Tile-Weighted Rate-Distortion Optimized Packet Scheduling for 360° Virtual-Reality Video Streaming

Haopeng Wang, *University of Ottawa, Ottawa, ON, K1N 6N5, Canada*
Haiwei Dong, *Huawei Canada Ottawa, ON, K2K 3J1, Canada*
Abdulmotaleb El Saddik, *University of Ottawa, Ottawa, ON, K1N 6N5, Canada*

A key challenge of 360° virtual-reality (VR) video streaming is ensuring high quality with limited network bandwidth. Currently, most of the studies focus on tile-based adaptive bit-rate streaming to reduce bandwidth consumption, where resources in network nodes are not fully utilized. This article proposes a tile-weighted rate-distortion (TWRD) packet scheduling optimization system to reduce data volume and improve video quality. A multimodal spatial–temporal attention transformer is proposed to predict viewpoint with probability that is used to dynamically weight tiles and their corresponding packets. The packet scheduling problem of determining which packets should be dropped is formulated as an optimization problem solved by a dynamic programming solution. Experiment results demonstrate that the proposed method outperforms the existing methods under various conditions.

With the popularity of virtual-reality (VR),[1] 360° video, also known as *VR video*, is also attracting more attention. Streaming high-quality VR videos faces challenges due to bandwidth requirements and varying network conditions. However, a user only sees the contents inside the viewport at a time, and too many resources are wasted in delivering the rest of the content that the user does not view. Therefore, tile-based adaptive bit-rate streaming approaches are proposed to reduce data volume. VR video streaming rarely uses the computational resources of network nodes. Network congestion and rebuffering occur when VR video is streamed over an inadequate bandwidth. Network nodes drop packets randomly, which can affect reconstructed video unexpectedly. If certain less important packets are dropped, the quality of the reconstructed video could degrade in a controllable and limited manner. Additionally, dropping packets outside the viewport could reduce distortion of its viewport.

1541-1672 © 2024 IEEE
Digital Object Identifier 10.1109/MIS.2024.3385313
Date of publication 5 April 2024; date of current version 30 July 2024.

This article proposes a tile-weighted rate-distortion (TWRD) packet scheduling system to reduce data volume and video distortion based on a multimodal spatial–temporal attention transformer. A VR video is divided into tiles, where we assume that each tile's frame has one packet.[2,3,4] The rate-distortion information of a packet consists of the rate and distortion, where the rate represents the size of the packet, and the distortion represents the quality impact if the packet is lost. As shown in Figure 1, a VR video is initially streamed at the lowest quality. As the video plays, the transformer model predicts future viewpoints with probabilities based on the user's historical viewpoint trajectory. Every tile is then weighted based on the probability. When the queue size of a network node exceeds a threshold regardless of the bandwidth conditions and video quality, the network node starts the proposed dynamic programming solution to determine a packet scheduling scheme and drops packets as scheduled to avoid network congestion and reduce quality distortion. The distortion mainly depends on the type of dropped packets. As shown in Figure 1, dropping an I-packet induces significant distortion that spans all frames within the group of pictures. Similarly,

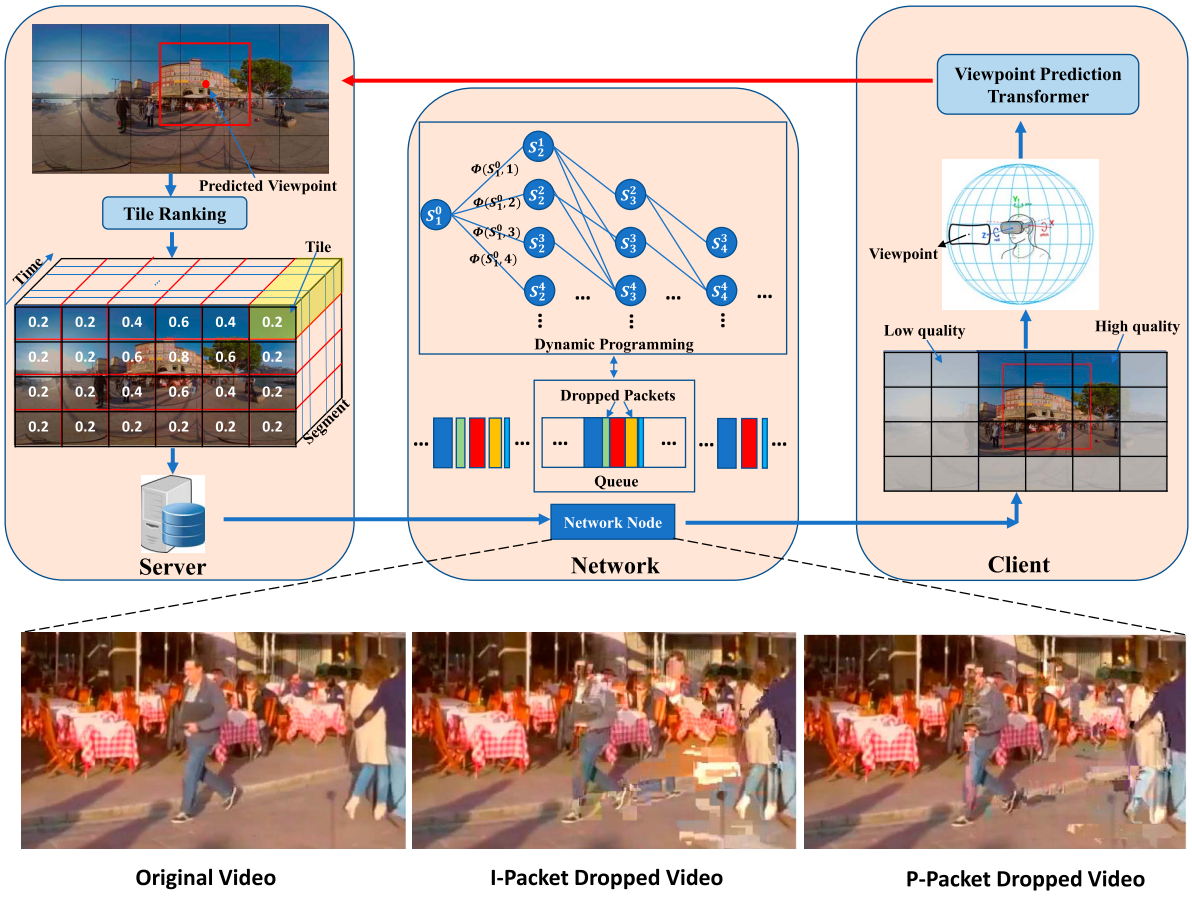

FIGURE 1. A TWRD packet scheduling optimization system for VR video streaming. Each packet has a color that indicates its importance, and a width indicating its size. Due to bandwidth restrictions, some packets are dropped according to the proposed method. The visual quality of the reconstructed video mainly depends on the type of the dropped packet.

if a P-packet is lost, noticeable distortion occurs, making all subsequent frames incapable of decoding. As other frames do not utilize B-packets, the drop of a B-packet results in the loss of only one reconstructed frame, which users cannot notice. The following is a summary of the contribution of this article:

- A TWRD optimized VR video streaming system is proposed to improve video quality.
- A multimodal spatial–temporal attention transformer is proposed to predict viewpoint with probability that is used to rank tiles dynamically.
- The packet scheduling problem is formulated as an optimization problem, which is solved by a proposed dynamic-programming-based solution.
- Experiment results demonstrate that the proposed approach outperforms other approaches under varying bandwidths.

RELATED WORK

There are many existing packet scheduling algorithms for traditional video streaming. The tail drop algorithm is a simple and popular packet scheduling algorithm. However, the tail drop does not differentiate among packet types. Hence, dropping video frames based on frame types (I, P, and B) with different priorities is proposed.[5] Moharrami et al.[5] drop packets in the order of B, P, and I. Gobatto et al.[2] preemptively drop non-intra random access point (IRAP) packets. These methods, however, do not take into account differences between frames of the same type as frames of the same type have different impacts on video quality. Therefore, more sophisticated methods are proposed.[3,4,6,7,8,11] Chakareski and Frossard[6] addressed the packet scheduling problem for multiple videos by characterizing video packets using rate-distortion information. However, they

aim to achieve fairness among multiple videos. Corbillon et al.[3] prioritize video packets using an evaluation function, taking into account frame type, dependency, and size. Li et al.[7] formulated the packet scheduling of traditional 2-D video at the transmitter as an optimization problem to minimize distortion with rate-distortion information, and solved it with a greedy algorithm. Nasralla et al.[8] proposed a content-aware packet scheduling method for video streaming. A utility function based on the temporal complexity and type of frames is proposed to prioritize packets. However, their work ignores the interdependencies of frames and the rate information. Chang et al.[4] developed a visibility model for B-frame loss to generate a visual score for each frame before video transmission. However, only B-frame is allowed to be dropped in their system.

Despite the rapid development of VR video, very few packet scheduling methods are proposed. Comşa et al.[9] proposed a packet scheduling method based on machine learning to allocate network resources for live VR video and other media applications instead of data reduction for a VR video. Chakareski[10] integrates content popularity and rate-distortion and base station information to generate packet scheduling for resource allocation. The system, however, cannot realize a user's viewport in practice because it uses only a frequency model to determine the popularity of VR content.

Table 1 compares the popular packet scheduling methods mentioned earlier for traditional and VR videos. Rather than reducing data volume in a network, existing VR video packet scheduling algorithms are mainly concerned with distributing resources among different traffic streams and multiple paths, which cannot cope with variations of limited bandwidth. Meanwhile, existing packet scheduling methods for VR video cannot take into account viewport importance in the transmission network. Hence, we propose a content-aware packet scheduling strategy to reduce data volume and quality distortion.

VIEWPOINT PREDICTION

Given a set of historical viewpoints $\{a_i\}_{i=1}^{A}$, where position $a_i \in \mathbb{R}^3$ and a sequence of video frames $\{f_j\}_{j=1}^{F}$ where frame $f_j \in \mathbb{R}^{H\times W\times C}$, and H, W, and C are the height, width, and channel number, respectively. We aim to predict a viewer's viewpoint trajectory $\{b_r\}_{r=1}^{B}$. As shown in Figure 2, the transformer contains a visual encoder, viewpoint encoder, and viewpoint decoder. The encoder–decoder architecture proposed in Vaswani et al.'s work[12] is used in our work. Each module is described next.

VISUAL ENCODER

Each frame is split into Z patches and patch $x_i \in \mathbb{R}^{h\times w}$ and $i = 1, \ldots, Z$, where h and w are height and width, respectively. Each patch is embedded into tokens with patch and positional embedding, same as the vision transformer.[13] The spatial transformer encoder only models interactions between tokens extracted from the same frame. The spatial attention score of patch x_i of frame f_j is given by

$$\text{Attention}_{\text{spatial}} = \text{softmax}\left(\frac{q_{(x_i,f_i)}}{\sqrt{d_k}} \cdot \{k_{(x_\alpha,f_j)}\}_{\alpha=1,\ldots,Z}^{T}\right) \quad (1)$$

where $q_{(x_i,f_j)}$ and $k_{(x_\alpha,f_j)}$ are the query vector of patch x_i and key vector of patch x_α in frame f_j, respectively. The d_k is the embedding dimension. A representation for each frame index h_i is obtained after the spatial encoder. The frame-level representations, $h_i \in \mathbb{R}^{d_k}$, are concatenated into $H \in \mathbb{R}^{F\times d_k}$ and then forwarded through a temporal encoder consisting of N transformer layers to model interactions between tokens from different temporal frames. The temporal attention score of frame f_j is given by

$$\text{Attention}_{\text{temporal}} = \text{softmax}\left(\frac{q_{(f_j)}}{\sqrt{d_k}} \cdot \{k_{(f_\beta)}\}_{\beta=1,\ldots,F}^{T}\right) \quad (2)$$

where $q_{(f_j)}$ and $k_{(f_\beta)}$ are the query vector of frame f_j and key vector of frame f_β, respectively. After being processed by the temporal encoder, the output tokens, $\{z_j\}_{j=1}^{F}$ and $z_j \in \mathbb{R}^{d_k}$, of this encoder are then obtained.

VIEWPOINT ENCODER

Here, the trajectory prediction is treated as a classification problem, rather than a regression problem, that predicts coordinates directly. The original position a_i is classified by the *k*-means clustering algorithm into different groups to get centroids. The centroid c_i is embedded into tokens with word embedding and position embedding. Similarly, the viewpoint encoder models the relationship between tokens extracted from the historical viewpoint trajectory. The output tokens, $\{y_i\}_{i=1}^{A}$, $y_i \in \mathbb{R}^{d_k}$, are obtained after the viewpoint encoder.

VIEWPOINT DECODER

The viewpoint decoder generates the viewpoint set $\{b_r\}_{r=1}^{B}$ by using the encoder embeddings, which are obtained by concatenating the visual and viewpoint tokens, $\{z_j\}_{j=1}^{F}$ and $\{y_i\}_{i=1}^{A}$, respectively. At each autoregressive step k, the viewpoint decoder relies on a causal transformer decoder, which cross-attends with the encoder outputs and self-attends with the tokens generated in previous steps to generate a representation. Then, the model predicts the next token in the trajectory with the representation.

TABLE 1. Comparison of the existing packet scheduling methods.

Citation	Video type	Resolution	Frame number[a]	FPS[b]	Formulated problem	Proposed method	Viewport aware in network
Moharrami et al.[5]	Traditional video	1280 × 720	—	30	Minimize quality degradation	Drop packets according to packet type	No
Gobatto et al.[2]	Traditional video	2560 × 1600	600	60	Minimize IRAP packet loss rate	Drop NIRAP packets	No
Chakareski and Frossard[6]	Traditional video	176 × 144	300	30	Maximize overall quality of multiple videos streamed over a limited bandwidth transmission channel with rate-distortion information	Subgradient method with Lagrangian relaxation used to compute Lagrange multiplier of nonconstrained problem	No
Corbillon et al.[3]	Traditional video	1920 × 1080	—	25	Minimize the degradation of video using an evaluation function considering frame type, dependencies, and size	Drop frames according to their importance obtained from evaluation function	No
Li et al.[7]	Traditional video	176 × 144	600	30	Minimize the distortion of video using rate distortion	The problem is solved with a greedy algorithm	No
Nasralla et al.[8]	Traditional video	640 × 416	100	25	Reduce packet delay and improve quality using a utility function based on temporal complexity and frame type	Drop packets according to packets priority obtained from utility function	No
Chang et al.[4]	Traditional video	720 × 480	36,000	30	Minimize visual score, indicating the visual impact of a frame loss	Drop B-frame according to the visual score	No
Comşa et al.[9]	VR video	—	—	—	Improve the fraction of time in the transmission-time interval by allocating the available frequency resources to different traffic classes	Reinforcement learning with continuous actor–critic learning automata algorithm	No
Chakareski[10]	VR video	3840 × 2048	450	30	Maximum VR video quality delivered from multiple base stations considering content popularity, rate distortion, and the information of base stations	The problem is solved with an approximate solution obtained using a faster iterative algorithm	No
Ours	VR video	3840 × 1920	500	25	Minimize distortion of the entire VR video and viewport over limited bandwidth transmission channel considering viewport and rate distortion	Dynamic programming containing state-transition equation and initial state to solve the optimization problem	Yes

[a]frame number: minimum total number of frames for a video
[b]fps: frames per second.

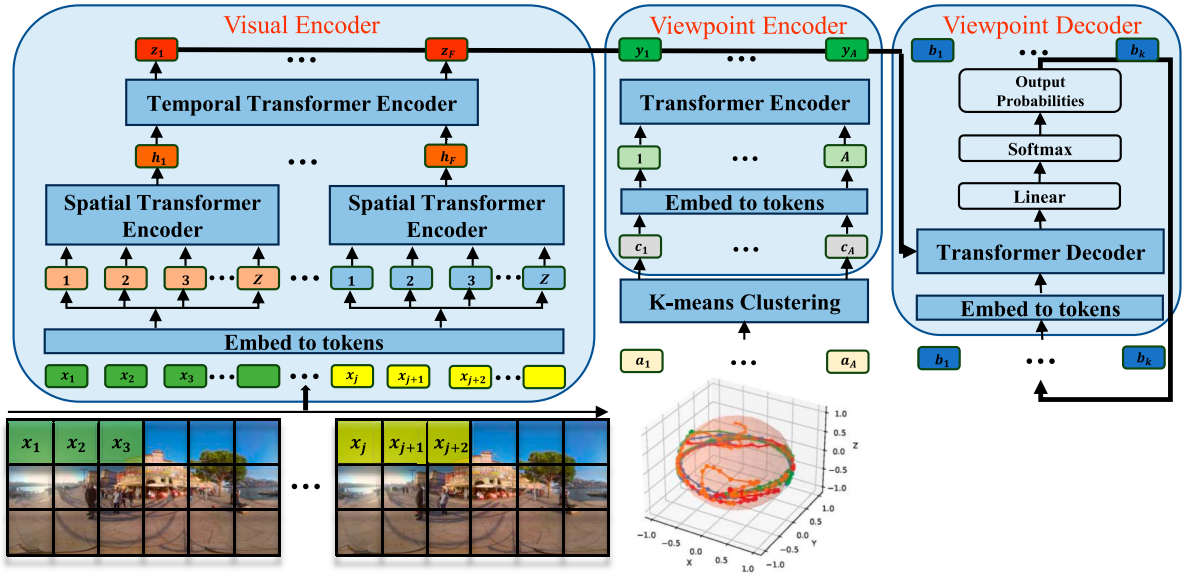

FIGURE 2. Architecture of the multimodal spatial–temporal attention transformer model for viewpoint prediction with classification method.

TILE RANKING

Existing methods[14] classify tiles based on visual overlap without considering viewpoint probability, resulting in constant weight and resource consumption despite varying viewpoint probabilities. Thus, we dynamically compute the tile's weight using perspective probability, and drop more packets inside the viewport when the probability is low and vice versa. We categorize the tiles into four classes based on "perceptual importance": class 4 for tiles entirely within the viewport (highest rank), class 3 for those in over half of the viewport, class 2 for tiles with less than half in the viewport, and class 1 for tiles outside the viewport. At time step t, we calculate the weight of each tile by

$$\lambda_t = \frac{E \cdot L}{M} \tag{3}$$

where E is the probability of prediction obtained from the transformer model. M is the number of classes, which is four in our work, and L is the class level, $L = 1, 2, 3, 4$. For example, as shown in Figure 1, the predicted viewpoint has a probability of 0.8. The weight of class-4 tiles is computed by $\lambda_t = \frac{0.8 \cdot 4}{4} = 0.8$, while the weight of class-3 tiles is $\lambda_t = \frac{0.8 \cdot 3}{4} = 0.6$.

PACKET SCHEDULING PROBLEM FORMULATION

Consider that a VR video is divided into t tiles $\boldsymbol{V} = \{V^0, V^1, ..., V^{t-1}\}$. All tiles have the same number of frames n. Therefore, the frame sequence of tile τ, $\tau = 0, 1, ..., t-1$, can be expressed by $V^\tau = \{p_0^\tau, p_1^\tau, ..., p_{n-1}^\tau\}$. The frame η, $\eta = 0, 1, ..., n-1$, of the tile τ can be defined as p_η^τ. Packet loss distortion is measured by using the structural similarity index measure (SSIM). We calculate distortion between two packets, p_i and p_j, with $d(p_i, p_j) = 1 - \text{SSIM}$.

For tile τ, a transmission subset with a length of l, $S^\tau = \{p_{s_0}^\tau, p_{s_1}^\tau, ..., p_{s_{l-1}}^\tau\}$, is selected from V^τ to transmit over the network channel. The reconstructed video sequence of V^τ because of S^τ is denoted as $C_{V^\tau}(S^\tau) = \{p_0^{\tau\prime}, p_1^{\tau\prime}, ..., p_{n-1}^{\tau\prime}\}$. Dropped frames are compensated for using the previous frame concealment. The $C_{V^\tau}(S^\tau)$ is constructed by replacing the dropped frame with the nearest previous neighboring frame in S^τ, which is

$$p_i^{\tau\prime} = p^\tau_{j=\max(s), s\in\{s_0, s_1, ..., s_{l-1}\}, j\le i}. \tag{4}$$

We assume that the distortions caused by multiple packet losses are additive.[7] The distortion $D(p_\eta^\tau)$ can be calculated by adding up the distortions between each frame and the corresponding reconstructed frame, which is

$$\begin{aligned} D(p_\eta^\tau) &= \sum_{i=0}^{n-1} d(p_i^\tau, p_i^{\tau\prime}) \\ &= d(p_\eta^\tau, p^\tau_{j=\max(s), s\in\{s_0, s_1, ..., s_{l-1}\}, j\le\eta}) + \sum_{i=0, i\neq\eta}^{n-1} d(p_i^\tau, p_i^{\tau\prime}). \end{aligned} \tag{5}$$

It shows that distortion of p_η^τ contains two parts. The first part indicates the distortion on p_η^τ itself,

without considering error propagation. Without considering error propagation, the first part can be written as

$$\sum_{i=0}^{n-1} d(p_i^\tau, p_i^{\tau\prime}) = \sum_{i=0}^{n-1} d(p_i^\tau, p^\tau_{j=\max(s), s\in\{s_0,s_1,\ldots,s_{l-1}\}, j\le i}) = d(p_\eta^\tau, p_{\eta-1}^\tau). \quad (6)$$

The second part is the sum of distortion on all frames except p_η^τ itself due to error propagation. We denote the second part as $\Omega(p_\eta^\tau)$. Note that $d(p_i^\tau, p_i^{\tau\prime}) = 0$ for $i < \eta$ as there is no previous distortion before the loss of packet p_η^τ. Therefore, (5) can be rewritten as

$$D(p_\eta^\tau) = \sum_{i=0}^{n-1} d(p_i^\tau, p^\tau_{j=\max(s), s\in\{s_0,s_1,\ldots,s_{l-1}\}, j\le i}) + \Omega(p_\eta^\tau). \quad (7)$$

Given V^τ and S^τ, a dropped set with length z, $K^\tau = \{p_{k_0}, p_{k_1}, \ldots, p_{k_{z-1}}\}$ is determined. Therefore, the total distortion of dropped set K^τ can be expressed by

$$\begin{aligned} D(K^\tau) &= \sum_{\eta\in K^\tau} D(p_\eta^\tau) \\ &= \sum_{\eta\in K^\tau}\sum_{i=0}^{n-1} d(p_i^\tau, p_i^{\tau\prime}) \\ &= \sum_{\eta\in K^\tau} d(p_\eta^\tau, p^\tau_{j=\max(s), s\in\{s_0,s_1,\ldots,s_{l-1}\}, j\le\eta}) + \sum_{\eta\in K^\tau}\Omega(p_\eta^\tau) \\ &= \sum_{i=0}^{n-1} d(p_i^\tau, p^\tau_{j=\max(s), s\in\{s_0,s_1,\ldots,s_{l-1}\}, j\le i}) + \Omega(K_\eta^\tau) \end{aligned} \quad (8)$$

Hence, the total weighted distortion over all the tiles incurred by $\boldsymbol{K} = \{K^0, K^1, \ldots, K^{t-1}\}$ can be calculated with

$$\begin{aligned} \tilde{D}(\mathbf{K}) &= \sum_{\tau=0}^{t-1}\lambda_\tau D(K^\tau) = \sum_{\tau=0}^{t-1}\sum_{\eta\in K^\tau}\lambda_\tau D(p_\eta^\tau) \\ &= \sum_{\tau=0}^{t-1}\sum_{\eta\in K^\tau}\Big\{\sum_{i=0}^{n-1}\lambda_\tau d(p_i^\tau, p^\tau_{j=\max(s), s\in\{s_0,s_1,\ldots,s_{l-1}\}, j\le i}) \\ &\quad + \lambda_\tau\Omega(p_\eta^\tau)\Big\} \\ &= \sum_{\tau=0}^{t-1}\sum_{\eta\in K^\tau}\Big\{\sum_{i=0}^{n-1}\tilde{d}(p_i^\tau, p^\tau_{j=\max(s), s\in\{s_0,s_1,\ldots,s_{l-1}\}, j\le i}) \\ &\quad + \tilde{\Omega}(p_\eta^\tau)\Big\} \\ &= \sum_{\tau=0}^{t-1}\sum_{\eta\in K^\tau}\tilde{D}(P_\eta^\tau) \end{aligned} \quad (9)$$

where λ_τ is the importance factor for the tile τ. We denote the weighted distortion of $\cdot$ as $\tilde{\cdot}$, e.g., $\lambda_\tau D(P_\eta^\tau)$ as $\tilde{D}(P_\eta^\tau)$.

Let $\omega = n \cdot t$, and the VR video has ω packets. The packet set P can be written as $P = \{p_0, p_1, \ldots, p_{\omega-1}\}$. Similarly, by assuming that the total number of dropped packets is f, the dropped set K can be expressed as $K = \{p_{k_0}, p_{k_1}, \ldots, p_{k_{f-1}}\}$. In the meantime, the transmission set S with length $g = \omega - f$ can be defined as $S = \{p_{s_0}, p_{s_1}, \ldots, p_{s_{g-1}}\}$ and $P = K \cup S$. Therefore, by referring to (8), (9) can be rewritten as

$$\begin{aligned} \tilde{D}(K) &= \sum_{i=0}^{f-1}\tilde{D}(p_{k_i}) \\ &= \sum_{i=0}^{w-1}\tilde{d}(p_i, p_{j=\max(s), s\in\{s_0,s_1,\ldots,s_{g-1}\}, j\le i}) + \tilde{\Omega}(K) \end{aligned} \quad (10)$$

Given available bandwidth, $R_{\max}$, the problem can be defined as

$$K^* = *\arg\min_K \tilde{D}(K), \ \text{subject to} \ R(S) \le R_{\max} \quad (11)$$

which means that the system needs to decide on a dropped set K (or a transmission set S) to meet the constrained bandwidth causing the minimum weighted distortion. We denote $K = P\backslash S$, where "\" means the "set difference," and $P\backslash S$ is a set consisting of the elements of P, which are not elements of S. The problem can be rewritten as

$$\begin{aligned} S^* &= \arg\min_S \tilde{D}(P\backslash S), \ \text{subject to} \ R(S) \le R_{\max} \\ &= \arg\min_S \Big\{\sum_{i=0}^{\omega-1}\tilde{d}(p_i, p_{j=\max(s), s\in\{s_0,s_1,\ldots,s_{g-1}\}, j\le i}) \\ &\quad + \tilde{\Omega}(P\backslash S)\Big\} \end{aligned} \quad (12)$$

DYNAMIC PROGRAMMING SOLUTION

We define $\tilde{D}(P\backslash S_m^{s_0:s_{m-1}})$ as the minimum weighted distortion caused by the selected transmission subset $S_m^{s_0:s_{m-1}} = \{p_{s_0}, p_{s_1}, \ldots, p_{s_{m-1}}\}$ that has m packets, starting with packet p_{s_0} and ending with packet $p_{s_{m-1}}$. According to (12), we have

$$\begin{aligned} \tilde{D}(P\backslash S_m^{s_0:s_{m-1}}) &= \min_{S_m^{s_0:s_{m-1}}}\Big\{\sum_{i=0}^{\omega-1}\tilde{d}(p_i, p_{j=\max(s), s\in S_m^{s_0:s_{m-1}}, j\le i}) \\ &\quad + \tilde{\Omega}(P\backslash S_m^{s_0:s_{m-1}})\Big\} \end{aligned} \quad (13)$$

As the first packet p_0 is very important (I-frame), it is always selected in set S, and so we have $s_0 = 0$. Let $e = s_{m-1}$. Therefore, $s_0 = 0$ and $s_{m-1} = e$ are removed from the optimization. Hence, (13) can be rewritten as

$$\begin{aligned} \tilde{D}(P\backslash S_m^{s_0:s_{m-1}}) &= \min_{S_m^{s_1:s_{m-2}}}\Big\{\sum_{i=0}^{\omega-1}\tilde{d}(p_i, p_{j=\max(s), s\in S_m^{0:s_{m-1}}, j\le i}) \\ &\quad + \tilde{\Omega}(P\backslash S_m^{s_0:s_{m-1}})\Big\} \end{aligned} \quad (14)$$

Similarly, we can define $\tilde{D}(P\backslash S_{m-1}^{s_0:s_{m-2}})$ as

$$\begin{aligned} \tilde{D}(P\backslash S_{m-1}^{s_0:s_{m-2}}) &= \min_{S_{m-1}^{s_1:s_{m-3}}}\Big\{\sum_{i=0}^{w-1}\tilde{d}(p_i, p_{j=\max(s), s\in S_{m-1}^{0:s_{m-2}}, j\le i}) \\ &\quad + \tilde{\Omega}(P\backslash S_{m-1}^{s_0:s_{m-2}})\Big\} \end{aligned} \quad (15)$$

As $0 < s_1 < s_2 < ... < s_{m-2} < e$ and $j \le i$, (14) can be expressed as

$$\begin{aligned}\tilde{D}(P\backslash S_m^{s_0:s_{m-1}}) = \min_{S_m^{s_1:s_{m-2}}} \Big\{ &\sum_{i=0}^{e-1} \tilde{d}(p_i, p_{j=\max(s), s\in S_{m-1}^{0:s_{m-2}}, j\le i}) \\ &+ \sum_{i=e}^{\omega-1} \tilde{d}(p_i, p_{j=\max(s), s\in S_{m-1}^{0:s_{m-2}}, j\le i}) \\ &- \sum_{i=e}^{\omega-1} \tilde{d}(p_i, p_{j=\max(s), s\in S_{m-1}^{0:s_{m-2}}, j\le i}) \\ &+ \tilde{\Omega}(P\backslash S_m^{s_0:s_{m-1}}) \Big\} \\ &+ \sum_{i=e}^{\omega-1} \tilde{d}(p_i, p_e). \end{aligned} \tag{16}$$

Because of $s_{m-2} < e$, we have

$$\sum_{i=e}^{\omega-1} \tilde{d}(p_i, p_{j=\max(s), s\in S_{m-1}^{0:s_{m-2}}, j\le i}) = \sum_{i=e}^{\omega-1} \tilde{d}(p_i, p_{s_{m-2}}). \tag{17}$$

Therefore, (16) can be rewritten as

$$\begin{aligned}\tilde{D}(P\backslash S_m^{s_0:s_{m-1}}) = \min_{S_m^{s_1:s_{m-2}}} \Big\{ &\sum_{i=0}^{\omega-1} \tilde{d}(p_i, p_{j=\max(s), s\in S_{m-1}^{0:s_{m-2}}, j\le i}) \\ &+ \tilde{\Omega}(P\backslash S_m^{s_0:s_{m-1}}) \\ &- \sum_{i=e}^{\omega-1} (\tilde{d}(p_i, p_{s_{m-2}}) - \tilde{d}(p_i, p_e)) \Big\}. \end{aligned} \tag{18}$$

As $\tilde{\Omega}(P\backslash S_m^{s_0:s_{m-1}}) = \tilde{\Omega}(P\backslash S_{m-1}^{s_0:s_{m-2}}) - \tilde{\Omega}(e)$, (18) can be expressed as

$$\begin{aligned}\tilde{D}(P\backslash S_m^{s_0:s_{m-1}}) = \min_{S_m^{s_1:s_{m-2}}} \Big\{ &\sum_{i=0}^{n-1} \tilde{D}(p_i, p_{j=\max(s), s\in S_{m-1}^{0:s_{m-2}}, j\le i}) \\ &+ \tilde{\Omega}(P\backslash S_{m-1}^{s_0:s_{m-2}}) - \tilde{\Omega}(e) \\ &- \sum_{i=e}^{\omega-1} (\tilde{d}(p_i, p_{s_{m-2}}) - \tilde{d}(p_i, p_e)) \Big\}. \end{aligned} \tag{19}$$

Let

$$\Phi(s_{m-2}, e) = \tilde{\Omega}(e) + \sum_{i=e}^{\omega-1} (\tilde{d}(p_i, p_{s_{m-2}}) - \tilde{d}(p_i, p_e)). \tag{20}$$

Equation (19) can be rewritten as

$$\begin{aligned}\tilde{D}(P\backslash S_m^{s_0:s_{m-1}}) = \min_{S_m^{s_1:s_{m-2}}} \Big\{ &\sum_{i=0}^{n-1} \tilde{d}(p_i, p_{j=\max(s), s\in S_{m-1}^{0:s_{m-2}}, j\le i}) \\ &+ \tilde{\Omega}(P\backslash S_{m-1}^{s_0:s_{m-2}}) - \Phi(s_{m-2}, e) \Big\}. \end{aligned} \tag{21}$$

By referring to (15), (21) can be rewritten as

$$\begin{aligned}&\tilde{D}(P\backslash S_m^{s_0:s_{m-1}}) = \\ &\min_{s_{m-2}} \Big\{ \min_{S_{m-1}^{s_1:s_{m-3}}} \Big\{ \sum_{i=0}^{\omega-1} \tilde{d}(p_i, p_{j=\max(s), s\in S_{m-1}^{0:s_{m-2}}, j\le i}) \Big\} \\ &+ \tilde{\Omega}(P\backslash S_{m-1}^{s_0:s_{m-2}}) - \Phi(s_{m-2}, e) \Big\} \\ &= \min_{s_{m-2}} \{ \tilde{D}(P\backslash S_{m-1}^{s_0:s_{m-2}}) - \Phi(s_{m-2}, e) \} \end{aligned} \tag{22}$$

where the first part represents the minimized distortion for the set $S_{m-1}^{s_0:s_{m-2}}$, while the second is the distortion reduction if packet e is selected by the set $S_{m-1}^{s_0:s_{m-2}}$ to generate the new set $S_m^{s_0:s_{m-1}}$. Equation (22) describes the state transition in dynamic programming. As the first frame is always selected by S, the initial state S_1^0 can be defined as

$$D(P\backslash S_1^0) = \sum_{e=1}^{m-1} d(p_0, p_e) + \Omega(P\backslash S_1^0). \tag{23}$$

With the initial state and state-transition equation, the system can compute the optimal packet scheduling scheme via a backtracking trellis diagram. As shown in the trellis diagram in Figure 1, the trellis node represents a transmission set that has a corresponding edge cost (distortion). For instance, the transmission set S_1^0 indicates that the current set has one packet and ends with packet 0 as packet 0 is always selected. $\Phi(S_1^0, 4)$ indicates the distortion cost when packet 4 is selected by set S_1^0 and generates a new set S_2^4. Meanwhile, the remaining bandwidth for the transmission set can be computed. If the value is zero, the transmission set is the final set leading to maximum distortion.

Suppose that there are n packets in a network node, the computational complexity of the proposed solution is $O(n^2)$. As a frame is decomposed into several (m) packets in the real world, the computational complexity is $O\left(\left(\frac{n}{m}\right)^2\right)$.

PERFORMANCE OF VIEWPOINT PREDICTION MODEL

Implementation Details

The dataset used in our work is MMSys18,[15] which contains 1083 viewpoint trajectories obtained from 57 participants with 19 4 K videos. The dataset is randomly split into training and testing sets, where the training set contains trajectories from the intersection of 70% of videos (13 videos) and 50% of users (24 users) by following the configuration in Track.[16] The videos are split into 24 tiles. The viewport is defined as a $120^\circ \times 120^\circ$ area. The time window of the input is set to 1 s. The model is tested with different output time windows from 1 to 5 s. Nonetheless, our system uses the 1-s future output. Five points and frames are sampled every second. Therefore, the model input has a sequence of five points and a sequence of five frames. The visual encoder has four temporal and spatial encoder blocks while the viewpoint encoder has two transformer encoder blocks. The viewpoint decoder has two decoder blocks. The model uses 12-multihead attention and an embedding dimension of 768. The

batch size is 100. The learning rate is 0.0005, with a decay rate of 0.99.

RESULTS AND ANALYSIS

In our experiments, average great-circle distance is used as the evaluation metric, which is the shortest distance between the ground-truth point and the predicted point on the surface of a sphere. The proposed method is compared with three methods on the MMSys18 dataset, as shown in Table 2. The baseline method uses the last input element as the output trajectory. The VPT360[17] only uses the encoder of the transformer. Track[16] is based on long short-term memory. The results of VPT360 and Track are obtained from the article and the pretrained model, respectively.

The bold numbers are the best scores. It can be seen that the average distances of all methods increase as the prediction time window increases. The proposed transformer performs best compared to other methods across different time windows.

EVALUATION OF PACKET SCHEDULING STRATEGY

Implementation Details

Four existing methods are compared with our method: baseline, which is the tail drop algorithm ignoring the difference between packets; equal-weighted rate-distortion (EWRD)[11] dropping packets based on real distortion by considering tiles equally; non-intra random access point (NIRAP)[2] preemptively dropping non-IRAP packets; and IPB[5] dropping packets based on frame types. The performances of all methods are evaluated on five metrics: total distortion, viewport distortion, total packet loss, viewport packet loss, and viewport bandwidth consumption. All the videos (six) of the testing set from MMSys18 are evaluated. All the methods are evaluated on two network trace scenarios: constant bandwidth trace (0.5–30 Mbps) and real-world trace. A 4G LTE dataset with throughput ranging from 0 to 173 Mbps[18] is used for real-world testing. The mean and standard deviation (STD) are presented in both scenarios.

CONSTANT BANDWIDTH SCENARIO

Distortion Analysis

The total distortions with different packet scheduling strategies across various bandwidths are shown in Figure 3(a). Distortion values are normalized and scaled to a range of 0–100 for comparison. The baseline, NIRAP, and IPB methods have similar distortion, while TWRD and EWRD perform better and have similar distortion. There is not much difference between TWRD and EWRD, while EWRD outperforms the others. The difference comes from the fact that EWRD uses knowledge about how dropping packets affect reconstructed video quality, while TWRD relies on weighted distortion. Thus, EWRD drops packets that have the least effect on reconstructed videos. Meanwhile, the distortions of all methods decrease as the bandwidth increases.

TWRD performs best on viewport distortion throughout the range, as shown in Figure 3(b), and EWRD achieves the second-best performance, followed by baseline, IPB, and NIRAP. For example, at 0.5-Mbps bandwidth, the improvements of TWRD are 28.5, 28.4, 27.8, and 6.4%, over baseline, NIRAP, IPB, and EWRD, respectively. As the bandwidth varies, improvement varies. TWRD exhibits average enhancements of approximately 74, 76, 70, and 35% when compared with baseline, NIRAP, IPB, and EWRD, respectively, across all bandwidths. Meanwhile, the average STDs across all bandwidths are 10.5, 13.3, 10.2, 7.6, and 4.5% for baseline, NIRAP, IPB, EWRD, and TWRD, respectively. Therefore, the quality of the viewport is greatly improved with the proposed approach compared with the other methods.

Packet Loss Analysis

The total and viewport packet loss rates for all strategies are shown in Figure 3(c) and (d). The total packet loss rate denotes the percentage of lost packets

TABLE 2. Comparison of viewpoint predictions for average great circle distance on MMSys18.

Method	Prediction time window				
	First s*	Second s	Third s	Fourth s	Fifth s
Baseline	0.321	0.52	0.672	0.788	0.878
VPT360[17]	0.215	0.399	0.55	0.667	0.754
Track[16]	0.281	0.464	0.609	0.723	0.809
Ours	**0.212**	**0.386**	**0.531**	**0.643**	**0.723**

*"s" indicates second.

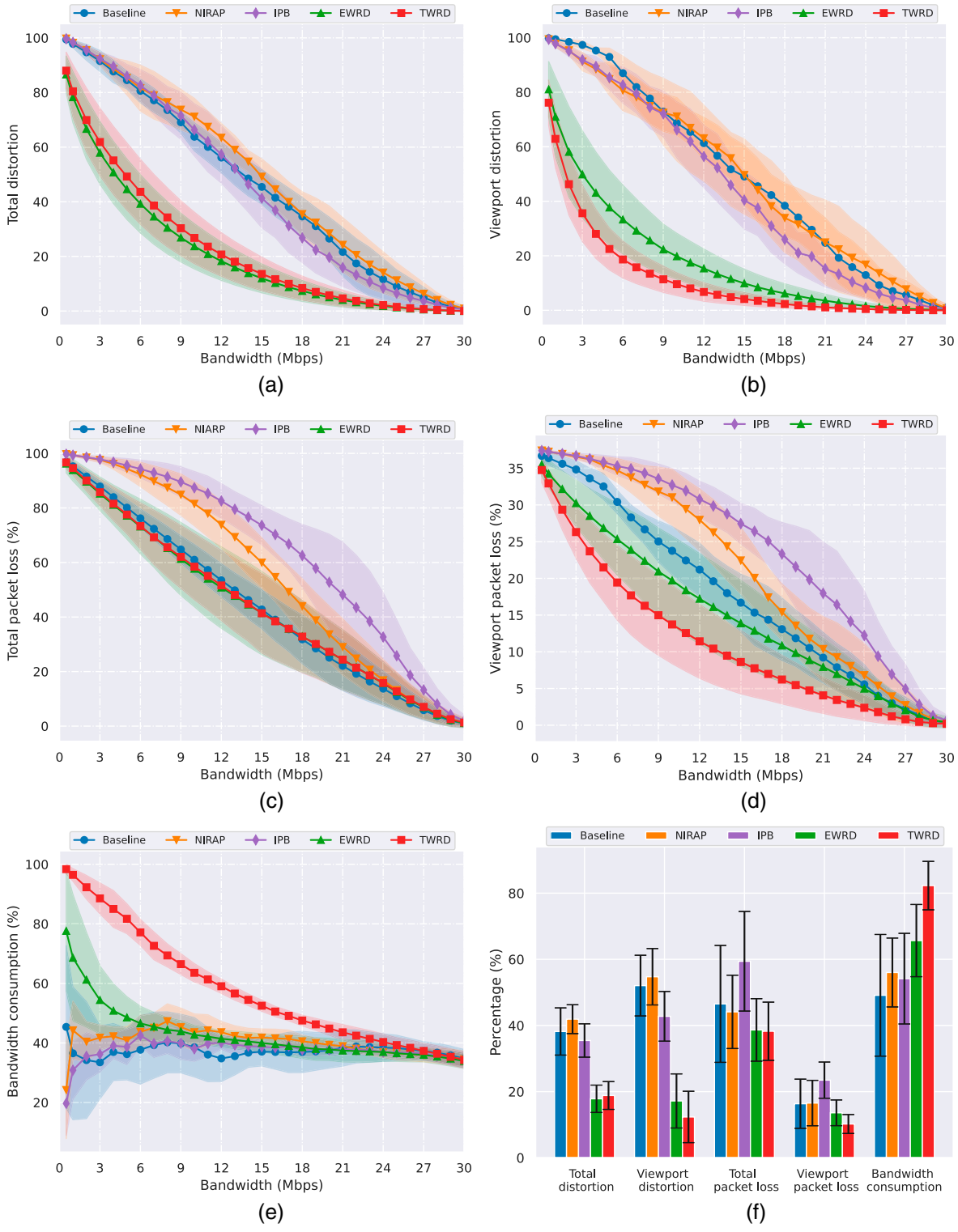

FIGURE 3. Experiment results for the five methods under different bandwidth scenarios. Constant bandwidth scenario: (a) Total distortion. (b) Viewport distortion. (c) Total packet loss. (d) Viewport packet loss. (e) Bandwidth consumption. Real-world trace scenario: (f) experiment results on the five metrics under the 4G LTE dataset.

relative to the total transmitted packets. Simultaneously, the viewport's packet loss rate indicates the percentage of lost packets in the viewport area compared with the total packets, which degrades video quality. Given that the viewport consists of a maximum of nine tiles, representing 37.5% of all tiles, the highest possible viewport packet loss rate is 37.5%. The total and viewport packet loss rates decrease as the bandwidth increases. As packets have different distortions, the loss of a small number of packets can cause the same or greater distortion than the loss of a large number of packets. Among the various methods, IPB exhibits the highest overall packet loss rate. As the B-type packets usually have small sizes, more B-type packets are dropped to meet the bandwidth constraint. Although the total packet loss of NIRAP is lower than IPB, NIRAP drops P-type and B-type packets, which causes the highest total distortion. TWRD avoids dropping packets in the viewport, which sometimes causes a higher total packet loss rate.

Although baseline, EWRD, and TWRD have similar total packet loss rates across all bandwidths, their viewport packet loss rates present huge differences, as shown in Figure 3(d). Noticeably, our method still performs well in terms of viewport packet loss, even under severely limited bandwidth conditions, such as 0.5 Mbps, where a majority of packets are dropped. The improvements of TWRD over EWRD, IPB, NIRAP, and baseline are approximately 2.1, 7.1, 7.1, and 4.8%, respectively, at a 0.5-Mbps bandwidth. As viewport packets are weighted and rarely dropped, TWRD always has the lowest viewport packet loss across bandwidths.

Bandwidth Consumption Analysis

As the simulated system has limited bandwidth without background traffic, the total bandwidth consumption for the entire VR video is the bandwidth itself. The percentage of viewport bandwidth consumption indicates how much bandwidth resource is allocated to the viewport. The higher the value, the better the method. The viewport bandwidth consumption is visualized, as shown in Figure 3(e). TWRD consumes the most bandwidth, followed by EWRD. Baseline, NIRAP, and IPB have similar bandwidth consumption rates. When the bandwidth is very limited, for instance, 0.5 Mbps, most of the packets, including those inside the viewport, are dropped. Therefore, baseline, NIRAP, and IPB have few resources allocated for the viewport, while TWRD prioritizes saving packets within the viewport, with TWRD achieving nearly 100% bandwidth utilization. Because the bandwidth resource required by the viewport is constant, the percentages of TWRD and EWRD decrease gradually with bandwidth increase. Noticeably, the values of baseline, NIRAP, and IPB fluctuate at roughly 40% over bandwidths as packets are dropped randomly, and each tile has the same probability of being dropped. The difference between all methods gradually decreases with bandwidth until sufficient bandwidth is provided.

Real-World Trace Scenario

The means and STDs across all videos and network traces are shown in Figure 3(f). TWRD performs best on four metrics except for total distortion, followed by EWRD, while baseline, NIRAP, and IPB have poor performance. For example, baseline, NIRAP, IPB, EWRD, and TWRD exhibit viewport distortions of 0.52, 0.55, 0.43, 0.17, and 0.12, respectively, accompanied by STDs of 9.2, 8.5, 7.5, 8.2, and 7.8. Additionally, the viewport packet loss rates for them are 16.3, 16.5, 23.5, 13.6, and 10.21 with STDs of 7.4, 6.9, 5.5, 3.9, and 2.9, respectively. TWRD's advantages and robustness are demonstrated by the experiment results on constant bandwidths and real-world traces.

Case Study Analysis

We also randomly choose two videos, Warship and PortoRiverside, and give detailed values for both videos in Table 3 to provide an in-depth analysis. The two videos are analyzed with the five metrics under different bandwidths (0.5, 5, 10, 15, 20, 25, and 30 Mbps). The values are listed in order: baseline, NIRAP, IPB, EWRD, and TWRD for every metric. Both videos present similar trends with different values due to the difference between the videos and the randomness of baseline, NIRAP, and IPB. For both videos, EWRD has the smallest overall distortion, and it is close to the values of TWRD. The proposed method still performs better than the others regarding viewport distortion. For the viewport packet loss rate, the proposed method has the lowest loss rate across bandwidths, which means that TWRD drops more packets that are beyond the viewport. In terms of viewport bandwidth consumption, TWRD always allocates high-bandwidth resources to its viewport.

Runtime Analysis

All the experiments are conducted using a testbed built on a computer with an Intel Core i9-11900F CPU and a Nvidia GTX3090 GPU. The transformer model has a runtime of roughly 63.1 ms for the 1-s predicted window, indicating that the runtime is acceptable.

Besides the transformer, the dynamic programming solution is activated only if the queue exceeds a threshold, which is 70% of the buffer size in our work. The average runtime of the dynamic programming is approximately 78.2 ms. However, as the packet scheduling is

TABLE 3. Case analysis for the five methods on the two videos: PortoRiverside and Warship.

Video	PortoRiverside							Warship						
Bandwidth	**0.5 Mbps**	**5 Mbps**	**10 Mbps**	**15 Mbps**	**20 Mbps**	**25 Mbps**	**30 Mbps**	**0.5 Mbps**	**5 Mbps**	**10 Mbps**	**15 Mbps**	**20 Mbps**	**25 Mbps**	**30 Mbps**
Total distortion	941	730.9	617.5	404.4	191.6	27	0	1143.9	1021	637.4	501.6	259.3	119	12.6
	954.6	800.4	722.5	546	379.3	81.2	0	1144.2	1019.7	883.8	714.1	440.2	149.5	19
	951.5	824.8	659	505.3	238.6	33.2	0	1142.9	1037.5	854.3	564.3	347.8	106	11.9
	786.2	**383.6**	**209.2**	**112.9**	**44**	**5.6**	0	**940.2**	**529**	**322.4**	**185.4**	**93**	**31.7**	**2.9**
	800.9	422.1	233.1	125.3	51.9	6.7	0	942.4	563.8	357.2	207.4	106.6	38	3.6
Viewport distortion	447.7	342.3	170.3	142	115	30.6	0	1273.2	1202.7	268.5	235.8	69	19.3	1.7
	454.8	369.5	312.4	229.5	67.1	6.2	0	635.2	553.6	492.9	432.9	243.5	27.8	2.4
	455.2	403.8	345.5	233.6	117.7	18.4	0	635.4	583.8	469.3	300.9	208.8	49.6	4.5
	328.7	125	66.4	31.9	17.3	2.8	0	429.8	165.6	96.2	59.5	31.4	13.9	1.6
	313.8	**75.1**	**28.9**	**10.3**	**4.4**	**0.6**	0	**429.8**	**105.4**	**43.3**	**23**	**9.8**	**3**	**0.3**
Total packet loss	**95.5**%	76.7%	59.7%	39.7%	22.9%	6.6%	0%	99.1%	98.3%	63.4%	**45.6**%	**29.6%**	**13.6%**	1.5%
	99.9%	96.7%	92.4%	76.2%	38.1%	7.5%	0%	99.8%	96.2%	86.2%	61.5%	38.6%	17.9%	2.1%
	99.9%	97.9%	95.1%	85.5%	66.6%	24%	0%	99.7%	96.9%	91.4%	77.6%	67.1%	41.2%	4.9%
	96%	**65.8**%	**41.1%**	**29.5%**	20.7%	**5.8%**	0%	**98.1**%	82.1%	65.4%	49.5%	34.2%	17.1%	**1.4%**
	95.7%	68.6%	45.2%	29.7%	**18.9%**	6.2%	0%	98.2%	**79.7**%	**63.4%**	48.8%	34.5%	18.9%	2.6%
Viewport packet loss	35.1%	25%	25.1%	17.2%	7.2%	3%	0%	36.6%	35.4%	20.2%	15.8%	8.5%	2%	0.1%
	37.5%	36%	35%	30.9%	19.5%	4.3%	0%	37.3%	36.1%	33.3%	23.1%	11.2%	2.2%	0.2%
	37.5%	36.7%	35.8%	32.5%	25%	8.9%	0%	37.4%	36.3%	34.2%	29.2%	25.5%	15.6%	1.6%
	35.5%	22.8%	14.3%	10.1%	7.4%	2%	0%	**35.8**%	26%	19.7%	14.2%	9.3%	4.3%	0.2%
	33.7%	**16.9**%	**9.6%**	**6%**	**3.6%**	**1%**	0%	**35.8**%	**19.6**%	**11.4%**	**7.2%**	**3.2%**	**0.8%**	**0%**
Bandwidth consumption	49.3%	47.5%	30%	30.3%	37.1%	33.8%	29.7%	70.6%	33.5%	42.9%	33.5%	33.7%	34.7%	30.1%
	17.5%	46.7%	28.2%	34%	31.4%	33.1%	29.7%	11.2%	43.5%	43.2%	39.5%	40.7%	40.8%	35.7%
	13.4%	29.3%	35.2%	35%	35.5%	34.1%	29.7%	16.7%	44.9%	46.4%	41.7%	38.7%	38.1%	35.4%
	73.1%	47.7%	40.6%	38.3%	33.7%	33.6%	29.7%	96.7%	59.3%	46.1%	39.5%	36.9%	35.1%	34.8%
	100%	**78.2**%	**61.6%**	**51%**	**41.4**%	**35.1**%	29.7%	**99.5**%	**89.1**%	**68.2%**	**53.9%**	**46.4%**	**40.8%**	**35.7%**

Note that the values for the five methods are listed in the following order for every metric: baseline, NIRAP, IPB, EWRD, and TWRD.

activated only if the queue exceeds a threshold, it has a limited impact on system delay. Meanwhile, the method drops packets in advance to avoid congestion, which can also reduce system latency. The runtime of our system on the real-world trace dataset is approximately 86.5 ms.

CONCLUSION

In this article, we proposed a TWRD packet scheduling system for VR video streaming based on a multimodal spatial–temporal attention transformer. The problem of determining the optimal packet scheduling scheme was modeled as an optimization problem and solved with the proposed dynamic programming-based solution. The experiment results show that the proposed method reduces the viewport distortion and achieves better performance than the other methods.

Furthermore, our approach can be improved in the future. In practice, our packet scheduling method may introduce a time cost during video streaming. Therefore, future work should focus on optimizing temporal cost reduction to ensure efficient and seamless video delivery to users. Additionally, as the proposed method reduces quality distortion at the network layer, it can be integrated with various application-layer strategies (e.g., rate adaptation, scalable coding, or re-encoding) to further enhance user experience. For instance, scalable coding can be adopted to save bandwidth by buffering the base layer for a long period of time and fetching enhancement layers for a shorter period to increase the quality of viewport.[19] Simultaneously, frame-level coding and streaming can be implemented, where coding is concentrated on the viewport, and other regions are encoded progressively, facilitating the video's periodic refreshment.[20]

REFERENCES

1. R. Tan et al., "Xsickness in intelligent mobile spaces and metaverses," *IEEE Intell. Syst.*, vol. 37, no. 5, pp. 86–94, Sep./Oct. 2022, doi: 10.1109/MIS.2022.3208485.
2. L. Gobatto, M. Saquetti, C. Diniz, B. Zatt, W. Cordeiro, and J. R. Azambuja, "Improving content-aware video streaming in congested networks with in-network computing," in *Proc. IEEE Int. Symp. Circuits Syst.*, 2022, pp. 1813–1817, doi: 10.1109/ISCAS48785.2022.9937451.
3. 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 *Proc. IEEE Int. Conf. Multimedia Expo Workshops*, 2016, pp. 1–6, doi: 10.1109/ICMEW.2016.7574700.
4. 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 Trans. Image Process.*, vol. 21, no. 8, pp. 3353–3363, Aug. 2012, doi: 10.1109/TIP.2012.2191567.
5. A. Moharrami, M. Ghasempour, and M. Ghanbari, "A smart packet type identification scheme for selective discard of video packets," *e-Prime—Adv. Elect. Eng., Electron. Energy*, vol. 4, Jun. 2023, Art. no. 100149, doi: 10.1016/j.prime.2023.100149.
6. J. Chakareski and P. Frossard, "Rate-distortion optimized distributed packet scheduling of multiple video streams over shared communication resources," *IEEE Trans. Multimedia*, vol. 8, no. 2, pp. 207–218, Apr. 2006, doi: 10.1109/TMM.2005.864284.
7. Y. Li, A. Markopoulou, J. Apostolopoulos, and N. Bambos, "Content-aware playout and packet scheduling for video streaming over wireless links," *IEEE Trans. Multimedia*, vol. 10, no. 5, pp. 885–895, Aug. 2008, doi: 10.1109/TMM.2008.922860.
8. 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," *Comput. Netw.*, vol. 140, pp. 126–137, Jul. 2018, doi: 10.1016/j.comnet.2018.05.014.
9. 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 Trans. Broadcast.*, vol. 67, no. 1, pp. 212–224, Mar. 2021, doi: 10.1109/TBC.2020.2983298.
10. J. Chakareski, "Viewport-adaptive scalable multi-user virtual reality mobile-edge streaming," *IEEE Trans. Image Process.*, vol. 29, pp. 6330–6342, 2020, doi: 10.1109/TIP.2020.2986547.
11. A. Abdelhadi, A. Gerstlauer, and S. Vishwanath, "Real-time rate distortion optimized and adaptive low complexity algorithms for video streaming," in *Proc. IEEE Int. Syst. Conf.*, 2019, pp. 1–8, doi: 10.1109/SYSCON.2019.8836788.
12. A. Vaswani et al., "Attention is all you need," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 6000–6010.
13. A. Dosovitskiy et al., "An image is worth 16x16 words: Transformers for image recognition at scale," in *Proc. Int. Conf. Learn. Representations*, 2021, pp. 1–21.
14. Y. Zhang et al., "EPASS360: QoE-aware 360-degree video streaming over mobile devices," *IEEE Trans. Mobile Comput.*, vol. 20, no. 7, pp. 2338–2353, Jul. 2021, doi: 10.1109/TMC.2020.2978187.
15. 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 *Proc. 9th ACM Multimedia Syst. Conf.*, 2018, pp. 432–437, doi: 10.1145/3204949.3208139.

16. 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 Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 9, pp. 5681–5699, Sep. 2022, doi: 10.1109/TPAMI.2021.3070520.
17. F.-Y. Chao, C. Ozcinar, and A. Smolic, "Transformer-based long-term viewport prediction in 360° video: Scanpath is all you need," in *Proc. IEEE 23rd Int. Workshop Multimedia Signal Process.*, 2021, pp. 1–6.
18. D. Raca, J. J. Quinlan, A. H. Zahran, and C. J. Sreenan, "Beyond throughput: A 4G LTE dataset with channel and context metrics," in *Proc. ACM Multimedia Syst. Conf.*, 2018, pp. 460–465.
19. A. T. Nasrabadi, A. Mahzari, J. D. Beshay, and R. Prakash, "Adaptive 360-degree video streaming using scalable video coding," in *Proc. ACM Int. Conf. Multimedia*, 2017, pp. 1689–1697, doi: 10.1145/3123266.3123414.
20. Y. Mao, L. Sun, Y. Liu, and Y. Wang, "Low-latency FoV-adaptive coding and streaming for interactive 360° video streaming," in *Proc. ACM Int. Conf. Multimedia*, 2020, pp. 3696–3704, doi: 10.1145/3394171.3413751.

HAOPENG WANG is a Ph.D. candidate with the School of Electrical Engineering and Computer Science, University of Ottawa, Ottawa, ON, K1N 6N5, Canada. His research interests include multimedia, extended reality, and artificial intelligence. Wang received his M.Sc. degree in electronic and communication engineering from the Beijing Institute of Technology. Contact him at hwang266@uottawa.ca.

HAIWEI DONG is a principal researcher at Huawei Canada and an adjunct professor at the University of Ottawa, Ottawa, ON, K2K 3J1, Canada. His research interests include artificial intelligence, multimedia, metaverse, and robotics. Dong received his Ph.D. degree from Kobe University. He is a Senior Member of IEEE. Contact him at haiwei.dong@ieee.org.

ABDULMOTALEB EL SADDIK is a distinguished professor at the University of Ottawa MBZUAI, Ottawa, ON, K1N 6N5, Canada and MBZUAI, Abu Dhabi, UAE. His research interests include the establishment of digital twins to facilitate the well-being of citizens using artificial intelligence, the Internet of Things and augmented reality/virtual reality to allow people to interact in real time with one another as well as with their smart digital representations. He is a Fellow of IEEE, the Royal Society of Canada, Engineering Institute of Canada, and Canadian Academy of Engineers and an Association for Computing Machinery distinguished scientist. Contact him at elsaddik@uottawa.ca.