T-Gaming: A Cost-Efficient Cloud Gaming System at Scale

Hao Chen, Xu Zhang, Yiling Xu, Ju Ren, *Member, IEEE*, Jingtao Fan, Zhan Ma, *Senior Member, IEEE*, and Wenjun Zhang, *Fellow, IEEE*

Abstract—Cloud gaming (CG) system could pursue both high-quality gaming experience via intensive computing, and ultimate convenience anywhere at anytime through any energy-constrained mobile devices. Despite the abundance of efforts devoted, state-of-the-art CG systems still suffer from multiple key limitations: expensive deployment cost, high bandwidth consumption and unsatisfied quality of experience (QoE). As a result, existing works are not widely adopted in reality. This paper proposes a Transparent Gaming framework called T-Gaming that allows users to play any popular high-end desktop/console games on-the-fly over the Internet. T-Gaming utilizes the off-the-shelf consumer GPUs without resorting to the expensive proprietary GPU virtualization (vGPU) technology to reduce the deployment cost. Moreover, it enables prioritized video encoding based on the human visual feature to reduce the bandwidth consumption without noticeable visual quality degradation. Last but not least, T-Gaming adopts adaptive real-time streaming based on deep reinforcement learning (RL) to improve user's QoE. To evaluate the performance of T-Gaming, we implement and test a prototype system in the real world. Compared with the existing cloud gaming systems, T-Gaming not only reduces the expense per user by 75 percent hardware cost reduction and 14.3 percent network cost reduction, but also improves the normalized average QoE by 3.6-27.9 percent.

Index Terms—Cloud gaming, cost-efficient, HVS-based compression, adaptive real-time streaming

1 Introduction

Video gaming is a multi-billion business with revenues reaching at $109 billion approximately in 2017 and with growth of 56 percent in past five years [1]. With its tremendous popularity over the world, game consumers are expecting high-quality gaming experiences anywhere and anytime without any constraints (such as the capability of the end device and underlying network) [2]. Game content producers also prefer a unified platform to distribute their content which could be accessed by heterogeneous users on-the-fly. However, existing gaming platforms, e.g., consoles (such as Xbox, Play Station), mobile devices as well as the personal computers (PC) cannot directly offer both satisfactory gaming quality and mobility simultaneously. To this end, cloud gaming is proposed to provide such ubiquitous fashion, where games are rendered at cloud with instantaneous scene compressed and delivered as video streams to the users, while an ultra lightweight terminal captures the user input signals (such as keyboard events, mouse clicks, joystick movements, etc.) and sends back to the cloud for interaction [2], [3].

Efforts have been devoted for cloud gaming since 2000s, resulting in several well-known startups, such as G-cluster, OnLive, Gaikai, NVIDIA GeForce Now, and more recent LiquidSky, etc. [2]. Research and development activities spread out from architecture design, to compression optimization, to network planning, and to user experience study, etc. [4]. However, these startups suffer from high operational cost and low quality of experience (QoE). None of them has proven to be very successful and prevailed massively due to the following challenges.

Unaffordable Commercial Graphical Virtualization Solutions. The majority of the existing cloud gaming systems are licensing the commercial software from either Citrix or VMware, together with the professional NVIDIA GPU (e.g., GRID series), to enable multiple virtualized GPU (vGPU) instances on a physical GPU hardware (of a computing node[1]). The other alternative is enforcing the dedicated GPU pass-through to each virtualized operating system (OS) that is solely occupied by a specific user. Either way offers very limited concurrent users (particularly for GPU hungry gaming sessions), making it obviously too expensive for a large-scale cloud deployment and operation.

- *H. Chen, Y. Xu, and W. Zhang is with the Cooperative Medianet Innovation Center, Shanghai Jiao Tong University, Shanghai 200240, China. E-mail: {chenhao1210, yl.xu, zhangwenjun}@sjtu.edu.cn.*
- *X. Zhang and Z. Ma are with the School of Electronic Science and Engineering in Nanjing University, Nanjing, Jiangsu 210000, China. E-mail: {xzhang17, mazhan}@nju.edu.cn.*
- *J. Ren is with the School of Information Science and Engineering in Central South University, Changsha, Hunan 410008, China. E-mail: renju@csu.edu.cn.*
- *J. Fan is with the Department of Automation in Tsinghua University, Beijing 100091, China. E-mail: fanjingtao@tsinghua.edu.cn.*

Manuscript received 2 Sept. 2018; revised 28 May 2019; accepted 31 May 2019. Date of publication 12 June 2019; date of current version 8 Nov. 2019. (Corresponding authors: Y. Xu and Z. Ma.)
Recommended for acceptance by Alba Cristina Magalhaes Melo PhD.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TPDS.2019.2922205

1. Here, a computing node is often referred as a server/workstation rack hosted in cloud, which is typically equipped with a CPU, a GPU, memory, hardware disk as well as other components to fulfill the computing, networking, storage and etc.

Rigorous Network Bandwidth Requirements. Using the default video codec supported in GPU, existing cloud gaming systems, such as GeForce Now[2] and PlayStation Now[3], require at least 5 Mbps steady bandwidth per user to sustain the affordable-quality gaming streaming. Even if such rigorous network bandwidth requirement can be satisfied by some mobile users (e.g., downlink), the expense of uplink network of a cloud gaming service platform is unbearable for millions of concurrent gaming subscriptions (e.g., a single *King of Glory* easily attracts more than 50 million daily active users in mainland of China).

Inconsistent user QoE. Due to user's mobility and the network dynamics including the network congestion and channel fading, the available network bandwidth in the end user varies from time to time, which incurs inconsistent QoE. To this end, existing systems such as the GeForce Now, implement a very coarse adaptation by empirically adjusting the frame rate and frame size according to the network status feedback via the periodic probing [5]. In most cases, conservative strategies, such as adaptive bit rate setting for compression, are usually used to avoid the video stalling, but of course with noticeable visual quality degradation. However, existing systems fail to provide acceptable QoE for lack of effective adaptive bit rate algorithm learned from the underlying network in real-time.

In this paper, we study how to reduce the operational cost for cloud gaming providers and improve the QoE for users simultaneously. With proposed lightweight graphical container based virtualization technology, dedicated acceleration scheme and self-learned streaming strategy, we design a novel cloud gaming system called *TransparentGaming* (T-Gaming or TG). On one hand, T-Gaming system only needs to fetch the rendered audio/video frame from the GPU *framebuffer* regardless of the content itself, thus, it is *transparent* to the game providers and generally applicable to all existing games (e.g., console, PC, mobile games, etc). On the other hand, T-Gaming system is playable on any devices facilitated with the basic functionalities for network connection, audio/video decoding, image display and user interface (UI) interaction, regardless of the underlying systems. It is *transparent* to users to enable the gaming experience at anywhere, over any network, and on any device. The system can be easily implemented on top of any GPU equipped cloud computing nodes to provide ubiquitous services for both content providers and game users.

T-Gaming system offers multiple key innovations that are integrated together to facilitate the practical cloud gaming at scale, in a functional prototype. To ensure the T-Gaming service at scale, we must delicately design it considering the tradeoff between the quality, latency, and cost. Our contributions are laid in two major aspects including system implementation and algorithmic optimization in the wild. First, from the system implementation perspective, a lightweight and economical TG-SHARE is proposed to enable OS-level application virtualization that offers higher concurrency at a significant operation cost reduction, without resorting to expensive virtual machine based solutions. Second, from the algorithmic perspective, T-Gaming implements a *real-time* priority based video encoding, leveraging the human visual system (HVS) inspired importance (i.e., saliency) map, to reduce the overall bandwidth without sacrificing the visual quality. However, most existing saliency algorithms only cares about the detection accuracy, rather than joint accuracy and latency that both matter in cloud gaming scenario. Moreover, instead of using heuristic algorithms based on simplified or inaccurate models hypothesized for the network environment, we introduce an adaptive real-time streaming policy using the deep reinforcement learning tool for the first time to perform adaptive bit rate control in the cloud gaming scenario. It considers both the quality of service (QoS) and QoE states observed from the past to make a decision, resulting in great and robust performance on various actual network conditions. Note that Pensieve [6] and its variants consider the video on demand application, rather our cloud gaming with more stringent latency and buffer requirements. More detailed introduction of innovations in this work are listed as follows.

A Lightweight Virtualization Technique, Called TG-SHARE, to Run Multiple Gaming Applications Simultaneously: Different from traditional graphical virtualization solution used and discussed in existing literatures, in which a hypervisor and many operating systems are used on the hardware layer for supporting concurrent users, TG-SHARE is more cost-effective to run multiple gaming applications (referred to as gApp) simultaneously on top of the same OS for different users. In TG-SHARE, the tight connection between *user* and *application* is decoupled. Thus the hypervisor layer can be removed (i.e., resulting in license fees reduction) and just one OS (i.e., incurring virtualization overhead reduction) is required to support concurrent gaming sessions. Meanwhile, TG-SHARE supports the off-the-shelf consumer GPUs without resorting to the expensive proprietary vGPU technology (e.g., NVIDIA GRID) to further reduce the deployment cost. Integrated with the graphical container (called AppBoX) in the T-Gaming system, TG-SHARE finally can increase the density of concurrent users (and reduce the expense per user) about 5x on average by further exploiting the hardware capabilities per computing node.

Prioritized Gaming Video Encoding based on Human Visual Features. Different from other prioritized video encoding works, a dedicated acceleration API (referred to as XCODER) is implemented to compress the instantaneous gaming scene adaptively according to the human visual perception and network estimation. In XCODER, we propose a HVS based prioritized video encoding model which aims at finding optimal encoding parameters, by first extracting user's focus in real-time and then allocating different bit rate on different regions accordingly by extending the visual model proposed in [7]. The prioritized video encoding model guarantees the balanced performance of encoding speed and quality in latency[4] sensitive cloud gaming scenarios. XCODER places higher bit rate for salient regions and less bit rate elsewhere to reduce the network bandwidth consumption by

2. https://shield.nvidia.com/support/geforce-now/system-requirements/2.

3. https://support.us.playstation.com/articles/en_US/KC_Article/PlayStation-Now-and-what-do-I-need-to-get-started.

4. We use delay and latency synonymously in this paper.

14.3 percent without noticeable visual quality degradation. The available bandwidth, which is estimated via our proposed reinforcement learning policy under the dynamic network, is fed into XCODER to optimize the spatial, temporal and amplitude resolutions (STAR) [8], [9], [10] of the compressed stream to provide better visual quality.

An Adaptive Real-Time Streaming (ARS) Policy that uses Reinforcement Learning to Dynamically adjust the rate to network fluctuations. To the best of our knowledge, we are the first to utilize deep reinforcement learning (DRL) techniques [6], [11] in latency sensitive cloud gaming to enhance the real-time communication (RTC) efficiency. Note that Pensieve [6] and other follow-up works are mainly focusing on the video on demand (VoD) applications, but not latency sensitive cloud gaming. On one hand, T-Gaming system can't directly adopt existing adaptive bit rate (ABR) algorithms proposed for VoD cases. There is nearly no buffer in the client, and zero knowledge of future video segment information in the server, which requires a complete new and innovative ARS for cloud gaming. On the other hand, existing ABR schemes for real-time streams are fundamentally heuristic algorithms, which are realized based on simplified or inaccurate models hypothesized for the deployment environment or by trial-and-error approach. Instead, our ARS learns to make ABR decisions through observations of the resulting performance of past decisions. ARS outperforms the state-of-the-art algorithms with normalized average QoE improvements ranging from 3.6 to 27.9 percent by learning the underlying network characteristics in real-time.

Through meticulous system design, T-Gaming achieves comparable response delay with the state-of-the-art commercial system and outperforms the well-known open source system with response delay decreased by 60 to 70 percent. And the cost per user is less than 25 percent of that using the commercial GRID technology.

2 Related Work

As a promising paradigm for users playing games anywhere at anytime, cloud gaming has attracted interests from both academic and industrial fields. Although existing cloud gaming solutions can be classified into two categories: 3D graphics streaming [12], [13], [14] and video streaming [15], [16], [17], most cloud gaming systems (OnLive, Gaikai, LiquidSky, GamingAnywhere, Parsec, etc.) adopt the video streaming approach in practice. These systems mainly focus on system virtualization, priority based video encoding, encoding quality optimization and adaptive video streaming.

System Virtualization aims to support multiple applications on a single computing node in order to increase the density of concurrent users and reduce the operational costs. Most of the existing cloud gaming systems, such as the LiquidSky[5], NVIDIA GeForce Now, etc., directly use the proprietary NVIDIA GRID technology to provide vGPU [3] using professional GPU devices. For each vGPU instance, dedicated resources, including CUDA cores and hardware encoding acceleration, are allocated for a specific game. An alternative way is dedicated GPU pass-through by which each user will solely occupy a standalone GPU chip. Note that the off-the-shelf consumer level GPUs can be used for the GPU pass-through approach. However, it limits the number of concurrent users running on a computing node as the maximum number of the PCI-E interfaces is fixed for sizable GPUs of a standard rack server.

In general, there are two types of virtualization technologies to offer multiple vGPUs, e.g., hypervisor-based virtualization, such as Xen[6], VMware[7] and KVM[8]; and OS-level virtualization (or called container) including Linux-VServer[9] and OpenVZ[10]. The hypervisor architecture is known for better hardware independence, data isolation and security. But it incurs heavy virtualization overhead [18], [19], resulting in noticeable performance degradation. Additionally, users may pay extra fees for the license of proprietary software, such as Xen Server. As for the OS-level virtualization, all virtual instances share a single operating system kernel (usually Linux), which promises the computational performance close to the native hardware. However, almost all popular games are developed and deployed on Windows platform. Thus, we cannot directly use the Linux container, but need to develop an independent Windows OS based graphical container to avoid heavy overhead induced by the existing hypervisor solutions (e.g., Xen, KVM, etc).

Priority based Video Encoding. Region-of-Interest (RoI) based video encoding is supported by many video standards to support prioritized encoding optimization (i.e., better quality for higher priority region, and vice versa). For example, X264[11] and NVENC[12] provide application programming interface (API) to enable RoI encoding. The key issue resides in the definition or detection of RoI. In the cloud gaming scenario, there are two main schemes to define RoI: one is using the prior knowledge (of gaming engines or the human visual focus), and the other is leveraging the salient characteristics of HVS. The work in [20] is a typical example using the prior knowledge, where rendering information in the game engine is used to differentiate foreground (i.e., RoI) and background (i.e., non-RoI). But this approach requires cloud gaming system to intervene the game engine, which is prohibited by almost all proprietary video games for anti-cheating and copyright protection. Another example is deriving RoI via eye gaze tracking [7], [21] equipment. However, it is not suitable for mainstream devices (e.g., mobile phone, PC, etc) that are not equipped with dedicated gaze tracker. Instead, more studies have focused on saliency driven RoI detection. The state-of-the-art saliency detection works focus on improving saliency detection accuracy for still images [22], [23], [24], [25] or videos [26], [27], [28]. But they fail to consider the detection speed or latency, which is a critical factor for latency sensitive cloud gaming systems.

Encoding Quality Optimization: Wu et al. [16] presented AdaPtive HFR vIdeo Streaming (APHIS) scheduling

5. http://liquidsky.com.

6. https://xenproject.org/.
7. https://www.vmware.com/.
8. https://www.linux-kvm.org/page/Main_Page.
9. http://linux-vserver.org/Welcome_to_Linux-VServer.org.
10. http://openvz.org/Main Page.
11. https://www.videolan.org/developers/x264.html.
12. https://developer.nvidia.com/nvidia-video-codec-sdk.

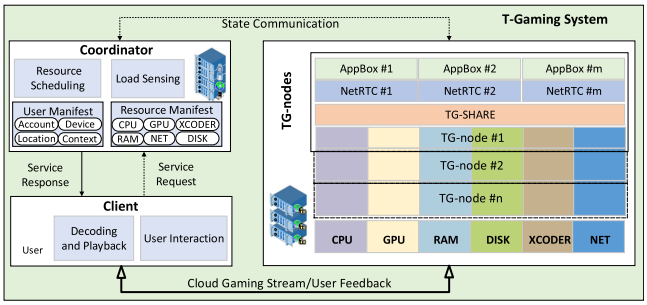


Fig. 1. Illustration of the *T-Gaming* cloud architecture: a modular view of T-Gaming Coordinator, TG-nodes and client.

framework for video frame selection to minimize the total distortion based on the network status, input video data and delay constraint. Liu et al. [20] introduced a rendering-based prioritized encoding technique to improve the perceived game quality according to network bandwidth constraints. Nan et al. [29] investigated the delay-Rate-Distortion optimization problem that the source rate between the video stream and the graphics stream is optimized to minimize the overall distortion under the bandwidth and response delay constraints. And Ahmadi et al. [30] introduced a game attention model for content adaption. However, these video bitrate control algorithms presume the bandwidth for encoding quality optimization.

Adaptive Video Streaming. Existing video streaming strategies are mainly based on the HTTP protocol [6], [31], [32], [33], [34], [35], [36]. However, the cloud gaming streaming is highly interactive and often operated based on real-time streaming protocols. To this end, Google congestion control (GCC) algorithm is proposed in [37] is to estimate available bandwidth through Kalman filtered end-to-end delay variations, and to compare the bandwidth with an adaptive threshold to dynamically throttle the sending rate. Cardwell et al. proposed another congestion-based control algorithm named bottleneck bandwidth and round-trip time (BBR) [38] that uses bottleneck bandwidth and round-trip propagation delay at TCP layer to determine available bandwidth and its sending rate. These algorithms are actually congestion control algorithms which estimate available bandwidth using instantaneously probed network QoS factors. Since they only consider physical network QoS and don't take any QoE metric as the optimal objective, these algorithms tend to overreact to a sudden network change. Recently, we have witnessed the trend to optimize the video streaming QoE via observed network and client states. Normally, these observed states are learned via a deep neural network to determine the future video bitrate, such as DRL based Pensieve [6] and QARC [39] for VoD applications.

3 System Architecture

This section details the architecture of T-Gaming system, which includes a coordinator, several TG-nodes and subscribed clients, as shown in Fig. 1. The T-Gaming is running in the cloud consisting of several coordinator nodes and hundreds of thousands of geo-distributed TG-nodes, where the coordinator nodes manage the resource manifest files and user requests, while the TG-nodes are built on common physical computing nodes. In a data center (DC), the coordinator nodes are configured beforehand and each coordinator is responsible for the management of TG-nodes in the same DC. A T-Gaming client can be either a fixed PC or mobile device, which realizes three major functions, i.e., gaming audio and video decoding that are supported by massively produced hardware chips, and user input command capturing such as keyboards events, mouses clicks and joystick movements, as well as the network connections.

As illustrated in Fig. 1, both the user and TG-node manifests are stored in their corresponding databases hosted in a coordinator node. Whenever a new user registers, its associated information (such as account, device, location, etc) is appended elastically into the user manifest database. Similarly, the resource manifest database is updated (including CPU, GPU, RAM, etc.) whenever a new TG-node joins the system. Upon a request to access T-Gaming service, he (or she) is asked to login to the system with a registered account. According to the manifest inquired by the account, the coordinator will allocate an appropriate TG-node with sufficient resource from a proper DC or edge computing farm. The node allocation is generally performed based on the load information of all running TG-nodes (via TG-SHARE) and the resource requirements of a new game request before launching the game. The TG-node/DC close to user geometrically would be assigned with higher priority and TG-node with more processing power is preferred by "extensive player" for better gaming experience. Specifically, if there exist a TG-node close to the user with enough resource for the game, the coordinator would select it to accommodate the game request. Otherwise, another node close to him/her would be activated. Moreover, the TG-node utilizes our proposed TG-SHARE solution to host the games and interact with the users remotely.

At the T-Gaming client, users equipped with even a low-performance PC or mobile device at hand (i.e., thin client) are capable to subscribe a high-quality cloud gaming service. The service can be accessed via either a browser or dedicated software application, each of which is compatible of prevailing WebRTC protocol[13]. A TG-node is a physical computing unit consisting of TG-SHARE, AppBox and NetRTC modules, mainly used for graphical application virtualization, gaming capturing and encoding, adaptive streaming as well as the behavior (user and network) learning respectively. The modularized functions of a TG-node is illustrated in Fig. 2. More details about the TG-node are unfolded as follows.

TG-SHARE. We have developed the TG-SHARE (Transparent Gaming Share), to allow the remote users to play games simultaneously over the Internet. TG-SHARE is a lightweight graphical application virtualization solution. It creates multiple isolated container box (AppBOX) to host gaming instances for different users on a single TG-node using ASTER[14]. This increases the number of concurrent users/applications about 5x on average, in comparison to those virtual machine (VM) based solutions. Such isolation is enforced to avoid cross interferences between application

13. https://webrtc.org/.
14. http://www.ibik.ru/.

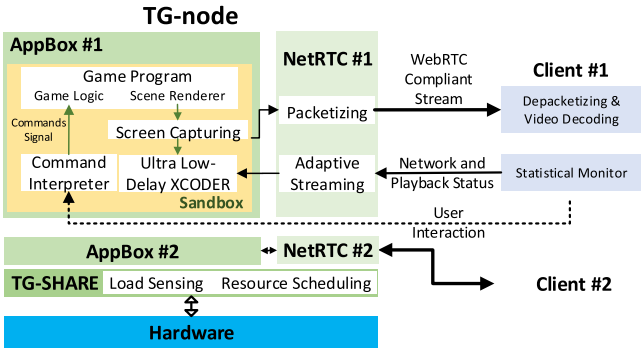


Fig. 2. Modularized functions of a TG-node.

instances to ensure the high-quality user experience as well as the user privacy.

To avoid running into resource overloads of a TG-node by multi-users, TG-SHARE senses the system activities periodically to monitor the load situations (i.e., GPU load, network utilization, I/O occupation, etc.) of the underlying hardware. Then it is also responsible for the communication of the system activities between this TG-node and the corresponding coordinator. With these information, the coordinator could determine whether to instantiate an gApp or not in this TG-node intelligently when a new access request comes. If the overload occurs in this TG-node, the coordinator would allocate another appropriate TG-node close to the user with sufficient hardware resources.

Application Box - AppBoX. The AppBox subsystem isolates independent gaming instances into gApps using the sandboxie (www.sandboxie.com). Hardware resources, such as CPU, GPU, XCODER, etc., are allocated to each gApp accordingly. For each game, its instantaneous graphical user interface (GUI) image scenes are actively fetched from the GPU frame buffer and compressed using the dedicated XCODER [40].

XCODER is a module providing three ways to perform the ultra low delay encoding. One is CPU-based encoding, which is realized in software fashion running on CPU; another is GPU-based encoding, which uses the hardware codec engine in GPU (e.g, NVENC in NVIDIA products); the third is dedicated hardware acceleration (HWA) based encoding, which is realized via a dedicated HWA chipset.

Directed by TG-SHARE, XCODER chooses the most appropriate means (i.e., CPU, GPU or dedicated HWA) adaptively to accelerate the compression according to the delay requirement (see Section 5), available resources and user context. For example, the most straightforward way is using GPU-based encoding, since gaming scenes are typically rendered using the GPU. However, the GPU resources are limited for encoding as the number of codecs supported by the consumer level GPU is usually very limited. For instance, only two NVENCs are feasible in a GTX1080 card. Alternatively, gaming frames can be encoded by means of CPU or dedicated HWA, but with extra delay, incurred by the raw data transfer from the GPU frame buffer. As proven in our former work [40], raw frames could be lightly compressed in an ultra-fast manner to reduce the data transfer latency. Thus GPU encoding will be the first choice, and dedicated HWA and CPU encoding methods would be activated when the GPU resource is exhausted.

Compression methods utilized in XCODER are compliant with the popular video coding standards, such as the H.264/AVC [41] and High-efficiency video coding (HEVC) [42], to ensure the pervasive accesses of T-Gaming. We could easily optimize the compression efficiency of XCODER by means of tuning the encoding parameters objectively (e.g., via default mean squared error (MSE) based loss) for gaming scenes. Additionally, gaming experience is very subjective, so we focus more on the perceptual quality enhancement to improve the coding efficiency and reduce the network consumption in this work.

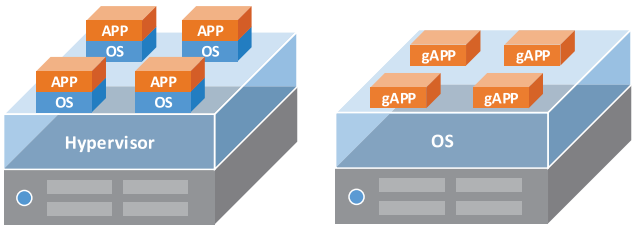
To this end, we introduce a prioritized video encoding based on gaming behavior learning, to reduce the network bandwidth consumption, but preserve nearly the same subjective quality. This is motivated by the fact that the user is extraordinary concentrated when engaging with a gaming session. Instead of applying the uniform quality scales for the entire scene, we utilize the unequal vision impacts [43] for central and peripheral areas to apply different compression factors on different areas. The details are unfold in Section 4.2. With such methodology, bit rate consumption will be reduced noticeably by 14.3 percent on average.

XCODER not only adapts compression according to the gaming behavior, but also from learned network dynamics through the NetRTC interface.

Network Real-Time Communication -NetRTC. The NetRTC encapsulates the compressed stream from XCODER into WebRTC compatible packets and delivers them to the remote users in real time. Meanwhile, user's input signals at the client, such as mouse clicks, keyboard strokes, as well as the joystick movements, are captured and sent back through the NetRTC to achieve the end-to-end interaction.

In NetRTC, we adopt Google's WebRTC as the transport protocol to ensure the ultra low-delay and real-time interaction. T-Gaming could provide transparent accesses through either a web browser or compatible software app from the heterogeneous user terminals, as most leading technology giants (e.g., Google, Microsoft, Apple, etc.) are supporting the WebRTC protocol in their software and hardware products. At the client side, the network conditions and playback status can be easily collected for T-Gaming server to perform adaptive streaming. By default, the WebRTC integrates the GCC algorithm [37] to proactively sense the network dynamics for congestion control.

Instead of using the native GCC algorithm, we propose an adaptive streaming algorithm using the DRL technology. NetRTC feeds the past network conditions and playback status into an ARS server and produces a target bitrate for adaptive video streaming. ARS represents its control policy as a neural network that maps "raw" observations (such as received bitrate, packet loss rate and round-trip time) to the bitrate decision at next time. ARS trains this neural network using A3C [44], a state-of-the-art actor-critic DRL algorithm. During the training, ARS starts with zero knowledge about the task at hand. It then gradually learns to make better decisions through reinforcement, in the form of reward signals that reflect video streaming QoE for past decisions. Through this way, ARS can remain stateless and operate only with observations which can be easily collected at client side. The details of its methodology and algorithm is elaborated in Section 4.3.



(a) traditional virtual architecture. (b) TG-SHARE virtual architecture.

Fig. 3. Comparison of traditional virtual architecture and TG-SHARE virtual architecture. Noting the gray box represents hardware resources.

To summarize, we have made quite a lot of progresses on top of our former works [40], [45], including the optimization of system implementation, the improvement of prioritized gaming video encoding algorithm, and the introduction of learned ARS to facilitate the practical cloud gaming at scale. For system implementation, it is upgraded by introducing the Coordinator subsystem to adjust the control flow to better manage the system and afford user context-aware services (see Section 4). An effective communication module of resource utilization state is newly integrated in T-Gaming system, enabling the coordinator to allocate a proper gaming instance for the subscriber. Additionally, better graphical virtualization solution with the sandboxie is introduced to better perform the application isolation, compared to our former work [40] (see Section 4.1). For prioritized gaming video encoding, we detect the saliency map using real-time saliency detection, instead of using the mouse or joystick movements in our former paper [40]. Saliency map and network estimation are jointly used to do the accurate bitrate allocation, rather the heuristic method provided in [40] (see Section 4.2). For adaptive real-time streaming, DRL based ARS is newly introduced in this work, while our former work [40] uses GCC algorithm directly. As shown later, ARS outperforms GCC with noticeable QoE improvement (see Section 4.3).

4 Design

In this section, we describe the design of T-Gaming, a system that provides ubiquitous services to both game providers and game users. We start by introducing a cost-effective graphical virtualization solution, without resorting to professional and expensive vGPU technologies. Then we describe the HVS-based prioritized video encoding, with which bandwidth consumption can be reduced significantly without subjective quality degradation. Finally, we explain DRL-based ARS mechanism, which observes and learns past streaming experiences to produce a proper target bitrate at next time for rate adaption.

4.1 Cost-Effective Graphical Virtualization

Generally in existing vGPU solutions, a hypervisor is built on hardware layer to virtualizate hardware resources for upper OS layers as shown in Fig. 3a. By running Apps on different OSs, which are managed by the hypervisor, it can isolate concurrent users. Such vGPU solution is facing several challenges: a) less concurrency improvement at the cost of heavy virtualization overhead introduced by the hypervisor; b) high license fee of commercial software (e.g., WMware and Xen) and expensive professional GPU (e.g., GRID cards).

Instead, we have developed our own resource sharing scheme - TG-SHARE, shown in Fig. 3b, to allow the remote users to play games in parallel. One one hand, TG-SHARE utilizes off-the-shelf consumer GPUs by GPU pass-through, without resorting to expensive proprietary virtualization technology, to enable a virtual machine (VM) to monopolize a GPU card. On the other hand, inspired by the fact that a user could manage multiple GUI-based applications concurrently (such as *Word*, *Photoshop*, *Game* etc.) on a single desktop system (or a VM), TG-SHARE is developed to decouple the tight connections between *user* and *application*. By introducing ASTER technologies, it is allowed to create a few GUI workplaces on top of a single node, where each user can manage its individual application sessions independently. To avoid cross interference in game programs, each application is isolated into a AppBoX using sandboxie. In AppBox, instantaneous images of an application GUI are grabbed and encoded for remote interaction. Within a VM, TG-SHARE uses the OS functions to monitor the system resources (CPU, GPU, RAM and so on) and perform the appropriate management. Resource utilization is also reported and used for the new gaming session and user coordination, to guarantee the load balancing.

Herein, a key issue is how to ensure acceptable gaming experiences with adequate hardware resources for concurrent users in the same TG-node. Inside TG-SHARE, we implement a socket communication module, which is used for reporting hardware usage to its corresponding coordinator node. The coordinator examines each load status of various hardware resources in this TG-node. If the node is underused, a gApp would be still instantiated in this node when a new request accesses. Otherwise, the coordinator would allocate a new proper TG-node, which is near to the user with sufficient hardware resources meeting the gaming requirements. It is worth to mention that the game context (such as the minimum requirement of hardware for the game), user context (such as "extensive player" and "whatever player") and device context (such as PC and mobile phone) are taken into consideration before allocating a new node, because the hardware requirements are varying for these contexts.

The differences between TG-SHARE virtualized applications and conventional non-virtualized applications can be unfolded into two aspects: 1) the former supports multiple concurrent gAPPs (for different games) running on a same physical GPU by different users, but the latter not; 2) the former supports data and control isolation between concurrent users, while the latter not.

Compared to our former work [40], we use ASTER technologies in graphical virtualization solution (i.e., TG-SHARE) in this work with the sandboxie in AppBoX to better perform application isolation. And an effective communication module of resource occupancy state is newly integrated in T-Gaming system, offering the systematic load balancing enforcement.

4.2 HVS-Based Prioritized Video Encoding

Considering the characteristics of HVS that a user would focus more his/her attention on the salient area (such as the "battle region") within the central vision [7], we introduce

prioritized encoding techniques to place higher quantization on non-salient regions to reduce the network bandwidth consumption without noticeable visual quality degradation. To achieve prioritized video encoding in latency sensitive cloud gaming, we have to guarantee the balanced performance of encoding speed and quality. Towards this, real-time saliency driven bit rate allocation is highly desired, where available bandwidth is sensed promptly from underlying network.

To tackle these challenges, we propose a HVS-based prioritized video encoding model which aims at finding optimal distribution of quantization parameter (QP) across the macroblocks in a frame, by first extracting user's focus in real time and then allocating different bit rate on different regions accordingly by extending the visual model proposed in [7].

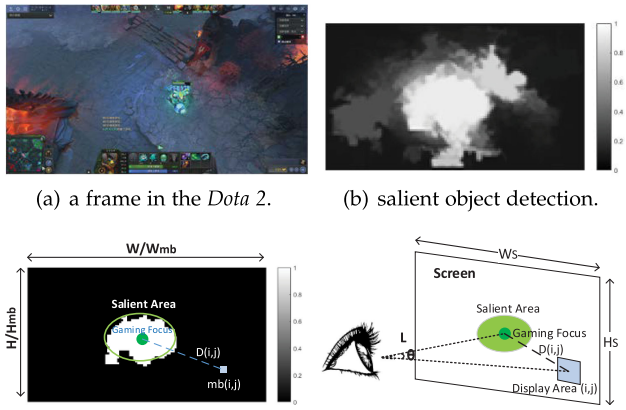
Compared with [40], we detect the saliency map as the representation of gaming focus by using image processing techniques (i.e., modified MinBarrier algorithm to get gaming focus), instead of using the mouse or joystick movements in [40]. After obtaining the saliency map, the bit rate allocation strategy is proposed by extending the visual perceptual model proposed in [7], not following the heuristic method provided in [40].

Extraction of Gaming Focus. As the latency is a key factor for cloud gaming, T-Gaming should guarantee that the focus extraction must be real-time. However, we are not able to obtain the focus directly through the render information from the game engine due to the transparency requirement of T-Gaming. On the other hand, existing gaze tracking [21] method may not be applicable to the cloud gaming scenario (usually with high motion pattern), as the gaze information may expire when received at server due to the high processing and transmission latency.

To solve this problem, we have extended the saliency detection MinBarrier [46] algorithm from pixel-level to macroblock (MB) level, for the accurate gaming focus extraction in real-time. Macroblock level processing is motivated by the fact that we focus on image regions while not pixels. Such MB-level MinBarrier detects the salient region by calculating the minimum barrier distance (MBD) transform in raster scanning order. Since it is performed via only image processing techniques, it is unnecessary for the player to have eye gaze devices in this work.

T-Gaming first downscales the frame of gaming videos with resolution $W \times H$ by a factor of $scale = W_{MB}$. W_{MB} is the MB width in pixels, i.e., 16 [41]. Based on the saliency map produced by MinBarrier, the salient object is extracted using a proper threshold Th. Then we obtain the gaming focus center (i_o, j_o) in MB domain by further calculating the centroid of the salient object in the map. Since human eyes only focus on one point at a time when engaging with a gaming session, there's no need to consider the multi-center cases.

Fig. 4 shows how to extract the gaming focus from a video frame. A frame with the resolution at 1920×1080 of *Dota 2* is illustrated in Fig. 4a, and it is resized by $scale = 16$ before fed into MinBarrier. The saliency map with size 120×68 is then generated, as shown in Fig. 4b. Using the threshold $Th = 0.25$, we get the salient object represented by white regions, and the center of gaming focus marked by a green circle in Fig. 4c.



(a) a frame in the *Dota 2*. (b) salient object detection.

(c) gaming focus extraction. (d) QP setting.

Fig. 4. Saliency detection.

Bit Rate Allocation. After extracting the gaming focus, we extend the visual model proposed in [7] to determine the quantization step (or equivalent QP) with respect to the MB location measured using the visual angle θ eccentrically from the center of the salient object. First, the distance between a MB located at (i, j) and the gaming focus center (i_o, j_o) is calculated by,

$$d(i,j) = \sqrt{(i - i_o)^2 + (j - j_o)^2}. \tag{1}$$

As illustrated in Fig. 4d, the d istance between the human eye and the screen is denoted as L, the screen size is $W_S \times H_S$, and the distance from the display area (i.e., the region mapped from the MB at (i, j)) to the gaming focus center on the screen is $D(i, j)$. Thus, $D(i, j)$ can be easily mapped using $d(i, j)$, i.e.,

$$D(i,j) = d(i,j) \cdot \frac{W_S}{W}. \tag{2}$$

Therefore, the degree of eccentricity (θ, i.e., visual angle) of the area from the center of the retina can be represented by

$$\theta(i,j) = \frac{180 \cdot \arctan(D(i,j)/L)}{\pi}. \tag{3}$$

Assuming the quantization step of the gaming focus is q_o, the quantization step for a MB located at $\theta(i, j)$ is derived by

$$q(i,j) = q_o \frac{\hat{q}(\theta(i,j))}{\hat{q}(0)}, \tag{4}$$

where

$$\hat{q}(\theta) = \frac{1}{c_3\sqrt{2\pi}} \cdot \mathrm{e}^{-\frac{|(c_2\theta)^{c_1}|}{2c_3^2}} + c_4, \tag{5}$$

with $c_i (i \in [1, 4])$ as model parameters that are either fixed or predicted using the content features [7].

On the other hand, a quantization-bitrate model is given in [8], [10], as

$$R(i,j) = \alpha \cdot q(i,j)^{-\beta}, \tag{6}$$

where α and β are content-dependent parameters, which can be predicted by weighted combination of extracted content features. These features include the residual (error) signal (such as the frame difference), the motion fields (such as the motion vector magnitude, and the motion direction activity), as well as the video frame contrast. We directly use the linear predictor proposed in [8] to determine the α and β parameter. As the encoded video bit rate should not exceed the available bandwidth B_T, which can be estimated by *Adaptive Real-time Streaming* (Section 4.3), we produce the target bit rate R from a constraint function $\sigma(B_T)$. Recalling the quantization-bitrate model in Eq. (6), we could derive the following equation:

$$R = \frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N} R(i,j) = \frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N} \frac{\alpha}{q(i,j)^{\beta}} = \sigma(B_T), \tag{7}$$

where M and N are the numbers of MB in y-axis and x-axis in a frame, and α and β are model parameters dependent on video content. The $\sigma(B_T)$ function performs two operations: scaling down the B_T to reserve room for the retransmission of lost packets and packet header overheads, and then truncating the result by a min-max thresholding for each game, to produce the final target bit rate R. Thus, with estimated available bandwidth and extracted gaming focus, we get the optimal distribution of quantization steps for all MBs in a frame by combining Eq. (1) to Eq. (7),

$$q(i,j) = \left(\frac{\alpha}{\sigma(B_T)N}\sum_{i=1}^{M}\sum_{j=1}^{N}\left(\frac{\hat{q}(0)}{\hat{q}(d(i,j))}\right)^{\beta}\right)^{\frac{1}{\beta}} \times \frac{\frac{1}{c_3\sqrt{2\pi}}\cdot \mathrm{e}^{-\left|(c_2\frac{\pi \arctan(\frac{D(i,j)}{L})}{180})^{c_1}\right|/(2c_3^2)} + c_4}{\frac{1}{c_3\sqrt{2\pi}} + c_4}. \tag{8}$$

Finally, we convert the quantization step $q(i,j)$ to the quantization parameter $QP(i,j)$, which is commonly used in practical encoder implementations. The relation between $QP(i,j)$ and $q(i,j)$ specified in H.264/AVC is

$$QP(i,j) = round[6 \times \log(q(i,j)) + 4]. \tag{9}$$

To evaluate the subjective quality loss using our prioritized encoding model, we propose a weighted peak signal-to-noise ratio (WPSNR) metric to model the impact of unequal quantization on different regions in a frame on user's perceptual quality. More specifically, we connect the $q(\theta)$ with our previous quality-quantization $Q(q)$ [9] model to have,

$$Q(\theta(i,j)) = \frac{1 - e^{\alpha_q(\frac{1}{c_3\sqrt{2\pi}}+c_4)/(\frac{1}{c_3\sqrt{2\pi}}\mathrm{e}^{-\frac{|(c_2\theta(i,j))^{c_1}|}{2c_3^2}}+c_4)}}{1 - e^{-\alpha_q}}. \tag{10}$$

Here, $Q(\theta(i,j))$ can be regarded as the visual weights to calculate weighted mean square error (WMSE) by

$$\mathrm{W}MSE(i,j) = \frac{1}{MN}\sum_{i=0}^{M-1}\sum_{j=0}^{N-1} Q(\theta(i,j))(f'(i,j) - f(i,j))^2, \tag{11}$$

where the f and f' represent the original and reconstructed pixel values in a video frame respectively. Then the WPSNR is derived by

$$\mathrm{W}PSNR(i,j) = 10log_{10}\frac{255^2}{WMSE}. \tag{12}$$

In Section 5, we would show that T-Gaming could save appreciable bitrates using HVS-based prioritized video encoding model compared to standard encoding method without noticeable perceptual quality degradation.

4.3 Adaptive Real-Time Streaming (ARS)

Another severe challenge for cloud gaming is that its QoS and QoE is most susceptible to network dynamics. To guarantee user's QoE, T-Gaming integrates ARS module to enable adaptive streaming toward these network dynamics. To the best of our knowledge, ARS generates a DRL-based adaptive bit rate (ABR) algorithm and applies it to latency sensitive gaming video streaming scenarios for the first time.

T-Gaming chooses UDP based WebRTC as its transport protocol. In default, GCC [37] and BBR [38] are used in WebRTC for network congestion control. We replace them with our ARS for adaptation. In the cloud gaming service, video encoding and streaming must be conducted simultaneously in real time, rather than preparing video chunks with different bitrates in advance like the VoD case. The server is not aware of the information of a future video fragment in this case. Meanwhile, packet loss rate is introduced as a factor affecting QoS and QoE due to the packet loss characteristic of RTP stream in WebRTC session. Furthermore, constrained by the low-latency interaction requirement for cloud gaming, T-Gaming has to maintain a short playback buffer at the client side. To this end, ARS plays a more important role in adapting to dynamic networks in order to ensure acceptable user's QoE with smooth video playback and acceptable video quality.

On the other hand, existing bitrate adaption algorithms for WebRTC protocol are usually based on underlayer congestion control algorithms, such as GCC and BBR. Actually, these congestion control algorithms require to estimate the available bandwidth by considering QoS parameters including the inter-packet round-trip time (RTT) difference and packet loss rate. This is the main drawback as they focus on congestion-control-based QoS factors while not considering user's QoE. Our ARS module is designed to achieve real-time bitrate adaption to network changes based on a QoE metric without awareness of further video segment data. We start by explaining the training methodology and then its implementation details.

Training Methodology. Similar to Pensieve [6], the first step of ARS is to generate an ABR algorithm using DRL. To achieve this, ARS runs a training phase in which the learning agent explores a real-time video streaming environment.

To accelerate the training, ARS trains ABR algorithms in a simple simulation environment that faithfully models the dynamics of real-time video streaming. To build this simulation environment, we first create a corpus of network traces by combining several public datasets [47], [48], [49], [50], which contains various network patterns including stationary ([47]) and non-stationary network conditions ([48], [49], [50]). Note that we do not train models individually for each network pattern due to practical implementation consideration. If the model was trained in each individual network pattern, there will be a large number of dedicated models for individual cases. It will introduce huge overhead when we deploy the T-Gaming system. For example, T-Gaming system must perform network pattern detection over various and massive patterns in the real world, which also introduces extra overhead for the system. Once an error or failure occurs in the detection, the performance of adaptive bit rate algorithm may degrade sharply. Furthermore, the pre-trained models would not be utilized efficiently, since a user generally lives in the environment only with limited network patterns, resulting in a waste of the majority of pre-trained models. Thus, ARS trains a general model with all available network traces. In Section 5, we will demonstrate that a generalized model could give a decent performance.

Unlike Pensieve, T-Gaming uses a more accurate simulator in the group of picture (GOP) level to train ARS as it needs quick response for the real-time gaming. This configuration benefits for fast action of T-Gaming to perform bitrate adaption when network fluctuation occurs. ARS simulator does not maintain an internal representation of the client's playback buffer. Instead, the playback buffer is maintained within a GOP. Another reason to choose the GOP level for the adaption is that the playback cannot be restored until the client receives a new instantaneous decoding refresh (IDR) frame. Once a packet of the IDR frame in a new GOP received, the playback buffer will be emptied to reduce the interaction latency. This makes great sense for cloud gaming services because users are very sensitive to the interaction latency. And the quicker the playback resumes, the higher the QoE can be achieved.

During the GOP-level adaption, the simulator computes packet loss rate and average RTT for each frame based on the frame sizes and the input network throughput traces. When the packet loss occurs, the client would send the Negative Acknowledgment (NACK)[15] to the simulator and the simulator would perform retransmission of lost packets consequently. The expectation probability that retransmitted packets can successfully be received is set by the simulator to mimic the network congestion. Moreover, the ARS simulator would try to retransmit these lost packets repeatedly in a GOP duration. If the time expires, the retransmitted packets are useless due to expiration for the real-time decoding constraint. Only if the received frames can be decoded, the simulator puts the newest frame into playback buffer to decrease the system response delay. The simulator carefully keeps track of freezing events which happens if the buffer occupancy is less than the duration of a GOP.

15. WebRTC 1.0: Real-time communication between browsers, https://www.w3.org/TR/2018/CR-webrtc-20180927/.

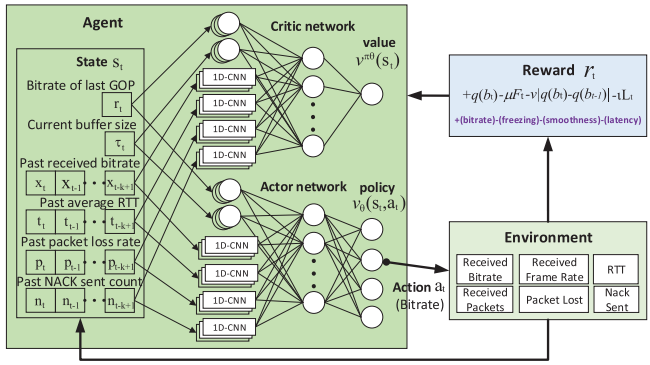


Fig. 5. The Actor-Critic algorithm that ARS uses to generate ABR policies.

During the model training, the simulator passes several state observations to the DRL agents for processing: the last selected GOP bitrate, the current buffer occupancy, the average RTT in the past GOP, the average throughput of the past GOP, the packet loss rate and the NACK sent count. The last selected GOP bitrate is the bitrate level that the simulator selected in the last round. The current buffer occupancy means how many frames received during a GOP. The average RTT consists of the propagation time, queuing time and processing time, while the processing time is ignored due to its small value compared to the formers. The packet loss rate is counted excluding the successful retransmitted packets using the NACK scheme. Using this GOP-level simulator, ARS can "experience" gaming video streaming 60 times faster compared to that in real experiment.

Now, we describe the training algorithm of ARS. As shown in Fig. 5, ARS training algorithm uses the A3C [44], the state-of-the-art actor-critic method which involves training two neural networks. After the streaming of each GOP t, ARS learning agents take state inputs $s_t = (\overrightarrow{r_t}, \overrightarrow{\tau_t}, \overrightarrow{x_t}, \overrightarrow{t_t}, \overrightarrow{p_t}, \overrightarrow{n_t})$ to its neural networks. The meaning of these symbols could be found in Table 1. The reward is defined and detailed in Section 5.

Upon receiving s_t, ARS DRL agent needs to take an action to determine the bitrate selected for the next GOP. The agent selects actions based on a policy, and neural network (NN) [51] is utilized to represent the policy with a manageable policy parameters, θ. Using the θ, we can represent the policy as $\pi_\theta(s_t, a_t)$. After applying each action, a reward r_t is provided by the simulated environment to the learning agent for that GOP. The primary goal of the ARS DRL agents is to maximize the expected cumulative (discounted) reward that it receives from the environment. Thus, we set the reward to reflect the performance of each

TABLE 1
Adopted Notation for the States input to ARS Neural Networks

Notation	Meaning
$\overrightarrow{r_t}$	Last selected bitrate.
$\overrightarrow{\tau_t}$	Playback buffer occupancy.
$\overrightarrow{x_t}$	Last received bitrate.
$\overrightarrow{t_t}$	Average RTT.
$\overrightarrow{p_t}$	Packet loss rate.
$\overrightarrow{n_t}$	NACK sent count.

GOP streaming by means of QoE modeling, which is illustrated in Section 5.

The actor-critic algorithm used by ARS to train its policy is a policy gradient method [52]. The key idea in policy gradient methods is to estimate the gradient of the expected total reward by observing the trajectories of executions obtained by following the policy. The detailed derivation can be found in [52].

ARS performs multiple (by default 16) learning agents in parallel to further speed up the training phase. Each learning agent is configured to experience a different set of network traces. A central agent aggregates the $(state, action, reward)$ tuples received from these agents continually to build a single ABR algorithm model. After computing a gradient and perform a gradient descent step using the tuples, the central agent updates the actor network and feed back the new model to the agents.

Implementation in T-Gaming System. To generate ABR algorithms, ARS passes the past 8 received bitrate measurements to a 1D convolution layer (CNN) with 128 filters, each with the size equal to 4 and the stride equal to 1. As shown in Fig. 5, the average RTT, the past packet loss rate and the count of past sent NACK are also passed to another 1D-CNN with the same shape respectively. Results from these layers are then aggregated with other inputs in a hidden layer that uses 128 neurons to apply the softmax function. The critic network uses the same NN structure, but its final output is a linear neuron (with no activation function). During the training, we use a discount factor $\gamma = 0.99$, and the learning rates for the actor and critic are configured to be 10^{-4} and 10^{-3} respectively. Additionally, the entropy factor β is controlled to decay from 1 to 0.1 over 10^5 iterations. We implemented this architecture using TensorFlow [53]. For compatibility, we leverage the TFLearn deep learning API[16] to declare the neural network during both the training and the system implementation.

Once ARS has generated the ABR algorithm using its simulator, it would apply the model's rules to T-Gaming sessions. To do this, the ARS module is deployed and run in T-Gaming server. By collecting the statistical states including *FrameRateReceived*, *bitsReceivedPerSecond*, *googRtt*, *packetsLost*, *packetsReceivedPerSecond*, and *NacksSent* of each WebRTC peer connection, it's easy to obtain the state s_t by simple transformations and send them back to the ARS server. Once these observations arrive at server, ARS feeds them into its actor network and produces a bitrate level. The bitrate is then truncated by a minimum bitrate threshold and a maximum bitrate threshold, which are configured differently for different games. Finally, the result is used as the target bitrate for adaptive video streaming, as described in Section 4.2. By this way, ARS remains stateless and benefits from observations that can be easily collected at WebRTC compatible receivers.

The main difference between ARS and Pensieve is the application scenario. Compared with Pensieve for VoD situation, ARS is proposed for leveraging various states observed in WebRTC streams to adaptively adjust output bitrate of T-Gaming in real time to achieve better user QoE, without future video segment information especially segment size.

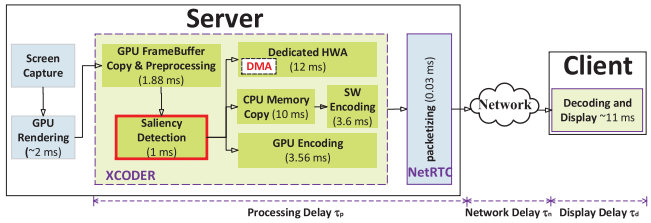


Fig. 6. Delay distribution regard to response delay in T-Gaming system.

The learning states of ARS are different from that of Pensieve and they must be collected in GOP level to pertain the real-time property. In a word, ARS is the first DRL-based algorithm that achieve bitrate adaption for cloud gaming with high performance.

5 Evaluation

To well understand the performance of T-Gaming, we conducted several experiments with real-life setup by collecting actual gaming data. Towards this goal, we built a prototype system in the center of China. The TG-node in the prototype system included 10 TG-nodes where each is equipped with 32G RAM, 512G SSD, 8G GTX1080 GPU, Intel 8-core i7 CPUs and a self-designed hardware chip offering 16-channel H.264/AVC and HEVC compression acceleration simultaneously. The system was hosted by a family type ISP offering constant 200 Mbps uplink.

The prototype system can be accessed via a unified URL[17]. Users can play the traditional high-end PC/mobile games through either a web browser or mobile app. To evaluate the performance of T-Gaming, we used a Linux-based Traffic Control (TC) tool to simulate various network conditions. The network bandwidth traces were prepared according to the corpus of test traces randomly selected from public network throughput datasets (detailed in Section 5.3).

5.1 Latency-Cost Efficient System

Since cloud gaming is more interactive than traditional streaming applications, the latency and the system cost become two key factors to the performance of a practical system. We start by evaluating the system latency of T-Gaming compared with other state-of-the-art systems, and then compare TG-SHARE with GRID based vGPU solution on virtualization cost per user.

Using the definition in [15], the response delay (RD) introduced by T-Gaming is divided into the processing delay (PD) in the XCODER, the playout delay (OD) in the client and the network delay (ND) during the packet delivery, as shown in Fig. 6.

Different codec implementations in XCODER can lead to different processing delays. Candidates include GPU-based, CPU-Software-based and HWA-based implementations (Section 3). We measured the processing delay τ_p for different implementations from the GPU *framebuffer* output to a 1080P@60 FPS (frames per second) compressed stream:

1) GPU-based implementation: For each frame, the GPU *framebuffer* copy took about 1.88 ms, followed

16. http://tflearn.org/.

17. http://www.anygame.info.

by the direct GPU encoding about 3.56 ms, resulting in $\tau_p^{GPU} = 5.44$ ms in total.

2) CPU-Software-based implementation: For this scheme, raw frames had to be transferred from the GPU *framebuffer* to the CPU memory first (consuming about 10 ms), and then encoded by softwares (whose delay depends on how many slices can be encoded in parallel). Our experiments show that 8-slice per frame can be encoded for about 3.6 ms, resulting in $\tau_p^{CPU} = 15.48$ ms.
3) HWA-based implementation: Even if the dedicated memory access (DMA) is used, τ_p^{HWA} is about 13.88 ms in total.

Moreover, we can reduce the memory transfer latency via the light compression inside GPU, such as the fast JPEG encoding technique [40]. Ideally, it only needs about 10 percent time cost compared with the existing raw frame exchange scheme, by incorporating the fast JPEG decoder into the HWA and CPU-based software. And we expect $\tau_p^{HWA} = 7.8$ ms and $\tau_p^{CPU} = 6.88$ ms then after incorporating the lightweight compression. In addition, only extra 1 ms delay per frame was introduced for saliency detection if prioritized video encoding based on HVS is applied.

Regarding the OD, it took about 11 ms to display a new received frame using a web browser, larger than that using a client application due to the refreshing and rendering restriction of the browser. Note that the WebRTC protocol only supports software video decoding currently, which enlarge the OD in T-Gaming system. Our experiments show that the OD can be reduced to about 6 ms in native T-Gaming clients by enabling hardware decoding and instant rendering for video streams, about 45.5 percent less than that using a web browser.

For ND, we had performed the delay measurements repeatedly between the T-Gaming server location and other three cities, at a frequency of every half an hour from the 9:00AM to the 9:00PM (from May 1 to July 31 in 2018). The distances between the T-Gaming server location (in the central China) and the three cities (in the fast east, fast south, and fast north of China) are about the same, approximately 1150 miles, to evaluate the ND with extreme situations. The averaged delay between T-Gaming and northern city was about 53.5 ms, while 42 ms for connections from eastern city, and 44 ms for accesses from southern city. For the increase of user numbers, we would deploy more game servers distributed geographically and allocate to users one with close proximity.

Moreover, we compared T-Gaming with GamingAnywhere (GA) [15], Parsec with software (SW) decoding, and Parsec with hardware (HW) decoding in terms of RD with only PD and OD. As we expect the similar ND if we setup the identical network environment for T-Gaming, GA and Parsec. To measure the RD, we calculated the latency from the moment that client activates a key event to the moment that the corresponding screen change is detected in the client. As shown in Table 2, we can observe that the more FPS, the less RD. This is because that a new frame can be captured faster with higher frame rate. The performance of T-Gaming and Parsec with software decoding is almost the same, both of them take about 45 ms to response the key event at 30 FPS, and 35 ms at 60 FPS. Using hardware decoding, Parsec can achieve less RD with 41 ms at 30 FPS and 31 ms for at 60 FPS, which is still comparable with our T-Gaming. Maybe GA is out of careful optimization as a open source cloud gaming system, both T-Gaming and Parsec outperform GA significantly, which only account for 30 to 40 percent of GA's RD. Though Parsec exhibits the similar response delay as our T-Gaming, it only supports Windows 8.1+ server at present, making it not supported for mobile games. And it requires an additional dedicated application client installation on each platform, which could not offer the transparent accessibility anywhere at anytime for content providers and game users.

TABLE 2
Response Delays (in ms) Comparison between T-Gaming, Parsec with SW Decoding as well as HW Decoding, and GamingAnywhere

Systems	30 FPS	60 FPS
T-Gaming	45.0	34.4
Parsec (SW dec)	47.4	33.8
Parsec (HW dec)	40.9	30.8
GA	115.9	115.8

As for virtualization cost, We compared TG-SHARE using the GTX1080 with GRID-based vGPU using the Telsa K2, which is adopted by the majority of existing cloud gaming vendors. We ran the top ten games both in China and globally on both platforms. TG-SHARE equipped with GTX1080 could support 6 users simultaneously, while only 4 users on GRID K2 platform. But, GRID K2 GPU is about 2x more expensive than the GTX 1080. Therefore, the cost of TG-SHARE per user is less than 25 percent of the per user cost by using the GRID K2.

5.2 Network Bandwidth Consumption

As we all know, network bandwidth consumption mainly depends on the gaming content characteristics and codec performance. We ran various genres of the 3D games, including massively multiplayer online role-playing games (MMORPG), hack and slash (H&S) games, and racing video games on the T-Gaming system. In this paper, we selected the *World of Warcraft* (WOW) as the typical representative of M-MORPG, *Devil May Cry* (DMC) as the typical representative of H&S games, and *Need For Speed* (NFS) as the typical representative of racing video games. Through these games, we evaluate the performance of T-Gaming system on network bandwidth consumption. Gaming scene encoding was performed using the GPU codec directly. For each game, five video sequences with 1080P resolution at 30 FPS were captured for testing. For a video sequence in regular gaming scenarios, 10 Mbps bandwidth is required for H.264/AVC compatible stream, but 5 Mbps is sufficient for HEVC compatible stream to offer high quality gaming experience [40]. This is because HEVC (e.g., version 1 published in 2013) is the latest video compression standard, which provides approximately 50 percent bit-rate savings for the same perceptual quality relative to the H.264/AVC standard (version 1 published in 2003) [42]. Thus, HEVC also offers better quality than H.264/AVC when having the same bit rate.

We performed video encoding for aforementioned video sequences using prioritized encoding model to further

TABLE 3
The Parameters of HVS-based Prioritized Video Encoding Model (8)

Name	Para.	Value
screen width (cm)	W_S	37.65
screen height (cm)	W_H	21.17
Distance between eyes & screen (cm)	D	40
Model parameter c_1 in [7]	c_1	3
Model parameter c_2	c_2	0.016
Model parameter c_3	c_3	-0.51
Model parameter c_4	c_4	1

reduce bandwidth consumption. In the simulation, we encode the video sequences for the game DMC, WOW and NFS using H.264/AVC codec with constant QP set {22,27,32,37,42} at 30 FPS. The model parameters are shown in Table 3 and the QP of gaming focus is set within the QP set, where c1 reflects the decay speed of the quality perception, c2 is correlated with the quality degradation factors, c3 is generally content-dependent and c4 indicates the visual angle goes to the max value. c1, c2, and c4 are used with fixed values in this paper (see Table 3).

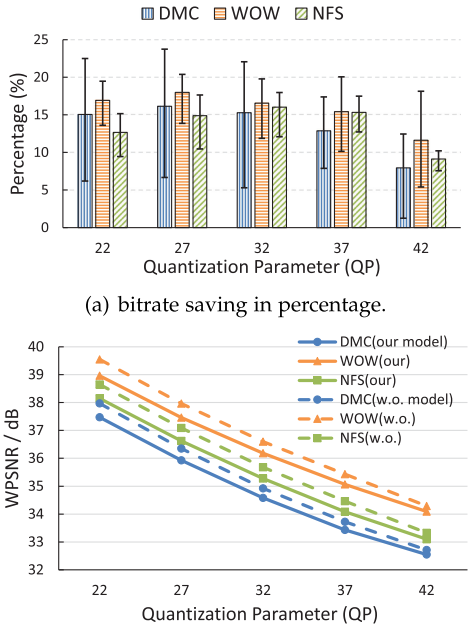
As shown in Fig. 7a, by incorporating the prioritized encoding model, bandwidth consumption can be significantly reduced about 14.3 percent on average (ranging from 8.0 percent to 18.0 percent) at similar perceptual quality. In Fig. 7a, we can observe that bitrate saving is about 15 percent with $QP = 22, 27, 32, 37$ setting for all testing sequences. For $QP = 42$ case, the bitrate saving only reaches about 10 percent. This is because that with the increase of quantization step, the speed of video bitrate reduction becomes slower, as we can easily find in Eq. (6). Thus when we set $QP = 42$, bitrate saving rate may reach a bottleneck and the bitrate saving is obviously less than other QP setting. At the same time, the WPSNR of these encoded sequences is calculated by Eqs. (11) and (12). The results are illustrated in Fig. 7b. In the figure, we can find only 0.4 dB WPSNR on average (ranging from 0.2 to 0.6 dB) is loss using HVS-based prioritized algorithm compared to standard encoding method, which is very acceptable in practice. Thus, compared to the standard encoding methods with equal quantization, our HVS-based prioritized encoding can reduce bandwidth consumption remarkably without noticeable subjective quality loss.

Furthermore, the prioritized encoding model computing would only cost about 1 ms per frame (Section 5.1). So it almost won't introduce too much extra delay for real-time video encoding in cloud gaming services, which makes our algorithm practical in real life.

5.3 Adaptive Streaming

Comparison Objectives. To evaluate ARS in T-Gaming, we compared it with existing ABR algorithms under realistic network conditions. The following are the state-of-the-art congestion control algorithms for WebRTC streams, in which estimated bandwidth is usually used as target bitrate for bit rate adaption.

1) Google congestion control (GCC) [37]: it utilizes the one way delay variation through a Kalman filter, which is experienced by packets traveling from a sender to a receiver, and packet lost rate in the traveling to estimate available bandwidth. It has been implemented and used in the Chrome WebRTC stack.
2) Bottleneck Bandwidth and RTT (BBR) [38]: it uses bottleneck bandwidth and round-trip propagation delay to determine its sending rate. We took received bitrate on client as throughput and *googRTT* measured by WebRTC as RTT.



(a) bitrate saving in percentage.

(b) WPSNR results derived by Eq. (11) and (12).

Fig. 7. Comparison of bitrate saving and WPSNR of testing sequences between using HVS-based prioritized video encoding model and without the model. Error bars span the deviations between the maximum/minimum and the average.

Network Setup. To train and test ARS under various network conditions, we created a corpus of network traces by combining several public datasets: a broadband dataset provided by the FCC[18], a 3G/HSDPA mobile dataset collected in Norway [48], a 4G/LTE dataset in Belgium [49] and a 3G/4G dataset from vehicular networks collected in Sydney [50]. Among these datasets, we considered original traces whose average throughput is greater than 0.5 Mbps and whose maximum throughput is 60 Mbps. Moreover, we used a random sample of 80 percent of our corpus as the training set for ARS; the remaining 20 percent as the test set for all ABR algorithms. In total, our test set comprises of over 30 hours of network traces.

We used *TC* tool to simulate different network conditions based on the test traces. A test video consisting of DMC, WOW and NFS game scenes, was encoded by the XCODER in the constant bit rate (CBR) mode in real time at bitrate ranging from 0.5 to 25 Mpbs, while the bitrate to adjust by ARS was restricted in {1.0, 2.0, 3.0, 4.5, 6.0, 8.0, 10.0, 12.0, 15.0, 18.0, 21.0, 25.0} Mbps. The QoS factors, such as the packet loss rate, the received bitrate and RTT, as well as the

18. https://www.fcc.gov/reports-research/reports/measuring-broadband-america/raw-data-measuring-broadband-america-2016.

playback states including output frame rate, were collected by WebRTC internals and returned back to the NetRTC module at the T-Gaming server.

QoE Metrics. Recently, there are many researches modeling the QoE of video streaming over HTTP [54], [55], [56]. However, these QoE models are mainly designed for evaluating video streaming performances with HTTP-based chunk downloading. For real-time video streaming in cloud gaming service, gamers are extremely sensitive to video freezing and bad video quality. To the best of our knowledge, there is few research works on QoE metrics for cloud gaming. In this paper, we extend the general QoE metric used by MPC [55] by adding a latency penalty component, which is defined as

$$QoE(t) = q(R_t) - \mu F_t - \nu|q(R_t - q(R_{t-1}))| - \iota L_t, \tag{13}$$

for a video streaming at time t. In the Eq. (13), $q(R_t)$ represents the *bitrate utility* to reward high video quality, where R_t donates the bitrate received at time t; $-\mu F_t$ represents the *freezing penalty* to penalize video stalling in a cloud gaming session, where F_t represents the frame freezing time during time $t-1$ to t; $-\nu|q(R_t - q(R_{t-1}))|$ represents the *smoothness penalty* to penalize changes in video quality for the favor of smoothness; and $-\iota L_t$ represents the *latency penalty* to penalize interaction latency from activating a user control to receiving the corresponding response at the client, where L_t is donated as the response delay induced by cloud gaming systems. μ, ν and ι are parameters referring as freezing penalty factor, smoothness penalty factor and latency penalty factor respectively. In other words, the QoE can be computed by subtracting the freezing penalty, smoothness penalty and latency penalty from the bitrate utility. Since the marginal improvement on perceived quality decreases at higher bitrates, similar to BOLA [34], we used logarithmic utility to represent $Q(R_t)$, i.e., $Q(R_t) = log(R/R_{\min})$. In our experiments, we set μ = 64 and ν = 0.5. This is because users are more sensitive to frame freezing and less sensitive to bitrate changes in cloud gaming services.

In the experiment, ARS's ABR algorithm was trained to optimize the QoE over the entire training corpus described in Section 5.3. In the testing, we obtain the QoE for the trace i, by

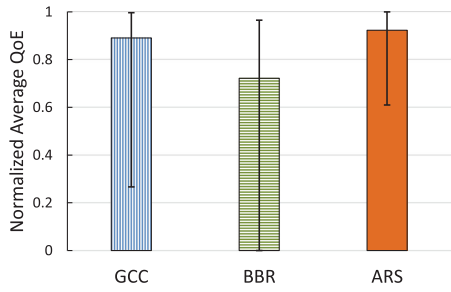
$$QoE(i) = \frac{1}{N_i}\sum_{t=1}^{N_i} QoE(i,t), \tag{14}$$

where N_i is the total trace time and $QoE(i,t)$ represents the QoE value at time t for the trace i. The resulting QoE values for all traces are then be normalized with the minimum and maximum value for these values to obtain normalized QoE, each by

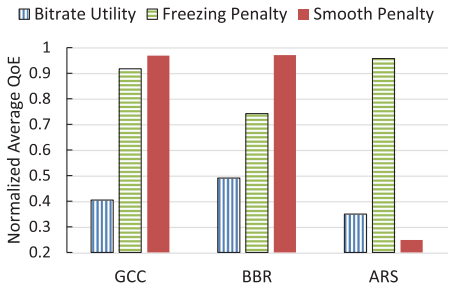
$$QoE_{norm}(i) = \frac{QoE(i) - QoE_{min}}{QoE_{max} - QoE_{min}}, \tag{15}$$

where QoE_{max} and QoE_{min} are calculated by maximizing and minimizing the QoE values for all traces in the testing. We finally get the overall normalized average QoE by averaging all obtained QoE normalized values for each trace, i.e.,

$$QoE_{norm\_avg} = \frac{1}{M}\sum_{i=1}^{M} QoE_{norm}(i), \tag{16}$$



(a) total normalized average QoE.



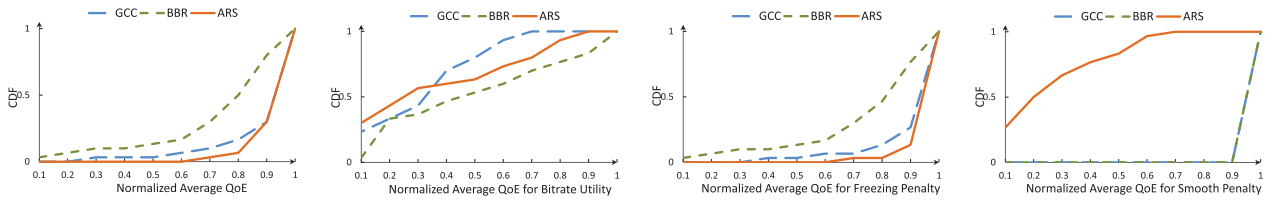
(b) normalized average QoE for each component metric.

Fig. 8. Comparing ARS in T-Gaming with existing ABR algorithms on various network conditions derived by our entire test corpus. Error bars span the deviations between the maximum/minimum and the average.

where M is the number of testing traces. The results are shown in Fig. 8a.

There are two key takeaways from these results shown in the Fig. 8a. Fist, we find that ARS outperforms all other algorithms on normalized average QoE under the test network conditions. ARS achieves 3.6 percent normalized average QoE higher than GCC and 27.9 percent higher than BBR. And ARS outperform the others with significant higher minimum value, which means ARS is more robust to various network patterns. This shows the different importance of learning states that influence ABR algorithm performance, as GCC take one-way-delay (OWD) and packet loss rate into consideration, while BBR focuses on RTT and throughput. This is because that packet lost rate is a critical factor for the WebRTC-based cloud gaming scenario. And we take *googRTT* measured by WebRTC in a higher time interval as input RTT to BBR algorithm, which may degrades its performance for available bandwidth estimation. While, ARS feeds six-dimensional states consisting of QoS parameters and playback states into its DRL network, and is aware of their impact factors by learning from the experiences. This makes ARS achieving better performance.

Second, we observe that the deviation between the maximum/minimum and the average for GCC and BBR is much larger than that of ARS. This is because that these algorithms have no knowledge of underlying network condition, so they must probe and adjust sending bitrate periodically for good performance. Once the bitrate is over adjusted, the performance decreases sharply in peak bitrate during the period. This creates a vicious circle for bit rate adaption, resulting in high QoE deviations and unstable performances. In contrast, ARS learns the feature of the underlying network by observing the network QoS and



(a) overall normalized average QoE. (b) normalized average QoE for bitrate utility. (c) normalized average QoE for freezing penalty. (d) normalized average QoE for smoothness penalty.

Fig. 9. Comparison of cumulative density function (CDF) of QoE for ARS and other ABR algorithms over 30 testing network traces.

playback states, and performs more stable and better based on these knowledge.

As overall QoE is determined by the bitrate utility, freezing penalty, and smoothness penalty components (since latency penalty is ignored), we try to find which component of QoE metric the improvement comes from. So we compute the contribution of each QoE metric component for overall QoE, and the results are shown in Fig. 8b. Similar to Fig. 8a, we normalize the average QoE contributed by each component with the minimum and maximum value in the results. The results are shown in Fig. 8b, and we could conclude that freezing penalty factor contributes most to overall QoE for ARS. This is aligned well with the fact that cloud-gaming users are more sensitive to frame freezing. Motivated by the QoE metric, (i.e., reward), ARS could produce optimal bit rate by learning the input states, which in turn benefits the QoE improvement in next time slot.

The cumulative density function (CDF) of the normalized average QoEs using ARS and other algorithms are illustrated in Fig. 9a. The CDF is computed with the normalized average QoE values over all 30 testing network traces. The conclusion can be drawn that ARS performs better than GCC and BBR under different network conditions. Fig. 9b shows the CDF of the bitrate utility for ARS is the highest at first and then tends to be relatively conservative compared with GCC and BBR results. Fig. 9c shows that freezing penalty for ARS is better than GCC and BBR, while Fig. 9d shows smoothness penalty for ARS is worse than the other two algorithms. This verifies that the freezing penalty factor contributes most to overall QoE for ARS algorithm. The reason why the worse results of smoothness penalty for ARS are produced is that we set a greater granularity of candidate bitrates used for adaption during the ARS training and testing, making the bitrate changes larger than GCC and BBR results. The smoothness penalty can be further optimized if the candidate bitrate granularity for ARS is set equivalent to GCC and BBR cases. However, this still not affect the fact that ARS outperforms GCC and BBR on the overall normalized average QoE over all testing traces.

6 Discussion

System Implementation. Since the coordinator is responsible for the user manifest management and resource management, the users cannot access the T-Gaming system if the coordinator fails. Currently, T-Gaming system adopts the mature solution provided by public cloud to offer the state-of-the-art performance. In the future, we would develop our own high availability mechanism and resource allocation algorithm in the cloud gaming data center. While, upon the unexpected shutdown of a running TG-node, T-Gaming supports to reboot the gaming session to another available node in a few minutes (similar as the new user request). Here "another available node" is selected based on the metrics of the geographical location to user and available hardware resources. Since almost all games do not support seamless hot-switching between different computers by themselves, T-Gaming currently does not consider seamless migration between nodes during a live gaming session. In addition, other specific research topics on the geographical distributed gamers and different user equipment worth for further exploration.

Response Time Reduction. Even though T-Gaming enables cloud gaming service with acceptable response delay, we can further reduce it to improve user QoE. On one hand, we can integrate hardware decoding at T-Gaming client, which can accelerate the processing by 8.9 to 28.7 percent, as shown in Section 5.1. On the other hand, T-Gaming will learn user specific gaming behaviors and then apply the learning model to detect RoI in the next version. Since these gaming behaviors are generally represented by low-dimension data, such as mouse and keyboard events, it's more efficient to detect the RoI. At the same time, the detected RoI is more accurate for user subjective perception.

Adaptive Resource and Rate Allocation. Currently T-Gaming supports both the web browser and mobile App as a client. To support prioritized video encoding, T-Gaming performs the bitrate allocation based on the type of the device, the size of the device's screen, and the distance between the player's eyes and the screen. However, different users have different distances to the screen even if they play the same game on the same device. Besides, the distance between a player and its device is hard to estimate. Fortunately, the distance always falls within an limited interval. For example, the mainstream screen size supporting 1080P resolution is ranging from 21.5 inches to 27 inches and the distance between a player's eyes and its screen is generally recommended from 50 cm to 75 cm. One solution is take the median of the interval and use it to train the rate allocation model. In the above example, we take the screen size as 24.25 inches and the distance between a player's eyes and its screen as 62.5 cm. By this way, the video perceptual quality is guaranteed for both fixed and mobile users. Regards to bit allocation category, we only consider quality domain (i.e., QP) in this paper. In later work, the joint bit allocate algorithm in quality, temporal and spatial domain for cloud gaming videos will be researched.

Periodic and Online Training. To accommodate user's request without long-time training, the RL-based ABR

algorithm is generated in an offline fashion. The computational demand of a trained model is lightweight. We implemented the ARS solution on a PC equipped with an i7-4790 @3.6 GHz CPU, and found that it costs CPU by about 0.1 percent. Further, ARS naturally supports periodical model updating, and even online realtime retraining, with the states collected in the cloud gaming sessions as new learning inputs. In the latter case, the training speed depends on the frequency of statistics returned by clients (1s as default). To avoid continuous model delivery, T-Gaming updates the algorithm periodically. By using offline and online training together, T-Gaming would achieve more robust bit rate adaption under diverse network conditions. We would like to defer this online trained ARS as our future work.

7 Conclusions

In this paper, we presented a transparent cloud gaming system called T-Gaming, which can provide excellent gaming experience to players anywhere at any time. T-Gaming is integrated with lightweight graphical container virtualization technology, dedicated acceleration scheme and self-learning-based streaming strategy, making it cost-efficient to provide ubiquitous services to both game providers and game users under various network conditions.

Through extensive evaluations, T-Gaming can achieve low response delay which only accounts for 30-40 percent of GA system, and low-cost gaming rendering solution which is less than 25 percent of per user cost by GRID based systems. Moreover, the bandwidth requirements of T-Gaming for high-quality video is less than the system without HVS-based prioritized encoding model by 14.3 percent on average. Over a broad set of network conditions, T-Gaming integrated with ARS outperforms existing ABR algorithms on normalized average QoE by 3.6–27.9 percent.

Acknowledgments

This paper is supported in part by National Natural Science Foundation of China (61650101, 61571215, 61702562, 61671265, 61902178), Natural Science Foundation of Jiangsu (BK20190295), Scientific Research Plan of the Science and Technology Commission of Shanghai Municipality (18511105400), and Shanghai Key Lab of Digital Media Processing and Transmission. H. Chen and X. Zhang contributed to this paper equally.

References

[1] Newzoo, 2017 global games market report, 2017. [Online]. Available: http://resources.newzoo.com/global-games-market-report-light-2017

[2] W. Cai, R. Shea, C.-Y. Huang, K.-T. Chen, J. Liu, V. C. Leung, and C.-H. Hsu, "A survey on cloud gaming: Future of computer games," *IEEE Access*, vol. 4, pp. 7605–7620, 2016.

[3] R. Shea, D. Fu, and J. Liu, "Cloud gaming: Understanding the support from advanced virtualization and hardware," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 25, no. 12, pp. 2026–2037, Dec. 2015.

[4] S. Shirmohammadi, M. Abdallah, D. T. Ahmed, Y. Lu, A. Snyatkov, et al., "Introduction to the special section on visual computing in the cloud: Cloud gaming and virtualization," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 25, no. 12, pp. 1955–1959, Dec. 2015.

[5] M. Suznjevic, I. Slivar, and L. Skorin-Kapov, "Analysis and qoe evaluation of cloud gaming service adaptation under different network conditions: The case of nvidia geforce now," in *Proc. 8th Int. Conf. Quality Multimedia Experience*, 2016, pp. 1–6. [Online]. Available: http://ieeexplore.ieee.org/document/7498968/

[6] H. Mao, R. Netravali, and M. Alizadeh, "Neural adaptive video streaming with pensieve," in *Proc. Conf. ACM Special Interest Group Data Commun.*, 2017, pp. 197–210.

[7] P. Guo, Q. Shen, M. Huang, R. Zhou, X. Cao, and Z. Ma, "Modeling peripheral vision impact on perceptual quality of immersive images," in *Proc. IEEE Visual Commun. Image Process.*, 2017, pp. 1–4.

[8] Z. Ma, M. Xu, Y.-F. Ou, and Y. Wang, "Modeling of rate and perceptual quality of compressed video as functions of frame rate and quantization stepsize and its applications," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 5, pp. 671–682, May 2012.

[9] Y. Xue, Y.-F. Ou, Z. Ma, and Y. Wang, "Perceptual video quality assessment on a mobile platform considering both spatial resolution and quantization artifacts," in *Proc. 18th Int. Packet Video Workshop*, 2010, pp. 201–208.

[10] Z. Ma, F. C. Fernandes, and Y. Wang, "Analytical rate model for compressed video considering impacts of spatial, temporal and amplitude resolutions," in *Proc. IEEE Int. Conf. Multimedia Expo Workshops*, 2013, pp. 1–6.

[11] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, vol. 1, Cambridge, MA, USA: MIT Press, 1998.

[12] X. Liao, L. Lin, G. Tan, H. Jin, X. Yang, W. Zhang, and B. Li, "Liverender: A cloud gaming system based on compressed graphics streaming," *IEEE/ACM Trans. Netw.*, vol. 24, no. 4, pp. 2128–2139, Aug. 2016.

[13] Y.-J. Chen, C.-Y. Hung, and S.-Y. Chien, "Distributed rendering: Interaction delay reduction in remote rendering with client-end GPU-accelerated scene warping technique," in *Proc. IEEE Int. Conf. Multimedia Expo Workshops*, 2017, pp. 67–72.

[14] X. Nan, X. Guo, Y. Lu, Y. He, L. Guan, S. Li, and B. Guo, "A novel cloud gaming framework using joint video and graphics streaming," in *Proc. IEEE Int. Conf. Multimedia Expo*, 2014, pp. 1–6.

[15] C.-Y. Huang, C.-H. Hsu, Y.-C. Chang, and K.-T. Chen, "Gaminganywhere: An open cloud gaming system," in *Proc. 4th ACM Multimedia Syst. Conf.*, 2013, pp. 36–47.

[16] 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 Trans. Circuits Syst. Video Technol.*, vol. 25, no. 12, pp. 1988–2001, Dec. 2015.

[17] Q. Hou, C. Qiu, K. Mu, Q. Qi, and Y. Lu, "A cloud gaming system based on NVIDIA GRID GPU," in *Proc. 13th Int. Symp. Distrib. Comput. Appl. Bus. Eng. Sci.*, 2014, pp. 73–77.

[18] R. Morabito, J. Kjällman, and M. Komu, "Hypervisors vs. lightweight virtualization: A performance comparison," in *Proc. IEEE Int. Conf. Cloud Eng.*, 2015, pp. 386–393.

[19] M. G. Xavier, M. V. Neves, F. D. Rossi, T. C. Ferreto, T. Lange, and C. A. De Rose, "Performance evaluation of container-based virtualization for high performance computing environments," in *Proc. 21st Euromicro Int. Conf. Parallel Distrib. Netw.-Based Process.*, 2013, pp. 233–240.

[20] Y. Liu, S. Dey, and Y. Lu, "Enhancing video encoding for cloud gaming using rendering information," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 25, no. 12, pp. 1960–1974, Dec. 2015.

[21] C. Kelton, J. Ryoo, A. Balasubramanian, and S. R. Das, "Improving user perceived page load times using gaze," in *Proc. USENIX Conf. Netw. Syst. Design Implementatio*, 2017, pp. 545–559.

[22] X. Zhang, T. Wang, J. Qi, H. Lu, and G. Wang, "Progressive attention guided recurrent network for salient object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 714–722.

[23] H. Chen and Y. Li, "Progressively complementarity-aware fusion network for RGB-D salient object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 3051–3060.

[24] W. Wang, J. Shen, X. Dong, and A. Borji, "Salient object detection driven by fixation prediction," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 1711–1720.

[25] M. Amirul Islam, M. Kalash, and N. D. Bruce, "Revisiting salient object detection: Simultaneous detection, ranking, and subitizing of multiple salient objects," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 7142–7150.

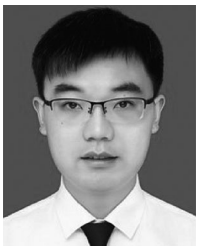
[26] L. Jiang, M. Xu, and Z. Wang, "Predicting video saliency with object-to-motion CNN and two-layer convolutional LSTM," arXiv preprint arXiv:1709.06316, 2017.

[27] L. Zhang, J. Dai, H. Lu, Y. He, and G. Wang, "A bi-directional message passing model for salient object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 1741–1750.
[28] G. Li, Y. Xie, T. Wei, K. Wang, and L. Lin, "Flow guided recurrent neural encoder for video salient object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 3243–3252.
[29] X. Nan, X. Guo, Y. Lu, Y. He, L. Guan, S. Li, and B. Guo, "Delay–rate–distortion optimization for cloud gaming with hybrid streaming," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 27, no. 12, pp. 2687–2701, Dec. 2017.
[30] H. Ahmadi, S. Khoshnood, M. R. Hashemi, and S. Shirmohammadi, "Efficient bitrate reduction using a game attention model in cloud gaming," in *Proc. IEEE Int. Symp. Haptic Audio Visual Environments Games*, 2013, pp. 103–108.
[31] J. Jiang, V. Sekar, and H. Zhang, "Improving fairness, efficiency, and stability in HTTP-based adaptive video streaming with festive," *IEEE/ACM Trans. Netw.*, vol. 22, no. 1, pp. 326–340, Feb. 2014.
[32] Y. Sun, X. Yin, J. Jiang, V. Sekar, F. Lin, N. Wang, T. Liu, and B. Sinopoli, "CS2P: Improving video bitrate selection and adaptation with data-driven throughput prediction," in *Proc. Conf. ACM SIGCOMM Conf.*, 2016, pp. 272–285.
[33] T.-Y. Huang, R. Johari, N. McKeown, M. Trunnell, and M. Watson, "A buffer-based approach to rate adaptation: Evidence from a large video streaming service," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 4, pp. 187–198, 2015.
[34] K. Spiteri, R. Urgaonkar, and R. K. Sitaraman, "Bola: Near-optimal bitrate adaptation for online videos," in *Proc. 35th Annu. IEEE Int. Conf. Comput. Commun.*, 2016, pp. 1–9.
[35] F. Chiariotti, S. D'Aronco, L. Toni, and P. Frossard, "Online learning adaptation strategy for dash clients," in *Proc. 7th Int. Conf. Multimedia Syst.*, 2016, Art. no. 8.
[36] J. van der Hooft, S. Petrangeli, M. Claeys, J. Famaey, and F. De Turck, "A learning-based algorithm for improved bandwidth-awareness of adaptive streaming clients," in *Proc. IFIP/IEEE Int. Symp. Integr. Netw. Manag.*, 2015, pp. 131–138.
[37] G. Carlucci, L. De Cicco, S. Holmer, and S. Mascolo, "Analysis and design of the google congestion control for web real-time communication (WEBRTC)," in *Proc. 7th Int. Conf. Multimedia Syst.*, 2016, Art. no. 13.
[38] N. Cardwell, Y. Cheng, C. S. Gunn, S. H. Yeganeh, and V. Jacobson, "BBR: Congestion-based congestion control," *Queue*, vol. 14, no. 5, 2016, Art. no. 50.
[39] T. Huang, R.-X. Zhang, C. Zhou, and L. Sun, "Qarc: Video quality aware rate control for real-time video streaming based on deep reinforcement learning," in *Proc. 26th ACM Int. Conf. Multimedia*, 2018, pp. 1208–1216. [Online]. Available: http://doi.acm.org/10.1145/3240508.3240545
[40] Y. Xu, Q. Shen, X. Li, and Z. Ma, "A cost-efficient cloud gaming system at scale," *IEEE Netw.*, vol. 32, no. 1, pp. 42–47, Jan./Feb. 2018.
[41] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the h.264/avc video coding standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 560–576, Jul. 2003.
[42] G. J. Sullivan, J. Ohm, W.-J. Han, and T. Wiegand, "Overview of the high efficiency video coding (HEVC) standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1649–1668, Dec. 2012.
[43] H. Strasburger, I. Rentschler, and M. Jüttner, "Peripheral vision and pattern recognition: A review," *J. Vis.*, vol. 11, no. 5, pp. 13–13, 2011.
[44] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 1928–1937.
[45] X. Zhang, H. Chen, Y. Zhao, Z. Ma, Y. Xu, H. Huang, H. Yin, and D. O. Wu, "Improving cloud gaming experience through mobile edge computing," *IEEE Wireless Commun.*, pp. 1–6, 2019.
[46] J. Zhang, S. Sclaroff, Z. Lin, X. Shen, B. Price, and R. Mech, "Minimum barrier salient object detection at 80 fps," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 1404–1412.
[47] F. C. Commission, Raw data - measuring broadband america, 2016. [Online]. Available: https://www.fcc.gov/reports-research/reports/measuring-broadband-americ%a/raw-data-measuring-broadband-america-2016
[48] H. Riiser, P. Vigmostad, C. Griwodz, and P. Halvorsen, "Commute path bandwidth traces from 3G networks: Analysis and applications," in *Proc. 4th ACM Multimedia Syst. Conf.*, 2013, pp. 114–118.
[49] J. van der Hooft, S. Petrangeli, T. Wauters, R. Huysegems, P. R. Alface, T. Bostoen, and F. De Turck, "Http/2-based adaptive streaming of HEVC video over 4G/lte networks," *IEEE Commun. Lett.*, vol. 20, no. 11, pp. 2177–2180, Nov. 2016.
[50] A. Bokani, M. Hassan, S. S. Kanhere, J. Yao, and G. Zhong, "Comprehensive mobile bandwidth traces from vehicular networks," in *Proc. 7th Int. Conf. Multimedia Syst.*, 2016, Art. no. 44.
[51] H. B. Demuth, M. H. Beale, O. De Jess, and M. T. Hagan, "Neural network design," 2nd ed. USA: Martin Hagan, 2014.
[52] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Proc. Advances Neural Inf. Process. Syst.*, 2000, pp. 1057–1063.
[53] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al., "Tensorflow: A system for large-scale machine learning," in *Proc. 12th USENIX Conf. Operating Syst. Des. Implementation*, 2016, vol. 16, pp. 265–283.
[54] A. Bokani, M. Hassan, S. Kanhere, and X. Zhu, "Optimizing HTTP-based adaptive streaming in vehicular environment using markov decision process," *IEEE Trans. Multimedia*, vol. 17, no. 12, pp. 2297–2309, Dec. 2015.
[55] X. Yin, A. Jindal, V. Sekar, and B. Sinopoli, "A control-theoretic approach for dynamic adaptive video streaming over HTTP," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 45, no. 4, pp. 325–338, 2015.
[56] T. Mangla, E. Halepovic, M. Ammar, and E. Zegura, "Mimic: Using passive network measurements to estimate HTTP-based adaptive video QoE metrics," in *Proc. Netw. Traffic Meas. Anal. Conf.*, 2017, pp. 1–6.

Hao Chen received the BE degree in electronics and information engineering from Northwestern Polytechnical University, China, in 2013. He is working toward the PhD degree at the Cooperative Medianet Innovation Center, Shanghai Jiao Tong University, China. His research focus on real-time video streaming, joint source-channel coding and machine learning.

Xu Zhang received the BSc degree in communication engineering from the Beijing University of Posts and Telecommunications, China, in 2012, and the PhD degree from the Department of Computer Science and Technology, Tsinghua University, China, in 2017. He is an associate professor with the School of Electronic Science and Engineering, Nanjing University, China. He was a visiting student at the Department of Electrical and Computer Engineering, University of Florida, USA, from 2015 to 2016. His research interests include content delivery networks, network measurement and cloud gaming.

Yiling Xu received the BS, MS and PhD from the University of Electronic Science and Technology of China, China, in 1999, 2001 and 2004, respectively. She is a full researcher of School of Electronic Information and Electronic Engineering, Shanghai Jiao Tong University, Shanghai, 200145, China. From 2004 to 2013, she was with Multimedia Communication Research Institute of Samsung Electronics Inc, Korea. Her research interest mainly in architecture design for next generation multimedia systems, dynamic data encapsulation, adaptive cross layer design, dynamic adaption for heterogenous networks and N-screen content presentation.



Ju Ren (S'13, M'16) received the BSc, MSc, PhD degrees in computer science, from Central South University, China, in 2009, 2012, and 2016, respectively. During 2013-2015, he was a visiting PhD student in the Department of Electrical and Computer Engineering, University of Waterloo, Canada. Currently, he is a tenure-tracking professor with the School of Computer Science and Engineering, Central South University, China. His research interests include Internet-of-Things, network computing and cloud computing. He is a co-recipient of the best paper awards of IEEE ICC'19 and IEEE IoP'18, and the most popular paper award of Chinese Journal of Electronics (2015-2018). He currently serves/has served as an associate editor for IEEE Transactions on Vehicular Technology and Peer-to-Peer Networking and Applications, a guest editor for IEEE Transactions on Industrial Informatics and IEEE Network, and a TPC member of many international conferences including IEEE INFOCOM'20/19/18, Globecom'17, WCNC'17, etc. He also served as the TPC chair of IEEE BigDataSE'19, a poster co-chair of IEEE MASS'18, a track co-chair for IEEE/CIC ICCC'19, IEEE I-SPAN'18 and VTC'17 Fall, and an active reviewer for over 20 international journals. He is a member of IEEE and ACM.

Jingtao Fan received the bachelor's and master's degrees in computer science and technology, and the PhD degree in optical engineering from the Changchun University of Science and Technology, Changchun, China, in 2003, 2007 and 2013, respectively. He is an associate professor of Department Automation, Tsinghua University. His research interests mainly include the computational photography and computer vision.

Zhan Ma (SM'19) received the BS and MS degrees from Huazhong University of Science and Technology (HUST), Wuhan, China, in 2004 and 2006, respectively, and the PhD degree from the New York University, New York, in 2011. He is now on the faculty of Electronic Science and Engineering School, Nanjing University, Jiangsu, 210093, China. From 2011 to 2014, he has been with Samsung Research America, Dallas TX, and Futurewei Technologies, Inc., Santa Clara, CA, respectively. His current research focuses on the next-generation video coding, energy-efficient communication, gigapixel streaming and deep learning. He is a co-recipient of 2018 ACM SIGCOMM Student Research Competition Finalist, 2018 PCM Best Paper Finalist, and 2019 IEEE Broadcast Technology Society Best Paper Award.

Wenjun Zhang (M'02, SM'10, F'12) received the BS, MS, and PhD degrees in electronic engineering from Shanghai Jiao Tong University, Shanghai, China, in 1984, 1987, and 1989, respectively. He joined the Faculty of Shanghai Jiao Tong University, in 1993, and became a full professor with the Department of Electronic Engineering in 1995. He holds more than 40 patents and authored or coauthored more than 90 papers in international journals and conferences. His main research interests include digital video coding and transmission, multimedia semantic processing, and intelligent video surveillance. He is a chief scientist with the Chinese National Engineering Research Centre of Digital Television (NERC-DTV), an industry/government consortium in DTV technology research and standardization, and the chair of the Future of Broadcast Television Initiative (FOBTV) Technical Committee. As the national HDTV TEEG project leader, he successfully developed the first Chinese HDTV prototype system in 1998. He was one of the main contributors to the Chinese Digital Television Terrestrial Broadcasting Standard issued in 2006 and is leading team in designing the next generation of broadcast television system in China since 2011.