Enhancing Quality of Experience for Collaborative Virtual Reality with Commodity Mobile Devices

Jiangong Chen* Feng Qian † Bin Li*
*Department of EE, The Pennsylvania State University - University Park, Pennsylvania, USA
†Department of CSE, University of Minnesota - Twin Cities, Minneapolis, Minnesota, USA
{jiangong, binli}@psu.edu, fengqian@umn.edu

***Abstract*—Virtual Reality (VR), together with the network infrastructure, can provide an interactive and immersive experience for multiple users simultaneously and thus enables collaborative VR applications (e.g., VR-based classroom). However, the satisfactory user experience requires not only high-resolution panoramic image rendering but also extremely low latency and seamless user experience. Besides, the competition for limited network resources (e.g., multiple users share the total limited bandwidth) poses a significant challenge to collaborative user experience, in particular under the wireless network with time-varying capacities. While existing works have tackled some of these challenges, a principled design considering all those factors is still missing. In this paper, we formulate a combinatorial optimization problem to maximize the Quality of Experience (QoE), defined as the linear combination of the quality, the average VR content delivery delay, and variance of the quality over a finite time horizon. In particular, we incorporate the influence of imperfect motion prediction when considering the quality of the perceived contents. However, the optimal solution to this problem can not be implemented in real-time since it relies on future decisions. Then, we decompose the optimization problem into a series of combinatorial optimization in each time slot and develop a low-complexity algorithm that can achieve at least $1/2$ of the optimal value. Despite this, the trace-based simulation results reveal that our algorithm performs very close to the optimal offline solution. Furthermore, we implement our proposed algorithm in a practical system with commercial mobile devices and demonstrate its superior performance over state-of-the-art algorithms. We open-source our implementations on https://github.com/SNeC-Lab-PSU/ICDCS-CollaborativeVR.**

I. Introduction

Virtual reality can provide an immersive and interactive experience for users and has become more and more popular with the rapid growth of 3D displaying and computer vision technologies. It together with the existing network infrastructure has spurred many fascinating collaborative VR applications such as VR-based education/training [1], VR-based collaborative art design [2], VR-based multi-user gaming [3], and social networks [4]. Indeed, in VR-based education, when a geography teacher introduces the galaxy to the students, with the help of VR technology, students can freely explore the universe and easily get the location and detailed information of each star. Moreover, they can interact with each other and discuss some particular interesting astronomy topics. Such a novel teaching environment will greatly improve learning efficiency by stimulating students' interest and providing an immersive learning experience.

In order to provide the best immersive user experience, the VR system should provide 1) high-resolution panoramic image rendering with a high frame rate: user wants to ensure a resolution with at least 2560×1440 pixels and have about 60 frames-per-second (FPS); 2) extremely low delay guarantees: the head motion-to-photon latency should be as low as possible, typically below 20ms for smooth movement and interaction; 3) seamless user experience: consistent VR image quality is required to avoid virtual reality sickness.

Existing mobile devices only support low-quality VR applications due to their constrained CPU/GPU capabilities. One of the most popular approaches to deal with such low capabilities on mobile devices is to offload compute-intensive rendering load to a powerful server, which streams the rendered frames to the mobile device through the wireless network (e.g., [5], [6]). Compared with existing commercial VR devices (e.g., Valve Index, Oculus Rift) that connect the devices to the server with wired cables, wireless streaming is more flexible and safe, resulting in a better user experience. However, supporting high-resolution VR applications in this manner would require hundreds of Mbps or even Gbps of bandwidth for each user, which cannot be supported by current commercial wireless network, let alone the collaborative VR with multiple users.

Fortunately, it is possible to deliver the partial panoramic content without affecting the user's visual perception if we can predict the user's 6-Degree-of-Freedom (DoF) motion (3 DoF for virtual location and the other 3 DoF for head orientation). As such, several existing works [7]–[12] have incorporated motion prediction into algorithm design. In particular, they split the whole panoramic images into many smaller blocks, named tiles, and combined motion prediction to stream part of those tiles to the users without affecting their visual perception. However, to the best of our knowledge, none of the existing work proposed a principled design of efficient rate allocation algorithms for collaborative VR applications with the goal of optimizing the quality of experience and deployed such algorithms into a practical system with commodity mobile devices. Although heuristic methods work well in those works, we believe it is far from trivial to develop a principled design that achieves better performance and guides us on adapting to different application scenarios.

We explicitly consider the collaborative VR application for

This research has been supported in part by NSF grants: CNS 1915122, CNS 2106090, CNS 2128489, CNS 2152658, and CNS 2152610.

multiple users, where various factors need to be considered regarding the users' Quality of Experience (QoE). Specifically, we define QoE as the linear combination of the quality of the perceived contents, the average VR content delivery delay, and variance of the perceptual quality for all users within a finite time horizon. Moreover, we consider the influence of imperfect motion prediction and incorporate it into our QoE formulation.

The challenges of the QoE optimization problem lie in that the QoE relies on future decisions, making the optimal solution non-implementable in real-time. One of the main contributions of this paper is to decompose the problem into a series of optimization problems and solve them independently. Besides, the discrete property of the quality levels further complicates the optimization problem since it becomes a Knapsack problem, which is well-known for its difficulty. In this paper, we take a principled and integrated approach to allocate quality levels for multiple users by incorporating motion prediction and tile partitioning to save the network bandwidth and to adapt to dynamic network conditions. The main contributions of this paper are summarized as follows:

- In Section II, we formulate a combinatorial optimization problem over a finite time horizon with the objective being the accumulative QoE that includes the average quality level considering the imperfect motion prediction, the average VR content delivery delay, and the variance of the quality level, subject to the available network throughput constraints.
- In Section III, realizing the challenge in obtaining the optimal solution and its physically non-implementability, we decompose it into a series of the combinatorial optimization problem in each time slot. Here, the combinatorial optimization problem is a knapsack problem with the concave objective function and convex constraints. We develop an efficient and low-complexity quality level allocation algorithm that yields at least $1/2$ of the optimal value.
- In Section IV, we perform a trace-based simulation to demonstrate the superior performance of our proposed algorithm over state-of-the-art algorithms. We observe that our proposed algorithm performs very close to the offline optimal solution and outperforms the state-of-the-art algorithms.
- In Section V and Section VI, we develop a real-world collaborative VR system to evaluate our proposed algorithm under two different experiment setups. The evaluation results show that our algorithm is quite robust to the dynamic network environment and outperforms existing algorithms. In particular, our algorithm gets an 81.9% improvement over the Firefly algorithm [8], and 12.1% improvement over the PAVQ algorithm [13] on the average QoE under the first experiment setup. Furthermore, our algorithm gets a 214.3% improvement over the PAVQ algorithm under the second experiment setup. We make the source code public for the benefit of the community.

II. System Model

We consider a collaborative VR system with N users that interact with each other via a wireless edge server, where the edge server performs rendering and transmits the rendered images to each user. We assume that the system operates in a time-slotted manner. The time slot duration is set to ensure that the rendered VR content can be delivered within one time slot and the average display rate is 60 frames-per-second (FPS) to meet the desired quality of experience (QoE). In addition to the frame rate, the content quality and its consistency as well as the content delivery delay significantly affect users' QoE.

We assume that each VR content can be encoded into L different quality levels. Let $\mathcal{Q} \triangleq \{1, 2, \ldots, L\}$ denote the set of quality levels, where a larger quality value corresponds to a better visual perception experience for users. Note that in practical scenarios, a larger quality level corresponds to a higher resolution or a smaller constant rate factor (CRF). In each time slot t, the edge server determines the quality level of the VR content for each user. We use $q_n(t) \in \mathcal{Q}$ to denote the quality level selected by the edge server for user n in time slot t. Each VR content has a panoramic view, but a user just needs to see about 20% of the panoramic view, which is known as *field of view* (FoV). Thus, if we could accurately predict the user's motion, we only require as low as 20% of the network bandwidth compared to the whole panoramic scene delivery. However, it is impossible to have perfect motion prediction. Nevertheless, it is possible to utilize various prediction algorithms to predict each user's motion with a high accuracy. As such, any existing motion prediction model can be applied to this paper to predict each user's 6-degree-of-freedom (DoF) motion (3 DoFs for virtual location and the other 3 for head orientation). To further handle the prediction error caused by the imperfect prediction, we deliver a portion that covers the FoV with some fixed margin.[1] To capture the impact of the imperfect prediction, we use $\mathbb{1}_n(t) = 1$ to denote whether the delivered portion covers the actual FoV (considering both virtual location and head orientation) in time slot t, and $\mathbb{1}_n(t) = 0$ otherwise. Hence, $q_n(t)\mathbb{1}_n(t)$ denotes the quality level of VR content that is successfully seen by user n in time slot t.

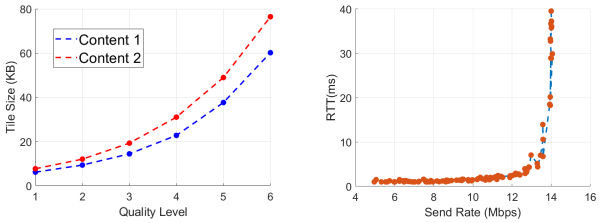
Besides the average perceptual quality of the VR contents, the QoE also accounts for the VR content consistency. A recent study (see [14]) reveals that compared with the displayed contents with relatively higher average quality but frequent stalls, a consistent display of contents with relatively lower average quality results in a higher QoE. As such, to characterize the VR content consistency, we use the variance $\sigma_n^2(T)$ of the quality level of VR content that is successfully viewed by user n within a finite time horizon T, i.e.,

$$\sigma_n^2(T) \triangleq \frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\left[\left(q_n(t)\mathbb{1}_n(t) - \frac{1}{T}\sum_{\tau=1}^{T} q_n(\tau)\mathbb{1}_n(\tau)\right)^2\right].$$

[1]The extended margin on FoV only helps in the prediction of 3 DoFs for head orientation. Although there are some other approaches to handle the prediction errors on virtual location, we have left them as future work.

The smaller the variance $\sigma_n^2(T)$, the better the seamless experience of user n. $\sigma_n^2(T)$ can be shown to be convex with respect to $(q_n(1), q_n(2), \cdots, q_n(T))$ (cf. [13, Lemma 1]).

The delay of VR content delivery from the edge server to each user also significantly affects the QoE performance. Such VR content delivery delay usually depends on the size of the VR content and the available network throughput, where the VR content with a higher quality level has a larger size. We use $d_n(r)$ to denote the average delivery delay from the edge server to user n when the VR content size is r. Let $f_c^R(q)$ be the function that maps the quality level q to the size of the VR content c.



(a) Tile size vs. quality level (b) RTT vs. send rate

Fig. 1: Convexity of functions $f_c^R(\cdot)$ and $d(\cdot)$

In Fig. 1a, we randomly select two different VR content and draw the tile size with different quality levels, where different quality levels refer to the tiles encoded with different Constant Rate Factor (CRF) values. A higher quality level results in a better visual perception of the content. We can observe from Fig. 1a that $f_c^R(q)$ is convex with respect to quality level q for each VR content. In Fig. 1b, we limit the throughput within 15 Mbps and continuously send data at a specific rate. At the same time, we start lots of ping processes in parallel to get the RTTs. We collect $100,000$ samples of RTTs to draw the result. We draw RTTs because it reflects the queueing delay, which is the main component of the delay under the local wireless network. Although the curve will shift with the variety of throughput limitations, the convexity always holds. As the sending rate is a linear function of the content size (since the time slot duration is fixed in our model), we can conclude that the average round trip time (i.e., $d_n(\cdot)$) is convex with respect to the content size r. As such, we assume that both functions $f_c^R(q)$ and $d_n(r)$ are convex and increasing, and hence the average delivery delay $d_n(f_{c(t)}^R(q_n(t)))$ is a convex function with respect to the quality level $q_n(t)$ when the VR content is $c(t)$ in time slot t.

We use $\text{QoE}_n(T)$ to denote the quality of experience (QoE) of user n within a finite time horizon T. Considering the tradeoff between all above components, we define the QoE as a linear combination of the quality of the perceived contents, the average VR content delivery delay, and variance of the perceptual quality, i.e.,

$$\text{QoE}_n(T) \triangleq \sum_{t=1}^{T} \left(\mathbb{E}\left[q_n(t)\mathbb{1}_n(t)\right] - \alpha \mathbb{E}[d_n(f_{c(t)}^R(q_n(t)))] - \beta \sigma_n^2(T) \right),$$

where both α and β are some positive constants that depend on the users' sensitivity on the content quality level, delivery delay, and the quality consistency. Particularly, a larger value of α is chosen for those applications which are more sensitive to the delay, like multi-user VR gaming. Similarly, we prefer a larger value of β when our model is applied to those applications requiring consistent content streaming like museum touring. Note that $\text{QoE}_n(T)$ is a concave function, since it is the weighted sum of a linear function and two concave functions.

Recall that the quality level $q_n(t)$ is the decision variable in our problem, where a higher quality level of the VR content requires a larger transmission rate. However, both the required transmission rates for each user and the server are limited by their respective available throughput. Let $B_n(t)$ and $B(t)$ be the available network throughput for user n and the server in time slot t, respectively. Note that we unify the units of content size $f_c^R(q)$ and the network throughput by fixing each time slot duration. Although both $B_n(t)$ and $B(t)$ have high temporal variability under the wireless network, they can be estimated at the beginning of time slot t. In this paper, we aim to maximize the total system QoE by allocating the quality level to each user while meeting the network throughput constraints. This can be achieved by solving the following optimization problem:

$$\max \quad \text{QoE}(T) \triangleq \sum_{n=1}^{N} \text{QoE}_n(T) \tag{1}$$

$$\text{s.t.} \quad \sum_{n=1}^{N} f_{c(t)}^R(q_n(t)) \leq B(t), \forall t \tag{2}$$

$$f_{c(t)}^R(q_n(t)) \leq B_n(t), \forall t, n. \tag{3}$$

However, the optimal solution to problem (1)-(3) is not physically implementable since it relies on future decisions on the quality levels. Moreover, since the decision variable (i.e., quality level) is discrete, the considered optimization problem has a combinatorial structure and thus is typically NP-hard. In the next section, we decompose this optimization problem into a sequence of combinatorial optimization problems and then develop an efficient quality allocation algorithm, which will be demonstrated to perform close to the optimal offline algorithm via trace-based simulations (cf. Section IV).

III. Algorithm Design and Analysis

In this section, we first decompose the original optimization problem (1)-(3) over a finite time horizon T into a sequence of combinatorial optimization problems, and then develop an efficient and low-complexity algorithm to solve the combinatorial optimization problem in each time slot.

The optimal solution to the problem (1)-(3) requires the full knowledge of available throughput for each user and the server and can be obtained via the dynamic programming approach. However, the obtained solution relies on future information and is not implementable in real-time. After carefully looking into the objective function (1), we find that the variance of the quality level of VR content successfully viewed by each user couples the quality level decisions over time. By leveraging variance iteration formula [15], we have

$$T\sigma_n^2(T) = \sum_{t=1}^{T} \frac{(t-1)\left(q_n(t)\mathbb{1}_n(t) - \overline{q}_n(t-1)\right)^2}{t}, \quad (4)$$

where $\overline{q}_n(t) = \frac{1}{t}\sum_{\tau=1}^{t} q_n(\tau)\mathbb{1}_n(\tau)$. The detailed derivation can be found in the Appendix A. Based on (4), we can decompose the original optimization problem into a series of combinatorial optimization in each time slot. In particular, in each time slot t, we need to solve the following problem:

$$\max_{(q_n(t))_{n=1}^N} \sum_{n=1}^{N} \mathbb{E}[q_n(t)\mathbb{1}_n(t)] - \alpha \sum_{n=1}^{N} \mathbb{E}[d_n(f_{c(t)}^R(q_n(t)))] \quad (5)$$
$$- \beta \sum_{n=1}^{N} \mathbb{E}\left[\frac{(t-1)\left(q_n(t)\mathbb{1}_n(t) - \bar{q}_n(t-1)\right)^2}{t}\right]$$

$$\text{s.t.} \sum_{n=1}^{N} f_{c(t)}^R(q_n(t)) \le B(t), \quad (6)$$

$$f_{c(t)}^R(q_n(t)) \le B_n(t), \forall n. \quad (7)$$

Such a decomposition is motivated by the fact that the average gap between the cumulative QoE by solving (5) over a finite time horizon T and the QoE by directly solving (1) converges to zero as $T \to \infty$ when the quality level is continuous rather than discrete. In particular, let $\widehat{\text{QoE}}(T)$ be the total QoE by sequentially solving (5) over a finite time horizon T and $\text{QoE}^*(T)$ be the QoE by solving (1). If the quality level is continuous, then

$$\lim_{T\to\infty} \frac{1}{T}\left(\widehat{\text{QoE}}(T) - \text{QoE}^*(T)\right) = 0. \quad (8)$$

This is true since $\text{QoE}(T)$ is concave. The rest of the proof follows the same line of that in [13, Theorem 1].

Let $\delta_n \triangleq \mathbb{E}[\mathbb{1}_n(t)]$. Then, the objective function (5) is equivalent to the following function:

$$\max_{(q_n(t))_{n=1}^N} \sum_{n=1}^{N} h_n(q_n(t)) \quad (9)$$

where $h_n(q_n(t)) \triangleq \delta_n q_n(t) - \alpha\mathbb{E}[d_n(f_{c(t)}^R(q_n(t)))] - \beta\left(\delta_n \frac{(t-1)\left(q_n(t)-\overline{q}_n(t-1)\right)^2)}{t} + (1-\delta_n)\frac{(t-1)\overline{q}_n^2(t-1)}{t}\right)$. Noting that δ_n is the successful prediction probability of user n, i.e., the probability that the delivered portion covers the actual FoV given a 6-DOF motion prediction algorithm. This successful prediction probability can be estimated via the average prediction probability $\overline{\delta}_n(t)$, which converges to δ_n as $t \to \infty$.

However, our objective function (9) is discrete with respect to the quality level and thus become a nonlinear Knapsack problem with concave objective function and convex constraints (see [16]), which is a well-known NP-hard problem. Hence, it is difficult to obtain the optimal solution when the number of users N is large. Like the classic solution to the traditional Knapsack problem, one natural way is to determine the quality level based on its obtained QoE value (called the value-greedy approach). Another way is based on the ratio of the obtained QoE value and the required rate (called the density-greedy approach). However, both these approaches can result in poor performance in certain cases that usually appear in a practical problem. For example, consider the case with $h_1(1) = 1, f_c^R(1) = 0.5, h_2(2) = 4, f_c^R(2) = 2.5$, where the available throughput for the server is 2.5. Then, the density-greedy approach allocates the rate to user 1 due to its larger density and cannot further allocate the rate due to the limited throughput. In such a case, the QoE equals 1 while the optimal QoE is 4 (by allocating quality level 2 to user 2). Hence, the density-greedy approach suffers from a great performance loss. Similarly, for the case with $h_1(1) = h_2(1) = h_3(1) = h_4(1) = 2, f^R(1) = 0.5, h_2(2) = 3, f^R(2) = 2$, where the available throughput for the server is 2. In such a case, the value-greedy approach selects the quality level 2 for user 2 and cannot allocate more quality levels due to the throughput limitation. In such a case, the QoE is equal to 3 while the optimal QoE is 8 (by selecting quality level 1 for the first four users). Hence, the value-greedy approach also suffers from severe performance loss.

Interestingly, in the first case, the value-greedy approach allocates quality-level 2 to user 2 and leads to a QoE of 4, which is optimal. Similarly, in the second case, the density-greedy approach allocates the quality level 1 for the first four users and results in the QoE of 8, which is also optimal. This naturally suggests combining the density-greedy and value-greedy approaches (called the density/value-greedy approach) and selecting the best quality level allocation from these two approaches.

Our quality level allocation algorithm (cf. Algorithm 1) is based on the density/value-greedy approach. We begin with the density-greedy approach to get a temporary result. Particularly, we start from the lowest quality level and select the user with the largest ratio of the QoE improvement to the increment on required rate (defined as η_n). We then increase the quality level by one for the selected user. Such a process will be repeated until one of the following three conditions happens: 1) the rate meets the constraints for the user or the server; 2) the quality level reaches the highest value; 3) the density is lower than zero. Note the 1) and 2) are verified via $quality\_verification(q, \mathcal{I})$ in Algorithm 1. After getting the allocation policy by the density-greedy approach, we use the value-greedy approach to get the other allocation strategy. It follows the same procedure except that we need to find the user with the largest QoE improvement (defined as v_n). Finally, we compare those two approaches and select the better quality level allocation policy with a higher objective function value. Different from the traditional knapsack problem, we utilize the increment of the quality level instead of individual

Algorithm 1: Density/Value-Greedy Algorithm

Given the cloud server throughput $B(t)$, and the throughput for user n $B_n(t)$;
Initialize: $\mathcal{Q}_d = \{1, 1, \ldots, 1\}$ and $\mathcal{I} = \{1, 2, \ldots, N\}$;
while $\mathcal{I} \neq \{\}$ **do**
  For each $n \in \mathcal{I}$, evaluate $\eta_n = \frac{h_n(q_n+1)-h_n(q_n)}{f_t^R(q_n+1)-f_t^R(q_n)}$;
  $n^* = \arg\max \eta_n$;
  if $\eta_{n^*} < 0$ **then**
    $\mathcal{I} = \{\}$;
  else
    $q_{n^*} = q_{n^*} + 1$;
    quality_verification(q_{n^*},$\mathcal{I}$);
  end
end
$V_d = \sum_i^N h_n(q_n), q_n \in \mathcal{Q}_d$;
Initialize: $\mathcal{Q}_v = \{1, 1, \ldots, 1\}, \mathcal{I} = \{1, 2, \ldots, N\}$;
while $\mathcal{I} \neq \{\}$ **do**
  For each $n \in \mathcal{I}$, evaluate $v_n = h_n(q_n + 1) - h_n(q_n)$;
  $n^* = \arg\max v_n$;
  if $v_{n^*} < 0$ **then**
    $\mathcal{I} = \{\}$;
  else
    $q_{n^*} = q_{n^*} + 1$;
    quality_verification(q_{n^*},$\mathcal{I}$);
  end
end
$V_v = \sum_i^N h_n(q_n), q_n \in \mathcal{Q}_v$;
if $V_d > V_v$ **then**
  return $\mathcal{Q}_d$
else
  return $\mathcal{Q}_v$
end
Def `quality_verification(`q_n, $\mathcal{I}$`):`
  if $q_n == L$ **then**
    $\mathcal{I} = \mathcal{I} \backslash \{n\}$;
  end
  if $f_t^R(q_n) > B_n(t)$ *or* $\sum_{n \in \mathcal{N}} f_t^R(q_n) > B(t)$ **then**
    $\mathcal{I} = \mathcal{I} \backslash \{n\}$;
    $q_n = q_n - 1$;
  end

elements. The density/value-greedy approach can be shown to achieve a value that is at least $1/2$ of the maximum value of the classic Knapsack problem [17]. We will show that such a result still holds in the nonlinear Knapsack problem with concave objective function and convex constraints.

Theorem 1. *Algorithm 1 reaches at least* $1/2$ *of the optimal value for our optimization problem* (5)-(7)*.*

Proof. Define the optimal solution of the discrete optimization problem is OPT, the solution of the density greedy algorithm is V_d and the solution of the value greedy algorithm is V_v. Define the density for user n when we want to improve its quality from j to $j+1$ as $\eta_{n,j} = \frac{h_n(q_j+1)-h_n(q_j)}{f_t^R(q_j+1)-f_t^R(q_j)}$, its value increment as $v_{n,j} = h_n(q_j + 1) - h_n(q_j)$. Notice that we are discussing the increment of the objective function instead of the objective function itself. Since our objective function is a concave function and the rate function is a convex function, we have $\eta_{n,j} \geq \eta_{n,j+1}$. Therefore, we are able to choose the quality improvement with the largest density when applying the density greedy algorithm. We define the index of the first quality improvement that is rejected due to the total rate limit is i_m, j_m, i.e., we achieve the total bandwidth limit when we want to increase user i_m's quality level from j_m to $j_m + 1$. Define V_p as the optimal solution when we can allocate partial of the quality improvement to the user. We can reach V_p if we follow the density greedy algorithm and allocate partial of the last quality improvement i_m, j_m to meet the rate limit, since we have consumed all available rate with the largest possible value increment. Clearly, OPT can only be smaller than V_p, since any solution of our problem is a feasible solution when a fraction of quality allocation is allowed. Therefore, we have

$$V_d + v_{i_m,j_m} \geq V_d + \alpha v_{i_m,j_m} = V_p \geq OPT, \qquad (10)$$

where $0 \leq \alpha \leq 1$. Based on the concavity of the objective function, we have $v_{n,j} \geq v_{n,j+1}$. Since we choose the user index with the largest value increment in the value greedy algorithm, we have

$$v_{i_m,j_m} \leq V_v/m. \qquad (11)$$

Combine (10) and (11), we have

$$V_d + V_v/m \geq OPT. \qquad (12)$$

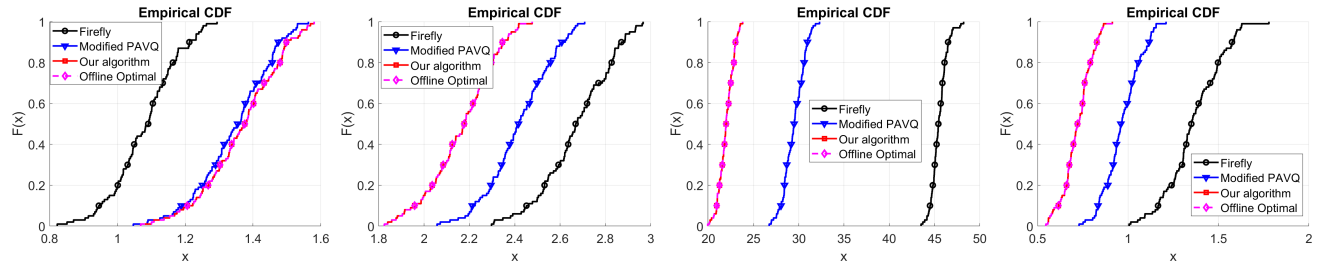
Because $m \geq 1$, the larger one between V_d and V_v is certainly larger than or equal to half of the optimal solution. Note that the individual bandwidth constraint (7) for each user only limits the range of j, which does not influence our result. □

IV. Trace-based Simulation

In this section, we develop a trace-based simulation platform to validate the efficiency of our algorithm compared with two state-of-the-art rate allocation algorithms for multiple users:

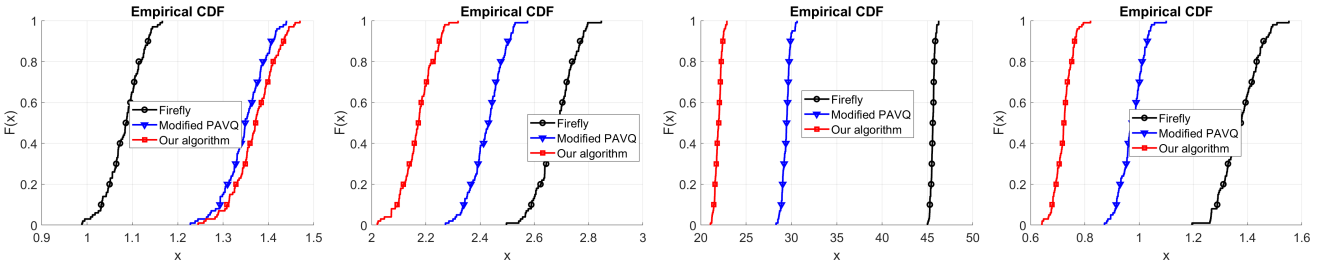
- Adaptive Quality Control algorithm in Firefly (see [8]), which uses Least Recently Used (LRU) algorithm to allocate the rate for multiple users. Due to its heuristic property and similar setup in the original paper, it can be directly deployed to our problem without modifications.
- Practical Adaptive Variance Aware Quality Allocation algorithm (PAVQ) [13]. Notice that we cannot directly apply this algorithm since it does not consider the delay in the original paper. For a fair comparison, we modify the way to calculate μ_i^P on its algorithm description in [13, Section V] to adapt to our problem setting.

We assume that the server has the perfect knowledge of the delay and throughput to avoid the performance degradation induced by the imperfect estimation. We will implement our algorithm in a real system and demonstrate that it is robust to the imperfect delay and throughput estimation in Section VI.



(a) QoE (b) Quality (c) Delay (d) Variance

Fig. 2: Trace-based simulation with 5 users



(a) QoE (b) Quality (c) Delay (d) Variance

Fig. 3: Trace-based simulation with 30 users

We use the motion trace dataset from [8], which is collected from two large VR scenes among 25 users. In each time slot, the server determines the quality level for each user from a quality set of six different levels. Once the quality level is selected, we will calculate the required rate based on the size of corresponding tiles generated in Section VI. The linear regression model is used to predict the 6-DoF motion in the next time slot due to its simplicity and efficiency. We set $\alpha = 0.02$ and $\beta = 0.5$ in the QoE definition (cf. Section II) to assign a higher priority to the variance compared with the delay.

We generate the network trace from two different network trace datasets: the broadband dataset provided by the FCC [18], and the 4G/LTE mobile dataset from the Ghent University [19]. We randomly generate half of the requested traces from the "Web browsing" category of the FCC dataset (see [20], [21]) in the March 2021 collection, which includes millions of network throughput traces. The other half of the requested traces are generated from Ghent's dataset. Notice that this dataset is relatively small (40 logs, five hours in total) such that we reuse some of the logs. We set the length of each trace to 300 seconds and the network throughput between 20 Mbps to 100 Mbps to avoid trivial video quality selection. The total bandwidth of the server is set to 36 Mbps times the number of users, which respects the average rate requirement of the tiles by a medium quality level.

Note that the network trace does not contain the content delivery delay information. We generate the delivery delay according to

$$d_n(f^R_{c(t)}(q_n(t))) = \frac{f^R_{c(t)}(q_n(t))}{B_n(t) - f^R_{c(t)}(q_n(t))} \tag{13}$$

where we recall that $B_n(t)$ is the maximum allowable rate for user n in time slot t. This models the delay as that in M/M/1 queueing system (Poisson arrival process, exponential service time), which is usually used to model the queueing delay in wireless transmission (see [22]). Since we assume that the server has the perfect knowledge of the network environment, it knows the delay corresponding to each quality level before it allocates the quality level for each user.

We perform simulation in two different setups with 5 users and 30 users. We have two different settings because when the number of users is small, we can use the brute force method to generate the optimal offline solution of problem (5)-(7) and see how close our approach is to the optimal solution. However, the collaborative VR-based system typically has many users, and thus we also consider the case with 30 users. For each simulation setup, we choose 100 different traces for each user from the dataset such that each user will have a different network throughput in each time slot. Notice that the network throughput in the dataset usually lasts for several seconds for each point, which is far more lager than the interval between each time slot (15ms under 66 FPS). Therefore, we just let multiple continuous slots share the same bandwidth until their cumulative time reaches the trace's duration.

Fig. 2 and Fig. 3 compare the performance of our proposed algorithm with state-of-the-art algorithms in terms of the cumulative probability distribution (CDF) of the performance

metrics when there are 5 and 30 users, respectively. We can observe from Fig. 2 that our proposed algorithm almost matches the offline optimal solution in the average QoE (see Fig. 2a), average quality (see Fig. 2b), average delivery delay (see Fig. 2c), and the variance of quality levels of delivered content (see Fig. 2d). It outperforms the Firefly and modified PAVQ algorithm in terms of the average QoE, as shown in Fig. 2a. This can also be seen from Fig. 2c and Fig. 2d that our algorithm improves the average delivery delay and the variance of the quality levels of the delivered content at the cost of reducing the average quality of received content, as shown in Fig. 2b. Interestingly, it shows that the modified PAVQ algorithm is also close to the optimal QoE value, yet it follows from a totally different quality allocation strategy, resulting in a large difference in separate components of QoE. We can observe similar phenomena when there are 30 users, as shown in Fig. 3.

To further evaluate the performance of our algorithm, we also develop a practical system and deploy our algorithm as well as two baseline algorithms on that system. We will introduce details for the system design and implementation in the following sections.

V. System Design

In this section, we design a collaborative VR system to deploy our algorithm in real-world and validate its efficiency. The system architecture is depicted in Fig. 4. We consider the specific scenario of remote learning when a teacher is teaching in a VR classroom, and all other users (i.e., the students) can interact with the teacher. The server is responsible for receiving the motion traces from users and delivering the VR content corresponding to the predicted 6-DoF motion and the selected quality level for each user. Once a user gets the VR content, it will be decoded and displayed on time. Next, we will introduce the methodology to design our system in detail.

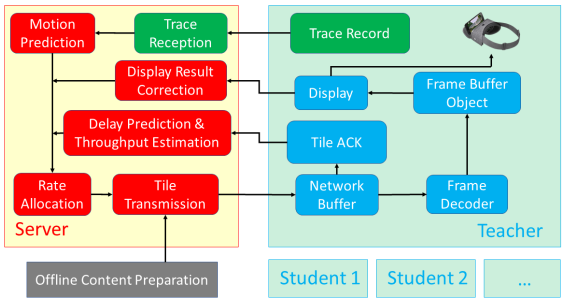


Fig. 4: System architecture

Offline tile rendering and encoding. To ensure that the system is scalable to multiple users, we have rendered all possible tiles of the scene in Unity before the transmission. Therefore, we do not need to consider the overhead of rendering and encoding the VR content into different quality levels. Meanwhile, the server will hold a buffer in the memory during the runtime to cache some of the tiles that avoids the swapping overhead. Since the user's future location is bounded by her moving speed in the virtual world, the server only needs to cache the tiles within a range of the user's current position and dynamically adjust the cached content corresponding to the user's movement. As such, once the server determines the requested VR content for a user, it can immediately start the transmission without any rendering, encoding, or processing delay. To facilitate the transmission, We project the panoramic scene into a rectangular texture using equirectangular method [23] and split each texture into four tiles as shown in Fig. 5. Note that we can also apply other projection methods to our system. We use FFmpeg [24] to encode the frames with different CRF values. All the tiles will be indexed by a video ID corresponding to their position, tile ID, and quality. We only need to search the video ID during the runtime, which greatly facilitates communication.

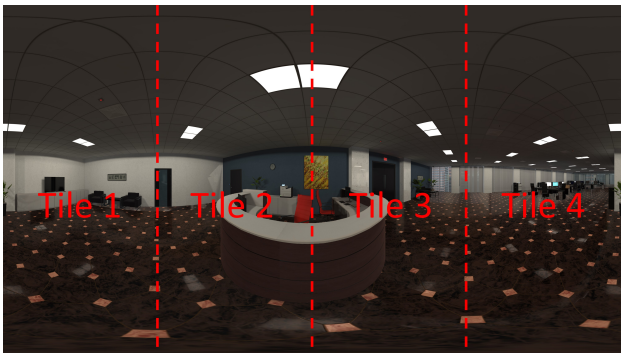


Fig. 5: Tile partitioning

Communication protocol. To better adapt to the real-time interactivity and meet the strict display deadline of the tiles, we use Real-Time Transport Protocol (RTP) in our system instead of traditional TCP to avoid the extra delay caused by the TCP rate control algorithm. RTP is built upon UDP such that we can concisely control the sending rate of the tiles and either retransmit the tiles or not.

Handling repetitive tiles. Avoiding the retransmission of the repetitive tiles that have already been delivered can significantly save the network bandwidth. However, we cannot know which packet has been successfully delivered since RTP is an unreliable protocol. To resolve this issue, we manually send acknowledgments (ACK) from the user to the server through TCP. As such, the server records the tiles that have already been delivered and will not transmit the same tiles again. However, due to the memory size limitation, the user cannot hold all received tiles in its RAM. To avoid computation-intensive swap operation on the user, we will release old tiles once the total number of tiles reaches the user-specific threshold (depending on the device's memory size). The user also sends ACKs to let the server know when the tiles are released. After that, the server will retransmit the tiles if they are requested again.

Pipelining of transmission and decoding. To enable the pipelining of the content delivery and the decoding, we reserve a time slot for the decoding on the user-side. For example, if

the server receives the pose from the teacher at the time slot t, it will deliver the predicted tiles at time slot $t+1$ to all users. Those tiles will be decoded on the users and displayed at the time slot $t+2$. Note that we use the hardware decoders on the client-side such that the transmission and the decoding work in parallel without conflicts of computation resources.

Delay measurement and prediction. Due to the asynchronous clock on the server and the user, we estimate the delay by computing the time duration between receiving the first and the last packet of the current time slot on the user-side. Note that we ignore the propagation delay since it is negligible under the wireless network with only one hop. As mentioned in Section II, the relationship between the delay and the rate is non-linear. Therefore, we use polynomial regression to predict the delay instead of linear regression to avoid extra performance degradation.

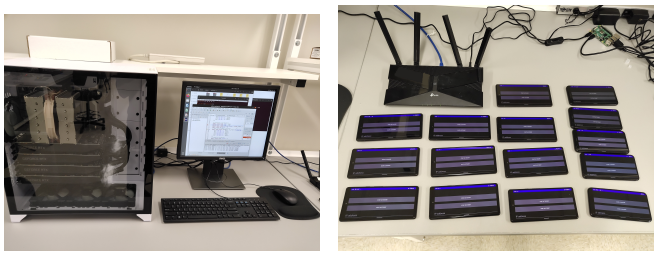
Motion prediction and throughput estimation. We use linear regression to predict the virtual position and head orientation in each axis independently, which follows the methodology in [8]. To further tolerate the prediction error, we add a margin on the FoV to select the delivered tiles and transmit all tiles that overlap with this margin. We estimate the available bandwidth for each user using Exponential Moving Average (EMA) algorithm.

Next, we will deploy our system into commercial devices and evaluate the performance of our algorithm compared with two baseline algorithms. The details on the software and hardware configuration as well as evaluation results can be found in the next section.

VI. Implementation and Real-World Evaluation

Implementation of the server and users. We develop our server application in Eclipse with Java language, which can be deployed on either a Windows or Linux computer. We only use the common Java packages like java.util and java.io without any extra burden on installing external packages. The user application is designed by Android Studio with the Android SDK and Java programming language. To display the tile received from the server, we use Open GL ES to reproject the equirectangular map to the panoramic view. To ensure the interactivity in the collaborative VR system, we do not prefetch the VR content. Each tile will either be displayed or dropped in each time slot. Android Media Codec is used to accelerate the decoding of the delivered tile by using multiple parallel decoders. Notice that the number of decoders depends on the device, while we set the number to 5 during the experiment to avoid the performance degradation caused by the decoding. Users will replay real users' motion traces and upload the trace to the server through TCP periodically. The total line of code (LoC) of our system is about 10,500.

Content preparation. We use the commercial VR scene Office [25] purchased in Unity asset store to evaluate our proposed algorithm. We split the whole panoramic scene into a grid world with the granularity of $5cm \times 5cm$ to provide smooth translational movement for users (see [8]). We have rendered the equirectangular map of the VR scene in 1440p (Quad HD, 2560×1440) resolution and encoded all tiles in six different quality levels with CRF values $\{15, 19, 23, 27, 31, 35\}$, separately. With a higher CRF value, the VR scene will be encoded in a lower bitrate, resulting in a lower visual perception quality. Therefore, we index those CRF values with corresponding quality levels $\{6, 5, 4, 3, 2, 1\}$ when running our algorithm. The content database capacity is about 171 GB. We use the user trace of the Office scene inherited from the paper [8], which matches the offline content.

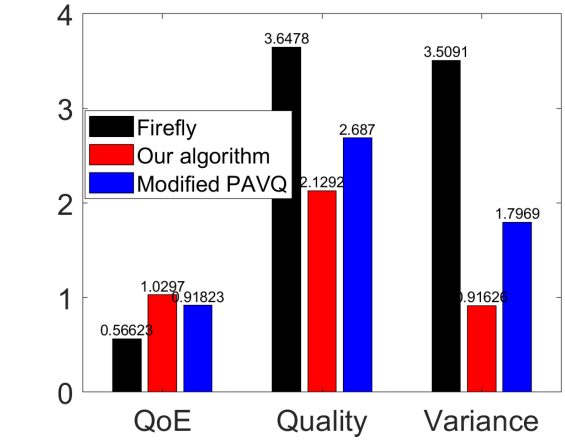


(a) Server (b) Clients: 15 smartphones

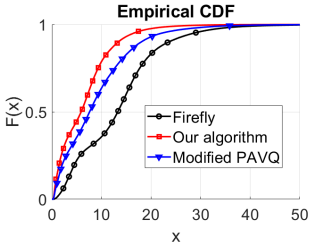
Fig. 6: Equipment in our system

Hardware and Environment. The system consists of fifteen off-the-shelf commercial smartphones (including ten Google Pixel 6, two Google Pixel 5 and three Google Pixel 4). One of the Google pixels is selected as the teacher user, while all other student users interact with it. Note that a single router cannot support such a large amount of users in our system such that we bridge two routers together using Ethernet to enhance the available bandwidth. The server is deployed on a Lambda workstation with Intel Core i9-10980XE CPU @ 3.00GHz $\times$ 36, NVIDIA GeForce RTX 3070 Graphics Card $\times$ 4, 128 GB memory, 4 TB disk, and Ubuntu 20.04 LTS. The prototype of our system is shown in Fig. 6. All smartphones connect to the TP-LINK Archer AX50 routers wirelessly, while we use an Ethernet cable to connect the server and a switch that communicates with the two routers. We split the 15 users into two groups and conduct the experiments under two different setups. Particularly, we first use only one router with 8 smartphones. Next, we extend the scale of our experiments by adding the other router with 7 smartphones. During the runtime, we use Linux TC [26] to limit the network throughput for each user to emulate the commercial network and consider network heterogeneity. To avoid trivial quality level allocation, we have five different throttle guidelines, listed as 40 Mbps, 45 Mbps, 50 Mbps, 55 Mbps, and 60 Mbps, which are randomly assigned to each user. Note that the actual throughput varies with time under the wireless network. We also limit the total available throughput of the server as 400 Mbps for the first experimental setup with one router and 800 Mbps for the second experimental setup with two routers to emulate widely-accepted 802.11 ac routers. We set the hyperparameters as $\alpha = 0.1$ and $\beta = 0.5$ in the QoE definition. We repeat the experiments five times to get the average results.

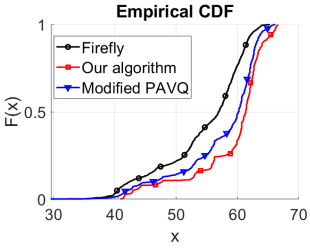
Result Analysis. Fig. 7 compares the performance of our proposed algorithm with the Firefly and modified PAVQ algorithm under the first experimental setup with 8 users, where



(a) QoE, Quality, and Variance

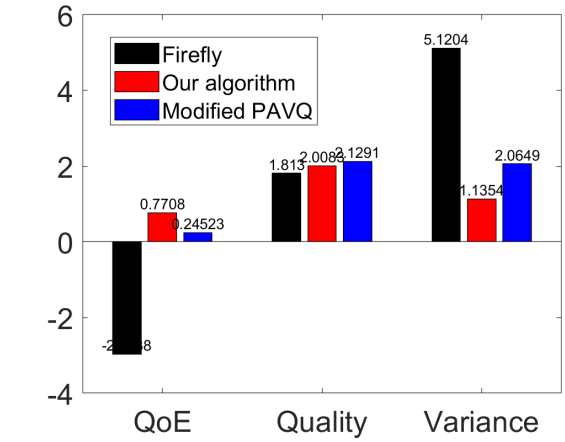


(b) Delay CDF

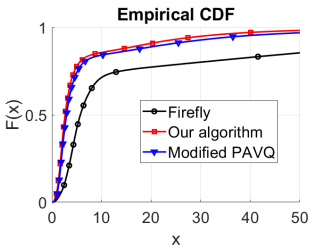


(c) FPS CDF

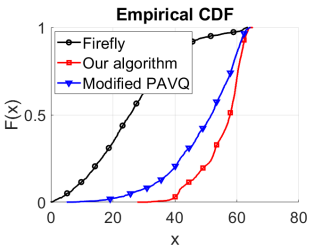
Fig. 7: Real-world evaluation results (8 users, single router)



(a) QoE, Quality, and Variance



(b) Delay CDF



(c) FPS CDF

Fig. 8: Real-world evaluation results (15 users, 2 routers)

we have drawn the average value of QoE, quality level, and the variance for all users within the whole time horizon of the trace for five experiments. We can observe from Fig. 7a that our proposed algorithm yields the largest average QoE and significantly reduces the VR content delivery delay (see Fig. 7b) and the variance of quality levels of the delivered content at the cost of acceptable quality degradation. In particular, our algorithm gets a 81.9% improvement over the Firefly algorithm and 12.1% improvement over the modified PAVQ algorithm on the average QoE of all users within a finite time horizon. Moreover, our proposed algorithm can also achieve the best frame rate compared with the Firefly and modified PAVQ, as shown in Fig. 7c. Specifically, our algorithm reaches 60 FPS on average in this system. The difference in the frame rate lies in the heterogeneous delay and packet loss of the VR content. With a larger VR content delivery delay, the content cannot be decoded and displayed on time, resulting in a missed frame. Fig. 8 shows the evaluation results under the second experimental setup with 15 users. In particular, the variance of the bandwidth capacity is even larger with two routers working together due to the possible wireless interference. It shows that both Firefly and modified PAVQ are vulnerable to the dynamic network environment and suffer from poor performance due to the inaccurate throughput estimation. In contrast, our algorithm is robust to such imperfect information and still has reasonably good performance. Specifically, our algorithm gets a 214.3% improvement over the modified PAVQ algorithm, while Firefly even reaches negative QoE.

VII. Related Work

In this section, we summarize three different areas which are close to our work: video streaming, virtual reality and 360° video, and the knapsack problem.

Video streaming. Video streaming is one of the hottest topics in the networking community. Various researchers have explored novel methods to dynamically allocate the quality of video content to achieve satisfactory QoE for users. Some of them proposed a principled design with novel mathematical formulation (e.g., [13], [21]), while others applied the reinforcement learning approach to the video streaming setup (e.g., [20], [27]). With high-speed and low-latency network development, researchers also make great progress on the real-time video streaming (e.g., [28]–[30]). Unlike traditional video streaming, the delay and packet loss significantly affect the QoE of users for real-time video streaming. Depending on this specific property, researchers have developed efficient methods to improve the QoE of users and deployed their approaches to commercial platforms to test the efficiency. Although advanced video streaming technologies have been widely used in large-scale commercial applications, virtual reality and panoramic video streaming are still open problems due to the unique challenges and opportunities.

Virtual reality and 360° video. VR and 360° video content delivery consume significantly more network bandwidth than traditional videos. Researchers explored various ways to optimize the system performance to achieve satisfactory

QoE for each user. In [7]–[10], the authors leveraged the motion prediction and tile partitioning and delivered partial panoramic content to the users without affecting their visual perception. In another line of work [31]–[33], the authors efficiently explored the computation capability on the user-side to reduce the communication and computation workload from the server-side. Some interesting works (e.g., [34]) utilized the fact the user is less sensitive to the quality of the panoramic content when she is moving or there is a change in luminance, and develop algorithms that adjust the visual quality of the panoramic content while maintaining satisfactory QoE. In more recent work (e.g., [35]–[37]), the authors explored efficient tile encoding approaches to significantly reduce the network bandwidth usage. To the best of our knowledge, none of the existing work considered the principled design for collaborative VR applications and deployed the algorithms into a local system with up to 15 users.

Knapsack problem. The knapsack problem is to select a set of items with different values and weights with the goal of maximizing the total value while meeting the desired weight, which has many applications such as cloud computing and networking. The problem possesses combinatorial nature and has been shown to be NP-hard. The researchers have explored various efficient ways to solve the traditional knapsack problem (c.f., [38]). Beyond the scope of traditional $0/1$ knapsack problem, there are also some works focusing on the concave or convex optimization function. [16] presented a thorough survey on algorithms and applications for the nonlinear knapsack problem. Besides, [39] considered a problem which is highly related to this paper which aims to maximize a separable concave objective function, subject to a single convex packing constraint, and converted it to an equivalent 0/1 knapsack problem.

VIII. Discussion

Online rendering and encoding. Since the teacher user always needs to adapt the VR content of the class, the offline rendering method may not be applicable due to its time-consuming exhaustive preparation. Ideally, we can use Unity and Nvidia NVENC to render and encode the tiles in real-time. However, the overhead of rendering and encoding for multiple quality levels makes it difficult to meet the synchronization performance required by the collaborative VR application. One possible solution is to coordinate multiple GPUs in a server to enable multiple encoders working in parallel with the rendering, which is also left for future work.

Handling packet loss. Since we use RTP to avoid the influence of the TCP rate control algorithm, it is inevitable to have packet loss during the transmission, which may lead to the performance degradation to some content. This may lead to a severe impact on the system performance when the network condition is poor. However, we do not incorporate it into our optimization problem formulation. Although our algorithm still outperforms the state-of-the-art algorithms without explicitly considering the packet loss, we believe it can be further improved by accounting for such information.

IX. Conclusion

In this paper, we considered the quality level allocation for geo-distributed collaborative VR applications, such as VR-based remote education/training. We aimed to provide all participating users with the best quality of experience (QoE) by formulating a combinatorial optimization problem over a finite time horizon. Our QoE metrics include the quality level, the average content delivery delay, and the variance of the viewing quality for users. We also considered the influence of imperfect motion prediction and incorporated it into our formulation. However, the optimal solution to such an optimization problem requires the knowledge of future information and thus is not physically implementable. As such, we decomposed this problem into a series of the combinatorial optimization problem in each time slot and developed an efficient and easily-implementable algorithm that yields at least $1/2$ of the optimal value. We performed trace-based simulation and demonstrated that our proposed algorithm performs very close to the optimal offline solution. Moreover, we developed a collaborative VR system to evaluate our proposed algorithm and open-sourced the implementations for the benefit of the community. The real-world experimental evaluations demonstrated the superior performance of our proposed algorithm over the state-of-the-art algorithms.

Appendix A
Variance Decomposition

$$
\begin{aligned}
T\sigma_n^2(T) &\overset{(a)}{=} (T-1)\sigma_n^2(T-1) \\
&\quad + \left(q_n(T)\mathbb{1}(T) - \bar{q}_n(T-1)\right)\left(q_n(T)\mathbb{1}(T) - \bar{q}_n(T)\right) \\
&\overset{(b)}{=} \sum_{t=1}^{T} \left(q_n(t)\mathbb{1}(t) - \bar{q}_n(t-1)\right)\left(q_n(t)\mathbb{1}(t) - \bar{q}_n(t)\right) \\
&\overset{(c)}{=} \sum_{t=1}^{T} \left(q_n(t)\mathbb{1}(t) - \bar{q}_n(t-1)\right) \\
&\quad \cdot \left(q_n(t)\mathbb{1}(t) - \frac{(t-1)\bar{q}_n(t-1) + q_n(t)\mathbb{1}(t)}{t}\right) \\
&= \sum_{t=1}^{T} \frac{(t-1)\left(q_n(t)\mathbb{1}(t) - \bar{q}_n(t-1)\right)^2}{t},
\end{aligned}
$$

where step (a) uses the variance iteration formula [15]; (b) is true by iteratively using the variance iteration formula; (c) is based on the iterative sample mean calculation.

References

[1] E. Bozki, P. Stark, H. Gao, L. Hasenbein, J.-U. Hahn, E. Kasneci, and R. Göllner, "Exploiting object-of-interest information to understand attention in vr classrooms," in *2021 IEEE Virtual Reality and 3D User Interfaces (VR)*. IEEE, 2021, pp. 597–605.

[2] R. A. Fuertes, "VR Collaborative Art is Now Possible Through MasterpieceVR Premium," 2017. [Online]. Available: https://edgy.app/vr-collaborative-art-is-now-possible-through-masterpiecevr-premium

[3] Y.-D. Malinov, D. E. Millard, and T. Bloun, "Stuckinspace: Exploring the difference between two different mediums of play in a multi-modal virtual reality game," in *2021 IEEE Virtual Reality and 3D User Interfaces (VR)*. IEEE, 2021, pp. 501–510.

[4] C. Hackl, "Social VR, Facebook Horizon And the Future of Social Media Marketing," 2020. [Online]. Available: https://www.forbes.com/sites/cathyhackl/2020/08/30/social-vr-facebook-horizon--the-future-of-social-media-marketing/?sh=30fb3fd05b19

[5] O. Abari, D. Bharadia, A. Duffield, and D. Katabi, "Enabling high-quality untethered virtual reality," in *14th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 17)*, 2017, pp. 531–544.

[6] L. Liu, R. Zhong, W. Zhang, Y. Liu, J. Zhang, L. Zhang, and M. Gruteser, "Cutting the cord: Designing a high-quality untethered vr system with low latency remote rendering," in *Proceedings of the 16th Annual International Conference on Mobile Systems, Applications, and Services*, 2018, pp. 68–80.

[7] F. Qian, B. Han, Q. Xiao, and V. Gopalakrishnan, "Flare: Practical viewport-adaptive 360-degree video streaming for mobile devices," in *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking*, 2018, pp. 99–114.

[8] X. Liu, C. Vlachou, M. Yang, F. Qian, L. Zhou, C. Wang, L. Zhu, K.-H. Kim, G. Parmer, Q. Chen *et al.*, "Firefly: Untethered multi-user {VR} for commodity mobile devices," in *2020 {USENIX} Annual Technical Conference ({USENIX}{ATC} 20)*, 2020, pp. 943–957.

[9] J. He, M. A. Qureshi, L. Qiu, J. Li, F. Li, and L. Han, "Rubiks: Practical 360-degree streaming for smartphones," in *Proceedings of the 16th Annual International Conference on Mobile Systems, Applications, and Services*, 2018, pp. 482–494.

[10] Y. Zhang, P. Zhao, K. Bian, Y. Liu, L. Song, and X. Li, "Drl360: 360-degree video streaming with deep reinforcement learning," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*. IEEE, 2019, pp. 1252–1260.

[11] J. Chen, B. Li, and R. Srikant, "Thompson-sampling-based wireless transmission for panoramic video streaming," in *2020 18th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOPT)*. IEEE, 2020, pp. 1–3.

[12] J. Chen, X. Qin, G. Zhu, B. Ji, and B. Li, "Motion-prediction-based wireless scheduling for multi-user panoramic video streaming," in *IEEE INFOCOM 2021-IEEE Conference on Computer Communications*. IEEE, 2021.

[13] V. Joseph and G. de Veciana, "Jointly optimizing multi-user rate adaptation for video transport over wireless systems: Mean-fairness-variability tradeoffs," in *2012 Proceedings IEEE INFOCOM*. IEEE, 2012, pp. 567–575.

[14] C. Yim and A. C. Bovik, "Evaluation of temporal variation of video quality in packet loss networks," *Signal Processing: Image Communication*, vol. 26, no. 1, pp. 24–38, 2011.

[15] B. Welford, "Note on a method for calculating corrected sums of squares and products," *Technometrics*, vol. 4, no. 3, pp. 419–420, 1962.

[16] K. M. Bretthauer and B. Shetty, "The nonlinear knapsack problem–algorithms and applications," *European Journal of Operational Research*, vol. 138, no. 3, pp. 459–472, 2002.

[17] J. Kleinberg and E. Tardos, *Algorithm design*. Pearson Education India, 2006.

[18] F. C. Commission, "Measuring broadband raw data releases - fixed." [Online]. Available: https://www.fcc.gov/oet/mba/raw-data-releases

[19] 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 Communications Letters*, vol. 20, no. 11, pp. 2177–2180, 2016.

[20] H. Mao, R. Netravali, and M. Alizadeh, "Neural adaptive video streaming with pensieve," in *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, 2017, pp. 197–210.

[21] X. Yin, A. Jindal, V. Sekar, and B. Sinopoli, "A control-theoretic approach for dynamic adaptive video streaming over http," in *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, 2015, pp. 325–338.

[22] J. F. Kurose, *Computer networking: A top-down approach featuring the internet, 3/E*. Pearson Education India, 2005.

[23] "Equirectangular projection." [Online]. Available: https://en.wikipedia.org/wiki/Equirectangular_projection

[24] "FFmpeg." [Online]. Available: http://ffmpeg.org/

[25] "QA Office and Security Room." [Online]. Available: https://assetstore.unity.com/packages/3d/environments/urban/qa-office-and-security-room-114109

[26] "Linux TC." [Online]. Available: http://man7.org/linux/man-pages/man8/tc.8.html

[27] R. Bhattacharyya, A. Bura, D. Rengarajan, M. Rumuly, S. Shakkottai, D. Kalathil, R. K. Mok, and A. Dhamdhere, "Qflow: A reinforcement learning approach to high qoe video streaming over wireless networks," in *Proceedings of the twentieth ACM international symposium on mobile ad hoc networking and computing*, 2019, pp. 251–260.

[28] A. Zhou, H. Zhang, G. Su, L. Wu, R. Ma, Z. Meng, X. Zhang, X. Xie, H. Ma, and X. Chen, "Learning to coordinate video codec with transport protocol for mobile video telephony," in *The 25th Annual International Conference on Mobile Computing and Networking*, 2019, pp. 1–16.

[29] H. Zhang, A. Zhou, J. Lu, R. Ma, Y. Hu, C. Li, X. Zhang, H. Ma, and X. Chen, "Onrl: improving mobile video telephony via online reinforcement learning," in *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking*, 2020, pp. 1–14.

[30] H. Zhang, A. Zhou, Y. Hu, C. Li, G. Wang, X. Zhang, H. Ma, L. Wu, A. Chen, and C. Wu, "Loki: improving long tail performance of learning-based real-time video adaptation by fusing rule-based models," in *Proceedings of the 27th Annual International Conference on Mobile Computing and Networking*, 2021, pp. 775–788.

[31] Z. Lai, Y. C. Hu, Y. Cui, L. Sun, N. Dai, and H.-S. Lee, "Furion: Engineering high-quality immersive virtual reality on today's mobile devices," *IEEE Transactions on Mobile Computing*, vol. 19, no. 7, pp. 1586–1602, 2019.

[32] J. Meng, S. Paul, and Y. C. Hu, "Coterie: Exploiting frame similarity to enable high-quality multiplayer vr on commodity mobile devices," in *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, 2020, pp. 923–937.

[33] M. Dasari, A. Bhattacharya, S. Vargas, P. Sahu, A. Balasubramanian, and S. R. Das, "Streaming 360-degree videos using super-resolution," in *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*. IEEE, 2020, pp. 1977–1986.

[34] Y. Guan, C. Zheng, X. Zhang, Z. Guo, and J. Jiang, "Pano: Optimizing 360 video streaming with a better understanding of quality perception," in *Proceedings of the ACM Special Interest Group on Data Communication*, 2019, pp. 394–407.

[35] C. Zhou, M. Xiao, and Y. Liu, "Clustile: Toward minimizing bandwidth in 360-degree video streaming," in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*. IEEE, 2018, pp. 962–970.

[36] X. Chen, T. Tan, and G. Cao, "Popularity-aware 360-degree video streaming," in *IEEE INFOCOM 2021-IEEE Conference on Computer Communications*. IEEE, 2021, pp. 1–10.

[37] M. Palash, V. Popescu, A. Sheoran, and S. Fahmy, "Robust 360○ video streaming via non-linear sampling," in *Proceedings of IEEE INFOCOM (the conference on computer communications)*, 2021.

[38] H. Kellerer, U. Pferschy, and D. Pisinger, "Multidimensional knapsack problems," in *Knapsack problems*. Springer, 2004, pp. 235–283.

[39] D. S. Hochbaum, "A nonlinear knapsack problem," *Operations Research Letters*, vol. 17, no. 3, pp. 103–110, 1995.