2018 30th International Teletraffic Congress

# Exploring the Transmission Behaviour of Overwatch: The Source of Lag

Florian Metzger
Chair of Communication Networks
University of Würzburg, Germany
florian.metzger@uni-wuerzburg.de

Roman Heger
University of Duisburg-Essen, Germany

***Abstract*—This paper explores the full chain of lag contribution factors in a specific online multiplayer game, namely Overwatch: From the creation of input events, over the network, and back to displaying the results on the local screen. Together they result in the dreaded end-to-end lag, which has a direct impact on the subjective quality one experiences when playing video games. In its investigation, this paper reveals surprising effects in the game's networking behavior that are omitted when colloquially talking about, e.g. a $60\,\mathrm{Hz}$ update rate, but must be considered nonetheless. These insights, gained from examining network traces of Overwatch matches that were played on a realistic, resource constrained PC, can then be used to refine end-to-end lag simulation models and reach a better understanding of all responsible lag components.**

## I. Introduction

In theory, video games are simple: Retrieve an input, process it and update the game state, then display the results. But in praxis this gets a lot more complicated through the dynamic behavior of all involved factors. Especially when you add competitive multiplayer in to the mix and must synchronize the game states of the (authoritative) game server and each of the clients. Each of these components can then contribute to the end-to-end (E2E) lag, which will have an impact on each players' interactivity, and such on the perceived quality of the game as well [1], [2].

As to the extent of the impact of this lag on the perceived quality, many different factors can play a role, not just, as originally thought, the genre of the game in question [3], [4]. In theory, the lag components of a video game can be measured [5] or modeled in simulation [6]. But in praxis the involved processes can get a lot more dynamic than stated in basic theory. The resource-constrained nature, where the video game can not maintain a stable frame rate, can play a major role. Most research in the past has simply assumed ideal conditions on this matter. Sufficient CPU and GPU resources to maintain, e.g., a frame rate of $60\,\mathrm{Hz}$, as well as sufficient server resources to handle all of the users' input events. Yet, in reality this is often not the case. According to the Steam Hardware Survey[1] the typical PC video game player tends not to have the latest or most powerful iteration of hardware components. But she still wants to turn up the game's settings to enjoy a better graphical fidelity, often at the cost of frame rate. This resource-constrainedness affects all associated components relevant to the E2E lag, including the command send rate from the game client to the server. Similar variations will occur at the server's side and impact the game state update transmission rates to the game clients. It should be noted, that there are many anecdotes of competitive players doing the exact opposite and reduce graphical details to achieve both less visual distraction and better performance.

This paper explores such a resource-starved case and its effect on the popular team-based first-person shooter Overwatch on the PC. We recorded several game sessions and on this basis examine and model all factors that influence the lag. These models are then used to refine the E2E lag simulation of [6], of which this work is a direct follow-up. By their very nature, the results are limited to this specific case and can not necessarily be easily generalized. And yet, the conditions in this experiment are not particularly rare. The game's ideal behavior is just that. Ideal, intended behavior in an environment where everything works perfectly. But due to the large variety in computer hardware and the fact that games often push the given hardware to their limits — to produce good image quality at acceptable performance levels — video games often run into resource limits. And this what makes the study of such corner cases nonetheless worthwhile.

## II. Examining Individual Lag Components

In any video game a multitude of cogs is working in unison to transform a player's inputs into meaningful actions inside the game world. A simplified depiction of the processes a player action must go through before finally being to able to be displayed in a multiplayer game with an authoritative server is given in Fig. 1.

Besides additional lag incurred through the input and output devices' hardware (i.e. in this case mouse, keyboard, and monitor) — which can on their own still be significant[2]. The components (and the corresponding terminology) investigated here are outlined in [6], and are

- the generation of the input events by the player,
- the command message send process (from the client to the server),

[1] https://store.steampowered.com/hwsurvey/

[2] See, e.g. the investigation of keyboard latency at https://danluu.com/keyboard-latency/, in some cases exceeding $50\,\mathrm{ms}$.

978-0-9883045-5-0/18/$31.00 ©2018 ITC
DOI 10.1109/ITC30.2018.00022

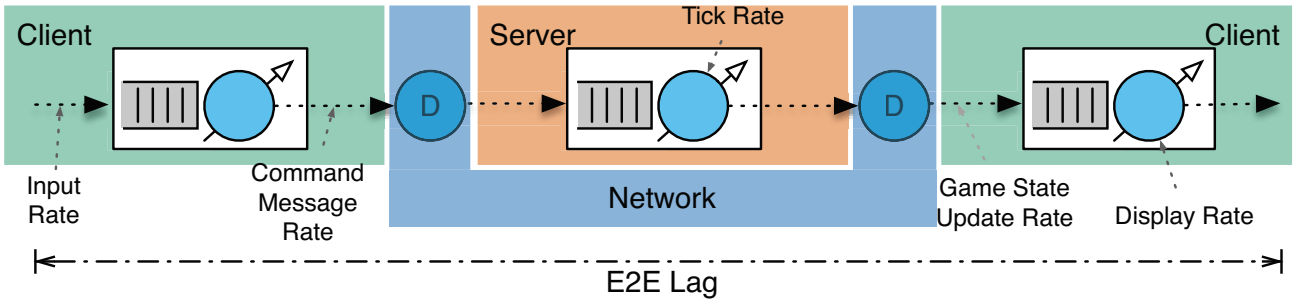


Figure 1. End-to-end lag model used as basis for this investigation. Several processes are assumed to be clocked, denoted by the diagonal arrows.

- the network round-trip time,
- the server's game state update rate (i.e. the so-called "tick rate"),
- the server's update send rate (server to clients),
- the client's frame rate (and additionally other lag-inducing factors at the client).

It should be noted, that there are usually lag compensation techniques [7] in place that can conceal parts of the lag from the player by speculatively displaying the player's action's results before being confirmed by the remote server. The specific measures that Overwatch takes have not yet been further examined. For this experiment several regular matches (i.e. Overwatch's default mode with two teams of six players each) were played from the perspective of one player on one PC that can not maintain a stable frame rate of 60 Hz throughout the match. Packet traces of the game's incoming and outgoing encrypted UDP traffic were recorded on the same PC and are the basis for this examination.

The code for models derived here as well as the refined lag simulation for this game is available in the public repository that hosts this paper.[3] The following sections each examine one of individual behavioral components of the game, as shown in the overview in Fig. 1.

### A. Command Messages

The command messages are the messages the game clients sends to the server, containing all the player's inputs during that cycle. The developers of OVERWATCH advertise a tick rate (i.e. the frequency the server updates its game state) of 60 Hz (or an Inter Arrival Time (IAT) of about 16.7 ms. Therefore, one would assume that the game's transmission behavior targets the same values to incorporate all player decisions (i.e. inputs) in each update cycle and notify them of the resulting game state.

However, when looking at the density of the command message IATs in Fig. 2, actually two modes are revealed, one at around 24 ms (or a send rate of 42 Hz) and the other one at 12 ms (83 Hz). Even though the overall mean IAT is 17.9 ms and thus just slightly longer than required for the potentially targeted transmission rate of 60 Hz.

[3]https://github.com/fmetzger/overwatch-lag-model

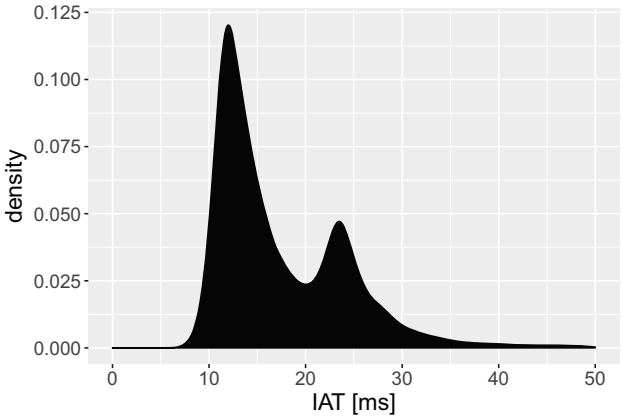


Figure 2. Continuous density plot of the command messages the client sends, showing a bimodal behavior instead of the expected single mode at 16.7 ms.

These modes are not distributed uniformly across the match but are rather separated into distinct transmission phases as the time series in Fig. 3 reveals. Here, the duration of the match can be divided into two alternating phases. The first phase reveals a (weak) third mode centered around 16.7 ms, i.e. the expected behavior of the game to send its inputs at a rate of 60 Hz to the server. But in the second phase, the game sends with both the IATs observed in Fig. 2 at a ratio of about four to one in favor of the 12 ms intervals. The exact reason for this behavior is, as of yet, unknown, but could possibly be attributed to the resource-constrained nature of the experimentation PC. This bimodal behavior could not be observed on a second, sufficiently dimensioned, PC.

In order to model this behavior, the trace was split at those phases and each phase modeled separately. The 60 Hz phases were fitted to a Gamma distribution with $\Gamma(\alpha = 4.437922, \beta = 208.366)$, the second phases were mixed by a Gamma distribution $\Gamma(\alpha = 49.32119, \beta = 3970.154)$ for the 83 Hz portion and a Normal distribution $\mathcal{N}(\mu = 0.02359103, \sigma = 0.001369922)$ for 42 Hz). Finally, the phase lengths are giving by a set of two Normal distributions. $\mathcal{N}(\mu = 28.85714, \sigma = 1.216385)$ models the length of the unimodal 60 Hz phase, and $\mathcal{N}(\mu = 40.83333, \sigma = 2.054805)$

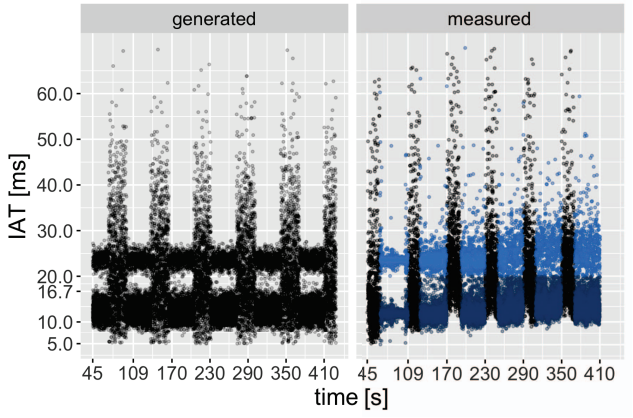


Figure 3. Time series of the client-sent command messages both measured in the experiment as well as generated from the model. The different phases are highlighted in different colors.

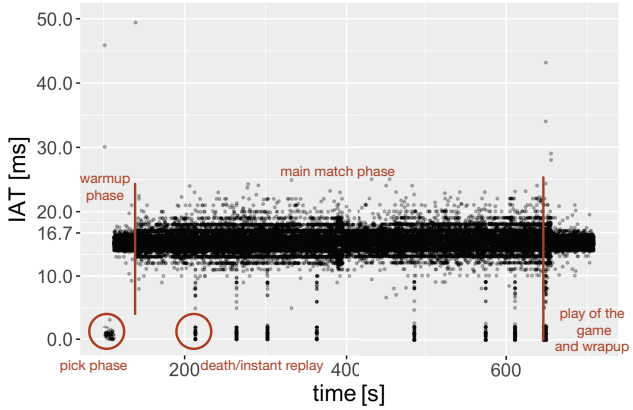


Figure 4. Time series of the received game state update messages. The game phases and non-interactive "kill cam" events are annotated.

gives the length of the bimodal phase. A random sample generated with this model is depicted in the left panel of Fig. 3.

### B. Game State Update Messages

The next lag component is the distribution of the game state update messages that the game server sends periodically to each connected game client. Ideally, such an update should be transmitted immediately after every game simulation "tick". A time series of the client's packet reception IATs of one match is depicted in Fig. 4. These properties clearly indicate that the server's messaging behavior depends on the game state, even though the tick rate targets a rate of 60 Hz.

The first block of very short IATs directly maps to the pick phase of OVERWATCH, where each player picks the hero (or class) she is going to play during this match. The next, short phase (visible by less spread in the samples) is the warmup phase, where both teams can not yet leave their respective starting areas and thus no meaningful action can occur. Only after this phase the actual game starts and the spread of the

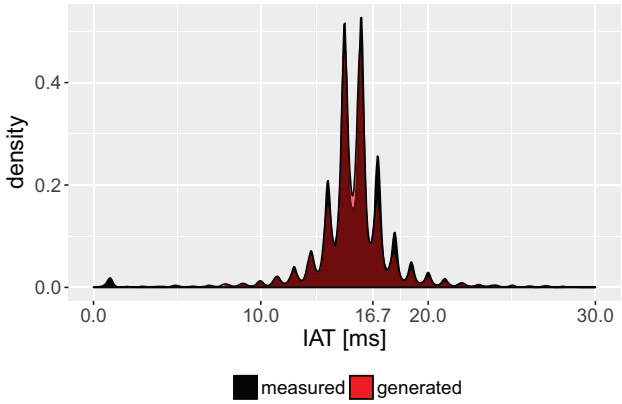


Figure 5. Density plot of both the measured and the fitted game state update messages received by the client.

IAT increases, possibly hinting at increased load at the server side. Similar the final phase of less sample spread depicts the game after the match has ended and the players see match statistics as well as a replay of a noteworthy event in the match (the "Play of the Game"). Every now and then the data exposes a quick burst of data in short intervals (i.e. the vertical bars in Fig. 4). This directly correlates to the so-called "kill cam". When the player dies during the match, and before she respawns, she is shown a quick replay of the event from the perspective of the shooter. Thus, this replay data needs to be transferred to the player first. This feature can also be turned off, eliminating the burst of data alongside. The reason for the grouping on the vertical axis into distinctly visible horizontal bars is not entirely clear yet, but is not necessarily present on every client PC. A quick check with the game running on another, more powerful computer did not exhibit this behavior. Nonetheless, it is present here and certainly influences the E2E lag and therefore should be considered in the model. Due to this oscillating behavior (cf. the density plot in Fig. 5) the update messaging process could best be modeled with a Cauchy distribution that has additionally been modulated with a sine function. This yields

$$f_{rcv}(x) = \frac{0.7\sin(171.6 + 3120x)^4 + 0.2}{0.9848153} f(x; x_0; \gamma)^{1.17}$$

for the density, with location $x_0 = 0.0155$ and scale $\gamma = 0.0011$ for the Cauchy distribution. It should be noted that this model is only applicable for the game's main phase and excludes the "kill cam"-events as well. Those are either not relevant to the outcome of the match (warmup phase and post-match statistics) or are entirely non-interactive for the player.

### C. Input Events

Another component is the distribution of the actual input events that the player generates with her input devices. The original simulation in [6] just assumed an exponential distribution with a rate that might not be entirely realistic for an actual game. Therefore, for OVERWATCH the inputs were

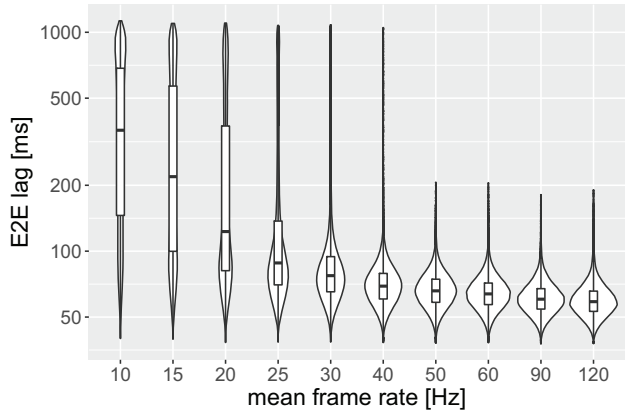


Figure 6. E2E lag results of a frame rate study using the OVERWATCH simulation. The denoted frame rates are the mean of the normal distribution.

recorded and a rate parameter for the exponential distribution was derived as a compound of button strokes and mouse movements ($\lambda = 102.1824$).

### D. Network Delay

Finally, the network delay must also be considered. But due to the high reliability of servers and the matchmaking that should place the player on to a server in the player's vicinity, the delay to the server should be low and stable. In a separate RTT measurement to the game server's IP address found in the investigated matches the RTT was about $34\,\mathrm{ms}$. This is used as a constant network delay component in the simulation. The low variance of the measured RTT does not justify a more complex distribution here. The actual value is of course highly dependent on, e.g. the player's geographical location and could be changed to any other arbitrary value, yielding a minimum value for the total end-to-end lag.

## III. OVERWATCH LAG SIMULATION

Now that all client-observable parameters have been examined, a simulation study is performed on the effects of the game's frame rate. This the only direct game parameter that the player can influence through the game's setting or through more capable hardware that influences the lag. But, since it is often tricky to maintain a stable frame rate in such resource constrained environments, the frame rate in this simulation is modeled as a normal distribution instead of a fixed value. This should account for the variations in the frame times (the time between two consecutive frames). Investigated here were frame rates between $10\,\mathrm{Hz}$ and $120\,\mathrm{Hz}$. The minimum value might occur in high stress situations when resource are severely constrained, and the maximum value representing what can be achieved with modern PC hardware and monitors. All other simulation parameters are derived from the models in the previous section or left as-is in the base simulation.

The results are depicted in Fig. 6. Starting at about $50\,\mathrm{Hz}$ the E2E lag reduction sees diminishing returns especially towards its outliers, which can exceed a lag of over a second below $50\,\mathrm{Hz}$. This correlates quite well with the generally accepted notion that (especially) online first person shooters require at least $60\,\mathrm{Hz}$ for an enjoyable experience. In praxis, only $30\,\mathrm{Hz}$, $60\,\mathrm{Hz}$ or higher frame rates will be targeted, due to otherwise occurring interactions with the monitor's refresh rate, which results in either screen tearing or an unstable IAT of the rendered frames.

## IV. CONCLUSION

This lag model for OVERWATCH is highly customized to a specific case of resource limitation and unfortunately not a one-size-fits-all solution for every game. But it is indicative of issues in such environments, because components relevant to the lag might negatively interact with other processes, e.g., the renderer, which can not keep its intended output rate, and thus causes the game's networking process to misbehave. Therefore, it gives much needed insight into the dynamics of video games and the deviations from the theoretical properties that are actually occurring in praxis. This specialized simulation demonstrates quite well the influence of the game's frame rate and input and messaging processes investigated here on the E2E lag, despite operating under near-ideal network conditions.

## REFERENCES

[1] M. Claypool and K. Claypool, "Latency and player actions in online games," *Commun. ACM*, vol. 49, no. 11, pp. 40–45, Nov. 2006.

[2] S. Schmidt *et al.*, "Towards the delay sensitivity of games: There is more than genres," in *2017 Ninth International Conference on Quality of Multimedia Experience (QoMEX)*, May 2017, pp. 1–6.

[3] S. Möller *et al.*, "Towards a new ITU-T recommendation for subjective methods evaluating gaming QoE," pp. 1–6, May 2015.

[4] I. Slivar *et al.*, "Cloud gaming qoe models for deriving video encoding adaptation strategies," in *Proceedings of the 7th International Conference on Multimedia Systems*, ser. MMSys '16, Klagenfurt, Austria: ACM, 2016, 18:1–18:12.

[5] J. Beyer *et al.*, "A method for feedback delay measurement using a low-cost arduino microcontroller: Lesson learned: Delay influenced by video bitrate and game-level," in *2015 Seventh International Workshop on Quality of Multimedia Experience (QoMEX)*, May 2015, pp. 1–2.

[6] F. Metzger *et al.*, "A comprehensive end-to-end lag model for online and cloud video gaming," in *PQS 2016 5th ISCA/DEGA Workshop on Perceptual Quality of Systems*, 2016, pp. 20–24.

[7] Y. W. Bernier, "Latency compensating methods in client/server in-game protocol design and optimization," in *Game Developers Conference*, vol. 98033, 2001.