Delay-Rate-Distortion Optimization for Cloud Gaming with Hybrid Streaming

Xiaoming Nan, Xun Guo, Yan Lu, Yifeng He, Ling Guan, Fellow, IEEE, Shipeng Li, Fellow, IEEE, and Baining Guo, Fellow, IEEE.

Abstract—Cloud gaming, as the emerging game service, has attracted significant attentions. However, traditional video streaming approach suffers from high bandwidth consumption, and traditional graphics streaming approach requires a long initial period to download game models. In this paper, we propose a novel hybrid streaming framework, jointly applying video streaming and graphics streaming to provide a high quality gaming experience. In the proposed framework, cloud servers not only transmit the encoded video frames, but also progressively transmit the graphics data, which are used to render a game frame to provide an additional reference to video encoder. Based on the proposed framework, we investigate the delay-Rate-Distortion (d-R-D) optimization problem, where 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. Experimental results demonstrate that the proposed hybrid streaming can achieve the lowest distortion under the constraints of bandwidth and response delay, compared to the traditional video streaming and graphics streaming.

Index Terms—Cloud gaming, delay-rate-distortion optimization, rate allocation, video encoding, progressive mesh

I. INTRODUCTION

The number of worldwide video game players has reached 1.78 billion in 2014 [1]. With the popularity of mobile devices, game players strongly desire the on-demand gaming experience. However, mobile devices are limited in memory and computation capacities. Video games may suffer from intolerable quality on thin devices. The emergence of cloud gaming effectively fills this gap by providing gaming as a service (GaaS) [2].

Currently, cloud gaming systems are developed by two different approaches: video streaming (also known as remote rendering GaaS [2]), and graphics streaming (also known as local rendering GaaS [2]). The difference between these two approaches is if the rendering is executed on cloud servers or on client devices. With video streaming, the cloud server renders the game scenes and streams the encoded game frames to client devices. The benefit is to reduce the hardware requirements at terminals [2]. OnLive [3] and GaiKai [4] use video streaming in their services. Huang et al. [5], [6] presented the first open source cloud gaming system using video streaming. Cai et al. [2] presented the remote rendering GaaS architecture and analyzed the related research issues. Cai et al. [7] explored the correlation of game videos for different players in the same game scene.

On the other hand, the hardware upgrade, graphics streaming is proposed as an alternative solution. In graphics streaming, game models and textures are downloaded to client devices and needs an initial buffering delay. Kalydo [8] adopts this approach. Nave et al. [9] presented a distributed gaming project with 3D streaming protocol. Cai et al. investigated cloud gaming decomposition problem in [10].

However, both streaming approaches have limitations. Video streaming consumes high bandwidth. In OnLive [3], the minimum bandwidth requirement is 2 Megabits per second (Mbps) and a higher resolution performance, like 720P (1280×720), requires more than 5 Mbps. Additionally, it is a challenge to meet response delay requirement in video streaming, since each frame has to go through the whole pipeline of rendering, capturing, encoding, transmission, decoding, and display [5]. Compared to video streaming, graphics streaming has a lower response delay. But in graphics streaming, the client has to receive the required graphics data prior to rendering the game scene, which is intolerable in the middle of a gaming session.

Besides streaming approaches, adaptive rate control is an important research topic in cloud gaming [11], [12], [13], [14], [15], [16], [17]. One research direction is to adaptively adjust encoding rate according to the significance of game objects. Shirmohammadi [11] presented a context aware approach to determine how important a game object is by building an importance matrix. Ahmadi et al. introduced a game attention model in [13] for content adaption in encoding game frames. Wang et al. [14], [15], [16] proposed rendering adaption schemes, where the bit rates were adjusted by dynamically varying the rendering parameters.

Another research direction is to introduce additional reference picture to save the encoding bit rate. Yea et al. [18] proposed to use the auxiliary information, like depth data, to generate the synthetic view, which provided a synthesized reference picture to reduce the bits needed for multiview video encoding. Zou et al. [19] proposed an in-loop view synthesis framework, where a view synthesis prediction was included in the skip candidate list in encoder.

Layered coding, as a new approach to reduce bit rate,
is studied in [20], [21], [22]. Chuah et al. [20], [21], [22] proposed a layered coding approach for mobile cloud gaming. In this approach, a game frame was separated as base layer and enhancement layer, where the base layer image was rendered on client device, while the enhancement layer image and information were encoded and transmitted by cloud server. Compared with [20], [21], the proposed work is different in the following senses. 1) Objective is different. The objective in the proposed work is to minimize the overall distortion during cloud gaming, while the objective in [20], [21] is to minimize the Shannon entropy of the enhancement layer and the rendering complexity of base layer. 2) Constraints are different. The proposed work is under the response delay and bandwidth constraints, while [20], [21] do not explicitly formulate these constraints. 3) Focus is different. The proposed work focuses on rate allocation in hybrid streaming, while [20], [21] focuses on the base layer rendering.

Besides the rate control, the response delay is an important quality of service (QoS) metric. Chen et al. [23], [24] studied the response delays in commercial cloud gaming systems. Miao et al. [25] proposed a collaborative rendering framework, where the local rendering is applied in order to conceal the interaction delay. Hariri et al. [26] proposed a statistical network traffic model based on the Hierarchical Hidden Markov Model (HHMM) to acquire the packet level statistics for first person shooter (FPS) games.

Existing cloud gaming systems are built either on video streaming or graphics streaming. The video streaming consumes high bandwidth and is sensitive to delay, while the graphics streaming requires buffering delay when players start a new game or move into a new scene. We propose a novel hybrid streaming framework. Based on the framework, we study the delay-Rate-Distortion problem to minimize the overall distortion. The main contributions are summarized as follows:

- We propose a hybrid streaming framework by jointly using video streaming and graphics streaming. During game playing, the server not only transmits the encoded video frame, but also sends the graphics data. The received graphics data are used to render a game frame, which provides an additional reference for video encoder. When encoding a captured game frame, the encoder will choose the reference frame with a lower residual error, from the previous frame and the rendered frame. As the accumulation of graphics data, prediction from the rendered frame will have a lower residual than the prediction from the previous frame, leading to a lower encoding bit rate.
- Based on the proposed framework, we investigate the delay-Rate-Distortion (d-R-D) optimization problem. We also develop an efficient rate allocation algorithm, as a practical solution to the d-R-D optimization problem, to allocate source rates to the video stream and the graphics stream. Experimental results demonstrate that the proposed hybrid streaming can effectively allocate rates to achieve a minimal distortion under the constraints of bandwidth and response delay.

The remainder of this paper is organized as follows. Sec. II presents the proposed hybrid streaming framework. Based on the framework, we study the d-R-D optimization problem in Sec. III and present extensive performance evaluations in Sec. IV. Finally, we conclude the paper in Sec. V.

II. HYBRID STREAMING FRAMEWORK FOR CLOUD GAMING

The proposed hybrid streaming framework is illustrated in Fig. 1(a). During game playing, user’s inputs are encoded and transmitted to cloud. By replaying user’s inputs, cloud server acquires user’s status, like position, viewpoint, and movement. Once a game frame is rendered, the server will capture the game frame and encode it by video encoder.

We introduce two functionally identical sync buffers at both cloud and client sides. Cloud server not only transmits the encoded video frame, but also sends the graphics data, including geometry meshes and textures. The received graphics data will be updated at the sync buffers and used to render a graphics frame. The rendered graphics frame provides an additional reference to video encoder. The reference frame with a lower residual will be used to reduce encoding bit rate. The proposed rate allocation algorithm dynamically allocates source rates to video stream and graphics stream to achieve the minimal overall distortion.

After the required game data have been transmitted to client, cloud server will skip video encoding. Once a skip signal is received at client, a switch will directly display the rendered graphics frame. In such a case, the game rendering is transferred from cloud server to client device, which reduces bandwidth consumption. To give a clear view, we also present the cloud gaming frameworks with the video streaming and the graphics streaming in Fig. 1(b) and Fig. 1(c), respectively.

By jointly using video streaming and graphics streaming, the proposed framework has the following advantages.

- The proposed framework eliminates the initial buffering delay compared to the traditional graphics streaming approach. When users move into a new game scene, the sync buffers have not received enough game data. The prediction from the previous frame generates a lower residual. Thus, cloud server will select the previous frame as reference, which is illustrated in Fig. 2(a). By streaming the encoded video frames, users can play the game immediately.
- The proposed framework reduces the overall bit rates compared to the traditional video streaming approach. As game playing, the graphics buffers keep accumulating game data. Once the sufficient game data have been received, prediction from the rendered frame generates a lower residual than the prediction from the previous frame, which is illustrated in Fig. 2(b). In video encoding, lower residuals lead to lower encoding bit rates. The received game data will be kept at server and client sides. If the player plays the same game again, the received game data will enable a quick start.
- The proposed framework provides a way to collaboratively utilize the rendering power at cloud and client. If
the cloud server detects that the client device has limited rendering capability, the cloud server will not transmit any game data and the video streaming will be performed. In such a scenario, the rendering load is solely taken by the cloud server. On the other hand, if the client devices, like laptop, PlayStation, or Xbox, have powerful rendering capability, the cloud server will transmit game data to client for local rendering. In this circumstance, the rendering workload is first taken by cloud server when the cloud server detects that the client device has limited rendering capability, the cloud server will not transmit any game data and the video streaming will be performed. Moreover, the simplified mesh can be reversed by the vertex split. As illustrated in Fig. 3, the vertex split operation adds a new vertex \( V_s \) near the vertex \( V_t \) and thus two new faces \( \{ V_s, V_t, V_i \} \) and \( \{ V_s, V_t, V_i \} \) can be constructed. Fig. 4(a) shows an equipped Orc warrior from WoW [28]. The initial mesh and the base mesh of the Orc warrior are shown in Fig. 4(b) and Fig. 4(c), respectively.

Besides the progressive mesh, we also apply the adaptive scalable texture compression (ASTC) [29] to compress the textures. ASTC is a block-based fixed-rate lossy texture compression algorithm. In ASTC, the compressed bit rates are flexible from 8 bits per pixel (bpp) to 0.89 bpp. In the proposed framework, we compress textures at two different bit rates,
where the high rate texture is compressed at 8 bpp and the low rate texture is compressed at 2 bpp. If the client bandwidth is not sufficient, the low rate textures will be transmitted to adapt to the limited bandwidth. Fig. 5(a) shows the ocular texture of the Orc warrior, and the corresponding textures compressed at 8 bpp and 2 bpp are shown in Fig. 5(b) and Fig. 5(c), respectively.

### III. Optimal Rate Allocation for Hybrid Streaming Framework

In this section, we investigate the source rate allocation problem in the proposed hybrid streaming framework. Specifically, we formulate the problem as a delay-Rate-Distortion (d-R-D) optimization, with respect to the allocated rates for video stream and graphics stream. Our objective is to achieve the minimal overall distortion under the limited bandwidth and the response delay constraints.

#### A. Problem Formulation

In the proposed hybrid streaming framework, if cloud servers allocate a higher source rate to video stream, it will take a longer time for the client to receive all the game data. On the other hand, if a higher rate is allocated to graphics stream, the visual quality in the video stream will be degraded. Therefore, we need to determine the optimal rate allocation for video stream and graphics stream.

Suppose the current time is $t$, and $R_v^t$ and $R_g^t$ are the allocated rates for video stream and graphics stream at $t$. We formulate the rate allocation problem as a d-R-D optimization. Let $D_{tot}$ be the total distortion at client side for the whole game playing period, $R_{tot}^t$ be the total bit rate allocation at $t$, $d_{tot}$ be the total response delay at $t$. To minimize the total distortion under the bandwidth and the response delay constraints, we can formulate the rate allocation problem as

$$\begin{align*}
\text{Minimize} & \quad D_{tot} \\
\text{subject to} & \quad R_{tot}^t \leq R_{max}, \\
& \quad d_{tot} \leq d_{max},
\end{align*}$$

where $R_{max}$ is the maximum bandwidth capacity and $d_{max}$ is the maximum tolerable response delay. In Eq. (1), the objective function represents the total distortion during game playing period. $R_v^t$ affects the quality of video streaming, while $R_g^t$ determines the length of the video streaming. In order to consider the current visual quality and the future video streaming length, we minimize the total distortion as the objective.

#### B. Gaming Process

We consider the whole gaming as a process. In order to model the total distortion, the process is partitioned into three disjoint periods: the past period before $t$, the current time at $t$, and the future period after $t$. Accordingly, $D_{tot}$ can be formulated as

$$D_{tot} = D_{pas}^t + D_{cur}^t + D_{fut}^t,$$

where $D_{pas}^t$ is the past distortion, $D_{cur}^t$ is the current distortion, and $D_{fut}^t$ is the future distortion. Since $D_{pas}^t$ occurs before $t$, it is a constant and not affected by the rate allocation at $t$. In the following, we will focus on the analysis of $D_{cur}^t$ and $D_{fut}^t$.

During game playing, cloud server progressively transmits game data. Let $\Omega^t$ denote the set of received game data until time $t$. Specially, $\Omega^0$ is the initial set. If the game is played for the first time, the graphics buffer is empty, i.e., $\Omega^0 = \emptyset$. As the game playing, the graphics data is received in the sync buffer, and accordingly, the set keeps increasing, i.e. $\Omega^t = \Omega^{t-1} \cup \{x\}^t$, where $\{x\}^t$ is the graphics data received in the interval between $t - 1$ and $t$. As more graphics data received, the quality of rendered frame will be refined in a further step.

Let $\Omega_0$ be the set of all required game data of the game scene, and $t_0$ be the expected time when $\Omega_0$ is reached. After $t_0$, all the required game data have been received at client, and the locally rendered frame will be displayed. Given $\Omega^t$ and $R_{G}^t$, the expected $t_0$ can be determined by $t_0 = t + \frac{\text{Bits}(\Omega_0 \setminus \Omega^t)}{R_{G}^t}$, where, $\Omega_0 \setminus \Omega^t = \{x \in \Omega_0 | x \notin \Omega^t\}$ represents the set of game data in $\Omega_0$ but not in $\Omega^t$ yet, $\text{Bits}(\cdot)$ is to acquire the number of bits, and $\frac{\text{Bits}(\Omega_0 \setminus \Omega^t)}{R_{G}^t}$ represents the expected remaining time to reach $\Omega_0$ under the current rate $R_{G}^t$. Specifically, if $\Omega_0 = \Omega^t$, we will have $\Omega_0 \setminus \Omega^t = \emptyset$ and thus $t_0 = t + \frac{\text{Bits}(\emptyset)}{R_{G}^t} = t$, which indicates that the locally rendered frame is already displayed at client.
Let $D_V$ be the distortion of the decoded video frame, and $D_G$ be the distortion of the locally rendered frame. If $t < t_0$, the distortion is determined by $D_V$; if $t \geq t_0$, the distortion is caused by $D_G$. Therefore, the current distortion $D_{cur}$ can be given by

$$D_{cur} = \begin{cases} D_V, & \text{if } t < t_0, \\ D_G, & \text{if } t \geq t_0, \end{cases}$$

(3)

where the close form functions of $D_V$ and $D_G$ will be investigated in Sec. III-C.

As analyzed before, the allocated rates $R_V^f$ and $R_G^f$ not only affect the current game quality, but also have effects on the future period. Although the accurate value of $D_{fut}^t$ is unknown at current time $t$, the expectation of $D_{fut}^t$ can be estimated.

Suppose $\hat{t}$ is the user’s playing time. In practice, the user may spend a different playing time at each time. Thus, $\hat{t}$ is a random variable. Suppose $t_V$ and $t_G$ are the remaining time of using the video frame and the rendered frame as game frame, respectively. According to the user’s playing time $\hat{t}$, $t_V$ and $t_G$ can be determined in two different situations. If the user’s playing time $\hat{t}$ is shorter than $t_0$, as illustrated in Fig. 6(a), the received game data are not sufficient to render a high quality frame even until the user finishes playing the game. Therefore, the video frame will be used during the gaming. On the other hand, if $\hat{t}$ is longer than $t_0$, as illustrated in Fig. 6(b), the video frame will be displayed from $t$ to $t_\hat{0}$, and the rendered frame will be displayed from $t_0$ to $\hat{t}$. Based on the above analysis, we can formulate $t_V$ and $t_G$ as

$$\begin{align*} t_V &= \hat{t} - t, \\
t_G &= 0, \\
t_V &= t_0 - t, \\
t_G &= t - t_0, \\
\end{align*}$$

if $\hat{t} < t_0$, if $\hat{t} \geq t_0$. (4)

The expected distortion $D_{fut}^t$ can be represented as the sum of the expected distortion of using the decoded video frame as game frame in the $t_V$ period and that of using the rendered frame as game frame in the $t_G$ period. $D_{fut}^t$ can therefore be formulated as

$$D_{fut}^t = \begin{cases} (t_V D_V + t_G D_G) \cdot f, & \text{if } \hat{t} < t_0, \\ ((t_0 - t) D_V + (\hat{t} - t_0) D_G) \cdot f, & \text{if } \hat{t} \geq t_0, \end{cases}$$

(5)

where $f$ is the frame rate.

C. Rate-Distortion (R-D) Analysis

In the proposed framework, the standard H.264 codec is modified by inserting the rendered graphics frame into the reference list. At client side, the rendered graphics frame is added into the reference list of decoder to acquire the correct picture. Since cloud gaming demands the real-time service, the coding structure is IPPPP coding mode.

According to studies [30], [31], the rate and the average distortion can be formulated as the functions of quantization step size. Suppose that the mean squared error (MSE) is used as the distortion measure. The relationship of the average distortion $D_V$ and the quantization step size $Q_{step}$ is given by [31] $D_V = MSE = \rho Q_{step} + \gamma$, where $\rho$ and $\gamma$ are the sequence parameters. In addition, the rate $R_V^f$ can be represented by [30] $R_V^f = \frac{\mu}{Q_{step}} + \nu$, where $\mu$ and $\nu$ are the sequence parameters. By eliminating $Q_{step}$, we can get the close form relationship between $D_V$ and $R_V^f$ as

$$D_V = \frac{\rho \mu}{R_V - \nu} + \gamma.$$  

(6)

The progressive mesh representation is a continuous and lossless representation [27], ASTC is a lossy texture compression. When $t \geq t_0$, $D_G$ is the distortion between the rendered image at client, which uses the received full geometry meshes and reconstructed textures, and the rendered image at cloud server, which uses the original mesh and textures. The distortion of ASTC is affected by the bit rate in the compressed texture [29]. Thus, $D_G$ can be approximated as a linear function of the distortion of ASTC, which is given by

$$D_G = m \sigma_{ASTC} + b.$$  

(7)

where $\sigma_{ASTC}$ is the distortion caused by ASTC, $m$ and $b$ are constant coefficients.

In order to verify Eq. (6) and Eq. (7), we capture two sequences from the online game WoW [28]: one is the Orc warrior sequence and the other is the Human warrior sequence. Fig. 7 shows fitting results of $D_V$ and $R_V^f$, where the distortion $D_V$ would decrease as the increase of $R_V^f$. We use Eq. (6) to fit the relationship between $D_V$ and $R_V^f$ and draw the fitted function curve in Fig. 7(a) and Fig. 7(b), where all $(R_V^f, D_V)$ points fall into the 95% confidence bounds of the fitting function. The narrow confidence bounds indicate that the rate-distortion model in Eq. (6) is reliable.
This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCSVT.2016.2595330, IEEE Transactions on Circuits and Systems for Video Technology

textures, while the frame rendered at client uses reconstructed textures. Fig. 8 presents data fitting results of $D_G$ and $\sigma_{ASTC}$. From Fig. 8, we can find $D_G$ can be approximated as a linear function of $\sigma_{ASTC}$, which justifies Eq. (7).

**D. Response Delay Analysis**

Besides distortion and rate, response delay is another important QoS metric in cloud gaming. The response delay is defined as the duration from the time when the user gives an input on the client device to the time when the resulting game frame is displayed to the user. There are four sources of delays in cloud gaming, including network delay, rendering delay, encoding delay, and playout delay.

- **Network delay** $d_{net}^t$: $d_{net}^t$ represents the network round-trip-time (RTT), which includes the transmission of user action to server and delivery of game frames to client.
- **Rendering delay** $d_{ren}^t$: $d_{ren}^t$ represents the delay for the game software at cloud server to process user input and render the next game frame.
- **Encoding delay** $d_{enc}^t$: $d_{enc}^t$ represents the time for the cloud server to capture the game frame and encode it into bit stream for the client.
- **Playout delay** $d_{pla}^t$: $d_{pla}^t$ represents the delay for the client to render graphics frame from sync buffer, decode and display the game frame.

Thus, the total response delay $d_{tot}$ can be formulated as the sum of the four components, which is given by

$$d_{tot}^t = d_{net}^t + d_{ren}^t + d_{enc}^t + d_{pla}^t.$$  

(8)

Among the four delays, the encoding delay $d_{enc}^t$ is the dominant component. The network delay $d_{net}^t$ can be measured by the Ping command, and $d_{ren}^t$ is game dependent, which can be acquired from the rendering frame rate at cloud server. If the cloud gaming system is proprietary and closed, $d_{ren}^t$ can also be approximated from the PC version of the game. Similar measurement techniques have been used in study [23].

In H.264/AVC encoding, the motion estimation is the most time consuming part, which takes more than 90% encoding time [32]. Similar to [32], [33], we approximate the encoding time by the motion estimation time (MET) in this work. When $t < t_\theta$, the encoding delay $d_{enc}^t$ can be given by [32]

$$d_{enc}^t \approx \frac{M(2\psi + 1)^2 \eta \cdot \alpha(R_V^t) \cdot c_0}{f_{CLK}}, \quad \text{if } t < t_\theta,$$

(9)

where $M$ is the number of macroblocks, $\psi$ is the search range, $\eta$ is the number of reference frames, $\alpha(R_V^t)$ is the ratios of the actual number of sum of absolute difference (SAD) operations to the theoretical total number of SAD operations, $c_0$ is the number of CPU cycles required for one SAD operation, and $f_{CLK}$ is the clock frequency of the CPU. As presented in [32], the function $\alpha(\cdot)$ in Eq. (9) can be fitted as an exponential function. Eq. (9) is the time for full search motion estimation. In practice, the fast search methods have lower delay. Eq. (9) is taken as a benchmark. If the encoding delay using full search can satisfy the delay constraint, the fast search will also satisfy the delay constraint.

In order to justify Eq. (9), we conduct experiments and draw the actual encoding delay $d_{enc}^t$ and the bit rate $R_V^t$ in Fig. 9. Fig. 9 indicates that the relationship can be well fitted by the function in Eq. (9), and all actual ($R_V^t$, $d_{enc}^t$) points are within the 95% confidence bounds.

Furthermore, when $t \geq t_\theta$, the client has received all the game data, and the locally rendered graphics frame is displayed. In such a case, the cloud server will skip encoding the frame. Based on the above analysis, the encoding delay is formulated as

$$d_{enc}^t \approx \begin{cases} 
\frac{M(2\psi + 1)^2 \eta \cdot \alpha(R_V^t) \cdot c_0}{f_{CLK}}, & \text{if } t < t_\theta, \\
0, & \text{if } t \geq t_\theta.
\end{cases}$$

(10)

As the last component, the playout delay $d_{pla}^t$ cannot be directly measured by the cloud server. When cloud server transmits the currently encoded frame to client, the decoding of the frame has not occurred yet. Inspired by study [23], we use the most recent playout delay to approximate the current playout delay at client. Specifically, the actual playout delay at time $t − 1$ is measured by the client and sent back to the cloud server.

With the delay components $d_{net}^t$, $d_{ren}^t$, $d_{enc}^t$, and $d_{pla}^t$, the cloud server can calculate the total response delay $d_{tot}$ by Eq. (8).
E. Rate Allocation Algorithm

Based on the above analysis, the optimal rate allocation problem in Eq. (1) can be formulated as

\[
\begin{align*}
\text{Minimize} \quad & D_{\text{tot}} = D_{\text{pax}} + D_{\text{cur}} + D_{\text{fut}} \\
\text{subject to} \quad & D_t = \begin{cases} 
D_V, & \text{if } t < t_0, \\
D_G, & \text{if } t \geq t_0,
\end{cases} \\
D_{\text{cur}}^{t+} = \begin{cases} 
(t-t)D_V \cdot f, & \text{if } t < t_0, \\
(t_0-t)D_V + (t-t_0)D_G \cdot f, & \text{if } t \geq t_0,
\end{cases} \\
R_{\text{tot}}^{t} \leq R_{\text{max}}, \\
R_{\text{tot}}^{t} = R_V^{t} + R_G^{t}, \\
d_{\text{enc}}^{t} \leq d_{\text{max}}, \\
d_{\text{enc}}^{t} = d_{\text{net}}^{t} + d_{\text{ren}}^{t} + d_{\text{enc}}^{t} + d_{\text{pla}}^{t}, \\
d_{\text{enc}}^{t} \leq f/t, \quad \text{if } t < t_0, \\
d_{\text{enc}}^{t} \approx \frac{\lambda(t-t_0)\eta(t-t_0)^2}{f(t-t_0)}, \quad \text{if } t \geq t_0.
\end{align*}
\]

In Eq. (11), the target of the rate allocation is to minimize the total distortion \(D_{\text{tot}}\) for the given bandwidth \(R_{\text{max}}\) and the given response delay \(d_{\text{max}}\), by optimizing source rates for video stream \(R_V^{t}\) and graphics stream \(R_G^{t}\).

According to the relationship of current time \(t\), time \(t_0\), and total playing time \(t\), the optimization problem Eq. (11) can be studied in the following cases.

Case 1: Given \(t_0 \leq t \leq \tilde{t}\), i.e., \(\Omega_\theta \subseteq \Omega\), the client has received all the required game data for the game scene. Therefore, the locally rendered frame from the graphics buffer has been used as the game frame at client. In such a case, we can get the following theorem that relates the optimal solution to the optimization problem in Eq. (11).

**Theorem 1.** Given \(t_0 \leq t \leq \tilde{t}\), the optimal solution \((R_V^{t}, R_G^{t})\) to the rate allocation problem in Eq. (11) is \(R_V^{t} = 0, R_G^{t} = 0\).

The proof of Theorem 1 is given in Appendix A. Intuitively, if the client can independently render the graphics frame, there is no need to stream the encoded video frame or graphics data to client. Accordingly, \(R_V^{t}\) and \(R_G^{t}\) will be reduced to 0.

Case 2: Given \(t \leq t < t_0\), the decoded video frame is displayed as game frame at t, and the expected game playing time \(\tilde{t}\) is not longer than \(t_0\). Thus, the client cannot receive all the required game data before the client finishes game playing. We then have the following theorem.

**Theorem 2.** Given \(t \leq \tilde{t} \leq t_0\), the optimal solution \((R_V^{t}, R_G^{t})\) to the rate allocation problem in Eq. (11) is \(R_V^{t} = \max\left(R_V^{t} | d_{\text{enc}}^{t} \leq \min(d_{\text{max}}^{t} - d_{\text{net}}^{t} - d_{\text{ren}}^{t} - d_{\text{pla}}^{t}, f), R_V^{t} \leq R_{\text{max}}^{t}\right), R_G^{t} = R_{\text{max}}^{t} - R_V^{t}.

The proof of Theorem 2 is given in Appendix B. The intuition of Theorem 2 is that, if the client cannot render a fine graphics frame until the end, the available source rate should be allocated to \(R_V^{t}\) as much as possible. A higher \(R_V^{t}\) can lead to a lower distortion. Thus, the optimal solution \(R_V^{t}\) will be the largest \(R_V^{t}\), which can simultaneously satisfy the constraints \(R_V^{t} \leq R_{\text{max}}^{t}\) and \(d_{\text{enc}}^{t} \leq \min(d_{\text{max}}^{t} - d_{\text{net}}^{t} - d_{\text{ren}}^{t} - d_{\text{pla}}^{t}, f)\).

Case 3: Given \(t < t_0 \leq \tilde{t}\), the client will display the decoded video frame as game frame from \(t\) to \(t_0\) and then display the rendered frame from \(t_0\) to \(\tilde{t}\). In this case, the distortion \(D_V\) is determined by \(R_V^{t}\), and the length of the video stream is determined by \(R_G^{t}\). Thus, it requires an optimal trade-off between \(R_V^{t}\) and \(R_G^{t}\) to minimize the total distortion. By analyzing Eq. (11), we can get the following Lemma.

**Lemma 1.** Given \(t < t_0 \leq \tilde{t}\), the optimal rate allocation problem in Eq. (11) is a convex optimization problem.

The proof of Lemma 1 is given in Appendix C. We use Lagrange multiplier method [34] to solve Eq. (11). Since \(d_{\text{enc}}^{t}\) is part of \(d_{\text{tot}}^{t}\), we substitute \(d_{\text{enc}}^{t}\) by \(d_{\text{enc}}^{t} \approx d_{\text{max}}^{t} - d_{\text{net}}^{t} - d_{\text{ren}}^{t} - d_{\text{pla}}^{t}\). Thus, the constraints \(d_{\text{enc}}^{t} \leq d_{\text{max}}^{t}\) and \(d_{\text{enc}}^{t} \leq 1/f\) can be replaced as \(d_{\text{enc}}^{t} \leq \bar{v}\), where \(\bar{v} = \min(d_{\text{max}}^{t} + d_{\text{net}}^{t} + d_{\text{ren}}^{t} + d_{\text{pla}}^{t})\). The Lagrange function of Eq. (11) is given by

\[
L(R_V^{t}, R_G^{t}, \lambda, \kappa) = D_{\text{tot}} + \lambda \cdot [R_V^{t} - R_{\text{max}}^{t}] + \kappa \cdot [d_{\text{enc}}^{t} - \bar{v}],
\]

(12)

where \(\lambda\) and \(\kappa\) are Lagrange multipliers associated with the inequality constraints. According to KKT conditions, the optimal primal solution \(R_V^{t}\) and \(R_G^{t}\) and dual solution \(\lambda^*\) and \(\kappa^*\) exist such that the following conditions can be satisfied simultaneously.

\[
\begin{align*}
\frac{\partial L(R_V^{t}, R_G^{t}, \lambda, \kappa)}{\partial R_V^{t}} & = 0, \\
\frac{\partial L(R_V^{t}, R_G^{t}, \lambda, \kappa)}{\partial R_G^{t}} & = 0, \\
\lambda^* \cdot [R_V^{t} - R_{\text{max}}^{t}] & = 0, \\
\kappa^* \cdot [d_{\text{enc}}^{t} - \bar{v}] & = 0.
\end{align*}
\]

(13a) (13b) (13c) (13d)

To solve the above equations, we use the Newton’s method [34] to iteratively find the solution. Suppose \(\delta^k = (R_V^{t+k+1}, R_G^{t+k+1}, \lambda^k, \kappa^k)\), \((\delta^k, \lambda^k+1, \kappa^k, \lambda^k+1 - \lambda^k, \kappa^k+1 - \kappa^k, \epsilon)\) can be solved from the following linear system [34].

\[
\begin{pmatrix}
\nabla^2 L & \nabla R_V^{t} & \nabla R_G^{t} \\
\nabla R_V^{t} & 0 & 0 \\
\nabla R_G^{t} & 0 & 0 \\
\end{pmatrix}
\begin{pmatrix}
\delta^k \\
\lambda^k+1 - \lambda^k \\
\kappa^k+1 - \kappa^k \\
\end{pmatrix}
= -\nabla D_{\text{tot}} + \frac{R_G^{t} - R_{\text{max}}^{t}}{d_{\text{tot}}^{t} - \bar{v}}
\]

(14)

With the solved \(\delta^k\), we can have \((R_V^{t+k+1}, R_G^{t+k+1}) = (R_V^{t}, R_G^{t}, R_{\text{max}}^{t} - R_{\text{tot}}^{t}, \lambda^k, \kappa^k, \epsilon)\). By repeating the above process, the optimal rate allocation \(R_V^{t}\) and \(R_G^{t}\) to Eq. (11) will be achieved until exceeding the maximum number of iterations or \(\delta^k \leq \epsilon\), where \(\epsilon\) is the vector of tolerable errors.

Based on the above discussions, we develop an efficient rate allocation algorithm, as illustrated in Algorithm 1. In Algorithm 1, the cloud server collects the required model parameters, investigates the relationships of \(t, t_0, \text{and } \tilde{t}\), and determines the optimal rate allocation \(R_V^{t}\) and \(R_G^{t}\). The rate allocation algorithm is conducted in every \(\Delta t\), which is set as one second in our work. In Algorithm 1, if \(\Omega_\theta \subseteq \Omega\), the rate allocation can be directly determined, and thus the computation complexity is \(O(1)\). If \(\Omega_\theta\) is not a subset of \(\Omega\), the rate allocation can be achieved by solving Eq. (14). According to [35], the number of iterations in Newton’s
Algorithm 1 Rate Allocation Algorithm for Hybrid Streaming Framework

1: Initialize model parameters $\rho$, $\mu$, $\nu$, $\gamma$, $\sigma_{ASTC}$, and the game data set $\Omega_o$.
2: Estimate the expected game playing time $\bar{t}$ according to the user’s historical statistics, and set $t = 0$.
3: Set $t_{\theta} = \infty$, if client device has no graphics rendering capacity.
4: while $t < \bar{t}$ do
5:   Check the game data set $\Omega^t$ in the sync buffer at cloud side.
6:   if $\Omega_\theta \subseteq \Omega^t$ then
7:      $R_{V*}^t = 0$, $R_{G*}^t = 0$,
8:   else
9:      Compute $\mathcal{D}_{tot}(R_{V*}^t, R_{G*}^t \mid \bar{t} \leq t_\theta)$, where $R_{V*}^t = \max \{ R_{V*}^t \mid d_{V*}^t \leq \min(d_{max} - d_{ren} - d_{pla} + 1/j), R_{V*}^t \leq R_{max} - R_{V*}^t \}$.
10:     Compute $\mathcal{D}_{tot}(R_{V*}^t, R_{G*}^t \mid \bar{t} > t_\theta)$, where $R_{V*}^t$ and $R_{G*}^t$ are acquired by repeatedly solve Eq. (14) until exceeding the maximum number of iterations or $\delta \leq \epsilon$.
11:    if $\mathcal{D}_{tot}(R_{V*}^t, R_{G*}^t \mid \bar{t} \leq t_\theta) \leq \mathcal{D}_{tot}(R_{V*}^t, R_{G*}^t \mid \bar{t} > t_\theta)$ then
12:       $R_{V*}^t = R_{V*}^t|_{\bar{t} \leq t_\theta}$, and $R_{G*}^t = R_{G*}^t|_{\bar{t} \leq t_\theta}$.
13:    else
14:       $R_{V*}^t = R_{V*}^t|_{\bar{t} > t_\theta}$, and $R_{G*}^t = R_{G*}^t|_{\bar{t} > t_\theta}$.
15:    end if
16: end if
17: $t = t + \Delta t$
18: end while

method in the worst case will be $O(\epsilon^{-2})$, where $\epsilon$ is the lowest tolerable error in $\epsilon$. Therefore, the computation complexity of Algorithm 1 is $O(\epsilon^{-2})$.

IV. PERFORMANCE EVALUATION

In this section, we evaluate the proposed hybrid streaming framework and the rate allocation algorithm.

A. Experiment Setup

In our experiments, a server with Intel i7 CPU 3.07GHz, 12G RAM, and NVIDIA GeForce GTX 660 is used as the cloud server, and a laptop with Intel i5 2.5GHz, 4G RAM, and Intel HD Graphics 4000 is used as the client device. We tested the proposed hybrid streaming on two games: the multiplayer online role-playing game WoW [28] and the FPS game Angry Bots [36]. WoW is a proprietary game. We exported the game models and textures of WoW and rendered the 3D game scenes. Two WoW game scenes are tested: the Orc warrior scene and the Human warrior scene. As for Angry Bots, the game assets and scripts are opened. Besides the game software, we use WoW Model Viewer [37] to export the WoW game assets, use OpenSceneGraph [38] to render game scenes, use OpenMesh [39] to generate progressive mesh representation, use Live555 [40] to set up RTSP server for video streaming, use the modified x264 as video codec, and use revised ffmpeg as the decoder at client. The screen resolution of all games is $1280 \times 720$ (720P). For fair comparisons, we record the avatar’s animation path. By replaying the animation event, the cloud server can control avatars to repeatedly move and perform exactly same actions. The test lasts for 5 minutes for each game scene. The game screenshots are shown in Fig. 10. During the evaluation, the captured game frame at cloud server is taken as the benchmark. We adopt PSNR as the quality metric by calculating the distortion between the displayed game frame at client and the benchmark at server. The maximal bit rate $R_{max}$ varies from 1 Mbps to 6 Mbps. When $R_{max}$ is under 2 Mbps (inclusive), the low rate textures are transmitted to client to save bandwidth. Otherwise, the high rate textures are used.

B. Comparison with Video Streaming Approach

We first compare the proposed hybrid streaming with the video streaming approach. The comparison of PSNR performance under different bandwidths is shown in Fig. 11. We can see that the proposed hybrid streaming framework can...
achieve a significantly higher average PSNR than the video streaming. The PSNR of the hybrid streaming increases by 8.35 dB, 6.54 dB, and 8.61 dB in average than the video streaming in Orc warrior, Human warrior, and Angry Bots scenes, respectively. In the hybrid streaming framework, the client can locally render a game frame, which has a better quality than the decoded video frame. Thus, the average PSNR is improved. To give a clear illustration, Fig. 12 and Fig. 13 compare the temporal PSNR performance. The values shown in these figures are the average PSNR of all the frames in each second. In Fig. 12 and Fig. 13, the initial PSNR of hybrid streaming is lower than the video streaming, because part of bit rates are allocated for delivering game data and accordingly the quality of video frame is affected. After the client has received all required game data, the PSNR of hybrid streaming increases. As shown in Fig. 12(c), the hybrid streaming has an initial PSNR of 29.62 dB, which is 5.86 dB lower than video streaming, but after \( t_\theta = 120 \) seconds, the hybrid streaming reaches 44.25 dB, which is 7.52 dB higher than video streaming. Overall, the average PSNR in hybrid streaming is 37.91 dB, which increases by 3.65 dB compared to video streaming in Fig. 12(c). Comparing Fig. 12 and Fig. 13, the cloud server spends less time on delivering game data when \( R_{max} \) is 6 Mbps, and as a result, \( t_\theta \) in Fig. 13 can be reached earlier than that in Fig. 12.

We also investigate the impact of response delay on the visual quality of cloud gaming. In the experiments, the network delay is set as 30 ms to simulate the real situation, the maximal bit rate \( R_{max} \) is 6 Mbps, and the response delay \( d_{max} \) is set as \{95, 120, 130, 140, 150, 200, 300\} ms. Fig. 14 shows the comparison of PSNR for different delays. From Fig. 14, it can be seen that the average PSNR increases with \( d_{max} \) but becomes flat for a large \( d_{max} \). This is because that a low \( d_{max} \) leads to a low \( d_{enc} \), which constrains the coding bit rate. When increasing \( d_{max} \), \( d_{enc} \) is increased, leading to a higher \( R'_{V} \) and a higher image quality. However, as the increase of \( d_{max} \), \( d_{enc} \) cannot be larger than \( 1/f \), and thus the image quality cannot be further improved. As shown in Fig. 14, when \( d_{max} \) reduces from 300 ms to 95 ms, the average PSNR in hybrid streaming drops 33.9% from 51.9 dB to 34.3 dB, while the average PSNR in video streaming drops 46.8% from 42.7 dB to 22.7 dB. Comparing the results, we can see that \( d_{max} \) has a less impact on the hybrid streaming framework. In hybrid streaming, when \( R_{V} \) is constrained by \( d_{max} \), the extra bit rates will be allocated to \( R_{G}' \), leading to a faster delivery of game data. Once receiving all game assets, the client can locally render the graphics frame. Furthermore, given \( d_{max} = 150 \) ms and \( R_{max} = 6 \) Mbps, we compare the PSNR under different network delays in Fig. 15. It can be seen from Fig. 15 that the average PSNR decreases with the increase of network delay, and the reduction of PSNR in the hybrid streaming is lower than the video streaming. The reason is that a high PSNR of locally rendered frame is not affected by the network delay. As shown in Fig. 15(a), when the network delay \( d_{net} = 80 \) ms, the hybrid streaming can achieve the average PSNR of 49.41 dB, which represents a 19.66 dB improvement over the video streaming.

Next, we compare the overall transmission data size in the hybrid streaming and the video streaming. Fig. 16 presents the overall data size with the \( R_{max} \) at 2 Mbps and 6 Mbps. It can be seen that the hybrid streaming can effectively save the overall transmission data size. Compared to the video streaming, the hybrid streaming in Fig. 16(c) reduces the overall transmission data size by 60.55% with \( R_{max} = 2 \) Mbps and by 85.08% with \( R_{max} = 6 \) Mbps. Moreover, we observe that the overall transmission in hybrid streaming at 6 Mbps is almost same as that at 2 Mbps. This is because that a higher \( R_{max} \) can allocate a higher rate for graphics stream \( R_{G}' \) and accordingly reduce the length of video streaming period, leading to a lower overall data size.
This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCSVT.2016.2595330, IEEE Transactions on Circuits and Systems for Video Technology

Fig. 14. Comparison of the PSNR performance for different delays when $R_{\text{max}} = 6$ Mbps: (a) Orc warrior scene, (b) Human warrior scene, and (c) Angry Bots scene.

Fig. 15. Comparison of the PSNR performance under different network delays when $d_{\text{max}} = 150$ ms and $R_{\text{max}} = 6$ Mbps: (a) Orc warrior scene, (b) Human warrior scene, and (c) Angry Bots scene.

C. Comparison with Graphics Streaming Approach

In this subsection, we compare the proposed hybrid streaming framework with the graphics streaming approach. In graphics streaming, the game assets are transmitted to client. After receiving all the necessary assets, the client can start rendering the game frame. For fair comparison, we use the same textures and game models in the hybrid streaming and graphics streaming. Fig. 17 compares the PSNR performance under different bandwidths. As shown in Fig. 17, the hybrid streaming can achieve a higher PSNR than the graphics streaming under the same bandwidth constraint. The PSNR of hybrid streaming increases by 3.31 dB, 3.68 dB, and 4.40 dB in average compared to the graphics streaming for the Orc warrior, Human warrior, and Angry Bots scenes, respectively. The reason is that graphics streaming takes a buffering period to collect all the required game assets. In this period, the client cannot render a high quality frame and has a very low PSNR. As a result, the overall quality is degraded. Compared to the graphics streaming, the proposed framework applies the streamed video frame in the initial period, and accordingly has a higher initial PSNR. To view the PSNR performance in more detail, Fig. 18 and Fig. 19 illustrate the temporal PSNR with the fixed $R_{\text{max}}$ at 2 Mbps and 6 Mbps, respectively. From Fig. 18 and Fig. 19, we can see that graphics streaming has a relatively low PSNR at the beginning, due to the lack of game

Fig. 16. Comparison of the overall data size when $R_{\text{max}}$ is fixed at 2 Mbps and 6 Mbps: (a) Orc warrior scene, (b) Human warrior scene, and (c) Angry Bots scene.

Fig. 17. Comparison of the PSNR performance under different bandwidths: (a) Orc warrior scene, (b) Human warrior scene, and (c) Angry Bots scene.
data. In contrast, the proposed hybrid streaming achieves a higher PSNR in this period. As a cost, hybrid streaming takes a longer period to transmit all the game data to client. We also compare the overall data size in the hybrid streaming and the graphics streaming in Fig. 20, the hybrid streaming consumes a higher transmission data size, due to the video streaming in the beginning period.

Fig. 18. Comparison of the temporal PSNR performance when $R_{max} = 2$ Mbps: (a) Orc warrior scene, (b) Human warrior scene, and (c) Angry Bots scene.

Fig. 19. Comparison of the temporal PSNR performance when $R_{max} = 6$ Mbps: (a) Orc warrior scene, (b) Human warrior scene, and (c) Angry Bots scene.

Fig. 20. Comparison of the overall data size when $R_{max}$ is fixed at 2 Mbps and 6 Mbps: (a) Orc warrior scene, (b) Human warrior scene, and (c) Angry Bots scene.

Fig. 21. Comparison of the average PSNR under different bandwidths with $d_{max}$ fixed at 150 ms: (a) Orc warrior scene, (b) Human warrior scene, and (c) Angry Bots scene.

D. Evaluation of Rate Allocation Algorithm

The proposed rate allocation algorithm is evaluated in this subsection. In the experiments, we select different rate allocation pairs of $(R_V^t, R_G^t)$, and compare the PSNR performance. The selected rate allocation pairs include: 1) the optimal allocation, which is acquired by solving the optimization problem in Eq. (11); 2) $(R_V^t, R_G^t) = (0.7R_{max}, 0.3R_{max})$ represents rate allocation with a high rate for video stream; 3) $(R_V^t, R_G^t) = (0.5R_{max}, 0.5R_{max})$ represents even rate allocation; and 4) $(R_V^t, R_G^t) = (0.3R_{max}, 0.7R_{max})$ represents rate allocation with a high rate for graphics stream. Special cases, like $(R_V^t, R_G^t) = (R_{max}, 0)$ and
Fig. 22. Comparison of the temporal PSNR with \( R_{\text{max}} \) fixed at 6 Mbps: (a) Orc warrior scene, (b) Human warrior scene, and (c) Angry Bots scene.

Fig. 23. Game screenshots of Doom3.

\((R_V^1, R_G^1) = (0, R_{\text{max}})\), which represent the video streaming and graphics streaming respectively, have been compared before and hence are not included in this evaluation. Fig. 21 compares the average PSNR under different bandwidths when the response delay \( d_{\text{max}} \) is fixed at 150 ms. It can be seen from Fig. 21 that for a given response delay \( d_{\text{max}} \), the proposed rate allocation algorithm can achieve a higher average PSNR compared to the other alternative rate allocation pairs. From Fig. 21, we can also find that the rate allocation \((0.7R_{\text{max}}, 0.3R_{\text{max}})\) achieves a high PSNR when \( R_{\text{max}} \) is lower than 2 Mbps, but a low PSNR when \( R_{\text{max}} \) is 6 Mbps.

The reason is that the PSNR in video encoding is not linearly increased with \( R_V^1 \). Therefore, when \( R_V^1 \) is relatively high, the increment of \( R_V^1 \) cannot get an equivalent return on the PSNR performance. Fig. 22 illustrates the temporal PSNR when \( R_{\text{max}} = 6 \) Mbps. Compared to the other rate allocations, the proposed rate allocation algorithm can achieve an optimal balance between the quality of video streaming and the length of video streaming.

**E. Evaluation of Scene Changes**

In this subsection, we evaluate the proposed hybrid streaming on scene changes. The evaluation is conducted on Doom3 for 30 minutes. Doom3 is a first person shooter video game [41]. In our experiments, the player starts in scene 1, moves to scene 2 and scene 3, and finally finishes gaming in scene 4. The game screenshots are shown in Fig. 23.

We compare the PSNR performance under different bandwidths between the hybrid streaming with video streaming and graphics streaming, respectively. The comparison result is shown in Fig. 24. As shown in Fig. 24, the proposed hybrid streaming achieves a higher PSNR performance compared to the video streaming and the graphics streaming. We also compare the temporal PSNR performance when \( R_{\text{max}} = 2 \) Mbps in Fig. 25. In Fig. 25(a), the hybrid streaming has a lower PSNR when the player moves to a new game scene. According to previous study [42] on cloud gaming, a PSNR of 30 dB and above is considered good quality for video streaming based cloud gaming service. Although the initial PSNR is low in hybrid streaming, the PSNR in hybrid streaming is still either higher than or close to 30 dB. After receiving required graphics data, the PSNR in hybrid streaming is significantly improved. In Fig. 25(b), the graphics streaming needs to download all required game objects before rendering the new scene. Thus, there is an initial buffering time when switching between scenes. Although it is possible to send the game objects before players entering into a new scene, it is still challenging to accurately predict the next game scene that the player will move to. In hybrid streaming, when the player moves to a new scene, the new game objects have not been received in the sync buffer. By using the previous frame as reference, the hybrid streaming switches back to video streaming mode. Therefore, the player is able to play the game immediately.

**F. Evaluation of Delays**

We evaluate the delay components in this subsection. The detailed comparison is shown in Table I. The delays are acquired from real measurement. Since the server and the client are connected in a local area network (LAN), \( d_{\text{max}}^n \) is less than 1 ms. The encoding delay \( d_{\text{enc}} \) in hybrid streaming occurs before \( t_g \). In the tests, \( t_g = 20 \) seconds on Orc
warrior scene and \( t_0 = 38 \) seconds on Angry Bots scene. After \( t_0 \), the local rendering is performed, and thus \( d_\text{enc}^t \) is 0. The hybrid streaming introduces a rendering step at client. Thus, the playout delay is higher than that in video streaming. Compared with the video streaming, the hybrid streaming has a comparable delay before \( t_0 \). After receiving all required game objects, the hybrid streaming can achieve a lower delay.

G. Evaluation under Varying Bandwidth

In this subsection, we performed experiments to evaluate the proposed framework under varying bandwidth conditions. The proposed d-R-D optimization is performed in every second to deal with network dynamics. The varying bandwidth \( R_{\text{max}} \) is acquired from real measurement. We conducted each measurement in every 10 seconds. The varying bandwidth is shown in Fig. 26(a). Fig. 26(b) compares the PSNR between the proposed hybrid streaming and video streaming under the varying bandwidth. Observing Fig. 26(b), we can find that the PSNR of video streaming and hybrid streaming varies with varying bandwidth. Observing Fig. 26(b), we can find that the PSNR of video streaming and hybrid streaming varies with varying bandwidth. Thus, the proposed framework under varying bandwidth conditions.

V. CONCLUSIONS

In this paper, we propose a novel hybrid streaming framework, which jointly applies video streaming and graphics streaming. Compared with the traditional video streaming, the proposed framework reduces the overall bit rates and achieves a higher overall image quality. Compared with the traditional graphics streaming, the proposed framework enables users to play a game immediately without the initial waiting time. Moreover, we study the d-R-D optimization for the proposed framework. Experimental results demonstrate that the proposed hybrid streaming framework can optimally allocate source rates to achieve the minimal distortion under the bandwidth and response delay constraints.

APPENDIX

A. Proof of Theorem 1

Proof: Given \( t_0 \leq t \leq \tilde{t} \), i.e. \( \Omega_0 \subseteq \Omega^t \), the objective in Eq. (11) is converted to \( D_\text{tot} = D_\text{pas}^t + D_\text{g} + (\tilde{t} - t_0)D_\text{G} \cdot f \), where \( D_\text{tot} \) is only determined by \( D_\text{G} \) and has nothing with \( D_\text{V} \). Since \( R_v^t \) has no effect on \( D_\text{tot} \), \( R_v^t \) will be reduced to 0. Since \( t_0 \leq t \), all required game data have been received by client. Thus, \( R_v^t \) will be reduced to 0. Theorem 1 is proved. ■

B. Proof of Theorem 2

Proof: Given \( t \leq \tilde{t} \leq t_0 \), the total distortion \( D_\text{tot} \) is determined by \( D_\text{V} \) and has no relationship with \( D_\text{G} \). Thus, \( R_v^t \) will not affect the total distortion. As analyzed in Sec. III-C, \( D_\text{V} \) is a monotonically decreasing function of \( R_v^t \). Therefore, the optimal allocation \( R_v^* \) is the largest \( R_v^t \) which simultaneously meets bandwidth and delay constraints, i.e. \( R_v^* = \max \{ R_v^t \mid d_\text{enc}^t \leq \min (d_\text{max} - d_\text{net}^t - d_\text{enc}^t - d_\text{pla}^t, 1/f) \}, R_v^t \leq R_{\text{max}} \}. \) Therefore, \( R_v^* \) achieves a lower overall image quality. Compared with the proposed framework reduces the overall bit rates and waiting time. Moreover, we study the d-R-D optimization allocation and has nothing with \( \Omega^t \).

\begin{align}
\n\n\n\end{align}

in which \( \alpha(R_v^t) \) is an exponential function of \( R_v^t \), as shown in Sec. III-D. In Eq. 15, the bandwidth constraint, the response delay constraint, and the encoding delay constraint are all convex. To prove the convexity of the objective function, we derive the Hessian matrix of \( D_\text{tot} \) as following.

\begin{align}
\n\n\n\end{align}

We can find that the Hessian matrix \( \nabla^2 D_\text{tot}(R_v^t, R_v^t) \) is positive semidefinite, and thus the objective function in Eq. (15) is convex [34]. Based on the above analysis, given \( t < t_0 < \tilde{t} \), the optimization problem in Eq. (11) is a convex optimization problem. The Lemma 1 is proved. ■

ACKNOWLEDGMENT

This work is partially supported by the Canada Research Chair Program and National Natural Science Foundation of China Key International Collaboration Program (No. 61210005).

REFERENCES

TABLE I

<table>
<thead>
<tr>
<th></th>
<th>$d_{\text{ref}}^n$ (ms)</th>
<th>$d_{\text{ren}}^n$ (ms)</th>
<th>$d_{\text{ref}}^p$ (ms)</th>
<th>$d_{\text{rep}}^p$ (ms)</th>
<th>Weighted average delay (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Orc warrior in</strong>&lt;br&gt;<strong>hybrid streaming</strong></td>
<td>$&lt; 1$</td>
<td>17.61</td>
<td>$&lt; 1$</td>
<td>39.46 (before $t_p$)</td>
<td>46.38</td>
</tr>
<tr>
<td><strong>Angry Bots in</strong>&lt;br&gt;<strong>hybrid streaming</strong></td>
<td>$&lt; 1$</td>
<td>16.18</td>
<td>$&lt; 1$</td>
<td>40.80 (before $t_p$)</td>
<td>50.67</td>
</tr>
<tr>
<td><strong>Orc warrior in</strong>&lt;br&gt;<strong>video streaming</strong></td>
<td>$&lt; 1$</td>
<td>14.48</td>
<td>43.54</td>
<td>32.3</td>
<td>90.32</td>
</tr>
<tr>
<td><strong>Angry Bots in</strong>&lt;br&gt;<strong>video streaming</strong></td>
<td>$&lt; 1$</td>
<td>13.29</td>
<td>45.67</td>
<td>35.71</td>
<td>94.67</td>
</tr>
</tbody>
</table>


