5th ISCA/DEGA Workshop on Perceptual Quality of Systems (PQS 2016)
29-31 August 2016, Berlin, Germany

A Comprehensive End-to-End Lag Model for Online and Cloud Video Gaming

Florian Metzger[1], *Albert Rafetseder*[2], *Christian Schwartz*

[1]Chair of Modeling of Adaptive Systems, University of Duisburg-Essen
[2]NYU Tandon School of Engineering

florian.metzger@uni-due.de, albert.rafetseder@univie.ac.at
christian.schwartz@gmail.com

Abstract

To further the knowledge of basic game mechanics and improve future video game Quality of Experience (QoE) studies we present a model and simulation for End-to-End (E2E) lag in networked and cloud computer games. E2E lag describes the latency between a user's action and the display of the action's results on the screen, thus providing both a basic measure for interactivity and a fundamental parameter to derived QoE metrics. In contrast to E2E lag models that focus on the network Round-Trip Time (RTT), the presented work also factors in the game's tickrate and framerate (and codec latencies in the case of cloud games). This reveals side effects that can entirely mask the network delay's contribution to the total lag, an important notion to consider for future game studies.
Index Terms: gaming, end-to-end lag, queuing model

1. Introduction

End-to-End (E2E) lag, i.e. the delay between an input event and the feedback of the event's result, ubiquitously impacts human interactions with computers. This notion has a rich history, comprising studies on both objective (e.g. task completion times) and subjective (user experience / QoE) metrics for various types of interactions. Online multiplayer and cloud video games form the perhaps most recent objects of investigation: Lag is a main governing factor in determining the interaction quality of video games, as a higher lag means an apparent detachment of a player's inputs from the game's resulting visible reactions. This has not been lost on researchers, prompting research on gaming QoE given various additional input parameters such as game categories, game task classifications, player activity, etc.

However, online video game QoE assessments appear oblivious to the inner workings of video games. This especially means understanding the main game loop with its tickrates as well as mechanics and implications surrounding the framerate. This paper, as a continuation of previous work conducted in [1], aims to illustrate the inner workings of the main game loop and highlight the different contributors to E2E lag. It describes a general model of E2E lag on based on intrinsic game and interaction factors, especially the framerate and tickrate. To demonstrate the model, this paper further provides a simulation implementing typical gaming scenarios. Results from this simulation indicate that the contributions of both framerate and tickrate to the E2E lag, and therefore on the subjective interaction quality of the game, easily exceed the influences of, e.g., the network connecting the game client and server.

The results presented and the custom tools that are available as free-open source software from the authors' public repository [2], may assist the design of future subjective quality assessment studies. They point at relevant parameters, and allow for surveying the amount of influence of these parameters.

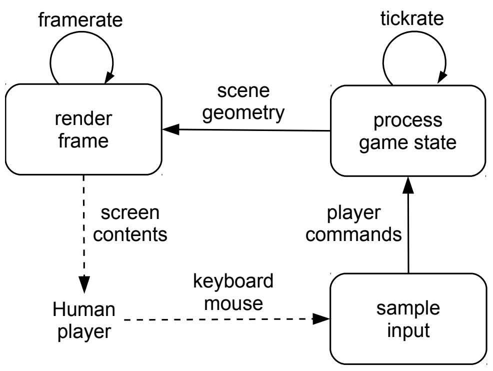

Figure 1: Basic model of a continuous main video game loop.

This paper is structured as follows. First, §2 gives an introduction to parts of video game engine terminology and properties relevant to this model, and reviews related work. Afterwards, §3 describes the abstract model to describe E2E lag, followed by a simulation parameter study in order to identify main influence factors for the E2E lag in §4. The paper concludes in §5 with a discussion of key findings and future perspectives.

2. Background

Some fundamental gaming terms and concepts need to be introduced first. At their core video games are essentially feedback-directed real-time simulators. Figure 1 overviews a simplified simulation loop: The game reads player input, updates the game state, and renders new screen contents. This loop repeats as long as the game is running. While the three parts are interrelated, and every component provides some form of input to the next, they may still proceed at their own intrinsic paces.

For example, the state of the game world is updated even if the player character does not move. For modeling purposes, we distinguish a generic process for player input, a *tickrate* that governs the frequency of game state updates, and a *framerate* determining how often the screen is updated every second. The *tickrate* governs the frequency of game state updates, and the *framerate* determines the update rate of the output image. This basic makeup can be found in almost any kind of game architecture. In client/server games, the game server is the single authoritative point of state synchronization for other connected clients. Peer-to-peer architectures are not investigated here. Player input events are usually also only sent to the server according to a specific clocked *command message rate*.

Actual game implementations may choose to update parts of the game state on different frequencies. For example, a physics effect that does not influence gameplay directly could be updated at only half the tickrate. Also, the parameters of the player input process depend on the game genre and on the player action currently performed. Examples for the tickrates of popular networked games include 64 Hz or 128 Hz for CS:GO, 20 Hz for MINECRAFT, or 30 Hz for DOTA 2.

2.1. Framerate

The framerate is the number of frames (i.e. visual images) displayed to the user per second. Below the threshold of *apparent motion* (about 16.67 frames per per second), objects on consecutive frames will appear as two distinct objects. Traditional video media play back at fixed framerates, e.g., 24 Hz. Video games are more flexible but also much more demanding on the framerate. First, the framerate in a game may fluctuate as the processing time of the render process depends on the scene complexity. Then, video games usually target higher framerates than other media do: 30 Hz, 60 Hz, or even 120 Hz, depending on the type of game. Higher framerates enable smoother camera movement and scene transitions, as games often present faster scenes when compared to videos; they also help in increasing the interactivity as video games constantly require input on short time scales to which the game reacts and displays the feedback. Therefore, the framerate influences the reactivity of a game, but can also be a source of latency itself.

2.2. End-to-End Lag

Lag in video gaming is often described solely on the basis of the network delay in an online game. It should be evident that the lag is a critically important factor for almost all games, as it governs the reaction time to in-game events.

However, focussing solely on network delay neglects other components that contribute to the lag, including the input device, the time to sample and process the input, the game engine and server and their tickrates, frame rendering time, and ultimately the time to display the frame on the monitor. Only if all sources are factored in the complete *E2E lag* is captured. Notably, this lag is usually not constant but can vary depending on the type of action triggered by the input. While some simple actions, say opening the menu, may have a very short lag, more complex interactions, e.g., issuing command that moves the player character in the game world, may take considerable longer to complete. Therefore, each video game will have a distinct "lag profile".

2.3. Related Work

Lag is a recognized contributor to QoE in video games, yet its mechanics outside of the network realm seem to have attracted less attention. In [3] Jarschel et al. identify some influence factors on the subjective quality of cloud gaming through a user survey for certain games and three different game categories (slow, medium, fast games) that have been subjected to worsening Quality of Service (QoS) parameters. Downstream packet loss and delay was noted be be especially problematic for achieving a good quality. Similarly, the authors of [4] observed the relationship of players quitting a Massively Multiplayer Online (MMO) game with deteriorating QoS. Additionally, a user study in [5] also showed a correlation of the QoE to the delay as well as the jitter for another MMO, in this case the total delay had more impact than the delay variation. Regarding the subjective quality in first person shooters, the authors of [6] find a strong impact of the delay and packet loss on the experienced quality. An ITU-T Recommendation [7] concerning subjectively measuring video game QoE is also in preparation, which discusses game-relevant QoS-metrics as well as the selection of players and games.

Other approaches examine the player objective performance through in-game metrics such as the game's highscore or the duration to achieve a certain task. A 2006 paper [8] categorizes player actions and their relationship to latency, with special regards for the precision and deadlines of actions. The "kills per minute" of players in the First-Person Shooter (FPS) QUAKE 3 are investigated by [9], which sees a steady decline of this subjective performance metric when increasing the network delay. A further paper [10] notes the influence of network QoS on in-game actions and specifically looks at player performance in first person games, with the performance being worse in a degraded network. Finally, the authors of [11] also find a strong and negative influence of high delay on the player's performance, in this case again in QUAKE 3.

On a more fundamental level, Ivkovic et al. [12] quantify the effect of local latency, including input, rendering, and output devices. They find that as the latency is increased from 11 to 164 ms, the time-to-completion for 3D targeting tasks increases by around 50% from 0.4 and 0.8 seconds for "easy", i.e. larger, and "difficult" targets respectively. The time the test subjects succeeded tracking a moving target reduces by 30% as the latency is increased. These results are qualitatively similar to a much earlier study [13] on the difficulty of reaching for static objects in Virtual Reality displays, conditioned on the virtual object's size and the motion-to-display lag.

3. An End-to-End Lag Model

The E2E lag T can be defined as the elapsed time between a player inputting some commands and the results of these commands being displayed on screen. The section describes this lag as a model for the case of an online dedicated client/server game. The player input is assumed to be a stochastic process U. Furthermore, input events are queued up and sent en bloc at specific intervals defined by the command rate c. A fixed tickrate g is associated with the computation process of the game server, and a framerate f with the rendering process. The server updates the state of the game world at every tick, and may spend some processing time P in order to do so. The processing time is a random variable which is assumed to follow a truncated normal distribution in our simulations. Once finished, an update message is sent to the client. The contents of this update message can then only be incorporated into the next rendering cycle and not the one that is currently in progress. While the rates of the command, tick, and render cycles are assumed to be constant for the sake of this model (though not necessarily identical), they are not operating in lockstep but independently of each other. This is represented in the model by a uniform random initial phase offset for each rate. Since the model's focus is on depicting an online game, it includes the network paths between the different entities. A left-truncated normally distributed random variable D represents the networking delay between game client and server. Figure 2 shows the queuing model for the online video game case, including the three clocked processes responsible for the game's interactions.

The model for cloud gaming is slight variation of this on-

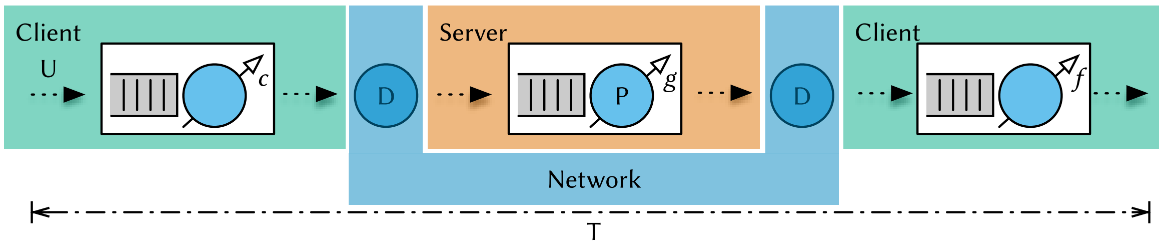

Figure 2: Queuing lag model in an online video game case.

line game model, adapting its notions of D, P, T, U, c, and f. However, the server now handles only one client, and becomes a *streaming server* that also renders and encodes the screen contents, adding a constant encoding delay e. On the client side, the stream is decoded (adding decoding delay d) before being displayed.

3.1. Model Limitations

These models do not attempt to capture all possible sources of lag that occur in actual gaming. Indeed, the models simplify the following aspects in the hope to make the results more tractable. First, the models ignore the delays contributed by input devices like keyboards, mice, and game controllers, estimated to be below 10 ms. The same goes for the lag of the display device which is typically in the range of 9 ms to 40 ms for a PC monitor, and often larger for TVs. The models can be extended to take those delay factors into account, but they were exempted for the sake of simplicity in this paper. Modern games go to great lengths to handle lag gracefully, and try to “work around it” in various ways. The methods for this vary, and implementations usually are not easy to examine, providing a good opportunity for future work. Such techniques can typically be subsumed under the term “lag compensation”, e.g., the game client tries to predict the server state from past knowledge, allowing for smoother local updates but possibly causing slight deviations and re-synchronization artifacts. It should be noted that none of these techniques alter the E2E lag itself, rather they just try to conceal it on a higher level. So while these mechanisms attempt to decrease the impact of lag on the QoE, they do not invalidate our examination of E2E lag.

4. Lag Simulation

Based on the introduced model a stochastic GNU R-based Discrete Event Simulation (DES) was created [2] and several distinct game scenarios investigated. Due to the influence of several stochastic processes (user inputs U, network delay D, server processing time P) and the differing offset of the clocked processes, a sufficient number of repetitions is required to provide meaningful results. The investigations here are intended to highlight some particular aspect in each of the scenarios. First, the input is modeled by an exponential distribution with a rate of 20 events per second. The offsets between the clocked processes are uniformly distributed in their respective intervals. It is further assumed that the server processing time P follows a left-truncated normal distribution with $\mu_P = 3\,\text{ms}$ and $\sigma_P = 0.1\,\text{ms}$. Additionally, the rate c at which command messages are sent to the game server is set equal to the server's tickrate g as the server would not process more commands either way. This might however increase the E2E lag in some

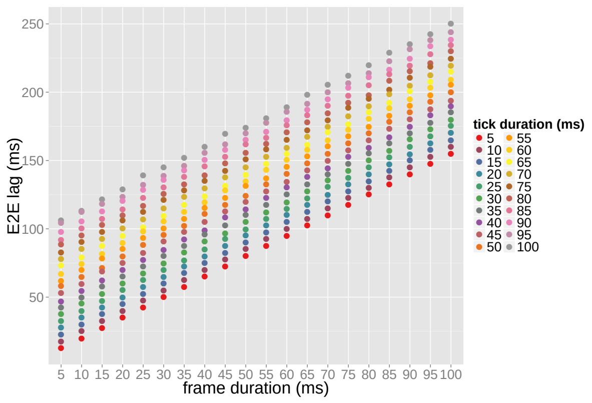

Figure 3: Median E2E lag under various frame and tick durations for a locally-running game (§4.1). Lower lag values are achieved at lower frame and tick durations; the frame duration has a larger influence on the E2E lag.

situations if a command message just misses one of the server's ticks and has to wait another full cycle. The evaluation of the presented scenarios was conducted on the basis of 1,000 repetitions for each setting. Each run consisted of a series of 100 input events and their associated E2E lag values. On this basis a median lag was calculated.

4.1. Best Case Scenario: Local Game

The first and simplest scenario to be investigated here is the case of the local game. In the version implemented here, the tickrate at a locally running quasi-server component is still present, therefore representing the best case an online multiplayer game could achieve without any network influences. Figure 3 plots the relationship between the frame duration (i.e., the inverse of the framerate), the tick duration, and the resulting median E2E lag. Every user input event traverses three queues with fixed-rate outputs ($c = g$, g, and f). In the ideal case, an input event occurs just before the command queue cycles, correctly aligned with the following server tick and frame render cycles. In this case the E2E lag will be slightly above the frame time that must elapse before the update can be rendered to screen, $T_{min} > f^{-1}$. In case the events are unfavorably offset against one another, an input event has to wait almost a full input cycle until it is processed further, reaching the server just after a tick has occurred, so it waits almost a full server tick. Ultimately, it has to wait for almost two frames until it is displayed.

Combined with the previous result the lag is bounded as follows, $f^{-1} < T < c^{-1} + g^{-1} + 2f^{-1}$. The mid-interval point between these two limits is $T_{mid} = \frac{3}{2}f^{-1} + g^{-1}|_{c=g}$ which co-

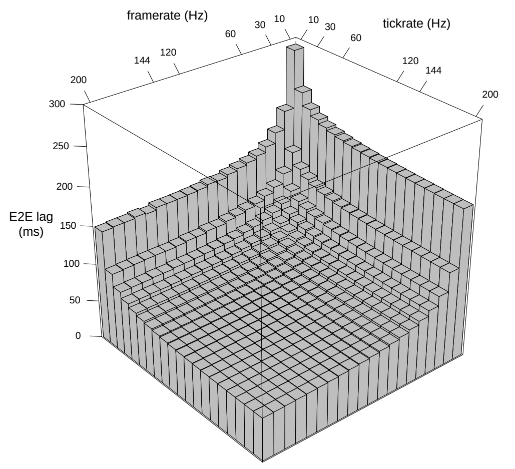

Figure 4: Influence of client framerate and server tickrate on the median E2E lag in the online game scenario (§4.2). Only for high rates f, g, the lag is dominated by the network round-trip and server processing time, $T \approx 2\mu_D + \mu_P = 43\,\text{ms}$.

incides roughly with the medians of the stochastic simulation. Looking at a typical 60 Hz video game with an equal tickrate (i.e. a frame duration of $\approx$ 16.6 ms) the median lag is in the range of 45 ms to 50 ms. So even under quasi-optimal circumstances and without factoring in the delay of the network and screen and input devices, there is already a considerable amount of E2E lag. The expression for T_{mid} and the simulation results also indicate that video games (and quality assessments thereof) should try to achieve the highest framerate possible to minimize its influence on the E2E lag and thus QoE.

4.2. Online Gaming

Next up is a more realistic online video game scenario, now with added network delay D and server processing time P. For this exemplary scenario, the one-way delay D was assumed to follow a left-truncated normal distribution, with $\mu_D = 20\,\text{ms}$ and $\sigma_D = 5\,\text{ms}$. Typical competitive online games today are expected to operate in such ranges. An RTT of 100 ms is often considered to be the upper limit for a good playing experience. Two things can be noted of the influence of frame and tickrates in this scenario, see Fig. 4. First, the framerate has a larger influence on the lag than the tickrate, as in §4.1: unfavorable desynchronization may “hold back” updates for up to two frame durations. Second, for low framerates and tickrates, the impact of network delay on the E2E lag is almost completely masked. Only if both rates are high enough, the network delay will play a more significant role. This masking effect has large implications for video games and their evaluation. Many evaluations solely examine the influence of the network delay, without any consideration to other contributing factors. Our results indicate that this might not be the best course of action. The masking effect likely shifts to lower values of the frame- and tickrates when a higher network delay is examined.

4.3. Cloud Gaming

The final simulation effort encompasses a cloud gaming scenario. Constant encode and decode delays e, d are in place at the game streaming server and client respectively. The frames are now generated by the server and need to be transported back

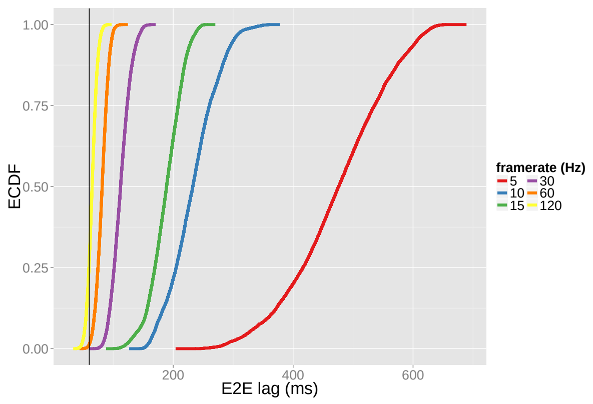

Figure 5: Influence of the rendering and streaming framerate on the E2E lag in the cloud scenario (§4.3). The vertical reference line denotes the average server processing time, network round-trip and codec delay $\mu_P + 2\mu_D + e + d = 68\,\text{ms}$.

to the client first. This is emulated by adding one frame time to account for the transmission of the encoded screen contents. The network is modeled as before. The command rate c is set to 200 Hz. Figure 5 shows the results of this scenario as ECDFs of the E2E lag for several framerates. As before, the framerate impacts the E2E lag more severely than the network delay does. This result is of particular interest considering how past studies have relied on similarly low framerates as $5 - 15\,\text{Hz}$ when assessing the network influence on cloud gaming QoE. Similarly, these results can provide guidelines for implementors of cloud gaming to factor in the framerate accordingly.

5. Conclusion

This paper presents a model for End-to-End (E2E) lag in video games, including online and cloud variants. The E2E lag represents the time elapsed between a player input event such as mouse movement or keystrokes and the display of the event’s results in the game on the local display. This lag is a main governing factor for Quality of Experience (QoE) in human-computer interaction in general, and video games in particular. The model is parameterized on the command rate at which batched user events are forwarded, the server tickrate and state processing time, the game’s local framerate, the network delay (for online games), and codec delays (for cloud games).

The model is simulated using Discrete Event Simulation (DES), showing the dominant influence of the game framerate on the E2E lag particularly for low framerates. It may even mask the influence of network delay, yet it appears underrepresented in previous work. On an abstracted level, the model helps to explain the mechanics behind lag in different game types and architectures. This is of interest to both actual implementations of games and study design for game QoE assessment.

Going forward, the model could be included in larger QoE frameworks; also, an analytical approach may provide further structural insights, and other lag sources such as input and output devices could extend the model. Lastly, the model should be validated in a practical setting.

Note: To foster participation and independent replication, the model simulation code is available as free, open-source software from the authors’ repository [2], as are the raw data.

6. References

[1] F. Metzger, A. Rafetseder, C. Schwartz, and T. Hoßfeld, “Games and frames: A strange tale of qoe studies,” in *Proceedings of the 8th international conference on quality of multimedia experience*, ser. QoMEX 2016, Jun. 2016.

[2] *A free, open-source lag simulation for online games*, `https://github.com/mas-ude/onlinegame-lag-sim/`.

[3] M. Jarschel, D. Schlosser, S. Scheuring, and T. Hossfeld, “An evaluation of QoE in cloud gaming based on subjective tests,” in *Innovative mobile and internet services in ubiquitous computing (imis), 2011 fifth international conference on*, Jun. 2011, pp. 330–335. DOI: `10.1109/IMIS.2011.92`.

[4] K.-T. Chen, P. Huang, and C.-L. Lei, “Effect of network quality on player departure behavior in online games,” *Parallel and distributed systems, ieee transactions on*, vol. 20, no. 5, pp. 593–606, May 2009. DOI: `10.1109/TPDS.2008.148`.

[5] M. Ries, P. Svoboda, and M. Rupp, “Empirical study of subjective quality for massive multiplayer games,” in *Systems, signals and image processing, 2008. iwssip 2008. 15th international conference on*, Jun. 2008, pp. 181–184. DOI: `10.1109/IWSSIP.2008.4604397`.

[6] V. Clincy and B. Wilgor, “Subjective evaluation of latency and packet loss in a cloud-based game,” in *Information technology: New generations (itng), 2013 tenth international conference on*, Apr. 2013, pp. 473–476. DOI: `10.1109/ITNG.2013.79`.

[7] S. Möller, J.-N. Antons, J. Beyer, S. Egger, E. N. Castellar, L. Skorin-Kapov, and M. Sužnjević, “Towards a new ITU-T recommendation for subjective methods evaluating gaming QoE,” 2015.

[8] M. Claypool and K. Claypool, “Latency and player actions in online games,” *Commun. acm*, vol. 49, no. 11, pp. 40–45, Nov. 2006. DOI: `10.1145/1167838.1167860`. [Online]. Available: `http://doi.acm.org/10.1145/1167838.1167860`.

[9] G. Armitage, “An experimental estimation of latency sensitivity in multiplayer quake 3,” in *Networks, 2003. icon2003. the 11th ieee international conference on*, Sep. 2003, pp. 137–141. DOI: `10.1109/ICON.2003.1266180`.

[10] K. Claypool and M. Claypool, “On frame rate and player performance in first person shooter games,” English, *Multimedia systems*, vol. 13, no. 1, pp. 3–17, 2007. DOI: `10.1007/s00530-007-0081-1`. [Online]. Available: `http://dx.doi.org/10.1007/s00530-007-0081-1`.

[11] M. Bredel and M. Fidler, “A measurement study regarding quality of service and its impact on multiplayer online games,” in *Proceedings of the 9th annual workshop on network and systems support for games*, ser. NetGames ’10, Taipei, Taiwan: IEEE Press, 2010, 1:1–1:6. [Online]. Available: `http://dl.acm.org/citation.cfm?id=1944796.1944797`.

[12] Z. Ivkovic, I. Stavness, C. Gutwin, and S. Sutcliffe, “Quantifying and mitigating the negative effects of local latencies on aiming in 3d shooter games,” in *Proceedings of the 33rd annual acm conference on human factors in computing systems*, ser. CHI ’15, Seoul, Republic of Korea: ACM, 2015, pp. 135–144. DOI: `10.1145/2702123.2702432`. [Online]. Available: `http://doi.acm.org/10.1145/2702123.2702432`.

[13] C. Ware and R. Balakrishnan, “Reaching for objects in vr displays: Lag and frame rate,” *Acm trans. comput.-hum. interact.*, vol. 1, no. 4, pp. 331–356, Dec. 1994. DOI: `10.1145/198425.198426`. [Online]. Available: `http://doi.acm.org/10.1145/198425.198426`.