Traffic Characteristics of Virtual Reality over Edge-enabled Wi-Fi Networks

Seyedmohammad Salehi† Abdullah Alnajim† Xiaoqing Zhu‡ Malcolm Smith‡
Chien-Chung Shen† Leonard Cimini†

†University of Delaware ‡Cisco Systems
{salehi, alnajim, cshen, cimini}@udel.edu, {xiaoqzhu, mmsmith}@cisco.com

Abstract—Virtual reality (VR) is becoming prevalent with a plethora of applications in education, healthcare, entertainment, etc. To increase the user mobility, and to reduce the energy consumption and production cost of VR head mounted displays (HMDs), wireless VR with edge-computing has been the focus of both industry and academia. However, transferring large video frames of VR applications with their stringent Quality of Service (QoS) requirements over wireless network requires innovations and optimizations across different network layers. In order to develop efficient architectures, protocols and scheduling mechanisms, the traffic characteristics of various types of VR applications are required. In this paper, we first compute the theoretical throughput requirements of an ideal VR experience as well as a popular VR HMD. We then examine the traffic characteristics of a set of VR applications using an edge-enabled Wi-Fi network. Our results reveal interesting findings that can be considered in developing new optimizations, protocols, access mechanisms and scheduling algorithms.

Index Terms—Virtual Reality, Traffic Characterization, WLAN, Edge Computing, IEEE 802.11.

I. INTRODUCTION

Virtual reality (VR) is a technology with myriads of use cases in education, healthcare, entertainment, gaming, to name a few. VR applications are sensitive to latency and packet loss, and thus, require reliable and low-latency access to powerful graphics processors to render the projected high-resolution scenes. Due to the limited graphics processing capability of the VR head-mounted displays (HMDs) and to reduce the energy consumption of HMDs for graphics processing and the HMD’s production cost, for a satisfying immersive experience, VR HMDs are typically connected, via a cable, to a server with a powerful graphics card for video frame rendering. In this architecture, although the HMD can access a server with advanced graphics capabilities, it becomes mobility limited.

Newer HMDs (e.g., Oculus Quest) are equipped with not only more capable GPU modules to independently render the video contents, but also wireless capability to facilitate untethered mobility. Although these HMDs allow more mobility, the on-device rendering consumes more energy and cannot keep up with the emerging very high-resolution contents. Nonetheless, since these HMDs are equipped with wireless modules, most of the rendering can be offloaded to an edge server in close proximity. Compared to cloud computing, edge/fog computing [1]–[3] brings the computation capability closer to the end user, which not only reduces the amount of traffic transferred over the Internet but also reduces the end-to-end latency. This architecture, however, incurs a heavy traffic load on the wireless access network: large, delay-sensitive VR physical layer protocol data units (PPDUs) with high video frame rate in the downlink (DL) and high frequency user’s tracking information (TI) in the uplink (UL), whereby most existing Wi-Fi standards (e.g., IEEE 802.11n and 802.11ac) are unable to guarantee the desired Quality of Service (QoS) requirements, especially in dense environments.

To meet the current and emerging traffic demands, IEEE 802.11 has evolved to 802.11ax [4], [5] (employing OFDMA, beamforming, multiuser (MU) MIMO, and more efficient access mechanisms) and 802.11ad/ay (employing millimeter-wave) [6]. However, various VR applications have different traffic load, delay and bandwidth requirements so that there exists a trade-off between satisfying the QoS requirements of VR applications and the overall throughput of the wireless networks. Thus, it is important to understand the traffic characteristics of different VR applications in order to design architectures, protocols and access mechanisms for Wi-Fi networks that satisfy the QoS requirements of specific VR applications even in dense deployments.

At a high level, VR applications can be classified into single-party (SP) and multi-party (MP). Intuitively, in an SP-VR application where HMD renders the frames, no communication with cloud or edge is necessary. Fig. 1 depicts the communications between HMD, edge and cloud for SP-VR and MP-VR applications. Frame rendering can either take place on the HMDs or at the edge servers. However, unlike SP-VR experience, in MP-VR, each party also communicates with a cloud server (either directly or via an edge server) to transfer and receive the shared user information (e.g., voice, updated avatar positions, etc.). It is also possible for the edge servers to communicate directly with each other in a peer-to-peer manner. For instance, in cooperative VR games, multiple players share noticeable amount of background in each scene. In such situations, the background layer can be rendered, cached and reused at the edge servers and the “delta” images rendered in real-time on the HMD [7]. Similarly, Furion [8] separated the image rendering between the edge server and the mobile device for Google’s Daydream VR HMDs. Vivo by Han et al. [9] optimized the data transfer by predicting user

1Application layer packet data unit (APDU) with video content.
Frame rendering can take place on the HMDs or on the edge servers. In SP-VR applications, no communications with cloud is necessary. However, in MP-VR applications, edge servers/HMDs send/receive shared user information (e.g., voice). For instance, user P2 in Zone 1 communicates with user P3 in Zone 2, although their frame rendering mode may be different.

As the UL transmissions of user’s TIs are very frequent, in multi-party applications, wireless channel access contentions are unavoidable. Ahn et al. [11] proposed to reduce the priority of the TI frames so as to reduce contention in 802.11ax networks. The authors further proposed an algorithm to dynamically adjust the frequency of the 802.11ax trigger frames and the video frame rate based on the predefined thresholds of packet loss ratio. Due to the large physical layer transmission delay of the current Wi-Fi standards, Zhang et al. [12] employed multiple Wi-Fi network interface cards to reduce the transmission delay and jitter of VR frames. Abari et al. [13] investigated the use of mmWave for a low-latency untethered VR experience.

Since VR gaming can be a paradigmatic example of applications requiring complex 3D rendering, Prem sankar et al. [14] employed an open-source cloud gaming platform, GamingAnywhere [15], to compare the response delay of edge computing and cloud computing. To achieve an eye-like VR experience, in [16], the authors analyzed the required throughput by comparing it to the real eye experience. In this paper, we compute the required throughput of VR applications for an eye-like experience as well as an acceptable video quality experience based on the specifications of one of the popular VR HMDs (i.e., Oculus Quest). Moreover, we present a case study of running different VR applications on an open source remote VR display (i.e., ALVR) and characterizing their traffic statistics. Our results are useful for developing new architectures and scheduling mechanisms for future Wi-Fi standards.

The rest of this paper is organized as follows. We explain the theoretical aspects of the required VR throughput in the next section. In Section III, we describe our testbed, network settings and traffic collection choices. In Section IV, we present the statistics of the experiments along with discussions. Section V concludes this paper with future research directions.

II. VR THROUGHPUT REQUIREMENTS

A. Preliminaries

The resolution or quality of an image is measured by the number of pixels that represent it. Since resolution depends on the size of the display, pixel per inch (PPI) can be more representative of the image quality. PPI is defined to be the ratio of the number of pixels along a dimension of a display to the length of that dimension.

\[
PPI = \frac{D_{px}}{D_{in}} \text{ or } \frac{W_{px}}{W_{in}} \text{ or } \frac{H_{px}}{H_{in}},
\]

(1)

where \(D_{px}, W_{px}, \) and \(H_{px}\) denote the number of pixels along the diagonal, the width, and the height of a display, respectively. Likewise, \(D_{in}, W_{in}, \) and \(H_{in}\) denote, in inches, the length of the diagonal, the width, and the height of the same display, respectively.

Image quality also depends on the distance of the display from the eyes as illustrated by Fig. 2. In that sense, pixels...
per degree (PPD) is a more accurate measurement of image quality which denotes the number of covered pixels per unit degree from an eye’s fovea to the display. As the distance between the screen and the eye fovea decreases, the size of the display area, covered by a unit degree, becomes smaller, so that eyes can detect more details of the displayed image. To compute PPD, we should first compute the Field of View (FoV) which is the maximum amount of the world observable by eyes and depends on the viewing distance and the screen size. For instance, to compute the horizontal FoV (FoV_h), as illustrated in Fig. 3, we can create two right triangles from the distance of the eye to the screen and, based on the following equation, compute FoV_h as 2\theta.

\[
FoV_h = 2 \times \tan^{-1} \left( \frac{W}{2 \times L} \right) \tag{2}
\]

where FoV_h denotes the horizontal Field of View in degree\(^2\), and L denotes the distance between the user’s eyes and the screen in inches. We can now compute PPD_h as follows.

\[
PPD_h = \frac{W_{px}}{FoV_h} \tag{3}
\]

From the Eq. 3, with the knowledge of FoV_h and PPD_h, we can compute the W_{px}. Computation of the vertical FoV, FoV_v, and vertical PPD, PPD_v, is similar. Fig. 4 shows the optimal viewing distance for different display sizes with different resolutions. The visual acuity threshold is computed based on 1 arcminute for normal vision (20/20) [17].

For an eye-like experience can be computed based on the following equation.

\[
T_{eye} = W_{px} \times H_{px} \times b_d \times R_f \tag{4}
\]

where from Eq. 3, \(W_{px} = FoV_h \times PPD_h\) and \(H_{px} = FoV_v \times PPD_v\); \(b_d\) denotes bit depth used to represent a color image (e.g., to represent three colors each with 256 different intensities, 3×8 bits are required), and \(R_f\) denotes the number of frames per second (FPS). With a typical PPD of 200 [16], 36 bits for full color representation, and frame rate of 150 FPS, the number of bits that is required to represent this image quality per second is 3.888 Terra bits.

B. Throughput Requirements for Oculus Quest

Oculus Quest, a VR HMD created by the Oculus VR division of Facebook, has a dual OLED panel with resolution of 1,440×1,600 per eye. Therefore, the required number of bits to represent a scene per second in this HMD for both eyes can be computed as follows.

\[
T_{HMD} = 2 \times W_{px} \times H_{px} \times b_d \times R_f \tag{5}
\]

Therefore, with 24 bits for each pixel’s color, and frame rate of 72 FPS, the number of required bits per second is 2×1440×1600×24×72 ∼ 8 Gb. In the case of offloading the rendering to the edge server, we need compression algorithms to encode a video frame on the edge server, transfer the frame to the HMD over the network and decode it at the HMD. Since there is a trade off between the maximum video compression and the compression time, the encoding techniques can be tuned. Assuming an average compression factor range of [200-300] [16], [18], the required throughput will be in the range of [26.5-40] Mbps. Moreover, with higher compression factors and rendering optimizations (mentioned in Section I), this required throughput can be reduced.

III. CASE STUDY: OCULUS QUEST

A. Network Topology and Settings

Oculus Quest is designed to operate in wireless mode for untethered communication with the cloud servers. Although Oculus Quest can be used in standalone mode for video frame

Fig. 5: The topology of our testbed.
rendering, owing to the wireless capability, frame rendering can also be offloaded to an edge server. To characterize the impact of edge-enabled VR applications over Wi-Fi networks, we deployed a server with a high-end graphics card (AMD Radeon VII) to serve as the edge server. As depicted in Fig. 5, this server is connected to a wireless router (ASUS TM-1900AC) via an Ethernet cable. An Oculus Quest HMD is also connected to this router wirelessly using channel 161 of 20 MHz. Oculus Quest can operate on dual bands and support Wi-Fi standards up to IEEE 802.11ac.

To be able to capture the Wi-Fi frames transmitted between the HMD and the wireless router and decode all the layers of the Internet protocol stack, the wireless network is intentionally made “open access.” Specifically, a laptop is connected to this wireless network to sniff the Wi-Fi frames by using Wireshark. Also, the Ethernet frames between the server and the wireless router are captured using Wireshark running on the server. In this experiment, we used the open source remote VR display called Air Light VR (ALVR), where the server application runs on the edge server and the client application runs on the HMD.

B. VR Traffic Collection

Since VR games can be representatives of complex 3D VR applications, for our analysis, we selected three VR games in ascending required video resolution: Rec Room, The Lab, and War Robots. Rec Room belongs to the category of multiplayer social interactive games, where it allows users to talk, create games and play with each other. War Robots and The Lab are single-player games. The former is a high-resolution game and the latter offers a series of different game types accessible through a hub room. Specifically, we play Xortex, a delay-sensitive shooting game. Traffic traces, source code, and the post-processing scripts can be accessed online [19].

C. Frame Rendering on the Edge Server

As illustrated in Fig. 7, in a typical VR experience, the HMD (periodically) collects and sends small APDUs of tracking information containing the position and orientation of the user’s head and controllers\(^4\) to the edge server to retrieve the appropriate video frame with size equal to the HMD’s display resolution. Although such data is usually small, it has a high frequency (e.g., 200-1000 Hz). Consequently, based on the received position and orientation data, the edge server locates, renders, encodes and transmits the target video frame back to the HMD. Such high-resolution video frames will be fragmented into small MAC layer packet data units (MPDUs), traverse over the networks (Ethernet and Wi-Fi) and reassembled at the HMD. After receiving the video frame, the HMD decodes, renders the updated haptics positions on the received video frame as overlay and submits the final rendered video frame to each lens.

D. H.264 vs. H.265 vs. H.266

High efficiency video coding (HEVC) or H.265 is originally designed for more efficient transferring of video frames over

\(^3\)This measurement can be either from timestamps included in the packet headers or from specific packets to solicit the clock of the other machine.

\(^4\)A user’s hands can also act as controllers.
and H.265 for the game of Rec Room. In this section, we compare the performance of H.264 over the Internet [20]. Due to unavailability of VVC encoder at the time of this study, we enable more efficient transfer of high quality video contents. Versatile video coding (VVC) or H.266, has recently been standardized specifically for encoding videos (4K to 16K) as well as 360° videos. With 30-50% better compression rate compared to HEVC, VVC will enable more efficient transfer of high quality video contents over the Internet [20]. Due to unavailability of VVC encoder at the time of this study, we compare the performance of H.264 and H.265 for the game of Rec Room.

IV. RESULTS AND DISCUSSION

In this section, we compare the statistics of running different VR games with video rendering at the edge-enabled wireless network. Each game is played for 90 seconds starting from the connection establishment between the AVLR client and server (which brings us to the Steam Hall, where we choose a specific game), to load the game and play it as long as the remaining time permits. Then, we present the statistics of different target bitrates and codecs for the game of Rec Room.

A. Application layer vs. MAC layer

In this experiment, the target video bitrate is 30 Mbps which reserves a 60 KB buffer size for packets at the client side (i.e., the HMD). Since both H.264 and H.265 use variable bitrate encoding, the actual instantaneous bitrate might be less or more than 30 Mbps depending on the codec type, motion level, frame interval, etc. The frame rate is set to 72 FPS which results in generation of video frames every 13.8 ms.

All voice and video application layer PDUs are sent via UDP protocol. However, owing to the uncrowded network, MAC layer retransmissions and forward error correction (FEC) at the ALVR client (on HMD), the packet loss ratio in our experiment is almost zero.

Since application video frames constitute more than 90% of downlink traffic, in this section, we first present the statistics pertaining to generation of the video frames at the application layer and then present the statistics at the MAC layer. Fig. 8 illustrates the quantity and size of generated video frames at the Application layer (i.e., APDUs) for different games with HEVC encoding. With higher graphics demand, larger video frames are generated. For instance, for Rec Room, 47% of overall video frames are larger than 30 KB. For The Lab and War Robots, 68% and 74% of video frames, respectively, are larger than 30 KB. This difference is also shown on the instantaneous rate subfigures where due to the higher graphics demands of the War Robots game, the application layer rate is mostly near 30 Mbps (after second 30 when the game starts).

In contrast, for Rec Room, the encoder can reduce the bitrate owing to lower graphics demands.

Fig. 9 illustrates the interval of video frames generated at the Application (i.e., APDU interval) and MAC layers (i.e., MPDU interval) for different games. Since the frame rate is set to 72 FPS, on average, every 13.8 ms a video frame is generated at the Application layer which gets fragmented into smaller 1442-byte frames at the MAC layer (1400-byte data, 8-byte UDP header, 20-byte IP header, and 14-byte MAC header and trailer). The majority of inter-frame intervals at the MAC layer is in the range [0.03-0.05] ms.

As mentioned earlier, in MP-VR applications each party also communicates with a cloud server (either directly or through an edge server) to transfer and receive the shared video frames.
user information (e.g., voice, updated avatar positions, etc.). Fig. 10 illustrates the cloud communications in DL and UL for Rec Room which starts from time 40 seconds when there is interaction among the players. Depending on the number of parties engaged in a game (or a portion of the game), the amount of DL traffic changes. In this example, the amount of DL traffic is more than 10 times higher than the UL traffic.

B. The impact of different codecs and bitrates on the network

As video frames of VR applications are large and delay-sensitive, transferring the rendered video frames from the edge server to an HMD incurs a heavy load on the wireless network. Therefore, the remote VR display application should strike a balance among the image quality of video frames, the frame size, and the end-to-end delay. For instance, with a given resolution, higher compression factor produces smaller frame sizes but might exceed the acceptable end-to-end delay. For the AMD graphics cards, ALVR employs Advanced Media Framework (AMF) SDKs\(^5\) for video rendering and encoding. Specifically, “Ultra-low latency” usage mode is predefined for video game streaming in which the number of bits per video frame is upper-bounded to the configured target bitrate.

Fig. 11 depicts a comparison between the video frame sizes and the number of frames for H.264 and H.265 encoders at 30 Mbps target bitrate, and H.265 at 60 Mbps target bitrate. With the same target bitrate of 30 Mbps, the average video frame size generated by H.265 is 24.8 KB which is \(~48\%\) smaller than that of H.264 (i.e., 48 KB). As graphics demand of Rec Room can be met with lower target bitrates, the AMD graphics cards, ALVR employs Advanced Media Framework (AMF) SDKs\(^5\) for video rendering and encoding.

\(^5\)https://github.com/GPUOpen-LibrariesAndSDKs/AMF
Fig. 11: Video frame size and instantaneous rate at the application layer for different encoders and target bitrates. The average rate at the Application layer for playing Rec Room with H.264-30Mbps, H.265-30Mbps, and H.265-60Mbps is 21.41, 14.13, and 22.09 Mbps, respectively.

Fig. 12: Delay components of different encoders with same target bitrate for 90 seconds of playing Rec Room. Although H.265 is superior in the rendering on the edge server and decoding on the HMD, rendering at the HMD is faster with H.264 encoder.

Increasing the target bitrate will not increase the clarity of images that constitute the game scenes. However, in high graphics demanding games, higher bitrates can increase the video quality of the games. Therefore, adjustment of target bitrate depending on the VR application can increase the game quality of overall network efficiency.

As mentioned earlier, to generate a video frame, the game engine requires the updated position and velocity of each engaged player. As an example, for Rec Room, Fig. 12 illustrates the CDF statistics of delay components for H.264 and H.265 at target bitrate of 30 Mbps. Total delay is round trip which is computed from the time a tracking information APDU is sent to the edge server until the time that the video frame (scene) is displayed at the HMD screen(s). Since the frequency of transmitted tracking information packets might be higher than the video frame rate, the game engine only considers the most updated tracking information in generating the next video frame. Therefore, rendering delay at the edge server is coupled with tracking information transmission delay. Although H.265 is superior to H.264 in most of the delay compartments, rendering delays on the HMD for H.264 are smaller than H.265. In addition, unlike H.265, H.264 is unable to encode the video frames within the bounds of frame rate (on average encoding with H.264 takes longer than $\frac{1}{72} = 13.88$ ms). Table I contains the average and 95th percentile statistics of delay components for different games and target bitrates. Running Rec Room with the target bitrate of 15 Mbps results in reduced quality of scenes. However, for Rec Room, there is no noticeable advantage in increasing the target bitrate to 60 Mbps compared to 30 Mbps. Therefore, adjustment of the
and production cost of the VR HMD. In this paper, we require the QoS of VR applications, rendering can be offloaded to the edge server which reduces the energy consumption and production cost of the VR HMD. In this paper, we theoretically computed the throughput requirements of an eye-like experience as well as a popular VR HMD (i.e., Oculus Quest). We then presented the realistic network statistics of playing different VR games in an edge-enabled wireless network. These network statistics can be used in other trace-driven simulations to develop new architectures, protocols, access mechanisms and scheduling algorithms.

Future research can focus on the optimization of (1) tracking information intervals (e.g., frequency adjustment based on the instantaneous VR experience and frame rate), and (2) target bitrate based on different games and instantaneous game situations. In the MAC layer, we plan to investigate a new MAC layer access category for VR traffic and new scheduling mechanisms so as to satisfy the QoS requirements of VR applications while not starving other traffic types.

ACKNOWLEDGEMENT

This study has been supported by a gift grant from Cisco Systems, Inc.

REFERENCES

[1] F. Bonomi, R. Milito, P. Natarajan, and J. Zhu, “Fog computing: A platform for internet of things and analytics,” in Big data and internet of things: A roadmap for smart environments. Springer, 2014, pp. 169–186.

[2] S. Chen, T. Zhang, and W. Shi, “Fog computing,” IEEE Internet Computing, vol. 21, no. 2, pp. 4–6, 2017.

[3] Y. C. Hu, M. Patel, D. Sabelka, N. Sprecher, and Y. Young, “Mobile edge computing—a key technology towards 5G.”

[4] B. Bellalta, “IEEE 802.11ax: High-efficiency WLANs,” IEEE Wireless Communications, vol. 23, no. 1, pp. 38–46, 2016.

[5] E. Khorov, A. Kryianos, A. Lyakhov, and G. Bianchi, “A Tutorial on IEEE 802.11ax High Efficiency WLANs,” IEEE Communications Surveys & Tutorials, 2018.

[6] S. Sun, G. R. MacCartney, and T. S. Rappaport, “A novel millimeter-wave channel simulator and applications for 5G wireless communications,” in 2017 IEEE International Conference on Communications (ICC), IEEE, 2017, pp. 1–7.

[7] Y. Li and W. Gao, “MUVR: Supporting multi-user mobile virtual reality with resource constrained edge cloud,” in 2018 IEEE/ACM Symposium on Edge Computing (SEC). IEEE, 2018, pp. 1–6.

[8] 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, 2019.

[9] B. Han, Y. Liu, and F. Qian, “Virio: visibility-aware mobile volumetric video streaming,” in Proceedings of the 26th Annual International Conference on Mobile Computing and Networking, 2020, pp. 1–13.

[10] W. Zhang, J. Chen, Y. Zhang, and D. Raychaudhuri, “Towards efficient edge cloud augmentation for virtual reality MMOGs,” in Proceedings of the Second ACM/IEEE Symposium on Edge Computing, 2017, pp. 1–14.

[11] J. Ahn, Y. Y. Kim, and R. Y. Kim, “Virtual reality-wireless local area network: Wireless connection-oriented virtual reality architecture for next-generation virtual reality devices,” Applied Sciences, vol. 8, no. 1, p. 43, 2018.

[12] H. Zhang, A. Elmokashfi, and P. Mohapatra, “WiFi and Multiple Interfaces: Adequate for Virtual Reality?” in 2018 IEEE 24th International Conference on Parallel and Distributed Systems (ICPADS). IEEE, 2018, pp. 220–227.
[13] 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.

[14] G. Premsankar, M. Di Francesco, and T. Taleb, “Edge computing for the Internet of Things: A case study,” IEEE Internet of Things Journal, vol. 5, no. 2, pp. 1275–1284, 2018.

[15] C.-Y. Huang, K.-T. Chen, D.-Y. Chen, H.-J. Hsu, and C.-H. Hsu, “GamingAnywhere: The first open source cloud gaming system,” ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM), vol. 10, no. 1s, pp. 1–25, 2014.

[16] E. Bastug, M. Bennis, M. Médard, and M. Debbah, “Toward interconnected virtual reality: Opportunities, challenges, and enablers,” IEEE Communications Magazine, vol. 55, no. 6, pp. 110–117, 2017.

[17] “Optimum HDTV viewing distance,” https://en.wikipedia.org/wiki/Optimum_HDTV_viewing_distance, accessed: September 16, 2020.

[18] J. Pavlic and J. Burkeljca, “FFmpeg based Coding Efficiency Comparison of H. 264/AVC, H. 265/HEVC and VP9 Video Coding Standards for Video Hosting Websites,” International Journal of Computer Applications, vol. 975, p. 8887, 2019.

[19] “VR Traffic Characteristics over Edge-enabled Wireless Networks,” https://www.eccis.udel.edu/~salehi/vr.html.

[20] “Versatile Video Coding Explained,” https://www.ericsson.com/4f92d7/assets/local/reports-papers/ericsson-technology-review/docs/2020/versatile-video-coding-explained.pdf, accessed: October 2, 2020.