electronics

MDPI

Article

EVeREst: Bitrate Adaptation for Cloud VR

Mikhail Liubogoshchev [1,2], Evgeny Korneev [1,2], Evgeny Khorov [1,2,*]

[1] Kharkevich Institute for Information Transmission Problems of the Russian Academy of Sciences, 127051 Moscow, Russia; liubogoshchev@wireless.iitp.ru (M.L.); korneev@wireless.iitp.ru (E.K.)
[2] Moscow Institute of Physics and Technology, 141701 Moscow, Russia
* Correspondence: e@khorov.ru

Abstract: Cloud Virtual Reality (VR) technology is expected to promote VR by providing a higher Quality of Experience (QoE) and energy efficiency at lower prices for the consumer. In cloud VR, the virtual environment is rendered on the remote server and transmitted to the headset as a video stream. To guarantee real-time experience, networks need to transfer huge amounts of data with much stricter delays than imposed by the state-of-the-art live video streaming applications. To reduce the burden imposed on the networks, cloud VR applications shall adequately react to the changing network conditions, including the wireless channel fluctuations and highly variable user activity. For that, they need to adjust the quality of the video stream adaptively. This paper studies video quality adaptation for cloud VR and improves the QoE for cloud VR users. It develops a distributed, i.e., with no assistance from the network, bitrate adaptation algorithm for cloud VR, called the Enhanced VR bitrate Estimator (EVeREst). The algorithm aims to optimize the average bitrate of cloud VR video flows subject to video frame delay and loss constraints. For that, the algorithm estimates both the current network load and the delay experienced by separate frames. It anticipates the changes in the users' activity and limits the bitrate accordingly, which helps prevent excess interruptions of the playback. With simulations, the paper shows that the developed algorithm significantly improves the QoE for the end-users compared to the state-of-the-art adaptation algorithms developed for MPEG DASH live streaming, e.g., BOLA. Unlike these algorithms, the developed algorithm satisfies the frame loss requirements of multiple VR sessions and increases the network goodput by up to 10 times.

Keywords: cloud VR; bitrate adaptation; quality of experience; real-time adaptive video; video traffic; virtual reality

Citation: Liubogoshchev, M.; Korneev, E.; Khorov, E. EVeREst: Bitrate Adaptation for Cloud VR. *Electronics* **2021**, *10*, 678. https://doi.org/10.3390/electronics10060678

Academic Editor: Osvaldo Gervasi

Received: 31 October 2020
Accepted: 4 March 2021
Published: 14 March 2021

1. Introduction

Numerous Virtual Reality (VR) applications have emerged recently to improve entertainment [1,2], medicine [3,4], engineering [5,6], and other spheres of everyday life. To provide an immersive experience, such applications require minimal feedback delay and high image quality. The real-time rendering of the high-quality virtual environment is computationally expensive and requires special high-performance hardware. Cloud VR was introduced to promote VR and reduce the cost of the headsets [7]. In cloud VR, the headset, also called the Head-Mounted Display (HMD), only monitors the user's actions and sends the data to the remote server. The server renders the environment according to the received data and sends it back to the HMD as a video stream. The HMD then plays the video back to the user and, maybe, slightly adjusts the picture if the user's viewpoint changes.

There are still many issues related to the architecture of cloud VR systems, i.e., the computational split between the cloud or edge server and HMD, the proactive generation of the future virtual scenes, and bandwidth allocation [8–10].

Such issues become more challenging if the system tries to minimize the bandwidth used for data delivery. In particular, recent studies showed that the video shall have

up to 2K resolution, or even higher, with a frame rate of at least 60 Frames Per Second (FPS), and a presentation delay below 50 ms with very high probability [11,12]. Naturally, VR applications may generate a 360° video so that the user could observe the whole environment. As a result, cloud VR streams have a very high bitrate and consume much network resource. However, the user cannot simultaneously watch the whole 360° picture, so to reduce the network resource consumption, tile-based streaming was proposed [13,14] and standardized [15]. In tile-based streaming, only some parts of the spherical picture (called tiles) are streamed at high quality, while other parts can have lower quality. An efficient implementation of tile-based streaming requires special algorithms to optimize the choice of the quality of the tiles [16,17] and an accurate prediction of the future actions of the user [14,18,19].

To provide freedom of movement, HMDs use wireless technologies to transmit and receive the data [20]. Hence, the dynamic changes in the wireless environment also need to be taken into account. Any additional VR flow can significantly increase the load imposed on a wireless network [21]. If the cloud VR applications produce a constant bitrate video stream, the changes in the wireless channel state and the number of active clients in the network can lead to under- or over-utilization of the network [22]. Consequently, the Quality of Experience (QoE), i.e., the measure of user satisfaction with the service [23], degrades [24]. Therefore, cloud VR applications shall adequately react to the changes in the network conditions and try to prevent video playback interruptions while providing the highest possible visual quality. Thus, cloud VR applications should adaptively adjust the bitrate of the video stream [16].

Although the existing studies try to optimize the QoE for VR from various perspectives, they simplify the structure of the resulting video stream. The papers that study VR from the networking domain mostly do not consider video compression, and the papers from the multimedia domain often assume that the required amount of network resource can be reserved to satisfy the requirements. Therefore, the problem of the optimal choice of the bitrate of the compressed video stream (which is possibly tiled) to guarantee its successful real-time delivery remains unresolved.

In this paper, we consider the class of interactive VR applications, including but not limited to single- and multi-player gaming, education and training, and engineering. Such applications are inherently interactive. The user actions alter the virtual environment, which is updated respectively in real-time. It makes proactive video buffering on the HMD almost impossible. Along with tight video frame delivery deadlines, it limits the room to maneuver for the adaptation algorithms and complicates finding the right balance between video quality and feedback delay.

We aim to improve the QoE of the cloud VR users with a novel bitrate adaptation algorithm. We consider a generic network architecture that is not specifically tailored to the cloud VR service. Therefore, the adaptation is performed in a distributed manner based on the measurements performed by the end-points.

We design an algorithm that estimates the current network load and keeps track of the frame delivery delays. Based on this information, it chooses the maximal video bitrate that both fits in the current network conditions and leaves some room for possible changes in the network state.

The main contributions of this paper are as follows. First, we introduce the system model that allows us to implement limited-delay real-time adaptive cloud VR applications. Second, we analyze the state-of-the-art video quality adaptation algorithms and show why most of the existing approaches are not fit for the video quality adaptation in cloud VR. Finally, we design a video bitrate adaptation algorithm that we call the Enhanced VR bitrate Estimator (EVeREst). Our algorithm aims to maximize the average bitrate of VR video streams subject to a limited playback interruption probability. For that, the algorithm estimates both the current network load and the delay experienced by separate frames.

The rest of the paper is organized as follows. In Section 2, we describe the cloud VR system and introduce the problem statement. Section 3 reviews and analyzes the existing approaches to real-time video bitrate adaptation. Section 4 describes the algorithm developed in this paper. In Section 5, we present and discuss the obtained numerical results. Finally, Section 6 concludes the paper.

2. Cloud VR System Description and Problem Statement

A simplified cloud VR system is presented in Figure 1. Multiple VR HMDs via a Base Station (BS) are connected to Mobile Edge Computing (MEC) servers. Each HMD tracks the user's actions and sends their descriptions to the MEC server. The server renders the virtual environment, generates the corresponding video images, and immediately sends them back to the HMD, which displays them to the user. We assume that the BS shares the network resources among the flows belonging to different clients so that each one receives its fair share [25,26]. The clients enter and leave the network and use cloud VR applications in between. The channel state also changes because of the variable environment [27] and the number of active clients.

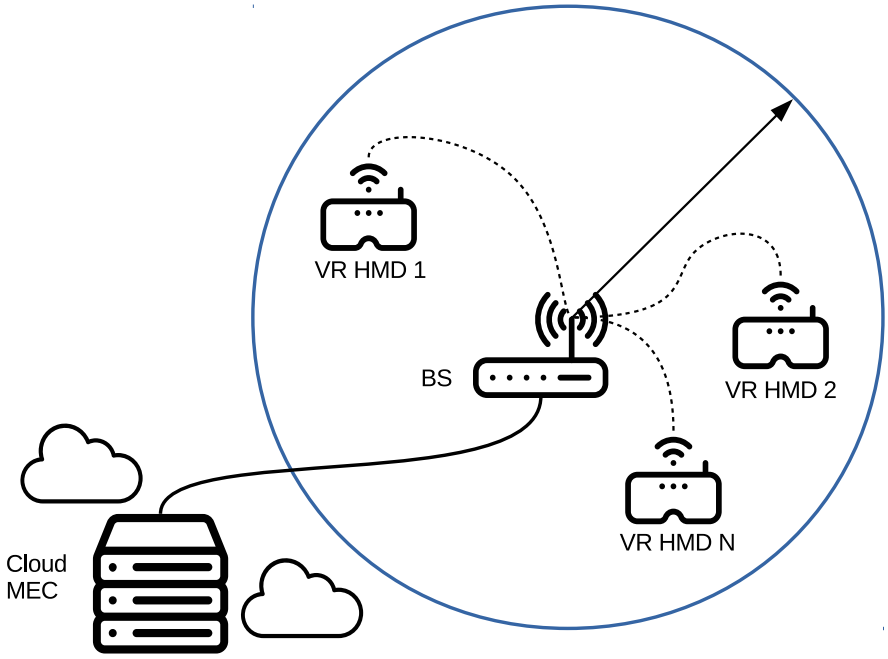


Figure 1. Cloud VR system.

Because the HMD needs to continuously display the video frames, the server shall generate the frames irrespective of the user actions. Therefore, each VR video flow can be modeled as a sequence of video frames generated by the server in real time with a fixed frequency ν and transmitted immediately. The video playback is organized as follows. The HMD accumulates a few frames in a buffer (often referred to as jitter-buffer) and only then starts the playback by fetching the frames one by one with the same frequency ν. The jitter-buffer compensates for the small unpredictable variations in the frame delivery rate. The initial depth of the buffer defines the maximal virtual scene refresh delay and the frame delivery time constraint D^{QoS}. If at some moment the buffer is empty, the video player repeats the previous frame and discards the late one when it arrives to keep the playback delay fixed. We call such a situation a frame loss. Note that although the buffer can have a size of a few frames, the scene refresh delay may be smaller in the considered system if, for example, the server predicts the user's actions for a few frames ahead.

Modern video codecs (e.g., h265 [28], AV1 [29]) efficiently compress the video stream, but make them much more bursty [30]. With these codecs, only a small part of the frames, namely key frames (or I-frames), are self-contained and can be decoded independently.

Other frames are called Predicted frames (P-frames). They carry only the differences between the current and the previous frame. Therefore, they are much smaller than the key frames. Such video compression dramatically reduces the average bitrate of the video at the cost of the increased burstiness of the flow: key frames are generated rarely but produce high loads on the network.

In this paper, we assume that higher bitrates for the same video codec improve the visual quality of the video. That is a valid assumption for any well-engineered video service [31,32]. Unfortunately, it is impossible to choose a certain video bitrate level at the beginning of the flow and keep it unchanged during the whole VR session because the network state dynamically changes [27]. A higher bitrate requires more network resources to transmit the video to the client and increases the probability of delays and losses if the resources are insufficient. Therefore, generating video with the highest possible bitrate can lead to QoE degradation [33]. In contrast, generating video with low bitrates reduces network resource consumption and frame loss probability. However, it also worsens the visual quality of the video and reduces the QoE.

Therefore, the applications shall adaptively choose the bitrate of the video. This paper assumes that to provide the efficient coexistence of different cloud VR operators in a generic network architecture that is not tailored to cloud VR service, the bitrate adaptation is performed in a distributed manner. Furthermore, we do not consider the problem of tile-based adaptation because, thanks to MPEG OMAF technology [15], this adaptation can be performed independently from the bitrate adaptation. The bitrate adaptation algorithm estimates the maximal amount of data that can be delivered to the client in time, and the tile-based adaptation decides on the size (e.g., in pixels) of each tile to fit in the constraint imposed by the bitrate adaptation.

Similar to the architecture considered by the MPEG DASH protocol [34], each client independently measures the state of the network and requests the corresponding video bitrate from a discrete set $\mathbb{B}$ based on some bitrate adaptation algorithm, while the server encodes and sends the video as requested. Similar to MPEG DASH, we assume that the video is encoded in chunks, and the server can change the bitrate only at the beginning of a chunk. The reason for such a limitation is in the video structure: the video stream with a new bitrate shall start from the keyframe. Therefore, to reduce the network resource consumption, we should switch the bitrate only when the next keyframe is generated. Note that this does not restrict the server from sending the frames as soon as they are generated. Despite many similarities with MPEG DASH, cloud VR applications require much lower delays, so feedback from the application and bitrate change requests shall be sent asynchronously, e.g., the application sends the bitrate change request when it decides to change the bitrate, but the server takes the request into account only when it generates the next keyframe.

For the considered bitrate adaptation scheme, the QoE of the cloud VR user can be reduced to the QoE for real-time video streaming because all the VR specifics (e.g., view adaptation) are not affected by the adaptation. Therefore, the aim of the bitrate adaptation algorithm is to optimize the real-time video streaming QoE. Unfortunately, such a QoE is a complex subjective metric, and no standardized and well-established method to estimate it exists [35–37]. However, in the considered scenario, based on recent studies [38,39], we can indicate the main factors that affect it. The first is the visual image quality. In the considered system, it corresponds to the average bitrate of the VR video flow. The second is the virtual scene refresh delay. However, in the considered system, the playback delay depends only on the initial jitter-buffer depth. Therefore, in the paper, we do not optimize this variable and consider it as a pre-defined system parameter. The third factor is the frame losses. To estimate its influence on the QoE, we introduce the concept of a VR session satisfied with the frame loss ratio, i.e., the session that has lost less than θ% of the frames. According to [12], a typical value of θ is approximately 0.5–2%. The final factor that is often addressed in the literature is the frequency of the video quality switches. Although we agree that this factor is important to provide an immersive experience, we do not focus on this metric

specifically because the importance of this factor is controversial, and some studies [40] show that users prefer eventual switches to higher quality over constantly low quality.

To aggregate these factors and simplify the interpretation of the results, we introduce the compound QoE metric, which we call goodput. It measures the total throughput of the cloud VR sessions satisfied with the frame loss ratio. When multiple cloud VR clients share the network resources, such a metric allows considering both the visual quality of their video streams and the accuracy of the network congestion estimation algorithm, which avoids overloading the network and achieves a limited frame loss ratio.

Our goal is to design a bitrate adaptation algorithm that will improve the QoE for the cloud VR users in the described system. We estimate the QoE as the goodput of the system.

3. Background in Video Adaptation

Many papers address the problem of video bitrate adaptation design. The majority of such papers propose the algorithms for MPEG DASH-based video-on-demand streaming. Such adaptation algorithms can be classified into buffer level-based [41–43], throughput estimation-based [44,45], reinforcement learning-based [46,47], and mixed techniques [48–50].

Apart from that, some papers, e.g., [51–53], study frameworks that assume centralized network-assisted bitrate adaptation. Such an approach simplifies the problem because intermediate network nodes have more information about the network state. Although the approach is very efficient, it requires the network to support the framework, which complicates its deployment [54].

Live streaming scenarios introduce additional constraints by limiting the maximal buffer size that is related to the playback delay. Therefore, the adaptation algorithms shall be re-designed properly. For example, the paper [55] extended the algorithm from [41] to the real-time streaming scenarios by extending the initially buffer-based algorithm with a throughput estimation-based module. In [56], the authors developed an algorithm based on buffer occupancy monitoring. They dynamically adjusted the buffer level threshold to choose the appropriate bitrate and aimed to transmit the video smoothly (i.e., with a low frequency of bitrate changes). In [57,58], the authors proposed reducing the delay by using HTTP/2-enabled server push methods. Finally, in [59], the authors developed an algorithm jointly utilizing playback rate control, latency-constrained bitrate control, and adaptive frame dropping.

Unfortunately, because of the chunk-based download structure, MPEG DASH is not fit for interactive video streaming, and even its low latency modification implies delays around a couple of seconds [60]. This structure allows the client to pre-buffer rather big amounts of video and smoothen the network capacity fluctuations. However, the bitrate adaptation for interactive video streaming is a more challenging task because the algorithms have a much lower space for the maneuver. Therefore, many papers develop various algorithms based on neural networks that predict the future channel state accurately and choose bitrates appropriately. The paper [61] designed a neural network that dynamically adjusts the bitrate and the playback rate of the downloaded video to reduce the probability of video stalling. With simulations, the authors showed that the proposed algorithm provides higher QoE than state-of-the-art ones in low latency streaming scenarios. Another neural network-based algorithm was proposed in [62]. The algorithm was designed for the remote control of unmanned aerial vehicles. It takes into account the fluctuations of the air-to-ground channel and predicts the channel capacity to stream the video with appropriate quality. Finally, the paper [63] considered a completely different approach and introduced a frame splitting technique. The video frame is divided into several sub-frames that are encoded sequentially at the cloud server. As long as the sub-frame is encoded, it can be sent to the HMD, and the server can proceed to encode the next sub-frame. This technique decreases the end-to-end latency because it smoothens the load imposed on the networks by the video flow, and the HMD starts decoding the video frame while the server is busy encoding the other parts of the same frame. On the downside, this solution

requires introducing large modifications to the video codecs to compose the frames from independently decoded sub-frames.

4. Algorithm Description

In this section, we describe the proposed VR bitrate adaptation algorithm. The algorithm consists of two building blocks: bitrate estimation and congestion estimation. The first block, bitrate estimation, decides whether the capacity of the connection between the server and the HMD is enough for timely delivery of the video stream with a higher bitrate, or instead, the bitrate shall be reduced to avoid VR frame losses; see Section 4.1. The second block, congestion estimation, intends to estimate whether the appearance of new traffic in the network will damage the VR flow; see Section 4.2. Finally, the interaction of these building blocks is described in Section 4.3.

4.1. Bitrate Estimation

The HMD keeps track of frame delivery times to determine whether it shall switch to another bitrate or continue receiving the video in the current bitrate. For that, the algorithm maintains two exponential averages D^{short} and D^{long} of the frame delivery delays calculated in the time windows t_{win}^{short} and t_{win}^{long} that correspond to short-term and long-term smoothing to track both local extremes and global trends. These averages are calculated as follows.

$$D^{\{short,long\}}(t) = \frac{\Delta t_{frame}}{t_{win}^{\{short,long\}}} x(t) + \left(1 - \frac{\Delta t_{frame}}{t_{win}^{\{short,long\}}}\right) D^{\{short,long\}}(t - \delta t), \quad (1)$$

where Δt_{frame} is time passed between the measurements and $x(t)$ is the measurement performed at time moment t.

The HMD calculates the frame delivery times as the delay between the reception of the first and the last packets belonging to the same frame and updates the values of D^{short} and D^{long}. Given D^{short} and D^{long}, the algorithm decides whether the frames arrive fast enough, and the bitrate of the video can be increased without an increase in losses, or the frames arrive too slow, and the bitrate should be decreased to avoid frame losses. Otherwise, if the frames arrive as expected, i.e., with the delay around frame periodicity $D_{period} = 1/\nu$, the bitrate shall remain unchanged.

In the developed algorithm, the decisions to increase or decrease the bitrate are based on different frame delivery duration averages. Because the algorithm needs to be sensitive enough and rapidly decrease the bitrate if the channel conditions degrade, it decreases the bitrate if D^{short} becomes lower than a pre-defined threshold D^{lower}. Similarly, the algorithm decides to increase the bitrate only if $D^{long} > D^{upper}$. To improve the stability of the algorithm, we implement a hysteresis-like reset of the averages. The average that indicated the bitrate change shall be reset to a pre-defined value: D^{long} resets to a higher value t_h and D^{short} to a lower value t_l. The pseudocode of the algorithm is presented in Algorithm 1. The algorithm runs each time the video frame is received, and its complexity is $O(1)$.

The aforementioned thresholds D^{lower} and D^{upper} depend on the exact set of bitrates $\mathbb{B}$ available for the client to choose. For example, typically, the bitrates double up the quality ladder [64]. In such a case, the threshold to increase the bitrate shall be $D^{lower} = 0.5D_{Period}$: the frames twice as big as the current ones can be downloaded on time. Similarly, the threshold to decrease the bitrate shall be $D^{upper} = 1.5D_{period}$. However, if the bitrate ladder grows with smaller steps, the thresholds may be closer to one.

Algorithm 1 EVeREst bitrate allocation.

▷ Frames come too slow
if $D^{short} \geq D^{upper}$ **then**
    $D^{short} = t_l$
    return SLOW_DOWN
end if
▷ Frames come very fast
if $D^{long} < D^{lower}$ **then**
    $D^{long} = t_h$
    return SPEED_UP
end if
▷ Doing well
return CONTINUE

4.2. Link Congestion Estimation

Apart from bitrate estimation, the algorithm estimates whether the available channel resources will be enough when the load increases because of a new heavy flow. For that, it estimates the full network capacity over the route between the server and the client and the share of the network allocated for the HMD. We employ the classical idea of the so-called fluid network model [65] and adapt it to modern wireless networks. The algorithm estimates the network capacity as the highest download rate of a small amount of data. Conversely, it estimates the average achievable network throughput as the download rate of a large amount of data. The implementation details of both of these stages are presented in the following sections.

4.2.1. Network Capacity Estimation

We use the P-frames as a small probing portion of data. For that, we split them into N_{gr} equal groups of packets of equal size, except for the last one, which can be smaller. The size of each group shall exceed the maximal portion of data that can be delivered simultaneously in the network. Let each group consist of N_p packets of size L_{MTU} that is the Maximal Transfer Unit (MTU) along the path between the server and the client. The server simultaneously sends all packets belonging to the same group and spreads the groups uniformly over the inter-frame interval. When the client receives all packets belonging to a group k, it calculates the download rate of this group in the following manner.

$$C_k^{est} = \frac{L_{MTU}(N_p - 1)}{recvTime_k[N_p] - recvTime_k[1]}, \tag{2}$$

where $recvTime_k[i]$ is the time instant when the client received the packet i of packet group k.

When all the packet groups are received, the HMD calculates the network capacity as the maximum among the obtained values:

$$C^{est} = \max_k C_k^{est}. \tag{3}$$

The network capacity estimation is performed each time a P-frame is received. Its complexity is $O\left(\frac{BD_{period}}{N_p L_{MTU}}\right)$, where B is the bitrate of the video.

4.2.2. Network Throughput Estimation

The key frames serve as large chunks of data that we use to estimate the network throughput. They are generated relatively seldom. The HMD determines the whole frame's delivery time $\Delta t_{delivery}$ as the delay between the reception of the first and the last packets of the frame and calculates the network throughput T^{est} as:

$$T^{est} = L_{frame} / \Delta t_{delivery}, \tag{4}$$

where L_{frame} is the length of the frame in bytes.

The obtained value is typically smaller than the network capacity because the network resources are shared with the other traffic in the network. The network throughput estimation is performed each time a key frame is received, and its complexity is $O(1)$.

4.2.3. User Number Estimation

We estimate the link congestion as the effective number of user-generated flows active in the network. Note that the real number of flows may differ, as well as their rate may differ from the rate of the considered VR flow. However, these flows generate congestion as the estimated effective number of flows. We calculate the same as in Equation (1) exponentially weighted moving averages of capacity $\bar{C}^{est}$ and throughput $\bar{T}^{est}$ with an averaging window length $t_{win}^{(user)}$ to flatten the fluctuations.

We assume that every user gets an equal share of the network resource, which is the typical mode of operation of the networks [25,26]. Therefore, we can estimate the number of users in the network as follows:

$$N_{users} = \left\lceil \frac{\bar{C}^{est}}{\bar{T}^{est}} \right\rceil, \tag{5}$$

where $\lceil x \rceil$ is the round-up operator. We round the number up because this way, we overestimate the network load. Therefore, we make the algorithm more conservative and prioritize the frame losses over video bitrate. We do this because the frame losses cause video stalling, which is considered to be one of the main factors that cause cybersickness [24].

The number of users is estimated each time a frame is received. Its complexity is $O(1)$.

4.3. Bitrate Adaptation

Once the algorithm decides whether to change the bitrate or not, it ensures that the new bitrate does not exceed the threshold C_{margin} calculated as:

$$C_{margin} = \frac{\bar{C}^{est}}{N_{users} + 1} \tag{6}$$

If the new bitrate is higher than C_{margin}, the algorithm chooses the best one that satisfies the condition $Bitrate \leq C_{margin}$. We introduce this threshold because the client changes the bitrate asynchronously, and the server cannot immediately proceed with the change request. Therefore, we should always expect that another client will enter the network, and the throughput will decrease to C_{margin}. In this way, the HMD always reserves the network resources for a sudden new client. When a new HMD starts consuming its share of the network capacity, the old ones have room to take appropriate measures. We do not have to reserve resources for more than just one client because we assume that the time granularity of the adaptation is small enough to render the probability of more clients entering the network during this interval negligible.

The bitrate clipping is performed each time a frame is received. Its computational complexity is $O(N_{bitrates})$, where $N_{bitrates}$ is the number of bitrates the client can choose. To summarize, the total complexity of the bitrate adaptation is $O\left(1 + \frac{BD_{period}}{N_p L_{MTU}} + N_{bitrates}\right)$. The bitrate adaptation is performed each time a video frame is received, i.e., approximately once in D_{period}. Therefore, the computational complexity of the algorithm is significantly

lower than the complexity of the other processes running at the HMD and can be easily run in real HMDs.

5. Numerical Evaluation

In this section, we use the network simulator NS-3 [66] to evaluate the performance of the developed algorithm and compare it against the bitrate adaptation algorithms for live video streaming known from the literature. In Section 5.1, we describe the considered scenario, the parameters of the algorithm, the metrics we use to evaluate the QoE, and the set of evaluated algorithms. In Section 5.2, we present and analyze the results of the simulation.

5.1. Scenario

We consider a 5G indoor hotspot network [67] where a few (N) clients (VR HMDs) are dropped inside a 50 m circle around a 5G small base station (gNB). The gNB has a wired connection to an MEC server that processes the commands from the HMDs and generates VR video streams. Users start their VR sessions at random moments, and the session duration is a random value that has a uniform distribution from 90 to 110 s. After the session ends, the user waits for some time and starts a new session. The duration of inter-session pause is a random value that has a truncated exponential distribution with an average of 30 s and upper and lower bounds equal to 10 and 60 s, respectively. The duration of the simulation run is 1000 s, and we run 100 simulation runs for each of the considered adaptation algorithms and the number of VR HMDs. The considered scenario, for example, corresponds to an engineering office with multiple users performing modeling in VR or an indoor multiplayer game. In such a scenario, the capacity of a cell may be relatively low, and the changes in the user activity will noticeably influence the network throughput of the other users. Although, in reality, the duration of the sessions and inter-session pauses can be much longer, we assume that scaling up the session parameters by some constant will not change the effects. At the same time, it will slow down the simulations and complicate obtaining statistically significant results.

The videos are generated in one of the following resolutions: 720p, 1080p, 1440p, 2160p. The optimal choice of the bitrate ladder clearly depends on the considered cloud VR deployment scenario and the network profile. However, we aim to develop and evaluate the bitrate adaptation algorithm, so we choose a single bitrate ladder based on the example from the industry. In particular, from [64], we obtain the following bitrate ladder recommended for the considered resolutions: {7.5, 12, 24, 60} Mbps. However, the services, like YouTube, typically compress the content [68–70], so the analysis of the videos stored on YouTube reveals that, in practice, much lower average bitrates are used [71]. Therefore, based on this analysis, we consider the following bitrate ladder: {3.2, 6.1, 12.3, 24.8} Mbps. We use FFmpeg [72] to encode the video using the x265 video encoder with real-time encoding parameters [73], and the server uses RTP over UDP [74] to stream the video to the clients. The frame delay limit (jitter-buffer depth) is 50ms or three frames, which is the maximal tolerable feedback delay in VR scenarios [12]. Table 1 lists other parameters of the scenario.

We evaluate the QoE of the end-users with the following metrics.

1. Number of satisfied sessions: The average number of cloud VR sessions that lost less than $\theta = 2\%$ of the frames (because of channel errors and frame delay violations).
2. Goodput: The total amount of data downloaded in the experiment by cloud VR clients during the satisfied sessions divided by the duration of the experiment.
3. Average bitrate: The average bitrate of the VR sessions.
4. Frame loss ratio: The average portion of frames lost during the VR session.
5. RB usage: The average share of the wireless channel resource blocks utilized during the experiment.
6. Bitrate switching frequency: The average number of bitrate changes during the session divided by the session duration.

We compare the following bitrate allocation algorithms:

1. CBR, 720p: The video with constant resolution 720p is downloaded independently from the channel state.
2. CBR, 1080p: The video with constant resolution 1080p is downloaded independently from the channel state.
3. CBR, 1440p: The video with constant resolution 1440p is downloaded independently from the channel state.
4. CBR, 2160p: The video with constant resolution 2160p is downloaded independently from the channel state.
5. BOLA: The algorithm described in ([55]).
6. Xie et al.: The algorithm described in ([56]).
7. EVeREst: The algorithm developed in this paper.

Table 1. Scenario parameters.

Parameter	Value
Carrier frequency	2 GHz
Bandwidth	20 MHz
Channel model	3GPP Indoor-Office [67]
gNB/UE TX Power	23 dBm
gNB antenna pattern	omnidirectional
gNB height	6 m
UE height	1 m
TTI duration	1 ms
L_{MTU}	1500 bytes
Experiment duration	1000 s
Number of experiment runs	100
Video codec	h265
Frame rate, ν	60 fps
GoP duration	60 frames

We implement the algorithms and set the parameters according to the original papers and the DASH.JS implementation [75]. To adapt them to the cloud VR system, we treat separate video frames as MPEG chunks so that the algorithms stay sensitive enough to the changes in the network state. Note that extremely low buffer levels of cloud VR streaming reduce the BOLA algorithm to its part called DYNAMIC in the original paper. This algorithm estimates the network throughput as the average chunk download rate in a small time window and chooses the maximal video bitrate that is less than the estimated throughput. To take the fluctuations of the frame sizes into account, instead of the average bitrates for each of the video resolutions, we use the effective bandwidth of each resolution, i.e., the minimal network throughput that is enough to satisfy the frame loss ratio requirement. To find it, we conduct the following simple experiment. We transmit the flow over a constant rate point-to-point link and find the minimum data rate of the link that is enough to satisfy the frame loss ratio requirement. The resulting "effective bitrate" ladder is {4.3, 7.9, 16.0, 32.0} Mbps.

The parameters of the EVeREst algorithm are presented in Table 2. We chose them based on the preliminary study. However, their values could be justified in the following way. The short frame delay average shall be sensitive enough to react to the changes at the scale of a single bitrate adaptation interval. In the considered system, the server can change the bitrate only once in a GoP. Therefore, t_{win}^{short} approximately equals the GoP duration (set to 1 s in our experiments). A long frame delay average instead shall track the

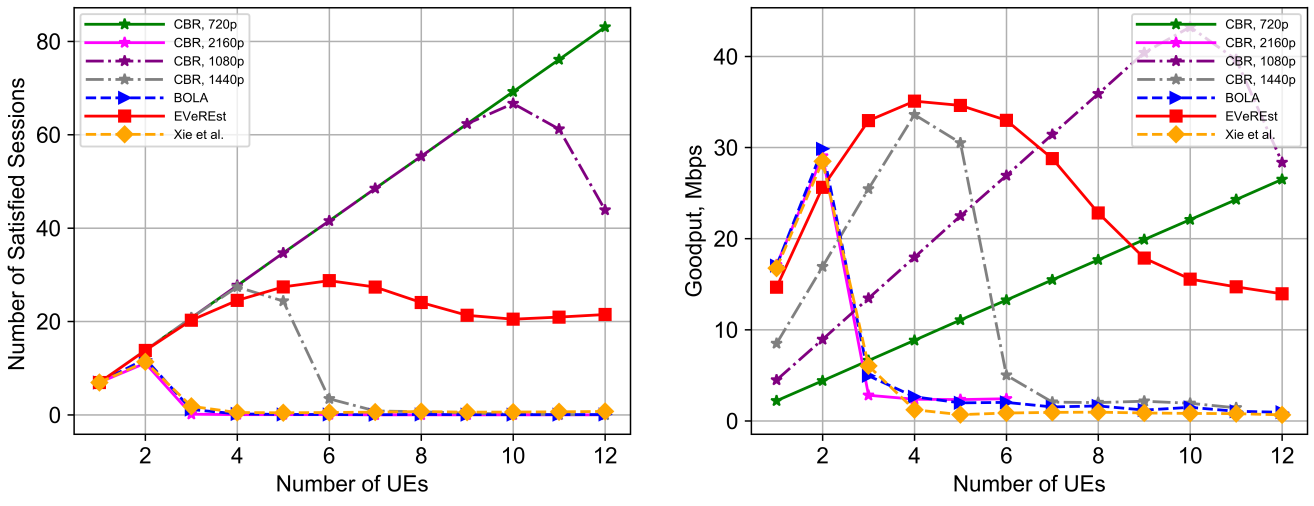
channel state, including wireless channel fluctuations and user activity. We find that, in the considered scenario, 5 s on average provides a good tradeoff between these two factors that influence the frame delivery delays perceived by the client. The same is true for the user dynamics averaging window because the channel fluctuations can also influence this estimation. Finally, the reset values after bitrate increase/decrease shall have the same order as a typical frame delay and introduce a small hysteresis to avoid multiple up/down switches in a row.

Table 2. Enhanced VR bitrate Estimator (EVeREst) algorithm parameters.

Parameter	Value
Short frame delay window, t_{win}^{short}	1 s
Long frame delay window, t_{win}^{long}	5 s
User dynamics window, t_{win}^{user}	5 s
Reset value for bitrate decrease, t_l	5 ms
Reset value for bitrate increase, t_h	20 ms

5.2. Simulation Results

Figure 2 presents the main QoE metrics for the considered algorithms obtained with simulations. Let us first analyze the performance of the CBR algorithms. Figure 2c–d illustrates the fundamental tradeoff between the reliability and the bitrate. The increase in the resource consumption leads to a higher frame loss probability and vice versa. Figure 2a shows that when the number of clients is small, the number of satisfied sessions grows linearly, which means that all the VR sessions are satisfied. However, the further increase in the number of UEs saturates the network (see Figure 2e). Its capacity becomes insufficient to transmit all the data in time; frame losses increase (see Figure 2d), and the QoE degrades fast. No VR sessions can be satisfied after the network capacity is reached, e.g., 5–6 UEs for CBR, 1440p. As for CBR, 720p, from Figure 2e, we can see that, in the considered scenario, it underloads the network. All sessions have low frame loss probability (see Figure 2a,d), but the average bitrate and the corresponding visual quality of the video are unsatisfactory (see Figure 2c). The results of CBR algorithms help us find how much cloud VR flows at a particular bitrate would utilize the network without violating the frame loss requirements and show why we need the bitrate adaptation algorithms.



(**a**) (**b**)

Figure 2. *Cont.*

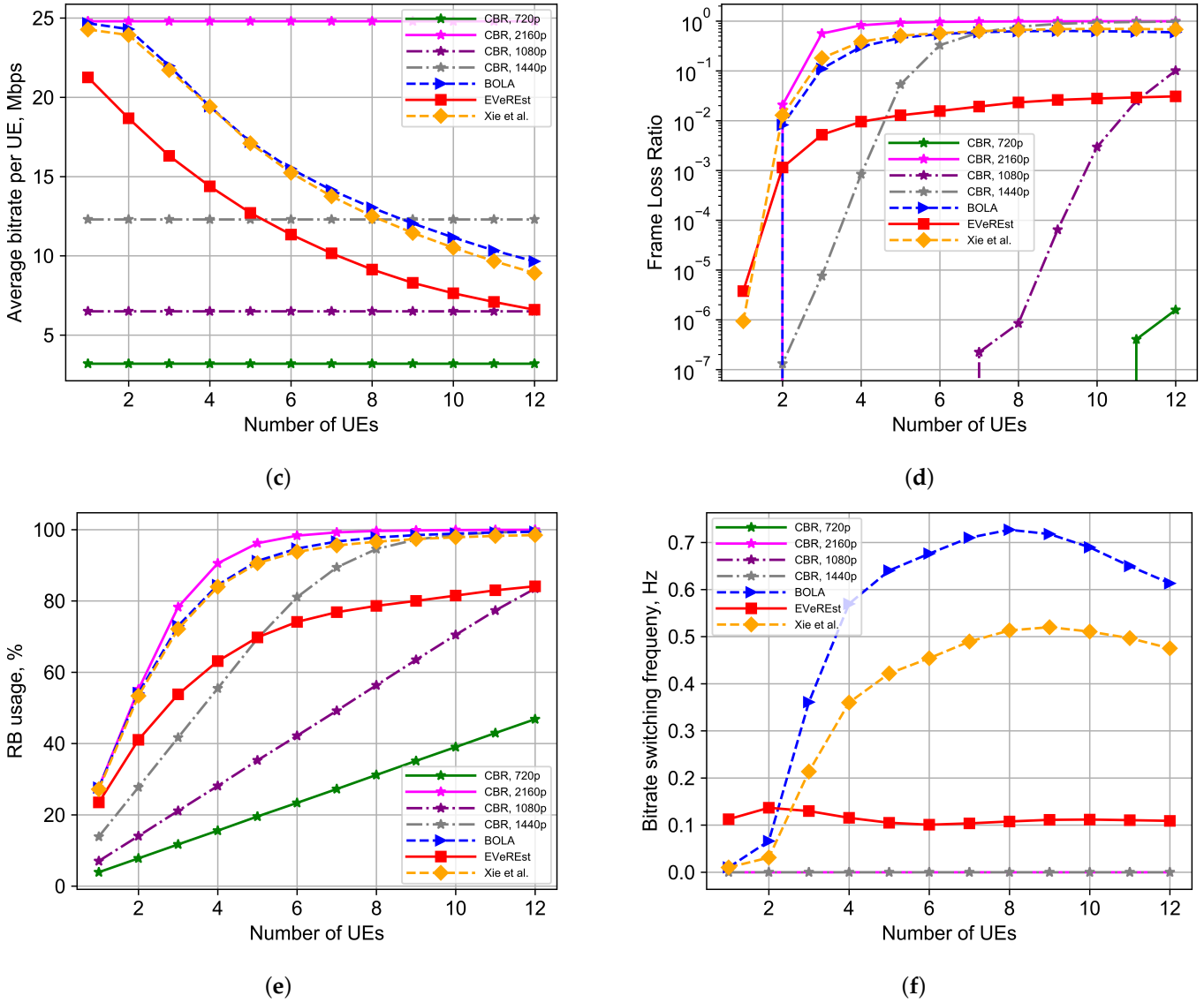


Figure 2. Comparison of the QoE provided by the algorithms. (**a**) The number of satisfied VR sessions. (**b**) The goodput of the system. (**c**) The average bitrate of cloud VR sessions. (**d**) The average frame loss ratio. (**e**) The average share of utilized resource blocks. (**f**) The average bitrate switching frequency.

Next, let us analyze the performance of state-of-the-art bitrate adaptation algorithms designed for live MPEG DASH video streaming. Figure 2a–d indicates that BOLA overestimates the throughput available to the video flow and cannot provide a satisfactory frame loss ratio when more than two UEs are present in the network. We identify two main reasons for it. The first one is the structure of the VR video flows. Because of the video compression, the frame sizes are highly variable, and the load imposed on the network by multiple VR flows fluctuates significantly. The average throughput perceived by the flows is higher than the throughput when a few large frames of different flows have to be downloaded simultaneously. Therefore, the average network throughput gives little insight into the correct choice of bitrate. The second reason is in the network state dynamics. Changes in user activity cause large throughput variations. When a new user starts a VR session, the lack of insurance from such events results in additional frame losses. The algorithm developed in [56] provides similar results, although it pays more attention to the buffer level than to the measured throughput. Furthermore, the fluctuations of the network state cause rather frequent bitrate switches (see Figure 2f); this also contributes to the low QoE.

Unlike other adaptive algorithms, EVeREst satisfies VR clients regardless of the network load (see Figure 2a). Figure 2c,e shows that it provides higher or equal average bitrate and network utilization if compared to CBR algorithms until the network capacity for the corresponding constant bitrate. Although at the edge of the network capacity, the adaptive algorithm typically provides higher frame loss probability and a lower number of satisfied sessions, it achieves an average frame loss ratio lower than the requirement (θ%) up to eight UEs in the network (see Figure 2d). We must highlight that thanks to the hysteresis in bitrate switching decisions, EVeREst provides a much lower bitrate switching frequency than other considered bitrate adaptation algorithms (see Figure 2f). Compared to the other considered adaptation algorithms, the developed algorithm provides up to 10 times higher goodput and much more consistent QoE when the network conditions change. Finally, because EVeREst performs only the bitrate adaptation, it can be easily modified for the other scenario or application that requires such a conservative adaptation. However, if we relax the delay requirements, the conservatism of the developed algorithm will definitely lead to some bitrate underestimation.

6. Conclusions

To facilitate the further development of VR applications, cloud VR technology is introduced. Unlike standalone VR, the cloud VR HMDs only have to track the user's actions and report them to the remote server. Based on the received data, the server renders the virtual environment, encodes its image into the video stream, and sends it back to the HMD. The HMDs use wireless technologies to transmit and receive the data to enable user mobility without restricting their movements.

To provide high QoE for the end-users, cloud VR applications shall avoid image freezes and low visual image quality and minimize the feedback and environment refresh delay. Most VR applications are highly interactive, so the VR video stream cannot be rendered and downloaded to the HMD in advance. Unstable network conditions and hard timing constraints further complicate the problem.

In the paper, we addressed the problem of improving QoE for interactive cloud VR applications through the bitrate adaptation. In contrast to other papers, we assumed no network assistance and designed the application-level client-side bitrate adaptation algorithm. The algorithm considers the bursty nature of the cloud VR traffic and the variations of the network state caused by the wireless channel fluctuations and the changing activity of the users.

We compared the developed algorithm with the bitrate adaptation algorithms designed for live MPEG-DASH-based video streaming that do not assume assistance from the network. The results of the simulations show that in the scenario with multiple cloud VR flows sharing a single wireless network, the developed algorithm provides much higher QoE to cloud VR users in terms of frame delay violation probability and system goodput. In particular, it can achieve up to 10 times higher goodput than the other considered algorithms and achieves much more stable performance in a wide range of network loads.

We see the following directions for future research. Although the algorithm designed in the paper provides rather high QoE to cloud VR users, to further reduce the delays, we have to either reserve some network resources or perform admission control. However, the bursty nature of cloud VR traffic requires finding the tradeoff between the resource overprovisioning and satisfactory QoE.

Author Contributions: Conceptualization, E.K. (Evgeny Khorov) and M.L.; methodology, M.L.; software, E.K. (Evgeny Korneev); investigation, E.K. (Evgeny Korneev) and M.L.; writing—original draft preparation, E.K. (Evgeny Korneev); writing—review and editing, M.L. and E.K. (Evgeny Khorov); visualization, E.K. (Evgeny Korneev); project administration, E.K. (Evgeny Khorov). All authors read and agreed to the published version of the manuscript.

Funding: The research was done at IITP RAS and supported by the Russian Government (Contract No 14.W03.31.0019).

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; nor in the decision to publish the results.

Abbreviations

The following abbreviations are used in this manuscript:

QoS	Quality of Service
QoE	Quality of Experience
HMD	Head-Mounted Display
MPEG DASH	Moving Pictures Experts Group Dynamic Adaptive Streaming over HTTP
VR	Virtual Reality
BS	Base Station
MEC	Mobile Edge Computing
UE	User Equipment
gNB	Next Generation NodeB
GoP	Group of Frames
P-frame	Predicted frame
MTU	Maximum Transmission Unit

References

1. McMahan, R.P.; Bowman, D.A.; Zielinski, D.J.; Brady, R.B. Evaluating display fidelity and interaction fidelity in a virtual reality game. *IEEE Trans. Vis. Comput. Graph.* **2012**, *18*, 626–633. [CrossRef]
2. Sreedhar, K.K.; Aminlou, A.; Hannuksela, M.M.; Gabbouj, M. Viewport-adaptive encoding and streaming of 360-degree video for virtual reality applications. In Proceedings of the 2016 IEEE International Symposium on Multimedia (ISM), San Jose, CA, USA, 11–13 December 2016; pp. 583–586.
3. Moglia, A.; Ferrari, V.; Morelli, L.; Ferrari, M.; Mosca, F.; Cuschieri, A. A systematic review of virtual reality simulators for robot-assisted surgery. *Eur. Urol.* **2016**, *69*, 1065–1080. [CrossRef]
4. Joda, T.; Gallucci, G.; Wismeijer, D.; Zitzmann, N. Augmented and virtual reality in dental medicine: A systematic review. *Comput. Biol. Med.* **2019**, *108*, 93–100. [CrossRef]
5. Su, P.; Wang, S. Virtual reality practice in architecture design. In Proceedings of the 2012 IEEE Symposium on Electrical & Electronics Engineering (EEESYM), Kuala Lumpur, Malaysia, 24–27 June 2012; pp. 98–101.
6. Wolfartsberger, J. Analyzing the potential of Virtual Reality for engineering design review. *Autom. Constr.* **2019**, *104*, 27–37. [CrossRef]
7. GSMA. *Cloud AR/VR Whitepaper*. Technical Report; 2019. Available online: https://www.gsma.com/futurenetworks/wiki/cloud-ar-vr-whitepaper/ (accessed on 8 March 2021).
8. Xu, T.; Sun, Y.; Xia, S.; Li, H.; Luo, L.; Chen, Z. Optimal Bandwidth Allocation with Edge Computing for Wireless VR Delivery. In Proceedings of the 2019 IEEE/CIC International Conference on Communications in China (ICCC), Changchun, China, 11–13 August 2019; pp. 903–907.
9. Zheng, C.; Liu, S.; Huang, Y.; Yang, L. MEC-Enabled Wireless VR Video Service: A Learning-Based Mixed Strategy for Energy-Latency Tradeoff. In Proceedings of the 2020 IEEE Wireless Communications and Networking Conference (WCNC), Seoul, Korea, 6–9 April 2020; pp. 1–6.
10. Sun, Y.; Chen, Z.; Tao, M.; Liu, H. Communications, caching, and computing for mobile virtual reality: Modeling and tradeoff. *IEEE Trans. Commun.* **2019**, *67*, 7573–7586. [CrossRef]
11. Zhang, W.; Chen, J.; Zhang, Y.; Raychaudhuri, D. Towards efficient edge cloud augmentation for virtual reality mmogs. In Proceedings of the Second ACM/IEEE Symposium on Edge Computing, San Jose, CA, USA, 12–14 October 2017; pp. 1–14.
12. Huawei. *White Paper for 5G Cloud VR Service Experience Standards*; Technical Report; 2019. Available online: https://carrier.huawei.com/~/media/CNBGV2/download/products/servies/White-Paper-for-5G-Cloud-VR-Service-Experience-Standards.pdf (accessed on 8 March 2021).
13. Gaddam, V.R.; Riegler, M.; Eg, R.; Griwodz, C.; Halvorsen, P. Tiling in interactive panoramic video: Approaches and evaluation. *IEEE Trans. Multimed.* **2016**, *18*, 1819–1831. [CrossRef]
14. Chakareski, J. VR/AR immersive communication: Caching, edge computing, and transmission trade-offs. In Proceedings of the Workshop on Virtual Reality and Augmented Reality Network, Los Angeles, CA, USA, 25 August 2017; pp. 36–41.
15. MPEG. *ISO/IEC ISO/IEC 23090-2:2019 MPEG-I Coded Representation of Immersive Media–Part 2: Omnidirectional Media Format*; Technical Report; ISO, Headquarters: Geneva, Switzerland, 2019.
16. Hooft, J.V.d.; Vega, M.T.; Petrangeli, S.; Wauters, T.; Turck, F.D. Tile-based adaptive streaming for virtual reality video. *ACM Trans. Multimedia Comput. Commun. Appl.* **2019**, *15*, 1–24. [CrossRef]

17. Ghaznavi-Youvalari, R.; Zare, A.; Fang, H.; Aminlou, A.; Xie, Q.; Hannuksela, M.M.; Gabbouj, M. Comparison of HEVC coding schemes for tile-based viewport-adaptive streaming of omnidirectional video. In Proceedings of the 2017 IEEE 19th International Workshop on Multimedia Signal Processing (MMSP), London, UK, 16–18 October 2017; pp. 1–6.
18. Liu, X.; Deng, Y. Learning-based Prediction, Rendering and Association Optimization for MEC-enabled Wireless Virtual Reality (VR) Network. *arXiv* **2020**, arXiv:2005.08332.
19. Zhang, S.; Tao, M.; Chen, Z. Exploiting Caching and Prediction to Promote User Experience for a Real-Time Wireless VR Service. In Proceedings of the 2019 IEEE Global Communications Conference (GLOBECOM), Big Island, HI, USA, 9–13 December 2019; pp. 1–6. [CrossRef]
20. Vega, M.T.; Liaskos, C.; Abadal, S.; Papapetrou, E.; Jain, A.; Mouhouche, B.; Kalem, G.; Ergüt, S.; Mach, M.; Sabol, T.; et al. Immersive Interconnected Virtual and Augmented Reality: A 5G and IoT Perspective. *J. Netw. Syst. Manag.* **2020**, *28*, 796–826. [CrossRef]
21. Elbamby, M.S.; Perfecto, C.; Bennis, M.; Doppler, K. Toward Low-Latency and Ultra-Reliable Virtual Reality. *IEEE Netw.* **2018**, *32*, 78–84. [CrossRef]
22. Sani, Y.; Mauthe, A.; Edwards, C. Adaptive bitrate selection: A survey. *IEEE Commun. Surv. Tutorials* **2017**, *19*, 2985–3014. [CrossRef]
23. Barman, N.; Martini, M.G. QoE modeling for HTTP adaptive video streaming–a survey and open challenges. *IEEE Access* **2019**, *7*, 30831–30859. [CrossRef]
24. Shahid Anwar, M.; Wang, J.; Ahmad, S.; Ullah, A.; Khan, W.; Fei, Z. Evaluating the factors affecting QoE of 360-degree videos and cybersickness levels predictions in virtual reality. *Electronics* **2020**, *9*, 1530. [CrossRef]
25. Eryilmaz, A.; Srikant, R. Fair resource allocation in wireless networks using queue-length-based scheduling and congestion control. *IEEE/ACM Trans. Netw.* **2007**, *15*, 1333–1344. [CrossRef]
26. Høiland-Jørgensen, T.; Kazior, M.; Täht, D.; Hurtig, P.; Brunstrom, A. Ending the anomaly: Achieving low latency and airtime fairness in wifi. In Proceedings of the 2017 {USENIX} Annual Technical Conference ({USENIX}{ATC} 17), Santa Clara, CA, USA, 12 July 2017; pp. 139–151.
27. Ferrand, P.; Amara, M.; Valentin, S.; Guillaud, M. Trends and challenges in wireless channel modeling for evolving radio access. *IEEE Commun. Mag.* **2016**, *54*, 93–99. [CrossRef]
28. Group, I.T.S.; others. *Series H: Audiovisual and Multimedia Systems: Infrastructure of Audiovisual Services–Coding of Moving Video. High Efficiency Video Coding*; Technical Report; ISO, Headquarters: Geneva, Switzerland, 2019.
29. de Rivaz, P.; Haughton, J. Av1 bitstream & decoding process specification. *Alliance Open Media* **2018**, 681. Available online: https://aomediacodec.github.io/av1-spec/av1-spec.pdf. (accessed on 8 March 2021).
30. Kalbkhani, H.; Shayesteh, M.G.; Haghighat, N. Adaptive lstar model for long-range variable bitrate video traffic prediction. *IEEE Trans. Multimed.* **2016**, *19*, 999–1014. [CrossRef]
31. Xu, J.; Zhou, B.; Zhang, C.; Ke, N.; Jin, W.; Hao, S. The impact of bitrate and GOP pattern on the video quality of H.265/HEVC compression standard. In Proceedings of the 2018 IEEE International Conference on Signal Processing, Communications and Computing (ICSPCC), Xi'an, China, 14–16 September 2018; pp. 1–5.
32. Laghari, A.A.; He, H.; Karim, S.; Shah, H.A.; Karn, N.K. Quality of experience assessment of video quality in social clouds. *Wirel. Commun. Mob. Comput.* **2017**, *2017*. [CrossRef]
33. Vlahovic, S.; Suznjevic, M.; Skorin-Kapov, L. The Impact of Network Latency on Gaming QoE for an FPS VR Game. In Proceedings of the 2019 Eleventh International Conference on Quality of Multimedia Experience (QoMEX), Berlin, Germany, 5–7 June 2019; pp. 1–3.
34. MPEG. *ISO/IEC 23009-1:2014 MPEG-DASH 2nd Edition Specification*; Technical Report; ISO, Headquarters: Geneva, Switzerland, 2014.
35. Bentaleb, A.; Taani, B.; Begen, A.C.; Timmerer, C.; Zimmermann, R. A survey on bitrate adaptation schemes for streaming media over HTTP. *IEEE Commun. Surv. Tutorials* **2018**, *21*, 562–585. [CrossRef]
36. Saleme, E.B.; Covaci, A.; Assres, G.; Comsa, I.S.; Trestian, R.; Santos, C.A.; Ghinea, G. The influence of human factors on 360 mulsemedia QoE. *Int. J. -Hum.-Comput. Stud.* **2020**, *146*, 102550. [CrossRef]
37. Zhou, L.; Wu, D.; Wei, X.; Dong, Z. Seeing isn't believing: QoE evaluation for privacy-aware users. *IEEE J. Sel. Areas Commun.* **2019**, *37*, 1656–1665. [CrossRef]
38. Hu, F.; Deng, Y.; Saad, W.; Bennis, M.; Aghvami, A.H. Cellular-connected wireless virtual reality: Requirements, challenges, and solutions. *IEEE Commun. Mag.* **2020**, *58*, 105–111. [CrossRef]
39. Li, J.; Feng, R.; Liu, Z.; Sun, W.; Li, Q. Modeling QoE of virtual reality video transmission over wireless networks. In Proceedings of the 2018 IEEE Global Communications Conference (GLOBECOM), Abu Dhabi, United Arab Emirates, 9–13 December 2018; pp. 1–7.
40. Duanmu, Z.; Liu, W.; Li, Z.; Chen, D.; Wang, Z.; Wang, Y.; Gao, W. Assessing the Quality-of-Experience of Adaptive Bitrate Video Streaming. *arXiv* **2020**, arXiv:2008.08804.
41. Spiteri, K.; Urgaonkar, R.; Sitaraman, R.K. BOLA: Near-optimal bitrate adaptation for online videos. In Proceedings of the IEEE INFOCOM 2016—The 35th Annual IEEE International Conference on Computer Communications, San Francisco, CA, USA, 10–14 April 2016; pp. 1–9.

42. Huang, T.Y.; Johari, R.; McKeown, N.; Trunnell, M.; Watson, M. A buffer-based approach to rate adaptation: Evidence from a large video streaming service. In Proceedings of the 2014 ACM conference on SIGCOMM, Chicago, IL, USA, 17–22 August 2014; pp. 187–198.
43. Yadav, P.K.; Shafiei, A.; Ooi, W.T. Quetra: A queuing theory approach to dash rate adaptation. In Proceedings of the 25th ACM international conference on Multimedia, Mountain View, CA, USA, 23–27 October 2017; pp. 1130–1138.
44. Li, Z.; Zhu, X.; Gahm, J.; Pan, R.; Hu, H.; Begen, A.C.; Oran, D. Probe and adapt: Rate adaptation for HTTP video streaming at scale. *IEEE J. Sel. Areas Commun.* **2014**, *32*, 719–733. [CrossRef]
45. Rainer, B.; Lederer, S.; Müller, C.; Timmerer, C. A seamless Web integration of adaptive HTTP streaming. In Proceedings of the 20th European Signal Processing Conference (EUSIPCO), Bucharest, Romania, 27–31 August 2012; pp. 1519–1523.
46. Bhattacharyya, R.; Bura, A.; Rengarajan, D.; Rumuly, M.; Shakkottai, S.; Kalathil, D.; Mok, R.K.; Dhamdhere, A. 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, Catania, Italy, 2–5 July 2019; pp. 251–260.
47. Bokani, A.; Hassan, M.; Kanhere, S.; Zhu, X. Optimizing HTTP-based adaptive streaming in vehicular environment using markov decision process. *IEEE Trans. Multimed.* **2015**, *17*, 2297–2309. [CrossRef]
48. Jiang, J.; Sekar, V.; Zhang, H. Improving fairness, efficiency, and stability in http-based adaptive video streaming with festive. In Proceedings of the 8th International Conference on Emerging Networking Experiments and Technologies, Nice, France, 10–13 December 2012; pp. 97–108.
49. Zhou, C.; Lin, C.W.; Zhang, X.; Guo, Z. TFDASH: A fairness, stability, and efficiency aware rate control approach for multiple clients over DASH. *IEEE Trans. Circuits Syst. Video Technol.* **2017**, *29*, 198–211. [CrossRef]
50. Qin, Y.; Jin, R.; Hao, S.; Pattipati, K.R.; Qian, F.; Sen, S.; Wang, B.; Yue, C. A control theoretic approach to ABR video streaming: A fresh look at PID-based rate adaptation. In Proceedings of the IEEE INFOCOM 2017—IEEE Conference on Computer Communications, Atlanta, GA, USA, 1–4 May 2017; pp. 1–9.
51. Krishnamoorthi, V.; Carlsson, N.; Halepovic, E.; Petajan, E. BUFFEST: Predicting buffer conditions and real-time requirements of HTTP (S) adaptive streaming clients. In Proceedings of the 8th ACM on Multimedia Systems Conference, Taipei, Taiwan, 20–23 June 2017; pp. 76–87.
52. Khorov, E.; Krasilov, A.; Liubogoshchev, M.; Tang, S. SEBRA: SAND-enabled bitrate and resource allocation algorithm for network-assisted video streaming. In Proceedings of the 2017 IEEE 13th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), Rome, Italy, 9–11 October 2017; pp. 1–8. [CrossRef]
53. Akyildiz, I.F.; Khorov, E.; Kiryanov, A.; Kovkov, D.; Krasilov, A.; Liubogoshchev, M.; Shmelkin, D.; Tang, S. xStream: A new platform enabling communication between applications and the 5G network. In Proceedings of the 2018 IEEE Globecom Workshops (GC Wkshps), Abu Dhabi, United Arab Emirates, 9–13 December 2018; pp. 1–6.
54. Barakabitze, A.A.; Barman, N.; Ahmad, A.; Zadtootaghaj, S.; Sun, L.; Martini, M.G.; Atzori, L. QoE management of multimedia streaming services in future networks: A tutorial and survey. *IEEE Commun. Surv. Tutorials* **2019**, *22*, 526–565. [CrossRef]
55. Spiteri, K.; Sitaraman, R.; Sparacio, D. From theory to practice: Improving bitrate adaptation in the DASH reference player. *ACM Trans. Multimed. Comput. Commun. Appl. (TOMM)* **2019**, *15*, 1–29. [CrossRef]
56. Xie, L.; Zhou, C.; Zhang, X.; Guo, Z. Dynamic threshold based rate adaptation for HTTP live streaming. In Proceedings of the 2017 IEEE International Symposium on Circuits and Systems (ISCAS), Baltimore, MD, USA, 28–31 May 2017; pp. 1–4.
57. Wei, S.; Swaminathan, V. Low latency live video streaming over HTTP 2.0. In Proceedings of Network and Operating System Support on Digital Audio and Video Workshop, Singapore, 19–20 March 2014; pp. 37–42.
58. Xiao, M.; Swaminathan, V.; Wei, S.; Chen, S. Dash2m: Exploring http/2 for internet streaming to mobile devices. In Proceedings of the 24th ACM international conference on Multimedia, Beijing, China, 19–24 October 2016; pp. 22–31.
59. Peng, H.; Zhang, Y.; Yang, Y.; Yan, J. A hybrid control scheme for adaptive live streaming. In Proceedings of the 27th ACM International Conference on Multimedia, Nice, France, 21–25 October 2019; pp. 2627–2631.
60. Öztürk, E.; Silhavy, D.; Einarsson, T.; Stockhammer, T. Low latency DASH-more than just spec: DASH-IF test tools. In Proceedings of the 11th ACM Multimedia Systems Conference, Istanbul, Turkey, 8–11 June 2020; pp. 353–356.
61. Chen, S.; Zhang, Y.; Peng, H.; Yan, J. A Joint Bitrate and Buffer Control Scheme for Low-Latency Live Streaming. In Proceedings of the International Conference on Intelligent Science and Big Data Engineering, Nanjing, China, 18–20 October 2019; Springer: Berlin/Heidelberg, Germany, 2019; pp. 369–380.
62. Xiao, X.; Wang, W.; Chen, T.; Cao, Y.; Jiang, T.; Zhang, Q. Sensor-Augmented Neural Adaptive Bitrate Video Streaming on UAVs. *IEEE Trans. Multimed.* **2020**, *22*, 1567–1576. [CrossRef]
63. Tian, S.; Yang, M.; Zhang, W. A Practical Low Latency System for Cloud-Based VR Applications. In Proceedings of the International Conference on Communications and Networking in China, Shanghai, China, 29 November–1 December 2019; pp. 73–81.
64. YouTube Recommeded Video Coding Settings. Available online: https://support.google.com/youtube/answer/1722171 (accessed on 25 November 2020).
65. Prasad, R.; Dovrolis, C.; Murray, M.; Claffy, K. Bandwidth estimation: metrics, measurement techniques, and tools. *IEEE Netw.* **2003**, *17*, 27–35. [CrossRef]
66. The Network Simulator ns-3. Available online: https://www.nsnam.org (accessed on 25 November 2020).

67. 3GPP. *Study on Channel Model for Frequencies from 0.5 to 100 GHz (3GPP TR 38.901 Version 15.0.0 Release 15)*; 3GPP, Headquarters: Sophia Antipolis, France, 2018.
68. Laghari, A.A.; He, H.; Khan, A.; Kumar, N.; Kharel, R. Quality of experience framework for cloud computing (QoC). *IEEE Access* **2018**, *6*, 64876–64890. [CrossRef]
69. Laghari, A.A.; He, H.; Shafiq, M.; Khan, A. Impact of storage of mobile on quality of experience (QoE) at user level accessing cloud. In Proceedings of the 2017 IEEE 9th International Conference on Communication Software and Networks (ICCSN), Guangzhou, China, 6–8 May 2017; pp. 1402–1409.
70. Laghari, A.A.; He, H.; Shafiq, M.; Khan, A. Assessment of quality of experience (QoE) of image compression in social cloud computing. *Multiagent Grid Syst.* **2018**, *14*, 125–143. [CrossRef]
71. Ragimova, K.; Loginov, V.; Khorov, E. Analysis of YouTube DASH Traffic. In Proceedings of the 2019 IEEE BlackSeaCom, Sochi, Russia, 3–6 June 2019.
72. FFmpeg h265 Encoding Guide. Available online: https://trac.ffmpeg.org/wiki/Encode/H.265 (accessed on 25 November 2020).
73. x265 Encoding Profiles. Available online: https://x265.readthedocs.io/en/release_3.3/presets.html (accessed on 25 November 2020).
74. Wang, Y.-K.; Sanchez, Y.; Schierl, T.; Wenger, S.;Hannuksela, M.M. *RTP Payload Format for High Efficiency Video Coding* ; RFC 7798; IETF, Headquarters: Fremont, CA, USA, 2016.
75. DASH-IF. dash.js. Available online: https://github.com/Dash-Industry-Forum/dash.js (accessed on 18 Septermber 2020).