An Introduction to Online Video Game QoS and QoE Influencing Factors

Florian Metzger, Stefan Geißler, Alexej Grigorjew, Frank Loh,
Christian Moldovan, Michael Seufert, Tobias Hoßfeld
University of Würzburg, Chair of Communication Networks, Würzburg, Germany
Email: <firstname>.<lastname>@uni-wuerzburg.de

***Abstract*—Online video games and cloud gaming are rapidly growing in pervasiveness. Their resource demands can put significant stress on the global communication infrastructure. And network conditions are amongst the chief factors that influence one's enjoyment while playing games. This makes it imperative for video games to be considered for network dimensioning, server placement or protocol development.**

For that reason, in this work we provide an introduction to the technical aspects of video games in general and of their network aspects in particular. This understanding forms the basis for a rich taxonomy of factors that influence and provide context to a video game's Quality of Service (QoS) and Quality of Experience (QoE). The taxonomy covers influence factors from all aspects involved in a video game, from the subjective player and game influence factors to the system and networking influence factors. Finally, this work gives an overview of conducted and ongoing research as well as future research opportunities while taking into account lessons learned from past approaches.

I. Introduction

Video games, and online video games in particular, have moved from being a niche hobby to being considered a mainstream media. In 2019 alone, the gaming industry generated $120 B in revenues with an audience of almost one billion people [1]. During the 2020 pandemic year the growth intensified, not only driven by global social distancing measures but also by the launch of a new console generation. For example, the US gaming market saw a 27 % increase [2]. Apart from the economic perspective, the online video game growth poses significant challenges and opportunities for communication networks. Cisco projected that gaming Internet traffic will grow ninefold between 2017 and 2022 [3]. This is aligned with the recent trend towards commercial Cloud Gaming solutions [4]. All these developments make online and cloud gaming an important and relevant research area. To accommodate these new and complex services, networks have to be scaled, algorithms and protocols have to be adjusted, user behavior has to be analyzed, and cloud environments have to be improved to deliver the ideal gaming Quality of Experience (QoE) to end uses.

Online video game developers and Internet Service Providers (ISPs) want to deliver the best possible gaming experience using the available physical network structures. In today's landscape, consisting of numerous and heterogeneous network providers and access technologies, it is a complex challenge for video game developers and publishers to provide acceptable service quality—i.e. Quality of Service (QoS)—to all users. The experience can degrade if any of the involved components at any one of the operators start to misbehave or become overloaded, potentially leading to increased customer churn rates and eventually also impacting revenues [5].

In addition to these technical challenges, the heterogeneous user base and gaming environments as well as the subjective perception of games reinforce the complexity. Not only do different types of games pose different requirements to the hardware and available network resources, but every player will have different tastes, access to different hardware and platforms as well as different past experiences and skills. All of these factors will also influence the subjective quality as perceived by the players—i.e. QoE [6]. In order to be able to deal with any potential degradation or issue and to generally improve both QoS and QoE of online video games it is crucial to understand all involved aspects. Only then, detailed and general models can be developed and specific scenarios, influence factors, and system interactions can be evaluated. The resulting research challenge is the *development of solutions to evaluate and improve both the QoS and QoE* for video games with network involvement. To this end, research questions concerning each aspect of the networked gaming setup need to be examined. On the game provider's side, problems like resource provisioning, scaling, load balancing as well as game distribution have to be solved. From a network perspective, operators have to ensure adequate bandwidth and delays for these highly interactive applications. Finally, at the users' side, the interplay between the received QoS and the resulting QoE, user behavior, and satisfaction need to be analyzed. The heterogeneous landscape with multiple stakeholders makes the evaluation of QoS and QoE in video gaming scenarios a non-trivial task.

In order to address these challenges and to provide a point of entry for research into this highly complex topic, the contribution of this work is *(i) to provide a general taxonomy of influence and context factors of the QoS and QoE of video games and, based on this taxonomy, (ii) to provide a comprehensive introduction into the inner workings of networked video games as well as an introduction to video game QoE research.*

To the best of our knowledge this is the first work that sheds light on the video game ecosystem as a whole. It includes both online video games as well as cloud gaming and offers a combined overview over the technical, networking, and subjective aspects. But this also means, that certain aspects

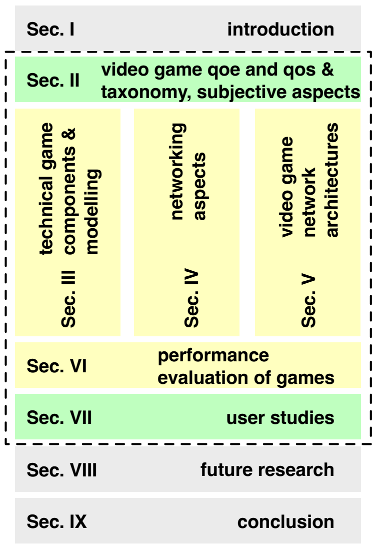

Sec. I
introduction
Sec. II
video game qoe and qos & taxonomy, subjective aspects
Sec. III
technical game components & modelling
Sec. IV
networking aspects
Sec. V
video game network architectures
Sec. VI
performance evaluation of games
Sec. VII
user studies
Sec. VIII
future research
Sec. IX
conclusion

Figure 1: Structural overview of this manuscript. Dashed box indicates the main contents of this work. Sections in green contain mostly subjective aspects, yellow sections primarily concern objective and technical aspects.

can not be treated with the utmost depth in this manuscript, as this would result in a full-length book. Instead, we will direct the interested reader to more in-depth material on those particular topics. Other surveys and tutorials in this area are often of a narrower focus, and can additionally become outdated rather quickly due to the rapidly developing field of QoE in cloud gaming. The technical aspects of cloud gaming services, the makeup of modern online games but also the users' expectations towards services have drastically changed today, thus inviting a fresh perspective on the field while not disregarding established fundamentals. Previous works include several general and QoE-focused publications on Cloud Gaming, including [7]–[9]. Of these, only the latter has been written with the current development of cloud gaming in mind (as of 2021). Other works focus solely on aspects of online video games and not on cloud gaming. Here, [10], [11] can give a good, albeit rather old, academically focused overview on online gaming, while [12] is written from the perspective of a game developer. To our knowledge, no other work attempts to cover all of these aspects and their recent developments together under one umbrella.

In order to facilitate our contribution, this work is split into portions that focus more on tutorial aspects of explaining certain video game relationships, while others primarily serve as survey to past and current research. These are further split into technical and subjective aspects. Figure 1 illustrates our approach. Section II presents a detailed taxonomy of influence factors of different categories, derived both from the knowledge of previous QoS and QoE research and from a fundamental comprehension of online video games. The taxonomy itself is illustrated in Figure 2, and serves as the conceptual reason for how we split this work up.

The subsequent sections guide through the aspects described in the taxonomy, starting with a brief overview of player and game influence and context factors in Section II-B. Section III then introduces technical and system aspects of video games, including a simple model representation of video game systems. Section IV is dedicated to introducing and exploring all network-related aspects of online and cloud gaming, while Section V focuses solely on covering their different networking architectures. Based on all these insights, the following two sections are concerned with the evaluation and assessment of QoS and QoE. Section VI covers the performance evaluation of games and system factors that can influence them. In particular, a series of measurement points are introduced that are able to cover specific influence factors. Section VII encompasses a survey of game QoS and QoE evaluation metrics, and how they are used in past video game user studies. With all this information compiled together, we formulate future challenges and open research tasks in Section VIII. Finally, Section IX concludes this work.

II. Video Game QoS and QoE

To start any discussion about QoE one has to first agree on a common understanding of what that term means for video games. QoE is often described as the acceptability of a service, or as the perceived, i.e. the experienced, quality of a service. A prominent definition is for example given in [6]. Common to all definitions of QoE is the subjective opinion component that distinguishes them from purely objective QoS metrics. The existing definitions of QoE are generic and unspecific to any topic and can thus be applied to video games as well.

However, there is also some overlap in the definitions of QoS, QoE and User Experience (UX). The overlap of UX (which also deals with interaction design and quality) and QoE is a broader subject of discussions and not a topic of this work [13]. For this work we can assume that QoE is a broader term, that deals with matters of interaction quality in addition to, e.g., image quality assessment.

Video game QoS metrics include measurable interactions of players with a game, such as task completion times or highscores. They are therefore also called *application layer* QoS factors or *player performance metrics*, as they describe factors directly associated with a game itself and not with some lower layers as other QoS metrics would. The existence of player performance metrics in many video games makes it often much easier to perform user studies solely based on these performance metrics, instead of directly assessing subjective QoE. While factors of the lower layers are directly influencing QoE factors, the relationship of application layer QoS with the QoE is not fully understood yet. There are no good models yet that would, for example, map a specific highscore to a subjective rating of that game. And just performing well in a game does not necessarily imply that you actually enjoy playing the game.

Two further aspects differ when it comes to interactive applications in particular. Compared to passively consumed media, the interactive nature of video games brings a number of additional system, human and contextual factors along that need to be taken into account. Second, the approaches to assessing video game QoE are more challenging, as the range of system parameters exceeds those of, e.g., video streaming. Video games implement very diverse and often complex sets of interaction mechanics, and there is no one-size-fits all assessment approach yet that would cover all aspects at once. This becomes even more complex when distinguishing between online video games and cloud gaming (more on that later). Cloud gaming introduces further spatial and temporal complexity through its RTP-based video streaming in addition to any other gaming QoE factor.

All of these circumstances and open issues have been recognized by the QoE community and are being worked on, e.g. see [14], [15]. Our work intends to contribute to these ongoing discussions. The taxonomy introduced in this section—in conjunction with the subsequent sections that explain and discuss the concrete taxonomy factors—tackles the issue of a common understanding of video games and influencing aspects for the purpose of video game studies. The second issue—of surveying video game assessment approaches—is the focus of the latter half of this manuscript. In that second half, Section VI highlights game performance metrics and where and how to measure them. These mostly center around the influence of lag and other network QoS factors, their sources and measurement approaches. And Section VII highlights QoE assessments performed in the past as well as general QoE assessment methodologies.

A. A Taxonomy of Video Game QoS and QoE Factors

Figure 2 depicts our approach at a cohesive taxonomy of direct and indirect video game QoE and QoS influencing factors, and forms the basis for the subsequent sections. The taxonomy consists of five larger factors that each represent a separate entity in video gaming: the *player*, the *game*, the *game client* (which runs the actual, user-visible game), the *game server* (runs and coordinates the backend of multiplayer online games in a centralized approach), and the *network*. Each factor can be drilled down further to more specific influencing properties of that entity. The figure gives concrete influencing factor examples for each of them. The separation of the game client into local and remote factors—and for example not just a differentiation between client and server—becomes necessary due to cloud gaming. Here, the game client can also be a remote entity (with its output provided to the user in the form of a video stream), but is otherwise functionally identical to a locally running game. Depending on whether the game client is used for cloud gaming or not, different influencing factors apply that would otherwise not.

In turn, the factors are allocated into three areas, covering subjective and context aspects (continued in Section II-B), technical and system aspects (continued in Section III), as well as networking aspects (continued in Section IV). The assignment is not unambiguous, as most of the factors contain aspects of several areas. For example, a game has several system aspects affixed to it with regards to its design and implementation of specific elements, but their perception and reception is subjective to the individual, and may thus result in different experiences. By this it is also placed in a larger context and environment.

The basic purpose of this taxonomy is to highlight all aspects and types of influencing factors that potentially impact the assessment of video game QoE. It specifically contains many factors that may not be immediately evident to be an influencing property during assessment. However, these still need to be considered, and may be the deciding parameter, e.g., for the creation of high-fidelity video game QoE models. Such QoE models are of high interest for the research community, especially when it concerns cloud gaming. Various approaches are already underway, e.g., by the ITU-T SG.12 in the form of a *Parametric bitstream-based Quality Assessment of Cloud Gaming Services* (P.BBQCG, [16]).

This is of course also not the first approach to such a taxonomy. Already the 2013 Qualinet whitepaper [6] debated three areas of influence factors (unspecific to video games): human, system, and context. Another 2013 approach to a video game taxonomy offered a separation of QoS and QoE factors and how they can be assessed with a questionnaire [17]. For a more recent and topical example, both ITU-T Rec. G.1032 [18] and Rec. P.809 [19] contain approaches with a focus on collecting game quality and QoE influencing factors. Our approach is aimed at extending all these prior approaches with additional factors and factor interactions that may not have been in the spotlight in the past, and explain why we think that they are important. We want to especially highlight concrete technical influence factors that may affect the QoS and QoE and have only been scarcely integrated into past taxonomies. The goal of this new taxonomy is also in line with those past approaches: to map out the landscape of video game parameters that are relevant for measurement experiments as well as user assessments, and to provide means to direct future research efforts towards factors that might not have been in the spotlight yet. Furthermore, we see this work as an additional contribution to currently ongoing efforts to standardize assessment and provide QoE models for cloud gaming [16].

B. Subjective and Context Aspects of Players and Games

The following sections discuss all areas of the taxonomy, supplemented with the necessary introductions into their respective topics. And the remainder of this section provides a brief overview of the aspects that focus on quality influencing factors directly belonging to players and their environment, as well as on factors belonging to an individual video game. This is purely meant as on overview over these subjective and context aspects and is in line with the approaches taken by prior taxonomies (e.g. [17]). Further information that pertains to the assessment of these factors is provided in Sections VI, VII and VIII.

1) The Player as a Quality-Influencing Factor: Player aspects and factors generally give context and meaning to the subject, as they describe a player and their past experiences, abilities, and preferences. However, this does not necessarily imply that these factors are only relevant for subjective studies. As video games are interactive applications some of these factors will also influence application layer QoS, such as the subjects' skills and past experiences with video games

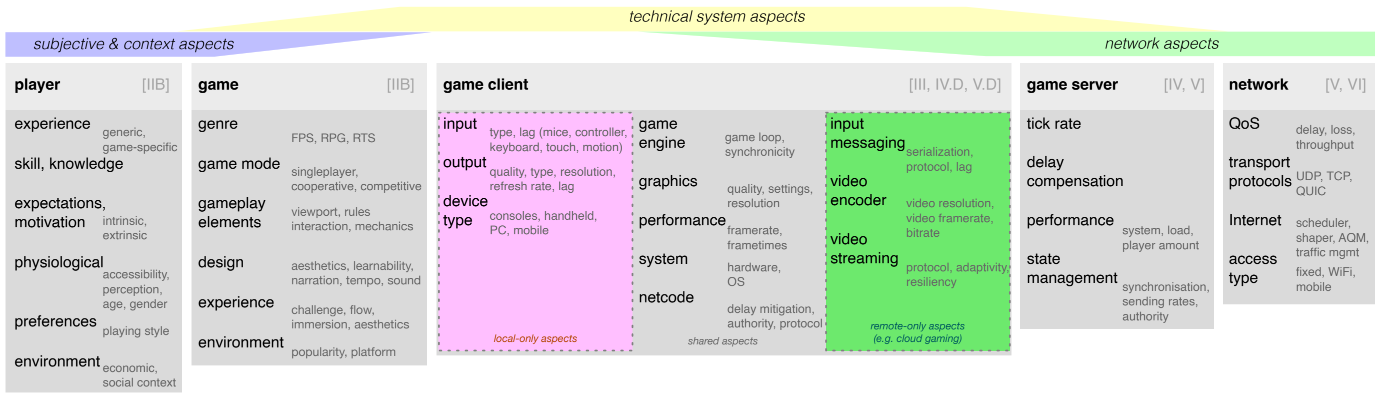

technical system aspects
subjective & context aspects
network aspects
player [IIB]
experience generic, game-specific
skill, knowledge
expectations, motivation intrinsic, extrinsic
physiological accessibility, perception, age, gender
preferences playing style
environment economic, social context
game [IIB]
genre FPS, RPG, RTS
game mode singleplayer, cooperative, competitive
gameplay elements viewport, rules interaction, mechanics
design aesthetics, learnability, narration, tempo, sound
experience challenge, flow, immersion, aesthetics
environment popularity, platform
game client [III, IV.D, V.D]
input type, lag (mice, controller, keyboard, touch, motion)
output quality, type, resolution, refresh rate, lag
device type consoles, handheld, PC, mobile
local-only aspects
game engine game loop, synchronicity
graphics quality, settings, resolution
performance framerate, frametimes
system hardware, OS
netcode delay mitigation, authority, protocol
shared aspects
input messaging serialization, protocol, lag
video encoder video resolution, video framerate, bitrate
video streaming protocol, adaptivity, resiliency
remote-only aspects (e.g. cloud gaming)
game server [IV, V]
tick rate
delay compensation
performance system, load, player amount
state management synchronisation, sending rates, authority
network [V, VI]
QoS delay, loss, throughput
transport protocols UDP, TCP, QUIC
Internet scheduler, shaper, AQM, traffic mgmt
access type fixed, WiFi, mobile

Figure 2: Overview of factors and aspects of the gaming environment that influence and provide context to QoS and QoE evaluations of video games. Concrete examples are depicted in a light gray font color. The provided section numbers direct to the primary sections where the corresponding aspects are discussed.

in general. Thus, these factors influence both the measurable player performance players have when interacting with a game, but also influence their impression of the game when doing so. However, the concrete relationship between player performance metrics and QoE has not yet been exhaustively investigated. This means, it is not clear if a high performance metric automatically implies that the player had a good subjective experience while playing, and vice versa. For example, losing a game has often been associated with increased frustration (see, e.g., [20]). But that does not describe the strength of the relationship, or even if it is always the case nor that frustration is necessarily a bad thing.

Among these influencing factors in our taxonomy, *experiences* relates to specific *skills* regarding specific game features as well as *expectations* on how certain game mechanics and features should work and feel.

Physiological factors directly describe the player themselves. These are the typical influencing factors of concern in subjective studies, e.g. age and gender, but include potential physical limitations and accessibility needs as well. For example, players with a form of color blindness might only be able to properly engage with games that provide appropriate accessibility options or players with repetitive strain injuries might tend to avoid fast-paced games with repetitive inputs. Thus, these physiological aspects can also describe how certain game mechanics and features feel to a certain individual.

Preferences describe factors like the game types a player tends to choose, or which game modes they prefer. Preferences may even include finer details, like the controller, typical graphical settings or type of gaming device they are comfortable with.

Environmental factors contextualize a subject's surroundings and, in our definition, also includes a wider range of *socio-economic* factors. These may include their social context, e.g. game recommendations by friends, economic factors, e.g. decisions whether they buy high-end PC hardware, full price games or prefer budget titles or subscription services, and also their physical surroundings during play, e.g. whether they play in their comfortable living room or mobile in public transport.

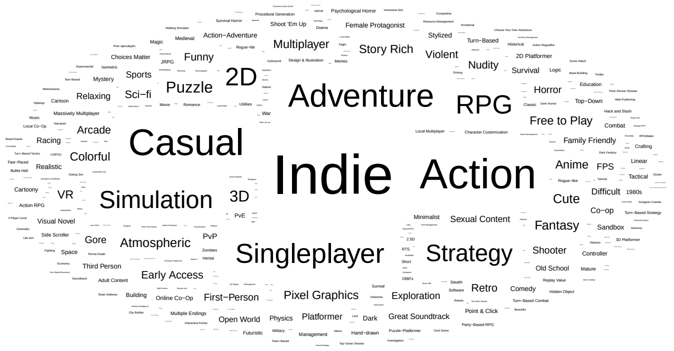


Figure 3: Tag cloud showing the diversity of tags attributed by users to games on the Steam platform. Font size scaled by the frequency of the tags (most used tag occurred 57617 times). Data collected via https://steamdb.info/tags/.

2) *The Video Game as a Quality-Influencing Factor:* Video games are made up of both technical as well as artistic design choices. While the technical aspects will be discussed in the subsequent sections and may explain significant portion of a resulting quality impression, artistic choices still can provide additional explanations and context for a specific result. Take the game *tempo* as an example. Slow games might be completely unfazed by high input delay or a low frame rate, while a fast game might be nigh unplayable under the same exact conditions. Thus, this warrants a separate discussion of these *game influencing factors* in this section.

The *genre*—e.g. a Real-time Strategy game (RTS), First-Person Shooter (FPS) or a racing game—can give a rough, albeit subjective, categorization of games, but does not necessarily govern more concrete factors pertaining to the games design and gameplay. Genres also do not follow a unified scheme on which aspects they describe, and range from describing technical properties (e.g., the camera viewport), economic aspects (e.g. "Indie" games) to artistic choices (e.g. a strategy game). Additionally, many games may belong to more than one genre. This is also highlighted by the diversity of tags seen in Figure 3, depicting all the genre tags that users associated with games on the Steam platform.

The available *game modes* describe aspects such as whether a game is played in singleplayer, cooperative or competitive mode, but also the number of concurrent players in a game session, as well as the difficulty and accessibility options the game provides.

Next, the actual *gameplay elements* are a group of factors on their own, and they describe which actions a player has to perform to solve the game's tasks and challenges, and which tools are given to them to perform these. These are often also dubbed game mechanics and may include elements of, for example, movement, combat or puzzle solving.

This directly leads to the *design* decisions taken to implement gameplay mechanics but also narrative and aesthetic aspects of the game. Among these, *precision* represents the temporal and spatial complexity of interactions within the game, randomness governs the need to react to unpredictable actions, and *tempo* measures the pace of interaction.

The classic *experience* factors describe one's background with a game and its design and gameplay elements—including challenge, flow and immersion—and are by definition subjective to a player and only captured by questionnaires or physiological measures.

Finally, *environmental* factors include a variety of different aspects, from a game's current popularity—and partly related to that, its potential appeal to new players—to its development budget and length, to the availability on different video game platforms and hardware requirements.

These insights into subjective and game-specific aspects are picked up again in the measurement experiment (Section VI) and subjective study sections (Section VII). Furthermore, these factors seamlessly lead over to the system influence factors and technical details of the game client in the following sections.

III. Technical Game Components and Modelling

This section provides a general understanding of technical and system influence factors. It explains the basic systems with their core components, introduces a simple model with key aspects for its interpretation and how they are used for communication between the systems. Thus, using the taxonomy's designations this section covers the technical system aspects of the game, client and server.

A. Core Components

Traditionally, video games were intended to be played on a single platform, to be self-contained and perform all tasks with no technical details exposed to the players. With the introduction of general purpose computers, the underlying platforms for video games began to diversify and games could no longer be reviewed independently of the deployed system. Further, when the Internet became available to a large playerbase, game developers leveraged this means of communication to enable multiplayer experiences over the network. This step massively increased the complexity of the games. They are no longer contained to a local system, but they now depend on external components that not only operate independently of the local core system, but also introduce further variation to the performance. The current condition of the network now has a direct influence on the perceived gaming experience, hence it needs to be fully understood.

1) Local game client: The local game client—or generally speaking, the local game system—always includes all *input* and *output* components that directly interface with the player. This includes general interfaces, such as mouse and keyboard, gaming-specific interfaces such as game controllers, means for communication such as microphone, headset and speakers, or various other motion controlling interfaces. The output usually is a monitor of various types, sizes, and other properties. Other variants are hand-held systems with built in displays, and head mounted displays that simultaneously capture the motion of the head.

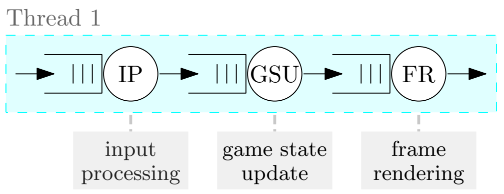
The *game engine* is the central piece of software that continuously computes new game states and presents them to the player. As the underlying systems may vary—different *systems* and *platforms* offer different types of CPUs, Graphics Processing Units (GPUs) and operating systems—the game engine usually provides means to adjust the game's graphical settings and by this also the game's *performance*, measured by it's frame rate. Such settings are necessary to ensure a good performance to all players and their different systems. Finally, if the game offers an online multiplayer mode, the underlying connectivity and the corresponding *netcode* (umbrella term for the concrete networking mechanisms implemented in a game, see Sec. III-C) are also implemented in the game client and represent important influence factors. Even if the local system can provide sufficient performance for the central game loop (cf. Section III-B), bad connectivity may still impact the experience severely.

2) External game server: If the game does rely on networking, the other side of the communication must be considered, too. The communication partner is often a dedicated game server that is responsible for a global game state computation. As such, it contains similar components as the local game client, without the graphics-related parts such as the GPU, and without direct interfaces for local input and output.

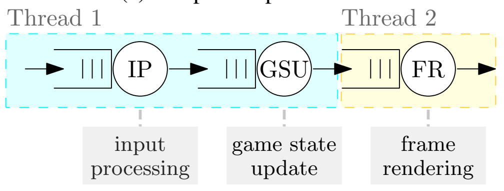
3) Cloud rendering and streaming server: Recently, the heterogeneity of local systems has lead to a trend that offloads computationally intensive tasks to another, external system. While the key components on the local game client remain the same by name, the local system's influence on the game performance is reduced such that almost any system with adequate input and output suffices. Instead, the actual client running the game now reside within a cloud server, and is responsible for game loop execution and frame rendering. It inherits the local client's system components and dependencies. The new local client is now heavily dependent on network connectivity. Even for single player games that were previously contained to their local system, since all input and output actions need to be exchanged between the local and remote system. The individual responsibilities of these three core components and their relations are covered in more detail in Sections IV, V, and VI.

B. Basic Video Game Model

The discussion of the individual steps of the game loop relies on a sequential model of processes. This model is



(a) Simple coupled model.



(b) Multi-threaded uncoupled model.

Figure 4: Basic models of a game engine's core loop. Adapted from [22].

introduced as a simple local queuing model here, and extended by external components in later sections.

In general, any video game, regardless of its genre, can be described as a discrete real-time simulation. A game is represented as a continuously running loop during which everything that is happening needs to be calculated, loaded, rendered, and displayed. The time available to perform these tasks at a high frequency is constrained by the need to display the output in real-time. Exceeding the available time may significantly reduce the playability and overall quality of the game [21]. The specific tasks performed within a game loop depend on the individual game and factors like the involvement of network communication.

In the most basic scenario, covered in this section and shown in Figure 4a—a locally rendered game with no network involvement—the loop consists of three steps: input processing (IP), game state update (GSU), and frame rendering (FR). During the first stage, input from the user is collected by querying attached peripherals. Based on user input and the current, discrete game state, the update stage is responsible for calculating the updated game state. This includes animation states, actor behavior, and the game logic itself. During the final stage, the updated scene is rendered and sent to a display in the form of an updated game frame [22].

In the simple coupled model (as depicted in Figure 4a and adapted from [22]) all three stages are executed sequentially in a single thread. In general, one stage must complete its computation before the information can be passed on to the next stage. However, the sequential nature of the model does not properly reflect the multi-tasking capabilities of modern platforms and could quickly run into performance problems, e.g., in fast-paced and technically complex games if those were actually implemented this way today. To this end, in order to leverage the multi-tasking capabilities of modern hardware, some processes can be executed independently and asynchronously and can have their own loop. For example, the game state could always update periodically, irrespective of whether there has been new input to process. Once there has been new input, this information is passed along as soon as the process is executed again. We refer to the latter as a *clocked process*.

Hence, the multi-threaded uncoupled model shown in Figure 4b is a more adequate representation of modern game systems, albeit still extremely simplified. It divides the loop into multiple threads that execute the rendering stage on the GPU asynchronously and independently of the CPU-bound game state update. More detailed descriptions of these and further models can be found in [22] and [23]. In the following sections, we will extend this simple system model to adapt it to different networking scenarios and include additional components required to properly model modern, networked video game systems.

C. Game Engine and Netcode

While it is impossible to cover every individual game's implementation, they are often based on generic *game engines* [24]. These provide pre-built libraries for the most common game functions. Most importantly, they provide (i) the means for 2D and 3D world construction and rendering with an underlying graphics library, (ii) interactions with this world such as moving around, and underlying physics such as gravity, (iii) serialization of world and object state, (iv) libraries that simplify or fully undertake sound, input, and data management, and finally (v) pre-built functions and interfaces that handle interactions with the network. The latter are often summarily called the *netcode*. What the game engine cannot provide is the actual game logic and how the game uses the components provided by the engine. Thus, while the engine is an influential factor, it is not solely responsible for the resulting game's behavior.

When reviewing the performance of games, those backed by the same engine often show similar technical system behavior. Previously mentioned configuration options are often provided by the game engine, and can be passed through to the player. However, it is common practice to adjust some functions and to provide individual implementations, effectively replacing the pre-built functions as required. For example, while most engines already provide basic functionality for online multiplayer many games still replace this with their own implementation that is better tailored to their needs of, e.g., fast-paced shooters with many simultaneous players. For an abstract understanding of the underlying behavior during performance evaluation, knowing the applied game engine may already be sufficient, but when individual scenarios involving game-specific logic are being reviewed, a more profound understanding of the individual game's implementation and logic is often required.

In particular, when reviewing the influence of the network on a game's performance, the underlying netcode and serialization techniques become important factors. In slowly paced games with a rich game state and game world, the effective data rate of the connection may actively influence the frequency at which actors and objects in the game load and are synchronized. Therefore, the game must provide efficient serialization techniques, and, e.g., only transfer state differences and only the state of objects that are currently of interest to the client. In fast paced games, mere milliseconds of delay can severely influence the resulting game state at the dedicated server. As a result, the netcode commonly includes mitigation techniques for delay and packet loss. The netcode is usually specifically designed for each individual game, and the implementation often depends on the game logic itself. For example, the game could prioritize the synchronization of objects close to the player and synchronize distant objects less frequently to improve performance. Fast paced games would apply various mechanisms that ensure that an action is performed as perceived by the local client. These mechanisms include giving the client more authority over its game states and let the client compute some game state updates by itself, estimate delays to re-apply an action in a previous state as intended by the client, or re-sending previous input commands at later game loop iterations in case of packet loss. During performance evaluation, the respective techniques and when they are applied by the game logic should be known. A more in-depth look at concrete networking mechanisms in games offers the subsequent Section IV.

In the era of cloud gaming, the network influences the perceived gameplay quality in a different way. Here, delay and bandwidth have a more direct influence on the output latency and quality. While the type of game often has a subjective influence on the experience, it should be understood that neither the game engine nor the netcode are directly related to cloud gaming performance. Instead, the cloud gaming system is a system separate from the actual game. It merely acts as a redirector of the game's inputs and outputs, similar to a remote desktop system. A local client only records user inputs and records them to a remote server where the game is running and rendering its outputs. These outputs are encoded into a video stream, sent back to the client and decoded there. Thus, the influencing factors of the cloud gaming system apply in addition to the game that it is running.

It is also possible to experience the effects of both cloud gaming and game engine networking at the same time, when multiplayer games are being rendered on a cloud server. The above game loop models from Figure 4 will be extended with these networking effects in the remainder of this paper.

IV. Networking Aspects

The following section describes networking aspects of video games, starting with general uses of networking in games as well as aspects on different layers, outlining the direct as well as indirect use of network communication during gameplay.

A. Networking in Games

Networking in modern games is crucial in multiple ways, and the traffic generated during multiplayer gaming is influenced by many aspects of the game itself as well as the surrounding environment. Network usage in modern games can be roughly divided into two blocks, which are discussed in the following: (i) direct and (ii) indirect gameplay communication.

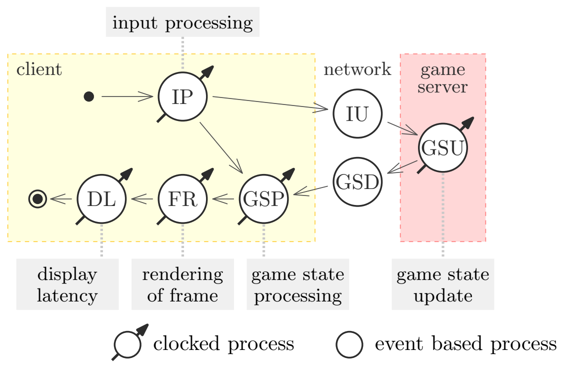
1) Direct gameplay communication: In a networked video game, some parts of the game pipeline run on one or more remote entities. Therefore, to be able to actually play the game, it is required that each gaming client exchanges data with these entities. Depending on the architecture, these entities can be either game servers or other gaming clients. In the following, the main types of gameplay communication will be described. Please note that the involved gameplay communication highly depends on the game type as well as the selected video game network architecture. Naturally, this also influences the critical QoS requirements for different game types using different architectures. Details on architectures are given in Section V, while Section VIII-B discusses the impact of game types.

Game input upload: The user interacts with the game by issuing commands using an input device in order to influence the game state. In a networked game, either the commands themselves (e.g., [25]), or the resulting game state differences (e.g., [26]) have to be uploaded to the game server or broadcast to the game peers. Basic input commands can also be aggregated (e.g., [27]) or locally abstracted into higher level commands before the upload, such as moves, events, or changes to objects [28].

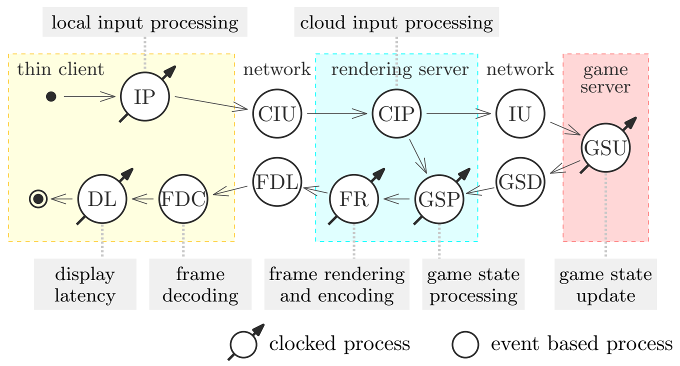
Game state download: Each pass of the game loop results in the creation of a new game state. The game state incorporates the decisions and inputs of each player subject to the game logic. The server, as the authority over the global game state, must synchronize this state back to all clients in a serialized format. The state synchronizations can be performed as standalone downloads (so-called *snapshots*) to each client, or the server can first calculate the differences to the previous state to reduce download sizes. The server can even present each client with a different view on the global game state in order to reflect, e.g., limited viewpoints of each client [29]. Upon reception by the client, the server's game state will be merged with the local game state, such that global state changes are reflected when rendering the game output (e.g., [27]). If there are multiple game servers—or only equal peers with no central server—responsible for a game session, the different game servers also have to keep a consistent state among them, and thus, need to exchange state information for synchronization [10].

Figure 5a extends the previously presented local models of the game loop with a game server component alongside the necessary adjustments to the client that were previously described. First of all, after the user's input has been processed locally, the input has to be packed into an input upload message. This input message is transmitted to the game server, which then incorporates that input (and those inputs of any other clients) into the next periodic global game state update. This new game state will be transmitted to the client, which merges the new game state with its locally computed version and renders a new frame. Note that multi-threaded uncoupled models (from Figure 4b) can be extended accordingly by inserting the same network components.

A special case is *cloud gaming*, also referred to as *game streaming*. In general, game streaming can be conducted between any two computers regardless of their location. This then encompasses both streaming a game from a cloud server



(a) Networked game loop with game input upload, game state download, and game state processing.



(b) Cloud gaming loop of a networked game; with game input upload, remote rendering of frames, and game video streaming.

Figure 5: Gaming loops with and without game streaming. Colors indicate which device the processes belong to. Circles with arrows indicate clocked processes, they run with a certain tick rate (e.g. 60 Hz) independently of available input.

to a local device and streaming between two local devices. The streaming client itself only is mostly just a video player and input handler and forwarder. All game inputs are forwarded to a remote rendering server which performs all resource-intense tasks. The server processes the game state, and renders and encodes the game video. The rendered video is then streamed back to the game client, who receives the video data, decodes and displays the game video to the user [4]. Figure 5b depicts the respective game loop, with the additional cloud rendering server highlighted in blue.

In multiplayer and online games, the rendering server—that performs all of the graphics-intense client task—and the game server—responsible for computing and synchronizing the game state between all clients—will be separate entities. After the game server has received the input uploads and computed the state updates, the game state is sent back to the rendering server, where it is rendered and encoded for the client. In other, typically singleplayer, cases, there is no separate game server. In this case the rendering server only acts as a remote game client and directly processes the game state itself. The discussion of game streaming will be continued in Section V-D.

2) Indirect gameplay communication: Indirect gameplay communication refers to any communication which is not an integral, continuous part of the actual gameplay, but which still is a prerequisite for playing the game or can interact with gameplay communication.

Game delivery and distribution, which includes downloading a digital copy of the game, are often essential prerequisites before a game can actually be played. Many contemporary games additionally also feature the purchase and subsequent download of additional Downloadable Content (DLC) or virtual goods. Prerequisite communication in a game also often includes authentication to a game platform, from which the clients are redirected to the game server, as well as account and session management. The required networking for this kind of indirect gameplay communication typically relies on standard Web technology for the communication with the gaming platforms or servers, including the download of large files to the user's device for game delivery and distribution, the authentication of users, and the encryption of exchanged data, e.g., with Transport Layer Security (TLS). As these are already well researched topics, which are not exclusive to online video games, they will not be discussed in any further detail in this work.

The participation of clients in a multiplayer match is often managed by a separate matchmaking server. This matchmaking server matches clients according to their preferences and skill [30]–[33], allocates or starts a game server for their new match, and finally redirects the clients to their assigned game server, which will be responsible for all direct gameplay communications. An important goal of matchmaking is to match those clients together which have similar Internet connection conditions (e.g. expressed as bandwidth and latency) to the game server [34]. To this end, players can often pre-select a geographic region, and the actual game server within the region is selected by the matchmaking.

It might also be necessary to download level or world data before the game starts, or even dynamically during gameplay (e.g. seen in the 2020 Microsoft Flight Simulator). This might cause waiting times until all clients have downloaded the required data. Depending on the game, that data can even be part of the game state download performed by game clients (and thus be a part of direct gameplay communication). However, some architectures require additional connections to dedicated world servers for obtaining level or world data, which interleaves with the gameplay communications. During a running game, game clients can upload analytics data to the game server or dedicated analytics servers. This can include statistics about the connection, such as link capacity and latency, which can be used to improve the networking or mechanics of the game [35], but also gameplay statistics, which can be displayed or queried outside of the actual gameplay (e.g., highscore lists). Such statistics can also unlock achievements, which in turn can trigger the download of data, such as animations, unlocked game content, or virtual goods, which can be used during gameplay. Some games also employ Digital Rights Management (DRM) to prevent copyright infringement, which can additionally require a persistent connection to an authentication server (always-on DRM).

Finally, gameplay might also trigger the communication of players via side channels, often with additional software for messaging or voice chats, causing additional traffic [36]–[38]. Some players also stream their gameplay to other spectators via live streaming platforms like YouTube or Twitch [39]–[41].

In summary, as a prerequisite or in addition to actual gameplay communication, several secondary applications may be used by players that result in traffic not directly related to gameplay. This can include text and voice communication applications with which player communicate when playing a game together. Such indirect gameplay communication also has to be considered when analyzing networked online games, since it might also immediately influence players' behavior and assessment of a situation.

B. Network Performance Influence Factors

With the distinction between indirect and direct gameplay communication at hand we can now more closely scrutinize the impact of network parameters and influence factors on video games. Indirect gameplay communication, such as game and content downloads, text and voice communication, or account and session management are closely related to classical Internet applications. Hence, insights from research regarding file downloads, VoIP, and Web browsing can largely be applied [42]. For example, file downloads are known to require high throughput but are relatively resilient against delay, jitter and loss. VoIP and Web browsing, on the other hand, are generally more sensitive to delay, while only requiring a limited amount of bandwidth with the former suffering especially under the influence of packet loss. Similarly, when it comes to direct gameplay communication between a game server and a client, studies have shown that, depending on the specific game, latency, jitter, as well as loss generally impact the gameplay quality. Bandwidth nowadays only plays a minor role during gameplay, while the continuous gameplay communication must be timely in its arrival, gameplay update sizes are only marginal when compared to typical residential Internet access speeds [43], [44], but may pose to be more problematic in mobile networks.

1) Online Game Network Performance: In the following, a more detailed look will be taken at what data is actually being sent through the network. In general, three scenarios prevail in modern games, which were already identified by early Internet games [45]. All scenarios involve an *input upload* (IU) step in Figure 5, as well as a *game state download* (GSU) step. However, the mechanisms differ with regards to which data is transmitted. First, the game client can simply compute the game state locally and transmit the full game state to other involved entities. This, however, requires additional computation on the client side as well as increases the needed upstream bandwidth. In return, the game server only has to notify clients about small deviations in their local game state to unify multiple clients. In the second scenario, clients transmit only the player input, such that the interactions with the game state are executed on the game server (Client-Server) or on another game client (Peer-to-Peer (P2P)). This reduces the complexity of the client but introduces additional latency, as the client has to wait for the updated game state from the server before being able to proceed. Additionally, the server has to return the full game state to the client in order to ensure consistency. In a third approach, only the portions of the game state that is relevant to a client are sent back from the game server. This comes with the additional complexity of deciding which information is relevant and which parts of the game state can be omitted. This task is commonly called *interest management*. When it comes to modern games, a combination of these mechanisms is often used to form a trade-off between efficiency and complexity [46]. All three scenarios have in common that the amount of data transmitted by a single client is, by today's standards, fairly low [45], [47]. Thus, this process is generally more sensitive to latency and jitter while being very resilient against throughput fluctuations.

Reference [48] investigated the bandwidth requirements at the game server and connected clients. They also evaluated the latency to resolve game state inconsistencies for three different architectures, namely, client-server, P2P, and P2P with a central arbiter for detecting state inconsistencies. They provided models for the bandwidth requirements and the maximum inconsistency period for all three architectures. With this, they identified both the large bandwidth requirement at the game server in a client-server architecture, as well as the large overhead in the P2P architecture to synchronize the game state. Their proposed hybrid architecture with a central arbiter combined the merits of both architectures.

Surveys of past online video game traffic studies are available at, for example, [49]–[51]. In addition, the sources and relevance of network latency, jitter, and loss have been investigated in [10]. The authors of [11] and [52] commented on acceptable latency in networked multiplayer games, stating that it ranges between $0.1\,\mathrm{s}$ to $1\,\mathrm{s}$ depending on the game genre. There, Real-time Strategy games were considered to be less sensitive than First-Person Shooters. These numbers were confirmed by past empirical studies [53]–[55]. But these results are only applicable for a very narrow type of game in a specific scenario and for a specific application layer metric. In general, genres might not be as clear-cut as previously assumed and do not necessarily govern the game's speed. Due to this, it is almost impossible to define generalizable QoS requirements without considering game type, video game network architecture, and required gameplay communication. This issue becomes even more apparent when attempting a full subjective assessment of the participants' opinions. However, these missing aspects would be important puzzle pieces on the road to a more complete and generalizable QoE model for video game. This notion will be picked up again in Sections VII and VIII. More details on the impact of network conditions on the game performance will be presented in Section VI.

2) Cloud Gaming Network Performance: When it comes to cloud gaming, the scenario changes substantially as, in addition to regular direct as well as indirect gameplay communication, player input needs to be streamed to the rendering server, which then has to return a high resolution, high bitrate video with as little delay as possible [56]. The upstream and downstream parts of this process are shown in Figure 5b as CIU and FDL, respectively. Hence, cloud gaming requires significantly more and different resources on the server side compared to regular online gaming. However, as both directions of this additional communication are closely related to well established applications, insights from other areas can be applied here. On the one hand, the upload of player input is similar to the regular input upload already discussed before as well as other latency-sensitive, low bandwidth applications such as Web browsing or remote control in industrial environments [57]–[59]. On the other hand, the video download behaves similar to live streaming or video conferencing, albeit with much stricter real-time requirements. The impact of network characteristics on adaptive streaming has already been studied in-depth in the past [60].

Finally, when it comes to the delay requirements, cloud gaming providers may employ proprietary mechanisms to reduce actual as well as perceived latency [61]–[63]. But such approaches are still subject to research and evaluation of their effectiveness. Naturally, network management solutions like DiffServ [64] or Active Queue Management (AQM) [65]—mechanisms to mark specific traffic types and reduce the queuing delay in the network—can also be deployed to improve QoS for gaming related network interactions within edge and core networks. While the details of these network management mechanisms are out of scope some future research directions on this regard are presented in Section VIII-C.

The following section provides a more detailed overview of client and server-based mechanisms that aim to improve the perceived latency of players.

C. Networking Mechanisms within Games

When it comes to understanding the networking behavior of video games, some aspects beyond the implementation and usage of network connectivity need to be taken into account. Games employ several application layer mechanics that either directly influence the resulting network traffic or change application layer behavior based on the current network state, forming a feedback loop in which the game behavior reacts to current network conditions, thereby influencing the resulting traffic characteristics.

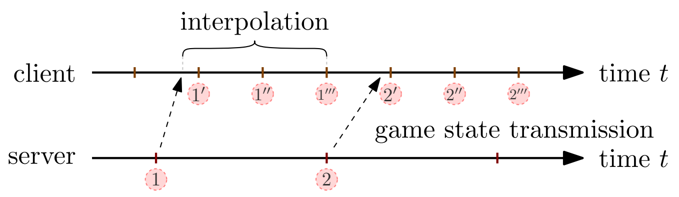
Similar to other network applications in which data protection, privacy as well as authentication concerns are highly relevant, the encryption of network traffic is widely used for direct and indirect gameplay communication. The encryption of network traffic offers some additional protections against cheating and manipulation of game state information. However, most games do not solely rely on data encryption to ensure tamper-proof gameplay, since cheating does not necessarily happen during transmission, but at the game client's system itself. Instead, the aspect of game state authority is leveraged to exclude the client from decision-making processes that could be used to manipulate the game state. In environments with authoritative game servers, the client still performs the local game state simulation. However, the resulting game state will be overwritten by the game state authority (i.e. the server) if the states don't match. If the local game client is not authoritative on the global game state, many opportunities for local manipulation are removed. This also means that any local input has to be validated by the authority and is only then included in the global game state [66]. If there is no central authority (e.g. in a P2P approach) consensus between the peers can be difficult to reach and slow to converge to a consistent state across all peers [67]. This is often considered too slow for fast-paced gameplay, nor resistant enough against cheating, and may impose a significant traffic overhead.

The type of game itself is also highly correlated with the occurring network traffic. The traffic patterns of slow, round-based games without real-time interactions will differ significantly from fast-paced, real-time shooters. The *tick rate*, the rate at which a game client and server exchange state information, generally describes the number of updates per second and is provided in Hz. With slow, round based games having tick rates as low as 1 Hz and fast paced shooters featuring tick rates of up to 120 Hz, it is clear that different games are more or less sensitive to jitter as well as packet loss.

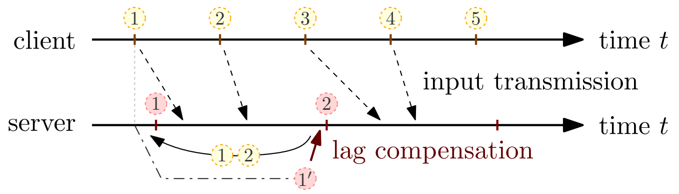
This sensitivity has lead to the introduction of several mitigation techniques applied to diminish, or at least conceal, the effect of network transmissions on gameplay features. The following paragraphs provide an overview over the most prominent mechanisms.

On of the earliest game state synchronization mechanisms is called *deterministic lockstep* [47]. The basic idea is that every game client executes the exact same code at the exact same tick rate, with every input being directly shared between clients. However, this can lead to the problem that the game state update cannot be executed until a game client or server has received all input uploads from all the players, which causes the game to stall [68]. While this is not a large problem for turn-based games, games with real-time interactions will suffer. In addition, to make the game look and feel smooth, the consequences of user interactions can be locally predicted and presented at a higher frequency (client-side prediction) before other players or the game server have acknowledged the input and updated the game state [69], [70]. The drawbacks of the deterministic lockstep mechanism are that it does not scale well for many players [26]. An improvement called *rewind and replay* or *deterministic rollback* was proposed, in which a game client could also predict the remote players' actions for a more responsive gameplay. When the remote players' input uploads are received, they are compared to the predictions. If there is a discrepancy, the game simulation is rewound to the first incorrect tick. Then, the game client re-predicts the inputs for each player based on the updated input stream, and advances the simulation to the current tick using the new prediction [27], [71]–[74].

As described above, the problem of synchronizing the game simulation between all game clients increases with the number of participating game clients. This issue can be solved by introducing a single, central authority—the game server, or host—which solely runs the game simulation, and thus, keeps a single, unified game state. All game clients send their input upload messages to the host, which executes all inputs and updates the game state. Afterwards, the server sends back state download messages to each game client. Since it might not be



(a) Client-side snapshot interpolation. The client interpolates its local game state based on the game states received from the server, until a new server state is received. This enables a higher rendering rate than what would be possible if the client directly used the server state.



(b) Server-side lag compensation. Based on the client's current delay to the server the server assigns the client's input messages to older game states and recomputes all subsequent states.

Figure 6: Timelines of snapshot interpolation and lag compensation mechanics.

feasible to transmit the complete game state, only snapshots of the game state are sent in intervals, which contains all or parts of the state of the game simulation. To further reduce the snapshot size, geometry transmission [75], snapshot compression [76] or other *delta compression* techniques can be employed. Here only the differences to the previous game state are transmitted [77]–[79]. Snapshots can be compiled in such a way that they only contain the parts of the game state of interest to a particular client. This *interest management* [80], [81] allows the game server to send a subset of the game state to each client, but requires a more complex management of the state updates. This, combined with the fact that the authority now solely resides with the server, removes a lot of potential to cheat from the client [66], [69].

The game clients take those snapshots to reconstruct a visual representation of the game simulation without running the game state simulation themselves. But since the intervals of the snapshot reception might be larger than the time between two frames, a further technique, called *snapshot interpolation*, can be employed. Hereby, the client shifts back its rendering time and interpolates the game state between the two last received snapshots, which adds constant but additional latency. Once the rendering time progresses towards the last snapshot and a new snapshot does not arrive in time, the game client has to switch to extrapolation, which can cause a gameplay degradation in case erroneous extrapolations have to be corrected [26], [69], [82]. A depiction of the interpolation mechanism is available in Figure 6a.

Typically a technique called *lag compensation* is employed in conjunction, which takes into account the amount of interpolation at the game client (cf. Figure 6b). This means, with lag compensation, the game server does not apply input actions to the current game state update. Instead, it takes into account the simulation time at which the user performed the actions, then rewinds the game simulation to the observed game state at the client, applies the input actions, and from there moves forward to the current game state. Note that lag compensation allows for each player to run on their own simulation clock with no apparent latency. However, paradox situations might occur, especially when clients experience significantly different latencies to the game server, e.g., the “shot through a wall”-inconsistency [69]. Finally, *state synchronization* is a hybrid approach borrowing from both deterministic lockstep and snapshot interpolation. The idea is to run the game simulation not only on the game server but also on the game clients. Here, each client again sends input messages to the authoritative game server, and the game server sends state download messages to keep the game simulations synchronized. As the simulation runs also on the clients, they do not have to wait for input from the game server, but the game can progress locally, i.e., by extrapolating from the last received game state. However, this can result in only approximate synchronization, which might require some effort to correct extrapolation divergence towards the global game state of the game server [29].

Most importantly, there is need for mechanisms that enable latency compensation or concealing [10]. A client will always observe a delay > 0 between sending an input upload message over a network to the game server or other game clients, and receiving one or more corresponding game state download messages. A simple solution, which was already presented above, is *client-side prediction*, such that the consequences of user interactions are locally predicted and rendered before game state download messages arrive. Similarly, the input actions of other players can be predicted and rendered before the corresponding game state download messages arrive. This mechanism is often called *dead reckoning* [83], when positions and paths of moving objects are predicted. Both mechanisms extrapolate from the game state, and thus, can be prone to prediction errors. In this case, the extrapolation divergence towards the global game state has to be corrected, which, if perceived by the end user, might negatively affect their experience with the game.

Finally, to further reduce the latency to and from the game server, the game server itself can be moved closer to clients. This is a use case of the emerging edge computing paradigm, which allows the use of computational resources at the edge of the network, close to the end user. However, the performance of such mechanisms highly depends on the availability of edge data centers and the geographic distribution of players. To enable good matchmaking for a game, game servers are often placed in the middle of larger population areas, as the matchmaking process requires a large pool of eligible players. To alleviate the problem, [84] presented a migration algorithm, which allowed a player to choose a better server and migrate the game state in the middle of a game. Considering world-spanning multiplayer online games [85] applied core selection to find an optimal node in the system for placing a virtual zone, and correspondingly the players interacting in that region. The authors of [86] considered first-person shooter games and proposed a platform that determines where and when to move game servers. A simulation-based performance evaluation of edge server resource migration policies was conducted in [87]. Today, due to the disruptively long pauses during gameplay, migrations are usually never performed during a running game session in fast-paced games in practice.

D. Online Video Game Protocol Stack

Besides the way games use the network for various communication purposes, they also rely on specific network stacks and protocols to optimize efficiency, resilience, as well as security.

A networked video game is, essentially, just like any other type of networked application, it sends and receives data over an IP network and wraps its data in an application layer protocol. Note that properties specific to layers 1 and 2 protocols are out of scope for this manuscript, but it should be emphasized that specific circumstances—like random access in a shared medium, such as WiFi, or issues like bufferbloat—will impact the delay-intolerant game communication in a negative way. It should also be noted that many games outsource their networking implementation to third party libraries provided by either the used game engine or explicitly use networking libraries like Valve's GameNetworkingSockets[1]. This means, even though there is no standardized game protocol stack, many games still share similar codebases and behavior.

When it comes to transport protocols, TCP and UDP also prevail for networked video games [88], including the usual advantages and disadvantages. TCP has the advantage of a persistent connection with congestion-controlled, reliable in-order delivery. However, the retransmissions required for reliable delivery can introduce Head-of-line blocking—i.e., newer segments are blocked by missing older segments that are still unacknowledged—and with that additional delay and jitter, two aspects detrimental for interactive real-time applications. Thus, TCP is mostly used to handle data with less strict deadlines and in cases in which reliability is more important, i.e., indirect gameplay communication. For example, it is not suited to transmit an immediate game state but possibly some future world data or player communication. In contrast, UDP allows for the fastest possible transmission in a lossy environment without intrinsic reliability or ordering guarantees. If such properties are still desired they can be tailored to the game on top of UDP.

Looking at research, [88], [89] supported TCP for situations in which occasional delay is acceptable, such as stateless queries, updates, or slow-paced online games. The authors of [45], [90] argue against the use of TCP for video games. These papers agreed that for fast-paced games, in which occasional lag is not tolerable, UDP should be used and required mechanisms, such as the retransmission of lost packets, have to be implemented by developers on top of UDP (e.g., [45], [91], [92]). Chen *et al.* evaluated the TCP performance in online games [93], and showed that TCP cannot perform well for Massively Multiplayer Online Role Playing Games (MMORPGs). The authors proposed guidelines for protocol design and extended their work in [88], which investigates more protocols, namely TCP, UDP, Datagram Congestion Control Protocol (DCCP) and Stream Control Transmission Protocol (SCTP) as well as other content-based transport strategies. They found that their proposed strategies could reduce End-to-End (E2E) delay and jitter significantly. Moreover, other transport protocols have been proposed in research for networked video games as well. The authors of [94] propose an energy-aware transport protocol for mobile multiplayer games. And a TCP-like transport protocol specifically designed for the transmission of game events has been developed in [95].

Moving up the stack, the protocols encountered in video game network communication start to differ from other applications. Unlike the Web, where HTTP and other standardized protocols are the defining elements, the gaming landscape has no such widespread standardization. Instead, developers often utilize third-party networking libraries and frameworks for general data serialization, state replication, and remote procedure calls such as Google's ProtoBuf[2] which is used in popular titles like `CounterStrike: Global Offensive` or `DOTA 2`[3]. These tools make it relatively easy to develop a custom communication protocol for each game (e.g. [96]), but makes comparative analyses between games more challenging.

There have been only scarce works in literature to analyze the traffic patterns of video games and while the observations made may not be entirely applicable to any game beyond the one investigated, they can still give valuable insights into what behaviors can be expected from video games, even though most studies have to operate on encrypted traffic and are limited to identifying patterns. In addition, even a simple version upgrade or a change in game settings, could entirely change the observed behavior. Past examinations include `Counter Strike` [97] and `Overwatch` [98] as well as several surveys [49]–[51] that cover traffic analyses of additional games. Further sources of information exist in the form of developer documentation [25], [28] and open-sourced approaches [71].

V. Video Game Network Architectures

Video game network architecture designs can generally be divided into two large categories, namely, client-server architectures and P2P approaches, and while other, hybrid approaches exist (e.g. [10], [99], [100]), these are less widely spread and are seldom used outside of research. In the following, we provide an entry point into the research body of network game architectures as well as the concept of cloud gaming. Table I provides a summary of the references in this section. Note that this table is designed to provide an index of the respective literature, and does not contain results, e.g., with respect to QoS or QoE implications of the presented architectures. As the interactions between architecture, game type, and gameplay communication of each game significantly impact the presented results, the outcomes of the discussed works are typically very specific to the applied methodology and considered use case, and thus, cannot easily be generalized.

[1]https://github.com/ValveSoftware/GameNetworkingSockets

[2]https://developers.google.com/protocol-buffers/

[3]https://github.com/SteamDatabase/Protobufs

Table I: Taxonomy of research work regarding video game network architecture.

Reference		Methodology	Research focus
		Client-Server Architecture	
Bang'97	[101]	Software Proposal	Distributed Computing
Cai'02	[102]	Implementation	World partitioning, scalability
Fied'02	[103]	Implementation	Scalability, rapid development, system evolution
Hsu'03	[104]	Implementation	Scalability, interest management, game state synchronization
Ng'03	[105]	Simulation	Scalability
Wang'04	[106]	Implementation	Scalability, load balancing, cost efficiency
Chen'05	[107]	Implementation	World partitioning
Lee'05	[108]	Simulation	Resource allocation, game state synchronization, server selection
Assi'06	[109]	Implementation	Bandwidth, congestion, reliability
Ta'06	[110]	Simulation	Scalability, world partitioning, server selection
Glin'07	[111]	Implementation	Scalability, game state synchronization
Plos'08	[112]	Implementation	Scalability, responsiveness
Khan'10	[113]	Simulation, Implementation	Cheating mitigation, consistency
Prod'16	[114]	Simulation	Self-adaptation, load balancing, availability
		Peer-to-Peer Architecture	
Diot'99	[115]	Implementation	Game state synchronization
Min'99	[116]	Simulation	Load balancing for distributed server architecture
Gaut'04	[117]	Implementation	Latency, event ordering, cheating prevention
Iimu'04	[118]	Implementation	World partitioning
Knut'04	[119]	Implementation	Interest management, game state synchronization, world partitioning
Roon'04	[120]	Implementation	World partitioning
ElRh'05	[121]	Implementation	Scalability
Wagn'05	[122]	Software Proposal	Communication protocols
Yu'05	[123]	Implementation, Simulation	Scalability, interest management
Bhar'06	[124]	Implementation	Game state synchronization, scalability, latency
Hamp'06	[125]	Implementation	Scalability, robustness, load balancing
Hu'06	[126]	Implementation	Scalability, topology management, latency
Fan'07	[127]	Implementation	Scalability, robustness, interest management
Neum'07	[67]	Discussion	Challenges of P2P architectures
Chan'07	[128]	Implementation	P2P message overhead, framework validation
Fan'10	[129]	Discussion	Interest management, event dissemination, persistency, cheating mitigation, incentive mechanisms
Yhay'13	[130]	Survey	Overview of P2P solutions
Abdu'15	[131]	Survey	Scalability, reliability, responsiveness
Buyu'15	[132]	Survey	World partitioning, scalability
Liu'15	[133]	Simulation	Cost optimization, P2P feasibility
		Hybrid Client-Server/Peer-to-Peer Architecture	
Cron'01	[134]	Implementation	Game state synchronization
Baue'02	[135]	Hardware Proposal	Network Infrastructure
Muel'05	[136]	Analytical Model	Scalability, P2P and Client-Server feasibility
Mora'06	[137]	Discussion	Consistency, latency, game state synchronization, cheating
Yang'07	[138]	Discussion	Bandwidth, latency
Ali'09	[99]	Survey	Scalability, reliability
Cart'10	[139]	Implementation	Scalability, responsiveness
Ricc'12	[100]	Discussion	Research challenges
Wang'12	[140]	Simulation	Load balancing, scalability
Carl'15	[141]	Simulation	Gossip based interest management, integration of P2P and Client-Server architecture
		Cloud Gaming Architecture	
Schm'99	[142]	Measurement	Performance of thin clients
Huan'13	[143]	Implementation, Measurement	Comparison of performance to existing systems
Wu'14	[144]	Optimization	Development of control algorithm for routing and server provisioning
Carr'20	[62]	Measurement	Traffic Classification of Google's Stadia cloud gaming service
Choy'14	[145]	Measurement	Latency for cloud gaming using general-purpose cloud infrastructure

A. Client-Server Architectures

All clients in a session of a client-server game send their respective user inputs to a single logical game server to be processed into a new session game state. The game server then performs the game state replication to all clients. Additionally, most client-server games compute a game state out of their locally available information which does not necessarily include all (or correct) information from other players. An advantage of this approach is that there is only one authoritative game state, namely the state managed by the server. This substantially eases game state synchronization, as the server's state simply overrules any deviations clients have computed locally since the server state replaces the local state. The approach eliminates the need for any direct synchronization or consensus protocols between clients.

In [146], different policies were outlined regarding the control and location of the server with two main variants. First, the server is installed together with the client and under full control of one player. Several potential disadvantages apply, including the susceptibility to local manipulation (i.e., cheating), asymmetry of delay among players, and lower availability, potentially leading to the remaining clients having to migrate the game session to another server if the server is not available any more. Second, and common to most larger competitive multiplayer games today, the servers are hosted and maintained by the game publisher or platform operator in a data center. Moreover, while network connectivity is obviously mandatory for online multiplayer games, many singleplayer games have also begun integrating client-server interactions, for example to simplify switching to a multiplayer mode in a running session, prevent cheating, or for business reasons. The client-server game architecture can also act as a centralized solution for other game-related activities, such as account management, monitoring, persistence, or partitioning of the game world [147]. The authors of [137] discuss client-server architectures for Massively Multiplayer Online Games (MMOGs), in which several thousand players interact with the game server.

While up until now only the use of a single server has been mentioned, it should always be understood as a logical (or virtual) server and in practice can be represented by more than one physical entity. For this, either a single game server is mirrored to several physical machines, or functionality is split amongst different entities that synchronize their state accordingly. This includes, for example, front-end servers, matchmaking servers, or database servers. Such virtualized servers can then be scaled up or scaled out, such that running server instances always meet the current load. Splitting responsibility is especially useful in the domain of MMOGs. Many different approaches have been proposed and used in practice. Such concepts include zoning, instancing, and replication [114]. Zoning partitions the game world into geographically disjoint zones, which are then assigned to different servers (e.g., [102]). Instancing creates multiple, independent instances of one zone and splits the players in this zone between the instances. The instances can then be run on different servers. Finally, replication distributes load by maintaining copies of the same game world (or zone) on multiple servers but distributes the clients between the servers. Contrary to instancing, replication should not be noticeable by players.

The following works are presented here to given an impression of the diverse directions research has taken in the past for client-server architectures. An early architecture concept was presented in [101], in which a remote server executes the game logic and clients only render the graphics. The necessary game data for this was to be pulled on-demand from a separate storage server. A completely distributed communication architecture was used by the authors of [115] for their implementation of a networked multiplayer game. Reference [116] investigated the issue of load balancing for dynamic change of workload in a networked game. Their proposed algorithm considered the geographical relationship among game units and the short response time of frequent user interactions. A review and analysis of current multiplayer online game system architectures in 2003 was given in [104]. In addition, the authors proposed a scalable, clustered server architecture. Further, the authors of [105] conducted a performance study on multi-server based systems for large distributed virtual environments and discussed various implementation issues. The concept of gamelets as a middle layer between monitoring and communication to increase scalability was introduced in [106], such that the logical partitions of the virtual world are assigned to a gamelet for processing. Here, a single server can then hold several gamelets at the same time.

A distributed algorithm was developed by [108] to select game servers for a group of clients participating in a large scale interactive online game session. Reference [107] showed a locality-aware dynamic load management algorithm for massively multiplayer games. Considering the same type of games, in [136], a scalability model was presented to analytically compare the suitability of different network topologies for certain classes of massively multiplayer game designs. A two-phase approach to efficiently assign the participating clients to servers was proposed in [110]. The goal was to enhance user experience in interacting within virtual environments, such as networked video games. Tackling the problem of distributing the server part of networked games over multiple servers, the authors of [124] presented a system to replicate game objects over different server nodes. Also, in [109] an architecture that supported zoning was developed. The authors of [137] discussed how to distribute server resources over several machines in massive multiplayer online games. A middleware system was developed in [111], which distributed the game state among participating servers, and supported parallel state update computations, as well as efficient communication and synchronization between game servers and clients. The proposed system could be applied for the high-level development of multi-server online games [112], [148]. Finally, the authors of [114] investigated an ecosystem for autonomous, self-adaptive hosting and operation of MMOGs on unreliable resources on commercial cloud services with limited availability.

B. Peer-to-Peer Architectures

In P2P gaming architectures there is no central authority and all game clients are equal. This should not be confused with a client-server scenario where one of the clients exclusively hosts the game, i.e. acts as the server. In full P2P scenarios there is no server. Typically, a fully-connected network layout is used and all clients maintain and update their own local game state using their local inputs in addition to those sent by other peers. The challenge is to ensure consistency in a real-time manner between all peers in the absence of authority and in the presence of network delays [35], [67], [99]. Specific notions of P2P mechanisms for gaming are outlined in [100], which include super peers, P2P overlays of distributed servers, or particular neighbor-based connectivity types. It can be advantageous that this architecture does not need a server, potentially removing one round-trip from the networking delay. Moreover, this architecture is robust without a single point of failure. A game session can continue if one client disconnects from it [67]. However, in a typical, fully connected scenario with n peers, $\frac{n(n+1)}{2}$ active connections have to be maintained, which increases the potential of failure and latency [35]. Some works criticized this lack of scalability [135], while others argued that P2P architectures were scalable, because each new peer adds additional, shared resources to the session [67].

The academic literature has many examples of and surveys about P2P game architectures, amongst others [122], [124], [132]. The following works again serve to give an overview of the breadth of works concerning P2P video game architectures. A detailed description how the P2P model was used successfully in the early `Age of Empires` games is given in [35]. The game simulation was synchronized with the help of an adaptive game speed, which dynamically changed the length of the turn to keep the animation and gameplay smooth over changing conditions in communication latency and processing speed. The authors of [117] presented a protocol for cheat-proof event ordering in P2P games, which reaches a playout latency independent of network conditions and can adapt to network congestion to optimize performance.

With regards to concrete architectures, the authors of [127] propose a framework for a P2P network with super peers. A P2P architecture was introduced in [128]. It provides a client-server-style programming model, hiding the complexity of setting up the P2P network among the clients. Another P2P approach was described in [149]. It exploits specific game knowledge to improve the mechanisms for requesting and providing information to clients. P2P architectures can also be adopted for massively multiplayer online games, which has gained a lot of attention from research in the past [119]–[121], [123], [125], [126]. Design issues and alternative approaches were discussed in [129]. The works covered the dissemination and persistence of the game state in large virtual worlds with many server-controlled Non-Player Characters (NPCs) with the additional goal of cheating mitigation. Considering the demanding game type of MMOG, the surveys in [130], [131] cover further P2P architectures for MMOG. More recently, the authors of [133] conducted a simulative performance evaluation of P2P protocols, and an architecture for fast-paced MMOG was presented in [150] presented. Finally, [151] presented formulae to compare client-server and P2P architectures with respect to the required bandwidth.

While pure P2P game architectures have seen much academic research, they do not play a significant role in practice today. Besides practical hurdles like NAT traversal, major concerns relate to achieving consensus and state replication in a constrained time and with regards to cheating prevention. This can practically be solved with significantly less effort in a client-server architecture.

C. Hybrid Client-Server/Peer-to-Peer Architectures

Many practical implementations employ hybrids of the two previous concepts. In the following, some notable examples are outlined, while more examples can be found in a survey by Ricci and Carlini [100].

An important hybrid approach are relay servers [152], which forward P2P packets between peers. Players can more easily connect to them in environments where direct connections might be prohibited, e.g. through NAT middleboxes. Relay servers can also reduce traffic in P2P games. Instead of distributing all messages to all other players, messages only have to be sent to the relay server, which then takes care of the distribution. Both [134] and [153] describe a hybrid architecture with mirrored game servers. Another approach was presented in [118]. It adapts multiplayer online games to P2P architectures by constructing game server clusters on a distributed hash table. Further, the authors of [138] presented a P2P architecture that additionally used multiple mirrored servers to maintain the game state. Other hybrid P2P cloud architectures for MMOGs were proposed in [139], [140]. Another approach, which integrates centralized and P2P architectures, was discussed in [141], specifically in order to support interest management. Reference [135] suggested to deploy middleboxes in the ISP to intercept game traffic and offload some game logic calculations to them. The authors of [113] presented an adaptable client-server architecture for mobile games, which dynamically decides where which parts of the game logic are executed, either client-side or server-side, reaching improved global consistency under high and varying latency network conditions. Finally, the required bandwidth and latency to resolve game state inconsistencies in different architectures was investigated and modeled in [48].

D. Cloud Gaming

In the past decade a new type of service called *game streaming* or *cloud gaming* has made technological advances and has been become increasingly popular. Instead of running a game locally, it is now rendered on a remote game client and the finished output is streamed to the player's end device. At the same time, input provided by the player is streamed to the remote game client for processing. This overall process is generally not considered a video game architecture like client-server or P2P, but rather as a different approach to running and playing games. Any game that is available on a local PC could therefore also be played via one of the open, self-hosted cloud gaming systems (e.g. ParSec or NVIDIA Gamestream). However, there are also closed cloud gaming platforms, such as Stadia or PlayStation Now, which offer only a small selection of games that also may not be available on any other platform. This mainly depends on the business model the operator has selected, c.f. e.g. [154].

Figure 7 illustrates the differences between online gaming and cloud gaming. In cloud gaming, the local device only acts as an input forwarder to a remote rendering server, which implements a remote version of the game client. This server forwards the inputs to and receives state updates from the game server. It then renders the game video, redirects its rendered frames into a video encoder, and transmits the stream towards the local client, where it is decoded and displayed. This means that the local device is not required to have any significant graphics processing power or computational resources to partake. However, it naturally comes at the cost of additional delay, as shown in Figure 5b, and potential visual quality degradation due to encoding, streaming, and decoding as discussed below.

While the first commercial service providing cloud gaming was launched as early as 2005, the technology only became

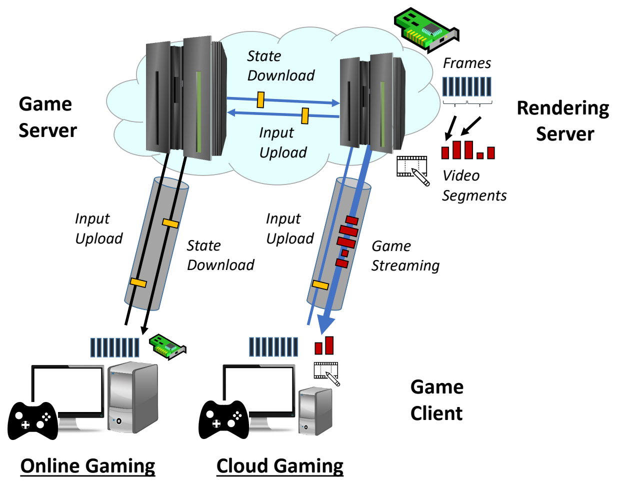


Figure 7: Differences between online gaming and cloud gaming.

popular with the first wave of commercial solutions like OnLive (2010) [155] and Gaikai (2014) [156]. However, these initial services were largely commercial failures due to technical and qualitative limitations. They also garnered little interest from consumers due to pricing and a limited selection of games at the time. Recently, a second wave of commercial services began to emerge, with PlayStation Now, Geforce Now, Amazon Luna, Microsoft Xbox Cloud Gaming, and Stadia being the most prominent installments of cloud gaming as of writing this manuscript.

Apart from commercial services, many existing game platforms also implement game streaming from a locally installed copy on one device to another. This can not only be used to stream the game from a more powerful computer to a TV or mobile device, but streaming can often also be performed over the Internet. Examples are Steam Remote Play, Xbox Console Streaming, NVIDIA Gamestream, and PlayStation Remote Play. The following paragraphs briefly survey the ongoing research work into these services and systems.

Already several years before the commercial rise of game streaming, [142] investigated the interactive performance when streaming a video game on an early thin client setup in a local area network. Also the authors of [157]–[159] presented early generic client-server systems for streaming 3D graphics and virtual environments to mobile devices. Other specialized systems for interactive video games were proposed in [143], [160], [161]. Regarding commercial cloud gaming platforms, two works [4], [162] described their architectures and conducted first performance evaluations of the first wave. Since then, a plethora of architectures and systems were investigated and open research challenges were identified, e.g., [163]–[166]. A comprehensive survey was performed in [8]. The same group of authors additionally provided some very insightful future perspectives in another article of the same year [167], not all of which have proven to become reality to this date. Finally, several recent works (e.g., [62], [168]) performed traffic analysis of current cloud gaming services. These analyses revealed that individual games may differ in their streaming traffic characteristics, such as the packet size, inter-packet times, and load.

Further, the authors of [169] investigated the networking behavior of Stadia, PSNow and GeForce Now and identified the used protocols as well as performance requirements regarding bandwidth and loss. They identified PSNow and GeForce Now to be based on the RTP protocol while State relies on the WebRTC API. Furthermore, in their tests all platforms could cope with a packet loss of up to 5% before a degradation of streaming quality occured.

Apart from those works, there is for the moment little academic research that focuses on these latest services due to the recency of the current, second wave of cloud gaming services.

Two core aspects govern most technological decisions of game streaming, namely the aspects of stream quality and latency. The latter concern begins with the placement of cloud gaming servers in order to keep the propagation delay below a tolerable threshold. This represents a strong contrast to most other cloud computing services, where E2E lag plays a diminutive role and data center placement might be more directed by factors like energy efficiency and multiplexing gains. For example, [170] suggested a maximum distance of $1600\,\mathrm{km}$ from a data center for streaming fast paced games. This would result in a round-trip propagation delay alone of slightly below $100\,\mathrm{ms}$ in addition to other lag factors, which is considered too high nowadays, as most current services operate at RTT ranges of about $15\,\mathrm{ms}$ to $30\,\mathrm{ms}$. The authors of [171] give some insights on this placement problem and how it relates to other lag factors. The authors of [144] formulated a constrained stochastic optimization problem to derive an online control algorithm for request dispatching and server provisioning. The same problem was tackled in [172] with a linear program focusing more on the resulting QoE, and [145] leveraged the edge computing paradigm to place additional servers closer to the game clients.

When it comes to video streaming technology, cloud gaming is most similar to traditional RTP-based video streaming on top of unreliable but non-blocking UDP transport. Indeed many current services simply use WebRTC (e.g. Stadia [62]), of which RTP is a main component. Additional video bitrate and quality adaptation mechanisms that adjust to current network conditions can be employed by the rendering server as well. These typically adapt the resolution, frame rate, or the compression of the encoded and streamed video, and can also be dynamically coordinated with the game's current output settings to achieve the best effects. For example, [173] presented a system, which can dynamically select the video frames and adjust forward error correction coding to achieve optimal video quality in mobile cloud gaming. A study on the video quality of commercial cloud gaming services of the first wave was conducted in [162].

It is important to note that one of the main differences of game streaming to on-demand video streaming services is that content cannot be pre-encoded and distributed, e.g., by a content delivery network. Instead, it is more similar to video conferencing, in that the video content has to be rendered and encoded right before the delivery. However, while some broadcast delays are acceptable when passively watching a live event, the interactivity of networked video games drastically confines the amount of acceptable delay to a minimum. Consequently, video rendering and encoding latencies have to be controlled and minimized in game streaming, which especially concerns the chain of video encoder, stream transmission, and decoder. Furthermore, the game's frame rate should be as high as possible, typically $60\,\mathrm{Hz}$ or beyond, with the video encoder operating at the same frequency with minimal coding delay [174]. Due to this low time budget, encoders will operate at lower coding efficiency, thus producing either a lower quality output or higher bitrate videos. Current services recommend an available bandwidth of about $50\,\mathrm{Mbit/s}$ for 1080p60 (Geforce Now) or 2160p60 (Stadia) video. Finally, buffering of the streamed video must be avoided, apart from a shallow jitter buffer. Thus, any disturbance that exceeds this buffer will result in a stall on the last received frame, and a subsequent skip to newly received frame to catch up the time lost. This is in concept similar to the playback interruptions in HTTP Adaptive Streaming (HAS), where it as been deemed the worst quality degradation [60]. This means, that adaptation technology has to ensure that stalling is avoided by conservatively selecting the enconding bitrate of the game video stream, which especially includes that motion and scene complexity have to be considered [175]. In addition, the non-blocking nature of UDP transmissions also helps in avoiding stalls.

VI. Performance Evaluation of Games

The previous sections have introduced numerous system and networking components of games. But it might not yet be immediately obvious how all these can affect the perceived quality. To understand this relation, we now look at how these components are used in games and in specific scenarios. The key metric of this section is the total delay—or End-to-End (E2E) delay—as it is known to have a large impact on the interaction quality of games. This term subsumes the induced delay of all system components, not just the network delay. The effects of delay differ not only with its magnitude, but also by its source. For example, network delay may cause severe problems in some aspects of a game while being entirely masked in another scenario. Therefore, the benefits and drawbacks of different measurement methodologies, metrics, and measurement points must be understood and applied correctly. This includes a basic understanding of a game's netcode, as different games may behave differently and expose different lag inducing components via the same measurement methodology. As an example, measuring just the network delay does not catch delays caused by the game engine, graphics card, operating system, input and output devices, or any intermediate buffers. In this section, common delay inducing components are discussed along with possible measurement points and scenarios which typically include these components.

A. Delay-Inducing Components

One critical aspect of delay-inducing components is that they are not all equally important, ordered in the same way, or even present at all in every observed game and every system that can be played on. Figure 8 shows the path of a user's input to the display of the results in different system models. This path represents all components that together make up the full *E2E lag*. Which model applies depends on the specific game and requires knowledge about the game's internals, e.g., whether it applies a coupled or uncoupled model (cf. Section III-B). These relations are also presented in different scenarios. With and without a game streaming solution, and a fully local game versus an external game state update server.

In addition, Figure 8 may also be used to derive measurement points for various metrics that define a game's performance. Thereby, Figures 8a–8d illustrate the lag between a button press and its corresponding information appearing on the screen, while Figure 8e is used to derive measurement points for the frame time[4] in the uncoupled game loop model. Note that most performance measures only capture part of the entire delay, e.g., purely software-based measurements cannot include delays caused by the display or input devices.

The following measurement points are used to classify common measurement methodologies in literature. The individually considered delay components are presented in the following paragraphs. While this description is self-contained, similar models and illustrations have been presented in literature and may be consulted for further details [12, Chapter 3, Figures 3.1–3.4].

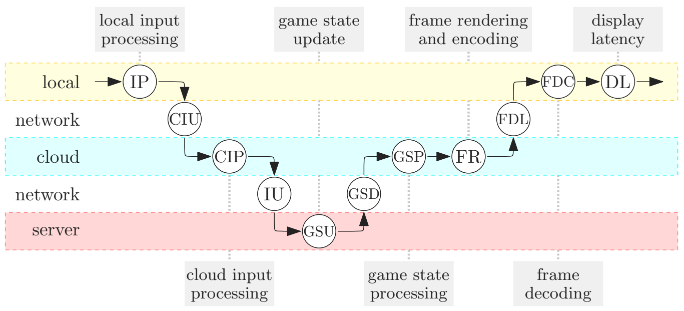
a) Input processing (IP), (CIP): This includes the input device [176], the interface of the device [177], potential delays caused by wireless communication, software-induced delays such as by the operating system [178], and the game engine's processing time. As an example, [179] collects and explains the delays of various keyboard models. Delays of other input devices are often of a similar magnitude.

b) Network delay (CIU), (IU), (GSD), (FDL): The network delay includes the packet delay of the entire network path, but also related components like processing delays, e.g. by the operating system's network stack, and engine-induced delays, e.g. data serialization and packetization. It is present in both game streaming (cloud input upload CIU, frame download FDL) and in networked video games (input upload UL, game state download GSD).

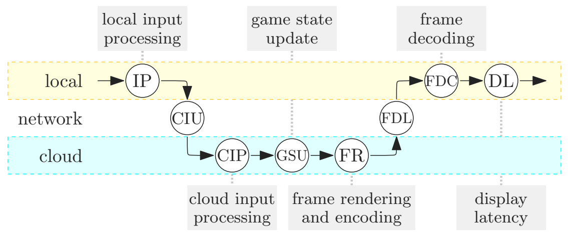
c) Game state update (GSU): This delay consists of the CPU time required for the computation of the new game state. The associated computation can be conducted locally (Figure 8d), on a streaming server (Figure 8b), or by the authoritative game server for networked games (Figure 8a and 8c). Note that user input arrives asynchronously at the game engine and does not trigger immediate computation, thus any new input needs to wait until the *ongoing* game state update cycle is finished before being considered for the next one. This represents one of the aforementioned clocked processes.

d) Game state processing (GSP): This type of delay is only present in networked games with external game state updates (Figure 8a and 8c) and consists of updating the

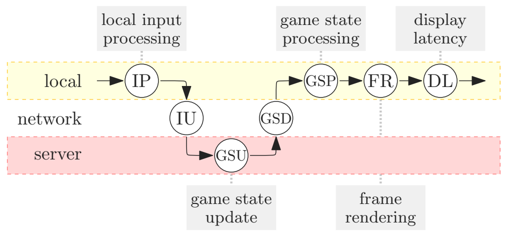
[4]A frame time is the time it takes to create and render a single frame, e.g. a frame time of about $16.6\,\mathrm{ms}$ would result in a frame rate of $60\,\mathrm{Hz}$.



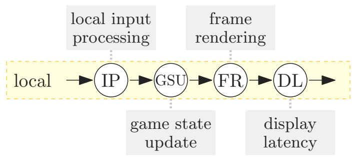
(a) External rendering (cloud) and external game state updates (server).



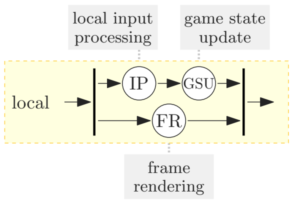
(b) External rendering (cloud) with cloud-local game state updates.



(c) External state updates (server) and local rendering.



(d) Local rendering and state updates.



(e) Example frame time (uncoupled model).

Figure 8: E2E lag for common scenarios (networked: yes/no, cloud: yes/no) and frame time model. Measurement points can be defined between any two individual delay components B_i.

local game state based on the information received from the authoritative game server. The current local view and preliminary state predictions need to be merged with the server's authoritative state.

e) Frame rendering (FR) and encoding: The frame rendering does take up a significant portion of the *frame time*. In particular, it is the time spent by the GPU to render the previously computed game state and scene. A new frame may only be rendered if the previous frame has been completed, potentially incurring additional delays. These frames may either be displayed directly[5] (Figure 8c and 8d) or encoded into a video in streaming setups (Figure 8a and 8b). Depending on the setup, the encoding may either be done in software, or by a dedicated hardware encoder such as `NVENC` [180].

f) Frame decoding (FDC): This step is only necessary in game streaming (Figure 8a and 8b). It consists of decoding the received frame and preparing it for display.

g) Display latency (DL): Finally, the display latency measures any delay after the frame leaves the local video card and is being processed by the display. It is influenced by the applied preprocessing steps, panel technology, and the refresh rate of the display panel. Modern displays with Variable Refresh Rate (VRR) can adjust their refresh rate to match the rendering rate in order to avoid artifacts or VSYNC delays, but also bringing another dynamicity into the display latency consideration.

h) Frame time: In contrast to the other delays, the frame time (or its inverse metric, frames per second) is not associated with a specific input or action taken by the user. It describes the time that is required to produce a new frame, with all associated computations necessary. As indicated by Figure 8e, this may include delay components from different actions, as the game state update process (GSU) may happen in parallel to the rendering process (FR) of the previous state's frame.

Note that more than one scenario of Figures 8a–8d might apply to an individual game, depending on the actual situation. Some actions could be predicted or executed locally, bypassing network components, in order to provide a smoother gaming experience, while others may need to travel the full network path before being displayed on screen.

B. Measurement Points and Metrics

Based on the models above, this section presents common measurement points in the literature and discusses their applicability to the presented scenarios. In general, four major measurement point classes can be derived, each allowing to obtain various metrics representing different performance aspects of their respective part of the system. These classes are later used to classify relevant literature in Table II.

a) Full End-to-End (E2E) lag d_{full}: This first class measures all occurring delays, beginning from the local button press up to the corresponding information being displayed on the screen. It is applicable to all four presented scenarios (Figures 8a–8d) and, depending on the scenario, measures the sum of all their respective delay components (from input processing to display latency). A high frame rate camera can be used to capture both the moment of the button press right before (IP) and the moment when the reaction is shown on the screen right after (DL). For a controlled measurement error, the camera speed should be at least twice as high as the refresh rate of the display, as the maximum measurement error is defined by the time between two camera frames. Often, the input device is modified to visually display the moment of button press via an external LED, e.g., hooked directly to the

[5]Directly may still mean storing the rendered frame in the video card's double or triple buffer before displaying it, further delaying the output but avoiding potential artifacts like frame tearing.

mouse. Contemporary variants of this method omit the camera in favor of a photoresistor affixed to the screen [181], or even provide a full closed loop solution that connect a photoresistor directly to a mouse[6] or with additional functionality in some modern monitors[7].

b) Game streaming network measurements: This specifically concerns network measurements for all applications of game streaming, i.e. Figures 8a and 8b. The measurement points revolve around the delay components (CIU) and (FDL), with the actual points depending on the obtained metric. When measuring the *Round-Trip Time (RTT)* d_{cloud}, the time difference between the moment before (CIU) and the moment after (FDL) is reported. Many measurements approximate this by simple `ping`-based RTT measurements, and thus omit the remaining components (CIP) to (FR). Besides taking into account the RTT as a mean delay (CIU + FDL) it is equally important to consider its variance over time (i.e. *jitter*) since it may also severely impact the experienced quality, even at an otherwise lower mean delay.

In addition to raw latency metrics, *packet loss* pl_{cloud} plays an important role in perceived gameplay smoothness. While packet loss on the receiving end (FDL) is often perceived as a visual degradation and artifacts in the output video stream, loosing an input data packet (CIU) can be a more disruptive experience as the expected reaction might not only be delayed but skipped entirely. Thus, typical approaches to ascertain the streaming video quality also apply here. Finally, the available *bandwidth* or used *data rate* br_{cloud} is an important metric, and is highly asymmetric. On the receiving end (FDL), the available bandwidth is directly related to the attainable quality of the video feedback from the cloud server. Note that input streams (CIU) typically consist of very frequent (e.g. $60\,\mathrm{Hz}$) but small packets. Hence their overhead should be considered, especially when redundancy is included to cope with potential loss.

c) Game server network measurements: The connection towards the authoritative game server (Figures 8a and 8c) can be measured with regard to *RTT* d_{serv}, i.e., mean delay (IU + GSD), jitter j_{serv} of this delay, *packet loss* pl_{serv} and required *bandwidth* (or bit rate) br_{serv}. Therefore, the measurement points are located right before (IU) and after (GSD) to measure the time difference between transmitting and receiving a network probing packet. Once again, the actual game state update (GSU) is commonly omitted and the reply is returned directly in order to isolate the networking delay during these measurements.

However, unlike in cloud gaming networking aspects, most online multiplayer games apply prediction techniques and predict preliminary game state updates in order to hide delays where possible. Therefore, network delay does not necessarily fully correlate with the subjectively experienced delay effect by a player of the game. In any case, the network delay still remains a crucial metric for a game's performance, as any such technique can lead to inconsistencies and rollbacks that are more severe at higher delays.

[6]NVIDIA LDAT: https://developer.nvidia.com/nvidia-latency-display-analysis-tool

[7]NVIDIA Reflex: https://developer.nvidia.com/reflex

d) Frame time measurements: Finally, *frame times* and *frames per second* can be measured in any of the considered scenarios (Figures 8a–8d). The appropriate measurement point is right after the rendering process (FR) has finished. The frame time measures the time between two *consecutive* unique frames, as opposed to comparing the timings of different measurement points like in the other metrics. However, it does represent more than just the pure GPU rendering time. It is influenced by *all* delay components that are located on the same machine as the one conducting the rendering process (FR), except for the external display (DL), i.e., (CIP), (GSP) and (FR) in Figure 8a; (CIP), (GSU) and (FR) in Figure 8b; (IP), (GSP) and (FR) in Figure 8c; and (IP), (GSU) and (FR) in Figure 8d. Depending on the engine's game loop model, these delays influence the frame time differently, e.g., for a coupled model (Figure 4a) it could be given by their sum (IP + GSU + FR). In an uncoupled model, such as in Figure 4b and 8e, a game's frames per second are usually bound by either the CPU or GPU, resulting in a frame time of $\max(\text{IP + GSU}, \text{FR})$. Some games are able to report the individual time portions of the CPU (IP + GSU) and GPU (FR), e.g. `DOOM` (2016), making them great candidates for in-depth measurements.

It should be noted that, while frame times and frames per second are the inverse of each other, the latter is often reported as a mean value over a larger period of time, e.g., an entire second. While this can still reflect the mean performance of a game, it fails to properly record stuttering or periodic hiccups in the rendering process, e.g., periodically skipping frames due to VSync [182]. Therefore, a time series of values or summary statistics beyond the mean, especially the standard deviation and the $99\,\%$ or even $99.9\,\%$ percentile, are the preferred approach to report frame times and frames per second.

A selection of related literature and community efforts that cover some of these measurement points is presented in Table II.

C. Lessons Learned

When conducting performance evaluations, including objective measurements and subjective evaluations, it is necessary to clearly communicate the applied measurement points and the type of observed scenario. Many of the described influence factors of the taxonomy are only visible at certain points of the chain, and could thus be overlooked if the measurement point is not carefully selected and known. Some delay components appear in multiple forms, such as networking and processing delays. Most importantly, the scenario should be properly identified and mapped to the corresponding model of Figure 8. Special care needs to be taken when considering delay mitigation and concealment techniques, since they are only visible in the full E2E lag, but also introduce speculative and even subjective components that can be very sensitive to the exact experimental design.

Some works listed in Table II have not mentioned their circumstances explicitly and fully. As such, only the measurement points were able to be identified clearly. But in many cases, the exact involved components and system parametrization of the study remained unclear, as this would require to have knowledge of the game's internal makeup (e.g. engine and programming). As a result, the surveyed performance evaluations are often inconclusive and open to interpretation with respect to some network performance requirements. Individual results sometimes contradict each other, possibly due to being based on different games—even if they are of the same genre—or slight variations in overlooked or unreported influencing factors of the respective experimental design.

In general, future evaluations should identify and control their selected scenario and measurement points as accurately as possible. More often than not, it might be necessary to measure the full E2E lag in order to not overlook any critical influences. In any case, evaluations should provide a comprehensive description of all involved components in between these measurements, while avoiding game- or genre-specific wording. To facilitate this, this section offers a common terminology of components and measurement points with which future works could be better compared. What's still missing now is to put these directly measurable effects into relation with the subjective effects that players experience. While subjective assessment will be covered in the next section, the mapping of QoS to QoE aspects for gaming is a more challenging topic and still mostly a matter of future work (and further discussed in Section VIII).

VII. Subjective User Studies

In this section we cover metrics and methods commonly used in subjective gaming user studies, beginning with network-related metrics that describe a negative impact on a video game. After that, we examine how QoE and in-game performance are affected when typical application layer problems occur.

Each user has different expectations when it comes to the perceived quality of a game subjected to certain additional influencing factors, as discussed in the taxonomy of Section II and with specific examples given in Section II-B. Some interaction effects can be described by the timeliness, precision, and predictability of the actions a player can execute in a game. Such player actions are impaired through lag, with specific games, and even specific actions within the same game, being more or less susceptible to lag. For example, competitive First-Person Shooters can be severely affected as they often benefit from quick reactions and good hand-eye coordination. But even in such games, lag might be more noticeable for some actions (e.g. targeting and shooting) than for others (e.g. unhindered movement). And it also depends on the specific game and how its actions are designed and implemented. As described previously, the general lag behavior can be derived and modeled with extensive measurements and gives insights into the interactions that contribute to a lag profile of various actions [98], [174]. However, this does not necessarily lead to a full understanding of the subjective experience under the given conditions as a multitude of additional influence factors exist as depicted in our taxonomy in Figure 2.

One attempt to summarily describe aspects of the subjective experience is by investigating *user engagement*, which has been the topic of several video game studies. An initial description of user engagement in video games is given in [206]. Here, the authors describe characteristics that motivate or engage someone to play, and continue playing. They conducted a user questionnaire with which they investigated which structural video game characteristics are important to players. According to [209] the engagement process consists of four stages: the point of engagement, the period of sustained engagement, disengagement and finally re-engagement. The extent of which different video game characteristics can be leveraged to predict user engagement and happiness has been studied in the past [207]. The authors came to the conclusion that audiovisual properties as well as punishment mechanics, like losing a life or having to start a level over, positively contributed to player happiness and the achievement of a flow state. Flow can also be described as an aspect of engagement, as a phase in which players achieve maximum immersion [208], [210].

The authors of [208] conducted a questionnaire which consists of 19 items to self-assess video game experiences and engagement. It has been suggested to be used as a measurement of video game engagement. Other studies suggested that engagement directly impacts happiness [211]. Finally, a survey on engagement and classification of engagement factors alongside consumer experience aspects has been conducted in [212]. In a follow-up study, the authors found that consumers first engage in playful consumption and gain experience from it, which leads to increased engagement [213].

Questionnaires have generally been an important tool to investigate the subjective natures of QoE and engagement for video games. Several works, including [208], [214]–[217], tackled the design of such questionnaires. Past works have demonstrated the challenges in their design. Questionnaires would need to define specific subjective factors they want to identify (e.g. flow, immersion, or frustration), and then design questions that can be unambiguously attributed to a single factor. The questions also need to be easily comprehensible by the subjects and need to be able to be mapped to a specific game in a specific experimental scenario (i.e. a post-hoc or intermittent approach). All of these challenges combined may explain why there is no one-size-fits-all solution available yet that encompasses all subjective aspects for gaming.

And the complexities of capturing all subjective impressions with questionnaires can also explain why it is often more promising to shift the investigation to player performance metrics, even though there are no reliable all-encompassing mappings from those player metrics to subjective experience developed yet. Therefore, the following sections highlight such works that investigated the influences of specific QoS factors on player performance metrics and QoE.

A. Delay Investigations

Inevitably, network impairments will be perceived by the player as delay, delay variation or stuttering. The authors of [202] investigated the influence of delay on various games using subjective and objective metrics. They concluded that both player performance as well as perceived quality decreased with increasing delay. A further user study performed in [201]

Table II: Metrics observed in important measurements and user studies on video games. The technical measures are defined in Section VI-B. The individual columns denote what features the individual study investigates. 'd' columns represent delay investigations, 'j' represents jitter, and 'pl' and 'br' denote packet loss and bitrate investigation respectively.

Reference		d_{full}	d_{cloud}	d_{serv}	j_{serv}	pl_{cloud}	pl_{serv}	br_{cloud}	br_{serv}	Frame rate	MOS	Player perf.	Play time	Special focus
Batt'19	[183]	✓	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗	various parameter studies with d_{full}
Zhao'17	[184]	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	head mounted display
Casi'15	[178]	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	no external camera, only mouse input
Ivko'15	[185]	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✗	lag compensation
Ware'94	[186]	✓	✗	✗	✗	✗	✗	✗	✗	✓	✗	✓	✗	VR, reaching behavior
Schm'17	[187]	✗	✓	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗	scenario classification
Sack'16	[188]	✗	✓	✗	✗	✗	✗	✓	✗	✗	✓	✗	✗	cloud gaming
Beye'15	[189]	(✓)	✓	✗	✗	✗	✗	✓	✗	✗	✓	✗	✗	cloud gaming, visual degradation, EEG power
Jars'13	[190]	✗	✓	✗	✗	✓	✗	✗	✗	✗	✓	✗	✗	cloud gaming
Clin'13	[191]	✗	✓	✗	✗	✓	✗	✗	✗	✗	✓	✗	✗	cloud gaming
Iqba'21	[192]	✓	✓	✗	✗	✓	✗	✓	✗	✗	✗	✗	✗	cloud gaming, delay components
Graf'21	[168]	✗	✓	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗	cloud gaming
Dome'21	[169]	✗	✗	✗	✗	✗	✗	✓	✗	✓	✗	✗	✗	cloud gaming
Lind'20	[193]	✗	✓	✗	✗	✓	✗	✗	✗	✗	✓	✓	✗	cloud gaming
Lee'12	[194]	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	command heaviness, fEMG potential
Jars'11	[7]	✗	✓	✗	✗	✓	✗	✗	✗	✗	✓	✗	✗	cloud gaming
Sliv'15	[195]	✗	✗	✗	✗	✗	✗	✓	✗	✓	✓	✓	✗	Steam in-home streaming, image quality
Clay'19	[196]	✗	✗	✓	✗	✗	✗	✗	✗	✗	✓	✓	✗	player accuracy
Hoan'17	[197]	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	✓	✗	MOBA
Beye'14	[198]	✗	✗	✓	✗	✗	✗	✗	✗	✗	✓	✗	✗	display size
Clay'10	[199]	✗	✗	✓	✗	✗	✗	✗	(✓)	✗	✗	✓	✗	game phases, player actions
Bred'10	[200]	✗	✗	✓	✓	✗	✓	✗	✗	✗	✗	✓	✗	connection type, FPS
Chen'08	[44]	✗	✗	✓	✓	✗	✓	✗	✗	✗	✗	✗	✓	MMORPG, large scale, player departure
Ries'08	[201]	✗	✗	✓	✓	✗	✗	✗	✗	✗	✓	✗	✗	MMORPG
Chen'06	[43]	✗	✗	✓	✓	✗	✓	✗	✗	✗	✗	✗	✓	MMORPG
Dick'05	[202]	✗	✗	✓	✓	✗	✗	✗	✗	✗	✓	✓	✗	survey of expectations
Beig'04	[203]	✗	✗	✓	✗	✗	✓	✗	✗	✗	✗	✓	✗	FPS
Nich'04	[204]	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	✓	✗	sports game
Shel'03	[54]	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	✓	✗	RTS
Pant'02	[53]	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	✓	✗	racing game
Clay'07	[205]	✗	✗	✗	✗	✗	✗	✗	✗	✓	✓	✓	✗	FPS
Wood'04	[206]	sound, graphics, background / setting, use of humor, brand assurance, character development, play time, duration of game, rate of play												structural characteristics
Leff'16	[207]	happiness, social, reward, punishment, narrative, manipulation, presentation, flow, psych. absorption, presence, immersion												structural characteristics, happiness
Broc'09	[208]	absorption, flow, presence, immersion, engagement												high school students

showed a correlation of QoE to the delay as well as the jitter in `World of Warcraft`. In this case the total delay had more impact than the delay variation. While looking at First-Person Shooters, the authors of [191] found a strong impact of the delay and packet loss on player experience. Ivkovic *et al.* [185] quantified the effect of local latency, including input, rendering, and output devices. They found that as the latency increased, the ability of the study participants to track targets decreased. Their results were qualitatively similar to a much earlier study [186] on the difficulty of reaching for static objects on virtual reality displays, conditioned to the virtual object's size and the motion-to-display lag.

Concerning cloud gaming, Chen et al. [162] already performed a suite of measurements on the first generation cloud gaming services and proposed a novel measurement methodology to assess the QoS—and especially the latency—of cloud gaming services. The same group of researchers also developed a testbed environment for cloud gaming [143], which can also be very helpful for delay investigations and other metrics. In another work, the authors of [218] give insights on delay requirements of streamed games and the implications for data center distance as well as placement. In [7] Jarschel *et al.* identified influence factors on the subjective quality of cloud gaming through a user survey for games in three different categories (slow, medium, fast games) that have been subjected to worsening QoS parameters. Downstream packet loss and delay was noted to be especially problematic for achieving good quality. Similarly, the authors of [44] observed a correlation of players that quit playing an MMOG with deteriorating QoS. A recent study [193] focused on faster and multiplayer games specifically affected by deteriorating network conditions. They find that the concept they dub 'frame age' (a portion of the total E2E delay) is highly correlated to the experienced quality. In 2021 the authors of [192] investigated three commercial cloud gaming services. Their novel evaluation approach was able to give estimates on specific game delay components, thus separating, e.g., the network delay from the cloud gaming server processing time. Depending on the game, the cloud server processing time was often much larger than the network delay. Additionally, the authors were able to demonstrate the bitrate adaptations these services perform under constrained conditions.

Many evaluation approaches also focus on in-game objective performance metrics, like highscores or task completion times. For example, a user study performed in [43] observed a decrease in the playing duration in an MMOG when the network QoS degraded. The authors of [52] categorized player actions and their relationship to latency with special regards for the precision and deadlines of actions. In the genre of Multiplayer Online Battle Arenas (MOBAs), [197], [219] measured the effects of lag on the player's ability to hit targets in `League of Legends`. The authors of [220], [221] investigated lag in `DOTA 2` using a novel crowdsourcing approach. Further, Claypool and Claypool [205] noted the influence of network QoS on in-game actions and specifically look at player performance in first person games. They observed significantly worse player performance in a degraded network.

As a further in-game performance metric, the "kills per minute" of players in the First-Person Shooter `Quake 3` were investigated by [222]. The study observed a steady decline of this metric with increasing network delay. QoE measurements from a custom-made racing video game again showed a strong dependence of the player's performance on the delay and suggest that a network RTT of $200\,\mathrm{ms}$ is barely usable and $500\,\mathrm{ms}$ completely unusable [53]. Finally, the authors of [200] found a strong and negative influence of high delay on the player's performance as well. Contrary, Beigbeder *et al.* [203], looked at player performance in `Unreal Tournament 2003` in a controlled in-game environment, and found almost no influence of increased delay. Player performance remained steady even at delay values of $200\,\mathrm{ms}$, indicating that the impact of delay on player performance varies between games and scenarios. Other influence factors besides the networking delay may be in play here, but were not investigated. This is supported by works like [174], that indicate that the networking delay does not necessarily need to be the dominating component of the total delay. Other factors, like the framerate, can play an equally important role and can not be neglected in delay investigations.

B. Jitter Investigations

The authors of [202] investigated the impact of jitter in First-Person Shooter (FPS) and racing online games by creating random delays for each packet between consecutive packets without correlation. They found that increasing the jitter led to a decrease in the Mean Opinion Score (MOS), but that its impact was much weaker compared to increased delay values. Similarly, [201] investigates the impact of jitter on the MOS for an MMORPG. They conclude that jitter is a performance indicator for MMORPGs and give it a higher negative impact factor in their QoE model than delay. A stronger impact of jitter was also observed on user departure behavior in an MMORPG by [44]. In another departure rate model for MMORPGs, [43] includes jitter as the highest impact factor in a regression equation. The authors of [200] studied the influence of different factors on the performance of games in automated tests without involving actual players. They found that jitter has little influence on the game score.

C. Packet Loss Investigations

In their large-scale study on player behavior in an MMORPG, [43] and [44] showed that packet loss in both directions, client-to-server and server-to-client, has a high impact on player departures. The authors of [203] investigated the impact of latency and packet loss in `Unreal Tournament 2003`. In their environment they did not detect an effect of packet loss on the accuracy or the overall score of players.

Some studies specific to cloud gaming came to different conclusions than what has been noted for packet loss in regular online gaming. For example, the authors of [7] and [190] noted strong video distortions even under low packet loss. Packet loss had the highest impact on QoE among all parameters in the study and was especially noticeable in slow and medium-paced games. Similarly, Clincy and Wilgor [191] observed that increased packet loss caused severe stuttering and led to a decrease in QoE when streaming a First-Person Shooter.

D. Bitrate and Video Quality Investigations

The bitrate, and other derived video quality metrics, are only applicable to streamed games. Locally rendered games will always just display the reference image (subject to the selected graphical settings in the game) without any visual degradation. The following works give an overview of past works that concern video game video quality.

The authors of [188] investigated the impact of the bitrate on the QoE in a cloud gaming system. They found that cloud gaming is a bandwidth-intensive application which requires a certain minimum bandwidth to maintain an acceptable visual quality. However, increasing the bandwidth beyond a certain threshold did not lead to a significant increase in QoE, as the QoE in an interactive environment is influence by more than just the visual quality. Importantly, they compared the QoE of actively playing participants to passive spectators and found that the passive viewers were more critical about video quality. This particular insight is often repeated when comparing passive to active media consumption.

The impact of the stream bitrate on the QoE was also investigated for `Steam`'s In-Home streaming [195] in 2015. The authors found a significant correlation between the video bitrate and the perceived graphics quality, but did not find any correlation with overall QoE or player performance in both a first person shooter and an action RPG. The authors of [189] examined Electroencephalography (EEG) readings in addition to a questionnaire to determine the impact of various influence factors on the QoE. Their test subjects were instructed to play with different video qualities settings (stated as $1\,\mathrm{Mbit/s}$ and $(10\,\mathrm{Mbit/s})$. They found that playing in low quality led to a slightly more tired state than high quality. In high quality, subjects experienced higher flow and immersion, and felt more competent and more pleasure. It should be noted that these values are much lower than what is considered standard in today's cloud gaming services, which currently stream with up to $50\,\mathrm{Mbit/s}$. A further overview of further QoE taxonomy and influence factors especially for mobile games is given in [198].

E. Application Layer Investigations

Besides networking-related properties, there are several game-related properties to keep in mind. The frame time or frame rate is an example of a key property described in Section VI-B, as it also interacts with and influences the E2E delay. Factors like display resolution, or even more abstract concepts like descriptions of the difficulty or tempo, can be key properties in subjective studies. Many games, especially PC games, also allow to change their graphical fidelity. Not only can this directly influence the enjoyment as visually more impressive games may give more enjoyment, but it also interacts with the performance and cloud gaming bitrate. Selecting higher quality settings usually results in higher visual complexity as well, in turn requiring a higher bitrate to maintain the same quality. Changing settings may also alter the game's framerate, and thus may introduce additional temporal influences that need to be considered. Preliminary investigations on this topic were conducted in [175], [223].

As an example, Slivar, Suznjevic, and Skorin-Kapov [195] found that the frame rate had a significant impact on the QoE in their experiments with `Steam` In-Home streaming. Other works in this area combined the examination of user ratings with questionnaire-based assessments to study diverse properties like detection, quality acceptance, difficulty, or retention and annoyance [187], [188]. A low frame rate led to a reduction in QoE here as well.

F. Lessons Learned

Due to their interactive nature, the experience of video games naturally relies heavily on subjective factors and the involved players. However, study results are influenced by not just these factors. On the contrary, other game-related, system or network influence factors have been shown to be equally influential, and should be considered for study design and during evaluations.

Furthermore, the impact of specific influence factors varies heavily, and depends on the specific study, the pursued goal as well as the specific game itself. For example, technical parameters such as tick rate and network delay may have negligible influence when it comes to slow, turn-based games, but are considered crucial key performance indicators for fast and competitive real-time games. Hence, a classification of the specific use case needs to be performed to identify relevant parameters for the goal of a measurement or user study. Studies should then be based on a diverse set of contemporary games that represent a cross-section of game influencing factors. For example, games of varying genre, tempo, modes, gameplay elements. But also games with different technical properties, from their graphical output and performance to how they implement network interactions.

Studies need to further make informed decision on their selection of metrics. Objective metrics that can represent visual and interaction quality, and assessment approaches that can capture the subjective experience related to that, such as well designed questionnaires. And finally, the interactive nature of games makes it all the more important to be able to accurately describe the players and their backgrounds—and include a diverse set of participants in the study—in order to put study results into perspective. Such experimental design can be performed with the help of insights gained from performance evaluations described in Section VI and from the influence factors described in the overall taxonomy throughout this manuscript. Initial approaches to testing guidelines are also available in standardization, e.g. in the form of ITU-T P.809 [19].

VIII. Future Research Directions

Before concluding this work, the following paragraphs summarize open research topics in the area of video games based on the observations made in this manuscript. While some of the following items may have no prior appearance in the research in this field, others need to be revisited in regular intervals since the landscape of video games is quickly changing, and previously observed characteristics, expectations, and insights may not hold true anymore when taking recent developments and new games or services into account.

A. QoS and QoE Measurements, Studies, and Models

The continued identification and assessment of general and game-specific video game quality metrics, player performance indicators, and subjective experiences remains crucial in future work. This concerns both the assessment of subjective experiences using questionnaires, interviews, or physiological measures as well as application layer QoS metrics. For the latter, popular examples from past research include the task completion time or game-specific scores under the influence of degraded QoS parameters. This is a direct continuation from the approaches and research works described in Section VI. Continuously validating and refining the results of previous studies with contemporary video games and new video game platforms that reflect the current state of the art is also an important aspect, especially when considering the age of some of the works referenced in this manuscript.

Of equal importance is the establishment of best practices and guidelines for experimental measurements and subjective studies specific to video games and beyond what the current literature and ITU recommendations provide. One concern relates to past gaming experiences, skills and expectations of subjects and how they should be factored into a study. Others consider for example the study length and training. Our taxonomy subsumes these into the player factors. Other best practices might concern selecting contemporary games with appropriate properties and a specific experimental and task design that can capture all desired information.

Using the data from prior and future game assessments QoS-to-QoE mappings can be constructed in order to generalize findings. Generalized models that relate network aspects to subjective quality could then serve in conjunction with network monitoring of online and streamed games to perform game-quality-aware network service management, which is of special interest for cloud gaming services.

The general, abstract approach to such mappings and models is depicted in Figure 9. The influencing factors from all aspects (see the taxonomy in Figure 2 for examples on concrete

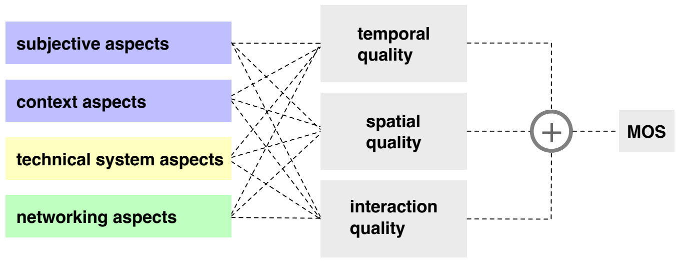


Figure 9: Abstract representation for a QoE model for online and cloud gaming. Quality is affected by aspects discussed in the taxonomy of Figure 2. Results from individual quality assements are then combined to a unified quality or MOS rating. Based on ITU-T G.1072 [224].

factors) affect the individual quality aspects (here separated into temporal, spatial and interaction). These individual quality aspects can then be combined again into a unified, overall quality scale for a specific scenario. One current concrete model for cloud gaming that follows this abstract approach is G.1072 [224]. The model linearly combines the results from separated quality assessments. But this may not be an ideal approach, since past subjective assessments have often experienced exponential reactions in response to a QoS degradation (see [225]). In addition, the strength with which a quality aspect is influenced by an influencing factor remains generally unclear and may depend on the specific circumstance. For example, depending on the specific game under test the network delay may play a significant role for the interaction quality, or it may not have an influence at all. This necessitates the need for further research and model development, and to take a closer look at the effects of influence factors.

B. Influence Factors and Parameter Interactions

The evaluation of correlations between subjective quality metrics of video games and external influence and context factors is crucial to further the understanding of QoE in video games. The taxonomy presented in this work is an abstract, qualitative approach to this issue, but a deeper understanding through future, quantitative evaluations of individual factors is mandated.

A chief endeavor to improve the significance of any individual game study is the categorization and clustering of video games based on observable, objective and technical properties, such as the game speed, temporal and spatial input precision or randomness. Results from past studies suggest that comparing results between video game QoE studies of games, even games of the same genre, may not be straightforward and depends on game factors beyond the genre. It especially depends on factors that describe a game's interaction characteristics, such as the tempo. Future research should thus evaluate appropriate metrics to quantify a game's tempo and define other metrics (e.g. metrics derived from our taxonomy) to classify and cluster games. These clusters could then serve to generalize results from studies and make them transferable. Metric candidates include the Actions per Minute (APM) for game tempo (see also [226]) and visual complexity metrics derived from video coding. Several works already pursue attempts to categorize games by player and game performance metrics [223], [227], [228]. Then, with such categories in hand, it might even be beneficial if category-specific metrics were introduced. For example, some high tempo categories (e.g. racing games) might rely on specific timings of certain actions, while this is completely irrelevant for games with no input deadlines (e.g. turn-based strategy). So, timing-based metrics will mostly not be applicable to the latter case.

C. Network-related Aspects

The network underlying any online or cloud-streamed game is always in flux, with new trends and approaches appearing all the time. While not specifically intended for gaming, recent trends have recognized the general need to provide an appropriate QoS to time-sensitive applications. This ranges from URLLC in 5G networks, to the standardization of Ethernet TSN by the IEEE, and especially to the efforts to provide low latency Internet in typical home environments with modern AQM (e.g. FQ-CoDel), WiFi improvements (akin to 802.11e), and improved transport protocols (e.g. IETF QUIC). The following four paragraphs highlight some of the most interesting network-related research areas for video games.

1) Novel transport protocols: Video game traffic characteristics and requirements should be taken into account when furthering the development and research of the Internet protocol stack. Online and streamed video games have specific needs with regards to the protocol stack that are different from many other Internet applications. However, these needs have not yet been exhaustively investigated and evaluated.

One main concern are future developments of the transport layer protocols and their aptitude for real-time communication. IETF QUIC [229], of which version 1 (which is still strongly tied to HTTP/3) was finalized in 2021, shows some very promising properties with regards to video games. With its optional reliability and non-blocking multi-stream capability it could better fulfill an online game's requirements than UDP or TCP.

Of further note are both the composition of games' application layer protocols and the influences of the access network. The delay variations of Wi-Fi networks can be especially detrimental to online and streamed games. Similar observations have already been made with other interactive traffic such as VoIP and video conferencing over Wi-Fi. Approaches that better coordinate the channel and fairly distribute radio resources can alleviate the issues of current real-time applications even in a shared access medium like Wi-Fi.

2) Network schedulers and shapers: Furthermore, appropriate transport layer AQM techniques could diminish the influence of the network on the E2E lag, since queuing delay can be a large contributor especially in congested, everyday situations. The concrete interactions of interactive application traffic with AQM and the optimal mechanisms are not yet conclusively researched. Especially game streaming could challenge any single FIFO queue system with its non-elastic high throughput and low delay demands. Thus, online video game and cloud gaming traffic can be a well suited use case for the evaluation of future AQM research. The detection and treatment of the demands of video game network flows by the schedulers could be improved when appropriately tagged with a Differentiated Services Code Point (DSCP), even though that may be bleached while traversing the Internet. A 2020 master's thesis [230] already touches on many of the potential benefits and challenges of using AQM and novel transport protocols for gaming with special attention to browser environments.

3) Lag mitigation and cloud gaming optimizations: Another aspect is to develop a deeper understanding of and further refinements to lag compensation, lag concealment and lag mitigation techniques in online video games. This includes their potential side-effects like cheating or rollbacks. Moreover, the existing approaches are designed to work for conventional multiplayer online games, not game streaming environments.

It would also be of considerable interest to advance the research on lag mitigation techniques that could be applied to cloud gaming. Initial ideas aim to predict user inputs at the streaming host and pre-compute and pre-render the resulting game states in order to 'skip' the network delay [61], [231], [232], enable the client to apply further post-processing steps on the received video stream, e.g. [233], or even attempt to modify the game world and scale the difficulty according to current lag [234].

The ongoing research surrounding cloud gaming goes well beyond pure lag optimizations. Particular interest resides also on the reduction of the high throughput dependence, primarily by tasking the client with more than just simple video decoding, e.g. moving parts of the game logic or rendering process back to the client. Examples of these approaches can be found in [235]–[238]. Although, none of these approaches has been realized in a practical system yet, as they can impose drastic overhead or interfere with game design and development decisions.

4) Server placement: A final networking challenge is the placement of game servers. The placement of online game servers and game streaming hosts are different problems with different objectives. While the main concern of placing online game servers is a low network delay to its players, its additional objective is a placement in the geographical center of a large audience to facilitate matchmaking. A good example of the possible approaches for delay optimizing placements can be found in [239], here with a focus on machine learning. In contrast, game streaming placement is much more concerned with delay minimization alongside providing sufficient throughput while considering economic and resource factors. Streaming servers can be much more resource intense in terms of hardware, power and network while demanding even lower delays than online game servers.

A low tolerance to delay implies very localized or edge placement, but would diminish the potential efficiency gains that larger, more centralized data centers would offer. However, the effectiveness of modern edge cloud and fog architectures might be worth to investigate, as they might be able to dynamically allocate the necessary resources to facilitate cloud gaming on a case-by-case and demand-orientated basis. In any case, the suitability of different networking and data center architectures for all types of video game servers is an important task.

Generally speaking, an optimal placement first requires a consensus of the relevant optimization criteria and on values that are suitable for gaming (e.g. a low delay and jitter). With these at hand, multi-objective placement optimizations can be performed, which pose an interesting research challenge themselves.

IX. Conclusion

It should be clear at this point that examining the QoS and QoE of video games from a network researcher's perspective is no trivial task. Not only do the ever increasing prevalence and economic importance make considering online video games important for network planning and traffic engineering, even beyond the current strong trend of cloud gaming. The interactive and real-time nature of gaming itself brings unique properties and challenges to the table. While the perceived quality of these games at first glance closely relates to network QoS, there are many more aspects at play as captured in our taxonomy.

The development of generally applicable models for the perceived quality of online video games under the influence of subjective, context, system, and network influence factors is crucial to further our understanding of video game QoE. However, the heterogeneity of modern games as well as the rate of innovation in the area make the definition of such models an increasingly complex task. To remediate this situation as well as to provide the networking community with a condensed introduction to the area of video games, we conducted a detailed survey of current research work in the area.

This is combined with a taxonomy that compiles and dissects subjective, contextual, technical, and network influence factors of video game QoS and QoE. The taxonomy forms a foundation that can be taken into account when conducting video game studies. Based on these two approaches we were able to derive a number of lessons learned that can provide utility for future studies. These lessons suggest to perform exhaustive parameter investigations to determine the influencing factors (e.g. as given by our taxonomy) that are significant for a specific experiment, and to select appropriate application performance metrics or subjective assessment tools. Both of these require a firm grasp of the chosen game's inner workings, its placement in the greater landscape of video games and in-depth knowledge of influencing and contextual factors.

Finally, we show that there are numerous open topics that require additional research. And advancing research on networking in video games can also assist research on other, closely related interactive applications with similar demands, from industrial and automotive remote control to tele-medicine. We believe that our discussion of available work and the derived taxonomy of QoE and QoS influence factors can direct researchers towards these extremely interesting topics.

If the reader is now left wanting to know more about the subject we can recommend some avenues to remedy this. To further one's understanding of the internal makeup of online video games, both the academic as well as the game developer perspective are worth looking into. For the former, summary works like Armitage's 2006 book [10] are available, while the latter can be approached by more contemporary books like [12] or by the numerous tech articles referenced throughout this manuscript. But if one is, understandably, more interested in the current landscape of gaming user study approaches and study results, we would strongly recommend starting with the current collection of recommendations and efforts as organized by the ITU-T, in particular [14], [18], [19], [224].

REFERENCES

[1] SuperData, *2019 year in review: Digital games and interactive media*, 2020.

[2] J. Porter, *US consumers spent record amounts on video games in 2020, NPD reports*, https://www.theverge.com/2021/1/15/22233003/us-npd-group-video-game-spending-2020-record-nintendo-switch-call-of-duty-animal-crossing-ps5-ps4, Accessed: 2021-11-16, 2014.

[3] *Cisco annual internet report (2018–2023) white paper*, Accessed: 2021-11-16, 2020. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html.

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

[5] L. Skorin-Kapov, M. Varela, T. Hoßfeld, and K.-T. Chen, "A survey of emerging concepts and challenges for QoE management of multimedia services," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, vol. 14, no. 2s, pp. 1–29, 2018.

[6] K. Brunnström *et al.*, "Qualinet white paper on definitions of quality of experience," P. L. Callet, S. Möller, and A. Perkis, Eds., 2013.

[7] M. Jarschel, D. Schlosser, S. Scheuring, and T. Hoßfeld, "An evaluation of QoE in cloud gaming based on subjective tests," in *2011 Fifth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*, IEEE, 2011, pp. 330–335.

[8] W. Cai *et al.*, "A survey on cloud gaming: Future of computer games," *IEEE Access*, vol. 4, pp. 7605–7620, 2016.

[9] A. Wahab, N. Ahmad, M. G. Martini, and J. Schormans, "Subjective quality assessment for cloud gaming," *J*, vol. 4, no. 3, pp. 404–419, 2021. DOI: 10.3390/j4030031. [Online]. Available: https://www.mdpi.com/2571-8800/4/3/31.

[10] G. Armitage, M. Claypool, and P. Branch, *Networking and online games: understanding and engineering multiplayer Internet games*. John Wiley & Sons, 2006.

[11] J. Smed, T. Kaukoranta, and H. Hakonen, "Aspects of networking in multiplayer computer games," *The Electronic Library*, vol. 20, no. 2, pp. 87–97, 2002.

[12] No Bugs Hare, *Development and Deployment of Multiplayer Online Games, Volume I: GDD, Authoritative Servers, Communications*. ITHare.com Website GmbH, 2017.

[13] I. Wechsung and K. De Moor, "Quality of experience versus user experience," in *Quality of Experience: Advanced Concepts, Applications and Methods*, S. Möller and A. Raake, Eds. Cham: Springer International Publishing, 2014, pp. 35–54. DOI: 10.1007/978-3-319-02681-7_3. [Online]. Available: https://doi.org/10.1007/978-3-319-02681-7_3.

[14] S. Schmidt, S. Zadtootaghaj, and S. Möller, *ITU-T standardization activities targeting gaming quality of experience*, 2021. [Online]. Available: https://records.sigmm.org/2021/03/24/itu-t-standardization-activities-targeting-gaming-quality-of-experience/.

[15] A. Perkis *et al.*, *Qualinet white paper on definitions of immersive media experience (IMEx)*, 2020. arXiv: 2007.07032 [cs.MM]. [Online]. Available: https://arxiv.org/abs/2007.07032.

[16] *Proposal for new work item P.BBQCG: Parametric bitstream-based quality assessment of cloud gaming services*, ITU-T SG 12 (Study Period 2017) Contribution 489, Apr. 2020. [Online]. Available: https://www.itu.int/md/T17-SG12-C-0489/en.

[17] S. Möller, S. Schmidt, and J. Beyer, "Gaming taxonomy: An overview of concepts and evaluation methods for computer gaming QoE," in *2013 Fifth International Workshop on Quality of Multimedia Experience (QoMEX)*, 2013, pp. 236–241. DOI: 10.1109/QoMEX.2013.6603243.

[18] *ITU-T recommendation G.1032: Influence factors on gaming quality of experience*, SERIES G: TRANSMISSION SYSTEMS AND MEDIA, DIGITAL SYSTEMS AND NETWORKS, Oct. 2017. DOI: 11.1002/1000/13396. [Online]. Available: http://handle.itu.int/11.1002/1000/13396.

[19] *ITU-T recommendation P.809: Subjective evaluation methods for gaming quality*, SERIES P: TELEPHONE TRANSMISSION QUALITY, TELEPHONE INSTALLATIONS, LOCAL LINE NETWORKS, Jun. 2018. [Online]. Available: https://www.itu.int/rec/T-REC-P.809/en.

[20] A. Nylund and O. Landfors, "Frustration and its effect on immersion in games : A developer viewpoint on the good and bad aspects of frustration," M.S. thesis, Umeå University, Department of Informatics, 2015, p. 32.

[21] M. Zamith *et al.*, "A game loop architecture with automatic distribution of tasks and load balancing between processors," *Proceedings of SBGames*, pp. 5–8, 2009.

[22] L. Valente, A. Conci, and B. Feijó, "Real time game loop models for single-player computer games," in *Proceedings of the IV Brazilian Symposium on Computer Games and Digital Entertainment*, vol. 89, 2005, p. 99.

[23] D. Sanchez-Crespo and D. S.-C. Dalmau, *Core techniques and algorithms in game programming*. New Riders, 2004.

[24] J. Gregory, *Game engine architecture*, 3rd ed. AK Peters/CRC Press, 2018.

[25] J. van Waveren, "The DOOM III network architecture," Id Software, Inc., Tech. Rep., 2006, Accessed: 2021-11-16. [Online]. Available: http://mrelusive.com/publications/papers/The-DOOM-III-Network-Architecture.pdf.

[26] G. Fiedler, *Snapshot interpolation*, https://gafferongames.com/post/snapshot_interpolation/, Accessed: 2021-11-16, 2014.

[27] Valve Developer Community, *Source multiplayer networking*, https://developer.valvesoftware.com/wiki/Source_Multiplayer_Networking, Accessed: 2021-11-16, 2005.

[28] M. Frohnmayer and T. Gift, "The TRIBES engine networking model: How to make the internet rock for multi-player games," in *Proceedings of the Game Developers Conference*, 2000. [Online]. Available: https://www.gamedevs.org/uploads/tribes-networking-model.pdf (visited on 11/16/2021).

[29] G. Fiedler, *State synchronization*, https://gafferongames.com/post/state_synchronization/, Accessed: 2021-11-16, 2015.

[30] Riot Games, *Matchmaking guide*, Accessed: 2021-11-16, 2019. [Online]. Available: https://support-leagueoflegends.riotgames.com/hc/en-us/articles/201752954-Matchmaking-Guide.

[31] A. E. Elo, *The rating of chessplayers, past and present*. Arco Pub., 1978.

[32] R. Herbrich, T. Minka, and T. Graepel, "Trueskill(tm): A bayesian skill rating system," in *Advances in Neural Information Processing Systems 20*, 2007.

[33] T. Minka, R. Cleven, and Y. Zaykov, "Trueskill 2: An improved bayesian skill rating system," Microsoft, Tech. Rep. MSR-TR-2018-8, 2018. [Online]. Available: https://www.microsoft.com/en-us/research/publication/trueskill-2-improved-bayesian-skill-rating-system/ (visited on 11/16/2021).

[34] J. van Dongen, *Why good matchmaking requires enormous player counts*, Accessed: 2021-11-16, 2014. [Online]. Available: http://joostdevblog.blogspot.com/2014/11/why-good-matchmaking-requires-enormous.html.

[35] M. Terrano and P. Bettner, "1500 archers on a 28.8: Network programming in Age of Empires and beyond," *Game Developer*, 2001. [Online]. Available: https://www.gamedeveloper.com/programming/1500-archers-on-a-28-8-network-programming-in-age-of-empires-and-beyond (visited on 11/16/2021).

[36] C. D. Nguyen, F. Safaei, and P. Boustead, "Optimal assignment of distributed servers to virtual partitionsfor the provision of immersive voice communicationin massively multiplayer games," *Computer Communications*, vol. 29, no. 9, pp. 1260–1270, 2006.

[37] G. Papp and C. GauthierDickey, "Characterizing multiparty voice communication for multiplayer games," in *ACM SIGMETRICS Performance Evaluation Review*, vol. 36, 2008, pp. 465–466.

[38] T. Triebel, B. Guthier, T. Plotkowiak, and W. Effelberg, "Peer-to-peer voice communication for massively multiplayer online games," in *Proceedings of the 6th IEEE Consumer Communications and Networking Conference (CCNC)*, 2009.

[39] M. Kaytoue, A. Silva, L. Cerf, W. Meira Jr, and C. Raïssi, "Watch me playing, i am a professional: A first study on video game live

streaming," in *Proceedings of the 21st International Conference on World Wide Web*, ACM, 2012, pp. 1181–1188.

[40] K. Pires and G. Simon, "Dash in twitch: Adaptive bitrate streaming in live game streaming platforms," in *Proceedings of the 2014 Workshop on Design, Quality and Deployment of Adaptive Video Streaming*, ACM, 2014.

[41] J. Deng, F. Cuadrado, G. Tyson, and S. Uhlig, "Behind the game: Exploring the twitch streaming platform," in *Proceedings of the International Workshop on Network and Systems Support for Games (NetGames)*, 2015.

[42] I. Rec, "G.1030-estimating end-to-end performance in IP networks for data applications," *International Telecommunication Union, Geneva, Switzerland*, vol. 42, 2005.

[43] K.-T. Chen, P. Huang, and C.-L. Lei, "How sensitive are online gamers to network quality?" *Communications of the ACM*, vol. 49, no. 11, pp. 34–38, 2006.

[44] K.-T. Chen, P. Huang, and C.-L. Lei, "Effect of network quality on player departure behavior in online games," *IEEE Transactions on Parallel and Distributed Systems*, vol. 20, no. 5, pp. 593–606, 2008.

[45] P. Lincroft, *The internet sucks: Or, what i learned coding X-Wing vs. TIE Fighter*, 1999. [Online]. Available: https://www.gamedeveloper.com/design/the-internet-sucks-or-what-i-learned-coding-x-wing-vs-tie-fighter (visited on 11/16/2021).

[46] Factorio Blog, *Friday facts nr. 149 - deep down in multiplayer*, https://www.factorio.com/blog/post/fff-149, Accessed: 2021-11-16, 2016.

[47] G. Fiedler, *Deterministic Lockstep*, https://gafferongames.com/post/deterministic_lockstep/, Accessed: 2021-11-16, 2014.

[48] J. D. Pellegrino and C. Dovrolis, "Bandwidth requirement and state consistency in three multiplayer game architectures," in *Proceedings of the 2Nd Workshop on Network and System Support for Games*, ser. NetGames '03, Redwood City, California, 2003.

[49] S. Ratti, B. Hariri, and S. Shirmohammadi, "A survey of first-person shooter gaming traffic on the internet," *IEEE Internet Computing*, vol. 14, no. 5, pp. 60–69, 2010.

[50] X. Che and B. Ip, "Packet-level traffic analysis of online games from the genre characteristics perspective," *Journal of Network and Computer Applications*, vol. 35, no. 1, pp. 240–252, 2012.

[51] M. Suznjevic and M. Matijasevic, "Player behavior and traffic characterization for MMORPGs: A survey," *Multimedia Systems*, vol. 19, no. 3, pp. 199–220, Aug. 2013.

[52] 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: https://doi.org/10.1145/1167838.1167860.

[53] L. Pantel and L. C. Wolf, "On the impact of delay on real-time multiplayer games," in *Proceedings of the 12th international workshop on Network and operating systems support for digital audio and video*, ACM, 2002, pp. 23–29.

[54] N. Sheldon, E. Girard, S. Borg, M. Claypool, and E. Agu, "The effect of latency on user performance in Warcraft III," in *Proceedings of the 2nd workshop on Network and system support for games*, ACM, 2003, pp. 3–14.

[55] T. Fritsch, H. Ritter, and J. Schiller, "The effect of latency and network limitations on MMORPGs: A field study of Everquest 2," in *Proceedings of 4th ACM SIGCOMM Workshop on Network and System Support for Games*, ser. NetGames '05, Hawthorne, NY, 2005.

[56] A. A. Laghari, H. He, K. A. Memon, R. A. Laghari, I. A. Halepoto, and A. Khan, "Quality of experience (QoE) in cloud gaming models: A review," *multiagent and grid systems*, vol. 15, no. 3, pp. 289–304, 2019.

[57] I. Pelle, J. Czentye, J. Dóka, and B. Sonkoly, "Towards latency sensitive cloud native applications: A performance study on aws," in *2019 IEEE 12th International Conference on Cloud Computing (CLOUD)*, IEEE, 2019, pp. 272–280.

[58] S. K. Barker and P. Shenoy, "Empirical evaluation of latency-sensitive application performance in the cloud," in *Proceedings of the first annual ACM SIGMM conference on Multimedia systems*, 2010, pp. 35–46.

[59] G. Brown *et al.*, "Ultra-reliable low-latency 5G for industrial automation," *Technol. Rep. Qualcomm*, vol. 2, p. 52 065 394, 2018.

[60] M. Seufert, S. Egger, M. Slanina, T. Zinner, T. Hoßfeld, and P. Tran-Gia, "A survey on quality of experience of HTTP adaptive streaming," *IEEE Communications Surveys Tutorials*, vol. 17, no. 1, pp. 469–492, 2015.

[61] N. Nicholson, *Exploring "negative latency"*, Accessed: 2021-11-16, 2019. [Online]. Available: https://nolannicholson.com/2019/12/16/exploring-negative-latency.html.

[62] M. Carrascosa and B. Bellalta, "Cloud-gaming: Analysis of Google Stadia traffic," *arXiv preprint arXiv:2009.09786*, 2020.

[63] D. Patterson, *Google Stadia's biggest challenge with streaming and meeting gamers' expectations*, Accessed: 2021-11-16, 2020. [Online]. Available: https://www.techrepublic.com/article/google-stadias-biggest-challenge-with-steaming-and-meeting-gamers-expectations/.

[64] K. Nichols, S. Blake, F. Baker, and D. Black, *RFC 2474: Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 headers*, Dec. 1998. [Online]. Available: https://tools.ietf.org/html/rfc2474.

[65] R. Adams, "Active queue management: A survey," *IEEE communications surveys & tutorials*, vol. 15, no. 3, pp. 1425–1476, 2012.

[66] G. Fiedler, *Never trust the client*, http://web.archive.org/web/20160427205903/http://gafferongames.com/2016/04/25/never-trust-the-client/, Accessed: 2021-11-16, 2016.

[67] C. Neumann, N. Prigent, M. Varvello, and K. Suh, "Challenges in peer-to-peer gaming," *ACM SIGCOMM Computer Communication Review*, vol. 37, no. 1, pp. 79–82, 2007.

[68] A. Weinkove, *Minimizing the pain of lockstep multiplayer*, http://www.tundragames.com/minimizing-the-pain-of-lockstep-multiplayer/, Accessed: 2021-11-16, 2015.

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

[70] F. Smith, *Synchronous rts engines and a tale of desyncs*, https://www.forrestthewoods.com/blog/synchronous_rts_engines_and_a_tale_of_desyncs/, Accessed: 2021-11-16, 2011.

[71] P. Miller, *The lag-fighting techniques behind GGPO's netcode*, https://www.gamedeveloper.com/programming/the-lag-fighting-techniques-behind-ggpo-s-netcode, Accessed: 2021-11-16, 2012.

[72] G. Gambetta, *Fast-paced multiplayer (part ii): Client-side prediction and server reconciliation*, 2013. [Online]. Available: https://www.gabrielgambetta.com/client-side-prediction-server-reconciliation.html (visited on 11/16/2021).

[73] R. Pusch, "Explaining how fighting games use delay-based and rollback netcode," *Ars Technica*, Oct. 2019. [Online]. Available: https://arstechnica.com/gaming/2019/10/explaining-how-fighting-games-use-delay-based-and-rollback-netcode/ (visited on 11/16/2021).

[74] B. House, *Choosing the right netcode for your game*, https://blogs.unity3d.com/2020/09/08/choosing-the-right-netcode-for-your-game/, Accessed: 2021-05-21, 2020.

[75] F. W. Li, R. W. Lau, and D. Kilis, "GameOD: An internet based game-on-demand framework," in *Proceedings of the ACM symposium on Virtual reality software and technology*, ACM, 2004, pp. 129–136.

[76] G. Fiedler, *Snapshot compression*, Accessed: 2021-11-16, 2015. [Online]. Available: https://gafferongames.com/post/snapshot%5C_compression/.

[77] P. A. Branch, A. L. Cricenti, and G. J. Armitage, "A markov model of server to client IP traffic in first person shooter games," in *2008 IEEE International Conference on Communications*, IEEE, 2008, pp. 5715–5720.

[78] D. Stefyn, A. Cricenti, P. Branch, *et al.*, "Quake III Arena game structures," *CAIA Technical Report 110209A*, 2011.

[79] J. Saldana, L. Sequeira, J. Fernández-Navajas, and J. Ruiz-Mas, "Traffic optimization for TCP-based massive multiplayer online games," in *2012 International Symposium on Performance Evaluation of Computer & Telecommunication Systems (SPECTS)*, IEEE, 2012, pp. 1–8.

[80] K. L. Morse, L. Bic, and M. Dillencourt, "Interest management in large-scale virtual environments," *Presence: Teleoperators & Virtual Environments*, vol. 9, no. 1, pp. 52–68, 2000.

[81] S. Benford, C. Greenhalgh, T. Rodden, and J. Pycock, "Collaborative virtual environments," *Association for Computing Machinery. Communications of the ACM*, vol. 44, no. 7, pp. 79–79, 2001.

[82] F. Sanglard, *Quake 3 source code review: Network model (part 3 of 5)*, https://fabiensanglard.net/quake3/network.php, Accessed: 2021-11-16, 2012.

[83] J. Aronson, "Dead reckoning: Latency hiding for networked games," *Game Developer*, 1997. [Online]. Available: https://www.gamedeveloper.com/programming/dead-reckoning-latency-hiding-for-networked-games (visited on 11/16/2021).

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

[85] P. B. Beskow, K.-H. Vik, P. Halvorsen, and C. Griwodz, "The partial migration of game state and dynamic server selection to reduce latency," *Multimedia Tools and Applications*, vol. 45, no. 1-3, pp. 83–107, 2009.

[86] V. Jalaparti, "Enabling seamless wide area migration of online games," University of Illinois, Tech. Rep., 2013, Master's thesis.

[87] V. Burger *et al.*, "Load dynamics of a multiplayer online battle arena and simulative assessment of edge server placements," in *Proceedings of the 7th International Conference on Multimedia Systems*, ser. MMSys '16, Klagenfurt, Austria, 2016.

[88] C.-C. Wu, K.-T. Chen, C.-M. Chen, P. Huang, and C.-L. Lei, "On the challenge and design of transport protocols for MMORPGs," *Multimedia Tools and Applications (special issue on Massively Multiuser Online Gaming Systems and Applications)*, pp. 7–32, 2009.

[89] C. Lernö, *Game servers: UDP vs. TCP*, https://1024monkeys.wordpress.com/2014/04/01/game-servers-udp-vs-tcp/, Accessed: 2021-11-16, 2014.

[90] G. Fiedler, *UDP vs. TCP*, https://gafferongames.com/post/udp_vs_tcp/, Accessed: 2021-11-16, 2014.

[91] G. Fiedler, *Reliability and congestion avoidance over UDP*, https://gafferongames.com/post/reliable_ordered_messages/, Accessed: 2021-11-16, 2008.

[92] F. Sanglard, *Quake engine code review : Network (2/4)*, https://fabiensanglard.net/quakeSource/quakeSourceNetWork.php, Accessed: 2021-11-16, 2009.

[93] K.-T. Chen, C.-Y. Huang, P. Huang, and C.-L. Lei, "An empirical evaluation of TCP performance in online games," in *Proceedings of the 2006 ACM SIGCHI international conference on Advances in computer entertainment technology*, 2006.

[94] S. Pack, E. Hong, Y. Choi, J.-S. Kim, D. Ko, *et al.*, "Game transport protocol: A reliable lightweight transport protocol for massively multiplayer online games (MMPOGs)," in *Multimedia systems and Applications V*, International Society for Optics and Photonics, vol. 4861, 2002, pp. 83–94.

[95] B. Anand, J. Sebastian, S. Y. Ming, A. L. Ananda, M. C. Chan, and R. K. Balan, "PGTP: Power aware game transport protocol for multi-player mobile games," in *2011 International Conference on Communications and Signal Processing*, IEEE, 2011, pp. 399–404.

[96] S. Mortenson, "Making a multiplayer game with go and grpc," 2020. [Online]. Available: https://mortenson.coffee/blog/making-multiplayer-game-go-and-grpc/ (visited on 11/15/2021).

[97] M. Claypool, D. LaPoint, and J. Winslow, "Network analysis of counter-strike and starcraft," in *Conference Proceedings of the 2003 IEEE International Performance, Computing, and Communications Conference, 2003.*, Apr. 2003, pp. 261–268. DOI: 10.1109/PCCC.2003.1203707.

[98] F. Metzger and R. Heger, "Exploring the transmission behaviour of overwatch: The source of lag," in *2018 30th International Teletraffic Congress (ITC 30)*, vol. 01, Sep. 2018, pp. 93–96. DOI: 10.1109/ITC30.2018.00022.

[99] A. F. Ali, A. S. Ismail, and A. Bade, "An overview of networking infrastructures for massively multiplayer online games," 5th Postgraduate Annual Research Seminar (PARS), UTM Technology University of Malaysia, Tech. Rep., 2009, Accessed: 2021-05-21. [Online]. Available: https://pdfs.semanticscholar.org/887b/374b1010ca23a01bb89b71e40fd31e34ce92.pdf.

[100] L. Ricci and E. Carlini, "Distributed virtual environments: From client server to cloud and P2P architectures," in *Proceedings of the International Conference on High Performance Computing Simulation (HPCS)*, 2012.

[101] R. A. Bangun and H. Beadle, "A network architecture for multiuser networked games on demand," in *Proceedings of International Conference on Information, Communications and Signal Processing*, IEEE, vol. 3, 1997, pp. 1815–1819.

[102] W. Cai, P. Xavier, S. J. Turner, and B.-S. Lee, "A scalable architecture for supporting interactive games on the internet," in *Proceedings of the Sixteenth Workshop on Parallel and Distributed Simulation*, ser. PADS '02, 2002.

[103] S. Fiedler, M. Wallner, and M. Weber, "A communication architecture for massive multiplayer games," in *Proceedings of the 1st Workshop on Network and System Support for Games (NetGames)*, Braunschweig, Germany, 2002.

[104] C.-c. A. Hsu, J. Ling, Q. Li, and C.-C. J. Kuo, "The design of multiplayer online video game systems," in *Multimedia Systems and Applications VI*, A. G. Tescher, B. Vasudev, V. M. B. Jr., and A. Divakaran, Eds., International Society for Optics and Photonics, vol. 5241, SPIE, 2003, pp. 180–191. DOI: 10.1117/12.512201. [Online]. Available: https://doi.org/10.1117/12.512201.

[105] B. Ng, F. W. Li, R. W. Lau, A. Si, and A. Siu, "A performance study on multi-server DVE systems," *Information Sciences*, 2003.

[106] T. Wang, C.-L. Wang, and F. C. Lau, "A grid-enabled multi-server network game architecture," in *3rd International Conference on Application and Development of Computer Games (ADCOG)*, 2004, pp. 18–25.

[107] J. Chen, B. Wu, M. Delap, B. Knutsson, H. Lu, and C. Amza, "Locality aware dynamic load management for massively multiplayer games," in *Proceedings of the Tenth ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, ser. PPoPP '05, 2005.

[108] K.-W. Lee, B.-J. Ko, and S. Calo, "Adaptive server selection for large scale interactive online games," *Computer Networks*, vol. 49, no. 1, pp. 84–102, 2005, Networking Issue in Entertainment Computing. DOI: https://doi.org/10.1016/j.comnet.2005.04.006. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128605001015.

[109] M. Assiotis and V. Tzanov, "A distributed architecture for MMORPG," in *Proceedings of 5th ACM SIGCOMM Workshop on Network and System Support for Games*, ser. NetGames '06, 2006.

[110] Duong Nguyen Binh Ta and Suiping Zhou, "Efficient client-to-server assignments for distributed virtual environments," in *Proceedings 20th IEEE International Parallel Distributed Processing Symposium*, 2006.

[111] F. Glinka, A. Ploß, J. Müller-lden, and S. Gorlatch, "Rtf: A real-time framework for developing scalable multiplayer online games," in *Proceedings of the 6th ACM SIGCOMM Workshop on Network and System Support for Games*, ser. NetGames '07, 2007.

[112] A. Ploss, S. Wichmann, F. Glinka, and S. Gorlatch, "From a single- to multi-server online game: A Quake 3 case study using RTF," in *Proceedings of the 2008 International Conference on Advances in Computer Entertainment Technology*, ser. ACE '08, 2008.

[113] A. M. Khan, I. Arsov, M. Preda, S. Chabridon, and A. Beugnard, "Adaptable client-server architecture for mobile multiplayer games," in *Proceedings of the 3rd International ICST Conference on Simulation Tools and Techniques*, ser. SIMUTools '10, 2010.

[114] R. Prodan and A. Iosup, "Operation analysis of massively multiplayer online games on unreliable resources," *Peer-to-Peer Networking and Applications*, vol. 9, no. 6, pp. 1145–1161, 2016.

[115] C. Diot and L. Gautier, "A distributed architecture for multiplayer interactive applications on the internet," *IEEE Network*, vol. 13, no. 4, pp. 6–15, 1999.

[116] Dugki Min, E. Choi, Donghoon Lee, and Byungseok Park, "A load balancing algorithm for a distributed multimedia game server architecture," in *Proceedings IEEE International Conference on Multimedia Computing and Systems*, 1999.

[117] C. GauthierDickey, D. Zappala, V. Lo, and J. Marr, "Low latency and cheat-proof event ordering for peer-to-peer games," in *Proceedings of the 14th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV)*, 2004.

[118] T. Iimura, H. Hazeyama, and Y. Kadobayashi, "Zoned federation of game servers: A peer-to-peer approach to scalable multi-player online games," in *Proceedings of 3rd ACM SIGCOMM Workshop on Network and System Support for Games*, ser. NetGames '04, 2004.

[119] B. Knutsson, H. Lu, W. Xu, and B. Hopkins, "Peer-to-peer support for massively multiplayer games," in *IEEE INFOCOM*, 2004.

[120] S. Rooney, D. Bauer, and R. Deydier, "A federated peer-to-peer network game architecture," *IEEE Communications Magazine*, vol. 42, no. 5, pp. 114–122, 2004.

[121] A. El Rhalibi and M. Merabti, "Agents-based modeling for a peer-to-peer MMOG architecture," *ACM Computers in Entertainment*, vol. 3, no. 2, 2005.

[122] F. R. Wagner, M. G. Martins, and A. T. Gómez, "A peer to peer architecture applied to multiplayer games," *Proceedings of the 14th International Conference on Networks (ICN)*, 2015.

[123] A. P. Yu and S. T. Vuong, "MOPAR: A mobile peer-to-peer overlay architecture for interest management of massively multiplayer online games," in *Proceedings of the International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV)*, 2005.

[124] A. R. Bharambe, J. Pang, and S. Seshan, "Colyseus: A distributed architecture for online multiplayer games," in *Proceedings of the 3rd Symposium on Networked Systems Design & Implementation (NSDI)*, 2006.

[125] T. Hampel, T. Bopp, and R. Hinn, "A peer-to-peer architecture for massive multiplayer online games," in *Proceedings of 5th ACM*

SIGCOMM Workshop on Network and System Support for Games (NetGames), 2006.

[126] Shun-Yun Hu, Jui-Fa Chen, and Tsu-Han Chen, “VON: A scalable peer-to-peer network for virtual environments,” *IEEE Network*, vol. 20, no. 4, pp. 22–31, 2006.

[127] L. Fan, H. Taylor, and P. Trinder, “Mediator: A design framework for P2P MMOGs,” in *Proceedings of the 6th ACM SIGCOMM Workshop on Network and System Support for Games (NetGames)*, 2007.

[128] L. Chan, J. Yong, J. Bai, B. Leong, and R. Tan, “Hydra: A massively-multiplayer peer-to-peer architecture for the game developer,” in *Proceedings of the 6th ACM SIGCOMM Workshop on Network and System Support for Games (NetGames)*, 2007.

[129] L. Fan, P. Trinder, and H. Taylor, “Design issues for peer-to-peer massively multiplayer online games,” *International Journal of Advanced Media and Communication*, vol. 4, no. 2, p. 108, 2010.

[130] A. Yahyavi and B. Kemme, “Peer-to-peer architectures for massively multiplayer online games: A survey,” *ACM Computing Surveys*, vol. 46, no. 1, 9:1–9:51, 2013.

[131] S. A. Abdulazeez, A. El Rhalibi, M. Merabti, and D. Al-Jumeily, “Survey of solutions for peer-to-peer MMOGs,” in *Proceedings of the International Conference on Computing, Networking and Communications (ICNC)*, 2015.

[132] E. Buyukkaya, M. Abdallah, and G. Simon, “A survey of peer-to-peer overlay approaches for networked virtual environments,” *Peer-to-Peer Networking and Applications*, vol. 8, no. 2, pp. 276–300, 2015.

[133] L. Liu, A. Jones, N. Antonopoulos, Z. Ding, and Y. Zhan, “Performance evaluation and simulation of peer-to-peer protocols for massively multiplayer online games,” *Multimedia Tools and Applications*, vol. 74, no. 8, pp. 2763–2780, 2015.

[134] E. Cronin, B. Filstrup, and A. Kurc, “A distributed multiplayer game server system,” University of Michigan, Tech. Rep., 2001.

[135] D. Bauer, S. Rooney, and P. Scotton, “Network infrastructure for massively distributed games,” in *Proceedings of the 1st Workshop on Network and System Support for Games (NetGames)*, 2002, pp. 36–43.

[136] J. Müller and S. Gorlatch, “GSM: A game scalability model for multiplayer real-time games,” in *Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies.*, 2005.

[137] M. Moraal, “Massive multiplayer online game architectures,” Radboud University, Nijmegen, The Netherlands, Bachelor’s Thesis, 2006.

[138] L. Yang and P. Sutinrerk, “Mirrored arbiter architecture: A network architecture for large scale multiplayer games,” in *Proceedings of the 2007 Summer Computer Simulation Conference*, ser. SCSC ’07, 2007.

[139] C. Carter, A. E. Rhalibi, M. Merabti, and A. T. Bendiab, “Hybrid client-server, peer-to-peer framework for MMOG,” in *2010 IEEE International Conference on Multimedia and Expo*, 2010.

[140] G. Wang and K. Wang, “An efficient hybrid P2P MMOG cloud architecture for dynamic load management,” in *The International Conference on Information Network 2012*, 2012.

[141] E. Carlini, L. Ricci, and M. Coppola, “Integrating centralized and peer-to-peer architectures to support interest management in massively multiplayer on-line games,” *Concurrency and Computation: Practice and Experience*, 2015.

[142] B. K. Schmidt, M. S. Lam, and J. D. Northcutt, “The interactive performance of SLIM: A stateless, thin-client architecture,” in *Proceedings of the Seventeenth ACM Symposium on Operating Systems Principles*, ser. SOSP ’99, Charleston, South Carolina, USA, 1999.

[143] C.-Y. Huang, D.-Y. Chen, C.-H. Hsu, and K.-T. Chen, “GamingAnywhere: An open-source cloud gaming testbed,” in *Proceedings of the 21st ACM international conference on Multimedia*, ACM, 2013, pp. 827–830.

[144] D. Wu, Z. Xue, and J. He, “iCloudAccess: Cost-effective streaming of video games from the cloud with low latency,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 24, no. 8, pp. 1405–1416, 2014.

[145] S. Choy, B. Wong, G. Simon, and C. Rosenberg, “A hybrid edge-cloud architecture for reducing on-demand gaming latency,” *Multimedia Systems*, vol. 20, no. 5, pp. 503–519, 2014.

[146] J. Saldana and M. Suznjevic, “QoE and latency issues in networked games,” in *Handbook of Digital Games and Entertainment Technologies*, R. Nakatsu and M. Rauterberg, Eds. Singapore: Springer Singapore, 2015, pp. 1–36.

[147] M. Schubert, “Chapter 3: Distributed game architectures,” Ludwig-Maximilians-Universität, Munich, Germany, Lecture Notes: Managing and Mining Multiplayer Online Games, 2017.

[148] F. Glinka, A. Ploss, S. Gorlatch, and J. Müller-Iden, “High-level development of multiserver online games,” *International Journal of Computer Games Technology*, 2008.

[149] M. S. Gibson and W. W. Vasconcelos, “A knowledge-based approach to multiplayer games in peer-to-peer networks,” *Knowledge and Information Systems*, 2018.

[150] P. Mildner, T. Triebel, S. Kopf, and W. Effelsberg, “Scaling online games with netconnectors: A peer-to-peer overlay for fast-paced massively multiplayer online games,” *ACM Computers in Entertainment*, vol. 15, no. 3, 3:1–3:21, 2017.

[151] Enchanted Age Studios, *Network bandwidth mathematics; peer-to-peer versus client/server*, https://web.archive.org/web/20170726175305/http://www.enchantedage.com/node/20, Accessed: 2021-11-16, 2008.

[152] J. van Dongen, *Joost’s dev blog: Relay servers*, http://joostdevblog.blogspot.com/2014/09/relay-servers.html, Accessed: 2021-11-16, 2014.

[153] E. Cronin, B. Filstrup, A. R. Kurc, and S. Jamin, “An efficient synchronization mechanism for mirrored game architectures,” in *Proceedings of the 1st Workshop on Network and System Support for Games*, ser. NetGames ’02, 2002.

[154] F. Metzger, A. Rafetseder, S. Schröder, and P. Zwickl, *The prospects of cloud gaming: Do the benefits outweigh the costs?* workingpaper, 2016.

[155] S. Perlman, *OnLive: Coming to a screen near you*, https://web.archive.org/web/20100312043136/http://blog.onlive.com/2010/03/10/onlive-coming-to-a-screen-near-you/, Accessed: 2021-11-16, 2010.

[156] S. Hollister, *Sony announces PlayStation Now, its cloud gaming service for TVs, consoles, and phones*, https://www.theverge.com/2014/1/7/5284294/sony-announces-playstation-now-cloud-gaming, Accessed: 2021-11-16, 2014.

[157] C.-F. Chang and S.-H. Ger, “Enhancing 3D Graphics on Mobile Devices by Image-Based Rendering,” in *IEEE Pacific Rim Conference on Multimedia*, 2002.

[158] F. Lamberti and A. Sanna, “A streaming-based solution for remote visualization of 3D graphics on mobile devices,” *IEEE transactions on visualization and computer graphics*, vol. 13, no. 2, pp. 247–260, 2007.

[159] R. W. N. Pazzi, A. Boukerche, and T. Huang, “Implementation, measurement, and analysis of an image-based virtual environment streaming protocol for wireless mobile devices,” *IEEE Transactions on Instrumentation and Measurement*, vol. 57, no. 9, pp. 1894–1907, 2008.

[160] D. De Winter *et al.*, “A hybrid thin-client protocol for multimedia streaming and interactive gaming applications,” in *Proceedings of the 2006 international workshop on Network and operating systems support for digital audio and video*, ACM, 2006.

[161] O.-I. Holthe, O. Mogstad, and L. A. Ronningen, “Geelix LiveGames: Remote playing of video games,” in *2009 6th IEEE Consumer Communications and Networking Conference*, IEEE, 2009, pp. 1–2.

[162] 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,” *Multimedia, IEEE Transactions on*, vol. 16, no. 2, pp. 480–495, Feb. 2014. DOI: 10.1109/TMM.2013.2291532.

[163] W. Cai, V. C. Leung, and M. Chen, “Next generation mobile cloud gaming,” in *2013 IEEE Seventh International Symposium on Service-Oriented System Engineering*, IEEE, 2013, pp. 551–560.

[164] O. Soliman, A. Rezgui, H. Soliman, and N. Manea, “Mobile cloud gaming: Issues and challenges,” in *Mobile Web Information Systems*, F. Daniel, G. A. Papadopoulos, and P. Thiran, Eds., Springer Berlin Heidelberg, 2013, pp. 121–128.

[165] K.-T. Chen, C.-Y. Huang, and C.-H. Hsu, “Cloud gaming onward: Research opportunities and outlook,” in *2014 IEEE International Conference on Multimedia and Expo Workshops (ICMEW)*, IEEE, 2014, pp. 1–4.

[166] S.-P. Chuah, C. Yuen, and N.-M. Cheung, “Cloud gaming: A green solution to massive multiplayer online games,” *IEEE Wireless Communications*, vol. 21, no. 4, pp. 78–87, 2014.

[167] W. Cai *et al.*, “The future of cloud gaming [point of view],” *Proceedings of the IEEE*, vol. 104, no. 4, pp. 687–691, 2016. DOI: 10.1109/JPROC.2016.2539418.

[168] P. Graff, X. Marchal, T. Cholez, S. Tuffin, B. Mathieu, and O. Festor, “An analysis of cloud gaming platforms behavior under different network constraints,” in *3rd International Workshop on High-Precision, Predictable, and Low-Latency Networking*, 2021. [Online]. Available: https://dl.ifip.org/db/conf/cnsm/cnsm2021/1570750103.pdf.

[169] A. Di Domenico, G. Perna, M. Trevisan, L. Vassio, and D. Giordano, “A network analysis on cloud gaming: Stadia, geforce now and psnow,” *Network*, vol. 1, no. 3, pp. 247–260, 2021. DOI: 10.3390/network1030015. [Online]. Available: https://www.mdpi.com/2673-8732/1/3/15.

[170] S. Perlman, *Beta testing at the speed of light*, https://web.archive.org/web/20100125174142/http://blog.onlive.com/2010/01/21/beta-testing-at-the-speed-of-light/, Accessed: 2021-11-16, 2010.

[171] T. Kämäräinen, M. Siekkinen, A. Ylä-Jääski, W. Zhang, and P. Hui, “A measurement study on achieving imperceptible latency in mobile cloud gaming,” in *Proceedings of the 8th ACM on Multimedia Systems Conference*, ser. MMSys’17, Taipei, Taiwan: Association for Computing Machinery, 2017, pp. 88–99. DOI: 10.1145/3083187.3083191. [Online]. Available: https://doi.org/10.1145/3083187.3083191.

[172] H.-J. Hong, D.-Y. Chen, C.-Y. Huang, K.-T. Chen, and C.-H. Hsu, “Placing virtual machines to optimize cloud gaming experience,” *IEEE Transactions on Cloud Computing*, vol. 3, no. 1, pp. 42–53, 2014.

[173] J. Wu, C. Yuen, N.-M. Cheung, J. Chen, and C. W. Chen, “Enabling adaptive high-frame-rate video streaming in mobile cloud gaming applications,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 25, no. 12, pp. 1988–2001, 2015.

[174] F. Metzger, A. Rafetseder, and C. Schwartz, “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. DOI: 10.21437/PQS.2016-5. [Online]. Available: https://www.isca-speech.org/archive_v0/PQS_2016/abstracts/4.html.

[175] M. Claypool, “Motion and scene complexity for streaming video games,” in *Proceedings of the 4th International Conference on Foundations of Digital Games*, ACM, 2009, pp. 34–41.

[176] H. Shimizu, “Measuring keyboard response delays by comparing keyboard and joystick inputs,” *Behavior Research Methods, Instruments, & Computers*, vol. 34, no. 2, pp. 250–256, 2002.

[177] R. Wimmer, A. Schmid, and F. Bockes, “On the latency of USB-connected input devices,” in *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, ACM, 2019, p. 420.

[178] G. Casiez, S. Conversy, M. Falce, S. Huot, and N. Roussel, “Looking through the eye of the mouse: A simple method for measuring end-to-end latency using an optical mouse,” in *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*, ACM, 2015, pp. 629–636.

[179] D. Luu, *Keyboard latency*. [Online]. Available: https://danluu.com/keyboard-latency/ (visited on 11/16/2021).

[180] A. Patait and E. Young, “High performance video encoding with NVIDIA GPUs,” in *2016 GPU Technology Conference (https://goo.gl/Bdjdgm)*, 2016.

[181] J. Beyer, R. Varbelow, J.-N. Antons, and S. Zander, “A method for feedback delay measurement using a low-cost arduino microcontroller: Lesson learned: Delay influenced by video bitrate and game-level,” in *Quality of Multimedia Experience (QoMEX), 2015 Seventh International Workshop on*, May 2015, pp. 1–2. DOI: 10.1109/QoMEX.2015.7148095.

[182] Digital Foundry. “Tech focus - v-sync: What is it - and should you use it?” Accessed: 2021-11-16, YouTube. (Nov. 10, 2018), [Online]. Available: https://youtu.be/seyAzw9zEoY.

[183] Battle(non)sense. “Battle(non)sense youtube channel - netcode analysis, input lag analysis, concept design, easy to grasp explanations.” Accessed: 2021-11-16, YouTube. (2019), [Online]. Available: https://www.youtube.com/user/xFPxAUTh0r1ty/videos.

[184] J. Zhao, R. S. Allison, M. Vinnikov, and S. Jennings, “Estimating the motion-to-photon latency in head mounted displays,” in *2017 IEEE Virtual Reality (VR)*, IEEE, 2017, pp. 313–314.

[185] 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*, ACM, 2015, pp. 135–144.

[186] C. Ware and R. Balakrishnan, “Reaching for objects in vr displays: Lag and frame rate,” *ACM Transactions on Computer-Human Interaction (TOCHI)*, vol. 1, no. 4, pp. 331–356, 1994.

[187] S. Schmidt, S. Zadtootaghaj, and S. Möller, “Towards the delay sensitivity of games: There is more than genres,” in *2017 Ninth International Conference on Quality of Multimedia Experience (QoMEX)*, IEEE, 2017, pp. 1–6.

[188] A. Sackl, R. Schatz, T. Hossfeld, F. Metzger, D. Lister, and R. Irmer, “QoE management made uneasy: The case of cloud gaming,” in *2016 IEEE International Conference on Communications Workshops (ICC)*, IEEE, 2016, pp. 492–497.

[189] J. Beyer, R. Varbelow, J.-N. Antons, and S. Möller, “Using electroencephalography and subjective self-assessment to measure the influence of quality variations in cloud gaming,” in *2015 Seventh International Workshop on Quality of Multimedia Experience (QoMEX)*, IEEE, 2015, pp. 1–6.

[190] M. Jarschel, D. Schlosser, S. Scheuring, and T. Hoßfeld, “Gaming in the clouds: QoE and the users’ perspective,” *Mathematical and Computer Modelling*, vol. 57, no. 11-12, pp. 2883–2894, 2013.

[191] V. Clincy and B. Wilgor, “Subjective evaluation of latency and packet loss in a cloud-based game,” in *2013 10th International Conference on Information Technology: New Generations*, IEEE, 2013, pp. 473–476.

[192] H. Iqbal, A. Khalid, and M. Shahzad, “Dissecting cloud gaming performance with decaf,” *Proc. ACM Meas. Anal. Comput. Syst.*, vol. 5, no. 3, Dec. 2021. DOI: 10.1145/3491043. [Online]. Available: https://doi.org/10.1145/3491043.

[193] S. Flinck Lindström, M. Wetterberg, and N. Carlsson, “Cloud gaming: A QoE study of fast-paced single-player and multiplayer gaming,” in *2020 IEEE/ACM 13th International Conference on Utility and Cloud Computing (UCC)*, 2020, pp. 34–45. DOI: 10.1109/UCC48980.2020.00023.

[194] Y.-T. Lee, K.-T. Chen, H.-I. Su, and C.-L. Lei, “Are all games equally cloud-gaming-friendly?: An electromyographic approach,” in *Proceedings of the 11th annual workshop on network and systems support for games*, IEEE Press, 2012, p. 3.

[195] 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 *2015 Seventh International Workshop on Quality of Multimedia Experience (QoMEX)*, IEEE, 2015, pp. 1–6.

[196] M. Claypool, A. Cockburn, and C. Gutwin, “Game input with delay: Moving target selection parameters,” in *Proceedings of the 10th ACM Multimedia Systems Conference*, ACM, 2019, pp. 25–35.

[197] D. C. Hoang, K. D. Doan, and L. T. Hoang, “Lag of legends: The effects of latency on league of legends champion abilities,” Worcester Polytechnic Institute, Tech. Rep., 2017.

[198] J. Beyer and S. Möller, “Assessing the impact of game type, display size and network delay on mobile gaming QoE,” *PIK-Praxis der Informationsverarbeitung und Kommunikation*, vol. 37, no. 4, pp. 287–295, 2014.

[199] M. Claypool and K. Claypool, “Latency can kill: Precision and deadline in online games,” in *Proceedings of the 2010 Multimedia Systems Conference*, Scottsdale, Arizona, USA, Feb. 2010.

[200] M. Bredel and M. Fidler, “A measurement study regarding quality of service and its impact on multiplayer online games,” in *2010 9th Annual Workshop on Network and Systems Support for Games*, IEEE, 2010, pp. 1–6.

[201] M. Ries, P. Svoboda, and M. Rupp, “Empirical study of subjective quality for massive multiplayer games,” in *2008 15th International Conference on Systems, Signals and Image Processing*, IEEE, 2008, pp. 181–184.

[202] M. Dick, O. Wellnitz, and L. Wolf, “Analysis of factors affecting players’ performance and perception in multiplayer games,” in *Proceedings of the Workshop on Network and System Support for Games*, Hawthorne, New York, USA, Oct. 2005.

[203] T. Beigbeder, R. Coughlan, C. Lusher, J. Plunkett, E. Agu, and M. Claypool, “The effects of loss and latency on user performance in Unreal Tournament 2003®,” in *Proceedings of 3rd ACM SIGCOMM workshop on Network and system support for games*, ACM, 2004, pp. 144–151.

[204] J. Nichols and M. Claypool, “The effects of latency on online madden nfl football,” in *Proceedings of the 14th international workshop on Network and operating systems support for digital audio and video*, ACM, 2004, pp. 146–151.

[205] K. T. Claypool and M. Claypool, “On frame rate and player performance in first person shooter games,” *Multimedia systems*, vol. 13, no. 1, pp. 3–17, 2007.

[206] R. T. Wood, M. D. Griffiths, D. Chappell, and M. N. Davies, “The structural characteristics of video games: A psycho-structural analysis,” *CyberPsychology & behavior*, vol. 7, no. 1, pp. 1–10, 2004.

[207] D. A. Laffan, J. Greaney, H. Barton, and L. K. Kaye, “The relationships between the structural video game characteristics, video game engagement and happiness among individuals who play video games,” *Computers in Human Behavior*, vol. 65, pp. 544–549, 2016.

[208] J. H. Brockmyer, C. M. Fox, K. A. Curtiss, E. McBroom, K. M. Burkhart, and J. N. Pidruzny, "The development of the game engagement questionnaire: A measure of engagement in video game-playing," *Journal of Experimental Social Psychology*, vol. 45, no. 4, pp. 624–634, 2009.

[209] H. L. O'Brien and E. G. Toms, "What is user engagement? a conceptual framework for defining user engagement with technology," *Journal of the American society for Information Science and Technology*, vol. 59, no. 6, pp. 938–955, 2008.

[210] P. Sweetser and P. Wyeth, "GameFlow: A model for evaluating player enjoyment in games," *Computers in Entertainment (CIE)*, vol. 3, no. 3, pp. 3–3, 2005.

[211] M. E. Seligman, *Flourish: A visionary new understanding of happiness and well-being*. Simon and Schuster, 2012.

[212] A. Z. Abbasi, D. H. Ting, and H. Hlavacs, "Proposing a new conceptual model predicting consumer videogame engagement triggered through playful-consumption experiences," in *International Conference on Entertainment Computing*, Springer, 2016, pp. 126–134.

[213] A. Z. Abbasi and A. B. S. A. Jamak, "Playful-consumption experience of videogame-play influences consumer video-game engagement: A conceptual model," *Global Business and Management Research*, vol. 9, no. 1s, p. 244, 2017.

[214] S. Schmidt *et al.*, "Requirement specification and possible structure for an opinion model predicting gaming QoE (G.OMG)," ITU, CH-Geneva, ITU-T Recommendation, May 2018, pp. 1–20.

[215] E. L.-C. Law, F. Brühlmann, and E. D. Mekler, "Systematic review and validation of the game experience questionnaire (GEQ) - implications for citation and reporting practice," in *Proceedings of the 2018 Annual Symposium on Computer-Human Interaction in Play*, ser. CHI PLAY '18, Melbourne, VIC, Australia: Association for Computing Machinery, 2018, pp. 257–270. DOI: 10.1145/3242671.3242683. [Online]. Available: https://doi.org/10.1145/3242671.3242683.

[216] F. Mäyrä and L. Ermi, "Fundamental components of the gameplay experience," *DIGAREC Series*, analysing immersion, no. 6, pp. 88–115, 2011.

[217] K. L. Norman, "GEQ (game engagement/experience questionnaire): A review of two papers," *Interacting with Computers*, vol. 25, no. 4, pp. 278–283, 2013.

[218] S. Choy, B. Wong, G. Simon, and C. Rosenberg, "The brewing storm in cloud gaming: A measurement study on cloud to end-user latency," in *Proceedings of the 11th annual workshop on network and systems support for games*, IEEE Press, 2012, p. 2.

[219] C. Mo, G. Zhu, Z. Wang, and W. Zhu, "Understanding gaming experience in mobile multiplayer online battle arena games," in *Proceedings of the 28th ACM SIGMM Workshop on Network and Operating Systems Support for Digital Audio and Video*, ACM, 2018, pp. 25–30.

[220] M. Hirth, F. Allendorf, F. Metzger, and C. Schwartz, "Assessing the accuracy of network estimations in the DOTA 2 game client," in *Proc. 5th ISCA/DEGA Workshop on Perceptual Quality of Systems (PQS 2016)*, 2016, pp. 20–24.

[221] M. Hirth, K. Borchert, F. Allendorf, F. Metzger, and T. Hoßfeld, "Crowd-based study of gameplay impairments and player performance in DOTA 2," in *Internet-QoE'19 Workshop*, Oct. 2019.

[222] G. Armitage, "An experimental estimation of latency sensitivity in multiplayer Quake 3," in *The 11th IEEE International Conference on Networks, 2003. ICON2003.*, IEEE, 2003, pp. 137–141.

[223] S. Zadtootaghaj, S. Schmidt, N. Barman, S. Möller, and M. G. Martini, "A classification of video games based on game characteristics linked to video coding complexity," in *2018 16th Annual Workshop on Network and Systems Support for Games (NetGames)*, IEEE, 2018, pp. 1–6.

[224] *ITU-T recommendation G.1072: Opinion model predicting gaming quality of experience for cloud gaming services*, SERIES G: TRANSMISSION SYSTEMS AND MEDIA, DIGITAL SYSTEMS AND NETWORKS, Jan. 2020. [Online]. Available: https://www.itu.int/rec/T-REC-G.1072/en.

[225] M. Fiedler, T. Hossfeld, and P. Tran-Gia, "A generic quantitative relationship between quality of experience and quality of service," *IEEE Network*, vol. 24, no. 2, pp. 36–41, 2010. DOI: 10.1109/MNET.2010.5430142.

[226] F. Metzger, N. Stulier, K. Borchert, and M. Hirth, "Relationship status: It's complicated. using APM as a QoE-qualifying tempo metric," in *2021 13th International Conference on Quality of Multimedia Experience (QoMEX)*, 2021, pp. 145–150. DOI: 10.1109/QoMEX51781.2021.9465396.

[227] S. S. Sabet, C. Griwodz, and S. Möller, "Influence of primacy, recency and peak effects on the game experience questionnaire," in *Proceedings of the 11th ACM Workshop on Immersive Mixed and Virtual Environment Systems*, ser. MMVE '19, Amherst, Massachusetts: ACM, 2019, pp. 22–27. DOI: 10.1145/3304113.3326113. [Online]. Available: http://doi.acm.org/10.1145/3304113.3326113.

[228] S. S. Sabet, S. Schmidt, S. Zadtootaghaj, C. Griwodz, and S. Möller, "Delay sensitivity classification of cloud gaming content," ser. MMVE '20, Istanbul, Turkey: Association for Computing Machinery, 2020, pp. 25–30. DOI: 10.1145/3386293.3397116. [Online]. Available: https://doi.org/10.1145/3386293.3397116.

[229] J. Iyengar and M. Thomson, *RFC 9000: QUIC: A UDP-Based Multiplexed and Secure Transport*, RFC, May 2021. [Online]. Available: https://www.rfc-editor.org/rfc/rfc9000.html.

[230] K. Lepola, "Managing network delay for browser multiplayer games," M.S. thesis, University of Helsinki, Faculty of Science, 2020. [Online]. Available: http://urn.fi/URN:NBN:fi:hulib-202011174485.

[231] S. S. Sabet, S. Schmidt, S. Zadtootaghaj, B. Naderi, C. Griwodz, and S. Möller, "A latency compensation technique based on game characteristics to mitigate the influence of delay on cloud gaming quality of experience," in *Proceedings of the 11th ACM Multimedia Systems Conference*, ser. MMSys '20, Istanbul, Turkey: Association for Computing Machinery, 2020, pp. 15–25. DOI: 10.1145/3339825.3391855. [Online]. Available: https://doi.org/10.1145/3339825.3391855.

[232] K. Lee *et al.*, "Outatime: Using speculation to enable low-latency continuous interaction for mobile cloud gaming," in *Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services*, ser. MobiSys '15, Florence, Italy, 2015, pp. 151–165. DOI: 10.1145/2742647.2742656. [Online]. Available: https://doi.org/10.1145/2742647.2742656.

[233] J. Kim, P. Knowles, J. Spjut, B. Boudaoud, and M. Mcguire, "Post-render warp with late input sampling improves aiming under high latency conditions," *Proc. ACM Comput. Graph. Interact. Tech.*, vol. 3, no. 2, Aug. 2020. DOI: 10.1145/3406187. [Online]. Available: https://doi.org/10.1145/3406187.

[234] E. Carlson, T. Fan, Z. Guan, X. Xu, and M. Claypool, "Towards usable attribute scaling for latency compensation in cloud-based games," in *Proceedings of the Workshop on Game Systems (GameSys '21)*, ser. GameSys '21, Istanbul, Turkey: Association for Computing Machinery, 2021, pp. 20–25. DOI: 10.1145/3458335.3460964. [Online]. Available: https://doi.org/10.1145/3458335.3460964.

[235] I. S. Mohammadi, M. Ghanbari, and M. R. Hashemi, "A hybrid graphics/video rate control method based on graphical assets for cloud gaming," *Journal of Real-Time Image Processing*, 2021. DOI: 10.1007/s11554-021-01159-y. [Online]. Available: https://link.springer.com/article/10.1007/s11554-021-01159-y.

[236] O. Mossad, K. Diab, I. Amer, and M. Hefeeda, "Deepgame: Efficient video encoding for cloud gaming," in New York, NY, USA: Association for Computing Machinery, 2021, pp. 1387–1395. DOI: 10.1145/3474085.3475594. [Online]. Available: https://dl.acm.org/doi/10.1145/3474085.3475594.

[237] M. Stengel, Z. Majercik, B. Boudaoud, and M. McGuire, "A distributed, decoupled system for losslessly streaming dynamic light probes to thin clients," in *Proceedings of the 12th ACM Multimedia Systems Conference*, ser. MMSys '21, Istanbul, Turkey: Association for Computing Machinery, 2021, pp. 159–172. DOI: 10.1145/3458305.3463379. [Online]. Available: https://doi.org/10.1145/3458305.3463379.

[238] G. K. Illahi, M. Siekkinen, T. Kämäräinen, and A. Ylä-Jääski, "Foveated streaming of real-time graphics," in *Proceedings of the 12th ACM Multimedia Systems Conference*, ser. MMSys '21, Istanbul, Turkey: Association for Computing Machinery, 2021, pp. 214–226. DOI: 10.1145/3458305.3463383. [Online]. Available: https://doi.org/10.1145/3458305.3463383.

[239] A. E. Alchalabi, S. Shirmohammadi, S. Mohammed, S. Stoian, and K. Vijayasuganthan, "Fair server selection in edge computing with q-value-normalized action-suppressed quadruple q-learning," *IEEE Transactions on Artificial Intelligence*, 2021. DOI: 10.1109/TAI.2021.3105087.

Florian Metzger is a postdoctoral researcher at the Chair of Communication Networks at the University of Würzburg since 2018. In 2015 he received his doctorate degree in computer science at the University of Vienna, Austria. Until 2018 he was a postdoctoral researcher at the chair of Modeling of Adaptive Systems at the University of Duisburg-Essen, Germany. His research focus are mobile networks and online and cloud gaming.

Stefan Geißler is heading the Cloud Applications and Networks Research Group at the Chair of Communication Networks at the University of Würzburg, Germany, where he also completed his PhD in 2022. His research topics include Software Defined Networking and Network Function Virtualization with a focus on performance evaluation as well as the investigation of Internet of Things technologies.

Alexej Grigorjew received his MSc degree in computer science from the University of Würzburg, Germany, in 2016. Currently he is a PhD student at the Chair of Communication Networks in Würzburg. His early research interests included SDN, NFV, and programmable data planes. Since 2018, his main focus has been low latency networking and TSN, with an emphasis on distributed latency bound analysis and network configuration.

Frank Loh is a research assistant at the University of Würzburg, Germany. Since 2017, he is a PhD student at the Chair of Communication Networks in Würzburg, Germany, with Prof. Tobias Hoßfeld. His main research focus includes QoS monitoring and QoE prediction, as well as Internet of Things technologies like LoRaWAN.

Christian Moldovan received his master's degree in computer science and the PhD degree from the University of Würzburg, Germany, in 2014 and 2021, respectively. His dissertation was on "Performance Modeling of Mobile Video Streaming". He is currently researching and developing indoor-localization technologies that rely on industrial 5G campus networks at Comnovo in Dortmund, Germany.

Michael Seufert received the Diploma and PhD degrees in computer science and the BSc degree in economathematics from the University of Würzburg, Würzburg, Germany in 2011, 2017, and 2018, respectively. He additionally passed the first state examinations for teaching mathematics and computer science in secondary schools in 2011. From 2012–2013, he was with FTW Telecommunication Research Center, Vienna, Austria, working in Research Area U "User-centered Interaction and Communication Economics", and from 2018–2019, he was with AIT Austrian Institute of Technology GmbH, Vienna, Austria, working in the Digital Insight Lab of the Center for Digital Safety and Security. Since 2019, he is the head of the "User-Centric Communication Networks" research group of the Chair of Communication Networks at the University of Würzburg, Würzburg, Germany. His research mainly focuses on Quality of Experience (QoE) of Internet applications, network management, artificial intelligence and machine learning for communication networks, as well as performance analysis and modeling of communication systems.

Tobias Hoßfeld is professor at the Chair of Communication Networks at the University of Würzburg, Germany, since 2018. He finished his PhD in 2009 and his professorial thesis (habilitation) "Modeling and Analysis of Internet Applications and Services" in 2013 at the University of Würzburg, where he was also heading the "Future Internet Applications & Overlays" research group. From 2014 to 2018, he was head of the Chair "Modeling of Adaptive Systems" at the University of Duisburg-Essen, Germany. He has published more than 100 research papers in major conferences and journals, receiving 5 best conference paper awards, 3 awards for his PhD thesis, and the Fred W. Ellersick Prize 2013 (IEEE Communications Society) for one of his articles on QoE. He is member of the advisory board of the ITC conference and the editorial board of IEEE Communications Surveys & Tutorials.