Adaptive Streaming and Packet Scheduling for VR Video

by

Haopeng Wang

Thesis submitted to the University of Ottawa
in partial fulfillment of the requirements
for the Doctorate in Philosophy degree
in Electrical and Computer Engineering

uOttawa

School of Electrical Engineering and Computer Science
Faculty of Engineering
University of Ottawa
Ottawa, Canada

© Haopeng Wang, Ottawa, Canada, 2024

Abstract

Over the past few years, the surge in VR (Virtual Reality) video traffic on networks has been remarkable. Nonetheless, a key challenge remains: ensuring a top-notch quality of experience (QoE) for VR video playback, especially when network bandwidth is limited. Prior studies have mainly focused on tile-based adaptive bitrate (ABR) streaming operating at the application layer on the server/client side to improve QoE, using single viewport prediction to conserve bandwidth. However, single-viewpoint prediction models face limitations due to uncertainties linked with head movement, making it difficult to handle sudden user motions effectively. To overcome these constraints, we propose a lightweight multimodal spatial-temporal transformer architecture, which generates multiple viewpoint trajectories and their corresponding probabilities while leveraging historical trajectory information. Consequently, we introduce a multi-agent reinforcement learning (MARL)-based ABR algorithm that capitalizes on multiple viewport prediction for VR video streaming at the application layer. Our algorithm strives to optimize various QoE objectives under diverse network conditions. To address the ABR problem, we formulate it as a Decentralized Partially Observable Markov Decision Process (Dec-POMDP) problem. To tackle this effectively, we develop a MAPPO (Multi-Agent Proximal Policy Optimization) algorithm within a centralized training and decentralized execution (CTDE) framework.

Meanwhile, we also improve QoE at the network layer by utilizing network resources in different network nodes during VR video streaming. We present an innovative system called tile-weighted rate-distortion (TWRD) packet scheduling optimization, which takes advantage of viewpoint prediction. The system dynamically assigns weights to tiles and their corresponding packets using the probability of viewpoint prediction. Due to limited bandwidth, the problem of packet scheduling arises, requiring the determination of

which packets should be dropped. To address this challenge, we formulate the problem as an optimization task, taking into account error propagation in the video. Our system leverages the weighted rate-distortion information of packets and applies dynamic programming techniques to design an optimal packet scheduling scheme. By selectively dropping packets at network nodes, our proposed system effectively reduces network congestion and enhances the overall performance of VR video streaming systems operating within bandwidth limitations.

Acknowledgements

It is my honor to do research with Prof. Abdulmotaleb El Saddik, who gave me the opportunity to be a member of MCRlab. I am deeply grateful for the opportunity to work with him. Having his passion, vision and motivation inspired me to overcome obstacles and work for my goal. He offered me insightful advice not just in research, but also in life. I greatly appreciate his kindness, humor, and friendship. Working with him under his guidance is an honor.

My thanks also go out to all of the MCRlab members. I am grateful for their advice and guidance regarding my thesis. Their questions in lab meetings always clarify my path forward. Furthermore, I would like to thank Haiwei Dong for his patient guidance and selfless help.

Finally, I would like to express my gratitude to my parents for their unconditional support, both mentally and financially. Their love, kindness, and understanding have inspired me to keep going with my thesis and prepare myself for the future.

Contents

List of Figures

List of Tables

List of Abbreviations

AR Augmented Reality

ABR Adaptive Bitrate

VR Virtual Reality

ML Machine Learning

FoV Field of View

HMD Head Mounted Display

DASH Dynamic Adaptive HTTP Streaming

LSTM Long Short-Term Memory

CNN Convolutional Neural Network

PPD Pixed per Degree

MARL Multi-Agent Reinforcement Learning

CTDE Centralized Training and Decentralized Execution

MAPPO Multi-Agent Proximal Policy Optimization

SARL Single-Agent Reinforcement Learning

QoE Quality of Experience

MDP Markov Decision Process

Dec-POMDP Decentralized Partially Observable Markov Decision Process

TWRD Tile-Weighted Rate-Distortion

EWRD Equal-Weighted Rate-Distortion

RL Reinforcement Learning

A3C Asynchronous Advantage Actor-Critic

FFN Feed-Forward Network

GOP Group of Pictures

DDPG Deep Deterministic Policy Gradient

NLP Natural Language Processing

GAE Generalized Advantage Estimation

PPO Proximal Policy Optimization

IPPO Independent Proximal Policy Optimization

MADDPG Independent Multi-Agent Deep Deterministic Policy Gradient

STD Standard Deviation

Chapter 1

Introduction

1.1 Background

The emergence of 360-degree video, also known as VR (Virtual Reality) video or panoramic video, has caused a revolution in the realm of immersive digital experiences [1]. In contrast to traditional 2D videos, VR video content delivers a mesmerizing and seamless visual encounter, enveloping viewers within an extensive 360-degree or 180-degree viewing range.

The quality of VR video playback has been much improved by recent hardware and rendering technique developments, which has led to its broad use as an engaging way to draw audience attention in a variety of contexts, such as concerts, museums, and sporting events. A growing number of VR videos are being shared on social media sites like Facebook, which indicates how popular virtual reality is becoming [2]. Unlike traditional media, VR videos evoke an emotional response and engage viewers in a way that traditional media can't. Due to this, VR videos typically receive more online clicks and are viewed longer than traditional videos. VR videos have become a permanent fixture in our digital world because they generate remarkable experiences and boost engagement.

VR videos are engaging pieces of media that use multiple cameras to capture footage

from every angle, effectively encapsulating a complete view of the surrounding environment within a 360-degree 3D sphere (see Fig. 1.1). VR environments can be created more easily with this remarkable content. Two-dimensional screens, such as mobile devices and computers, offer viewers the freedom to move their viewing direction, resulting in a panoramic-like experience that lets them explore the video from any vantage point. Furthermore, the viewer's engagement can be further enhanced by displaying the content on specialized devices like head-mounted displays (HMDs) and projectors arranged in a spherical or partially spherical configuration [3].

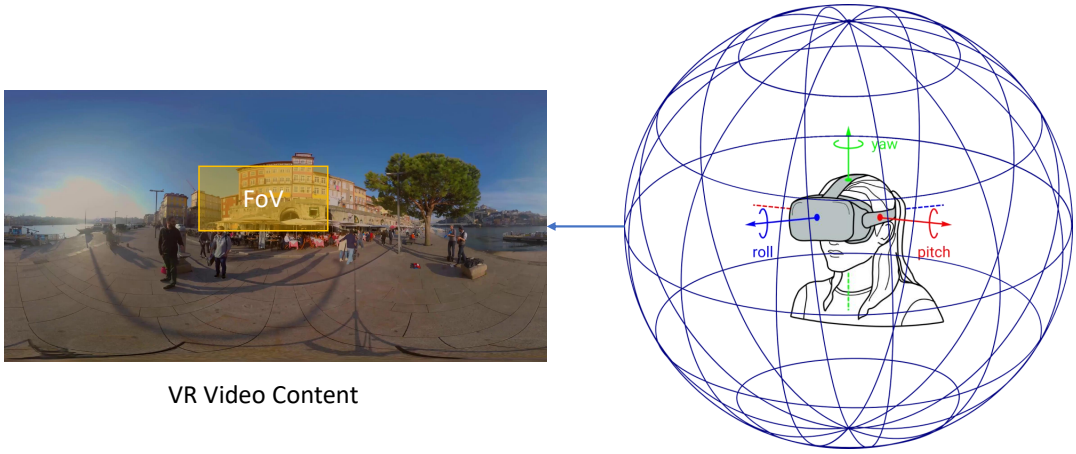

Figure 1.1: A user donning a VR headset engages in immersive viewing of a VR video, where the content dynamically adapts to the user's head movements.

VR has a wide range of applications in fields such as education, real estate, medicine, economics, and more. Due to restrictions on physical contact, the COVID-19 pandemic has played a significant role in the development and adoption of these technologies. There is no doubt that augmented reality (AR) is being used in a wide variety of industries, with notable applications including home shopping[4] and holographic conference meetings [5]. VR videos and immersive technologies have become increasingly popular in the education sector due to the proliferation of online content and their affordability and efficiency [6], [7]. VR media allows viewers to explore and interact with certain areas of interest by simply rotating their heads. This immersive and interactive experience increases user engagement and creates more compelling visual stories. The use of VR video streaming

in education is acknowledged as a critical tool for fostering students' technical creativity and supporting their research. In addition, it promotes an active learning environment instead of the traditional passive learning environment so that complex subjects, ideas, or concepts can be better understood. By encouraging deeper learning and providing a variety of new opportunities, this dynamic approach to teaching raises the bar for the entire educational experience. Therefore, it increases students' desire to learn more about the topics they are studying, allowing them to experiment and gain a deeper understanding.

The rapid progress of these technologies is being propelled by major corporations, exemplified by Meta (formerly Facebook) and their concept of the "Metaverse." Spearheaded by Mark Zuckerberg, the CEO of Meta, the company actively champions the idea of a virtual universe that integrates social connections, entertainment, gaming, health, work, education, and commerce. The Metaverse provides immersive experiences that transcend the limitations of physical reality, and technologies like Blockchain, NFTs, and Cryptocurrency play vital roles in enabling virtual business operations and asset ownership within this digital realm. Embracing this vision, JP Morgan launched a virtual lounge on the Blockchain-based VR platform called "Decentraland" on February 15th, 2022, tapping into the vast market potential of the Metaverse. VR video plays a pivotal role in this vision by providing a groundbreaking avenue for storytelling, empowering video makers and content creators to narrate their tales in a profoundly immersive and interactive manner. This innovative form of video content can be presented as live broadcasts or pre-recorded experiences, finding applications in virtual tours, education, entertainment, and more. In training and education, VR video allows users to engage with real-world scenarios within a safe and controlled environment. The utilization of HMDs to view VR videos is expected to continue surging in popularity, as users seek to immerse themselves further in captivating virtual experiences.

To summarize, the benefits of VR video and VR encompass fostering creativity and interest in education, generating diverse business and employment prospects within the

Metaverse, offering a virtual communication platform that closely emulates face-to-face interactions, and delivering exceptional entertainment experiences that surpass traditional forms of entertainment.

1.2 Motivation

The evolution of VR video can be divided into three distinct stages, as illustrated in Table 1.1 [8]. Each stage is characterized by its specific technical requirements, encompassing aspects such as resolution, field of view (FoV), refresh rate, bitrate, pixel per degree (PPD) and more.

Estimated time for commercial use	**2022-2024** (First stage)	**2025-2027** (Second stage)	**2028-2032** (Third stage)
Content resolution	7680x3840 (8K)	11520x5760 (12K)	23040x11520 (24K)
Terminal resolution	1920x1920	3840x3840	9600x9600
Field of view	110°x110°	120°x120°	150°x150°
Bit per color	8	10	12
Encoding standard	H.265	H.265	H.266
Refresh rate	60 fps	90 fps	120 fps
Pixel per degree	17	32	64
Uncompressed bit rate	42.47 Gb/s	179.16 Gb/s	1146.62 Gb/s
Transmitting bit rate H.265(250:1) H.266(400:1)	170Mb/s (Full-view) 42 Mb/s (FoV)	712 Mb/s (Full-view) 319 Mb/s (FoV)	2.87 Gb/s (Full-view) 1.99 Gb/s (FoV)
Uplink bit rate	30Kb/s	50Kb/s	80Kb/s

Table 1.1: The evolution of VR video. Three phases are defined with different technical specifications.

Stage 1 (2022-2024): During this initial stage, the client screen for each eye could achieve a resolution of up to 1920x1920, enabling users to experience a PPD of 17 with streamed 8K content. The uplink bit rate, which represents the required transmission

rate from the client (i.e., head-mounted device) to the video server, is relatively low. This is because uplink data primarily consists of feedback signals and collected sensory data. Although most technical requirements for this stage have already been met, it is estimated that an additional two years are necessary to release mature products that fully align with the specified specifications.

Stage 2 (2025-2027): In the second stage, significant improvements are expected in screen resolution, content quality, and chip performance. As a consequence, the bandwidth requirements will increase compared to the first stage. To develop VR video systems that meet all the technical specifications of this stage, a development period of 3 to 5 years is anticipated.

Stage 3 (2028-2032): The third stage marks the full maturity of VR video technologies. VR video systems in this phase would be capable of fully satisfying the human eye's requirements with a resolution of 60 PPD, a frame rate of 150 fps, and a field of view of $150^\circ \times 120^\circ$ [9]. This level of performance would provide users with the best experience possible. Furthermore, this stage will witness the implementation of advanced technologies like light field rendering and the H.266 encoding standard. Despite the current VR video technologies being in the early stage due to scientific and technical challenges related to bandwidth, computation, and reliability, the ultimate goal of VR video is to offer a fully immersive experience that simulates human visual perception at full resolution.

VR video system provides an immersive user experience which is hard to obtain in traditional video devices. Many various VR productions are developed and released in recent years. However, VR video poses several challenges and bottlenecks that are required to be explored and addressed to make the streaming system viable. In this section, some challenges and discussions are provided.

1. To optimize the immersive experience, one needs to consider VR video device characteristics, such as processing power, storage, and energy consumption. However, the answer to achieving maximum immersion depends on various parameters, including network congestion, transmission link, VR video cost, compression algo-

rithms, and more. Therefore, a "Shannon-Like" theory is required to address the challenges in VR video networks effectively.

2. Distortions can occur at various stages of VR video, from acquisition to display. To mitigate this issue, new algorithms or technologies are necessary to introduce less noise during acquisition, projection, encoding, transmission, and display.

3. Due to the variability in network conditions and user movement, a VR video system should be intelligent and robust enough to employ an adaptation mechanism, addressing diverse environmental factors, to ensure a high quality of experience (QoE) for users.

4. The fluctuating nature of network conditions requires dedicated quality evaluation metrics and methods for VR video systems. Traditional video QoE models may not fully consider the nuances of VR video content. Therefore, more quantification methods are needed to assess the QoE of various VR video services. Moreover, mapping QoE requirements to QoS (quality of service) requirements should receive more attention.

5. Compared to traditional video, VR video contains an abundance of information. The future application of 6 DoF (degrees of freedom) holds great potential, as it allows refocusing based on the user's gaze and depth map, enhancing the immersive experience and making it closer to the reality.

6. Privacy is a significant concern for users. With the development of VR video, an intelligent mechanism should be developed to automatically preserve users' privacy while using VR technologies.

7. VR video systems, with their computing, storage, and communication demands, require significant data transmission and power supply. Lowering energy consumption and addressing heat dissipation issues pose technical challenges. Additionally,

seamless charging technologies like wireless power charging should be developed to cater to the needs of VR video devices.

1.3 Existing Problems

Because of the large bandwidth needs, streaming VR footage in good quality can be somewhat difficult. Gigabit-level bandwidth is required for VR videos with resolutions up to 24K and frame rates of 120 frames per second [10]. During playback, fluctuating network conditions can also affect video quality. The user can only see a certain amount of content at a time due to the limited field of view. As a result, the unseen portions of the VR video consume large amounts of bandwidth, resulting in inefficient resource consumption.

To deal with the high bandwidth requirements of VR video streaming, researchers have proposed tile-based adaptive bitrate (ABR) streaming techniques [1], [10]–[13]. Adaptive bitrate streaming strategies combine temporal and spatial approaches to optimize the streaming process. A temporal adaptive bitrate streaming technique, such as Dynamic Adaptive Streaming over HTTP (DASH) [14], dynamically selects the right bitrate for the video depending on fluctuating bandwidth availability. Tile-based adaptive bitrate streaming splits video files into tiles and encodes them at different bitrates. The encoded video is then divided into segments, each lasting about one to ten seconds. Each segment's bitrate is automatically adjusted during playback in accordance with the current state of the network in order to give the viewer the best possible viewing experience. With tile-based adaptive bitrate streaming, network resources are dynamically allocated and the bitrate is adjusted at the segment level to provide the best video quality within the bandwidth available. A high-quality VR video streaming service is crucial for providing customers with immersive and pleasurable experiences while maximizing bandwidth utilization and network efficiency. It is necessary to conduct further research and develop adaptive streaming approaches in order to fully realize the potential of VR

content distribution.

A highly effective VR video streaming strategy involves dividing the content into spatial tiles. As many accurate sensors, such as HD cameras and gyroscopes, are built into the VR client, the position of the user's viewpoint can be estimated. The viewpoint trajectories of different VR users have spatial and temporal similarities for a given VR scene, which can be modeled as a pattern [1]. Based on past data and comparisons of similar spatial and temporal characteristics, the system can forecast viewpoint trajectories. A variety of methods are used in this prediction process, including:

- Linear Regression: Based on historical data, this method models the relationship between spatial and temporal variables to predict viewpoint trajectory [3].
- Clustering Algorithms: Based on historical data, this method predicts viewpoint trajectory by using statistical techniques to model spatial and temporal relationships [15].
- Convolutional Neural Networks (CNNs): Since CNNs are built to interpret spatial information, they are ideal for analyzing VR video for viewpoint trajectory predictions [16].
- Long Short-Term Memory Networks (LSTMs): Sequential data can be processed by the LSTM over time. In VR video data, they are excellent for capturing temporal patterns, enabling accurate viewpoint trajectory predictions [17].
- Transformer Models: Since the transformer has the ability to model the temporal relationships between elements in a sequence, it also performs well on trajectory prediction [18].
- Reinforcement Learning (RL): RL algorithms enable systems to learn from interactions with the environment and optimize decision-making. The application of RL to VR video streaming can improve the experience of users by predicting viewpoint trajectories [19].

By utilizing various prediction approaches, the system can optimize the streaming process by delivering the most relevant content to the user's field of view. In order to ensure a smooth and engaging VR video streaming experience, sophisticated prediction methods must be implemented.

During the preparation of the video for client streaming, the system ranks the tiles according to certain standards, such as how close they are to the user's perspective. In order to adjust to changing network circumstances, the system uses beam search [1], greedy programming [12], and single-agent reinforcement learning (SARL) [20], [21] to determine the appropriate bitrate for each tile. In this method, tiles within the user's viewport are concentrated on in order to ensure the best possible streaming performance. These high-priority tiles are streamed in high quality, boosting the user's initial perception and engagement. Tiles that are outside of the viewport and not in the current field of view, on the other hand, may not stream at all or only stream at a lower quality. This optimizes bandwidth efficiency by reducing the amount of data that has to be transmitted by removing content that is currently not visible to the user. The technology maximizes the effectiveness of streaming by dynamically modifying the bitrate of each tile according to its importance to the user's point of view. By allocating more bandwidth to crucial tiles in the user's range of vision, it guarantees a fluid and captivating experience. Less relevant tiles are simultaneously given less bandwidth, so resources are used efficiently without sacrificing the quality of content that matters most to the user.

VR videos have rapid movements that are difficult to predict using existing algorithms. These methods mostly employ past trajectories or video information to estimate a single-viewpoint trajectory, which may not be able to account for unanticipated changes in the user's perspective. Additionally, the tile-based structure of VR media makes it more challenging to apply reinforcement learning (RL) approaches. When a VR video is split into several tiles, RL techniques cannot be utilized effectively. As there are many tiles and complex interactions between them, the challenge lies in deciding how to proceed for each tile. Thus, it becomes increasingly difficult to determine an optimal policy

using RL approaches in this tile-based framework. A more sophisticated prediction model can precisely capture abrupt viewpoint changes in virtual reality recordings, which will address these constraints. Further, improving the quality of VR video experiences and optimizing system performance requires exploring new solutions to apply RL efficiently to tile-based VR video streaming systems.

When tile bitrates are obtained simultaneously using SARL-based methods, a high-dimensional action space can be created. In VR videos, SARL creates an N^M action space that has M tiles and N levels of bitrate. For example, in VR video, 72 tiles and 6 bitrate levels yield a massive action area of 6^{72}. SARL algorithms cannot be directly applied to a high-dimensional action space due to computational difficulties. As SARL has a large action space, it requires careful dimensionality reduction. A function approximation or discretization can help balance the size of the action space and the granularity of the tile bitrate. It is still being studied how to manage large action spaces in VR streaming using SARL. SARL-based VR video streaming systems need to optimize their bitrate decisions and performance in high-dimensional action spaces in order to improve their performance and scalability. Alternatively, the MARL method (Multi-Agent Reinforcement Learning) assigns an agent to each tile in order to address the tile-based streaming challenge. This strategy, however, adds many agents, which may limit scalability. For example, with 72 tiles and 6 bitrate levels, the MARL-based method would involve 72 agents, each operating in a 6-dimensional action space. Including multiple agents increases the complexity of coordination and communication among them, posing challenges in scalability and computational resources. The large number of agents and action spaces can increase computational overhead and reduce training and decision-making efficiency. In VR video streaming, MARL-based systems must handle many agents while scaling. The method's scalability issues can be solved by improving agent coordination and communication, as well as training and decision-making. These challenges underscore the complexities involved in efficiently applying RL techniques to tile-based VR video streaming systems. Addressing these issues will improve RL-based

VR video streaming methods.

VR video streaming systems must optimize data transmission efficiency and quality, especially on bandwidth-constrained networks. Tile-based adaptive bitrate streaming reduces server-client data volume. Despite these efforts, VR video streaming systems underutilize network node computational capabilities. Underutilization can cause network congestion and rebuffering when streaming VR films over low-bandwidth networks. When incoming data rates exceed outgoing rates, network nodes may drop packets. Random packet drops can disrupt reconstructed video quality. However, judicious selection of which packets to discard, especially those of lower value, can allow for controlled and restricted loss of video quality. Dropping packets outside the viewport reduces visual distortion. Determining the best packets to delete to decrease bandwidth is difficult and requires advanced algorithms. Complex and continuing research in VR video streaming systems aims to minimize bandwidth utilization while retaining video quality. To overcome these issues and improve VR video streaming, especially in bandwidth-constrained scenarios, creative methods must be explored.

1.4 Contributions

We aim to improve QoE at two layers: the application layer by proposing a multi-agent reinforcement learning based adaptive bitrate streaming method and the network layer by proposing a tile-weighted rate-distortion (TWRD) packet scheduling method.

1.4.1 MARL Adaptive Streaming

The tile-based adaptive bitrate streaming technology encounters challenges in ensuring a high QoE for users due to the dynamic nature of network conditions and the unpredictability of user head movements. Furthermore, SARL methods struggle with handling environmental uncertainties compared to MARL. To overcome these limitations, we propose a novel multi-viewpoint prediction strategy based on the transformer model [22]

to effectively handle sudden head movements during VR video playback. Additionally, we introduce a MARL-based ABR streaming system that leverages the multi-viewpoint prediction to adapt to varying network conditions. The objective of our solution is to enhance the QoE for users by precisely predicting their perspectives and efficiently allocating network resources to provide high-quality video streaming at the application layer in VR environments. In order to improve the performance and quality of VR video streaming systems, we propose combining multi-viewpoint prediction and MARL-based adaptive bitrate streaming. This strategy enhances the user experience greatly, especially when dealing with fluctuating network conditions. By using MARL-based adaptive bitrate streaming, we are able to forecast numerous views efficiently, enabling viewers to view VR videos in high quality even under difficult network conditions. By enabling smooth and immersive virtual reality, this technology is significantly improving the accessibility and enjoyment of VR material across a variety of network conditions.

Contributions:

- At the application layer, we provide a unique VR video adaptive bitrate streaming approach that blends adaptive bitrate streaming based on multi-agent reinforcement learning with multi-viewpoint prediction utilizing an attention mechanism. The goal of this approach is to improve the VR video streaming system's overall performance.

- Our approach includes a multi-viewpoint prediction method that utilizes a transformer model. By treating the prediction task as a classification problem and leveraging the attention mechanism and information from various users' viewpoint trajectories, our method achieves accurate predictions and performs well in viewpoint prediction.

- In order to maximize a specified QoE metric, adaptive bitrate streaming is formulated as a Dec-POMDP (Decentralized Partially Observable Markov Decision Process) optimization problem. To obtain the optimal solution for this Dec-POMDP

problem, we propose a MARL method that employs the MAPPO (multi-agent proximal policy optimization) algorithm within the centralized training and decentralized execution framework (CTDE).

- To assess the effectiveness of the proposed method, a comparative analysis is conducted against existing algorithms for VR video streaming. Experimental results demonstrate that the approach outperforms other algorithms in terms of the defined QoE metric across different network conditions.

1.4.2 TWRD Packet Scheduling

In contrast to the predominant focus on tile-based adaptive bitrate streaming for mitigating bandwidth consumption at the application layer, we also propose an innovative optimization system called tile-weighted rate-distortion based on viewport prediction operating at the network layer. Our aim is to address the underutilization of network node resources and optimize the VR video streaming process effectively. The TWRD system introduces a unique approach by leveraging viewport prediction, obtained through a transformer model, to dynamically assign weights to tiles and their associated packets. This weighting mechanism optimizes the allocation of resources based on the predicted importance of each tile within the user's field of view. In order to optimize resource utilization and enhance the streaming experience, our system prioritizes the transmission of significant tiles so they will be transmitted in a timely manner. In order to overcome bandwidth limitations, we formulate a packet scheduling problem that determines which packets are dropped. As part of the optimization problem, the decision-making process incorporates error propagation within the video. Our system schedules packets efficiently and effectively using weighted rate-distortion information and dynamic programming techniques. TWRD offers many advantages. Streaming VR video is optimized by dropping packets at network nodes in order to reduce congestion and perform optimally under bandwidth constraints. We have developed a system that greatly improves streaming quality while preserving precious bandwidth by maximizing resource allocation

and reducing unnecessary transmissions.

Contributions:

- At the network layer, we propose a tile-weighted rate-distortion packet scheduling system. In network environments with limited bandwidth, this approach aims to improve performance. In our system, tiles and packet distortions are weighted based on viewport prediction. An optimal packet transmission strategy is computed using the weighted rate-distortion information. Additionally, we discard packets strategically at network nodes to avoid congestion and to optimize the quality of virtual reality video streaming within network constraints. As a result of the proposed system, the overall streaming experience will be enhanced and VR video material will perform optimally even in situations in which bandwidth is limited.

- A dynamic tile ranking mechanism is presented that utilizes a transformer model for predicting viewports. Based on the prediction probability, tile weights are adjusted both inside and outside the viewport. It ensures that tiles are assigned rational weights in accordance with their pertinence to the line of sight of the user. Through dynamic adjustment of weights, the mechanism optimizes resource allocation by prioritizing the transmission of tiles that are more likely to be visible to the user. In this methodology, critical frames are transmitted first and less significant frames are transmitted later, thus improving the quality of the streaming experience.

- Packet scheduling is regarded as an optimization problem with bandwidth constraints by incorporating weighted rate distortion data. A solution based on dynamic programming is devised to optimize transmission scheduling. The proposed solution facilitates efficient packet scheduling by considering the rate-distortion characteristics and importance of packets. The proposed solution optimizes the packet scheduling process through the utilization of dynamic programming techniques. This guarantees an efficient allocation of scarce bandwidth resources while simultaneously maximizing the quality of streaming. .

- The proposed approach is compared with other existing methods to evaluate its effectiveness in VR video streaming. Experimental results demonstrate that our approach consistently outperforms the alternative approaches across various bandwidth conditions. This comparison highlights the superior performance of our proposed method, indicating its effectiveness in improving the streaming quality and overall user experience.

1.5 Scholarly Achievements

- **Refereed journal papers:**

  1. **H. Wang**, Y. Zhou, and A. E. Saddik, "Vitasi: A real-time contactless vital signs estimation system," Computers and Electrical Engineering, vol. 95, p. 107 392, 2021.
  2. Q. Yu, **H. Wang**, F. Laamarti, and A. El Saddik, "Deep learning-enabled multi-task system for exercise recognition and counting," Multimodal Technologies and Interaction, vol. 5, no. 9, p. 55, 2021.
  3. **H. Wang**, D. P. Tobón V., M. S. Hossain, and A. E. Saddik, "Deep learning (DL)-enabled system for emotional big data," IEEE Access, vol. 9, pp. 116 073–116 082, 2021.
  4. **H. Wang**, H. Dong and A. E. Saddik, "Tile-Weighted Rate-Distortion Optimized Packet Scheduling for 360° VR Video Streaming," IEEE intelligent system.
  5. **H. Wang**, H. Dong and A. E. Saddik, "Multi-Agent Reinforcement Learning Based Rate Adaptation for VR Video Streaming with Multi-Viewpoint Prediction," IEEE Transactions on Mobile Computing.
  6. **H. Wang**, R. Martinez-Velazque, H. Dong and A. E. Saddik, "Experimental Studies of Metaverse Streaming," IEEE Consumer and Electronic Magazine

7. M. Wang, **H. Wang** and A. E. Saddik, "An Framework for Digital Twin Generation and Animation,"

1.6 Thesis Organization

The thesis is organized as follows:

- Chapter 2 introduces the related works of VR video, which mainly contains VR video streaming, viewpoint prediction, bitrate adaptation, packet scheduling and deep learning.
- Chapter 3 presents the proposed multi-agent reinforcement learning bitrate adaptation streaming with the multi-viewpoint prediction at the application layer on client/server.
- Chapter 4 presents the proposed tile-weighted rate-distortion packet scheduling at the network layer on network nodes.
- Chapter 5 summarizes the thesis content and contributions, and briefly discusses possible future research directions.

Chapter 2

Background and Related Work

In this chapter, we show an overview of the related work to VR video in terms of VR video streaming, viewpoint prediction, rate adaptation, packet scheduling, and deep learning. We first discuss VR video streaming in sections 2.1, 2.2, and 2.3, then present existing methods for viewpoint prediction in section 2.4, where most work mainly focuses on single-viewpoint prediction. Then, we review the adaptive bitrate streaming using heuristic and RL-based algorithms in section 2.5. The related work about packet scheduling algorithms proposed for traditional and VR video is reviewed in section 2.6. The deep learning including transformer and reinforcement learning is discussed in section 2.7.

2.1 VR Video Streaming Mechanism

Video streaming has become increasingly popular due to the advancements made in video compression algorithms. Both the academia and industry are actively working on developing multimedia streaming solutions. However, the challenge lies in supporting real-life streaming of VR videos. These videos pose unique demands in terms of real-time processing and delivery, setting them apart from other types of data network traffic. To address this challenge, Fig. 2.1 presents an ecosystem that outlines the fundamental principles of VR video streaming. This ecosystem encompasses each step involved in the

lifecycle of a VR video, from acquisition to consumption by end users.

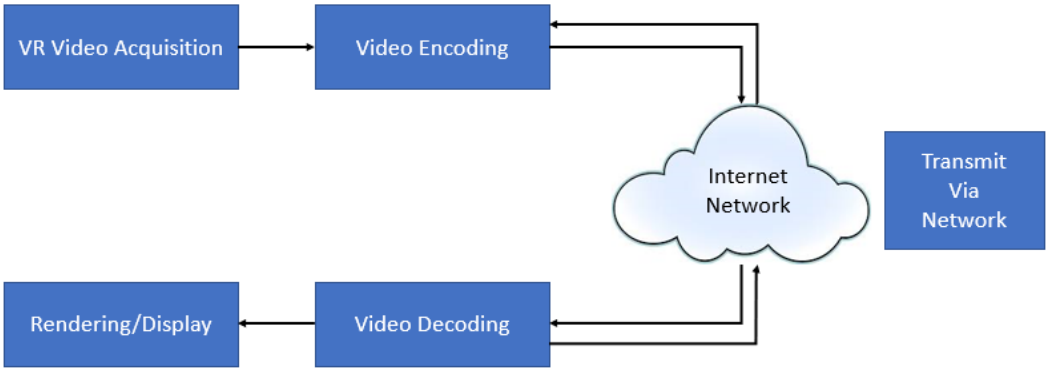

Figure 2.1: VR video streaming framework.

The realm of omnidirectional cameras encompasses a variety of models, such as the Ricoh Theta, Gear 360, and GoPro, among others [23]. These cameras come with various sensors designed to capture a full 360-degree view. Advancements in stereoscopic omnidirectional camera systems, such as Facebook Surround 360, have enabled the capture of stereo views in all directions [24]. Nevertheless, capturing dynamic scenes with stereoscopic omnidirectional cameras presents significant challenges in building a professional capturing system, primarily due to self-occlusion problems between the cameras. To address these challenges, automatic image stitching relies on various planar models, such as affine and cubic transformations, to align the camera's different views. However, this process can lead to view blending and distortion when projecting onto a spherical surface [25].

In the context of video stitching, specific considerations come into play. Video stabilization is crucial for cameras in motion, while video synchronization is essential for individual sensors [26]. The stitching process aims to create a seamless 360-degree representation, often employing a planar representation before mapping and encoding [27]. A planar representation allows the reuse of existing frames, involving encoding, packaging, and transmission steps. However, it is worth noting that the acquisition process itself may introduce significant visual distortions. The absence of synchronization among cameras can lead to motion discontinuities, impacting the overall 360-degree video streaming

framework. Additionally, capturing omnidirectional stereoscopic 3D content poses additional challenges [28], [29]. The planar transformation model, however, allows multiple views to be synchronized if the projection center of both cameras is the same. It is important to address potential keystone effects resulting from the convergence of two cameras at slightly different planes, as well as the cardboard effect, which can cause unnatural object flattening [25].

2.1.1 Projection and Encoding

Once the content is captured and stitched using advanced tools, the next step is to project the VR sphere into a 2D planar format for efficient coding and transmission over networks with limited bandwidth. In the realm of VR video compression, various projection approaches can be employed to determine the optimal techniques for compression and coding. One commonly employed solution is the equirectangular projection (ERP), which finds widespread use in 360-degree video streaming services like YouTube and Jaunt VR. ERP involves the process of flattening a sphere onto a 2D surface surrounding the viewer [30].

Efficient coding plays a crucial role in evaluating video compression methods, particularly for streaming panorama, ultra HD (UHD), and 360-degree video content. The latest widely implemented standard is the High-Efficiency Video Coding (HEVC), which was developed by the JCT-VC (Joint Collaborative Team on Video Coding) and standardized by VCEG (ITU-T Video Coding Experts Group) and MPEG (ISO/IEC Moving Picture Expert Group). HEVC aims to explore future video coding technologies, and the most promising advancements are incorporated into the Joint Exploration Test Model. In 2016, MPEG defined the functional requirements for FVC [31]. Additionally, various companies have independently devised their own video coding formats, such as VP8 [32], VP9 [33], and VP10 [34]. Introducing the novel Versatile Video Coding (VVC) standard, defined in MPEG-I Part3, which offers support for three distinct types of videos: High Dynamic Range (HDR), Standard Dynamic Range (SDR), and VR-oriented video. While

maintaining the same subjective video quality as HEVC, VVC aims to achieve a compression performance 30-50 percent better than HEVC [35]. As video streaming and VR technologies continue to advance, the demand for more efficient video compression techniques remains constant to meet the requirements of panorama, UHD, and 360-degree video content.

2.1.2 Transmission

Distributing VR video content presents several technical obstacles due to the stringent requirements of low latency and high video rates associated with omnidirectional signals [36]. Despite these challenges, the traditional transmission protocols for standard video frames continue to be utilized for the delivery of 360-degree video. During the packaging stage, cutting-edge streaming frameworks such as Dynamic Adaptive Streaming over HTTP (DASH) are employed to package the data, offering effective solutions for over-the-top (OTT) services. In the context of VR videos, the prevalence of MPEG-DASH-based content delivery solutions has risen due to their ability to leverage existing delivery architectures with minimal extensions [37]–[40].

Prominent streaming technologies, including tile-based streaming [41] and viewport-based streaming [42], rely on the DASH to dynamically determine the viewing area's quality based on the available network conditions. However, adaptive streaming of VR content introduces vulnerabilities to transmission losses, potentially impacting the user experience. Compared to viewport-agnostic streaming, viewport-based streaming is particularly susceptible to distortions commonly associated with DASH, such as quality degradation during rapid head movements, variations in spatial quality, buffering, and delays. These factors can significantly affect the viewing experience, especially when the compressed content is consumed through an HMD, thereby influencing the QoE in immersive applications. Unfortunately, these implications are often overlooked or underestimated.

2.1.3 Rendering and Display

Enabling user interaction in VR content necessitates a series of inverse steps to be undertaken. These steps encompass decoding, unpacking, transformation, and rendering. To visualize the rendered content, an inverse mapping technique from a plane to a sphere is employed, which is typically implemented across various devices such as HMDs, monitors, tablets, or smartphones. Nevertheless, consuming 360-degree content on advanced HMDs with cutting-edge display capabilities can introduce new distortions that necessitate careful attention, such as aliasing and blurring. Addressing these distortions is of utmost importance to ensure a premium viewing experience of high quality. Moreover, distortions associated with stereoscopic displays persist in HMDs and can lead to several challenges, including misinterpretation of displayed content and difficulties in perceiving object velocities accurately. These challenges should be duly acknowledged and resolved to enhance the overall visual quality and user experience in VR environments.

2.2 Existing Transmission Methods

VR video streaming poses several challenges, including high bandwidth requirements, low latency, and high frame rates, to deliver a seamless experience as users change their viewing directions within the 360-degree video. Due to the limited computational capabilities of client devices, VR video systems often rely on powerful remote servers for rendering high-resolution content and providing an immersive experience. The server transmits the rendered video to the client over a cable or wireless network, while the client is responsible for display and basic computation. Traditional VR video transmission methods typically stream the entire 360-degree video at a consistent quality, which demands high bandwidth and optimal network conditions. This approach, however, can be inefficient as much of the transmitted data lies outside the user's viewport, never being viewed or used. In addition, fluctuating network conditions can cause congestion and rebuffering when streaming 360-degree videos. To address these issues, adaptive stream-

ing techniques have been developed and implemented in VR video systems. Content delivery and video quality can be adjusted according to network congestion and available capacity using these techniques. Video content is delivered using adaptive streaming by dynamically adjusting streaming quality based on network conditions. As a result, seamless playback is ensured, and network resources are effectively utilized.

2.2.1 DASH

MPEG-DASH is also known as Dynamic Adaptive Streaming over HTTP (DASH). This transmission protocol is widely used because of its straightforward implementation with TCP/IP and its adaptive bitrate streaming capability over HTTP protocol [14]. The DASH protocol is used to automatically transmit different quality content over the Internet based on the bandwidth capacity of the user. A video is divided into segments with short intervals on the server, and each segment is encoded at various quality levels with any encoding standard. Encoded segments are streamed into the client over the Internet when a user starts viewing the content. A user device plays video when it receives streaming data and decodes it. The video player will automatically switch to a lower or higher quality segment to adapt to network conditions. For instance, if the user's current bandwidth is highly limited, the video will be played at a lower quality level using less bandwidth. DASH is an MPEG standard that offers a multimedia specification for transmitting content over HTTP utilizing the adaptive bitrate method [43]. DASH is designed to be highly compatible with the current internet infrastructure, as it imposes minimal processing burden and remains transparent to middleboxes. Moreover, its ability to employ alternative adaptation approaches enables it to adapt to different network conditions. In recent times, DASH has gained widespread popularity for streaming two-dimensional videos over the World Wide Web. The functioning of DASH streaming involves dividing videos into small segments. Each segment has multiple versions with different bitrates [44]. Based on the viewer's perspective, the streaming client requests the appropriate HTTP resource. The main viewpoint areas are streamed with higher

quality while the other areas are streamed with lower quality. Consequently, a video player can seamlessly transition between different quality levels during video playback without any interruptions.

2.2.2 Viewport-based Transmission

In the DASH mechanism, videos can be divided into small segments both temporally and spatially. This allows for the adoption of viewport-adaptive streaming technologies, including tile-based streaming and viewport-based streaming, to control video quality based on varying network conditions. Specifically, in the viewport-based transmission mechanism, the server prepares different quality representations for the DASH segments of a 360-degree video based on predefined viewing directions. With head and eye tracking sensors built into HMDs, users can experience enhanced immersion and interactiveness, even during quick head movements. When a user rapidly moves their head, the corresponding perspective or viewport is streamed at a high quality to match the user's new viewing direction, while the other parts of the video are streamed at lower quality. This dynamic adaptation ensures that the user's current viewport receives the highest quality video, optimizing the streaming experience while reducing bandwidth requirements. By leveraging viewport-based transmission and considering user head movements, the DASH mechanism can effectively manage video quality and bandwidth utilization in real-time, providing an immersive and interactive VR video experience.

2.2.3 Tile-based Transmission

In tile-based transmission, videos are divided into smaller spatial tiles that are independently encoded at different quality levels [45]. During streaming, higher-quality tiles covering the user's viewport are transmitted, while lower-quality tiles outside the viewport may be streamed at a reduced quality or not streamed at all [46]. The selection of tiles is based on viewport prediction and stored on the client side for reconstruction when needed. Tile-based streaming follows the traditional approach of HTTP-based adaptive

streaming for temporal segmentation of videos [45]. However, it introduces spatial division into smaller tiles, allowing for more precise control over video quality based on the user's viewport [46]. Prior studies have focused on tile-based coding and explored various tiling features in 360-degree video streaming. For instance, one approach adjusts the resolution of video tiles based on the user's viewport, utilizing high-quality tiles within the viewport and lower-quality tiles outside of it [45]. Other research has investigated bitrate allocation optimization considering tile content, available bandwidth, and viewing regions [46]. Efforts have been made to address potential issues and ensure a seamless viewing experience. Techniques such as overlapping margins and alpha blending at tile boundaries have been utilized to minimize disruptions and provide smooth transitions between tiles [47]. Recent advancements include the use of multiple hierarchical representations for tiles based on the user's viewport, enabling smoother quality degradation and more efficient quality control during streaming [11], [48], [49]. Tile-based streaming offers flexibility in adapting video quality to the user's viewport, optimizing bandwidth usage, and enhancing the efficiency of VR video streaming systems.

2.2.4 Machine-learning Based Methods

At present, VR systems only provide limited immersive experience because of the limitation of computing power and network capacity, which usually has a negative influence on the user's QoE. To break the challenges of VR systems, many novel methods for the optimization of VR are proposed in academic and industrial fields. Research on QoE provides valuable insights into VR video streaming design and optimization. QoE models could help the system determine how to partition, encode, and transmit VR video. And there are many VR researchers who have successfully optimized VR video streaming by using different QoE models. For instance, as mentioned in the previous section, adaptive video streaming technologies are adopted to reduce bandwidth needs and optimize the network transmission, but the VR system should deliver the user's viewport scene according to the user's head movement while a user switches to a new

perspective. Therefore, head movement prediction is becoming an indispensable task for VR optimization. Meanwhile, the VR system could adjust the VR video transmission to complex and diverse network conditions according to the bandwidth prediction. With the rapid development of machine learning, many optimization algorithms based on machine learning are proposed.

Table 2.1 shows some research on 360-degree video streaming with machine learning in order to improve the QoE of VR video. Qian et al. [3] uses linear regression to predict head movement, and the experiments show that LR with tile-based streaming reduces bandwidth consumption by up to 80%. Zhang et al. [1] test four machine learning methods: Static, Ridge Regression, Linear Regression, and Support Vector Regression to predict viewpoint, and the experiments show that the LR performs better if the prediction window size is greater than 1 second, otherwise RR is the best. Qian et al. [50] adopt the LSTM model and ensemble learning to predict bandwidth and viewpoint. Filho et al. [51] proposed a Reinforcement Learning model that is used to adapt to variable video transmission and they also propose a two-stage model for QoE evaluation. Yu et al. [52] use the Markov Decision Process (MDP) to adapt variable bitrate video streaming over HTTP. In addition, CNN model [53] considering eye and head movement data is developed to improve video quality. Some algorithms such as post-decision state[54] and Q-learning [55] are proposed to improve the QoE with constant bitrate. Indeed, many methods can be employed simultaneously to improve the performance of VR video streaming.

2.3 Existing VR Streaming Systems

2.3.1 FlashBack

To solve the limitation of GPU power in HMD VR devices, Boos et al. [56] proposed a novel solution, named FlashBack, to provide high-quality VR experience on mobile-rendered HMDs. FlashBack prerenders and cashes all possible frames that VR users may

Reference	Method	Scheme
[3]	LR	Predicted viewport
[1]	LR-RR	Predicted viewport/bandwidth
[50]	LSTM	Predicted viewport/bandwidth
[51]	RL	Improved adaptive VR streaming
[52]	MDP	Improved variable bitrate
[53]	CNN	Improved video quality
[54]	Post-decision state	Improved constant bitrate
[55]	Q-learning	Improved constant bitrate

Table 2.1: Machine learning-based methods to improve QoE

encounter. Rendering is completed offline and recorded in the system to build a cache full of panoramic frames. FlashBack builds and maintains a hierarchical storage cache index at runtime in order to quickly look up frames that a user will view. If a cache is lost, the system would make a fast approximation of the correct frame from available cache entries that closely matches the user's view by using mesh warping which is a computer graphics technique independent of scene complexity. In addition, FlashBack works for both static scenes and dynamic scenes. In comparison to a mobile VR configuration rendered on mobile phones, FlashBack improved framerates by up to 8 times, reduced energy consumption per frame by 97 times, and improved latency by 15 times.

2.3.2 Furion

Lai et al. [57] introduced a frame, Furion, that streams high-quality VR video and achieves an immersive VR experience on a mobile device and wireless network. Because a frame in VR can be split into two parts: foreground interactions and background environment, Furion uses a cooperative rendering framework that renders lighter-load, less predictable foreground interactions on the client and a predictable background environ-

ment that includes rich details and complex textures on the server. Meanwhile, by using video compression and parallel decoding on multiple cores on the smartphone, the Furion can support high-quality VR applications on today's smartphone via a WiFi network, with a delay of less than 14 ms and 60 FPS.

2.3.3 Flare

Qian et al. [50] developed a VR video streaming system, Flare, on mobile devices. Flare uses a tile-based streaming mechanism, which only fetches several tiles that cover the user's viewport. In addition, to address the problem of viewport prediction, they collect a dataset including head movement traces from 130 users. An online method based on their collected data is proposed to predict the future viewport and determine the corresponding tile's quality. The experiment shows that Flare achieves huge (18x) quality level improvement in a WiFi environment and reduces bandwidth up to 35% compared to other non-adaptive methods. Furthermore, Flare is a universal framework for VR video streaming that does not require specific encoding technologies.

2.3.4 EPASS360

Zhang et al. [1] presented a high QoE 360-degree video streaming system based on viewpoint prediction and rate allocation for different tiles. The system adopts ensemble learning of LSTM and tests the proposed method on three different databases, which shows that their method provides high accuracy on viewpoint prediction compared with other methods. Meanwhile, an LSTM model is used to predict bandwidth usage. According to the prediction results, the system generates and sets a QoE target, formulates a QoE optimization function, and calculates the optimal rate allocation for tiles. The rate allocation model based on the prediction would solve the QoE perception optimization problem by dividing a video into tiles and allocating high resolution to tiles that a user may view in the future. EPASS360 achieves at least 5% QoE improvement under different conditions compared to state-of-the-art streaming strategies.

2.4 Viewpoint Prediction

Accurate viewpoint prediction is pivotal in minimizing bandwidth costs and delivering superior streaming experiences. One prevalent approach involves employing linear regression techniques to predict a user's future viewpoints based on their historical data [3], [11]. However, these LR-based methods assume linear head movements, leading to potential bias issues. To overcome this limitation, researchers have introduced probabilistic models to estimate the distribution of prediction errors, effectively improving LR methods' performance [12], [58]. To achieve even better performance, novel techniques have emerged, focusing on extracting temporal and spatial features from diverse users' viewpoint trajectories. These methods capitalize on the spatial and temporal similarities present in various users' viewpoint data to predict the current user's trajectory based on their historical records [15], [59].

In addition to the aforementioned techniques, deep learning-based methods have gained substantial attention for viewpoint prediction. Romero et al. [17] design an LSTM model that leverages previous viewpoint positions and saliency maps to predict forthcoming viewpoints. Zhang et al. [1] adopt LSTM models and take the average of prediction results to achieve the final prediction. Fu et al. [20] leverage LSTM and self-attention mechanisms, while Chao et al. [18] utilize transformer encoders for accurate viewpoint prediction. These deep learning approaches aim to enhance prediction accuracy by incorporating supplementary information, such as video content and saliency maps, into their models. Furthermore, reinforcement learning techniques have emerged as valuable tools for viewpoint prediction [19].

The diverse array of methods for viewpoint prediction exemplifies the persistent endeavors to enhance the precision and efficacy of VR video streaming systems. Through the utilization of historical data, video content, and sophisticated modeling techniques, these approaches contribute to an improved overall streaming experience while optimizing bandwidth utilization. Despite the advancements, existing methods in VR video

streaming systems predominantly concentrate on predicting single-viewpoint trajectories. Nonetheless, there is a growing awareness of the necessity for multiple-viewpoint prediction to encompass various potential outcomes for similar past trajectories. In this context, Guimard et al. [60] propose a method for multiple-viewpoint prediction by analyzing public viewpoint data. They emphasize the significance of considering multiple viewpoints due to the divergent potential futures for comparable past trajectories. Their approach employs a discrete variational learning method to generate multiple trajectories. Nevertheless, the absence of probabilities for the prediction results necessitates the integration of an additional likelihood estimator to derive these probabilities.

2.5 Rate Adaptation

The challenge of rate adaptation in VR video streaming revolves around optimizing QoE, a known NP-hard problem [20]. To tackle this, heuristic-based methods such as dynamic programming [61], beam search [1] and the greedy algorithm [12] have been proposed. These heuristics aim to strike a balance between different factors to enhance user experience. However, they often suffer from high computational complexity and struggle to achieve optimal results under diverse network conditions.

To address these challenges, RL-based techniques have been explored. One approach involves ABR strategies that determine the bitrate for individual tiles, considering the expanded action space with multiple bitrate levels and tiles. The Asynchronous Advantage Actor-Critic (A3C) algorithm [62] is commonly used in this context [20], [21], [63]. RL offers promise in adapting to dynamic network conditions and learning from experience. To simplify the action space, bitrate adaptation based on the viewport region has been proposed. Dividing VR videos into viewport and non-viewport regions, Zhang et al. [64] set the bitrate for viewport tiles using the A3C algorithm, while non-viewport tiles receive the lowest bitrate. Similarly, Kan et al. [65] divide VR videos into viewport, marginal, and invisible regions, determining the bitrates for all three regions using an

A3C-based RL model. Another approach by Wei et al. [66] involves a two-step strategy, determining segment bitrate through RL and individual tile bitrates using game theory, accounting for view prediction and segment bitrate.

2.6 Packet Scheduling

Numerous strategies have been proposed to tackle the challenge of rate adaptation in traditional video streaming by employing network-adaptive packet scheduling algorithms. Among these, the tail drop algorithm, a widely used and straightforward approach, discards packets from the queue's tail during traffic congestion [67]. Meanwhile, priority scheduling algorithms prioritize packet dropping based on frame types [68]. However, the drawback of these techniques is that they ignore variations amongst frames of the same kind, which could affect the reconstruction's quality. More sophisticated methods have been put out in response to these restrictions to produce ideal packet scheduling schemes and accurately simulate how packets affect video quality. To handle packet scheduling issues for many movies over constrained network links, for example, optimization frameworks have been devised [69]. Moreover, content-aware packet scheduling techniques prioritize packets based on frame types and temporal complexity, accounting for interdependencies between them [70]. Additionally, models for visibility have been created to produce visual scores for frames, allowing for the selective removal of frames to lower the bitrate [71].

Although packet scheduling for standard video streaming has advanced, there aren't many specific techniques for VR video transmission these days. However, a few related strategies have been investigated. For instance, machine learning-based algorithms have been applied to allocate network resources for live VR video and other media applications [72]. Furthermore, based on network conditions across several channels, multipath TCP has been used to dynamically decide the bitrate for the user's viewport [73]. Moreover, given the popularity of VR content and the use of statistical models to assign weights

to individual tiles, joint transmission systems including many base stations have been proposed [74]. The current approaches are still limited in spite of these efforts. Some ignore the particular data reduction requirements for VR video, focusing instead on the distribution of resources among various traffic kinds. Others might fail to take into consideration the practical manifestation of viewports or ignore variations among individual users. Consequently, more studies and advancements in packet scheduling techniques that are especially adapted to the particular needs of VR video transmission are crucial.

It becomes obvious that standard video streaming methods have received more thorough study attention compared to VR video streaming methods. While present VR video packet scheduling techniques mostly focus on resource distribution over many channels and numerous traffic streams, traditional video approaches largely focus on minimizing the amount of data in the network. Interestingly, the server/client determines the packet scheduling scheme, and network node computational resources are not given much thought. Moreover, the existing VR video packet scheduling techniques do not take viewport relevance into account within the transmission network. Most viewport-aware approaches are proposed for bitrate selection in the server/client, and thus, the crucial information of the viewport is not shared across the network. To address these limitations, we propose a novel Tile-Weighted Rate-Distortion packet scheduling strategy that aims to simultaneously reduce data volume and optimize VR video streaming performance. This innovative approach takes into account the tile-weighted rate-distortion information, allowing for informed decisions regarding packet scheduling while considering the significance of each viewport in the network. By doing so, our proposed TWRD method enhances the overall streaming experience and achieves a more efficient data utilization in VR video transmission.

2.7 Deep Learning

Deep learning has become a dominant and highly popular field within the realm of machine learning. Its remarkable success in various domains, including computer vision, natural language processing, and reinforcement learning, has elevated its prominence in the scientific community.

Within the domain of deep learning, a wide range of techniques is employed, including supervised, unsupervised, and reinforcement learning methods, each serving different purposes in data processing.

Supervised Machine Learning: In the supervised setting, neural networks are trained to make predictions or classify data by learning from labeled datasets. Both input features and corresponding target variables are fed into the network during training. Through the process of backpropagation, the neural network adjusts its parameters to minimize the error between predicted and actual targets. Applications of supervised learning include image classification, sentiment analysis, language translation, and more, where convolutional neural networks and recurrent neural networks are commonly employed.

Unsupervised Machine Learning: Unsupervised learning involves training neural networks to discover patterns or group data without the presence of labeled information. In this scenario, target variables are not available, and the machine autonomously identifies underlying patterns or relationships within the data. Techniques like encoding network and generative AI can be applied for unsupervised problems, encompassing clustering, association rules and anomaly detection.

Reinforcement Learning: Reinforcement learning focuses on training agents to make decisions within an environment, aiming to obtain maximum rewards. The agent interacts with the environment, taking action and receiving the corresponding reward. Deep reinforcement learning techniques, such as Deep Deterministic Policy Gradient (DDPG) and Deep Q networks (DQN), are applied to learn policies that represent sets of actions

that optimize cumulative rewards for various tasks, such as robotics and video games. For instance, deep reinforcement learning models can train robots to grasp, navigate, and manipulate objects. Agents can learn optimal strategies through continuous interaction with the environment using these approaches.

There is a wide range of applications for deep learning, including computer vision and natural language processing (NLP). Deep learning networks are used by computers to understand and interpret images and videos. The use of deep learning in computer vision can be found in a variety of domains, including:

- Object detection and recognition: A deep learning model can detect and position objects precisely within images and videos, allowing autonomous vehicles and surveillance systems to operate.
- Image classification: A deep learning model can accurately categorize images into discrete categories, which makes it useful for healthcare, defect detection, and security applications.
- Image segmentation: With deep learning models, specific attributes and objects within images are detected and classified based on the ability to partition images into distinct regions.

Deep learning's wide-ranging applicability and high efficacy have solidified its status as a pivotal domain within the field of machine learning. The applications of this technology are constantly growing, leading to progress in other areas and motivating additional research and development.

2.7.1 Reinforcement Learning

An important subdomain of machine learning is reinforcement learning. Basically, reinforcement learning involves letting the agent learn what to do based on the environment state in order to maximize long-term rewards. MDP can be used to model the learning

process. Fig. 2.2 shows the interaction between the agent and the environment. By extracting multilevel features from high-dimensional abstract inputs (such as images), deep learning has demonstrated its powerful representation capabilities in image classification, NLP, and other fields. A deep neural network has also been proven to be a general function approximator, allowing it to approximate value functions and policies in complex tasks involving very high levels of input complexity. Consequently, DRL has attracted much attention in the past few years.

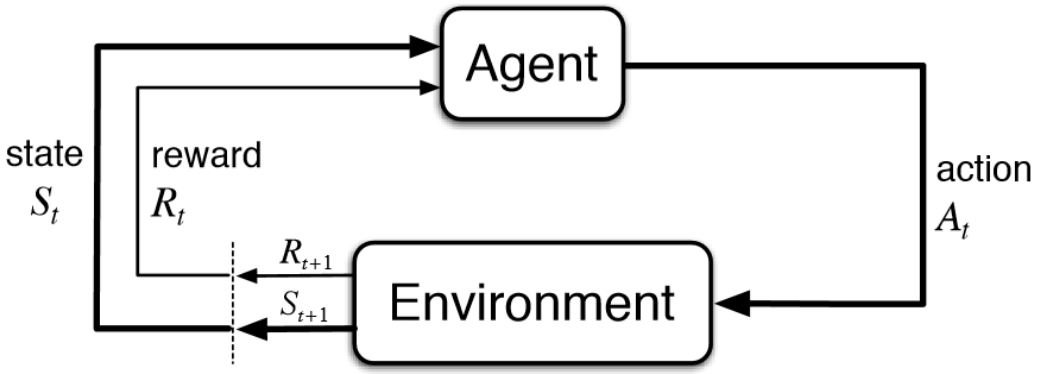

Figure 2.2: The interaction between agent and environment in reinforcement learning: At every time step, the agent gets an observation from the environment and selects an action based on its policy. Once the system conducts the action to the environment, the agent obtains a reward and a new observation from the new environment.

2.7.1.1 MDP

The MDP (Markov Decision Process) framework is a classic approach for addressing sequential decision-making problems. It operates under certain assumptions: the environment is Markovian, indicating that the state at the next time step is solely influenced by the current state, irrespective of past states; the environment is fully observable, allowing the agent to perceive all environmental information at any given moment. However, due to the limitations of these assumptions in certain contexts, various MDP variants have been proposed, such as partially observable MDP. Fundamentally, an MDP encom-

passes a set of states, actions, transition probabilities, rewards, and a discount factor. Here's a brief explanation of each component:

- States: A state represents a possible situation or configuration that the agent can encounter during decision-making. Based on the current state of the environment, the agent chooses actions.
- Actions: During each state, the agent conducts an action from among a variety of actions. The agent selects an action based on its current state and its decision-making policy.
- Transition Probabilities: The transition probability describes the likelihood that an action will cause a state to transition from one to another. They are used to capture the dynamics of a system and to describe uncertainty in its environment.
- Rewards: A reward represents the utility or desire associated with being in a particular state and taking a specific action. The agent aims to maximize the cumulative rewards it receives in the long run.
- Discount Factor: Discount factors are values between 0 and 1, which indicate the preference of agents for immediate versus future rewards. Long-term planning and short-term gains can be balanced with this method.

The goal of an MDP is to find the optimal policy, which entails mapping states to actions to maximize expected cumulative rewards. The optimal policy is usually learned from interactions with the environment, either through dynamic programming algorithms or reinforcement learning methods.

2.7.1.2 Multi-agent Reinforcement Learning

A multi-agent system can be characterized as a Markov game [75], where multiple agents make simultaneous decisions and take actions based on the current state. The actions of each agent influence the state transitions within the environment. In certain complex

tasks, agents might have limited observations, perceiving only a portion of the overall environment. Prior to the emergence of deep learning techniques, researchers utilized RL to address fully cooperative, fully competitive, and mixed tasks.

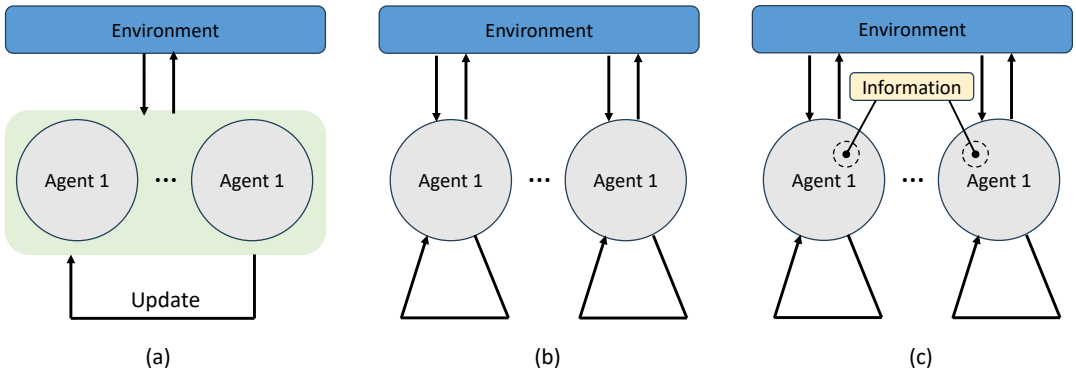

Figure 2.3: Various Training Approaches in the Multi-Agent Setting. (a) CTCE: a joint policy is shared and maintained among all agents. (b) DTDE: each agent updates its individual policy independently. (c) CTDE: agents exchange additional information during training, but this information is not utilized during testing.

In stage t, each agent selects and executes an action based on its own policy. Each agent receives a reward as immediate feedback to the state transition as the system evolves from one state to the next under joint action. Similar to the single-agent problem, each agent aims to optimize the rewards received over time by adjusting its policy. Multiple-agent training has long been a computational challenge because state and action spaces become exponentially more complex as agents multiply. In agent training, two paradigms commonly exist: centralized and distributed. In centralized training, policies are updated by exchanging information among agents during the training process. However, during testing, this additional information is typically discarded. As an alternative, training can also be performed in a distributed manner, where each agent independently updates its own policy and develops individualized strategies. Apart from the training paradigm, agents can also differ in their action selection approach. Two execution schemes are identified. Centralized execution involves agents being guided by a centralized unit that calculates joint actions for all agents. In contrast, decentralized execution

entails agents selecting actions based on their individual policies, without centralized coordination.

As shown in Fig. 2.3, in distributed training decentralized execution (DTDE), each agent is associated with a policy that maps local observations to a distribution over individual actions. Each agent learns independently since no information is shared between them. In the CTCE (Centralized Training with Centralized Execution) scheme, a centralized executor is employed to represent the joint policy. This centralized executor takes the distributed observations as input and maps them to a set of distributions over individual actions. In the CTDE (Centralized Training with Decentralized Execution) scheme, each agent maintains its own individual policy, where local observations are mapped to a distribution over individual actions. During training, agents have access to additional information, but this information is not utilized during testing or deployment.

2.7.2 Transformer

The transformer model [22], a prominent deep learning architecture, has gained widespread adoption in various domains. Initially proposed as a sequence-to-sequence machine translation model, the transformer has evolved to become the go-to architecture for pretrained models (PTMs) and has achieved state-of-the-art performance on diverse tasks. Its versatility has led to its extensive usage not only in language-related applications but also in fields such as computer vision, audio processing, chemistry, and life sciences.

The transformer model consists of an encoder and a decoder, both constructed from multiple identical blocks. In the encoder, every block comprises a multi-head self-attention module along with a position-wise feed-forward network (FFN). Residual connections are applied around each module, allowing for the model's depth to be increased, and Layer Normalization is used for regularization [76], [77]. In contrast, In the decoder, the blocks feature cross-attention modules connecting the self-attention modules and the FFNs. Furthermore, modifications are made to the self-attention modules in the decoder to prevent attending to subsequent positions, ensuring proper sequential decoding. The

architecture of the vanilla transformer is shown in Figure 2.4.

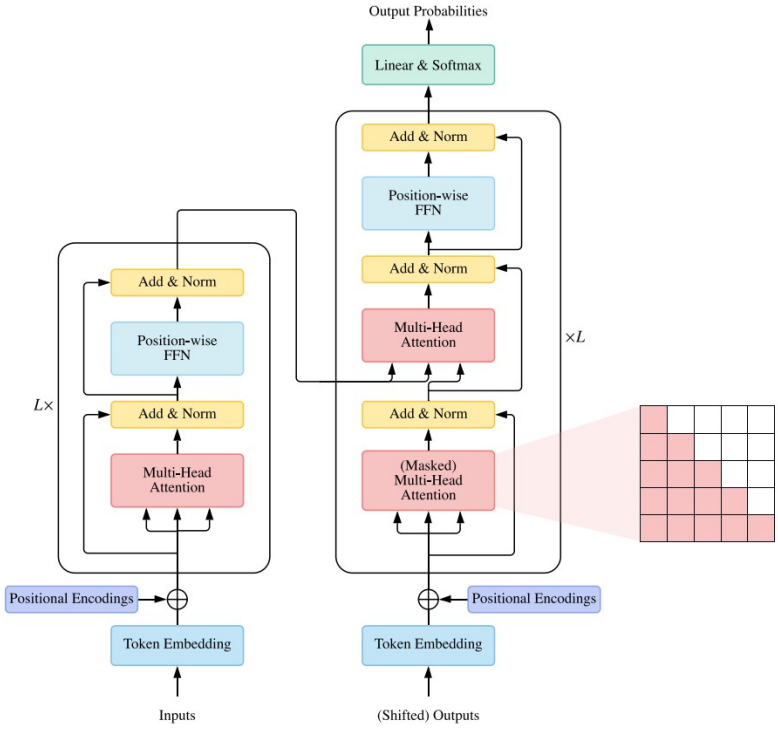

Figure 2.4: Architecture of the transformer.

Transformers utilize a self-attention mechanism to determine the importance of each token in a sequence when processing each token. The attention mechanism allows the model to effectively capture contextual relationships and dependencies. Let's denote an input sequence of tokens as $X = x_1, x_2, ..., x_n$, where n is the length of the sequence. Transformers are composed of encoders and decoders, which are both composed of multiple layers. Each encoder layer has two sublayers: a multi-head self-attention layer and a position-wise FNN. The multi-head self-attention is defined as follows:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \text{head}_2, ..., \text{head}_h) \cdot \mathbf{W}^O \tag{2.1}$$

where each head_i is computed as:

$$\text{head}_i = \text{Attention}(Q \cdot \mathbf{W}_i^Q, K \cdot \mathbf{W}_i^K, V \cdot \mathbf{W}_i^V) \tag{2.2}$$

Here, Q, K, and V represent the query, key, and value matrices respectively. The matrices $\mathbf{W}_i^Q, \mathbf{W}_i^K,$ and $\mathbf{W}_i^V$ are learnable linear transformations specific to each attention head i.

The output of the attention mechanism is concatenated and linearly transformed by $\mathbf{W}^O$. The attention mechanism, denoted as Attention(Q, K, V), computes attention weights based on the similarity between the query (Q) and key (K) matrices. The attention weights are used to weigh the values (V) matrix, producing the output. The attention mechanism can be formulated as:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.3}$$

where d_k represents the dimension of the key matrix. The position-wise feed-forward neural network applies a two-layer fully connected network to each position in the sequence independently:

$$\text{FFN}(x) = \max(0, x\mathbf{W}_1 + \mathbf{b}_1)\mathbf{W}_2 + \mathbf{b}_2 \tag{2.4}$$

An additional sub-layer is included in the decoder, which performs multi-head attention over the encoder's output. The attention block in the decoder is similar to the one in the encoder, but it additionally applies a mask to prevent attending to future positions during training. The output of the transformer model is obtained from the final decoder layer and passed through a linear layer followed by a softmax activation function to produce the probability distribution over the target token.

Chapter 3

MARL Adaptive Streaming

In this chapter, we present and evaluate our proposed multi-agent reinforcement learning-based adaptive bitrate streaming at the application layer operating on client/server for VR video with multi-viewpoint prediction. We first give an overview of the proposed system in section 3.1. The multi-viewpoint prediction based on the transformer is introduced in section 3.2. After that, the MARL adaptive bitrate streaming system is presented in section 3.3. The experiment results for multi-viewpoint prediction and MARL rate adaptation are shown in section 3.4.

3.1 Introduction

The conventional solutions encounter challenges when dealing with varying network conditions and users' sporadic head movements in tile-based ABR streaming setups. Furthermore, SARL struggles with uncertainties in the environment compared to MARL. To address these issues, we present a fresh approach: an innovative multi-viewpoint prediction method, leveraging the transformer architecture [22], to effectively handle abrupt head movements. Complementing this, we introduce a MARL-based ABR streaming system that adapts to fluctuations in the network conditions. Our emphasis lies in streamlining the MARL framework by reducing the number of agents. This is accom-

plished by implementing specific rules to determine the optimal bitrates for individual tiles within the system.

- Rule 1: In our strategy, we first divide the VR video into separate viewport regions, determined by the positions of the viewpoints. This partitioning prevents excessive overlap among viewports, especially when viewpoints are in close proximity. Should any overlapping areas arise, we employ a probability-based approach to assign them to the relevant viewport region, considering the likelihood of being viewed by the user. By giving priority to regions with higher viewing probabilities, we can optimize the distribution of visual content, ultimately enriching the overall user experience.

- Rule 2: We implement a bitrate assignment strategy within our system to ensure seamless viewing of VR content, regardless of irregular head movements. Based on the user's viewpoint, this strategy assigns the lowest bitrate to tiles outside the current viewport region. This method effectively optimizes network resource utilization by prioritizing transmission of tiles within the immediate field of view of the user. In this way, the user's irregular movements do not affect the VR experience, ensuring a smooth and immersive experience.

- Rule 3: Our proposed system minimizes the number of agents in the MARL framework by using a simplified approach. All tiles within a viewport region are set to the same bitrate. Therefore, the MARL system is significantly simplified, and agents are determined by viewport regions rather than individual tiles. This simplified approach improves MARL's manageability and computational efficiency. It eliminates the need to handle individual tiles separately by optimizing bitrate allocation for each viewport region. With fewer agents, the decision-making process within ABR streaming becomes simpler and resource allocation more efficient.

Figure 3.1 illustrates the architecture of the proposed ABR streaming system for VR video. This system comprises various essential elements and processes aimed at

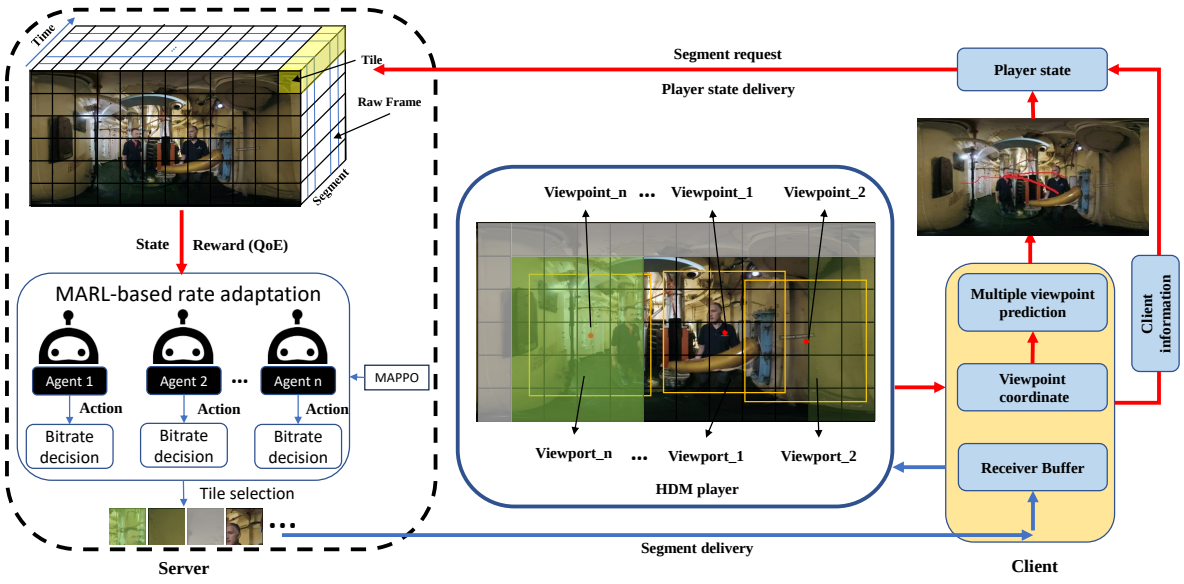

Figure 3.1: The proposed system architecture for the MARL-based VR video streaming system utilizing predicted multiple viewpoint trajectories is as follows: Each viewport region is assigned a bitrate by an agent. The quality of a tile varies based on the region's viewing probability and the prevailing network conditions when it is streamed to a client.

optimizing the streaming experience:

- Video Segmentation: The VR video is temporally segmented and spatially divided into tiles, all of which are stored on a server. This division enables the efficient delivery and playback of VR video content.
- Viewpoint Initialization: At the onset of VR video viewing, the user's viewpoint is initially set to the video's center, serving as the starting point for subsequent operations like viewpoint prediction and adaptation.
- Viewpoint Prediction: Utilizing historical viewing traces, the viewpoint prediction module forecasts multiple future viewpoint trajectories. These trajectories are accompanied by viewing probabilities, indicating the likelihood of the user shifting their focus to specific regions within the VR video.
- Viewport Division: Based on the prediction results, the VR video is segmented

into three distinct viewport regions, each corresponding to a predicted viewpoint trajectory. Additionally, a separate region encompasses all tiles not falling within these three regions.

- and available data to make informed decisions regarding bitrate allocation for each viewport region. The primary objective is to optimize streaming quality and enhance the overall user experience.

- MARL-based Rate Adaptation: The MARL-based rate adaptation module incorporates prediction results and other player information, leveraging environmental states

- Bitrate Streaming: Different bitrates are assigned to tiles within the various viewport regions, as shown with varying transparencies in Figure 3.1. Tiles in the grey region receive the lowest bitrate, prioritizing the transmission of tiles within the user's immediate field of view.

By employing this system architecture, the proposed ABR streaming system dynamically adjusts bitrate allocation based on predicted user viewpoints. This adaptive approach significantly improves streaming quality, delivering a captivating and immersive VR experience.

3.2 Multi-Viewpoint Prediction

In order to optimize content segments for high-quality streaming, considering the user's anticipated future viewpoint, we propose a multimodal spatial-temporal attention multi-viewpoint prediction algorithm. Unlike sequential processing models like LSTMs, our approach utilizes the transformer architecture [22], which directly captures temporal relationships among sequence elements through the self-attention mechanism. This direct spatial-temporal capturing enhances the efficiency and effectiveness of the prediction process significantly. In our proposed method, we introduce a multimodal spatial-temporal

encoder-decoder transformer model, enriched with a clustering algorithm, to predict multiple future viewpoints along with their associated probabilities. Unlike treating viewpoint trajectory prediction as a regression problem, which directly predicts coordinates, we frame it as a classification problem. This classification-based approach enables us to categorize and assign probabilities to different predicted viewpoints, offering a more comprehensive understanding of the user's anticipated behavior. The incorporation of the multimodal spatial-temporal attention transformer architecture and clustering algorithm into our framework ensures accurate and efficient prediction of multiple viewpoints. The transformer's capability to capture temporal dependencies and the clustering algorithm's capacity to handle complex patterns and relationships contribute to the robustness and effectiveness of the viewpoint prediction process.

3.2.1 Model Input and Output

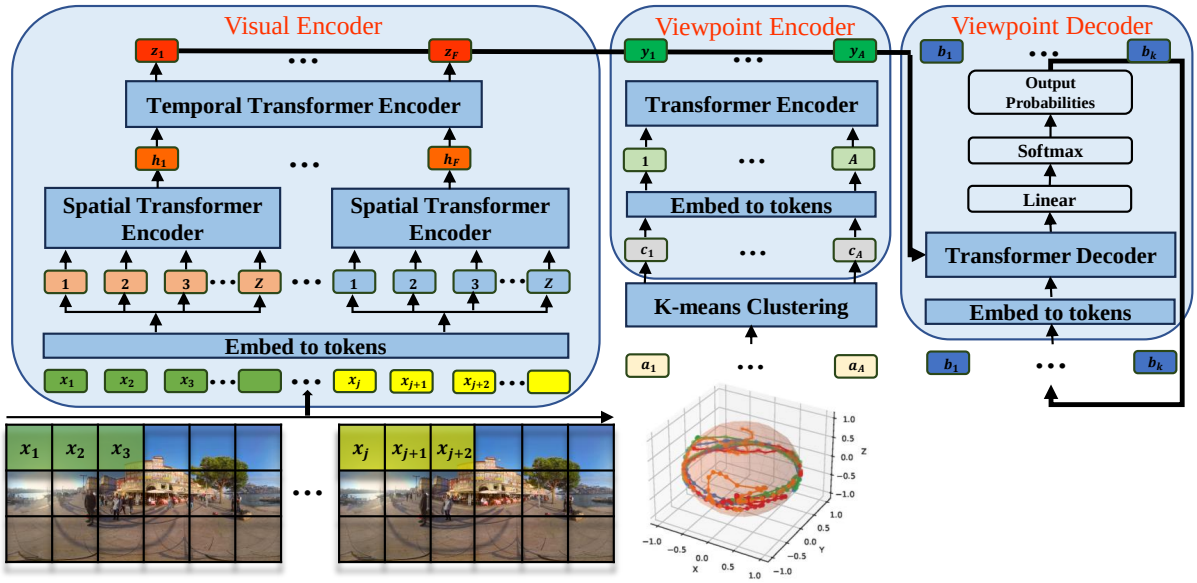

Figure 3.2: The architecture of the proposed multimodal spatial-temporal attention transformer for multiple-viewpoint prediction, employing a classification approach.

We present a multimodal spatial-temporal attention model to predict multiple viewpoint trajectories. Viewpoint positions are defined in a 3D Cartesian coordinate system.

With a historical set of viewpoints denoted as $\{a_i\}_{i=1}^{A}$, where each position $a_i \in \mathbb{R}^3$, and a sequence of video frames $\{f_j\}_{j=1}^{F}$, where each frame $f_j \in \mathbb{R}^{H \times W \times C}$ with H, W, and C representing height, width, and channel number. Our approach is centered on predicting J potential future viewpoint trajectories $\{b_r^{\zeta}\}_{r=1}^{B}$ from time step 1 to time step B seconds, where $\zeta = 1, 2, ..., J$ and $r = 1, 2, ..., B$, leveraging the user's historical viewpoint trajectory $\{a_i\}_{i=1}^{A}$. As depicted in Fig. 3.2, our transformer architecture encompasses a visual encoder, a viewpoint encoder, and a viewpoint decoder. Our work adopts the encoder-decoder architecture proposed by Vaswani et al. [22] serves as the basis for our work. A detailed introduction to each module follows.

3.2.2 Positional Embedding

The transformer's ability to process viewpoints simultaneously brings an inherent challenge as it lacks awareness of the positional relationships among individual elements in the input sequence. To address this limitation, it becomes imperative to introduce positional information to aid the transformer in effectively understanding the input sequence. In our approach, we adopt the positional encoding mechanism proposed in [22] to overcome this challenge.

$$\begin{aligned} PE(pos, 2i) &= sin(pos/10000^{2i/d_{model}}) \\ PE(pos, 2i+1) &= cos(pos/10000^{2i/d_{model}}) \end{aligned} \tag{3.1}$$

In our system, we use sine and cosine functions with varying frequencies to compute the positional encoding. The variable *pos* represents the position, and each dimension of the positional encoding (denoted by i) spans from 0 to $d_{model}/2$. Employing sinusoids with frequencies ranging from 2π to $10000 \cdot 2\pi$, we ensure a unique encoding for each position in the input sequence. This positional encoding mechanism significantly enhances the transformer's understanding of the relative positions and order of the input elements. By incorporating positional encoding, we provide the transformer with essential contextual cues to distinguish the sequential flow of viewpoints within the sequence. As a result, our transformer model becomes adept at capturing sequential dependencies

and comprehending the positional significance of each viewpoint. This enhancement empowers the transformer to make more informed predictions, considering both the content and temporal context of the viewpoints, leading to improved performance and higher prediction accuracy.

3.2.3 Visual Encoder

Every frame undergoes division into Z patches where each patch $x_i \in \mathbb{R}^{h \times w}$, with i ranging from 1 to Z, and h and w representing height and width. Similar to the methodology in ViT [78], each patch undergoes tokenization with both patch and positional embedding. The spatial attention score for patch x_i in frame f_j is expressed as follows:

$$Attention_{spatial} = softmax(\frac{q_{(x_i,f_i)}}{\sqrt{d_k}} \cdot \{k_{(x_\alpha,f_j)}\}_{\alpha=1,\ldots,Z}^{T}) \tag{3.2}$$

Here, $q_{(x_i,f_j)}$ and $k_{(x_\alpha,f_j)}$ represent the query vector for patch x_i and the key vector for patch x_α in frame f_j, respectively. The embedding dimension is denoted by d_k. Following the spatial encoder, a representation is obtained for each temporal index h_i and concatenated into a matrix $H \in \mathbb{R}^{F \times d_k}$. Subsequently, this matrix is passed through a temporal encoder. The temporal attention score for frame f_j is defined as:

$$Attention_{temporal} = softmax(\frac{q_{(f_j)}}{\sqrt{d_k}} \cdot \{k_{(f_\beta)}\}_{\beta=1,\ldots,F}^{T}) \tag{3.3}$$

Here, $q_{(f_j)}$ and $k_{(f_\beta)}$ represent the query vector for frame f_j and the key vector for frame f_β, respectively. Following the processing by the temporal encoder, the resulting output tokens $\{z_j\}_{j=1}^{F}$, where $z_j \in \mathbb{R}^{d_k}$, are obtained from this encoder.

3.2.4 Viewpoint Encoder

In addressing trajectory prediction, the strategy shifts towards a classification paradigm rather than a regression approach that directly foretells coordinates. Through the application of the K-means clustering algorithm, the original position a_i is categorized into distinct clusters, yielding centroids. Instead of using original viewpoint coordinates, we

associate each viewpoint with a specific cluster and use the cluster's centroid as the representative input for the transformer model. This dimensionality reduction aims to enhance the prediction process's efficiency. These centroids, represented by c_i, are then subjected to tokenization employing both word embedding and position embedding. This embedding procedure empowers the model to effectively capture and process the temporal dynamics and relationships within the viewpoint sequence. The outcome viewpoint encoder comprises a set of output tokens $\{y_i\}_{i=1}^{A}$, $y_i \in \mathbb{R}^{d_k}$, which is the product of the viewpoint encoder's processing. These representations are then fed into the decoder, which is responsible for generating predictions for multiple subsequent viewpoints.

3.2.5 Viewpoint Decoder

Deriving from the encoder embeddings, formed by the fusion of visual and viewpoint tokens $\{z_j\}_{j=1}^{F}$ and $\{y_i\}_{i=1}^{A}$, the viewpoint decoder generates the set of viewpoints $\{b_r^{\zeta}\}_{r=1}^{B}$. To determine the probabilities associated with each predicted viewpoint, we apply the softmax activation function. To improve the prediction process, the decoder incorporates previous outputs as inputs for subsequent predictions, allowing the model to benefit from its past predictions and enhance accuracy.

3.3 Rate Adaptation

3.3.1 QoE Metric Design

Let's consider a VR video that is divided into M tiles and F segments, with each segment having a duration of Δt (set to 1 second). The t-th segment, where $t \in 1, 2, ..., M$, represents the segment at time step t. In this context, we aim to predict J viewpoint trajectories, resulting in the VR video being comprised of J viewport regions, while the remaining tiles outside the viewport regions form the rest region.

For the t-th segment, we denote the viewing probability of the j-th viewport region

as ψ_t^j. Buffer occupancy at time t is represented by b_t, while the network throughput at time t is defined as φ_t. The bitrate of the m-th tile, where $m \in 1, 2, ..., M$, is denoted as r_t^m. Additionally, we use $\Phi(tile_t^m)$ to indicate the size of the m-th tile with bitrate r_t^m. Consequently, the total size of segment t, denoted as $\Phi(Seg_t)$, can be calculated using the following equation:

$$\Phi(Seg_t) = \sum_{m=1}^{M} \Phi(tile_t^m) \tag{3.4}$$

Furthermore, the size of region j, denoted as $\Phi(Reg_t^j)$, can be computed as the sum of the sizes of all tiles within the region j:

$$\Phi(Reg_t^j) = \sum_{m \in Reg_t^j} \Phi(tile_t^m) \tag{3.5}$$

In the above equation, Reg_t^j represents all tiles within the region j.

The QoE model for segment t takes into account several crucial aspects, as detailed below:

Viewport Quality: Since the user's primary focus is on the content within the viewport region, determining the quality of this region plays a pivotal role in assessing QoE. By considering J viewport regions, the viewport quality is evaluated as the weighted average quality of these regions, represented as:

$$QoE_t^1 = \sum_{j=1}^{J} \psi_t^j \cdot q_t^j \tag{3.6}$$

Here, q_t^j indicates the quality (bitrate) of the j-th viewport region in segment t, where higher bitrates correspond to better perceived video quality.

Viewport Temporal Variation: Abrupt changes in bitrate between consecutive viewports can lead to discomfort, such as dizziness or headaches. To address this concern, the temporal variation between viewports is considered, quantifying the difference in bitrate between the same viewport regions of two successive segments, given by:

$$QoE_t^2 = \sum_{j=1}^{J} \psi_t^j \cdot (q_t^j - q_{t-1}^j) \tag{3.7}$$

Spatial Variation: Users may view different viewport regions with corresponding probabilities, and quality variations between these regions can lead to distorted edges. Therefore, the spatial variation between regions is also taken into account, expressed as:

$$QoE_t^3 = \frac{1}{2}\sum_{j=1}^{J}\sum_{i\in U_t^j}\psi_t^j \cdot \psi_t^i \cdot (q_t^j - q_t^i) \tag{3.8}$$

Here, U_t^j represents the set of all viewport regions excluding region j.

Rebuffering: Rebuffering occurs when the time required to download a segment exceeds the buffer occupancy, causing interruptions in playback. Rebuffer time is denoted as:

$$QoE_t^4 = \max\left(\frac{\Phi(Seg_t)}{\varphi_t} - b_t + \Delta t, 0\right) \tag{3.9}$$

The overall QoE is calculated as a weighted sum of the above factors, represented by Equation 3.10, where α_* are weighting parameters adjusted based on the user's preferences:

$$QoE_t = \alpha_1 QoE_t^1 - \alpha_2 QoE_t^2 - \alpha_3 QoE_t^3 - \alpha_4 QoE_t^4 \tag{3.10}$$

Due to the distinct preferences embodied in various Quality of Experience (QoE) parameters, four sets of weighting parameters, denoted as $(\alpha_1, \alpha_2, \alpha_3, \alpha_4)$, have been chosen to address four distinct QoE objectives. The α_1 is geared towards maximizing the quality of viewport regions, given their pivotal role in influencing overall QoE. The $\alpha_2, \alpha_3, \alpha_4$ are used to minimize temporal quality variation, spatial quality variation, and rebuffering time, respectively.

3.3.2 Multi-Agent RL Problem Formulation

The process of determining the bitrate for multiple viewport regions can be expressed as a Dec-POMDP, as mentioned in Oliehoek's concise paper [79]. A Dec-POMDP is characterized by a tuple, denoted as $(\mathbb{D}, \mathbb{S}, \mathbb{A}, P, \mathbb{O}, O, R, \gamma)$, and can be described as follows:

- $\mathbb{D}$ represents a collection of n agents, designated as $1, \ldots, n$.
- $\mathbb{S}$ denotes the set of possible states.
- $\mathbb{A}$ comprises a set of joint actions, where each agent $i \in \mathbb{D}$ has access to its corresponding action set $\mathbb{A}^i$.
- The transition probability function P describes the impact of a joint action on the environment.
- $\mathbb{O}$ consists of joint observations, with $\mathbb{O}^i$ representing the observation set available to agent i.
- The observation probability function is denoted as O.
- The reward function R captures the overall reward obtained by the agents based on their decisions.
- Finally, γ (a value within the range of $(0, 1]$) represents the discount factor used in the process.

At each time stage t, individual agents in the system make decisions by selecting actions a_t^i from their respective action sets $\mathbb{A}^i$, based on the observations o_t^i available to them. These separate actions collectively form a joint action denoted as $\boldsymbol{a_t} = a_t^1, \ldots, a_t^n$. As a result of this joint action, the global state transitions from the current state $\boldsymbol{s_t} \in \mathbb{S}$ to the subsequent state $\boldsymbol{s_{t+1}} \in \mathbb{S}$, governed by the transition probability function $P(\boldsymbol{s_{t+1}}|\boldsymbol{s_t}, \boldsymbol{a_t})$. Furthermore, each agent receives a new observation o_{t+1}^i based on the observation probability function $O(o_{t+1}^i|\boldsymbol{s_{t+1}}, \boldsymbol{a_t})$, while the team receives a collective reward according to the reward function $R(\boldsymbol{s_t}, \boldsymbol{a_t})$. The discount factor plays a critical role in balancing immediate and future rewards to optimize decision-making. The Dec-POMDP framework's primary objective is to determine a joint policy $\boldsymbol{\pi}$, consisting of a set of policies, that maximizes the discounted expected cumulative rewards at a global

level. This optimization ensures an efficient ABR strategy. To clarify the terminology in our proposed approach:

Set of Agents: In our system, each individual viewport region is associated with a dedicated agent responsible for determining the bitrate specifically tailored to that region. This agent utilizes its observations and context to make informed decisions regarding the optimal bitrate for the assigned viewport region.

States and Observations: At each time step t, each Agent i receives a specific local observation denoted as $o_t^i = (\vec{\sigma_t^i}, \vec{\varphi_t^i}, \vec{\omega_t^i}, \psi_t^i, c_t^i, l_t^i, b_t^i)$. Here, $\vec{\sigma_t^i}$ captures the download times of the previous k segments, $\vec{\varphi_t^i}$ represents the network throughput for those same k segments, $\vec{\omega_t^i}$ is a vector containing the available sizes of the viewport region for the next segment, ψ_t^i indicates the viewport probability specific to viewport region i, c_t^i denotes the remaining number of segments, l_t^i represents the previous bitrate chosen, and b_t^i reflects the current buffer level.

The global state $\boldsymbol{s_t} = o_t^1, \ldots, o_t^n$ represents the concatenated observations from all agents. This global state comprises information from all the individual agents, allowing for a comprehensive understanding of the system's current status and providing the necessary context for each agent's decision-making process. By sharing relevant observations through the global state, the agents can coordinate their actions effectively and optimize the adaptive bitrate strategy for the entire system.

Set of Actions: At each time step t, Agent i selects an action a_t^i, which corresponds to determining the bitrate for viewport region i for the subsequent segment. The individual actions of all agents are concatenated to form the joint action denoted as $\boldsymbol{a_t} = (a_t^1, \ldots, a_t^n)$. This joint action represents the collective decisions made by all agents in the system regarding the bitrate allocation for their respective viewport regions. By coordinating their actions through the joint action, the agents contribute to the overall optimization of the adaptive bitrate strategy, aiming to enhance the QoE and streaming performance for the users.

Rewards: The global reward resulting from the joint action, denoted as r_t, is rep-

resented as the QoE metric QoE_t (as defined in Eq. 3.10). This global reward provides an overall evaluation of the collective performance of the system. By maximizing the global reward, the agents strive to collectively optimize the streaming quality and user satisfaction, fostering an efficient and effective ABR streaming system. Within the decentralized system, each agent receives a local reward that assesses the consequences of its chosen action within a given state. Specifically, the local reward at time t is derived from the global reward, as as $\psi_t^i \cdot q_t^j + \psi_t^j \cdot (q_t^i - q_{t-1}^i) + \sum_{j \in U_t^i} \psi_t^i \cdot \psi_t^j \cdot (q_t^i - q_t^j) + QoE_t^4$. The metric represents the perceived quality and experience associated with the selected bitrate allocation strategy for the agent's assigned viewport region.

3.3.3 Multi-Agent PPO Solution

In this study, we propose MAPPO, a MARL algorithm for optimizing bitrate adaptation for streaming VR videos in multi-agent environments. MAPPO is an extension of Proximal Policy Optimization (PPO), originally introduced by Schulman et al. in their work [80].

For each agent i, a local policy is defined as $\pi_{\theta_i}^i$, representing the agent's decision-making strategy based on its specific parameters θ_i. Additionally, the joint policy $\boldsymbol{\pi_\theta(a|s)}$ captures the probability of selecting a joint action $\boldsymbol{a}$ given the global state $\boldsymbol{s}$. As each agent independently makes decisions at each time step, the joint policy can be expressed as $\boldsymbol{\pi_\theta(a|s)} = \prod_{i \in \mathbb{D}} \pi_{\theta_i}^i(a^i|s^i)$. This formulation allows us to take into account all agents' contributions jointly, as they determine jointly the optimal joint action based on their individual local policies and the current global situation. In dynamic and complex multi-agent environments, we aim to improve the adaptive bitrate streaming performance by leveraging MARL and MAPPO.

The ultimate goal of the multiple agents is to collaboratively uncover the most effective joint policy $\boldsymbol{\pi_\theta}$ that maximizes the overall discounted cumulative expected reward. This cumulative reward, often referred to as the return, is defined after the joint action $\boldsymbol{a_t}$ at time t as the sum of future rewards discounted by a factor of γ. The expression

can be written as follows:

$$G_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + ... = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \tag{3.11}$$

In this equation, $r_{t+1}, r_{t+2}, r_{t+3}, ...$ represent the sequence of rewards obtained at subsequent time steps, while γ serves as the discount factor, balancing the significance of immediate rewards versus future rewards. The cumulative return G_t reflects the total reward an agent can anticipate receiving starting from time step t and extending into the future.

The expected return, serving as the global state value function under policy $\boldsymbol{\pi_\theta}$ for global state $\boldsymbol{s}$, is mathematically defined as follows:

$$V^{\boldsymbol{\pi_\theta}}(\boldsymbol{s}) = E_{\boldsymbol{\pi_\theta}}[G_t | \boldsymbol{s_t} = \boldsymbol{s}] \tag{3.12}$$

The definition of the global state-action value function under the policy $\boldsymbol{\pi_\theta}$ is as follows:

$$Q^{\boldsymbol{\pi_\theta}}(\boldsymbol{s}, \boldsymbol{a}) = E_{\boldsymbol{\pi_\theta}}[G_t | \boldsymbol{s_t} = \boldsymbol{s}, \boldsymbol{a_t} = \boldsymbol{a}] \tag{3.13}$$

Hence, the global state value function can be expressed as:

$$V^{\boldsymbol{\pi_\theta}}(\boldsymbol{s}) = \sum_{\boldsymbol{a} \in \mathbb{A}} Q^{\boldsymbol{\pi_\theta}}(\boldsymbol{s}, \boldsymbol{a}) \boldsymbol{\pi_\theta}(\boldsymbol{a}|\boldsymbol{s}) \tag{3.14}$$

Subsequently, the objective function is formulated as:

$$\begin{aligned} J(\boldsymbol{\theta}) &= \sum_{\boldsymbol{s} \in \mathbb{S}} d^{\boldsymbol{\pi_\theta}}(\boldsymbol{s}) V^{\boldsymbol{\pi_\theta}}(\boldsymbol{s}) \\ &= \sum_{\boldsymbol{s} \in \mathbb{S}} d^{\boldsymbol{\pi_\theta}}(\boldsymbol{s}) \sum_{\boldsymbol{a} \in \mathbb{A}} Q^{\boldsymbol{\pi_\theta}}(\boldsymbol{s}, \boldsymbol{a}) \boldsymbol{\pi_\theta}(\boldsymbol{a}|\boldsymbol{s}) \end{aligned} \tag{3.15}$$

where $d^{\boldsymbol{\pi_\theta}}(\boldsymbol{s})$ denotes the stationary distribution of the Markov chain for policy $\boldsymbol{\pi_\theta}$. In order to maximize the objective function, the gradient of $J(\boldsymbol{\theta})$ with respect to $\boldsymbol{\theta}$ is

computed using:

$$\begin{aligned}
\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) &= \nabla_{\boldsymbol{\theta}} \sum_{\boldsymbol{s}\in\mathbb{S}} d^{\boldsymbol{\pi}_{\boldsymbol{\theta}}}(\boldsymbol{s}) \sum_{\boldsymbol{a}\in\mathbb{A}} Q^{\boldsymbol{\pi}_{\boldsymbol{\theta}}}(\boldsymbol{s},\boldsymbol{a}) \boldsymbol{\pi}_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{s}) \\
&\propto \sum_{\boldsymbol{s}\in\mathbb{S}} d^{\boldsymbol{\pi}_{\boldsymbol{\theta}}}(\boldsymbol{s}) \sum_{\boldsymbol{a}\in\mathbb{A}} Q^{\boldsymbol{\pi}_{\boldsymbol{\theta}}}(\boldsymbol{s},\boldsymbol{a}) \nabla_{\boldsymbol{\theta}} \boldsymbol{\pi}_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{s}) \\
&= \sum_{\boldsymbol{s}\in\mathbb{S}} d^{\boldsymbol{\pi}_{\boldsymbol{\theta}}}(\boldsymbol{s}) \sum_{\boldsymbol{a}\in\mathbb{A}} \boldsymbol{\pi}_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{s}) Q^{\boldsymbol{\pi}_{\boldsymbol{\theta}}}(\boldsymbol{s},\boldsymbol{a}) \frac{\nabla_{\boldsymbol{\theta}} \boldsymbol{\pi}_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{s})}{\boldsymbol{\pi}_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{s})} \\
&= E_{\boldsymbol{\pi}}[Q^{\boldsymbol{\pi}_{\boldsymbol{\theta}}}(\boldsymbol{s},\boldsymbol{a}) \nabla_{\boldsymbol{\theta}} \log \boldsymbol{\pi}_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{s})]
\end{aligned} \tag{3.16}$$

where $E_{\boldsymbol{\pi}}$ denotes the expectation over distributions of state $\boldsymbol{s}$ and action $\boldsymbol{a}$, with $\boldsymbol{s}$ following the stationary distribution $d^{\boldsymbol{\pi}_{\boldsymbol{\theta}}}$ and $\boldsymbol{a}$ following the policy $\boldsymbol{\pi}_{\boldsymbol{\theta}}$. To assess the relative superiority of an action compared to others, we employ the concept of an advantage function. The advantage function captures the relative advantage of a specific action and can be defined as:

$$A^{\boldsymbol{\pi}_{\boldsymbol{\theta}}}(\boldsymbol{s},\boldsymbol{a}) = Q^{\boldsymbol{\pi}_{\boldsymbol{\theta}}}(\boldsymbol{s},\boldsymbol{a}) - V^{\boldsymbol{\pi}_{\boldsymbol{\theta}}}(\boldsymbol{s}) \tag{3.17}$$

Consequently, the gradient of $J(\boldsymbol{\theta})$ can be rewritten utilizing the advantage function [81]:

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = E_{\boldsymbol{\pi}}[A^{\boldsymbol{\pi}_{\boldsymbol{\theta}}}(\boldsymbol{s},\boldsymbol{a}) \nabla_{\boldsymbol{\theta}} \log \boldsymbol{\pi}_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{s})] \tag{3.18}$$

As the parameters θ^i and the joint policy $\boldsymbol{\pi}_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{s}) = \prod_{i\in\mathbb{D}} \pi^i_{\theta_i}(a^i|s^i)$ are independent, the partial derivative of $J(\boldsymbol{\theta})$ with respect to the local parameter θ^i can be expressed as:

$$\nabla_{\theta^i} J(\boldsymbol{\theta}) = E_{\boldsymbol{\pi}}[A^{\boldsymbol{\pi}_{\boldsymbol{\theta}}}(\boldsymbol{s},\boldsymbol{a}) \nabla_{\theta^i} \log \pi^i_{\theta^i}(a^i|s^i)] \tag{3.19}$$

By obtaining all the partial derivatives $\nabla_{\theta^1} J(\boldsymbol{\theta}), \nabla_{\theta^2} J(\boldsymbol{\theta}), ..., \nabla_{\theta^n} J(\boldsymbol{\theta})$, we can obtain the gradient of $J(\boldsymbol{\theta})$ at $\boldsymbol{\theta}$. If each agent's stochastic gradient ascent, performed on their respective θ^i, results in only a small displacement from $\boldsymbol{\theta}$, we can enhance the joint policy globally.

It is important to emphasize that once the advantage function $A^{\boldsymbol{\pi}_{\boldsymbol{\theta}}}(\boldsymbol{s},\boldsymbol{a})$ is computed, it enables the local calculation of $\nabla_{\theta^i} J(\boldsymbol{\theta})$ for each agent. This capability allows for the implementation of centralized training and distributed execution, extending the application of policy gradient algorithms to the domain of MARL. In the scenario depicted

in Figure 3.3, each agent receives its specific observation, which contains unique player information. At the same time, joint actions and global observations are supplied to the critics, providing a comprehensive view of the multi-agent environment. The centralized critics enhance training stability by considering the states and actions of all agents collectively, resulting in a more robust and efficient learning process. On the other hand, during the exploration and execution phases, the distributed agents do not require access to this additional information, enabling them to operate independently. This combination of centralized training and distributed execution ensures that the proposed MAPPO algorithm efficiently optimizes bitrate adaptation streaming for VR video in a multi-agent setting, facilitating a seamless and immersive streaming experience for users.

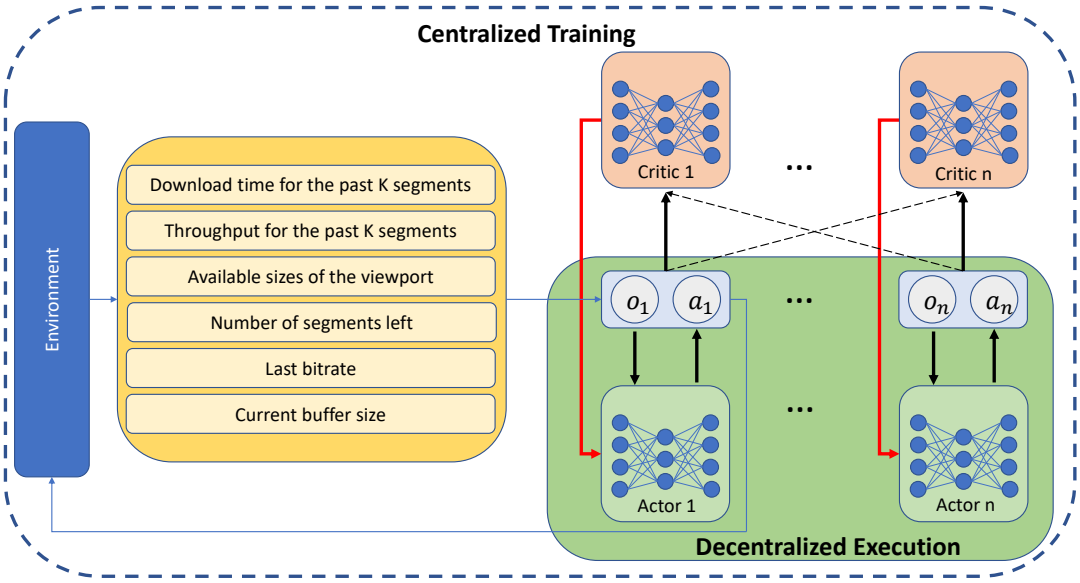

Figure 3.3: The proposed MARL with MAPPO algorithm operating within the CTDE framework.

The implementation of MAPPO is guided by the equation presented in Eq. 3.19. To achieve efficient and effective multi-agent learning, MAPPO employs two separate networks for each agent: a policy network (actor) with parameters θ^i, and a value function network (critic) with parameters ϕ^i. During the learning process, a clipped probability ratio between old and new policies is utilized, ensuring that the stochastic gradient

ascent of each agent's θ^i remains constrained within a small range of $\boldsymbol{\theta}$. To generate training data, trajectories are collected from the environment, forming sequences denoted as $\tau = \boldsymbol{s}0, \boldsymbol{a}0, r_0, ..., \boldsymbol{s}T-1, \boldsymbol{a}T-1, r_{T-1}, \boldsymbol{s}_T$, where T represents the trajectory horizon. At each time step t, the estimated discounted return $\hat{G}_t$ for the trajectory τ is computed using the Bellman equation:

$$\hat{G}_t = \sum_{i=t}^{T-1} \gamma^{i-t} r_t + \gamma^{T-t} V_{\boldsymbol{\phi}}(\boldsymbol{s}_T) \tag{3.20}$$

Here, $V_{\boldsymbol{\phi}}$ refers to the global value function parameterized by $\boldsymbol{\phi}$.

The truncated Generalized Advantage Estimation (GAE), introduced by Schulman et al. [80], a method for calculating the estimated global advantage function on trajectory τ, as indicated by $\hat{A}(\boldsymbol{s}t, \boldsymbol{a}t)$. The GAE utilizes the temporal difference residual δ_t to estimate discrepancy between the immediate reward r_t and the sum of the discounted value of the next state $\gamma V\boldsymbol{\phi}(\boldsymbol{s}t+1)$ and the value of the current state $V_{\boldsymbol{\phi}}(\boldsymbol{s}_t)$.

Mathematically, the GAE estimation is computed as follows:

$$\begin{aligned} \hat{A}(\boldsymbol{s}_t, \boldsymbol{a}_t) &= \delta_t + (\gamma\lambda)\delta_{t+1} + ... + (\gamma\lambda)^{T-1-t}\delta_{T-1} \\ &= \sum_{l=0}^{T-1-t} (\lambda\gamma)^l \delta_{t+l} \end{aligned} \tag{3.21}$$

In the GAE method, the hyperparameter $\lambda \in [0, 1]$ balances the trade-off between bias and variance. In order to improve the accuracy of the advantage function estimation, the parameter λ controls the relative weighting of immediate rewards (bias) and future rewards (variance). Thus, the GAE method is one of the key components in the MAPPO algorithm that enables efficient and accurate policy updates to maximize the cumulative rewards and enhance overall system stability.

The PPO-Clip algorithm, introduced by Schulman et al. in their work [80], can be incorporated into the MAPPO framework to enhance training stability and prevent excessively large policy updates. PPO-Clip defines the surrogate objective function for agent i follows:

$$J^{CLIP}(\theta^i) = E_{\boldsymbol{s}_t, \boldsymbol{a}_t}[min(r_t^i(\theta^i)\hat{A}(\boldsymbol{s}_t, \boldsymbol{a}_t), g(\epsilon, \hat{A}(\boldsymbol{s}_t, \boldsymbol{a}_t)))] \tag{3.22}$$

where

$$g(\epsilon, \hat{A}(\boldsymbol{s}_t, \boldsymbol{a}_t)) = \begin{cases} (1+\epsilon)\hat{A}(\boldsymbol{s}_t, \boldsymbol{a}_t), \hat{A}(\boldsymbol{s}_t, \boldsymbol{a}_t) \geq 0 \\ (1-\epsilon)\hat{A}(\boldsymbol{s}_t, \boldsymbol{a}_t), \hat{A}(\boldsymbol{s}_t, \boldsymbol{a}_t) < 0 \end{cases} \tag{3.23}$$

Here, ϵ is a hyperparameter that controls the acceptable range of policy divergence, often set to 0.2. The ratio $r_t^i(\theta^i)$ represents the probability ratio between the new policy $\pi_{\theta^i}(a_t^i|s_t^i)$ and the old policy $\pi_{\theta^i_{old}}(a_t^i|s_t^i)$ for agent i at time step t. It quantifies the difference between the updated policy and the previous policy. To prevent significant deviations or drastic changes in the policy distribution, the PPO-Clip algorithm bounds the value of $r_t^i(\theta^i)$ within a specific range, namely $[1-\epsilon, 1+\epsilon]$, effectively clipping the policy update. The function $g(\epsilon, \hat{A}(\boldsymbol{s}_t, \boldsymbol{a}_t))$ plays a crucial role in controlling the policy update range:

$$r_t^i(\theta^i) = \frac{\pi_{\theta^i}(a_t^i|s_t^i)}{\pi_{\theta^i_{old}}(a_t^i|s_t^i)} \tag{3.24}$$

The PPO-Clip constraint keeps policy updates controlled during training, balancing exploration and exploitation. As a result, the MAPPO algorithm in bitrate adaptation streaming for VR video within a multi-agent environment performs more effectively and converges faster.

To update the policy network parameter θ^i for agent i, the MAPPO algorithm maximizes the PPO-Clip objective using a dataset of trajectories denoted as $D = \tau$. The objective is to find the maximum of the following expression:

$$\theta^i = \arg\max_{\theta^i} \frac{|D|T}{1} \sum_{\tau \in D} \sum_{t=0}^{T-1} J^{CLIP}(\theta^i) \tag{3.25}$$

In this context, $|D|$ represents the number of trajectories in the dataset D, and T signifies the time horizon of each trajectory. The MAPPO algorithm maximizes the sum of clipped surrogate objective functions over all time steps and trajectories. On the other hand, the value network parameter ϕ^i for agent i undergoes updating through the minimization of the mean-squared error. This process involves computing the gradient of the mean-squared error with respect to ϕ^i and employing gradient descent updates to

iteratively improve the value network.

$$\begin{aligned} \phi^i &= \arg\min_{\phi^i} L(\phi^i) \\ &= \arg\min_{\phi^i} \frac{1}{|D|T} \sum_{\tau \in D} \sum_{t=0}^{T-1} (V_{\phi^i}(\boldsymbol{s_t}) - \hat{G}_t)^2 \end{aligned} \tag{3.26}$$

Here, $V_{\phi^i}(\boldsymbol{s_t})$ represents the predicted value function for the state $\boldsymbol{s_t}$ using the current value network parameters ϕ^i, and $\hat{G}_t$ denotes the estimated discounted return for the trajectory τ at time step t. To minimize mean-squared error between estimated returns and predictions, the value network is trained. This training process ensures that the value network provides accurate estimates of future rewards, which is crucial to guiding policy improvement during training. MAPPO learns and improves the agents' policies for bitrate adaptation streaming of VR video in a multi-agent environment by iteratively updating the policy and value networks.

During the training process, the parameters θ^i of the policy network for agent i and ϕ^i of the value network for agent i are updated using stochastic gradient ascent and descent, respectively. These updates aim to optimize the PPO-Clip objective for policy improvement and minimize the mean-squared error for value function estimation. The update for the policy network θ^i is given by:

$$\theta^i \leftarrow \theta^i + \alpha \nabla_{\theta^i} J^{CLIP}(\theta^i) \tag{3.27}$$

Here, α represents the learning rate for the policy network, and $\nabla_{\theta^i} J^{CLIP}(\theta^i)$ denotes the gradient of the PPO-Clip objective with respect to the policy network's parameters θ^i. As the stochastic gradient ascent step updates the policy parameters to maximize the PPO-Clip objective, the performance of the policy is improved.

On the other hand, the update for the value network ϕ^i is performed through stochastic gradient descent and is given by:

$$\phi^i = \phi^i - \beta \nabla_{\theta^i} L(\phi^i) \tag{3.28}$$

In this equation, β represents the learning rate for the value network, and $\nabla_{\theta^i} L(\phi^i)$ denotes the gradient of the mean-squared error loss with respect to the value network's

parameters ϕ^i. Stochastic gradient descent updates the value network parameters in a direction that minimizes the mean-squared error loss, making the value function more accurate at estimating future rewards. When the policy and value networks are updated iteratively during training, their performance gradually improves and converges to optimal values. This leads to more effective bitrate adaptation in multi-agent VR video streaming.

3.4 Experiment

3.4.1 Implementation Details

System setup: For performance evaluation of the proposed ABR streaming system, a testbed is established, as shown in Figure 3.4, consisting of three main components: a VR video host, a client, and a simulated network implemented using NS3 [82]. The testbed leverages the MMSys18 dataset [83], which provides diverse viewpoint trajectories suitable for the study's objectives. VR videos in the dataset are segmented into 1-second durations, each containing a grid of $6 \times 12 = 72$ tiles. To simulate adaptive bitrate streaming, FFmpeg is employed to encode videos at different bitrate levels, ranging from 0.3 Mbps to 4.3 Mbps, enabling multiple quality levels. In the MARL approach, three agents are responsible for predicting future viewpoint trajectories in different tiles. The VR video is stored on the video host, utilizing an nginx server to manage incoming video requests. Upon receiving a video request, the VR video player, developed using dash.js, streams the video to the client. This testbed facilitates experimental analysis and assessment of the ABR streaming system's performance within a realistic VR video streaming environment.

Hardware: Oculus Quest 2 is the VR headset produced by Facebook subsidiary Reality Labs. It is a standalone device that does not require a computer or external sensors. The headset includes a display, processor, and tracking system. It is extremely easy to use and set up because no additional devices are required to use it. In addition to

providing a sharp and immersive visual experience, the Oculus Quest 2 also features a fast-switch LCD screen with a resolution of 1832 x 1920 pixels per eye. In addition, the headset has a high refresh rate of 90Hz, which helps to reduce motion sickness and improve VR performance overall. With built-in cameras, Quest 2 tracks movement in physical space using inside-out tracking. There is no need for external sensors or tracking devices since the device allows full freedom of movement. Additionally, the controllers are tracked in 3D space, making interaction with virtual objects precise and intuitive. In general, the Oculus Quest 2 provides a powerful and accessible VR experience. It has a standalone design, a high-quality display, inside-out tracking, and an extensive library of content. Whether you want to play games, experience immersive storytelling, or connect with others, this VR experience is for you.

NS3 Simulated Network: As a proof of concept, three computers are utilized for different purposes. As shown in Fig. 3.4, the left computer with the Windows 10 OS (operating system) is used as a VR video server. The middle computer with Ubuntu 20.04 OS is utilized as a simulation host for the NS3 simulation network. The right computer with a Windows 10 OS designated as Oculus Quest host connects Oculus Quest 2 with USB-C cable, since Oculus Quest 2 is not compatible with the Ubuntu OS. The rendering server is connected to the simulation host via several Ethernet cables and a few switches. Meanwhile, the simulation host is also connected to the Oculus Quest 2 via the Oculus Quest host. In the simulation host, two network cards, NIC0 and NIC1, are employed. An NS3 model called tap bridge is utilized to connect the NS3 simulation network to the real network. Two tap devices (i.e. tap device0 and tap device1) are respectively created in two ghost nodes (i.e. ghost0 and ghost1), where the ghost node makes the tap device recognizable in the Linux system. The “IPC" link is the network tap mechanism in the underlying OS. Simply speaking, ghost node0 is the entrance to the NS3 simulation network, and ghost node1 is the exit to the real network. The blue dash line indicates the data flow from the server to the client.

Network traces: To evaluate the effectiveness of our proposed method and to compare

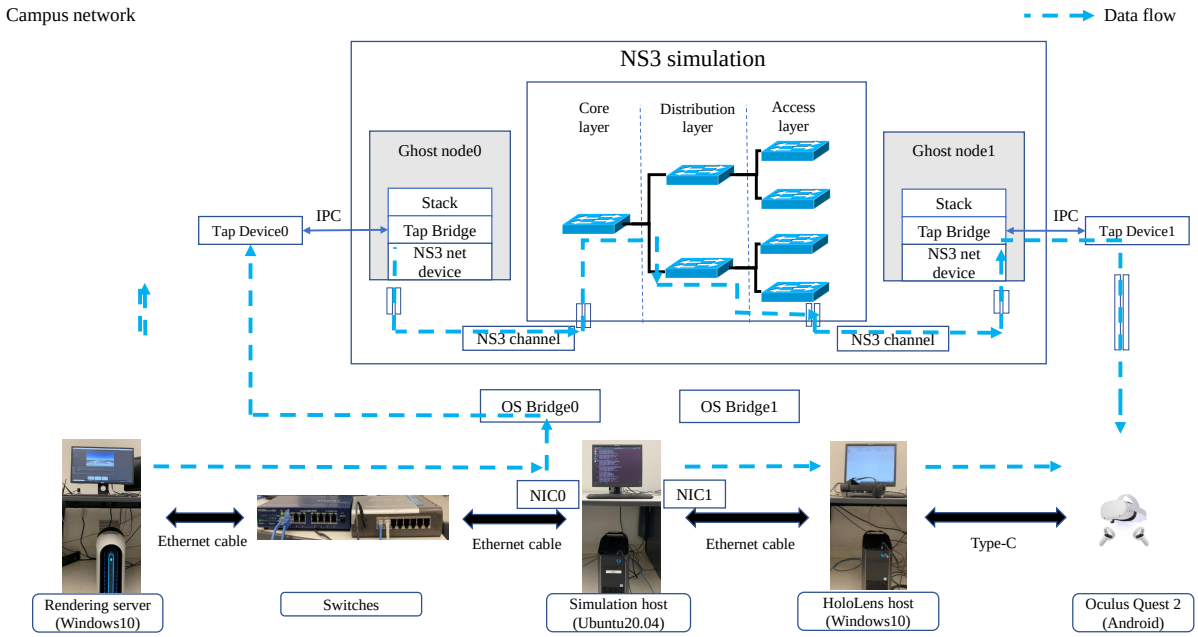

Figure 3.4: The testbed for our proposed system which contains a VR video host, a NS3-simulation host, and client.

it with other ABR methods in real-world network conditions, we leverage two publicly available network trace datasets: FCC and HSDPA. These datasets offer valuable real-world network traces that enable us to assess the performance of our approach in realistic network scenarios. To ensure fair experiments, we randomly split the dataset into a training set and a test set. About 80% of the traces are used for training, while the remaining 20% are kept aside for testing. However, it's worth noting that some network traces in these datasets may have limited throughput values that cannot support the lowest bitrate levels of VR videos. To address this limitation, we augment the network throughput values by 3 Mbps, ensuring that all bitrate levels can be accommodated by the network traces during the evaluation process. By employing these publicly available network trace datasets, we can comprehensively assess the performance and robustness of our proposed method. Comparisons with other ABR methods under real network conditions provide valuable insights into the effectiveness of our approach and its potential advantages in practical scenarios.

QoE objectives: Each of the four weighting parameters $(\alpha_1, \alpha_2, \alpha_3, \alpha_4)$ represents

a different objective in our evaluation, as shown in Eq. 3.10. A baseline weighting parameter set of (1, 1, 1, 1) aims to maximize the quality of viewport regions without specifically focusing on other factors. It provides a reference point for comparing other objectives. The second set of weighting parameters is (1, 2, 1, 1), which emphasizes minimizing temporal quality variations. The quality of the viewport regions should be optimized along with maintaining consistency across consecutive segments or frames. (1, 1, 2, 1) is the third set of weighting parameters that focuses on minimizing spatial quality variation. Therefore, ensuring a uniform quality level across viewport regions is as important as optimizing the overall quality. A fourth set of weighting parameters gives priority to minimizing rebuffering time (1, 1, 1, 2). Therefore, it is important to reduce interruptions during video playback due to buffering, while also taking into consideration the overall quality. With these four sets of weighting parameters, we can analyze different QoE objectives and gain a better understanding of how they impact the performance of our ABR streaming system. By analyzing different QoE factors, we are able to calculate the relative trade-offs and design an optimal system configuration. In real-world network conditions, we can fine-tune the system by selecting the appropriate weighting parameters to deliver an improved user experience for VR video streaming.

Transformer model training: The experiment was conducted using the PyTorch framework on a computer with an Intel Core i7-9700K CPU and a NVIDIA GTX 1080 Ti GPU. The input trajectory time window was defined as 1 second, and the output time windows ranged from 1 to 5 seconds. Within each second, we collected five data points, thereby creating a 5-point input sequence. For flexibility, we varied the length of the output sequence from 5 to 25 points, enabling comprehensive predictions. To streamline data processing, we performed coordinate normalization to the unit sphere and organized the data into 1500 bins, simplifying the information representation. The model's dimensionality (d_{model}) was set to 512, ensuring a suitable level of expressiveness while avoiding computational limitations. Throughout the training process, we carefully managed the batch size, employing a size of 100 to optimize the model efficiently. The

Adam optimization algorithm, incorporating a learning rate of 0.0005, was applied to iteratively update the model's parameters during training. To maintain adaptability, we introduced a decay rate of 0.99 for the learning rate, allowing the model to converge effectively.

MARL model training: The training of the MARL model is carried out on the same computer utilized for the transformer model, ensuring consistency and efficient resource usage. Both the actors and critics receive observations, which are concatenated into an array and then fed into a CNN for processing. The CNN architecture enables the model to capture spatial patterns and dependencies in the input data, crucial for making informed decisions. To control the learning process effectively, we set the learning rate to 0.0001, facilitating gradual updates to the model's parameters while avoiding overshooting during training. To balance immediate and future rewards, we assign a discount factor (γ) of 0.99, which allows the model to prioritize long-term rewards. A hyperparameter λ of 0.95 is also used in training to affect the GAE method. As a result of tuning this parameter, the bias and variance of the advantage function are balanced, ensuring accurate and reliable updates.

3.4.2 Results and Analysis for Viewpoint Prediction

Using two essential metrics, we compare our method with five existing ones. In this case, the great circle distance indicates the shortest distance on a sphere between a predicted viewpoint and the actual ground truth at a specific time point on a sphere. In order to evaluate prediction accuracy, this distance measurement is used. We also employ the average great-circle distance, which calculates the average error across all future time steps. As a result, this metric provides a comprehensive view of the overall performance of the model over time, providing insight into the model's ability to forecast trajectory in a consistent and reliable manner. Our method predicts multiple trajectories, so we use BMS (best of many samples) to evaluate it. This strategy involves generating multiple predictions (denoted as J) and selecting the trajectory with the minimum error when

compared to the ground truth trajectory as the final result. By adopting this approach, we ensure that our method's final predictions are the most accurate and reliable among the multiple possibilities.

To compare our proposed method fairly with existing approaches, we evaluate it against five other methods. The baseline method uses the last input element as the output trajectory, providing a simple reference for comparison. VPT360 [18] utilizes the encoder of the transformer model without the decoder to predict the viewport trajectory. Track [17] is based on LSTM, and DVMS [60] is designed for multiple-viewpoint trajectory prediction using the discrete variational multiple sequence approach . MANTRA [84], originally proposed for vehicle trajectory prediction using memory augmented networks, is modified in our case to predict multiple-viewpoint trajectories. To distinguish the number of predicted trajectories for each method, we use the notation *1, *2, and *3, where * represents the corresponding method. For example, ours1, ours2, and ours3 indicate that our proposed method predicts 1, 2, and 3 trajectories, respectively. This enables us to explore the impact of the number of predicted trajectories on the model's performance and identify the optimal configuration for our adaptive bitrate streaming system.

Figure 3.5 provides valuable insights into the performance comparison of our proposed method and other existing methods in terms of the great-circle distance metric. When the number of predicted trajectories (J) is set to 1, VPT360 achieves the best result, with our proposed method demonstrating the second-best performance. At the first second, our proposed method performs comparably to VPT360 in terms of the great-circle distance metric. However, as time progresses, the difference between VPT360 and our proposed method becomes more apparent. This disparity is primarily due to the error introduced by using the centroid instead of the actual viewpoint in the predictions made by our method.

On the other hand, when considering the DVMS, MANTRA, and our proposed method, as the number of trajectories (J) increases, the great-circle distance metric

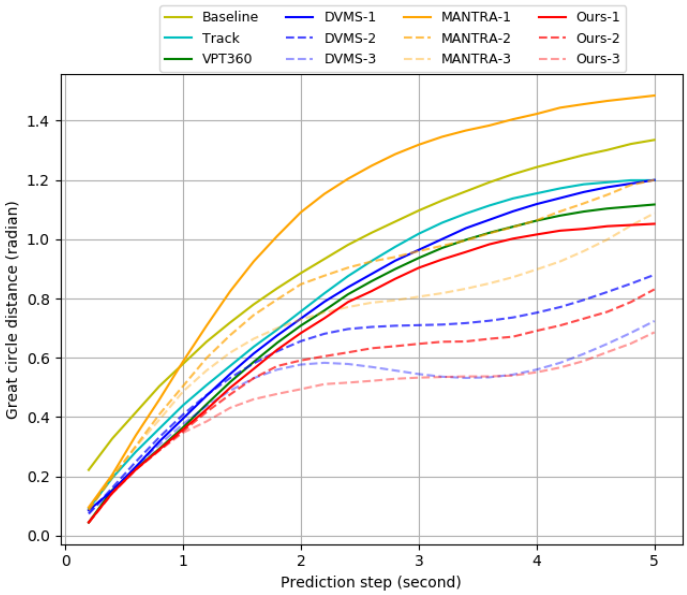

Figure 3.5: Comparison of results among various methods based on great circle distance. The *1, *2, and *3 represent the number of predicted trajectories for method *.

decreases. This indicates that considering multiple trajectories enhances the prediction accuracy for all three methods. Notably, when J is set to 3, our proposed method surpasses the performance of the other existing methods. It is essential to mention that while the proposed method may exhibit a larger distance compared to DVMS between the 3rd and 4th seconds when J is 3, the overall performance of the proposed method is superior across most of the time period under evaluation. This indicates the effectiveness and robustness of our approach in predicting multiple-viewpoint trajectories and optimizing the adaptive bitrate streaming system's performance in a realistic VR video streaming environment.

Table 3.1 provides a comprehensive overview of the experimental results for the average great-circle distance metric. The table highlights the best scores in bold and underlines the second-best scores. Upon analyzing the results, it becomes evident that our proposed method consistently outperforms the other existing methods, achieving the smallest average error across all time windows when J is set to 3. This emphasizes the effectiveness and superiority of our approach in predicting multiple-viewpoint trajecto-

Table 3.1: Comparative analysis of viewpoint prediction based on average great circle distance.

Method	Prediction Time Window				
	1st s	2nd s	3rd s	4th s	5th s
Baseline	0.409	0.592	0.734	0.848	0.939
VPT360 [18]	0.215	0.399	0.550	0.667	0.754
Track [17]	0.357	0.526	0.674	0.789	0.870
DVMS1 [60]	0.246	0.432	0.583	0.703	0.796
DVMS2	0.245	0.409	0.506	0.562	0.614
DVMS3	0.228	0.373	<u>0.438</u>	<u>0.464</u>	<u>0.501</u>
MANTRA1 [60]	0.337	0.624	0.830	0.969	1.068
MANTRA2	0.299	0.516	0.651	0.743	0.825
MANTRA3	0.292	0.473	0.576	0.646	0.717
Ours1	0.212	0.386	0.531	0.643	0.723
Ours2	<u>0.210</u>	<u>0.364</u>	0.452	0.505	0.557
Ours3	**0.209**	**0.329**	**0.393**	**0.430**	**0.468**

ries and optimizing the adaptive bitrate streaming system's performance in real network conditions.

Figure 3.6 showcases the capabilities of our proposed method in predicting multiple future viewpoint trajectories and their corresponding probabilities. In this example, we input a 1-second past viewpoint trajectory into the transformer model, which allows us to generate three potential future trajectories with their associated viewing probabilities. Trajectory 1 exhibits a remarkable alignment with the ground truth, indicating the high accuracy and reliability of our method in predicting the most likely trajectory that the user will follow. On the other hand, trajectories 2 and 3 capture alternative possible paths that the user might take, which are learned by the transformer from the viewpoint trajectories of other users. These alternative trajectories serve to account for the inherent

uncertainty in user behavior, enabling our method to provide a more comprehensive and probabilistic view of the future trajectories.

Figure 3.6: An illustration of multiple predicted trajectories generated by the transformer model.

3.4.3 Results and Analysis for Rate Adaptation

3.4.3.1 Methods for Comparison

To evaluate the performance of our proposed method, we compare it with several existing methods as follows:

- Buffer-based (BB) [85]: This approach employs a buffer-based algorithm to determine the bitrate for the entire VR video. Its objective is to keep the buffer occupancy at a minimum of 5 seconds. Additionally, if the buffer occupancy surpasses 15 seconds, the method automatically chooses the highest available bitrate.

- MPC [86]: This method determines the same bitrate for all viewport regions by considering the buffer level information and throughput prediction. It aims to maximize the given QoE.

- Pensieve [87]: The method utilizes buffer level information and throughput prediction to calculate a single bitrate for all viewport regions. Its primary goal is to maximize the QoE provided to the user.

- DRL360 [64]: DRL360 employs a SARL approach with A3C to determine a uniform bitrate for all tiles within the viewport while assigning the lowest bitrate to tiles outside the viewport. In our case, we utilize the DRL360 method to sequentially select the bitrate for multiple regions within the viewport, one at a time.

- 360SRL [20]: In contrast to simultaneously determining the bitrate for all tiles, 360SRL adopts a sequential approach by using SARL with A3C to select the bitrate for each tile individually, one after the other.

- MADDPG [88]: MADDPG stands as one of the most popular and versatile off-policy MARL algorithms. It was introduced by Lowe et al. as an extension of the DDPG algorithm [89]. The key innovation of MADDPG lies in its use of a centralized Q-function that incorporates observations and actions from all participating agents. This centralized approach addresses the non-stationarity challenge and enhances the stability of multi-agent training. Despite DDPG's original design for continuous actions, MADDPG ingeniously adopts the gumbel-softmax [90] trick to handle discrete actions effectively. By leveraging these combined techniques, MADDPG becomes a powerful tool for tackling diverse and complex MARL problems, making it a go-to choice in the field.

- QMix [91]: QMix emerges as a Q-learning algorithm specially tailored for multi-agent cooperative tasks featuring a global reward. The fundamental concept underlying QMix is value decomposition, which involves structuring the global Q function as the result of a "mixer" neural network. This mixer network takes the individual agent Q functions as inputs and combines them to produce the global Q function.

- Independent PPO (IPPO) [92]: IPPO stands as a decentralized MARL algorithm, where the agents operate independently and do not share information directly with each other. In this approach, each agent receives the global reward signal but updates its policy solely based on its local observation. The lack of direct communication between agents in IPPO allows for scalable and efficient implementations, as information exchange and coordination between agents are minimized. Instead, each agent makes decisions based on its own experiences and the global reward, facilitating individual learning and exploration within the multi-agent environment.

3.4.3.2 QoE Comparision

The experimental findings showcased in Figure 3.7 clearly validate the remarkable performance of our proposed method in contrast to other existing ABR methods across a diverse range of QoE objectives. Notably, our method excels when users prioritize high video quality, minimal temporal and spatial variations, and reduced rebuffering time, as evidenced by the results in Figures 3.7a, 3.7b, 3.7c, and 3.7d, respectively.

Regarding the QoE objective (1,1,1,1), our proposed method consistently outperforms BB, MPC, Pensieve, 360SRL, DRL360, MADDPG, QMix, and IPPO, demonstrating impressive improvements of 85.5%, 61.9%, 57.7%, 7.5%, 8.0%, 6.7%, 5.5%, and 4.8%, respectively, in terms of the normalized average QoE. Moreover, for QoE objectives (1,2,1,1), (1,1,2,1), and (1,1,1,2), our method consistently exhibits significant enhancements, yielding improvements ranging from 3.6% to 81.5%, 1.7% to 83.9%, and 4.5% to 84.4% in terms of QoE, respectively. These compelling results underscore the efficacy and adaptability of our proposed method across different QoE objectives. Collectively, our method consistently outshines existing ABR methods, attesting to its proficiency in elevating the user's Quality of Experience across various QoE objectives. These findings not only endorse the robustness of our approach but also establish its potential to revolutionize video streaming services, providing users with an enriched and tailored viewing experience in real-world scenarios.

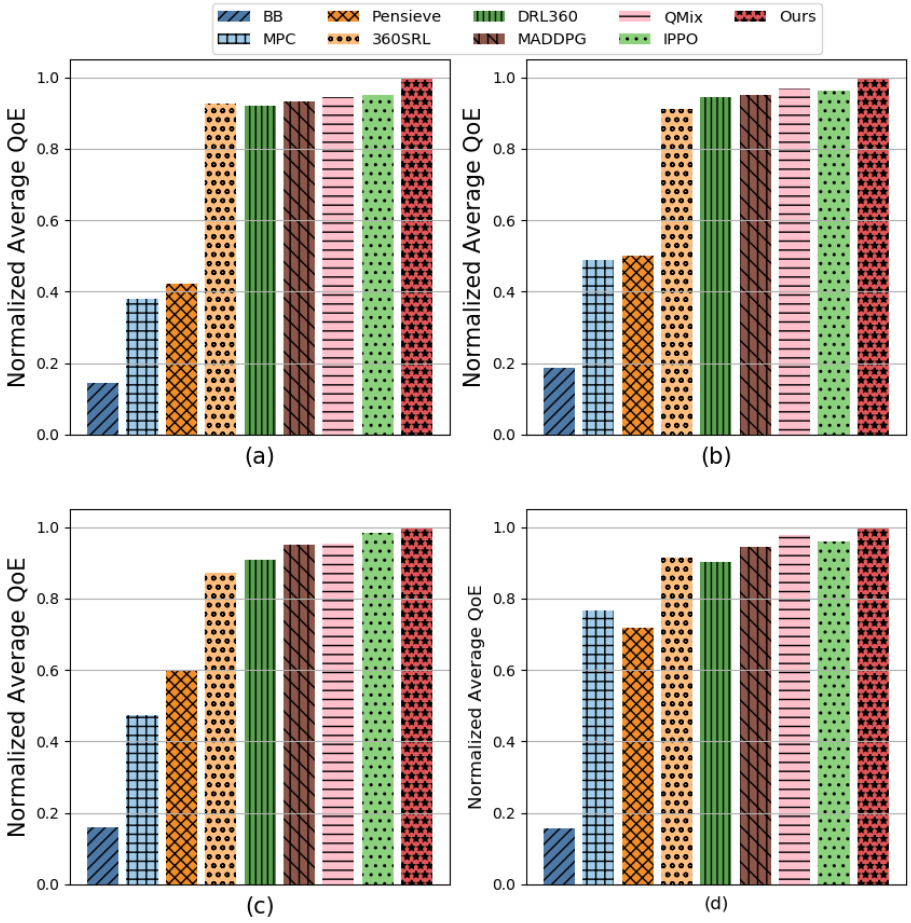

Figure 3.7: A comparison between our method and existing ABR algorithms using four different QoE objectives: (a) (1,1,1,1), (b) (1,2,1,1), (c) (1,1,2,1), and (d) (1,1,1,2).

The BB method, which relies solely on buffer occupancy without considering QoE metrics, consistently maintains the same QoE score across different objectives, but it underperforms compared to other methods. By selecting the bitrate for the entire VR video without considering the viewport and individual tiles, the BB method wastes a significant amount of bandwidth on tiles outside the viewport regions. Both the MPC and Pensieve methods allocate the same bitrate to all viewport regions, neglecting variations in user attention towards different regions. Consequently, bandwidth resources are not efficiently utilized, particularly in regions with lower viewing probabilities, resulting in inferior QoE scores for MPC and Pensieve when compared to SRL360, DRL360, MADDPG, QMix, IPPO, and our proposed method, which adaptively determine the bitrate for each tile based on the viewport probability. The QoE scores of SRL360, DRL360, MADDPG, QMix, IPPO, and our proposed method significantly outperform BB, MPC, and Pensieve across different QoE objectives. This highlights the advantage of adaptive bitrate determination for individual tiles, considering the probability of viewport visibility.

IPPO's performance is noteworthy as it closely rivals our proposed method and outperforms other SARL-based methods for most QoE objectives, except for (1,1,1,2). This indicates the advantage of employing MARL techniques. However, IPPO's performance falls slightly behind our proposed method due to the independent nature of its agents, lacking communication and coordination. Our proposed method, on the other hand, effectively leverages centralized Q-function and coordination among agents, yielding superior performance in diverse QoE scenarios.

3.4.3.3 QoE Breakdown

To gain insights into the average QoE improvements achieved by our proposed method, we perform an in-depth analysis of each approach concerning the constituent components of the QoE definition (Eq. 3.10). Focusing on the QoE objective (1,1,1,1) and illustrated in Fig. 3.8, we compare our method to existing ABR algorithms, paying close attention

to the average quality of the viewport regions, the penalties related to temporal quality variation, spatial quality variation, and rebuffering. These observations extend to the other QoE objectives as well.

The BB, MPC, and Pensieve methods determine bitrate for the entire VR video or all viewport regions, leading to suboptimal video quality due to underutilized bandwidth. Despite no penalties for spatial quality variation, these methods incur higher penalties for both temporal quality variation and rebuffering. In contrast, our proposed method, alongside SRL360, DRL360, MADDPG, QMix, and IPPO, dynamically allocates bitrates for individual tiles within the viewport. Consequently, they achieve higher average quality in the viewport regions and effectively mitigate penalties associated with temporal and spatial quality variations. Moreover, their adaptive bitrate allocation significantly reduces rebuffering penalties, resulting in substantial overall QoE enhancements.

SARL-based methods, though effective in optimizing the QoE of the current viewport, have limitations when it comes to considering the presence of other viewport regions. Consequently, they tend to incur higher penalties for temporal variation, spatial variation, and rebuffering. In contrast, the remarkable performance gains achieved by our proposed method can be largely attributed to its ability to efficiently mitigate rebuffering across diverse networks. By ensuring a sufficient buffer to handle fluctuations in network throughput, our method reduces rebuffering by 9.5% to 45.4%. It excels in balancing the optimization of the QoE metric, managing each factor effectively, rather than simply striving for superiority in individual QoE elements, which sets it apart from other ABR methods. For example, our proposed method achieves video quality that is approximately 1.3% lower than 360SRL while being 8.7% higher than Pensieve in terms of the temporal variation penalty. This exemplifies the method's capability to strike a harmonious balance among different factors, ultimately leading to the optimized overall QoE metric.

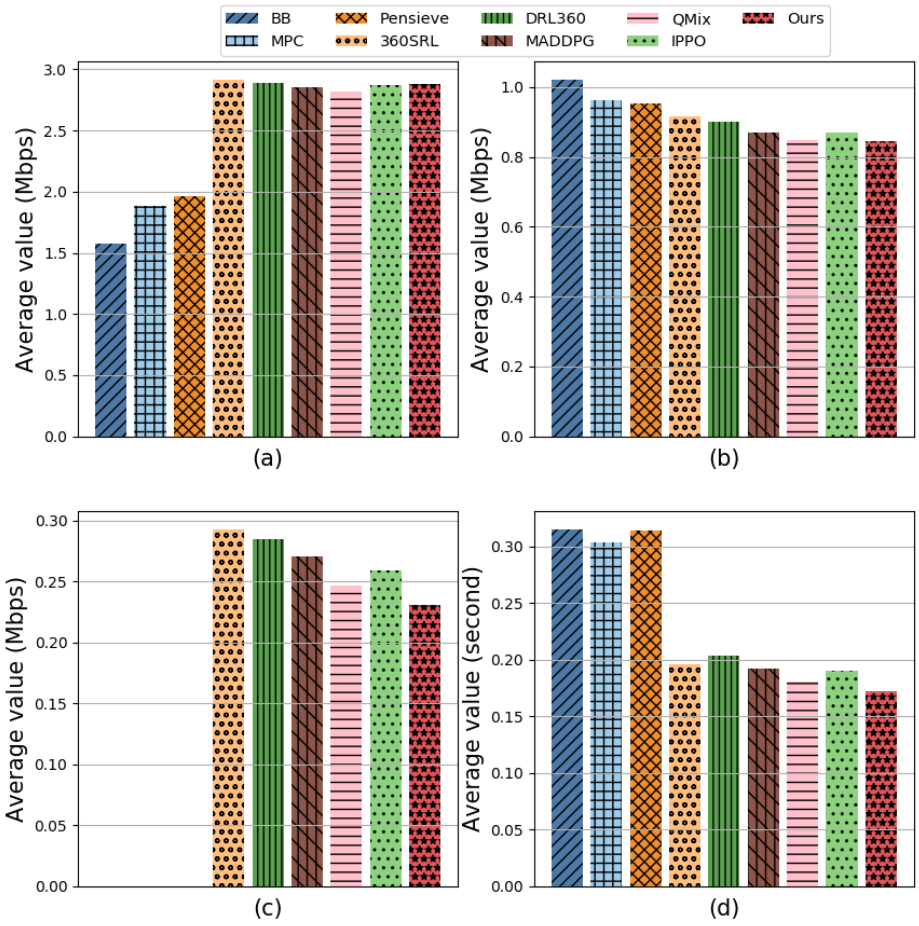

Figure 3.8: Evaluating our method against existing ABR algorithms based on the following criteria: (a) viewport quality, (b) viewport temporal variation, (c) viewport spatial variation, and (d) rebuffering. This analysis is conducted with the QoE objective (1,1,1,1).

3.4.3.4 Results Visualization

To thoroughly compare our method with other approaches, we conducted an extensive evaluation of QoE adaptability to bandwidth variations over time, as demonstrated in Fig. 3.9. We assessed the QoE scores under two distinct network conditions. In Fig. 3.9 (a), "trace 1" depicted a network condition with lower bit rates and minimal fluctuations, having an average of around 4.96 Mbps and a standard deviation (STD) of approximately 1.17 Mbps. Conversely, "trace 2" represented a network condition with higher bit rates and greater fluctuations, showcasing an average of about 6.91 Mbps and an STD of approximately 3.37 Mbps. Fig. 3.9 (b) displayed the QoE scores obtained for "trace 1". The BB method, primarily considering buffer occupancy and treating VR video as a traditional video to determine the bitrate, maintained relative stability over a specific period. However, both the MPC and Pensieve methods exhibited substantial fluctuations for both network traces. This arose from frequent bitrate adjustments required for all viewport regions, considering the probabilities associated with each region to adapt to varying network conditions. For "trace 1", our proposed method showcased an average QoE score improvement of about 0.74, ranging from 1.4% to 68.1%, compared to other methods. The STD of the QoE score was approximately 0.04, signifying a reduction of approximately 13.6% to 81%. Transitioning to the QoE scores on "trace 2" (Fig. 3.9 (c)), all methods exhibited higher QoE scores with greater fluctuation compared to the results of "trace 1" due to the characteristics of "trace 2". Our proposed method achieved a mean QoE score of approximately 1.18 on "trace 2", representing an improvement of about 4.4% to 93.4% over other methods. The STD of our proposed method was approximately 0.07, indicating a 23.6% increase. The fluctuation in QoE scores was primarily influenced by bandwidth fluctuations and viewpoint prediction probabilities. While our proposed method might not attain the best performance over time under different network conditions, it consistently delivered a high average QoE value, ensuring stable and satisfactory user experiences.

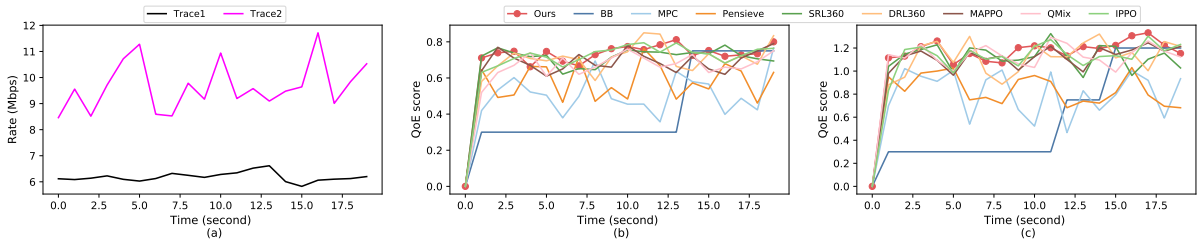

Figure 3.9: Visualization of the Quality of Experience (QoE) using the proposed method on two network traces with the QoE objective (1,1,1,1): (a) Network traces, (b) QoE on network trace 1, (c) QoE on network trace 2.

3.4.3.5 Impact of Network Structure

The performance of the proposed method in MARL is significantly influenced by the number of convolutional layers in the neural network architecture. To investigate the impact of network structure on the average quality of experience, we conducted experiments by modifying the number of convolutional layers in the CNN from 1 to 6, and the outcomes are summarized in Table 3.2. The evaluation was performed with different configurations, ranging from a single layer with 16 channels to six layers with channel configurations of (16, 32, 64, 128, 256, 512) for each layer. Notably, the proposed method with a single-layer structure demonstrated the lowest QoE score, as it faced challenges in effectively handling the observation information. However, with an increase in the number of CNN layers, the performance of the proposed method witnessed significant improvement. The results highlighted that augmenting the number of convolutional layers enhances the method's capability to capture complex patterns and extract meaningful features from the input data. Consequently, this improved feature extraction empowered the method with better decision-making abilities, resulting in higher average quality of experience for the users.

The performance of the proposed method was evaluated with various CNN configurations, revealing significant improvements in average QoE scores. Notably, the highest QoE score was achieved when the CNN had 5 layers, resulting in a substantial enhance-

Table 3.2: Analyzing the average QoE impact with configuration (1,1,1,1) concerning the variation in the number of CNN layers.

Number of Layer	Average QoE
1 (16)	2.29
2 (16,32)	2.47
3 (16,32,64)	2.56
4 (16,32,64,128)	2.63
5 (16,32,64,128,256)	2.83
6 (16,32,64,128,256,512)	2.76

ment of approximately 7.6% compared to other network structures. However, it was observed that the performance slightly decreased when the network structure consisted of 6 layers. As a result, a CNN configuration with 5 layers was selected for further experiments. The impact of the network structure on the proposed method's performance is remarkable. Specifically, the average QoE value with a 5-layer structure demonstrated a notable improvement of 20.5% compared to the 1-layer structure. This finding indicates that increasing the depth of the CNN architecture enables better representation and understanding of the underlying video data, leading to improved decision-making and ultimately resulting in higher QoE for users.

Furthermore, the running time of the proposed method for the bitrate decision process on requested video chunks was evaluated. Remarkably, the proposed method efficiently completed the bitrate determination for all viewport regions in just around 4 milliseconds. This impressive speed demonstrates its capability to meet real-time requirements for real-world applications, ensuring efficient and timely decision-making for optimal QoE delivery.

Chapter 4

TWRD Packet Scheduling

In this chapter, we present and evaluate our proposed tile-weighted rate-distortion packet scheduling on network nodes for VR video transmission. We first provide an overview of the proposed system in section 4.1. The tile ranking mechanism based on viewpoint prediction is introduced in section 4.2. We formulate the packet scheduling as an optimization problem in section 4.3 and show the dynamic programming solution in section 4.4, The evaluation for the proposed TWRD packet scheduling is shown in section 4.5.

4.1 Introduction

We propose a novel system called Tile-Weighted Rate-Distortion (TWRD), which is specifically designed to address efficient packet scheduling in scenarios with varying bandwidth constraints. The video quality distortion is used as QoE in the system. This system partitions the VR video into both temporal segments and spatial tiles, as depicted in Fig. 4.1. Moreover, each tile is assigned a weight based on the probability of viewpoint prediction, thus capturing its significance in the overall viewing experience. In the TWRD system, a packet's rate-distortion information comprises two key components: the rate, which indicates the packet size, and the distortion, representing the quality impact if the packet is lost. Notably, we introduce the concept of a weighted distortion, which accounts

for the importance of each packet based on the weight associated with its corresponding tile. Upon receiving a client's request for a video segment, the server initiates the streaming process of that specific segment over the network. To minimize distortion and ensure high-quality content delivery to the viewport, a network node leverages dynamic programming to determine an optimal packet scheduling scheme considering the available bandwidth. As per the scheduling scheme, packets are selectively dropped following a predetermined schedule. The TWRD system proves highly effective in maintaining a consistently high quality of the viewport during VR video viewing. By optimizing packet scheduling decisions, it successfully reduces distortion and substantially enhances the overall viewing experience for users.

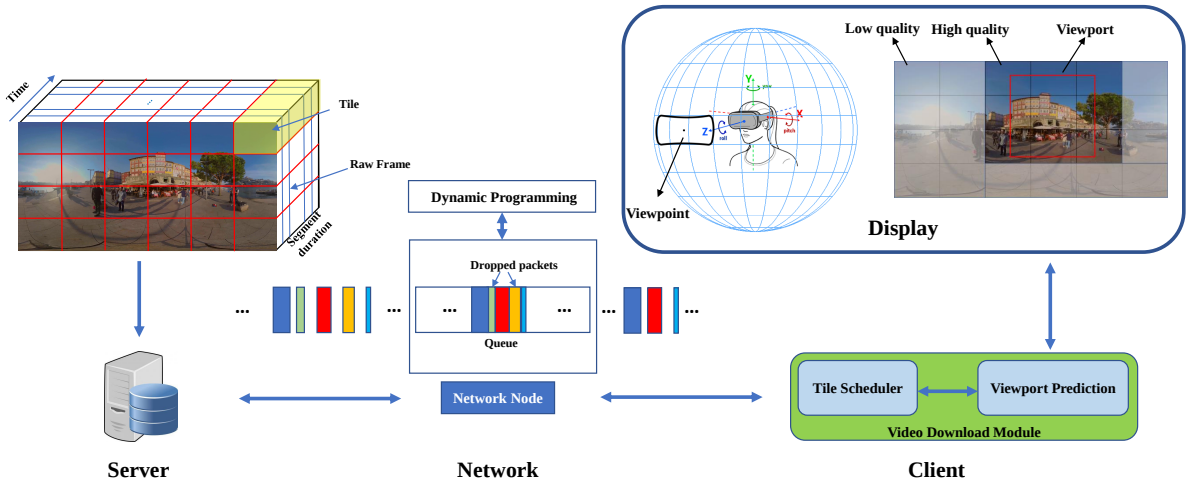

Figure 4.1: The tile-Weighted Rate-Distortion Packet Scheduling Optimization System designed for VR video streaming. The VR video is transmitted in the form of packets, where each packet is assigned a color denoting its importance and a width indicating its size. To address bandwidth constraints, the proposed method employs intelligent packet dropping.

4.2 Tile Ranking

In our study, we partition the VR video into 24 tiles, as illustrated in Fig. 4.1. These tiles are categorized into four distinct classes based on their "perceptual importance." Tiles that are entirely within the viewport receive the highest rank (class-4), while those appearing in more than half of the viewport are classified as class-3. Tiles with less than half of their area within the viewport are assigned to class-2, and tiles outside of the viewport are categorized as class-1. To determine the weight of each tile at time step t, we utilize the following formula:

$$\lambda_t = \frac{E \cdot L}{C} \tag{4.1}$$

Here, E represents the probability of prediction obtained from the transformer model, and C denotes the number of classes, set to 4 in our research. L corresponds to the class level, taking values of 1, 2, 3, or 4. As shown in Fig. 4.2 , tile 9 is fully inside the viewport, so it is class 4. Similary, tile 15, 2 and 1 are class 3, 2, 1, respectively.

Figure 4.2: Tiles are classified into differnt classes.

By calculating the weight difference between two different classes, l_1 and l_2, as $\frac{E \cdot \alpha}{C}|l_1 - l_2|$, we can adaptively adjust the significance of tiles based on their predicted visibility. When the viewport has a low probability of being viewed, the weight difference between tiles inside and outside the viewport decreases. This dynamic ranking approach allows the system to selectively drop more packets within the viewport when their visibility

probability is low, and vice versa. As a result, the content delivery is optimized to cater to the viewer's viewport, enhancing the overall quality of experience.

4.3 Rate-Distortion Optimization

4.3.1 Distortion Definition of Packet Loss

A video that undergoes compression using a standard algorithm, like H.264, consists of three frame types: I-frame, P-frame, and B-frame. Whenever a frame is dropped, it causes error propagation due to interframe dependencies. Let's take a GOP (Group of Pictures) illustrated in Figure 4.3, comprising frames f_0 to f_8, as an example. If the initial I-frame (f_0) is omitted, the error extends throughout the entire GOP until the next I-frame (f_9) appears in the subsequent GOP. An I-frame's absence causes error propagation, impacting all frames in the GOP until it is corrected by the next I-frame. Likewise, dropping the P-frame (f_3) leads to incorrect decoding of the subsequent P-frame (f_6) and B-frame (f_2). The error arising from f_3 propagates further to all subsequent frames within the GOP. However, as B-frames are not referenced by other frames, the omission of a B-frame (f_1) doesn't induce error propagation within the GOP. Despite error propagation being limited to a few frames within a GOP, we treat it as if it affects all frames in a video, even though the majority of frames have an error value of 0.

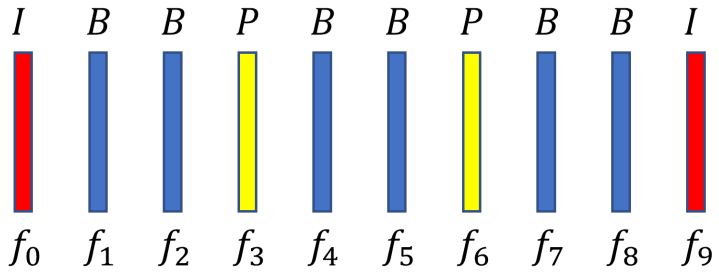

Figure 4.3: In the given sequence of pictures: f_0 and f_9 are I-frames (intra-coded frames); f_3 and f_6 are P-frames (predictive-coded frames); and the rest are B-frames (bi-directional predictive-coded frames).

Let us consider a VR video comprising t tiles denoted as $\boldsymbol{V} = V^0, V^1, ..., V^{t-1}$, where each tile contains an equal number of frames denoted as n. Thus, the frame sequence of tile τ, where τ ranges from 0 to $t-1$, can be represented as $V^\tau = p_0^\tau, p_1^\tau, ..., p_{n-1}^\tau$. Introducing the index η, where η varies from 0 to $n-1$, we define frame η of tile τ as p_η^τ. In our research, we adopt the perspective of considering a tile-frame (a frame of a tile) as a transmission packet. We assume that each tile-frame is encapsulated within a packet for transmission. Although a frame contains several packets in real world, the proposed algorithm can also be applied in practice. Consequently, if a packet is lost during the transmission process, the corresponding tile-frame is also lost. Here, it is essential to clarify that in this context, the term "frame" specifically refers to a frame belonging to a tile rather than a panoramic frame. To avoid confusion, we emphasize that in our study, the terms "packet" and "frame" are used interchangeably, as they denote the same unit of data transmission. To evaluate the impact of packet loss, we employ the structural similarity index measure (SSIM), which quantifies the similarity between two images. The SSIM values range from 0 to 1, where a value of 0 indicates no similarity between the two images, and a value of 1 indicates complete similarity. By calculating the dissimilarity between two packets, denoted as p_i and p_j, using the formula $d(p_i, p_j) = 1 - SSIM$, we can quantify the distortion caused by packet loss. This distortion metric allows us to assess the deviation between the original and received packets based on their image content.

For each tile τ, the frame sequence $V^\tau = p^\tau 0, p^\tau 1, ..., p^\tau \eta, ..., p^\tau n - 1$ encompasses the frames associated with that specific tile. In order to facilitate transmission over the network channel, we carefully choose a transmission subset $S^\tau = p^\tau s_0, p^\tau s_1, ..., p^\tau s_{l-1}$ from the frame sequence V^τ, where l represents the length of the subset. The resulting video sequence of tile V^τ, based on the transmitted subset S^τ, is denoted by the reconstructed video sequence $C_{V^\tau}(S^\tau) = p_0^{\tau\prime}, p_1^{\tau\prime}, ..., {p_{n-1}^\tau}'$. In the event of frame loss during transmission, we employ the previous frame concealment technique to compensate for the missing frames. In the reconstructed video sequence $CV^\tau(S^\tau)$, any dropped frame

is substituted with the nearest previous neighboring frame found within the transmitted subset S^τ, ensuring that the missing frame is reconstructed using the closest available reference frame.

$$p_i^{\tau\prime} = p^\tau_{j=max(s),s\in\{s_0,s_1,\ldots,s_{l-1}\},j\leq i} \tag{4.2}$$

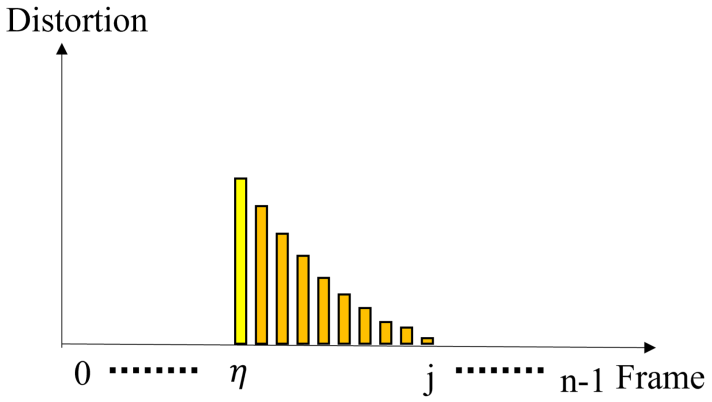

Figure 4.4: Loss-induced distortion in tile τ due to the absence of packet η. The yellow color illustrates distortion without error propagation, while the orange color indicates distortion resulting from error propagation.

When a packet p^τ_η is lost, the resulting distortion $D(p^\tau_\eta)$ reflects the combined impact on all affected frames, accounting for error propagation. As depicted in Figure 4.4, subsequent frames following p^τ_η exhibit non-zero distortion values due to factors like spatial filtering and intra-refreshing techniques, causing the impact to gradually diminish across successive frames until it disappears at a distant frame [93]. Importantly, in our analysis, we assume that distortions arising from multiple packet losses are independent and additive [94].

To efficiently manage computational complexity, a standard approach is to use an approximation that decomposes the total distortion into smaller packet distortion functions and optimizes them individually [95]. By doing so, the overall distortion $D(p^\tau_\eta)$ can be calculated by summing up the distortions between each frame and its corresponding reconstructed frame. This reconstructed frame is obtained through decoding or error

concealment, aiming to compensate for the lost or corrupted packet.

$$\begin{aligned} D(p_\eta^\tau) &= \sum_{i=0}^{n-1} d(p_i^\tau, p_i^{\tau\prime}) \\ &= d(p_\eta^\tau, p^\tau_{j=max(s),s\in\{s_0,s_1,...,s_{l-1}\},j\leq\eta}) + \sum_{i=0,i\neq\eta}^{n-1} d(p_i^\tau, p_i^{\tau\prime}) \end{aligned} \tag{4.3}$$

Dividing the distortion of p_η^τ into two components, the first component reflects the distortion on p_η^τ itself, independent of error propagation, as shown in the yellow section of Figure 4.4. To calculate this component, we compute the difference between p_η^τ and its nearest previous neighboring frame. The resulting reconstructed sequence based on this first component is denoted as $C_{1V^\tau}(S^\tau) = p_0^{\tau\prime}, p_1^{\tau\prime}, ..., p_\eta^{\tau\prime}, ..., p_{n-1}^{\tau}{}' = p_0^\tau, p_1^\tau, ..., p_{\eta-1}^\tau, ..., p_{n-1}^\tau$. In this reconstructed sequence, every frame remains the same as in the original sequence except for p_η^τ, which is replaced by $p_{\eta-1}^\tau$. This first component, without considering error propagation, can be expressed as

$$\begin{aligned} \sum_{i=0}^{n-1} d(p_i^\tau, p_i^{\tau\prime}) &= \sum_{i=0}^{n-1} d(p_i^\tau, p^\tau_{j=max(s),s\in\{s_0,s_1,...,s_{l-1}\},j\leq i}) \\ &= d(p_\eta^\tau, p_{\eta-1}^\tau) \end{aligned} \tag{4.4}$$

The second component accounts for the accumulated distortion on all frames, excluding p_η^τ, resulting from error propagation. This cumulative distortion is computed by comparing each original frame with its corresponding distorted counterpart. We denote this second component as $\Omega(p_\eta^\tau)$. It is essential to note that for frames with indices $i < \eta$, the distortion $d(p_i^\tau, p_i^{\tau\prime})$ is zero, as no previous distortion occurs before the loss of packet p_η^τ. Hence, the equation for calculating the total distortion $D(p_\eta^\tau)$ can be modified as

$$D(p_\eta^\tau) = \sum_{i=0}^{n-1} d(p_i^\tau, p^\tau_{j=max(s),s\in\{s_0,s_1,...,s_{l-1}\},j\leq i}) + \Omega(p_\eta^\tau) \tag{4.5}$$

With the tile frame sequence V^τ and the selected transmission subset S^τ at hand, we can simultaneously identify a dropped set $K^\tau = p_{k_0}, p_{k_1}, ..., p_{k_{z-1}}$, where z denotes the length of the dropped set. To quantify the overall distortion of the dropped set K^τ, we

express it using the following equation:

$$\begin{aligned} D(K^\tau) &= \sum_{\eta \in K^\tau} D(p_\eta^\tau) \\ &= \sum_{\eta \in K^\tau} \sum_{i=0}^{n-1} d(p_i^\tau, p_i^{\tau\prime}) \\ &= \sum_{\eta \in K^\tau} d(p_\eta^\tau, p^\tau_{j=max(s), s \in \{s_0, s_1, ..., s_{l-1}\}, j \leq \eta}) + \sum_{\eta \in K^\tau} \Omega(p_\eta^\tau) \\ &= \sum_{i=0}^{n-1} d(p_i^\tau, p^\tau_{j=max(s), s \in \{s_0, s_1, ..., s_{l-1}\}, j \leq i}) + \Omega(K_\eta^\tau) \end{aligned} \tag{4.6}$$

When confronted with multiple packet losses, we can partition the resulting distortion into two distinct components. Without considering error propagation, the first component can be computed as the sum of distortions between each original frame $p^\tau i$ and its corresponding reconstructed frame $p^\tau j = max(s), s \in s_0, s_1, ..., s_{l-1}, j \leq i$, yielding $\sum_{i=0}^{n-1} d(p^\tau i, p^\tau j = max(s), s \in s_0, s_1, ..., s_{l-1}, j \leq i)$. This component calculates the distortion as if it were caused by individual packet losses, independently assessing the impact of each dropped packet. On the other hand, the second component, denoted as $\Omega(K^\tau)$, accounts for the cumulative distortion propagated by the loss of multiple packets in the set K^τ. It captures the collective effect of error propagation across all dropped packets. Consequently, the overall distortion caused by multiple packet losses is determined by combining these two components, considering both the individual frame-level distortions and the distortion propagated due to multiple packet losses.

Let's consider an example with tile τ, which consists of 5 frames: $V^\tau = p_0^\tau, p_1^\tau, p_2^\tau, p_3^\tau, p_4^\tau$. If the dropped set $K^\tau = p_1^\tau, p_3^\tau$ occurs, it results in a transmission set $S^\tau = p_0^\tau, p_2^\tau, p_4^\tau$. The reconstructed video sequence based on the transmission set is denoted as $C_{V^\tau}(S^\tau) = p_0^{\tau\prime}, p_1^{\tau\prime}, p_2^{\tau\prime}, p_3^{\tau\prime}, p_4^{\tau\prime}$. It is essential to emphasize that, in this particular scenario, since we are solely considering the reconstruction without error propagation, the resulting video sequence based on the first part becomes a repetition of frames from the transmission set: $C1_{V^\tau}(S^\tau) = p_0^\tau, p_0^\tau, p_2^\tau, p_2^\tau, p_4^\tau$. In this case, the dropped frames are replaced with their nearest previous neighboring frame from the transmission set. To accurately assess the distortion caused by the first part, we analyze the dissimilarities between the original

frames and their corresponding reconstructed frames.

$$\begin{aligned}
&\sum_{i=0}^{n-1} d(p_i^\tau, p^\tau_{j=max(s),s\in\{0,2,4\},j\leq i}) \\
&= d(p_0^\tau, p_0^\tau) + d(p_1^\tau, p_0^\tau) + d(p_2^\tau, p_2^\tau) + d(p_3^\tau, p_2^\tau) + d(p_4^\tau, p_4^\tau) \\
&= d(p_1^\tau, p_0^\tau) + d(p_3^\tau, p_2^\tau)
\end{aligned} \tag{4.7}$$

In this given instance, the second part can be calculated by summing the distortions for the dropped packets p_1^τ and p_3^τ, denoted as $\Omega(K^\tau) = \Omega(p_1^\tau) + \Omega(p_3^\tau)$. To evaluate the total weighted distortion across all tiles affected by the set $\boldsymbol{K} = K^0, K^1, ..., K^{t-1}$, we must take into account the distortions from both the first and second parts of the dropped packets. The total weighted distortion can be expressed as:

$$\begin{aligned}
&\widetilde{D}(\boldsymbol{K}) = \sum_{\tau=0}^{t-1} \lambda_\tau D(K^\tau) = \sum_{\tau=0}^{t-1} \sum_{\eta\in K^\tau} \lambda_\tau D(p_\eta^\tau) \\
&= \sum_{\tau=0}^{t-1} \sum_{\eta\in K^\tau} \Big\{ \sum_{i=0}^{n-1} \lambda_\tau d\big(p_i^\tau, p^\tau_{j=max(s),s\in\{s_0,s_1,...,s_{l-1}\},j\leq i}\big) \\
&+ \lambda_\tau \Omega(p_\eta^\tau) \Big\} \\
&= \sum_{\tau=0}^{t-1} \sum_{\eta\in K^\tau} \Big\{ \sum_{i=0}^{n-1} \widetilde{d}\big(p_i^\tau, p^\tau_{j=max(s),s\in\{s_0,s_1,...,s_{l-1}\},j\leq i}\big) \\
&+ \widetilde{\Omega}(p_\eta^\tau) \Big\} \\
&= \sum_{\tau=0}^{t-1} \sum_{\eta\in K^\tau} \widetilde{D}(P_\eta^\tau)
\end{aligned} \tag{4.8}$$

In this given equation, the variable λ_τ holds the significance factor for each tile τ. To determine the total weighted distortion, we individually scale the distortion of each packet by its corresponding weight. Hence, we denote the weighted distortion of a packet P_η^τ as $\widetilde{D}(P_\eta^\tau)$, where $\widetilde{D}(P_\eta^\tau) = \lambda_\tau D(P_\eta^\tau)$. Through the assignment of importance factors to each tile, we effectively adjust the impact of packet losses in different parts of the video. By considering $\boldsymbol{K} = K^0, K^1, ..., K^{t-1}$, the set of dropped packets over all tiles, we can calculate the total weighted distortion by summing up the weighted distortions of all packets within each tile.

Let's delve into a VR video sequence comprising n frames and t tiles. This sequence can be conveniently represented as a packet set $P = p_0^0, p_1^0, ..., p_{n-1}^0, ..., p^{t-1}0, p^{t-1}1, ..., p^{t-1}n-1$, containing a total of $\omega = n \cdot t$ packets. For simplicity, we'll rewrite the packet set P as $P = p_0, p_1, ..., p\omega - 1$. Suppose there are f dropped packets, constituting the dropped set $K = p_{k_0}, p_{k_1}, ..., p_{k_{f-1}}$. Conversely, the transmission set S, with $g = \omega - f$ packets, can be defined as $S = p_{s_0}, p_{s_1}, ..., p_{s_{g-1}}$, where $p_{k_0}, p_{k_1}, ..., p_{k_{f-1}}$ represent the dropped packets, and $p_{s_0}, p_{s_1}, ..., p_{s_{g-1}}$ represent the successfully transmitted packets. With Equation 4.6 as a reference, we can reframe Equation 4.8 as follows:

$$\begin{aligned} \widetilde{D}(K) &= \sum_{i=0}^{f-1} \widetilde{D}(p_{k_i}) \\ &= \sum_{i=0}^{w-1} \widetilde{d}\big(p_i, p_{j=max(s),s\in\{s_0,s_1,...,s_{g-1}\},j\leq i}\big) + \widetilde{\Omega}(K) \end{aligned} \tag{4.9}$$

4.3.2 Distortion Minimization Problem Formulation

The task at hand involves optimizing the trade-off between bandwidth utilization and distortion minimization for a VR video transmission system. We are given the available network bandwidth, $R_{\max}$, and a packet sequence representing the VR video, $P = p_0, p_1, ..., p_{\omega-1}$. The primary goal is to identify a dropped subset, $K = p_{k_0}, p_{k_1}, ..., p_{k_{f-1}}$, from the packet stream P, adhering to the bandwidth constraint while minimizing the weighted distortion. Simultaneously, we construct the transmission subset, $S = p_{s_0}, p_{s_1}, ..., p_{s_{g-1}}$, containing the remaining packets to be transmitted over the network channel, with $P = K \cup S$. By finding the optimal sets K and S, we can achieve an optimal balance between utilizing bandwidth efficiently and minimizing distortion in the transmitted video.

$$K^* = \arg\min_K \widetilde{D}(K),\ s.t.\ R(S) \leq R_{max} \tag{4.10}$$

The optimization objective, as shown in Equation 4.10, is to find the set K that minimizes the weighted distortion $\widetilde{D}(K)$ while adhering to the constraint $R(S) \leq R_{max}$. This involves making a decision on which packets to include in the dropped set K and

which packets to transmit in the transmission set S to achieve an optimal trade-off between bandwidth utilization and distortion minimization in the VR video stream. We can represent K as the set difference between P and S, denoted as $K = P \backslash S$, which includes the packets present in P but not in S. Accordingly, the problem can be restated as follows:

$$\begin{aligned} S^* &= \arg\min_S \widetilde{D}(P \backslash S), s.t. R(S) \leq R_{max} \\ &= \arg\min_S \Big\{ \sum_{i=0}^{\omega-1} \widetilde{d}\big(p_i, p_{j=max(s), s\in\{s_0,s_1,\ldots,s_{g-1}\}, j\leq i}\big) \\ &+ \widetilde{\Omega}(P \backslash S) \Big\} \end{aligned} \tag{4.11}$$

4.4 Dynamic Programming Solution

To overcome the computational challenges posed by the exponential number of possibilities, we present an efficient dynamic programming-based solution to address the problem stated in Eq. 4.11. In our approach, redundant calculations are avoided and computational complexity is significantly reduced by breaking the problem into smaller subproblems and reusing computed results. With the function $\widetilde{D}(P \backslash S^{s_0:s_{m-1}} m)$ to represent the minimum weighted distortion for a selected transmission subset $S_m^{s_0:s_{m-1}}$, the dynamic programming equation introduced earlier can be used to recursively compute the function. The recursive strategy effectively explores different combinations without exhausting all possible combinations.

Dynamic programming provides an efficient means of determining the optimal dropped set K or transmission set S that minimizes weighted distortion while adhering to bandwidth constraints. The algorithm strikes an effective balance between computational efficiency and finding the best solution. By leveraging the benefits of dynamic programming, we can effectively handle larger video streams with a considerable number of packets while delivering accurate results in a reasonable amount of time.

The term $\widetilde{D}(P\backslash S^{s_0:s_{m-1}}m)$ represents the minimum weighted distortion caused by the selected transmission subset $S^{s_0:s_{m-1}}m = p_{s_0}, p_{s_1}, ..., p_{s_{m-1}}$, comprising m packets starting with p_{s_0} and ending with $p_{s_{m-1}}$. With respect to Equation 4.11, we can represent this as follows:

$$\widetilde{D}(P\backslash S_m^{s_0:s_{m-1}}) = \min_{S_m^{s_0:s_{m-1}}} \Big\{ \sum_{i=0}^{\omega-1} \widetilde{d}\big(p_i, p_{j=max(s),s\in S_m^{s_0:s_{m-1}},j\leq i}\big) + \widetilde{\Omega}(P\backslash S_m^{s_0:s_{m-1}}) \Big\} \tag{4.12}$$

Given that the initial packet p_0 (I frame) is always part of the transmission set S, we can deduce that $s_0 = 0$. Moreover, we define e to represent the index s_{m-1}. By excluding the first and last elements from the optimization, we can represent Equation 4.12 as follows:

$$\widetilde{D}(P\backslash S_m^{s_0:s_{m-1}}) = \min_{S_m^{s_1:s_{m-2}}} \Big\{ \sum_{i=0}^{\omega-1} \widetilde{d}\big(p_i, p_{j=max(s),s\in S_m^{0:s_{m-1}},j\leq i}\big) + \widetilde{\Omega}(P\backslash S_m^{s_0:s_{m-1}}) \Big\} \tag{4.13}$$

Let $\widetilde{D}(P\backslash S^{s_0:s_{m-2}}m-1)$ represent the minimum weighted distortion caused by the selected transmission subset $S^{s_0:sm-2}m-1 = p_{s_0}, p_{s_1}, ..., p_{s_{m-2}}$, which consists of $m-1$ packets, starting with packet p_{s_0} and ending with packet $p_{s_{m-2}}$. To calculate $\widetilde{D}(P\backslash S_{m-1}^{s_0:s_{m-2}})$, we can express it as:

$$\widetilde{D}(P\backslash S_{m-1}^{s_0:s_{m-2}}) = \min_{S_{m-1}^{s_1:s_{m-3}}} \Big\{ \sum_{i=0}^{w-1} \widetilde{d}\big(p_i, p_{j=max(s),s\in S_{m-1}^{0:s_{m-2}},j\leq i}\big) + \widetilde{\Omega}(P\backslash S_{m-1}^{s_0:s_{m-2}}) \Big\} \tag{4.14}$$

As $0 < s_1 < s_2 < ... < s_{m-2} < e$ and $j \leq i$, the Eq. 4.13 can be expressed as

$$\widetilde{D}(P\backslash S_m^{s_0:s_{m-1}}) = \min_{S_m^{s_1:s_{m-2}}} \Big\{ \sum_{i=0}^{e-1} \widetilde{d}\big(p_i, p_{j=max(s),s\in S_{m-1}^{0:s_{m-2}},j\leq i}\big) + \sum_{i=e}^{\omega-1} \widetilde{d}(p_i, p_e) + \widetilde{\Omega}(P\backslash S_m^{s_0:s_{m-1}}) \Big\} \tag{4.15}$$

The summation $\sum_{i=e}^{\omega-1} \widetilde{d}(p_i, p_e)$ calculates the distortion without considering error propagation after the packet e, with a comparison against the subsequent packets. Thus, we can express Eq. 4.15 more concisely as:

$$\begin{aligned}
\widetilde{D}(P\backslash S_m^{s_0:s_{m-1}}) = \min_{S_m^{s_1:s_{m-2}}} \Big\{ & \sum_{i=0}^{e-1} \widetilde{d}\big(p_i, p_{j=max(s),s\in S_{m-1}^{0:s_{m-2}},j\le i}\big) \\
& + \sum_{i=e}^{\omega-1} \widetilde{d}\big(p_i, p_{j=max(s),s\in S_{m-1}^{0:s_{m-2}},j\le i}\big) \\
& - \sum_{i=e}^{\omega-1} \widetilde{d}\big(p_i, p_{j=max(s),s\in S_{m-1}^{0:s_{m-2}},j\le i}\big) \\
& + \widetilde{\Omega}(P\backslash S_m^{s_0:s_{m-1}}) \Big\} \\
& + \sum_{i=e}^{\omega-1} \widetilde{d}(p_i, p_e)
\end{aligned} \tag{4.16}$$

Due to $s_{m-2} < e$, we have

$$\sum_{i=e}^{\omega-1} \widetilde{d}\big(p_i, p_{j=max(s),s\in S_{m-1}^{0:s_{m-2}},j\le i}\big) = \sum_{i=e}^{\omega-1} \widetilde{d}\big(p_i, p_{s_{m-2}}\big) \tag{4.17}$$

Hence, we can rewrite Eq. 4.16 as

$$\begin{aligned}
\widetilde{D}(P\backslash S_m^{s_0:s_{m-1}}) = \min_{S_m^{s_1:s_{m-2}}} \Big\{ & \sum_{i=0}^{e-1} \widetilde{d}\big(p_i, p_{j=max(s),s\in S_{m-1}^{0:s_{m-2}},j\le i}\big) \\
& + \sum_{i=e}^{\omega-1} \widetilde{d}\big(p_i, p_{j=max(s),s\in S_{m-1}^{0:s_{m-2}},j\le i}\big) \\
& - \sum_{i=e}^{\omega-1} \widetilde{d}(p_i, p_{s_{m-2}}) + \Omega(P\backslash S_m^{s_0:s_{m-1}}) \Big\} \\
& + \sum_{i=e}^{\omega-1} \widetilde{d}(p_i, p_e) \\
= \min_{S_m^{s_1:s_{m-2}}} \Big\{ & \sum_{i=0}^{\omega-1} \widetilde{d}\big(p_i, p_{j=max(s),s\in S_{m-1}^{0:s_{m-2}},j\le i}\big) \\
& + \widetilde{\Omega}(P\backslash S_m^{s_0:s_{m-1}}) \\
& - \sum_{i=e}^{\omega-1} \big(\widetilde{d}(p_i, p_{s_{m-2}}) - \widetilde{d}(p_i, p_e)\big) \Big\}
\end{aligned} \tag{4.18}$$

Since $\widetilde{\Omega}(P\backslash S_m^{s_0:s_{m-1}}) = \widetilde{\Omega}(P\backslash S_{m-1}^{s_0:s_{m-2}}) - \widetilde{\Omega}(e)$, Eq. 4.18 can be expressed as

$$\begin{aligned}\widetilde{D}(P\backslash S_m^{s_0:s_{m-1}}) = \min_{S_m^{s_1:s_{m-2}}} \Big\{ & \sum_{i=0}^{n-1} \widetilde{D}\big(p_i, p_{j=max(s),s\in S_{m-1}^{0:s_{m-2}},j\le i}\big) \\ & + \widetilde{\Omega}(P\backslash S_{m-1}^{s_0:s_{m-2}}) - \widetilde{\Omega}(e) \\ & - \sum_{i=e}^{\omega-1} \big(\widetilde{d}(p_i, p_{s_{m-2}}) - \widetilde{d}(p_i, p_e)\big) \Big\}\end{aligned} \tag{4.19}$$

Let

$$\Phi(s_{m-2}, e) = \widetilde{\Omega}(e) + \sum_{i=e}^{\omega-1} \big(\widetilde{d}(p_i, p_{s_{m-2}}) - \widetilde{d}(p_i, p_e)\big) \tag{4.20}$$

Eq. 4.19 can be rewritten as

$$\begin{aligned}\widetilde{D}(P\backslash S_m^{s_0:s_{m-1}}) = \min_{S_m^{s_1:s_{m-2}}} \Big\{ & \sum_{i=0}^{n-1} \widetilde{d}\big(p_i, p_{j=max(s),s\in S_{m-1}^{0:s_{m-2}},j\le i}\big) \\ & + \widetilde{\Omega}(P\backslash S_{m-1}^{s_0:s_{m-2}}) - \Phi(s_{m-2}, e) \Big\}\end{aligned} \tag{4.21}$$

By referring to Eq. 4.14, the Eq. 4.21 can be rewritten as

$$\begin{aligned}&\widetilde{D}(P\backslash S_m^{s_0:s_{m-1}}) = \\ &\min_{s_{m-2}} \Big\{ \min_{S_{m-1}^{s_1:s_{m-3}}} \Big\{ \sum_{i=0}^{\omega-1} \widetilde{d}\big(p_i, p_{j=max(s),s\in S_{m-1}^{0:s_{m-2}},j\le i}\big) \Big\} \\ &+ \widetilde{\Omega}(P\backslash S_{m-1}^{s_0:s_{m-2}}) - \Phi(s_{m-2}, e) \Big\} \\ &= \min_{s_{m-2}} \Big\{ \widetilde{D}(P\backslash S_{m-1}^{s_0:s_{m-2}}) - \Phi(s_{m-2}, e) \Big\}\end{aligned} \tag{4.22}$$

Within the given equation, the initial part corresponds to the minimum distortion for the set $S^{s_0:s_{m-2}}m-1$, while the second part represents the reduction in distortion when packet e is incorporated into $S^{s_0:sm-2}m-1$ to form the new set $S^{s_0:sm-1}m$. Equation 4.22 illustrates how the total distortion of $S^{s_0:sm-1}m$ with a length of m can be obtained by considering the previous total distortion of the set $S^{s_0:sm-2}m-1$ with a length of $m-1$, and taking into account the addition of a new packet to $S^{s_0:sm-2}m-1$. This equation succinctly captures the state transition process in dynamic programming. Since the first frame is always included in S, we can define the initial state $S^0 1$ as:

$$D(P\backslash S_1^0) = \sum_{e=1}^{m-1} d(p_0, p_e) + \Omega(P\backslash S_1^0) \tag{4.23}$$

By utilizing the initial state S_1^0 and the state transition equation presented in Equation 4.22, the system can efficiently compute the optimal packet scheduling scheme through a backtracking approach. This process involves exploring all possible combinations of packets while evaluating their weighted distortions and ensuring adherence to the bandwidth constraint. The backtracking algorithm allows the system to incrementally build the transmission subset S by considering different combinations of packets and selecting the one that minimizes the total weighted distortion. This dynamic programming-based approach significantly reduces the computational complexity compared to brute force search, making it a practical and effective solution for the packet scheduling problem in VR video transmission.

4.5 Evaluation of Packet Scheduling Strategy

4.5.1 System Setup

To assess the functionality and performance of our system, we investigate a common network scenario where the incoming link's data rates exceed those of the outgoing link at a specific network node. We set up an experimental environment, as illustrated in Figure 3.4, consisting of a VR video server, a client, and a simulated network environment based on NS3 [82]. The VR video content is structured into one-second segments and further partitioned into 24 distinct tiles to optimize processing and transmission efficiency.

Upon receiving a video request, the server swiftly retrieves the VR video content from storage and initiates the seamless streaming process to the client. Each packet's distortion is carefully weighed, taking into account the probability of viewpoint prediction, and these weighted rate-distortion values are seamlessly integrated into the packet header before being efficiently transmitted across the network. To introduce a bottleneck scenario, a well-calibrated constrained link is artfully established within the NS3 simulated network. Leveraging the rich rate-distortion information, the ingeniously designed proposed method is meticulously put into action within the system, artfully determining

the most optimal packet scheduling scheme for packets stemming from each and every segment of the VR video. The pivotal network node, elegantly positioned on the left side of the illustrious NS3 simulator, as prominently depicted in the captivating Figure 3.4, rises to the occasion, deftly assuming the pivotal role of making intelligent, informed, and thoughtful decisions regarding the packet forwarding and dropping, all with the singular mission of effectively and elegantly minimizing weighted distortion while gracefully navigating the confines of the limited bandwidth available on the outgoing link.

4.5.2 Implementation Details

Apart from the proposed TWRD approach, we incorporated two other well-known methods in our experimentation. The baseline method, known as tail drop, handles packet dropping during network congestion without considering individual packet differences. It relies on random dropping to adhere to the bandwidth limitation. We also tested the rate-distortion packet scheduling method for traditional video streaming [96], referred to as EWRD (equal-weighted rate-distortion). In EWRD, all tiles are treated equally, receiving equal weight with $\lambda_\tau = 1$. The system schedules packets based on real distortion, rather than weighted distortion. To comprehensively investigate and analyze the three methods, TWRD, EWRD, and baseline, we evaluated their performance across three critical aspects: distortion, packet loss rate, and bandwidth consumption. Five metrics, namely overall distortion, viewport distortion, overall packet loss rate, viewport packet loss rate, and viewport bandwidth consumption, were used to measure their effectiveness. The labels "*_total" and "*_viewport" indicate the results for each method in the entire VR video and the viewport, respectively. For instance, TWRD_viewport displays the proposed TWRD method's performance within the viewport, while EWRD_total reflects the results of the EWRD method across the entire VR video. Initially, we focused on visualizing and analyzing the experimental results of the VR video "PortoRiverside." Additionally, in the cross-validation section, we presented results from the "PortoRiverside" and "Warship" VR videos to assess the proposed system's robustness. Both videos

boast 4K resolution, last for 20 seconds, and are encoded using the H.264 compression standard with a GOP structure of 25 frames following the format “IBBPBB...".

4.5.3 Total Distortion Analysis

Figure 4.5 displays the cumulative distortions observed across all tiles of the VR video "PortoRiverside" when employing three distinct packet scheduling strategies under varying bandwidth conditions, denoted in Mbps. Notably, the baseline strategy consistently exhibits significantly higher distortions compared to the other two strategies, regardless of the bandwidth range. The overall distortion levels between the TWRD and EWRD strategies show minimal disparity, although the EWRD strategy showcases superior performance. This distinction arises from EWRD's ability to intelligently drop packets with the least impact on the reconstructed video quality, leveraging its knowledge of dropped packet effects. TWRD, however, relies solely on weighted distortion, which can result in slightly higher distortion. In all three methods, the distortions steadily reduce until zero as bandwidth increases up to 30 Mbps, indicating the absence of packet drops as bandwidth increases. Thus, a minimum bandwidth of 30 Mbps is sufficient for flawless delivery of "PortoRiverside". VR video streaming experiences are highly dependent on adequate bandwidth availability, as this finding illustrates.

4.5.4 Viewport Distortion Analysis

The viewport, which represents the area the user focuses on, plays a crucial role in determining the quality of the overall experience. Figure 4.6 demonstrates that the TWRD strategy consistently performs the best across the bandwidth range, while the EWRD strategy also outperforms the baseline. The improvement in viewport distortion provided by TWRD compared to EWRD remains relatively stable, ranging from 2 to 4 times, throughout different bandwidths due to intentional packet dropping in EWRD. However, when comparing TWRD and the baseline at a bandwidth of 10 Mbps, the improvement in viewport distortion is approximately 11 times greater, surpassing the

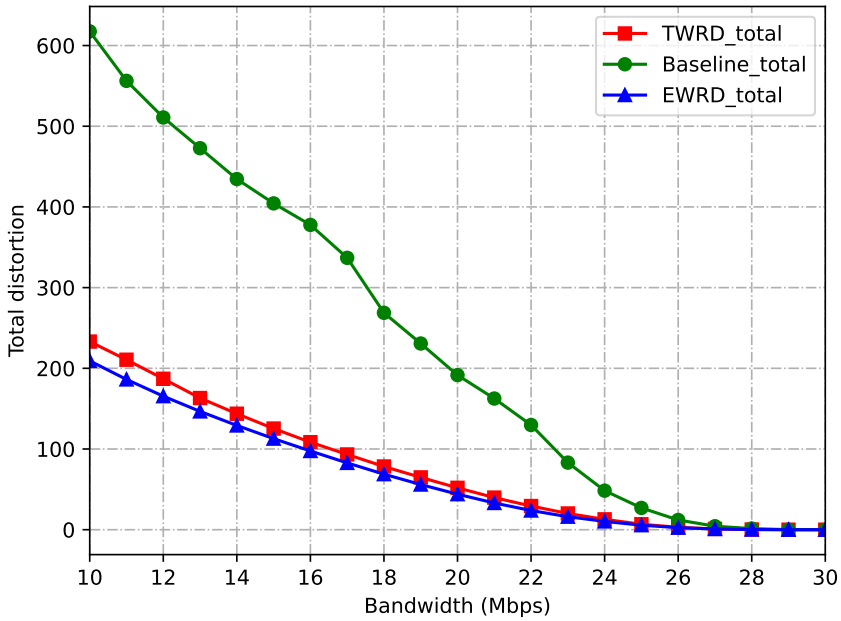

Figure 4.5: Comparing total distortions of all tiles in VR video "PortoRiverside" among three methods at different bandwidths.

overall distortion improvement of about 3 times. This improvement becomes even more pronounced, reaching up to 29 times at a bandwidth of 17 Mbps. This discrepancy occurs because when the bandwidth is extremely limited, packets within the viewport are also dropped, leading to no improvement in either TWRD or the baseline. However, when there is sufficient bandwidth to transmit all packets, no packets within the viewport are dropped, resulting in no distinction between the two methods. As a result, the performance gain offered by the proposed method increases with higher bandwidth, peaks at a certain point, and then gradually diminishes with further increases in bandwidth. This behavior is illustrated in Figure 4.6, where the gain of the proposed method over the baseline is approximately 11 times at a bandwidth of 10 Mbps, reaches its maximum at around 29 times at 17 Mbps, and gradually decreases to about 16 times at 25 Mbps, and eventually becomes negligible at 30 Mbps. Therefore, the proposed approach significantly enhances the quality of the viewport compared to the baseline strategy.

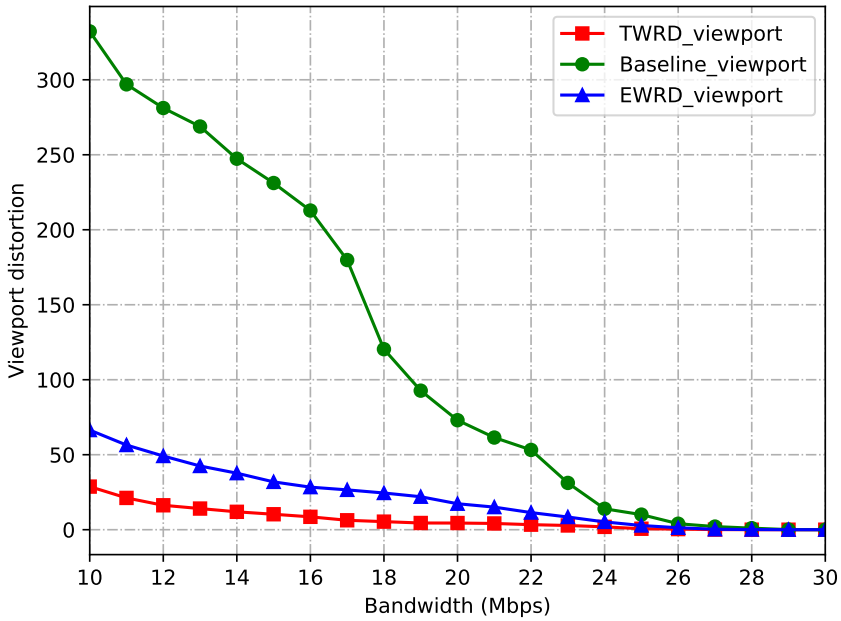

Figure 4.6: Viewport distortions on VR video "PortoRiverside" for three methods at varying bandwidths.

4.5.5 Packet Loss Analysis

Figure 4.7 displays the viewport bandwidth consumption for the three packet scheduling methods at varying bandwidth levels. As expected, all three strategies show an upward trend in viewport bandwidth consumption with increasing available bandwidth. At lower bandwidths, the baseline strategy consumes the least amount of viewport bandwidth, which is expected due to its higher packet loss rates and, consequently, lower data transmission. However, as the available bandwidth increases, the viewport bandwidth consumption of the baseline strategy also increases significantly, surpassing that of TWRD and EWRD.

On the other hand, both TWRD and EWRD exhibit similar trends, with slightly higher viewport bandwidth consumption compared to the baseline strategy at lower bandwidths. However, as the bandwidth becomes sufficient to accommodate most packets without significant loss, TWRD and EWRD converge to a similar level of viewport bandwidth consumption. This is consistent with their ability to optimize the packet

scheduling and allocate the available bandwidth more effectively to minimize packet losses and distortions, resulting in improved overall video quality and reduced bandwidth waste. Overall, the results indicate that TWRD achieves a better trade-off between distortion minimization and bandwidth utilization compared to the baseline and EWRD methods. By considering both the importance of packets and their impact on distortion, TWRD strikes a balance between delivering a high-quality VR video experience and efficiently utilizing the available network bandwidth.

4.5.6 Bandwidth Consumption Analysis

In the context of the streaming system with limited bandwidth and no background traffic, the total bandwidth consumption for the VR video is directly proportional to the available bandwidth. The percentage of viewport bandwidth consumption plays a crucial role in evaluating the efficiency of resource allocation. Higher percentages signify more effective allocation methods. The visual representation of viewport bandwidth consumption in Figure 4.8 reveals the following trends: Throughout the range of available bandwidths, TWRD consistently exhibits the highest consumption, followed by EWRD, while the baseline strategy demonstrates the lowest consumption. This pattern arises due to the viewport's fixed bandwidth requirement, causing the percentages for TWRD and EWRD to gradually decline as overall bandwidth increases. Notably, the baseline strategy showcases fluctuations in the consumption percentages, hovering around 30% for different bandwidths. This behavior stems from the random dropping of packets in the baseline method. Despite fewer packet losses with increasing bandwidth, each tile has an equal probability of being dropped, leading to the observed fluctuations.

In contrast, both TWRD and EWRD employ rate-distortion information to selectively drop packets, resulting in a decreasing trend of bandwidth consumption for both strategies. For example, at 10 Mbps bandwidth, TWRD consumes around 61.6% of the total bandwidth, which is approximately twice as efficient as the baseline (30.0%) and 1.5 times better than EWRD (40.6%). As the available bandwidth continues to increase,

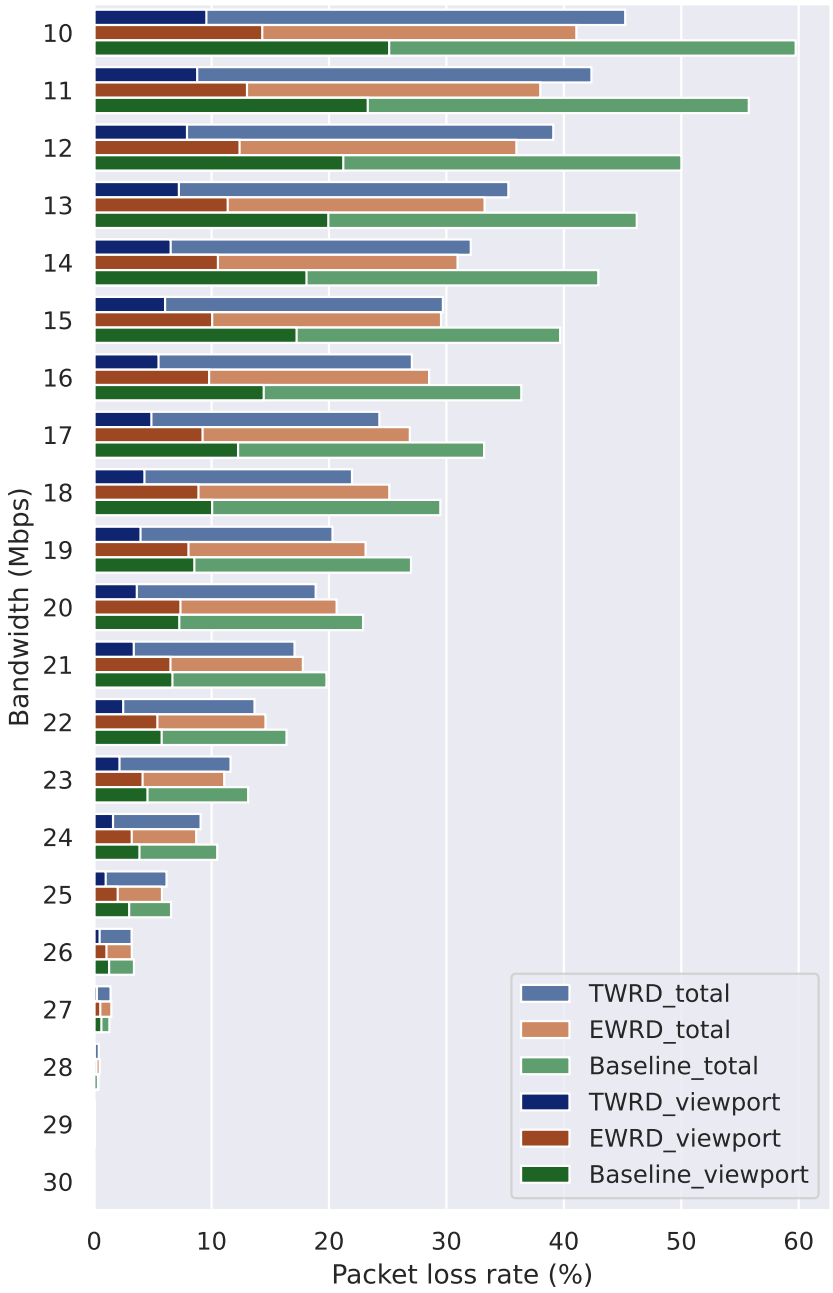

Figure 4.7: The packet loss rate experienced by all tiles and the viewport in the VR video "PortoRiverside" using three different methods at varying bandwidth levels.

the differences in bandwidth consumption between the three methods gradually diminish until they reach 30 Mbps. At this point, representing the minimum sufficient bandwidth for the VR video, all methods converge to the same consumption percentage of 29.7%, indicating that the viewport data accounts for 29.7% of the entire VR video.

In conclusion, the allocation of bandwidth resources to the viewport demonstrates higher efficiency in TWRD and EWRD compared to the baseline strategy, with TWRD achieving the highest consumption and providing a significant improvement over the other methods. However, as the available bandwidth increases, the distinctions between the three strategies become less pronounced, ultimately converging to the same percentage at the minimum sufficient bandwidth.

4.5.7 Evaluation on Different Videos

Table 4.1 presents a comprehensive analysis of two VR videos, Warship and PortoRiverside, under varying bandwidths (10 Mbps, 15 Mbps, 20 Mbps, 25 Mbps, 30 Mbps), utilizing five evaluation metrics. The metrics are evaluated for three methods: TWRD, EWRD, and the baseline. Although the videos exhibit similar trends, the values differ due to their inherent characteristics and the random nature of the baseline method.

EWRD consistently demonstrates the lowest overall distortion across both videos, closely approaching the performance of TWRD under different bandwidths. However, it should be noted that the Warship video experiences non-zero distortion at 30 Mbps, indicating that a minimum bandwidth of 32 Mbps is necessary for distortion-free delivery. Regarding viewport distortion, the differences between the methods are relatively smaller in the Warship video compared to the PortoRiverside video. For instance, at 15 Mbps, TWRD outperforms EWRD by approximately three times in the Warship video and seven times in the PortoRiverside video, reaffirming the superiority of the proposed method.

In terms of total packet loss rate, the baseline method achieves the lowest rate for the Warship video at most bandwidths (15 Mbps, 20 Mbps, 25 Mbps). However, the

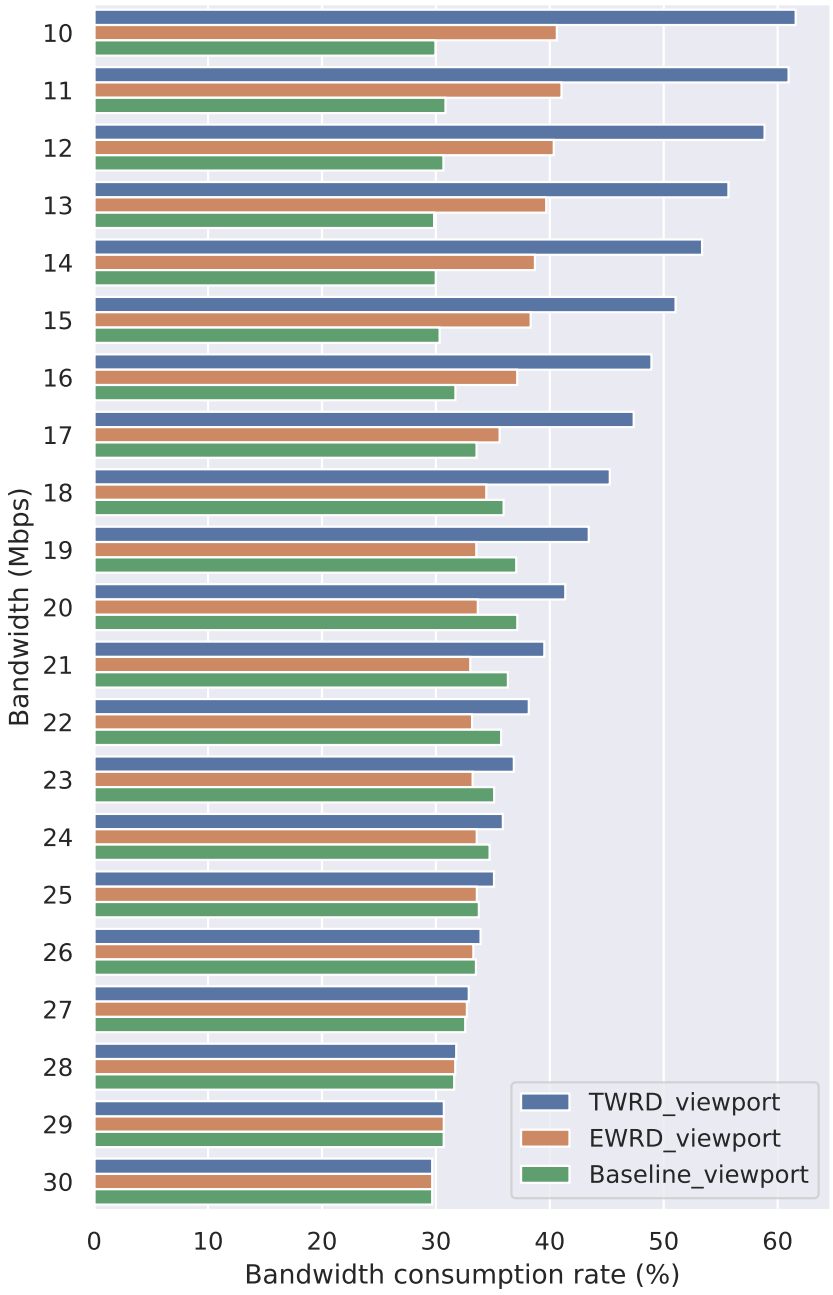

Figure 4.8: Viewport bandwidth consumption for three methods at different bandwidth levels.

differences among the methods are minimal, with the largest gap being around 4.5% for bandwidths of 20 Mbps and 25 Mbps. Higher packet loss rate does not necessarily imply higher distortion, as some dropped packets have a more significant impact than others. The baseline's higher loss rate suggests that it drops crucial packets, leading to more substantial distortion, while the other methods' dropped packets have less impact, resulting in smaller distortions. This highlights the effectiveness of the proposed method.

Furthermore, the proposed TWRD consistently achieves the lowest viewport packet loss rate across different bandwidths, indicating its ability to selectively drop non-viewport packets. TWRD also allocates a higher proportion of bandwidth resources to the viewport compared to the other methods. As the bandwidth increases to 30 Mbps, all three methods converge to allocating approximately 29.7% of the bandwidth to the viewport.

The findings presented in Table 4.1 demonstrate the generality and superior performance of the proposed TWRD method under diverse bandwidth conditions and across different VR videos.

Table 4.1: Results of the experiments conducted on the two videos, PortoRiverside and Warship, using the three methods.

Video	Metric	10 Mbps	15 Mbps	20 Mbps	25 Mbps	30 Mbps
PortoRiverside 	Total distortion	233.1	125.3	51.9	6.7	0
		209.2	**112.9**	**44.0**	**5.6**	0
		617.5	404.4	191.6	27.0	0
	Viewport distortion	**28.7**	**10.3**	**4.4**	**0.6**	0
		66.4	31.9	17.3	2.8	0
		332.3	231.2	73.0	10.1	0
	Total packet loss	45.2%	29.7%	**18.9%**	6.2%	0%
		41.1%	**29.5%**	20.7%	**5.8%**	0%
		59.7%	39.7%	22.9%	6.6%	0%
	Viewport packet loss	**9.6%**	**6.0%**	**3.6%**	**1.0%**	0%
		14.3%	10.1%	7.4%	2.0%	0%
		25.1%	17.2%	7.2%	3.0%	0%
	Viewport bandwidth	**61.6%**	**51.0%**	**41.4%**	**35.1%**	29.7%
		40.6%	38.3%	33.7%	33.6%	29.7%
		30.0%	30.3%	37.1%	33.8%	29.7%
Warship 	Total distortion	357.2	207.4	106.6	38.0	3.6
		322.4	**185.4**	**93.0**	**31.7**	**2.9**
		816.3	456.7	264.9	133.5	15.3
	Viewport distortion	**43.3**	**23.0**	**9.8**	**3.0**	**0.3**
		96.2	59.5	31.4	13.9	1.6
		457.1	168.1	69.3	34.6	4.7
	Total packet loss	**63.4%**	48.8%	34.5%	18.9%	2.6%
		65.4%	49.5%	34.2%	17.1%	**1.4%**
		63.8%	**46.1%**	**30.1%**	**14.3%**	1.7%
	Viewport packet loss	**11.4%**	**7.2%**	**3.2%**	**0.8%**	**0%**
		19.7%	14.2%	9.3%	4.3%	0.2%
		21.5%	12.9%	7.5%	3.2%	0.4%
	Viewport bandwidth	**68.2%**	**53.9%**	**46.4%**	**40.8%**	**35.7%**
		46.1%	39.5%	36.9%	35.1%	34.8%
		42.1%	44.6%	42.2%	39.0%	35.4%

Note: The values for three methods are listed in the order TWRD, EWRD, and baseline for every metric.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

We propose two solutions at the application layer and network layer to improve QoE, separately. The proposed MARL adaptive bitrate streaming system aims to improve VR video streaming performance under limited bandwidth conditions by predicting multiple-viewpoint trajectories. It utilizes a lightweight transformer network to classify future trajectories and their probabilities, while an attention mechanism and historical information further enhance the network's performance. Using the multiple predictions, a MARL approach is employed to optimize QoE across varying bandwidths. Dec-POMDP is used to solve the optimization problem to manage multi-agent interactions as well as uncertainty in environmental conditions. By assigning each agent to determine the bitrate for a specific viewport region, the system reduces the number of agents, leading to more efficient decision-making. As part of the MARL framework, MAPPO is integrated, a state-of-the-art algorithm for cooperative multi-agent scenarios. A comparison of the proposed method to existing ABR methods shows that it outperforms existing methods, significantly improving QoE metrics under a wide range of bandwidth conditions.

TWRD optimizes packet scheduling schemes for VR video streaming under limited bandwidth by dynamically weighting tiles and packets based on viewport prediction prob-

abilities. TWRD solves the optimization problem using dynamic programming by taking into account tile-weighted rate-distortion information. It allows intelligent packet dropping at network nodes, prioritizing packets to minimize distortion and improve viewport quality. Compared with other existing methods, TWRD effectively reduces viewport distortion and overall performance, demonstrating its superior ability to optimize VR video streaming in low-bandwidth environments. By leveraging viewport prediction probabilities, dynamic programming, and weighted rate-distortion information, the TWRD system proves to be an effective solution for enhancing VR video streaming experiences.

5.2 Future Work

Acknowledging the limitations of our current approach, we recognize the need for future developments to enhance its real-world applicability. To improve user QoE, a joint optimization framework that considers the cascading influence of rate adaptation and packet scheduling could be explored. The simplicity of our proposed rate adaptation scheme, based on a single QoE metric, contrasts with the more complex real-world QoE metrics. Investigating end-to-end learning-based methods that identify efficient metrics, specifically tailored for tile-based 360-degree video streaming, holds promise for further advancements. Due to the Covide-19, user studies can not be conducted to evaluate our methods in practice. Therefore, some experiments on various users can be implemented to evaluate user experience in the future.

Practical deployment of our methods requires careful consideration of various issues. One challenge lies in the generalization of the MARL method across heterogeneous user and video conditions, including fluctuating network throughput and diverse video complexities. Overfitting risks in the MARL ABR, driven by neural network policy approximations, could be addressed, particularly in scenarios with temporally correlated training samples. Additionally, our adaptive streaming and packet scheduling may introduce time costs during video streaming in practice, as the proposed algorithms require

additional time to complete their tasks. Thus, future work should focus on optimizing time cost reduction to ensure efficient and seamless video delivery to users. The dynamic programming is used due to the lack of network trace data now. Hence, network traces of VR video streaming can be collected, and deep learning methods could be used for packet scheduling in the future.

References

[1] Y. Zhang, Y. Guan, K. Bian, *et al.*, "Epass360: Qoe-aware 360-degree video streaming over mobile devices," *IEEE Transactions on Mobile Computing*, pp. 1–1, 2020.

[2] C.-L. Fan, S.-C. Yen, C.-Y. Huang, and C.-H. Hsu, "Optimizing fixation prediction using recurrent neural networks for 360° video streaming in head-mounted virtual reality," *IEEE Transactions on Multimedia*, vol. 22, no. 3, pp. 744–759, 2020.

[3] F. Qian, L. Ji, B. Han, and V. Gopalakrishnan, "Optimizing 360 video delivery over cellular networks," in *Proceedings of the 5th Workshop on All Things Cellular: Operations, Applications and Challenges*, 2016, pp. 1–6.

[4] I. D. Inc. "10 of the best augmented reality (ar) shopping apps to try today." (), [Online]. Available: `https://www.indigo9digital.com/blog/how-six-leading-retailers-use-augmented-reality-apps-to-disrupt-the-shopping-experience`. (accessed: 10.06.2023).

[5] J. Brewer. "Cisco launcheswebex hologram, an ar meeting solution." (), [Online]. Available: `https://newsroom.cisco.com/c/r/newsroom/en/us/a/y2021/m10/wx1-hologram-release-placeholder.html`. (accessed: 10.06.2023).

[6] J. Reyna, "The potential of 360-degree videos for teaching, learning and research," in *INTED2018 proceedings*, IATED, 2018, pp. 1448–1454.

[7] G. Lampropoulos, V. Barkoukis, K. Burden, and T. Anastasiadis, "360-degree video in education: An overview and a comparative social media data analysis of the last decade," *Smart Learning Environments*, vol. 8, no. 1, pp. 1–24, 2021.

[8] *Huawei-ilab*, `https://www-file.huawei.com/-/media/corporate/pdf/ilab/cloud_vr_oriented_bearer_network_white_paper_en_v2.pdf`, Accessed: 2022-01-02.

[9] E. Bastug *et al.*, "Toward interconnected virtual reality: Opportunities, challenges, and enablers," *IEEE Commun. Mag.*, vol. 55, no. 6, pp. 110–117, Jun. 2017.

[10] Y. Mao, L. Sun, Y. Liu, and Y. Wang, "Low-latency fov-adaptive coding and streaming for interactive 360° video streaming," in *Proceedings of the 28th ACM International Conference on Multimedia*, 2020, pp. 3696–3704.

[11] M. Hosseini and V. Swaminathan, "Adaptive 360 VR video streaming: Divide and conquer," in *Proceedings of IEEE International Symposium on Multimedia*, 2016, pp. 107–110.

[12] L. Xie, Z. Xu, Y. Ban, X. Zhang, and Z. Guo, "360ProbDASH: Improving QoE of 360 video streaming using tile-based HTTP adaptive streaming," in *Proceedings of the 25th ACM International Conference on Multimedia*, 2017, pp. 315–323.

[13] M. Hosseini, "View-aware tile-based adaptations in 360 virtual reality video streaming," in *Proceedings of IEEE Virtual Reality*, 2017, pp. 423–424.

[14] T. Stockhammer, "Dynamic adaptive streaming over HTTP– standards and design principles," in *Proceedings of the 2nd ACM conference on Multimedia systems*, 2011, pp. 133–144.

[15] S. Petrangeli, G. Simon, and V. Swaminathan, "Trajectory-based viewport prediction for 360-degree virtual reality videos," in *Proceedings of IEEE International Conference on Artificial Intelligence and Virtual Reality*, 2018, pp. 157–160.

[16] A. Nguyen, Z. Yan, and K. Nahrstedt, "Your attention is unique: Detecting 360-degree video saliency in head-mounted display for head movement prediction," in *Proceedings of the 26th ACM International Conference on Multimedia*, 2018, pp. 1190–1198.

[17] M. F. R. Rondón, L. Sassatelli, R. Aparicio-Pardo, and F. Precioso, "Track: A new method from a re-examination of deep architectures for head motion prediction in 360° videos," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 9, pp. 5681–5699, 2022.

[18] F.-Y. Chao, C. Ozcinar, and A. Smolic, "Transformer-based long-term viewport prediction in 360° video: Scanpath is all you need," in *Proceedings of IEEE 23rd International Workshop on Multimedia Signal Processing*, 2021, pp. 1–6.

[19] M. Xu, Y. Song, J. Wang, M. Qiao, L. Huo, and Z. Wang, "Predicting head movement in panoramic video: A deep reinforcement learning approach," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 11, pp. 2693–2708, 2019.

[20] J. Fu, Z. Chen, X. Chen, and W. Li, "Sequential reinforced 360-degree video adaptive streaming with cross-user attentive network," *IEEE Transactions on Broadcasting*, vol. 67, no. 2, pp. 383–394, 2021.

[21] Z. Jiang, X. Zhang, Y. Xu, Z. Ma, J. Sun, and Y. Zhang, "Reinforcement learning based rate adaptation for 360-degree video streaming," *IEEE Transactions on Broadcasting*, vol. 67, no. 2, pp. 409–423, 2021.

[22] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, "Attention is all you need," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017, pp. 6000–6010.

[23] L. E. Gurrieri and E. Dubois, "Acquisition of omnidirectional stereoscopic images and videos of dynamic scenes: a review," *Journal of Electronic Imaging*, vol. 22, no. 3, p. 030 902, 2013.

[24] R. Anderson, D. Gallup, J. T. Barron, *et al.*, "Jump: Virtual reality video," *ACM Transactions on Graphics (TOG)*, vol. 35, no. 6, pp. 1–13, 2016.

[25] S. Knorr, S. Croci, and A. Smolic, "A modular scheme for artifact detection in stereoscopic omni-directional images," in *Proceedings of the Irish Machine Vision and Image Processing Conference*, 2017.

[26] W. Jiang and J. Gu, "Video stitching with spatial-temporal content-preserving warping," in *2015 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2015, pp. 42–48.

[27] S. Akhshabi, L. Anantakrishnan, C. Dovrolis, and A. C. Begen, "Server-based traffic shaping for stabilizing oscillating adaptive streaming players," in *Proceeding of the 23rd ACM Workshop on Network and Operating Systems Support for Digital Audio and Video*, ser. NOSSDAV '13, Oslo, Norway: Association for Computing Machinery, 2013, pp. 19–24, ISBN: 9781450318921.

[28] J. Tan, G. Cheung, and R. Ma, "360-degree virtual-reality cameras for the masses," *IEEE MultiMedia*, vol. 25, no. 1, pp. 87–94, 2018.

[29] M. U. Younus, M. A. Nadeem, W. Abbas, L. Yong, and R. Shafi, "Design and implementation of wireless sensor networks integrated with rfid tags for navigation purpose," in *2016 2nd IEEE International Conference on Computer and Communications (ICCC)*, 2016, pp. 2240–2245.

[30] X. Corbillon, F. De Simone, and G. Simon, "360-degree video head movement dataset," in *Proceedings of the 8th ACM on Multimedia Systems Conference*, ser. MMSys'17, Taipei, Taiwan: Association for Computing Machinery, 2017, pp. 199–204, ISBN: 9781450350020.

[31] D. Grois and A. Giladi, "Performance assessment of av1, x265 and vvenc open-source encoder implementations compared to vvc and hevc reference software models," ser. MHV '23, Denver, CO, USA: Association for Computing Machinery, 2023, pp. 146–147, ISBN: 9798400701603.

[32] J. Bankoski, P. Wilkins, and Y. Xu, "Technical overview of vp8, an open source video codec for the web," in *2011 IEEE International Conference on Multimedia and Expo*, 2011, pp. 1–6.

[33] D. Mukherjee, J. Bankoski, A. Grange, *et al.*, "The latest open-source video codec vp9 - an overview and preliminary results," in *2013 Picture Coding Symposium (PCS)*, 2013, pp. 390–393.

[34] D. Mukherjee, H. Su, J. Bankoski, *et al.*, "An overview of new video coding tools under consideration for VP10: the successor to VP9," in *Applications of Digital Image Processing XXXVIII*, A. G. Tescher, Ed., International Society for Optics and Photonics, vol. 9599, SPIE, 2015, 95991E.

[35] M. Wien, J. M. Boyce, T. Stockhammer, and W.-H. Peng, "Standardization status of immersive video coding," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 9, no. 1, pp. 5–17, 2019.

[36] Y. Li, A. Markopoulou, J. Apostolopoulos, and N. Bambos, "Content-aware playout and packet scheduling for video streaming over wireless links," *IEEE Transactions on Multimedia*, vol. 10, no. 5, pp. 885–895, 2008.

[37] K. Liu, Y. Liu, J. Liu, A. Argyriou, and X. Yang, "Joint source encoding and networking optimization for panoramic video streaming over lte-a downlink," in *GLOBECOM 2017-2017 IEEE Global Communications Conference*, IEEE, 2017, pp. 1–7.

[38] C. Concolato, J. Le Feuvre, F. Denoual, *et al.*, "Adaptive streaming of hevc tiled videos using mpeg-dash," *IEEE transactions on circuits and systems for video technology*, vol. 28, no. 8, pp. 1981–1992, 2017.

[39] M. U. Younus, S. S. H. Bukhari, W. Abbas, and R. Shafi, "Optimized indoor lighting system to save energy through window blinds management," in *2017 3rd IEEE International Conference on Computer and Communications (ICCC)*, IEEE, 2017, pp. 2817–2821.

[40] C. Zhou, Z. Li, J. Osgood, and Y. Liu, "On the effectiveness of offset projections for 360-degree video streaming," vol. 14, no. 3s, Jun. 2018, ISSN: 1551-6857.

[41] C. Ozcinar, A. De Abreu, and A. Smolic, "Viewport-aware adaptive 360° video streaming using tiles for virtual reality," in *2017 IEEE International Conference on Image Processing (ICIP)*, 2017, pp. 2174–2178.

[42] K. K. Sreedhar, A. Aminlou, M. M. Hannuksela, and M. Gabbouj, "Viewport-adaptive encoding and streaming of 360-degree video for virtual reality applications," in *2016 IEEE International Symposium on Multimedia (ISM)*, 2016, pp. 583–586.

[43] T. Stockhammer, "Dynamic adaptive streaming over http– standards and design principles," in *Proceedings of the second annual ACM conference on Multimedia systems*, 2011, pp. 133–144.

[44] K. Doppler, E. Torkildson, and J. Bouwen, "On wireless networks for the era of mixed reality," in *2017 European Conference on Networks and Communications (EuCNC)*, IEEE, 2017, pp. 1–5.

[45] R. Skupin, Y. Sanchez, C. Hellge, and T. Schierl, "Tile based hevc video for head mounted displays," in *2016 IEEE International Symposium on Multimedia (ISM)*, 2016, pp. 399–400.

[46] M. Graf, C. Timmerer, and C. Mueller, "Towards bandwidth efficient adaptive streaming of omnidirectional video over http: Design, implementation, and evaluation," in *Proceedings of the 8th ACM on Multimedia Systems Conference*, New York, NY, USA: Association for Computing Machinery, 2017, pp. 261–271.

[47] M. Yu, H. Lakshman, and B. Girod, "Content adaptive representations of omnidirectional videos for cinematic virtual reality," in *Proceedings of the 3rd International Workshop on Immersive Media Experiences*, ser. ImmersiveME '15, Brisbane, Australia: Association for Computing Machinery, 2015, pp. 1–6, ISBN: 9781450337458.

[48] C. Ozcinar, J. Cabrera, and A. Smolic, "Visual attention-aware omnidirectional video streaming using optimal tiles for virtual reality," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 9, no. 1, pp. 217–230, 2019.

[49] D. V. Nguyen, H. T. T. Tran, A. T. Pham, and T. C. Thang, "An optimal tile-based approach for viewport-adaptive 360-degree video streaming," *IEEE Journal*

on Emerging and Selected Topics in Circuits and Systems, vol. 9, no. 1, pp. 29–42, 2019.

[50] F. Qian, B. Han, Q. Xiao, and V. Gopalakrishnan, "Flare: Practical viewport-adaptive 360-degree video streaming for mobile devices," in *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking*, Association for Computing Machinery, 2018, pp. 99–114, ISBN: 9781450359030.

[51] R. I. T. D. C. Filho, M. C. Luizelli, S. Petrangeli, *et al.*, "Dissecting the performance of VR video streaming through the VR-EXP experimentation platform," *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 15, no. 4, Dec. 2019, ISSN: 1551-6857.

[52] L. Yu, T. Tillo, and J. Xiao, "Qoe-driven dynamic adaptive video streaming strategy with future information," *IEEE Transactions on Broadcasting*, vol. 63, no. 3, pp. 523–534, 2017.

[53] C. Li, M. Xu, X. Du, and Z. Wang, "Bridge the gap between VQA and human behavior on omnidirectional video," *Proceedings of the 26th ACM international conference on Multimedia*, Oct. 2018.

[54] F. Chiariotti, S. D'Aronco, L. Toni, and P. Frossard, "Online learning adaptation strategy for dash clients," in *Proceedings of the 7th International Conference on Multimedia Systems*, Association for Computing Machinery, 2016, ISBN: 9781450342971.

[55] M. T. Vega, D. C. Mocanu, R. Barresi, G. Fortino, and A. Liotta, "Cognitive streaming on android devices," in *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, 2015, pp. 1316–1321.

[56] K. Boos, D. Chu, and E. Cuervo, "Demo: Flashback: Immersive virtual reality on mobile devices via rendering memoization," in *Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services Companion*,

New York, NY, USA: Association for Computing Machinery, 2016, p. 94, ISBN: 9781450344166.

[57] Z. Lai, Y. C. Hu, Y. Cui, L. Sun, N. Dai, and H.-S. Lee, "Furion: Engineering high-quality immersive virtual reality on today's mobile devices," *IEEE Transactions on Mobile Computing*, vol. 19, no. 7, pp. 1586–1602, 2020.

[58] Z. Xu, X. Zhang, K. Zhang, and Z. Guo, "Probabilistic viewport adaptive streaming for 360-degree videos," in *Proceedings of 2018 IEEE International Symposium on Circuits and Systems*, 2018, pp. 1–5.

[59] A. T. Nasrabadi, A. Samiei, and R. Prakash, "Viewport prediction for 360° videos: A clustering approach," in *Proceedings of the 30th ACM Workshop on Network and Operating Systems Support for Digital Audio and Video*, 2020, pp. 34–39.

[60] Q. Guimard, L. Sassatelli, F. Marchetti, F. Becattini, L. Seidenari, and A. D. Bimbo, "Deep variational learning for multiple trajectory prediction of 360° head movements," in *Proceedings of the 13th ACM Multimedia Systems Conference*, 2022, pp. 12–26.

[61] L. Xie, X. Zhang, and Z. Guo, "CLS: A cross-user learning based system for improving QoE in 360-degree video adaptive streaming," in *Proceedings of the 26th ACM International Conference on Multimedia*, 2018, pp. 564–572.

[62] V. Mnih, A. P. Badia, M. Mirza, *et al.*, "Asynchronous methods for deep reinforcement learning," in *in Proceedings of International conference on machine learning*, 2016, pp. 1928–1937.

[63] S. Park, M. Hoai, A. Bhattacharya, and S. R. Das, "Adaptive streaming of 360-degree videos with reinforcement learning," in *Proceedings of IEEE Winter Conference on Applications of Computer Vision*, 2021, pp. 1838–1847.

[64] Y. Zhang, P. Zhao, K. Bian, Y. Liu, L. Song, and X. Li, "DRL360: 360-degree video streaming with deep reinforcement learning," in *Proceedings of IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*, 2019, pp. 1252–1260.

[65] N. Kan, J. Zou, C. Li, W. Dai, and H. Xiong, "RAPT360: Reinforcement learning-based rate adaptation for 360-degree video streaming with adaptive prediction and tiling," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 32, no. 3, pp. 1607–1623, 2022.

[66] X. Wei, M. Zhou, S. Kwong, H. Yuan, and W. Jia, "A hybrid control scheme for 360-degree dynamic adaptive video streaming over mobile devices," *IEEE Transactions on Mobile Computing*, vol. 21, no. 10, pp. 3428–3442, 2022.

[67] R. Mittal, R. Agarwal, S. Ratnasamy, and S. Shenker, "Universal packet scheduling," in *Proceedings of the 14th ACM workshop on Hot Topics in Networks*, 2015, pp. 1–7.

[68] H. Cha, J. Oh, and R. Ha, "Dynamic frame dropping for bandwidth control in MPEG streaming system," *Multimedia Tools and Applications*, vol. 19, no. 2, pp. 155–178, 2003, ISSN: 13807501.

[69] J. Chakareski and P. Frossard, "Rate-distortion optimized distributed packet scheduling of multiple video streams over shared communication resources," *IEEE Transactions on Multimedia*, vol. 8, no. 2, pp. 207–218, 2006.

[70] M. M. Nasralla, M. Razaak, I. U. Rehman, and M. G. Martini, "Content-aware packet scheduling strategy for medical ultrasound videos over LTE wireless networks," *Computer Networks*, vol. 140, pp. 126–137, 2018, ISSN: 1389-1286.

[71] Y.-L. Chang, T.-L. Lin, and P. C. Cosman, "Network-based H.264/AVC whole-frame loss visibility model and frame dropping methods," *IEEE Transactions on Image Processing*, vol. 21, no. 8, pp. 3353–3363, 2012.

[72] I.-S. Comşa, G.-M. Muntean, and R. Trestian, "An innovative machine-learning-based scheduling solution for improving live UHD video streaming quality in highly dynamic network environments," *IEEE Transactions on Broadcasting*, vol. 67, no. 1, pp. 212–224, 2021.

[73] W. Wei, J. Han, Y. Xing, K. Xue, J. Liu, and R. Zhuang, "MP-VR: An MPTCP-based adaptive streaming framework for 360-degree virtual reality videos," in *Proceedings of IEEE International Conference on Communications*, 2021, pp. 1–6.

[74] J. Chakareski, "Viewport-adaptive scalable multi-user virtual reality mobile-edge streaming," *IEEE Transactions on Image Processing*, vol. 29, pp. 6330–6342, 2020.

[75] M. L. Littman, "Markov games as a framework for multi-agent reinforcement learning," in *Machine learning proceedings 1994*, Elsevier, 1994, pp. 157–163.

[76] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[77] J. L. Ba, J. R. Kiros, and G. E. Hinton, *Layer normalization*, 2016. arXiv: `1607.06450 [stat.ML]`.

[78] A. Dosovitskiy, L. Beyer, A. Kolesnikov, *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," in *in Proceedings of International Conference on Learning Representations*, 2021.

[79] F. A. Oliehoek and C. Amato, *A concise introduction to decentralized POMDPs*. Springer, 2016.

[80] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[81] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[82] G. F. Riley and T. R. Henderson, "The ns-3 network simulator," in *Modeling and Tools for Network Simulation*, Springer, 2010, pp. 15–34.

[83] E. J. David, J. Gutiérrez, A. Coutrot, M. P. Da Silva, and P. L. Callet, "A dataset of head and eye movements for 360° videos," in *Proceedings of the 9th ACM Multimedia Systems Conference*, 2018, pp. 432–437.

[84] F. Marchetti, F. Becattini, L. Seidenari, and A. D. Bimbo, "Mantra: Memory augmented networks for multiple trajectory prediction," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 7143–7152.

[85] T.-Y. Huang, R. Johari, N. McKeown, M. Trunnell, and M. Watson, "A buffer-based approach to rate adaptation: Evidence from a large video streaming service," in *Proceedings of the 2014 ACM Conference on SIGCOMM*, 2014, pp. 187–198.

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

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

[88] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS'17, Long Beach, California, USA: Curran Associates Inc., 2017, pp. 6382–6393, ISBN: 9781510860964.

[89] T. P. Lillicrap, J. J. Hunt, A. Pritzel, *et al.*, "Continuous control with deep reinforcement learning," in *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2016.

[90] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with gumbel-softmax," in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, OpenReview.net, 2017.

[91] T. Rashid, M. Samvelyan, C. S. De Witt, G. Farquhar, J. Foerster, and S. Whiteson, "Monotonic value function factorisation for deep multi-agent reinforcement learning," *J. Mach. Learn. Res.*, vol. 21, no. 1, Jun. 2022, ISSN: 1532-4435.

[92] C. S. de Witt, T. Gupta, D. Makoviichuk, *et al.*, "Is independent learning all you need in the starcraft multi-agent challenge?" *arXiv preprint arXiv:2011.09533*, 2020.

[93] K. Stuhlmuller, N. Farber, M. Link, and B. Girod, "Analysis of video transmission over lossy channels," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 6, pp. 1012–1032, 2000.

[94] Y. Liang, J. Apostolopoulos, and B. Girod, "Analysis of packet loss for compressed video: Does burst-length matter?" In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 5, 2003, pp. V–684.

[95] Y. Li, A. Markopoulou, J. Apostolopoulos, and N. Bambos, "Content-aware playout and packet scheduling for video streaming over wireless links," *IEEE Transactions on Multimedia*, vol. 10, no. 5, pp. 885–895, 2008.

[96] X. Corbillon, F. Boyrivent, G. A. De Williencourt, G. Simon, G. Texier, and J. Chakareski, "Efficient lightweight video packet filtering for large-scale video data delivery," in *Proceedings of IEEE International Conference on Multimedia & Expo Workshops*, 2016, pp. 1–6.