Improving Cloud Gaming Experience through Mobile Edge Computing

Xu Zhang, Hao Chen, Yangchao Zhao, Zhan Ma, Yiling Xu, Haojun Huang, Hao Yin, and Dapeng Oliver Wu

Abstract

With the development of 4G/5G technology and smart devices, more and more users begin to play games via their mobile devices. As a promising way to enable users to play any games, cloud gaming is proposed to stream game scene rendered remotely in the cloud with the format of video. However, it faces major challenges in terms of long delay and high network bandwidth. To this end, a novel framework named *EdgeGame* is proposed to improve the cloud gaming experience by leveraging resources in the edge. Compared to existing cloud gaming systems, *EdgeGame* offloads the computation-intensive rendering to the network edge instead, which can reduce network delay and bandwidth consumption greatly. Moreover, *EdgeGame* introduces deep reinforcement learning in the edge to adjust the video bitrates adaptively to accommodate the network dynamics. Finally, we implemented a prototype system and compared it with an existing cloud gaming system. The experiments show that *EdgeGame* can reduce the average network delay by 50 percent and improve user's QoE by 20 percent.

Introduction

The global gaming market was valued at US$106.87 billion in 2017 and is expected to reach US$158.33 billion by 2023 growing at a CAGR of 6.77 percent. One of the key drivers is the growing adoption of smartphones and other mobile devices. According to Newzoo's report, 50 percent of the global games market will be from mobile games in 2020 (Global Gaming Market — Forecasts from 2018 to 2023. https://www.researchandmarkets.com/research/7ntx4c/global gaming?w=4, 2018). With the prevalence of mobile games, users expect to enjoy any game from anywhere and anytime, even with limited computation/storage/power capacity. To this end, cloud gaming is proposed and has attracted much attention in the literature [1, 2] and from industry (The Power of Cloud Gaming, NVIDIA, http://www.nvidia.com/object/cloud-gaming.html, 2018).

Cloud gaming is a promising way to guarantee users' gaming experience as long as their clients have the basic capacities for network connection, audio/video decoding and image display. For cloud gaming, games are rendered remotely in the cloud and streamed to users as a video sequence, while users' interactive movements, including keyboard events and mouse clicks, are captured from the client and sent back to the cloud. By offloading the computation-intensive rendering and storage-intensive hosting to the cloud, cloud gaming can overcome the challenge that mobile devices can hardly run the huge-resource-consuming games [1, 3]. However, there still remain significant challenges toward its widespread deployment due to the features of cloud gaming.

Low Delay Requirement: As an application with strong interactivity, cloud gaming has rigorous requirements on the network delay. When the latency is higher than 50ms, players will leave the game as the gaming experience is too bad to bear [1]. However, in the existing cloud-based strategy, players interact with servers in the remote cloud directly. On one hand, the network delay is lower limited by the physical distance between players and the cloud, not to mention the influence of network dynamics. Due to the limited numbers of data centers, the network delay is too high for guaranteeing users' gaming experience in most cases. On the other hand, the long transmission path between the players and the cloud is more likely to experience network congestions, which will further increase the delay.

High Bandwidth Consumption: To sustain the affordable-quality gaming experience, the recommended downstream bandwidth is 3Mb/s for a single user [2]. Moreover, the required bandwidth is the minimum bandwidth along the path from the cloud server to the mobile users. On one hand, the aggregated mobile gaming traffic will cause congestion in the core network, making the network unable to bear other services. On the other hand, the cloud server can hardly provide enough downstream bandwidth due to the physical limitation when the number of users increases to hundreds of millions. What's worse, the huge bandwidth consumption in the data center incurs high operational cost for service providers.

Sensitive Users' QoE: As users are highly engrossed when playing games, any flaws during the game, such as rebuffering and unsmoothness, would degrade user's Quality of Experience (QoE) greatly. However, networks are always highly dynamic. On one hand, network congestion would occur from time to time, especially when flash crowds happen. On the other hand, the access networks for mobile users are always unstable and fragile. How to guarantee users' QoE in dynamic networks is a great challenge.

Digital Object Identifier:
10.1109/MWC.2019.1800440

Xu Zhang, Yangchao Zhao and Zhan Ma are with Nanjing University; Hao Chen and Yiling Xu are with Shanghai Jiao Tong University; Haojun Huang is with China University of Geosciences; Hao Yin is with Tsinghua University; Dapeng Oliver Wu is with University of Florida.

1536-1284/19/$25.00 © 2019 IEEE

With the advent of 5G, more and more computing, storage and networking resources are integrated with the base station, known as edge devices or edge nodes in the paradigm of Mobile Edge Computing [4, 5]. This article proposes a novel framework called *EdgeGame* to enable mobile users to enjoy any game from anywhere and anytime. EdgeGame takes advantage of the infrastructure in the edge to reduce network delay and save bandwidth cost. Moreover, *EdgeGame* introduces reinforcement learning in the edge to optimize users's QoE. We first describe the system framework of *EdgeGame* and then detail each component in *EdgeGame*. Next, we provide a case study to verify the reinforcement learning algorithm in improving user's QoE. The final section concludes our work and illustrates some open issues for future study.

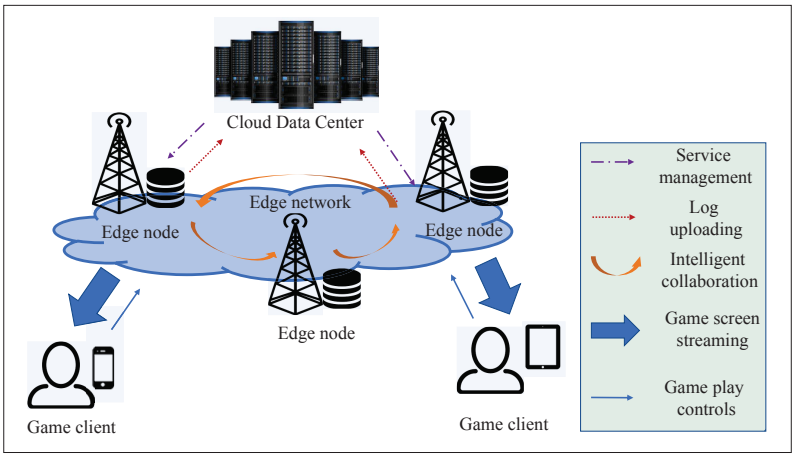

FIGURE 1. Framework of EdgeGame.

System Overview

Cloud gaming consumes a large number of computation and communication resources simultaneously, which limits its prevalence greatly. This is because the cloud needs to update the game logic and render the game scene according to the player's commands in real time, which is highly computation-intensive. Moreover, the game scene is sent to the player as a video sequence for interaction with low delay requirements, consuming large amounts of bandwidth across the entire network from the cloud to end users. To this end, *EdgeGame* takes advantage of the computation and caching resources in the edge to decrease the requirements of communication resources. By adapting the distribution of edge nodes, *EdgeGame* can achieve a flexible trade-off between the computation, caching and communication resources. Furthermore, *EdgeGame* adopts artificial intelligence (AI) approaches in the edge to guarantee users' QoE within the dynamic edge network.

As shown in Fig. 1, the computation-intensive GPU rendering is placed at the edge node, which streams the scenes as a video sequence to the game client, while the game client is responsible for displaying the video as well as collecting the player's commands and sending the interactions back to the edge nodes. For multi-player games, the game logic should be synchronized between the involved edge nodes. In this situation, the edge nodes would collaborate with each other to improve players' QoE. For example, messages about the game logic updating can be routed by other intermediate edge nodes in order to achieve lower network delay. Meanwhile, edge nodes upload the game log to the cloud data center in case the game should be resumed or replayed.

Mobile Edge Computing in *EdgeGame*: In order to reduce network delay and the bandwidth consumption from the cloud, *EdgeGame* places the 2D/3D graphic rendering at the edge. On one hand, the user client is free from the computation-intensive task, making the game playable on any device. On the other hand, users fetch the gaming video from the "closest" edge nodes, which can shorten network delay and split the traffic from the cloud to numerous edge nodes.

Artificial Intelligence in *EdgeGame*: In order to guarantee users' QoE even within dynamic wireless networks, *EdgeGame* adopts artificial intelligence in the edge nodes. The traffic for game logic updating is routed intelligently, relayed by the edge nodes. Intelligent routing can choose the "best" path by the trade-off between the bandwidth and network delay. Moreover, the traffic from the edge nodes to the users is adjusted using a deep-reinforcement-learning-based algorithm to accommodate the varying bandwidth in the dynamic network.

Components

As mentioned earlier, *EdgeGame* consists of three key components, including the game clients, the edge network, and the cloud data center, where the thin game client contains the basic communication and display capacities. In the following, we highlight the other two components and detail *EdgeGame*'s workflow.

Cloud Data Center

As a new computing paradigm, cloud computing supports services scaling up or down flexibly and service provider pay as you go for resource consumption. In this way, more and more service providers deploy their services in the cloud. However, the speed of data transportation from the edge to the cloud makes the paradigm unsuitable for online games, as games require the network delay to be low enough to guarantee users' QoE during the interaction. To this end, *EdgeGame* decouples the multiple functions in cloud gaming, and schedules them among the cloud data center and the edge nodes. In detail, the cloud data center acts as the portal, the monitoring system, the naming system and the management platform for gaming.

Portal: Basically, the data center maintains the account/password database for all the users and all the games, and it is responsible for game type selection, registering new users, existing account verification and account recovery. When a user requests to play a game, the data center would first send the login page to them and then provide the account login services.

Monitoring: Next, the data center has a bird's-eye view of *EdgeGame* by monitoring the network status, the edge node overhead, the running game performance and so on. As to the network status, *EdgeGame* takes advantage of the widely-distributed edge nodes to measure the network, including network latency, network bandwidth and packet

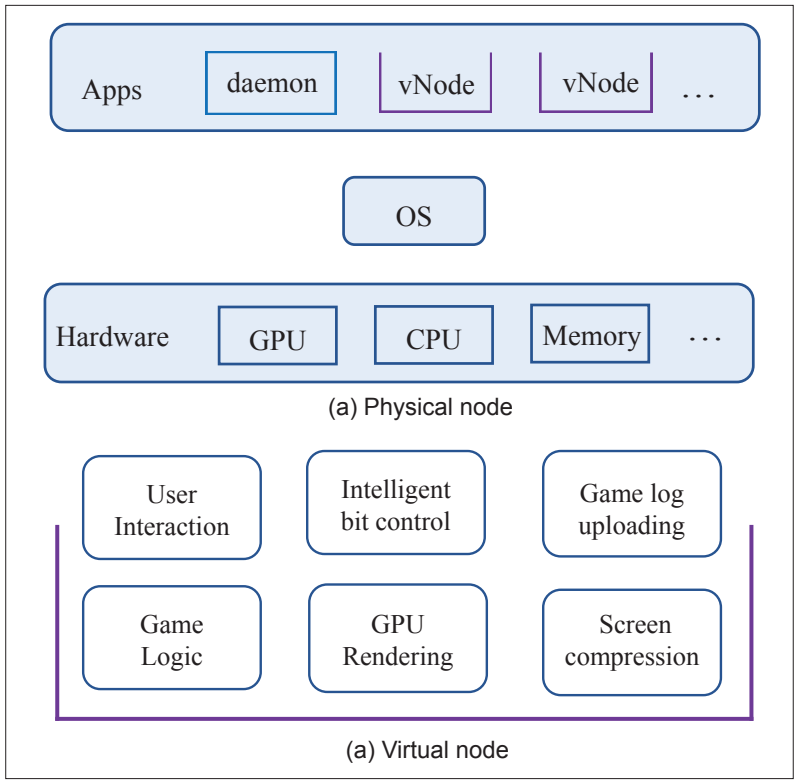

FIGURE 2. The structure of an edge node.

loss rate [6]. As to the edge node overhead and running game performance, the edge nodes would report them to the data center periodically.

Naming: Then, when a user makes a request to play a game, the request will be routed to the cloud data center. The cloud data center determines all the feasible edge nodes that can provide good enough service to the user, that is, the network delay and the available bandwidth between the user and the edge node satisfies the requirements of the objective game. Next, the cloud data center selects the best edge node considering the load balancing and the service cost. Finally, the address of a newly-launched virtual node (vNode) on the selected edge node is sent back to the user.

Management: Another important role that the cloud data center plays is to manage the whole system. For example, after selecting the vNode for a user, the cloud data center should send a vNode-launching demand and the CPU/GPU/Memory configuration scheme to the edge node based on the type of game.

Apart from the above functions, the cloud data center also has the ability to provide gaming services to the users in case no suitable edge node exists to provide satisfactory gaming services to users. Moreover, it collects the gaming logs from the edge nodes about the users' operations, the network status, the bitrate of the gaming video and so on. Using these data, *EdgeGame* can construct the intelligent control model using artificial intelligent approaches and support resuming the game by performing game logic transfer if failure happens.

Edge Network

In MEC, more and more service providers have begun to deploy computing, storage and networking resources in the network [7], which are well connected to the Internet and available for use by nearby devices [8]. These nodes in close proximity to the users enable a dramatic improvement in customer experience through low latency interaction with computation and storage resources just one hop away from the user. Considering this, *EdgeGame* offloads the computation-intensive GPU rendering in gaming to the edge nodes and frees the game players from expensive CPU/GPU hardware, making the game playable on any device.

As shown in Fig. 2a, *EdgeGame* adopts the sandbox technique to accommodate multiple games running on the same physical nodes simultaneously, forming multiple vNodes. A game running in a vNode has lower startup time compared with running in a virtual machine (VM). Moreover, the vNode can separate the running games from the underlying operating system in the physical node, preventing unwanted changes from happening to the data, programs and applications that rest safely on the edge nodes. Apart from the vNodes, a daemon is also running in the physical node, which is responsible for communicating with other edge nodes and data centers, managing and monitoring the vNodes.

When a user makes a request for playing a cloud game, a vNode would be launched in an edge node nearby, which is responsible for hosting the game. When the user is playing the game, the vNode accepts the demands from the user, updates the game logic, renders the game scene, compresses the scene into videos, and streams the video to the user. In detail, a vNode has the following functions, as shown in Fig. 2b.

User Interaction: This function mainly acts as the interaction interface with the game client, including receiving the control demands from the player and streaming the game video to the player based on the Web Real-time Communication (WebRTC) protocol [9].

Game Logic: This function maintains the status of the current game and updates the status according to the commands received from the user interaction function.

GPU Rendering: This function is responsible for rendering the game scene based on the 2D/3D game scene files and the latest game status provided by the game logic function.

Screen Compression: This function generates the realtime game video based on the rendering game scene, which can be directly displayed on the game client.

Intelligent Bit Control: This function analyzes the network conditions, and determines the compression parameters for the screen compression function using deep reinforcement learning techniques.

Game Log Uploading: This function will forward the control demands to the cloud data centers in case the edge node crashed or the played game needs to be replayed afterward.

Note that these edge nodes form an edge network and can collaborate with each other if needed. For example, service providers can take advantage of the widely-distributed edge nodes to predict the path performance between an arbitrary game player and an edge node or the path performance between two arbitrary edge nodes. After obtaining the network performance, any

messages from the cloud server to a game player can be relayed by the edge nodes if shorter paths are found between the edge nodes [10]. This happens when an edge node crashed, the related game status is sent from the cloud data center to newly-launched vNodes in other edge nodes. *EdgeGame* adopts a deep-learning-based routing strategy similar to [11].

Workflow

Figure 3 shows the workflow when *EdgeGame* provides the gaming service to users. First, a user sends requests to the cloud data center, logs in the system and selects the objective game. Second, the cloud data center chooses the appropriate edge node based on the network delay/bandwidth between the user and edge nodes and the load conditions on the edge nodes, and sends a vNode-launching demand with the CPU/GPU/Memory configuration to the edge node based on the type of game. In the meantime, the cloud data center returns the address of the vNode to the user. Third, the user begins to play the game and the game client sends the input signals to the vNode, such as the keyboard events, the mouse clicks and joystick movements. Fourth, receiving the user's input, the vNode updates the game logic, renders the game scene, and streams the game scene as a video sequence back to the user. Note that the bit rate of the video is adjusted adaptively to accommodate the dynamical network. At the same time, the vNode uploads the log of the user's input to the cloud data center. During the game, the last two procedures execute alternately until the game ends. The preparation can be finished during the game starting/loading/two-sided matching.

Artificial Intelligence in the Edge

To enable games to be played on any device, cloud gaming offloads the computation-intensive GPU rendering to the cloud and streams the game scenes to users as video sequences, while users' interactive movements, including keyboard events and mouse clicks, are captured from the client and sent back to the cloud. As mentioned previously, this paradigm suffers from long network delay, high bandwidth consumption and sensitive users' QoE. To overcome the first two challenges, *EdgeGame* adopts the emerging edge computing approach by offloading the game rendering to the nearby edge node rather than the remote cloud. As to the last challenge, *EdgeGame* adopts the artificial intelligence approache in the edge to guarantee users' QoE within dynamic networks.

Neural Adaptive Game Streaming (NAGS): As the available bandwidth between the user and the edge node varies from time to time, *EdgeGame* would also adjust the bit rate of the gaming video sequence dynamically. However, existing adaptive streaming strategies mainly focus on video streaming using the HTTP protocol, which does not apply to cloud gaming using the WebRTC protocol, while the default rate-control algorithm Google Congestion Control (GCC) for WebRTC relies on the accurate prediction of the network status [9]. Moreover, existing QoE models cannot be used for cloud gaming directly as they are designed for HTTP-chunk-based video streaming [12].

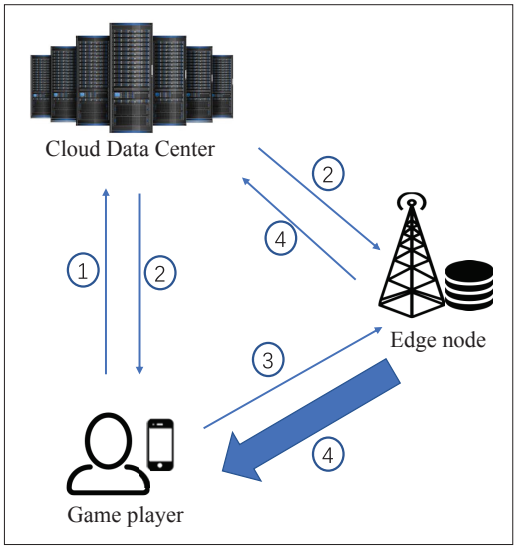

FIGURE 3. Workflow in EdgeGame.

To this end, we first define the QoE model based on the GOP (Group of Pictures), which is a collection of successive pictures within a coded video stream. Encountering a new GOP in a compressed video stream means that the decoder does not need any previous frames in order to decode the next ones. The QoE is based on the video bitrate of each GOP, the smoothness of videos between adjacent GOPs, and the rebuffering in the game client. In detail, the QoE for a video streaming at time t is defined as follows.

$$QoE_t = \log(R_t/R_{min}) - \mu T_t - \nu|\log(R_t/R_{t-1}), \quad (1)$$

where R_t is the bitrate at time point t and $\log(R_t/R_{t-1})$ represents the quality perceived by a user; T_t is the stalling time and $|\log(R_t/R_{t-1})|$ is the penalty for video bitrate changes, that is unsmoothness; μ and ν are two parameters referred to as the freeze penalty factor and unsmooth penalty factor, respectively.

On the other hand, we develop a deep-reinforcement-learning-based adaptive bit rate algorithm called NAGS. The algorithm selects the bitrates for the future game video streaming based on the data collected from the game clients, without assumptions about the network condition change rule.

As shown in Fig. 4, *EdgeGame* inputs the bitrate of the last GOP, the current buffer occupancy in game client, the past k received bitrates, the past k jitter buffer times, the past k packet loss rates and the past k NACK sent counts into the network. In detail, after the streaming of each GOP t, *EdgeGame* takes state inputs $s_t = (r_t, \tau_t, x_t, b_t, p_t, n_t)$ to its neural networks, where r_t is the bitrate at which the last GOP was streamed; τ_t is the current buffer level; x_t is the received bitrate measured for the last video GOP; b_t is the current jitter buffer size level; p_t is the packet loss rate tallied; and n_t is the NACK sent count during the last GOP streaming. Moreover, *EdgeGame* trains the neural network using the state-of-the-art actor-critic RL algorithm named A3C [13].

Performance

We built a prototype system in Mainland China. In the system, the cloud data center is placed in the central part in the country,

To enable games to be played on any device, cloud gaming offloads the computation-intensive GPU rendering to the cloud and streams the game scenes to users as video sequences, while users' interactive movements, including keyboard events and mouse clicks, are captured from the client and sent back to the cloud.

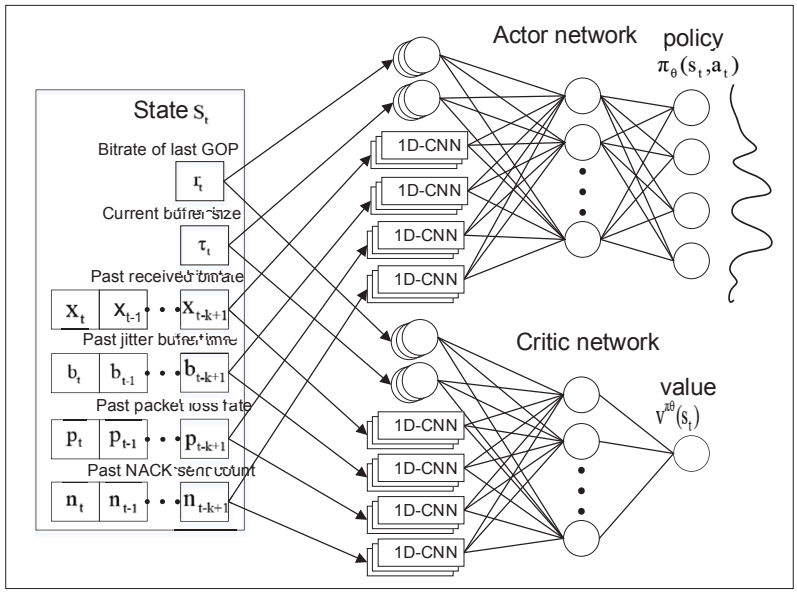

FIGURE 4. The learning network to adjust the bit rate adaptively in EdgeGame.

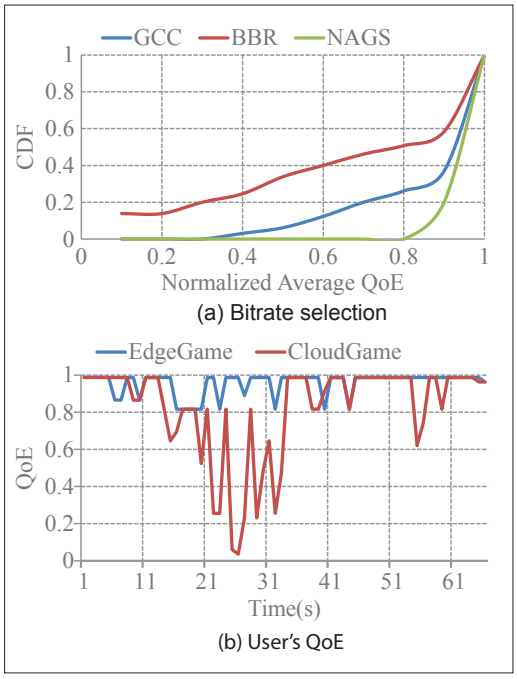

FIGURE 5. The performance comparison between EdgeGame and existing schemes: a) Bitrate selection; b) User's QoE.

while the edge node is located in the same city with the game player. To avoid "cold start" in the initial stage, we trained an offline neural network model based on several public bandwidth datasets [14]. Given a pre-trained model offline as a priori, *EdgeGame* enables the neural network to update periodically as new actual data arrives after being deployed in the real environment.

By streaming real cloud gaming video sequences over the simulated network following the actual gaming process logic, the network QoS (Quality of Service) parameters and playback statuses were easily obtained. The test gaming videos include Devil May Cry (DMC), World of Warcraft (WOW) and Need For Speed (NFS) game scenes. The optional bitrates for the gaming video are 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} mb/s. The bitrate of the last GOP, the current buffer occupancy in the game client, the past *k* received bitrates, the past *k* jitter buffer times, the past *k* packet loss rates, and the past *k* NACK sent counts, were collected by WebRTC internals and returned back to the game server. We compared NAGS with existing selection strategies, including the GCC integrated in WebRTC [9] and a bottleneck bandwidth and RTT based control algorithm (BBR) [15]. As shown in Fig. 5a, the normalized average QoE higher than 0.8 is highest in NAGS, followed by GCC, and BBR is the lowest. This is because that NAGS can take more factors into consideration when determining the bitrate.

Next, we compared the user's QoE when playing games in *EdgeGame* with that in a traditional cloud gaming system (CloudGame) [2]. In our experiment, the average network delay in *EdgeGame* is 16.2ms while the average network delay in CloudGame is larger than 44.2ms. Moreover, users' QoE are more stable and higher in *EdgeGame* than in CloudGame, as shown in Fig. 5b. This is because the long network paths between game players and the remote cloud servers are more likely to experience network dynamics, such as network congestion.

Open Issues

In this section, we discuss two open issues for *EdgeGame*'s development in the future.

Crowdsourced Edge Network: As deploying edge nodes widely in the large-scale network incurs high investment cost for service providers, an alternative solution is to utilize the computation, storage and network resources of end users' idle devices. Although the game client in *EdgeGame* is designed to be thin in order to support mobile users, a considerable portion of the users would play the games on their home computers. These computers with abundant resources will be idle when the users go to work or sleep. In the future, *EdgeGame* will encourage these users or even private/professional GPU owners to contribute their devices to the edge network. These devices are closer to users, thus providing better services to game players. Moreover, the investment cost for service providers is greatly reduced by decreasing the number of dedicated edge nodes. Note that users' computers will be less stable compared with dedicated edge nodes, so *EdgeGame* should design a specific mechanism to guarantee users' QoE with the collaboration between users' computers and dedicated edge nodes.

Blockchain-Based Incentive Mechanism: A key question for crowdsourced edge networks is how to stimulate users to contribute their devices to the edge network. A possible mechanism is that the users can play games for free or even make money if they contribute their devices. Moreover, the free time or the revenue is based on the amount of contributed resources and duration time. In the future, *EdgeGame* will introduce blockchain-based technologies to solve users' doubts about their contribution. Users will share their resources with the edge network and be compensated for their participation by token-based contributions from gamers.

Conclusion

This work introduces a novel framework called *EdgeGame* to enable cloud gaming at scale, which includes the thin game client, the edge network and cloud data center. To reduce network delay and bandwidth consumption in existing cloud gaming system, *EdgeGame* takes advantage of the computation and caching resources in the edge to decrease the requirements of communication resources by offloading the game rendering to the nearby edge node rather than the remote cloud. Moreover, *EdgeGame* defines a novel QoE metric based on GOP for cloud gaming and adopts a deep-reinforcement-learning-based adaptive bit rate algorithm in the edge to guarantee users' QoE in dynamic networks. Future work includes the design of a crowdsourced edge network and a Blockchain-based incentive mechanism.

Acknowledgment

This work was supported in part by the National Key Research and Development Program under Grant No. 2016YFB1000102; in part by NSF ECCS-1509212; and in part by the National Natural Science Foundation of China under Grant No. 61529101, No. 61571215, No. 61650101, and No. 61672318. The corresponding authors are Zhan Ma, Yiling Xu, and Dapeng Oliver Wu.

References

[1] Y. Xu *et al.*, "A Cost-Efficient Cloud Gaming System at Scale," *IEEE Network*, vol. 32, no. 1, Jan. 2018, pp. 42–47.
[2] C.-Y. Huang *et al.*, "GamingAnywhere: The First Open Source Cloud Gaming System," *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 10(1s), Jan. 2014, pp. 10:1–25.
[3] D. Zhang *et al.*, "Delay-Optimal Proactive Service Framework for Block-Stream as a Service," *IEEE Wireless Commun. Lett.*, vol. 7, no. 4, Aug. 2018, pp. 598–601.
[4] L. Xiao *et al.*, "Security in Mobile Edge Caching with Reinforcement Learning," *IEEE Wireless Commun.*, vol. 25, no. 3, June 2018, pp. 116–22.
[5] X. Zhang *et al.*, "Resource Provisioning in the Edge for IoT Applications with Multi-Level Services," *IEEE Internet of Things J.*, 2018, pp. 1–1.
[6] X. Zhang *et al.*, "SSL: A Surrogate-Based Method for Large-Scale Statistical Latency Measurement," *IEEE Trans. Services Computing*, vol. PP, no. 99, 2017, pp. 1–1.
[7] H. Yin *et al.*, "Edge Provisioning with Flexible Server Placement," *IEEE Trans. Parallel & Distributed Systems*, vol. 28, no. 4, Apr. 2017, pp. 1031–45.
[8] T. X. Tran *et al.*, "Collaborative Mobile Edge Computing in 5G Networks: New Paradigms, Scenarios, and Challenges," *IEEE Commun. Mag.*, vol. 55, no. 4, Apr. 2017, pp. 54–61.
[9] G. Carlucci *et al.*, "Analysis and Design of the Google Congestion Control for Web Real-Time Communication (webrtc)," *Proc. 7th Int'l. Conf. Multimedia Systems*, MMSys '16, New York, NY, USA, 2016. ACM, pp. 13:1–13:12.
[10] C. Lumezanu *et al.*, "Triangle Inequality Variations in the Internet," *Proc. 9th ACM SIGCOMM Conf. Internet Measurement*, IMC '09, New York, NY, USA, 2009, ACM, pp. 177–83.
[11] N. Kato *et al.*, "The Deep Learning Vision for Heterogeneous Network Traffic Control: Proposal, Challenges, and Future Perspective," *IEEE Wireless Commun.*, vol. 24, no. 3, 2017, pp. 146–53.
[12] H. Mao, R. Netravali, and M. Alizadeh, "Neural Adaptive Video Streaming with Pensieve," *Proc. Conf. ACM Special Interest Group on Data Communication, SIGCOMM '17*, New York, NY, USA, 2017, ACM, pp. 197–210.
[13] V. Mnih *et al.*, Asynchronous Methods for Deep Reinforcement Learning, *Proc. Int'l. Conf. Machine Learning*, 2016, pp. 1928–37.
[14] H. Riiser *et al.*, "Commute Path Bandwidth Traces from 3G Networks: Analysis and Applications," *Proc. 4th ACM Multimedia Systems Conference, MMSys '13*, New York, NY, USA, 2013, ACM, pp. 114–18.
[15] N. Cardwell *et al.*, "BBR: Congestion-Based Congestion Control," *Queue*, vol. 14, no. 5, Oct. 2016, pp. 50:20–50:53.

Biographies

Xu Zhang (xzhang17@nju.edu.cn) is an associate professor in the School of Electronic Science and Engineering, Nanjing University, China. He received the B.S. degree in communication engineering from Beijing University of Posts and Telecommunications, China, in 2012, and the Ph.D. degree from the Department of Computer Science and Technology, Tsinghua University, China, in 2017. He was a visiting student in 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 computing.

Hao Chen (chenhao1210@sjtu.edu.cn) received his B.E. degree in electronics and information engineering from Northwestern Polytechnical University, China in 2013. He is pursuing his Ph.D. degree at the Cooperative Medianet Innovation Center, Shanghai Jiao Tong University, China. His research focuses on video streaming, joint source-channel coding and machine learning.

Yangchao Zhao (zhaoyangchao@smail.nju.edu.cn) received his B.E. degree from the School of Electronic Science and Engineering, Nanjing University, China in 2017. He is pursuing his Master's degree at the Vision Lab, Nanjing University. His research focuses on video streaming, video rate model, reinforcement learning and deep learning.

Zhan Ma (mazhan@nju.edu.cn) received his B.S. and M.S. degrees from Huazhong University of Science and Technology, Wuhan, China, in 2004 and 2006, respectively, and his Ph.D. degree from New York University in 2011. He is now on the faculty of Nanjing University. From 2011 to 2014, he was with Samsung Research America and Futurewei. His research focuses on video compression, gigapixel streaming, and multi-spectral signal processing.

Yiling Xu (yl.xu@sjtu.edu.cn) received the B.S., M.S. and Ph.D. degrees from the University of Electronic Science and Technology of China, China, in 1999, 2001, and 2004 respectively. She is a full researcher with the School of Electronic Information and Elecronic Engineering, Shanghai Jiaotong University, Shanghai, 200145, China. From 2004 to 2013, she was with the Multimedia Communication Research Institute of Samsung Electronics Inc, Korea. Her research interest is 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.

Haojun Huang (hhj0704@hotmail.com) is a professor in Networking Engineering with the College of Computer at China University of Geosciences. He received his B.S. degree from the School of Computer Science and Technology, Wuhan University of Technology in 2005, and his Ph.D. degree from the School of Communication and Information Engineering, University of Electronic Science and Technology of China in 2012. He worked as a postdoctor at the Research Institute of Information Technology at Tsinghua University from 2012 to 2015, and a lecturer at Wuhan University from 2015 to 2017. His current research interests include wireless communication, ad hoc networks, big data, and software-defined networking.

Hao Yin (h-yin@mail.tsinghua.edu.cn) is a professor with the Research Institute of Information Technology (RIIT) at Tsinghua University. He received the B.S., M.E., and Ph.D. degrees from Huazhong University of Science and Technology, Wuhan, China, in 1996, 1999, and 2002, respectively, all in electrical engineering. He was elected as the New Century Excellent Talent of the Chinese Ministry of Education in 2009, and won the Chinese National Science Foundation for Excellent Young Scholars in 2012. His research interests span broad aspects of multimedia communication and computer networks.

Dapeng Oliver Wu [S'98, M'04, SM'06, F'13] (dpwu@ieee.org) is a professor with the Department of Electrical and Computer Engineering, University of Florida, FL, USA. He received his Ph.D. in electrical and computer engineering from Carnegie Mellon University, Pittsburgh, PA, in 2003. His research interests are in the areas of networking, communications, signal processing, computer vision, machine learning, smart grid, and information and network security. Currently, he serves as Editor in Chief of *IEEE Transactions on Network Science and Engineering*.

A key question for crowdsourced edge networks is how to stimulate users to contribute their devices to the edge network. A possible mechanism is that the users can play games for free or even make money if they contribute their devices. Moreover, the free time or the revenue is based on the amount of contributed resources and duration time.