics in 11 out of 12 cases (three correlation metrics × four QoE/performance metrics). Only in the case of the graphical score using MIC does it lose out,

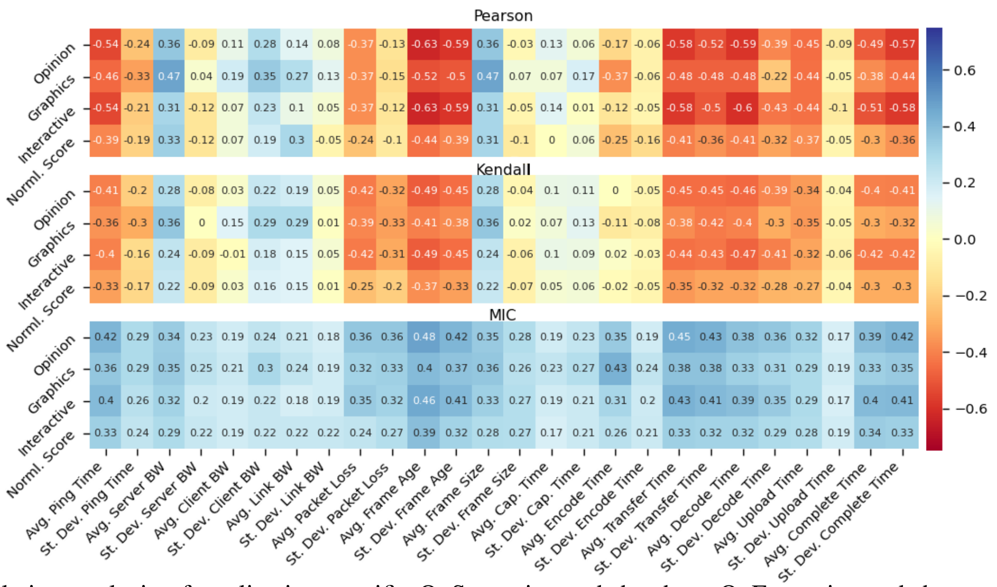
Figure 4: Correlation analysis of application-specific QoS metrics and the three QoE metrics and the normalized in-game performance metric, using three different correlations metrics (Pearson, Kendall, MIC)

and in that case it is ranked second, closely after the average encoding time, which as we will see in Section VII makes up a significant fraction of the frame age. The importance of the average frame age and its standard deviation is also visible when looking at the top-5 correlations using each correlation metrics. See Table II. For all three correlation metrics, these two frame-age metrics are together responsible for the four top-4 entries of each respective top-5 lists. These entries are significantly correlated. For example, the highest p-value for any of the top-5 entries in Table II are $2.20 \cdot 10^{-26}$ (Pearson), $3.38 \cdot 10^{-29}$ (Kendall), and bounded by $1.28 \cdot 10^{-6}$ (MIC). Furthermore, the largest p-value value for any of the correlations associated with the average frame age is no larger than $1.88 \cdot 10^{-16}$ (MIC with $1.28 \cdot 10^{-6}$). The very small p-values observed for average frame age (e.g., all much smaller than $p < 0.05$) confirms that it always is significantly correlated with the QoE and in-game performance, regardless of metric.

There are, however, many other metrics that are much less correlated with QoE and in-game performance than the frame age. For example, the null hypothesis of "no correlation" is rejected (in favor of the alternative hypothesis of "non-zero correlation") with 95% confidence ($p < 0.05$) in only 44.97% (Pearson), 34.02% (Kendall) and 36.98% (MIC) of the total pairwise cases shown in Figure 4.

Looking closer at the correlation matrices, we also observe strong correlations for the average ping time (with QoE metrics) and the two transfer time metrics (with QoE metrics). In contrast, we observe very small (individual) correlations for the standard deviation of both frame size and server bandwidth. As long as the averages were sufficiently high, we did not observe big QoE penalties for sessions with big variations in frame sizes or bandwidths.

We have not observed much difference between the best

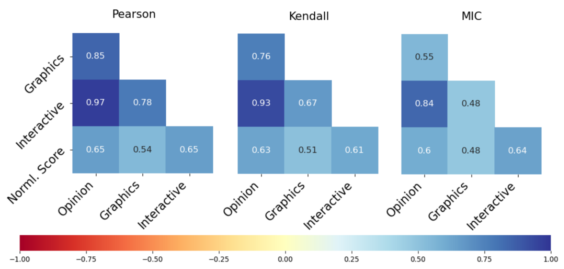
Figure 5: QoE-to-QoE correlations

metrics when using the Pearson and Kendall correlations; e.g., the same correlations show up in almost the same order. One reason for this is that some of the relationships indeed are of linear nature. Furthermore, the Pearson scores typically are slightly higher than the corresponding Kendall scores, indicating that the linear correlations are stronger than the ordinal association measured by Kendall. Visually, the MIC differs the most. This is primarily due to MIC using a different scale (without negative values). However, in the end, high values (dark cells) are observed for roughly the same columns and the average frame age is still the best predictor for three out of four QoE/performance metrics.

Finally, when interpreting the above results, it should also be noted that the QoE and in-game performance metrics themselves are highly correlated. This is illustrated in Figure 5. Here, we note that the opinion score (MOS) is most strongly correlated with the interactive scores (pairwise correlations of 0.97, 0.93 and 0.84), and relatively less correlated with the graphical scores (0.85, 0.76, and 0.55). This is most likely an effect of the game's fast-paced nature.

In summary, the above results show that the frame age is a prominent measurement to determine the players' QoE

Table II: Top-5 highest QoS-QoE correlations using each correlation metric

Rank	Pearson	Kendall	MIC
1	Avg. Frame Age & Gen. Op. Score	Avg. Frame Age & Gen. Op. Score	Avg. Frame Age & Gen. Op. Score
2	Avg. Frame Age & Inter. Score	Avg. Frame Age & Inter. Score	Avg. Frame Age & Inter. Score
3	St. Dev Frame Age & Inter. Score	St. Dev. Frame Age & Inter. Score	St. Dev Frame Age & Gen. Op. Score
4	St. Dev Frame Age & Gen. Op. Score	St. Dev. Frame Age & Gen. Op. Score	St. Dev Frame Age & Inter. Score
5	Avg. Ping Time & Gen. Op. Score	Avg. Transfer Time & Gen. Op. Score	Avg. Ping Time & Gen. Op. Score

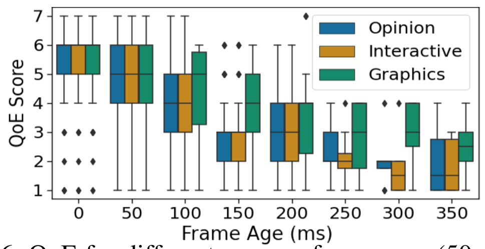

Figure 6: QoE for different average frame ages (50 ms bins)

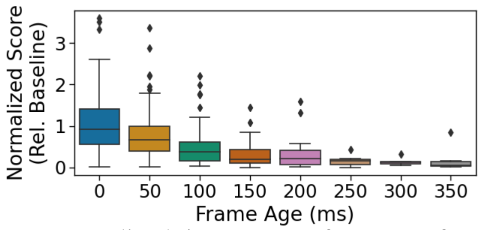

Figure 7: Normalized in-game performance for different average frame ages (50 ms bins)

(regardless of QoE metric). We also note that some of the QoS metrics making up the frame age also scores high in some cases (e.g., the transfer time and encoding time), making them promising components to optimize when wanting to reduce the frame age and improve QoE.

B. Age-based QoE Analysis

Motivated by the above observations, we next present a brief, initial characterization how the average frame age impacts the three QoE metrics and the in-game performance. The frame age is further analyzed in Sections VI and VII.

Figures 6 and 7 present box plots for the three QoE measurements and the normalized in-game performance, respectively, as a function of the average frame age. Here, we use a bin size of 50 ms. As expected from the correlation results above, all four metrics degrade significantly with increasing frame age. For example, for the games with an average frame age of 0-50 ms we see a median normalized in-game score of 0.872 (0.925 if including baselines), whereas the median scores quickly drops to 0.203 when the frame age is 150-200 ms. Furthermore, while we observe some normalized scores above one (outperforming the baseline) for all bins up to a frame age of 200 ms, most normalized scores above one occur in the 0-50 ms bin.

Looking at the three QoE metrics, we can clearly see that the opinion score and interactive score behave very similarly, and that they both are much more affected by the frame age than the graphical score is. The fast-paced nature of the game is likely a contributing factor to these differences.

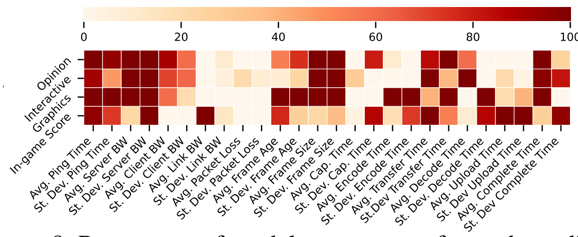

Figure 8: Percentage of model occurrences for each predictor

The above observations again highlight that frame age is a good QoS proxy for users' in-game performance and QoE.

C. Model-based Parameter Selection

The frame age is not the only good predictor of QoE and in-game performance. In fact, the frame age itself can be seen as the sum of multiple timing-related QoS metrics. Some of these QoS metrics are themselves correlated, while others may help explain complementing aspects. In this section, we use multivariate regression models to identify the set of QoS variables that together best model the three QoE metrics and the normalized in-game performance.

For this analysis, we performed a best subset regression analysis on the data. To ensure that collinearity has the smallest possible impact, we have used Mallow's C_p to assess the models. For the remaining models, we have calculated how often each predictor variable occurs. Figure 8 shows the percentage of times that each QoS metric occurs in the models obtained using the above method. Note that higher percentages (darker boxes) suggest that the metric typically is included in the corresponding multivariate regression model, and that the variable (together with the other frequently occurring variables) are among the best predictors.

Interestingly, while the average frame age does show up in many models, it is not the most frequently occurring QoS metrics. Instead, there are other metrics that individually correlate less strongly with the QoE (or in-game performance) that are included in more models. These metrics provides complementing information to the other metrics included in the models and therefore allow the combined models to explain more of the variations in the QoE. Examples of such metrics are the average and standard deviation of the server bandwidth and frame sizes. These metrics have much weaker individual correlations, but are included more frequently than the frame age as they contribute complementing information not contributed by other included metrics. We also note that some of the timing metrics that make up the frame age (i.e., capture time, convert time, encode time, transfer time, decode time, upload time, and complete time; see breakdown in Section VII) have many model occurrences for at least one of the four QoE/performance metrics.

In many cases, the secondary metrics provide complementing information for different models and their weights therefore differ between the models of the four QoE and in-game performance measures. (See metrics with big frequency differences; covering a span from light to dark colors in the same column.) Examples of QoS metrics that only are frequently included in the models of one QoE/performance metric are the link bandwidth (only in-game performance) and the upload time (in-game performance) and the encoding time (graphical score).

VI. Non-QoS factors impacting QoE

Thus far we have seen that frame age is the QoS metric that best explains the three QoE metrics and in-game performance, and that some of the sub-components making up the frame age may provide added value to multivariate models (as they contribute complementing information). In this section, we consider the impact that some important non-QoS metrics have on these observations, and in Section VII we provide a breakdown of the frame age and analyze autocorrelations, cross correlations, and other relationships between the components making up the frame age.

While QoE and in-game performance clearly depend on various QoS factors, they are also impacted by non-QoS factors. Here, we extend our analysis to account and control for player-specific biases and (in the multiplayer context) whether a player wins or loses.

A. *Player Differences*

Clearly, the opinions and performance of any two players will never be the same. It is therefore valuable to take into account the impact that each individual player has on the model. For this purpose, we (i) repeated the analysis for each player individually and (ii) extend the regression-based analysis using categorical variables, allowing us to account for player identity within a single model. Both these analysis methods are well suited for our dataset, as such analysis is best performed on datasets including only a few players that play many games. (As should be clear now, our methodology and dataset was designed with the analysis presented in Section VI in mind.)

Figure 9 presents a per-player breakdown of the QoE metrics and normalized in-game performance as a function of the average frame age. While this plot highlight some differences in the QoE scoring of the two players (e.g., player 1 tends to provide higher graphics scores compared to player 2, for which we see a sharper decline in opinion scores), the general trends are relatively similar. When looking at the in-game performance, we observe some differences that might be related to skill differences between

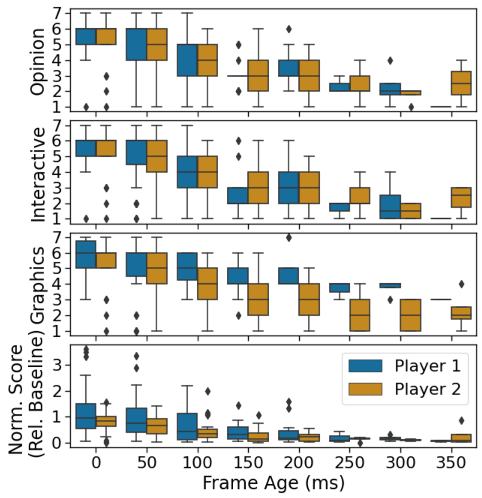

Figure 9: QoE and in-game performance for the two players during games with different average frame age (50 ms bins)

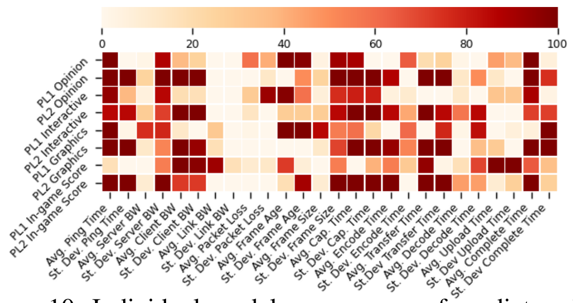

Figure 10: Individual model occurrences of predictors broken down for player 1 (PL1) and player 2 (PL2)

the players. For example, for almost all frame ages, player 1 has noticeably larger variation in its normalized in-game performance compared to player 2. This also results in player 1 having more instances where the player outperforms the baseline tests within the same session.

We next look at the model occurrences of each QoS metric when we use data from one player at a time. Figure 10 presents these results. Interestingly, the most frequent QoS metrics differ substantially between the players but are relatively more similar for the different QoE models of the same player. However, in all cases, both players have at least some metric included that is highly correlated with or contribute significantly to the frame age (e.g., average ping time, transfer time). For player 1, the frame age was, in fact, frequently included for the three QoE metrics. These results are consistent with our main observations for the aggregate model, which showed that despite frame age having the strongest individual correlation, several other factors are more frequently occurring in the multivariate models.

We also performed a step-wise regression analysis to select the best model when allowing for categorical predictor variables to separate player biases in the scoring. With this analysis only the normalized in-game performance model ended up including a player category. However, when using

Table III: Linear regression with categorical player variable. All values non-zero with 95% confidence except one ("*").

Model	Intercept	Slope	Player
Opinion	5.958 ± 0.140	-0.013 ± 0.001	-0.323 ± 0.144
Interactive	5.885 ± 0.151	-0.014 ± 0.001	-0.175 ± 0.156 (*)
Graphics	6.204 ± 0.144	-0.010 ± 0.001	-0.901 ± 0.149
In-game	1.171 ± 0.060	-0.003 ± 0.000	-0.276 ± 0.062

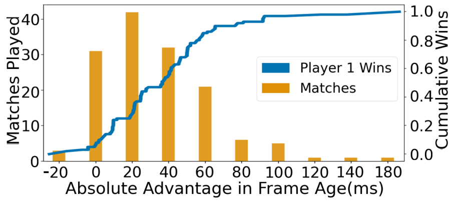

Figure 11: Player 1's frame age advantage (yellow using 20 ms bins) and cumulative normalized win count

categorical variables together only with the frame age, we did observe bigger differences. Table III presents the estimated constants for the four regression models when using a categorical variable to distinguish between players. We note that the player category was statistically significant in 3 out of 4 models (the interaction model being the exception) and that the biggest differences (between the players) again was observed for the graphical opinion score.

B. Multiplayer results & impact of wins/losses

Due to the adaptive algorithm used to equalize win probabilities, the multi-player experiments were performed using a much narrower span of network conditions. For example, we did not introduce any packet losses, and the largest average latencies were typically less than half of the maximum latencies in the single-player experiments. This can also be seen in Figure 11. Here, we show the number of games that player 1 had a certain average advantage in the frame age (using 20 ms bins) and the conditions under which it accumulated its wins (using CDF). At this time, we can also note that only player 2, who was the better player during our experiments, ever experienced average frame ages above 100 ms.

Due to lack of space and this much narrower parameter range, in this section, we limit our analysis to demonstrate the relative impact of player differences, their skills, and the impact of winning or losing games may have on the QoE scores. To illustrate this, Figure 12 shows the QoE scores (using a box plot) for different average frame ages (50 ms bins), broken down both per player (1 vs. 2), and whether a player won or lost the game. It is worth noting that there are very few results for the 100-150 ms bin. (The 150-200 ms bin was removed as it only contained one data point.)

In addition to general player differences such as those that we observed before (e.g., player 1 giving higher QoE scores than player 2, for the same conditions) and generally declining QoE scores as the frame age increases, the most striking observation is the impact of a player winning/losing

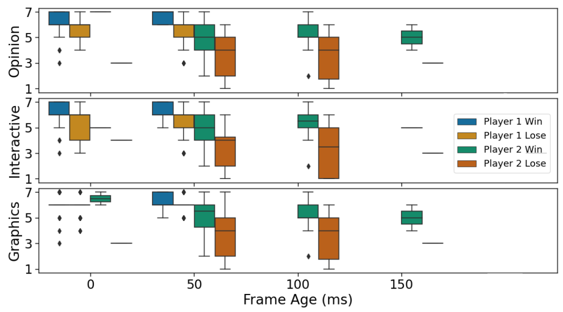

Figure 12: QoE scores dependence of win/loss and for different average frame ages (50 ms bins)

a game. For example, when winning the game, both players give significantly higher QoE scores for all three QoE metrics, including for the graphical score, which one may have expected would not be affected. This is interesting as it shows a clear bias introduced due to the emotions that a win/loss may cause. This result helps further explain the high correlations observed between the in-game performance and the general opinion score seen for the single-person game (see Figure 5). The above results also show the value of balancing the conditions so to provide both players a similar opportunity to win. Having said that, in the end, with our current method, player 2 still won 58.5% of the games. While longer play sessions may further reduce the win imbalance, the better player will always have a better chance of winning at the start of the sessions (as both players start with the same network conditions before the penalty can be adaptively adjusted; see Section III). A perfect balance should, therefore, not be expected.

VII. Frame-by-Frame Analysis

A. Frame age breakdown

Figure 13 shows an example trace from one of the baseline tests. Here, we show both the absolute time (top) and relative fraction of the time (bottom) consumed by each step in the end-to-end process making up the frame age. Both subfigures are stacked in the order that the steps take place in the end-to-end process: (i) capture time, (ii) convert time, (iii) encode time, (iv) transfer time, (v) decode time, (vi) upload time, and (vii) complete time. The first three steps happen at the server, and the last three happen at the client.

For this scenario, the complete time (last step on the client), transfer time, and encode time (last step on the server) are responsible for the majority of the frame age. While these components often dominate also in other scenarios, both the absolute values and the relative fractions may differ. This is illustrated in Table IV, where we provide summary statistics for the average (and standard deviation) of the time spent in each step of the end-to-end process for four different example scenarios: "low-low", "low-high", "high-low", and "high-high". The first tag in each scenario label corresponds to the added latency (low = 0 ms or high =

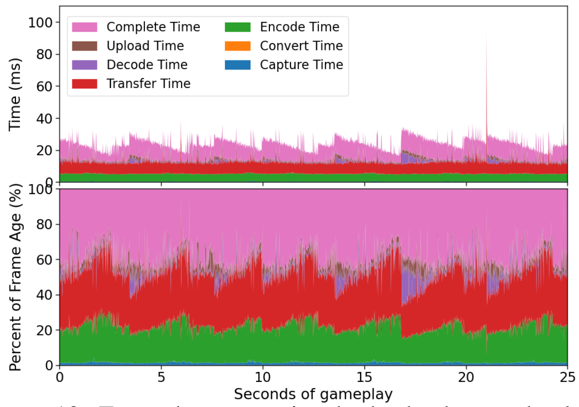

Figure 13: Example trace using both absolute and relative per-frame times to break down the frame age

Table IV: Scenario-based frame age breakdown. All reported values are measured in milliseconds (ms) and shown using the mean (μ) and standard deviation (σ) as $\mu(\sigma)$.

	Low-low	Low-high	High-low	High-high
Capture	0.78 (0.26)	0.73 (0.21)	0.77 (0.26)	0.77 (0.24)
Convert	0.39 (2.68)	0.37 (2.68)	0.29 (0.09)	0.28 (0.08)
Encode	5.10 (0.28)	5.11 (0.18)	4.86 (0.17)	4.88 (0.20)
Transfer	3.47 (3.69)	3.44 (3.40)	204.81(3.78)	211.35(35.61)
Decode	2.66 (3.40)	1.74 (3.32)	2.01 (2.86)	7.89 (7.70)
Upload	1.62 (0.40)	1.16 (0.58)	1.29 (0.68)	2.02 (0.47)
Complete	10.41 (2.81)	9.75 (2.70)	10.05(2.81)	71.03(122.34)
Other	1.78 (1.18)	1.82 (1.35)	1.54 (1.71)	35.74 (66.68)
Frame age	26.21(6.84)	24.14(6.97)	225.60(6.21)	334.18 (120.15)

200 ms), and the second tag corresponds to the added packet loss rate (low = 0% and high = 1%). Here, we have used all per-frame data for the single-player sessions associated with these conditions.

To validate that system constraints and data collection does not impact our results, we also monitored memory, disk, bandwidth, and CPU utilization. Figure 14 provides some example traces for the example scenario above. These measurements are made at 60 Hz. For the CPU trace, we use a moving average (with standard deviations marked using shading), whereas, for the others, we use the raw sample values. First, note that the special data collection frames, seen as periodic spikes in client's send rate (top-right subfigure) and bytes written to disk by the server (bottom-right), do not appear to significantly impact any other performance metric and the spikes do not appear to coincide with the frame age spikes. Second, note that both the client and server operate at relatively intermediate CPU utilization and have relatively stable memory utilization. We have not found the CPU and memory constraints to impact our performance, and they themselves do not appear to be impacted much by any of the frame-age related spikes.

The reverse saw-tooth pattern observed in Figure 13 is mainly due to sudden spikes in the decode time, which then slowly taper off. Some of the most extreme spikes are due to a few frames with particularly high transfer times. We stress

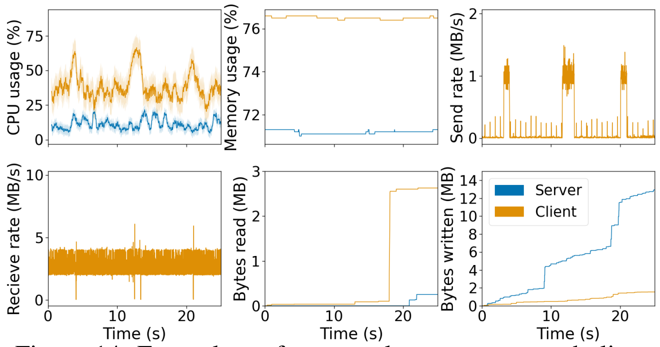

Figure 14: Example performance logs at server and client

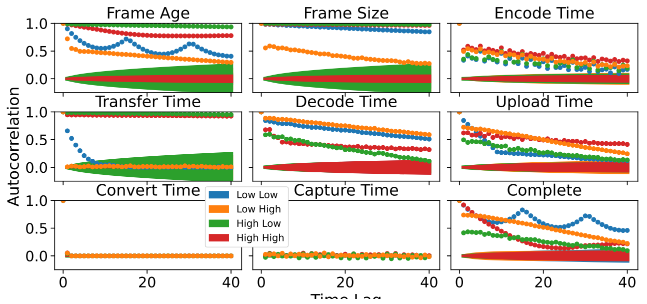

Figure 15: Auto-correlations for different QoS metrics (subplots) and time lags (x-axis) under four different network conditions (colors).

that these spikes typically are associated with transfers from the server to the client, not the data collection frames sent from the client to the server (every 10 seconds). To better understand temporal time dependencies, we next present auto-correlation and cross-correlation analysis.

B. *Auto- and cross-correlation analysis*

The semi-periodic pattern observed in Figure 13 (top) suggests that the per-frame times are not independent and that there are temporal dependencies. To better understand if there are temporal dependencies within the processing times of each step, we have calculated the auto-correlations for the frame age, the frame size, and each of the seven steps making up the frame age. Figure 15 presents these auto-correlations using lags of 1-to-40 frames for the four example scenarios: "low-low", "low-high", "high-low", and "high-high" (as defined above). For each metric, we also include a 95% confidence interval for the auto-correlation (represented by the colored cones around the value 0).

Interestingly, only the convert time and capture time fall within the 95% confidence interval (curves within cones) for all scenarios; meaning that they are not auto-correlated. The transfer time also satisfies this independence property, but only for the low latency scenarios (low-low and low-high). All other metrics have clear time dependencies for most scenarios.

The highest auto-correlations with large lags (right-hand-side of curves) are observed in high latency scenarios (green

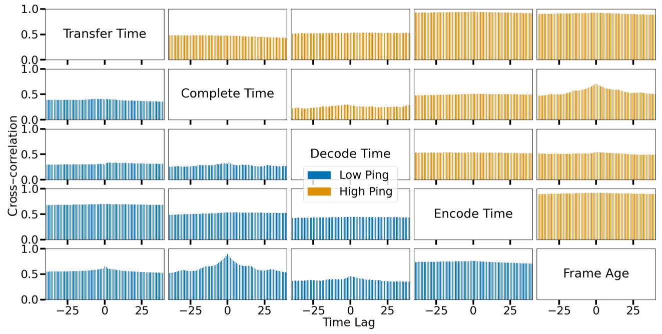

Figure 16: Cross-correlations for different QoS metrics under different conditions: low (blue) or high (yellow) ping

and red curve) with the transfer time, frame age, and frame size metrics. Here, it should be noted that a lag of 15 frames corresponds to 250 ms of game playing (60 frames per second) and that these lags, therefore, are of similar order-of-magnitude in size as the average transfer times when considering the high-low (205 ms) and high-high (211 ms) scenarios. For most other metrics and scenarios, the auto-correlations decrease noticeably with increasing lags, showing that such time dependencies may be ephemeral.

The above auto-correlation results clearly indicate that per-frame measurements can not be treated as independent samples. Therefore, a per-frame based correlation analysis of the different QoS metrics is not suitable from a statistical standpoint. To better understand the correlation between the dominating steps of the frame age (transfer time, complete time, decode time, encode time) and the frame age itself, we instead performed a cross-correlation analysis. Figure 16 shows the average cross correlation between each of these metrics as calculated using all time series for the low-low (blue, below diagonal) and high-low (yellow, above diagonal) scenario. Here, we have first calculated the cross-correlations for each session falling into these two scenarios and then reported the averages across all such sessions. While the averaging reduces some initial spikes and most cross correlations are relatively constant across lags, we do observe a small spike around a lag of zero between complete time and frame age. This spike is in part due to the sharp (and very short) spikes in complete times (pink) that immediately follow sharp spikes in transfer times (red) seen if zooming in very closely in figures such as Figure 13. We also observe somewhat more noticeable cross-correlation spikes between the frame age and the individual processing steps for the low-low scenario (blue). However, in general it can be noted that the cross-correlations are all positive and many of the values are relatively high (close to 1); in the case of cross-correlations with frame age, again highlighting their significant impact on the frame age.

VIII. Conclusions

This paper presented the evaluation results from targeted user-based experiments in which we collected detailed per-frame statistics during user tests in which we manipulated the network conditions experience by cloud gamers. Of special interest was the relationship between objective and easier to collect QoS factors and more subjective quality of experience (QoE) metrics and the players' in-game performance (only available to the client and server, not a network operator or third-party cloud provider, for example). Using a range of correlation-based metrics, different classes of regression models, and a detailed breakdown of the frame age (the QoS metric that we find is most strongly correlated with the users' QoE and in-game performance), we provide new insights into the impact of network conditions on the gamers' QoE and the different QoS metrics impact on the frame age and QoE under different network conditions.

The paper provides a strong empirical case for the importance of maintaining a low frame age, shows that end-to-end latencies typically impact QoE more than packet losses, and provides examples how temporary spikes in frame age caused by one end-to-end-processing step can have extended negative impact through the impact this can have on other processing steps (e.g., transfer time spikes translating into complete time spikes, followed by decode time backlogs). Our results emphasize developers' need to consider and optimize the different parts making up the frame age and reduce time dependencies between them. By ensuring stable, low frame age developers and providers can provide cloud gamers with enjoyable, high-quality gaming experiences. Promising mitigation approaches that may reduce temporal performance degradation and even help handle cloud service failures include distributed solutions that allow the rendering to be moved between VMs and clients [18].

Interesting future work include applying the analysis methodology presented here also to the context of cloud-based delivery of mixed reality [31]. Another seemingly promising direction for future work, derived from our multi-player observations, is the development of more advanced, adaptive techniques that account for various aspects that may impact the user's perceived experience (e.g., winning a game, achieving certain levels within the game, or otherwise scoring within a game) that may help compensate for temporal degradation in the frame age, for example.

Acknowledgements: This work was funded in part by the Swedish Research Council (VR).

References

[1] R. D. Yates, M. Tavan, Y. Hu, and D. Raychaudhuri, "Timely cloud gaming," in *Proc. IEEE INFOCOM*, 2017.

[2] "The Global Cloud Gaming Market Is on Track to Generate Revenues of $3.2 Billion by 2023 — Newzoo." [Online]. Available: https://newzoo.com/insights/articles/cloud-gaming-business-market-revenues-and-ecosystem/

[3] D. N. Reshef, Y. A. Reshef, H. K. Finucane, S. R. Grossman, G. McVean, P. J. Turnbaugh, E. S. Lander, M. Mitzenmacher,

and P. C. Sabeti, "Detecting novel associations in large data sets," *Science*, vol. 334, no. 6062, pp. 1518–1524, 12 2011.

[4] K. Raaen and A. Petlund, "How much delay is there really in current games?" in *Proc. ACM MMSys*, 2015.

[5] M. Claypool and D. Finkel, "The Effects of Latency on Player Performance in Cloud-Based Games," in *Proc. ACM SIGCOMM Workshop on Network and Systems Support for Games (NetGames)*, 2014.

[6] M. Dick, O. Wellnitz, and L. Wolf, "Analysis of Factors Affecting Players' Performance and Perception in Multiplayer Games," in *Proc. ACM SIGCOMM Workshop on Network and System Support for Games (NetGames)*, 2005.

[7] V. Clincy and B. Wilgor, "Subjective evaluation of latency and packet loss in a cloud-based game," in *Proc. ITNG*, 2013.

[8] M. Jarschel, D. Schlosser, S. Scheuring, and T. Hoßfeld, "An evaluation of QoE in cloud gaming based on subjective tests," in *Proc. IMIS*, 2011.

[9] 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 *Proc. ACM SIGCOMM Workshop on Network and System Support for Games (NetGames)*, 2004.

[10] S. S. Sabet, S. Schmidt, S. Zadtootaghaj, C. Griwodz, and S. Moller, "Towards applying game adaptation to decrease the impact of delay on quality of experience," in *Proc. IEEE ISM*, 2019.

[11] M. Claypool and K. Claypool, "Latency can kill: Precision and deadline in online games," in *Proc. ACM MMSys*, 2010.

[12] C.-H. Hsu, H.-J. Hong, C.-F. Hsu, T.-H. Tsai, C.-Y. Huang, and K.-T. Chen, "Enabling Adaptive Cloud Gaming in an Open-Source Cloud Gaming Platform," *IEEE Trans. on Circuits and Systems for Video Technology*, pp. 2078–2091, 12 2015.

[13] G. K. Illahi, T. V. Gemert, M. Siekkinen, E. Masala, A. Oulasvirta, and A. Ylä-Jääski, "Cloud Gaming with Foveated Video Encoding," *ACM Trans. on Multimedia Computing, Communications, and Applications*, vol. 16, no. 1, pp. 1–24, 3 2020.

[14] I. Slivar, M. Suznjevic, and L. Skorin-Kapov, "Game categorization for deriving QoE-driven video encoding configuration strategies for cloud gaming," *ACM Trans. on Multimedia Computing, Communications and Applications*, vol. 14, no. 3s, pp. 1–24, 6 2018.

[15] C. Zhang, J. Yao, Z. Qi, M. Yu, and H. Guan, "vGASA: Adaptive scheduling algorithm of virtualized GPU resource in cloud gaming," *IEEE Trans. on Parallel and Distributed Systems*, vol. 25, no. 11, pp. 3036–3045, 2013.

[16] Z. Xue, D. Wu, J. He, X. Hei, and Y. Liu, "Playing high-end video games in the cloud: A measurement study," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 25, no. 12, pp. 2013–2025, 2014.

[17] C.-Y. Huang, C.-H. Hsu, Y.-C. Chang, and K.-T. Chen, "GamingAnywhere: an open cloud gaming system," in *Proc. ACM MMSys*, 2013.

[18] J. Bulman and P. Garraghan, "A cloud gaming framework for dynamic graphical rendering towards achieving distributed game engines," in *Proc. USENIX HotCloud*, 2020.

[19] R. Shea, J. Liu, E. Ngai, and Y. Cui, "Cloud gaming: Architecture and performance," *IEEE Network*, vol. 27, no. 4, pp. 16–21, 2013.

[20] K. T. Chen, Y. C. Chang, H. J. Hsu, D. Y. Chen, C. Y. Huang, and C. H. Hsu, "On the quality of service of cloud gaming systems," *IEEE Trans. on Multimedia*, vol. 16, no. 2, pp. 480–495, 2 2014.

[21] I. Slivar, M. Suznjevic, and L. Skorin-Kapov, "The impact of video encoding parameters and game type on QoE for cloud gaming: A case study using the steam platform," in *Proc. QoMEX*, 2015.

[22] Steam Support, "Steam Remote Play," 2020. [Online]. Available: https://support.steampowered.com/kb_article.php?ref=3629-riav-1617

[23] "clumsy, an utility for simulating broken network for Windows Vista / Windows 7 and above." [Online]. Available: https://jagt.github.io/clumsy/

[24] G. A. Miller, "The magical number seven, plus or minus two: some limits on our capacity for processing information," *Psychological Review*, vol. 63, no. 2, pp. 81–97, 3 1956.

[25] R. C. Streijl, S. Winkler, and D. S. Hands, "Mean opinion score (MOS) revisited: methods and applications, limitations and alternatives," *Multimedia Systems*, vol. 22, no. 2, pp. 213–227, 3 2016.

[26] "Geometry Wars: Retro Evolved på Steam." [Online]. Available: https://store.steampowered.com/app/8400/Geometry_Wars_Retro_Evolved/

[27] S. S. Sabet, C. Griwodz, and S. Möller, "Influence of primacy, recency and peak effects on the game experience questionnaire," in *Proc. ACM MMVE*, 2019.

[28] "SpeedRunners på Steam." [Online]. Available: https://store.steampowered.com/app/207140/SpeedRunners/

[29] W. IJsselsteijn, Y. de Kort, K. Poels, A. Jurgelionis, and F. Bellotti, "Characterising and Measuring User Experiences in Digital Games," in *Proc. ACE*, 2007.

[30] M. Suznjevic, I. Slivar, and L. Skorin-Kapov, "Analysis and QoE evaluation of cloud gaming service adaptation under different network conditions: The case of nvidia geforce now," in *Proc. QoMEX*, 2016.

[31] K. Toczé, J. Lindqvist, and S. Nadjm-Tehrani, "Performance study of mixed reality for edge computing," in *Proc. IEEE/ACM UCC*, 2019.