On-the-Fly Edge Transcoding for Interactive VR

Andreas Casparsen, Federico Chiariotti, and Jimmy Jessen Nielsen

***Abstract*—The recent rise of the Cloud Virtual Reality (VR) paradigm, in which VR frames are streamed from a remote server to the user's Head-Mounted Display (HMD), poses some interesting challenges from a networking perspective. VR flows are high-throughput and have strict latency requirements. In this paper, we analyze on-the-fly edge transcoding, which follows the opposite philosophy from the more common slicing approach: instead of adapting resource allocation to the content, we compress the frames to fit into the allocated bandwidth, guaranteeing limited latency. Our results show that this strategy is effective in maintaining low latency, but the picture quality is highly dependent on the computing power of the Base Station (BS).**

I. Introduction

The development of eXtended Reality (XR) technology has significantly accelerated over the past few years, helped along by the strong push by tech giants like Meta and Steam to commercialize its applications and the COVID-19 lockdowns. Recent forecasts [1] predict that the number of XR devices sold yearly will exceed 100 million by 2025, and industrial and manufacturing applications are emerging along with commercial, recreative ones. One of the most interesting developments in the field is Cloud XR, which allows for higher Quality of Experience (QoE) and lighter Head-Mounted Display (HMD) devices: while the first generation of headsets was self-contained, performing all the necessary processing and rendering locally, the significant computing power of the Cloud and network edge can be used to improve quality, relying on the 5G and beyond infrastructure to stream the rendered flow [2]. However, while streaming passive Virtual Reality (VR) content can use buffered content to provide a smooth experience even in wireless channels [3], interactive content cannot be buffered, and the consequences of network impairments can be much more serious [4]. Excessive delays can produce a mismatch between visual input and proprioceptive sensations, which causes *cybersickness* [5].

The concept of *motion-to-photon latency* [6], i.e., the delay between a control action by the user and its effect in the virtual environment, as shown to the user through the HMD, has emerged as a Key Performance Index (KPI) to avoid these issues. At the same time, we need to consider the strain that the XR content can put on the Radio Access Network (RAN), which 3GPP has recently analyzed [7], due to the extremely challenging requirements for future XR systems [8]. Furthermore, even the use of the intra-frame encoding option, which is supposed to output a Constant Bit Rate (CBR) stream, cannot completely eliminate the random component in the frame size [9]: as the frame size is only partially predictable, and might fluctuate by about 30% of the average frame size, provisioning capacity is complex [10]. In order to fulfil the latency requirement, the resource allocation algorithm would need to integrate a prediction on the future frame sizes [11].

Naturally, there are two possible kinds of adaptations to provide reliable service: the first is network slicing, i.e., allocating enough resources to the VR user to maintain the required QoE, and the second is to compress the content so as to fit into the available bandwidth. The second approach entails a certain level of control over the VR application [12], and direct communication with the Cloud server that generates the content. However, there are also some issues: firstly, neither the HMD nor the Cloud server knows the future resource allocation, which is performed by the Base Station (BS). The communication between the client and server is also affected by the propagation delay, as the server might not be close to the user, making the adaptation relatively sluggish. However, recent works have demonstrated the capability of Mobile Edge Computing (MEC)-enabled BSs to set the quality for a video flow, and even transcode it to adapt its bitrate [13].

In this work, we combine the prediction model from [9] with an edge transcoding model to solve this issue: we define a theoretical model of the components of motion-to-photon latency, then propose an adaptive transcoding solution that can compress frames to allow them to be delivered within the required latency. We derive some analytical results on the distribution of the latency and picture quality for both cases, then compare our solution with an ideal, fully adaptive network slicing system, as well as with a legacy system without transcoding. Our simulation results show that transcoding is an effective way to provide low-latency service under variable network conditions, at the cost of a slightly lower picture quality, but that its effectiveness is highly dependent on the amount of available computational resources on the edge node.

The rest of this paper is divided as follows: first, Sec. II presents the system model and the basic components of the motion-to-photon latency. We then evaluate the performance of the proposed scheme in Sec. III. Finally, Sec. IV concludes the paper and presents some possible avenues of future work.

II. System Model

We consider a Cloud VR service, in which a VR user experiences a virtual environment through an HMD. As high-quality content requires significant computing capabilities, the generation and encoding of the content is performed by a remote server in the Cloud, which streams it to the HMD. We then consider a stream of individual *frames* of size $F(k)$, where k is the frame index, which are generated at a frame rate

Andreas Casparsen (aca@es.aau.dk), Federico Chiariotti (fchi@es.aau.dk), and Jimmy Jessen Nielsen (jjn@es.aau.dk) are with the Department of Electronic Systems, Aalborg University, Denmark. This work was partly funded by the European Commission as part of the IntellIoT project, under the H2020 framework grant no. 957218.

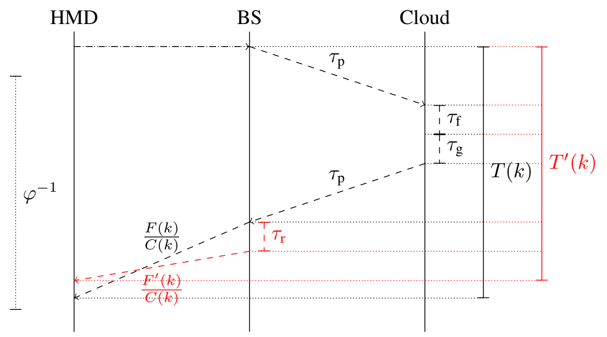


Fig. 1: Components of the motion-to-photon latency for the considered system model.

of φ Frames per Second (FPS). The VR content is encoded using the H.264 Periodic Intra Refresh option, which makes the frames quasi-CBR [9].

We then list the components of the motion-to-photon latency, which are shown in Fig. 1: first, the motion of the VR user is detected using inertial sensors and transmitted from the HMD to the Cloud, where the results of the actions are calculated and the new frame is generated. In VR, the size of the feedback packets from the HMD to the server is negligible[1]. Feedback packets from the user are generated with a frequency φ (i.e., one per frame), so the time between the instant when the server receives the feedback on the user's movement and the generation of the next frame is a uniform random variable $\tau_f \sim \mathcal{U}\left(0, \varphi^{-1}\right)$. The one-way propagation time from the BS to the Cloud server is denoted as τ_p, and the computation time required by the server to generate and encode the content is denoted as τ_g. The propagation time from the HMD to the BS is considered negligible here, but it can be easily added to the model. After the new frame of size $F(k)$ is generated, it is transmitted to the BS through the core network. Assuming that the back- and fronthaul links are high-capacity fiber optic, this also requires a time τ_p, independently of the frame size. However, the bottleneck of the connection is in the RAN, i.e., in the wireless connection between the BS and the HMD. If we consider a network slicing algorithm to allocate capacity $C(k)$ to frame k, the resulting motion-to-photon latency $T(k)$ is given by:

$$T(k) = \tau_f + 2\tau_p + \tau_g + \frac{F(k)}{C(k)}. \tag{1}$$

While the first two terms are constant, and mostly independent from resource provisioning, the third one is highly dependent on the variable frame size, as even the quasi-CBR encoding results in frame size differences up to 30% around the average [9]. Maintaining a motion-to-photon latency below 20 ms, the target set by IEEE [14] for XR content, would then require an extremely fast adaptation of the network slicing for the RAN, providing the correct amount of resources for each frame, or a huge overprovisioning to accommodate even the largest frames.

If adapting the resource allocation on such a short timescale is infeasible, which would be the case in current systems, in which the network slicing allocation is updated only after several seconds, there is another option to maintain $T(k)$ below the threshold. A MEC-enabled BS can decode and transcode the oversized frames, i.e., the ones for which $T(k) > T_{\max}$, as defined by (1), performing a rougher quantization of the video parameters and reducing $F(k)$, at the cost of a slightly reduced picture quality. Naturally, this operation also requires some time, which depends on the amount of computational resources available for the task, denoted as $Z(k)$ in the following. If the number of operations for the decoding and transcoding is $\mathcal{E}$ (which we assume is independent from the frame size itself), the overall computation time $\tau_r(k)$ is given by $\tau_r(k) = \mathcal{E}Z(k)$. If we denote the transcoded size as $F'(k)$, we get the new motion-to-photon latency $T'(k)$, shown in red in the figure:

$$T'(k) = \tau_f + 2\tau_p + \tau_g + \mathcal{E}Z(k) + \frac{F'(k)}{C(k)}. \tag{2}$$

This on the fly transcoding can be adapted based on the amount of available computing and communication resources, which the BS knows, as it is the entity that manages them. We can then set the constraint on the compression level that allows us to meet the motion-to-photon latency target $T_{\max}$ as:

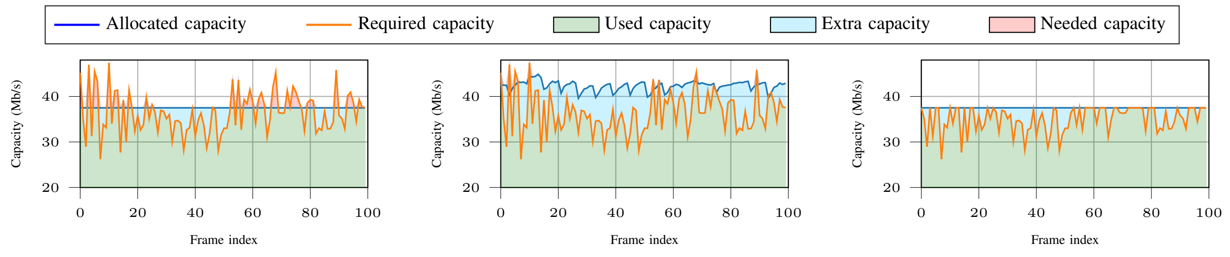
$$F'(k) \leq (T_{\max} - \varphi^{-1} - 2\tau_p - \tau_g - \mathcal{E}Z(k)) \cdot C(k). \tag{3}$$

We consider the worst-case scenario in which the motion happened right after the previous feedback packet was sent, so the feedback recording delay is φ^{-1}. To maintain stability, we also need to impose a further condition on $C(k)$, i.e., $\mathbb{E}[C(k)] \geq \varphi \mathbb{E}[F'(k)]$. Depending on the frame rate and on the motion-to-photon latency constraint $T_{\max}$, this condition might be stricter or looser than (3). We can then give the condition in which the transcoding is performed, compressing the original frame to fit into the latency budget:

$$F(k) > C(K) \min\left(T_{\max} - (\tau_g + 2\tau_p + \varphi^{-1}), \varphi^{-1}\right). \tag{4}$$

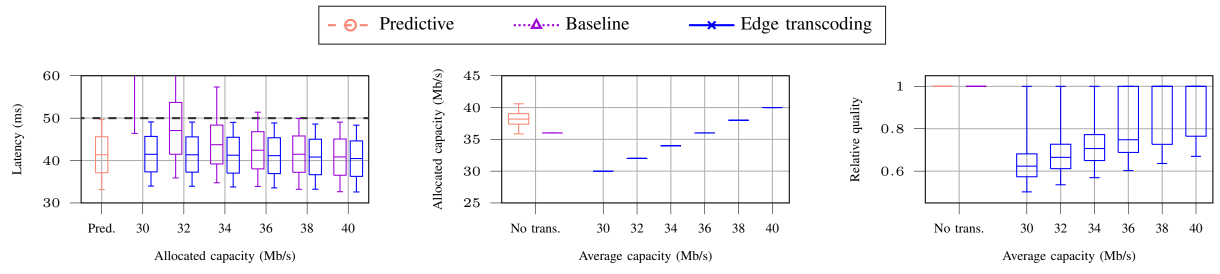
We give a visual representation of the effect of different schemes in Fig. 2, which compares the effects of predictive slicing and transcoding. The baseline, shown in Fig. 2a, is a classical slicing scheme, which assigns a fixed capacity to the VR flow: there is a clear trade-off between overprovisioning (and consequently wasting some resources, shown in blue) and latency, as some frames are too big to be transmitted with the required motion-to-photon latency. Predictive slicing, shown in Fig. 2b, takes a conservative approach, attempting to predict the size of the next frame and provisioning resources for the worst case. In general, using prediction can allow the scheme to waste fewer resources than setting a large fixed capacity. Finally, edge transcoding is shown in Fig. 2c: as frames are compressed to fit into the allocated capacity, latency is limited, at the cost of a lower picture quality.

[1]In Augmented Reality (AR), we would also have to consider the video frames sent from the user to the server, as the camera feed needs to be transmitted to integrate the virtual content into it.



(a) Fixed rate slicing (b) Predictive slicing (c) Edge transcoding

Fig. 2: Scheduling methods utilizing a fixed data rate compared against an adaptive, and the transcoding principle.



(a) Motion-to-photon latency. (b) Assigned capacity. (c) Picture quality.

Fig. 3: Performance of the schemes as a function of the targeted capacity percentile.

III. Simulation Settings and Results

In this section we perform a set of simulation scenarios to analyze the performance of different scheduling schemes. These are a predictive slicing scheme presented in [9], which sets a constant rate for N frames at a time. A baseline that utilizes a fixed rate for all frames, and the proposed transcoding scheme also utilizes a fixed rate. All schemes strive to adhere to a pre-defined maximum motion-to-photon latency $T_{\max}$: the transcoding scheme ensures that the latency bound will be respected, reducing the picture quality when necessary as determined in (4). The size of the transcoded frame is determined in (3), and we simply define the relative picture quality of compressed frames as $\frac{F'(k)}{F(k)}$, without going into more complex quality metrics [3]. The simulations use the parameters from Table I, unless otherwise stated.

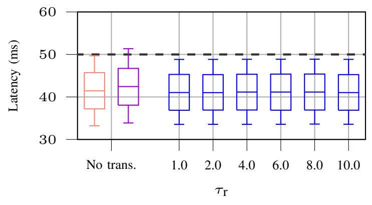
First, we analyze the effect of changing the capacity in the fixed slicing. Fig. 3 compares the performance of the schemes in terms of latency, resource efficiency, and picture quality. As Fig. 3a shows, the baseline scheme is able to meet the motion-to-photon latency deadline for 95% of frames (corresponding to the upper whisker of the boxplot) if $C \geq 38$ Mb/s, while the predictive scheme is able to achieve a slightly better performance with the same average capacity as the baseline scheme. As Fig. 3b shows, the predictive slicing scheme has a variable capacity, which depends on the predicted size of the next frame. If we consider the transcoding scheme, we notice that it can achieve a latency that is guaranteed to be below $T_{\max}$ even at the lowest capacity, as the degree of freedom in this case is the picture quality: when capacity is insufficient, there is a trade-off between latency and quality, and the transcoding scheme sacrifices the latter to guarantee the former, as Fig. 3c shows. In general, a higher capacity corresponds to a higher picture quality, while the latency remains the same. The opposite applies for the other schemes, where quality remains the same, but latency improves.

We can also consider the effect of τ_{r} on the efficiency of the edge transcoding scheme: the computing power available at the edge node is a critical factor, as longer transcoding times take up a larger portion of the motion-to-photon latency budget, reducing the quality. With the specified parameters, the time available for the downlink transmission of each frame, equal to $T_{\max} - (2\tau_{\text{p}} + \tau_{\text{g}} + \varphi^{-1})$, is 13.33 ms: naturally, if $\tau_{\text{r}} \geq 13.33$ ms, transcoding is impossible at any level of compression. Even if τ_{r} is lower, it has a significant effect on the performance of the scheme, as Fig. 4 shows. If we consider latency distribution in Fig. 4a, we can notice that the transcoding scheme still delivers all frames within the latency budget. However, this comes at the cost of the picture quality, as Fig. 4b shows: as τ_{r} increases, the quality is significantly degraded, and the median transcoded frame for $\tau_{\text{r}} = 10$ ms is about 25% of the original quality. While actual QoE is nonlinear, and might be better represented by a logarithmic curve, this is still an extreme degradation, and computing power is crucial for the scheme.

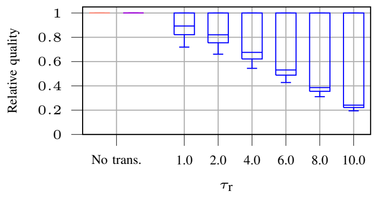
Finally, we can take a deeper look at performance by analyzing the full Cumulative Distribution Functions (CDFs)

TABLE I: Scenario parameters.

Par.	Value	Par.	Value	Par.	Value
$T_{\max}$	50 ms	τ_{g}	4 ms	τ_{p}	8 ms
τ_{r}	3 ms	C (fixed)	36 Mb/s	p_s (pred.)	0.99
S (pred.)	6 frames	N (pred.)	6 frames	τ (pred.)	1 frame
R	30 Mb/s	φ	60 FPS	Content	Virus Popper

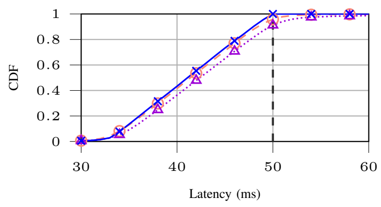


(a) Motion-to-photon latency.

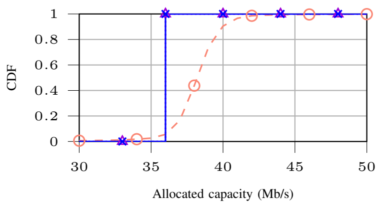


(b) Picture quality.

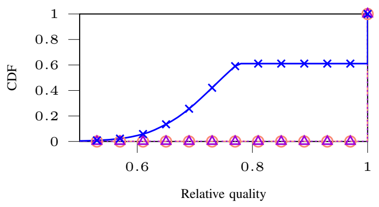
Fig. 4: Performance of the schemes as a function of the transcoding time τ_r.



(a) Motion-to-photon latency.



(b) Assigned capacity.



(c) Picture quality.

Fig. 5: Performance CDFs for the 4 different schemes using parameters from Table I.

of the main performance metrics, using the parameters from Table I. Fig. 5a shows the CDF of the latency, and as we discussed above, the edge transcoding scheme is the only one that achieves 100% on-time delivery, where Predictive and Baseline only achieve around 95% and 90%, respectively. Naturally, the performance of predictive slicing has a cost, which is clearly visible from Fig. 5b: while the fixed rate schemes always have $C = 36$ Mb/s, the required capacity for predictive slicing is higher, and changes over time depending on the predicted frame size. Finally, the cost of the transcoding scheme is presented in Fig. 5c: while the other schemes always transmit frames at full quality, over 60% of frames are transcoded and compressed, with a compression ratio between 0.5 and 0.8. Additionally, combining predictive allocation and transcoding yields a negligible performance increase compared to the fixed rate scheduling with transcoding. That is when the fixed rate capacity is set to be equal that of the average of the predictive. An improvement in average quality can be found, but is $< 0.1\%$. We believe this is related to the poor performance of the fixed rate scheme being derived from poor resource provisioning. Once a frame is delayed, the consequence is the delay of subsequent frames. This chain effect is not present in transcoding, as frames are guaranteed to be transmitted within the time window.

IV. Conclusions and Future Work

In this paper, we proposed an edge transcoding scheme that can reduce quality to maintain a limited motion-to-photon latency. Our transcoding scheme is complementary to predictive network slicing, which adapts the amount of resources allocated to a VR flow to the predicted requirements in terms of capacity, and chooses a different degree of freedom in the trade-off between capacity requirements, motion-to-photon latency, and picture quality in VR.

There are several avenues for possible future work on the subject, which include more realistic models of the MEC-enabled BS and the slicing operation, as well as a joint optimization of the communication and computing resources.

References

[1] K. Chauhan, “Global AR/VR forecast,” Counterpoint Research, Tech. Rep., 2021. [Online]. Available: https://www.counterpointresearch.com/xr-vrar-headset-shipments-grow-10-times-cross-100-million-units-2025/ (Accessed: April 2022)
[2] A. Mehrabi, M. Siekkinen, T. Kämäräinen, and A. yä Jääski, “Multi-tier CloudVR: Leveraging edge computing in remote rendered virtual reality,” *ACM Trans. Multimedia Comp., Comm. App. (TOMM)*, vol. 17, no. 2, pp. 1–24, 2021.
[3] F. Chiariotti, “A survey on 360-degree video: Coding, quality of experience and streaming,” *Comp. Comm.*, vol. 177, pp. 133–155, 2021.
[4] K. Bouraqia, E. Sabir, M. Sadik, and L. Ladid, “Quality of experience for streaming services: measurements, challenges and insights,” *IEEE Access*, vol. 8, pp. 13 341–13 361, 2020.
[5] J.-P. Stauffert, F. Niebling, and M. E. Latoschik, “Latency and cybersickness: Impact, causes, and measures. a review,” *Frontiers in Virtual Reality*, vol. 1, p. 582204, 2020.
[6] M. Yang, J. Zhang, and L. Yu, “Perceptual tolerance to motion-to-photon latency with head movement in virtual reality,” in *Picture Coding Symp. (PCS)*. IEEE, 2019.
[7] 3GPP, “Study on XR (Extended Reality) Evaluations for NR,” 3GPP, Technical Report (TR) 38.838, 2021.
[8] F. Hu, Y. Deng, W. Saad, M. Bennis, and A. H. Aghvami, “Cellular-connected wireless virtual reality: Requirements, challenges, and solutions,” *IEEE Comm. Mag.*, vol. 58, no. 5, pp. 105–111, 2020.
[9] M. Lecci, F. Chiariotti, M. Drago, A. Zanella, and M. Zorzi, “Temporal characterization of XR traffic with application to predictive network slicing,” in *23rd Int. Symp. World of Wireless, Mobile and Multimedia Net. (WoWMoM)*. IEEE, 2022.
[10] S. Salehi, A. Alnajim, X. Zhu, M. Smith, C.-C. Shen, and L. Cimini, “Traffic characteristics of Virtual Reality over edge-enabled Wi-Fi networks,” *arXiv preprint arXiv:2011.09035*, 2020.
[11] F. Wei, S. Qin, G. Feng, Y. Sun, J. Wang, and Y.-C. Liang, “Hybrid model-data driven network slice reconfiguration by exploiting prediction interval and robust optimization,” *IEEE Trans. Net. Serv. Mgmt.*, 2021.
[12] A. Arfaoui, R. El-Azouzi, M. Haddad, and E. Sabir, “Flexible network slicing assisted 5G for video streaming with effective and efficient isolation,” in *32nd Int. Teletraff. Cong. (ITC)*. IEEE, 2020, pp. 156–164.
[13] B. Jedari, G. Premsankar, G. Illahi, M. Di Francesco, A. Mehrabi, and A. Ylä-Jääski, “Video caching, analytics, and delivery at the wireless edge: a survey and future directions,” *IEEE Comm. Surv. Tut.*, vol. 23, no. 1, pp. 431–471, 2020.
[14] “IEEE Standard for Head-Mounted Display (HMD)-Based Virtual Reality (VR) Sickness Reduction Technology,” *IEEE Std 3079-2020*, 2021.