Efficient Short-form Video Streaming: An Integration of Dynamic Bitrate Adaptation and Predictive Segment Preloading

Nguyen Hong Lich[1,3], Truong Thu Huong[1], Nguyen Viet Hung[1,2], Pham Ngoc Nam[3,*]
[1]*School of Electrical and Electronic Engineering, Hanoi University of Science and Technology, Hanoi, Vietnam*
[2]*Faculty of Information Technology, East Asia University of Technology, Bac ninh, Vietnam*
[3]*College of Engineering and Computer Science, VinUniversity, Hanoi, Vietnam*
[*]Corresponding email: nam.pn@vinuni.edu.vn

***Abstract*—Nowadays, short-form videos have become popular, particularly among mobile users. However, unlike traditional videos, users can flexibly engage in brief viewing sessions and effortlessly skip content that fails to captivate their interest. This user behavior causes significant data wastage during short-form video streaming. From another perspective, fluctuations in network conditions during streaming pose additional challenges in providing a good preloading strategy to meet users' quality of experience (QoE). In this paper, we develop a new solution to improve the quality of user experience (QoE). Our scheme adapts to fluctuating network conditions, varying available bitrates, and user behavior patterns to optimize preloading strategies. Furthermore, our solution automatically adjusts the video downloading speed to ensure the appropriate amount of data downloaded for each video. Experiments show that our method improves QoE by ranging from 10% to 25% compared to reference methods.**

***Index Terms*—Short-form video, QoE, bitrate adaptation, preloading**

I. Introduction

Short-form videos have experienced a remarkable surge in popularity, spearheaded by platforms such as TikTok. Forecasts suggest that TikTok's reach may expand to a staggering 1.8 billion users by the end of 2024 [1]

Short-form videos are commonly viewed on mobile devices and often in full-screen mode. Users frequently navigates those videos by scrolling up or down, moving back and forth through suggested playlists. Users of such short video services have minimal attention, with up to 70% skipping and not watching the entire video content [2]. Therefore, employing adaptive strategies becomes imperative to mitigate resource wastage.

From another perspective, these videos are stored online and divided into multiple segments playing simultaneously. In such a system, each video is encoded into multiple versions with different quality levels to adapt to varying network conditions. Thus, to optimize user experience while efficiently managing data usage during short-form video streaming over time, a mobile user agent is developed to determine which segments will be downloaded and at what quality. It faces the challenge of preloading sufficient data for smooth playback while minimizing the cache size and re-buffering, especially during sudden bandwidth drops [3]. While large buffers can enhance QoE [4], [5], [6], they can also lead to data wastage when bandwidth drops suddenly. Additionally, considering users' habits of stopping or skipping early, excessively large buffers contribute to data wastage, as preloaded content in the buffer may go unused [7].

Numerous studies focusing on optimizing short-form video streaming through dynamic bitrate adaptation and predictive segment preloading are currently facing the following problems: Firstly, network variability leads to fluctuations in connection quality, thereby requiring dynamic adaptation of the video stream's bitrate to ensure playback smoother without lag. Secondly, the whole process of dynamic bitrate adaptation and segment preloading requires rapid decision-making and system response time to control streaming sessions in real time effectively.

Therefore, in this paper, we propose an efficient short-form video streaming solution integrated with adaptive bitrate adaptation and a predictive segment preloading scheme based on user behaviors.

Our strategy involves downloading each video segment and selecting an appropriate timing and versions to be downloaded that aims to optimize user experience while minimizing data wastage. These downloaded segments do not necessarily belong to the video the user is currently watching but can be from other videos. This approach aids users in managing subsequent videos promptly as segments of those videos may already have been downloaded.

The contributions of this study can be summarized as follows:

- Improve User Experience beyond the reference methods by expanding options of bitrates (i.e. offering 4 bitrate versions instead of 3), so as viewers have a better chance of finding a bitrate that matches their available bandwidth, leading to fewer buffering events and a smoother viewing experience.
- Increase time efficiency in finding the optimal solution among a larger number of options above by employing Dynamic programming. By breaking down the problem into smaller subproblems and solving these subproblems recursively and avoiding redundant calculations by stor-

ing intermediate results in a table or cache, dynamic programming ensures that the best combination of options is identified without redundantly re-evaluating the same subproblems. This greatly reduces the time complexity compared to the reference approaches, making it feasible to explore a larger space of options efficiently. Thus, the solution can be feasibly implemented for real-time video streaming system.

- Propose more accurate bandwidth estimations to enhance the Quality of Experience (QoE) for viewers and optimize bandwidth utilization across a spectrum of network bandwidth conditions. These estimations should account for both relatively stable variations and significant fluctuations in bandwidth. The methodology to accurately estimate varying bandwidth conditions involves employing Moving Average Convergence Divergence (MACD).

The remaining of the paper is as follows. Section II gives an overview of the related work. The proposed method is described in Section III, followed by an evaluation in Section IV. Finally, conclusions of achievement, challenges and future work are discussed in Section V.

II. Related Work

Some studies [2], [8] have predominantly concentrated on supporting a single quality level per video, which has left room for improvement in terms of adaptive encoding strategies. Recently, [9] proposed a more comprehensive approach to quality-level support, addressing this limitation. This implies that previous studies primarily emphasized a singular level of quality, while our method expands upon this by incorporating multiple quality levels. Furthermore, it is noteworthy that studies like those conducted by [2], [8] primarily focus on aspects such as data wastage due to their support for only one quality level.

The studies [7], [10], [11] focused on segment calculation and pre-downloading strategies based on user behavior factors. While these approaches implement partial loading strategies considering factors such as user viewing time, prediction, and network throughput conditions, they lack automatic adaptation capabilities. Instead, they rely on predefined constraints set by developers. However, this reliance on predefined rules may limit their ability to optimize performance dynamically. As a result, these methods may not effectively respond to changes in network conditions or user behaviors in real time, potentially leading to suboptimal video streaming experiences such as increased buffering or lower video quality, particularly in dynamic network environments.

In the realm of enhancing short video streaming experiences by automatically optimize preloading strategies and bitrate selection, notable solutions include **Next-One** [12], **JPBA** [13] and **Fixed-preload** [14]. Next-One, which focuses on preloading only the current and subsequent videos, defers the preloading of the next video until the current one is fully downloaded. This approach, reminiscent of the strategy employed by the Douyin application, aims to minimize buffering interruptions. On the other hand, JPBA introduces a dynamic buffer size determined by the estimated harmonic mean of bandwidth and user retention rate. Through the use of Model Predictive Control (MPC), JPBA calculates adaptive bitrate levels for each chunk within a pre-defined horizon window, aiming to optimize the user experience. However, the computational complexity of JPBA may pose challenges, particularly with increasing server scale. Finally, Fixed-preload, serving as a baseline algorithm, adjusts preloading based on the current probability of retaining a user, facilitating the preloading of up to four chunks. Despite the advancements made by existing methods, there remains opportunity for further enhancements in terms of balancing among Quality of Experience (QoE), runtime efficiency, and mitigating data wastage.

Streaming short-form videos on online platforms requires high streaming performance to ensure good video quality and a smooth user experience. However, meeting the huge demand and ensuring system scalability remains a challenge. Therefore, in this paper, we design a system to improve user experience performance by using the Model Predictive Control (MPC) method, which aims to determine the quality level, maximizing the user's QoE while balancing user QoE and data wastage.

III. Proposed Method

Our proposed system architecture is presented in Fig.1 in which a MACD model is deployed for bandwidth estimation, followed by buffer size adjustment and bit rate adaptation. It will improving user experience QoE and reducing waste for short-form video streaming.

A. MACD-BASED BANDWIDTH ESTIMATION

The Moving Average Convergence/Divergence (MACD) [15], famous for its effectiveness in data analysis, is a valuable tool for bandwidth estimation. Its ability to detect trend shifts and momentum changes makes it ideal for predicting future bandwidth trends and enhancing decision-making in adaptive streaming systems. By utilizing MACD's insights into short-term and long-term moving averages, bandwidth estimation can achieve more precise predictions. Additionally, MACD employs different motion bias determinations for current and subsequent videos, calculating the move using the moving average of the current and following videos to determine the optimal bit rate level for downloading the next segment. MACD is derived from subtracting two Exponential Moving Averages (EMAs), one for a shorter period and the other for a longer period. Its calculation can be expressed as follows:

$$\text{MACD} = \text{EMA}_{\text{short}} - \text{EMA}_{\text{long}} \tag{1}$$

$$\text{EMA}_t = \frac{\text{EMA}_{t-1} \times (1-\alpha) + \text{bandwitdth}_t}{\text{w}_t} \tag{2}$$

$$\alpha = \frac{2}{\text{window} + 1} \tag{3}$$

$$\text{w}_t = \text{w}_{t-1} + (1-\alpha)^t \tag{4}$$

where:

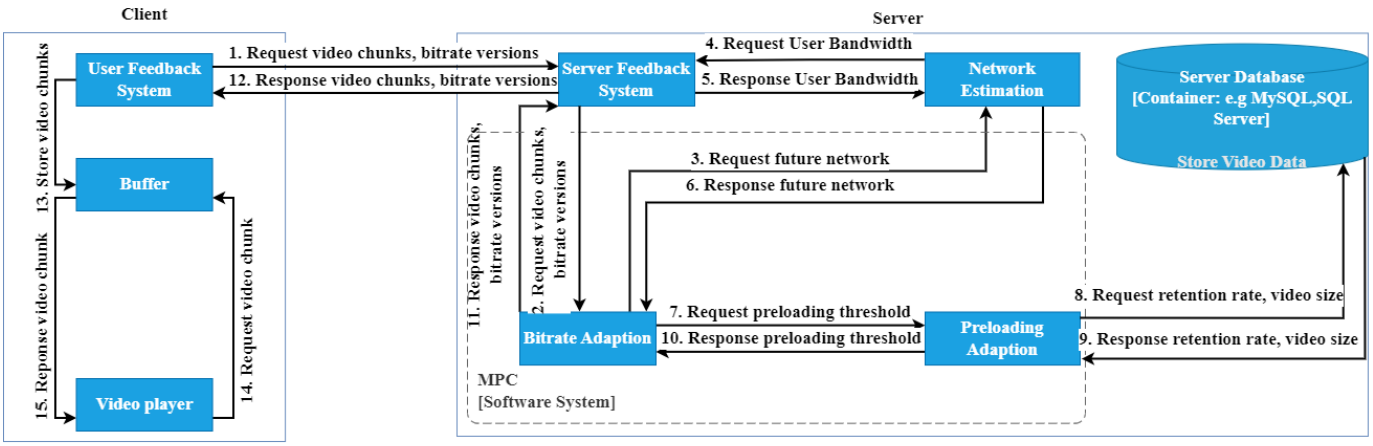

Fig. 1: System Architecture

- w: serves as a cumulative weight factor that adjusts iteratively, ensuring accurate weighting.
- $window$: denotes the moving average period, specifying the duration over which data points are considered.

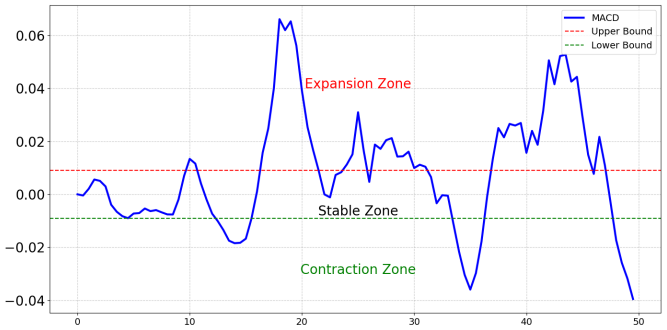

Fig. 2: MACD with Upper, Lower Bound Limits

MACD oscillates around zero, with positive values indicating short-term EMA surpassing long-term EMA, and negative values suggesting the opposite. Unlike some indicators, MACD lacks predefined upper and lower bounds. It helps identify trends, confirm uptrends, enhance understanding of user behavior, and improve user quality, as illustrated in Figure 2.

When MACD falls within the range (UB, LB), it indicates that the network condition is stable. During this phase, the harmonic mean technique is employed to guarantee minimal fluctuations in the estimated network bandwidth.

Conversely, if MACD exceeds UB or falls below LB, it signifies an agile state of the network condition. To adapt quickly, we employ an exponentially weighted moving average filter, widely used for bandwidth estimation. Depending on the network state, our methodology uses two filters for bandwidth estimation, as outlined below:

$$BW_e(cb) = \begin{cases} \gamma_1 BW_e^H(cb-1) + (1-\gamma_1)BW_s(cb-1) \\ \text{MACD} \in \text{(UB,LB)} \\ \gamma_2 BW_e(cb-1) + (1-\gamma_2)BW_s(cb-1) \\ \text{MACD} \notin \text{(UB,LB)} \\ BW_s(cb-1) \\ cb = 1 \end{cases} \tag{5}$$

$BW_e(cb)$ represents the estimated bandwidth for current bandwidth cb, $BW_e^H(cb-1)$ denotes the harmonic mean of the last n iterations, computed as follows:

$$\text{BW}_e^H(cb-1) = \frac{n}{\sum_{j=cb-n}^{cb} \frac{1}{BW_e(j)}} \tag{6}$$

Where:

- $BW_e(cb-1)$: the estimated bandwidth for the time;
- $BW_s(cb-1)$: the measured bandwidth in downloading the last bandwidth is calculated by Eq. 8.
- γ_1 and γ_2 represent the variable weights.

$$\gamma_1 = \frac{1}{1+\text{e}^{-K*(b-B_0)}} \quad \gamma_2 = \frac{1}{1+\text{e}^{k*\Delta}}$$

Where

$$b = \frac{|BW_s(cb-1) - BW_e(cb-1)|}{BW_e(cb-1)} \tag{7}$$

$$\Delta = \frac{BW_s(cb-1) - \overline{BW}_s(cb-1)}{\overline{BW}_s(cb-1)} \tag{8}$$

$$BW_s(cb) = C_k^i \tag{9}$$

K and B_0 are parameters of the logistic function. C_k^i is the average download speed introduced by work [13].

During stable periods, we use the harmonic mean technique to smooth minor fluctuations in estimated bandwidth. The harmonic mean is advantageous due to its robustness to larger outliers and its suitability for calculating average rates. We

gather data from n samples to compute the harmonic mean, as described in Equation 6. Here, we use $n = 20$ samples for analysis.

B. Bit rate adaptation and Segment Preloading

1) Stage 1: Buffer Size Adaptation and Segment preloading: At this stage, dynamic buffer size $BS_{th}^{i^{cur}}(k)$ is adjusted for each video by the algorithm proposed in [13], based on cross-user behavior, using the estimated future bandwidth (5) and the chunk-level user retention rate RET_k^i. A higher retention rate implies a need for a larger buffer to accommodate prolonged viewing sessions, while a lower rate suggests that users might switch to the next video, favoring a smaller buffer size to minimize data wastage.

Our method then calculates download video ID DV_id, which determines whether to prioritize preloading the current video or refer to preloading a future one, based on a comprehensive assessment between estimated buffer occupancy $B_{th}^{video\_id}(k)$ and current buffer size of recommended videos B^{video_id}.

Algorithm 1: Buffer Size Adaptation

Input: Rebuffering time, current video sizes, current video ID i^c, recommended video data
Output: Download video ID DV_{id}
for all $video\_id$ in $remcommended\_videos$ **do**
  Calculate $B_{th}^{video\_id}(k)$ [13]
  if $B_{th}^{video\_id}(k) \geq B^{video\_id}$ **then**
    $DV_{id} = video\_id$
  end if
end for
Return DV_{id}

2) Stage 2: Bitrate adaptation: At this stage, future network throughput has been predicted reasonably accurately for a short-term chunk horizon $[k, k+(\mu-1)]$, given the relatively stable estimated network over brief intervals. Leveraging this insight, Dynamic Programming within Model Predictive Control (MPC) is employed to select bitrate for the next preloading chunk. MPC operates within a short moving horizon μ, using predicted throughput and chunk-level user retention rates to identify the optimal bitrate selection, maximizing a reward function. Subsequently, the user agent downloads the next chunk at the selected bitrate, and the horizon advances to $[k+1, k+\mu]$, repeating for every video chunk. It's important to note that, in video streaming, the reward function typically incorporates information from past downloaded chunks. The benefit of using Dynamic Programming is that instead of exhaustively evaluating all state combinations within the prediction horizon, dynamic programming recursively computes and updates the optimal state trajectory. This approach reduces complexity by reusing previously computed results and addressing overlapping sub-problems. By efficiently exploring the state space and optimizing control actions based on historical data, dynamic programming enables MPC to make accurate decisions while maintaining computational tractability, especially in scenarios with extended prediction horizons or complex system dynamics.

In Algorithm 2, the initial step involves simulating future states from the current state. Subsequently, utilizing dynamic programming, the algorithm computes the states to facilitate the evaluation of potential future scenarios. Following this, rewards are assigned to each state, and the bitrate configuration that maximizes this reward is selected. The primary objective remains to identify the optimal bitrate level for preloading video segments, thereby enhancing the user experience during streaming sessions.

Algorithm 2: Model Predictive Control

Input: All previous bandwidth samples, the data size of the next μ chunks, current buffer size, the bit rate of the last chunk, download video ID: DV_{id}
Output: Return the R_{best} that give maximum reward $Reward_{max}$
Estimate future bandwidth with Eq. 5
for $step = 0$ to $horizon - 1$ **do**
  $newStates$ = $\emptyset$
  for all $state$ in $states$ **do**
    $Reward_{cur}$ = $SumBitrateDownloaded - (RebufferingTime + SmoothnessDifference + WastedData)$
    if $Reward_{cur} \geq Reward_{max}$ **then**
      $Reward_{max} \leftarrow Reward_{cur}$
      $R_{best} \leftarrow R_{cur}$
    end if
    for all $bitrate$ in $bitrates$ **do**
      Generate all future $bitrate$ using for next state to $newStates$
    end for
  end for
  $states = newStates$
end for
Return R_{best}

IV. Evaluation

A. Experimental Settings

In each experimental scenario, the Quality of Experience (QoE) and Running Time of the proposed method are eval-

TABLE I: Characteristics of Video Content [14]

Video Id	Video Duration	Video Theme
01	17	Study
02	26	Entertainment
03	37	Life
04	40	Life
05	47	Life
06	06	Entertainment
07	125	Game

TABLE II: Network Prediction parameters

Name	Value
EMA_{short}	12 samples
EMA_{long}	26 samples
UB	0.009
LB	-0.009
K	31.50
B_0	0.460
n	20.00

uated across various network conditions and user behaviors, using datasets provided by [14] and [16]. The dataset comprises seven short videos with diverse content characteristics, detailed in Table I. Additionally, it incorporates network traces and user retention rate information.

The network traces are classified into four types: high, medium, low, and mixed, each consisting of twenty network traces spanning 2900 seconds. Network throughput updates every 500ms to simulate real network conditions accurately. Each video is encoded into three bitrate levels in the Variable Bitrate (VBR) mode, and subsequently divided into multiple one-second chunks (τ=1).

During the bandwidth prediction step, parameters are configured as shown in Table II. Our method initiates by pairing a long-term EMA EMA_{long} and short-term EMA EMA_{short} corresponding to the standard pair of MACD - (26,12) [17], with 26 representing the long term trend, and 12 representing the short term fluctuations in bandwidth. Subsequently, the boundary to specify the network traffic status (UB, LB) is set to (-0.009, 0.009) based on historical data analysis, serving as thresholds to distinguish between stable and varying network traffic conditions.

Furthermore, n, representing the number of samples used in the harmonic mean calculation, is set to 20.

Additionally, parameters K and B_0 inherent to the logistic function utilized in traffic prediction are refined through gradient descent iterations. To elaborate, we calculate the partial derivatives of the cost function concerning K and $B0$, which represent the sensitivity of the Reward Function to changes in these parameters. Throughout each iteration, we fine-tune the values of K and B_0 in a manner that optimizes the Reward function, guided by the symmetric derivative approximations [18]. As the results, K, representing the steepness of the logistic curve, is tuned to 31.5, while B_0, the midpoint of the curve, is tuned to 0.46. These values best align with the observed network traffic data. This ensure to accurately weight factors in bandwidth estimation.

This study conducts the comparative analysis across 10 users and 10 network traces under 2 distinct network conditions, utilizing 4 bitrate levels. These experiments are performed on a 64-bit Windows 11 laptop featuring an Intel® UHD Graphics 620 and 16GB of RAM. The performance of our proposed method is compared with the following reference methods **JPBA** [13], **Fixed-preload** [14], **Next-One** [12].

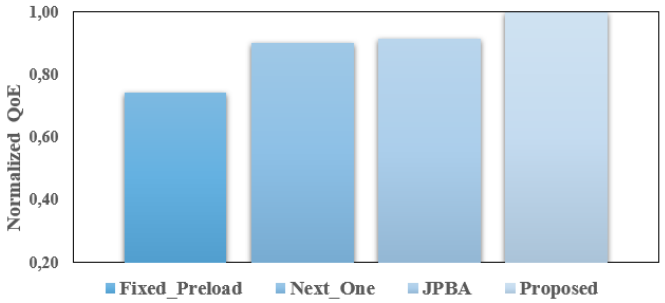

Fig. 3: Normalized QoE of our approach vs. the reference methods

To ensure fair and consistent evaluation across all methodologies, we employ a standardized Quality of Experience (QoE) formula provided by [13], that offers a comprehensive assessment of the user experience by accounting for the factors, in formula 10 and formula 11.:

$$QoE = \sum_{i=1}^{N} QoE_i \tag{10}$$

$$QoE_i = \sum_{j=1}^{J_i} QoE_i(sp_1 * BR_j^i - sp_2 * RBT_j^i - \sum_{j=1}^{J_i-1} (sp_3 * |BR_i^j - BR_{i+1}^j|) \tag{11}$$

where:

- BR_j^i:chunk bitrate
- $|BR_i^j - BR_{i+1}^j|$: chunk bitrate variations, measure the consistency of the bitrate throughout the video playback with higher variations indicating potential disruptions in streaming quality.
- RBT_j^i:re-buffering times, representing the intervals during which the video pauses to buffer additional content affecting the continuity of the viewing experience.
- sp_1, sp_2, sp_3: the scaling parameters denoting the importance or significance attributed to individual factors, as outlined in [16], [19].

B. Experimental Results and Analysis

In terms of QoE, Figure 3 displays varying levels of quality of experience (QoE) with **Fixed-Preload** exhibiting a notable 25% lower QoE compared to our Proposed method, indicating a significant difference in user experience. On the other hand, both **Next One** and **JPBA** perform similarly with approximately 10% lower QoE compared to our proposed method. Since the proposed method can automatically adjust content quality based on available network conditions and user device capabilities, it allows for a more enjoyable and consistent viewing experience, avoiding excessive stuttering or loss of quality.

Regarding to running time, as illustrated in Figure 4, with the simplification in the algorithm, **Fixed Preload** and **Next**

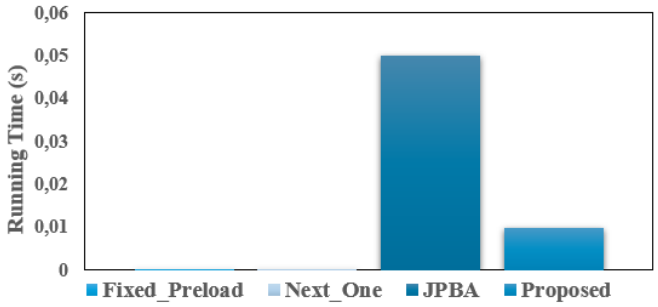

Fig. 4: Running time of our approach vs. the reference methods

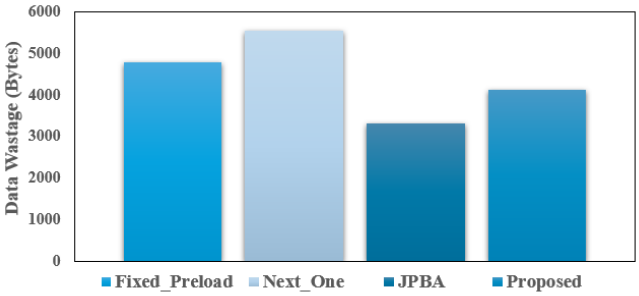

Fig. 5: Data wastage of our approach vs. the reference methods

One exhibit comparable runtime of 0.033 ms; In contrast, **JPBA** requires a longer execution time of 50 ms, while our proposed algorithm operates at $9ms$. While our proposed method may not achieve the fastest average runtime, it offers substantial enhancement in QoE. Additionally, the runtime of $9ms$ makes our proposed solution feasible for deployment within the context of real-time adaptive short video streaming systems.

Regarding to data wastage as seen in Figure 5, **Fixed-Preload** exhibits a data wastage of 4773.56 bytes, while **Next One** shows a slightly higher wastage of 5544.83 bytes. In contrast, **JPBA** demonstrates lower data wastage of 3325.19 bytes. Our proposed method proves to be a harmonious balance compared to JPBA, with a bit higher data wastage of 4116.16 bytes, but higher QoE, and lower running time.

V. Conclusions

In conclusions, the proposed solution has demonstrated its effectiveness in providing satisfactory short-form video streaming services across diverse network conditions, contributing to the advancement of short-form video services in the digital landscape. For our future work, potential endeavors could focus on further minimizing data wastage by a machine learning model.

Acknowledgment

This research is funded by Vingroup and supported by Vingroup Innovation Foundation (VINIF) under project code VINIF.2020.DA03.

References

[1] Business of Apps, "TikTok Statistics," https://www.businessofapps.com/data/tik-tok-statistics/, 2024, accessed on February 22, 2024.

[2] Y. Zhang, Y. Liu, L. Guo, and J. Y. Lee, "Measurement of a large-scale short-video service over mobile and wireless networks," *IEEE Transactions on Mobile Computing*, 2022.

[3] V. H. Nguyen, D. T. Bui, T. L. Tran, C. T. Truong, and T. H. Truong, "Scalable and resilient 360-degree-video adaptive streaming over http/2 against sudden network drops," *Computer Communications*, vol. 216, pp. 1–15, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S014036642400001X

[4] H. T. Tran, N. P. Ngoc, A. T. Pham, and T. C. Thang, "A multi-factor qoe model for adaptive streaming over mobile networks," in *2016 IEEE Globecom Workshops (GC Wkshps)*. IEEE, 2016, pp. 1–6.

[5] N. V. Hung, T. D. Chien, N. P. Ngoc, and T. H. Truong, "Flexible http-based video adaptive streaming for good qoe during sudden bandwidth drops," *EAI Endorsed Transactions on Industrial Networks and Intelligent Systems*, vol. 10, no. 2, pp. e3–e3, 2023.

[6] V. H. Nguyen, N. N. Pham, C. T. Truong, D. T. Bui, H. T. Nguyen, and T. H. Truong, "Retina-based quality assessment of tile-coded 360-degree videos," *EAI Endorsed Transactions on Industrial Networks and Intelligent Systems*, vol. 9, no. 32, pp. e2–e2, 2022.

[7] H. N. Viet and H. T. Thu, "An effective segment pre-fetching for short-form video streaming," *International Journal of Computer Science and Network Security*, vol. 23, no. 3, pp. 81–93.

[8] G. Zhang, K. Liu, H. Hu, and J. Guo, "Short video streaming with data wastage awareness," in *2021 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE, 2021, pp. 1–6.

[9] S. Zhu, T. Karagioules, E. Halepovic, A. Mohammed, and A. D. Striegel, "Swipe along: a measurement study of short video services," in *Proceedings of the 13th ACM Multimedia Systems Conference*, 2022, pp. 123–135.

[10] H. Zhang, Y. Ban, X. Zhang, Z. Guo, Z. Xu, S. Meng, J. Li, and Y. Wang, "Apl: Adaptive preloading of short video with lyapunov optimization," in *2020 IEEE International Conference on Visual Communications and Image Processing (VCIP)*. IEEE, 2020, pp. 13–16.

[11] H. Nguyen, T. N. Dao, N. S. Pham, T. L. Dang, T. D. Nguyen, and T. H. Truong, "An accurate viewport estimation method for 360 video streaming using deep learning," *EAI Endorsed Transactions on Industrial Networks and Intelligent Systems*, vol. 9, no. 4, 2022.

[12] J. He, M. Hu, Y. Zhou, and D. Wu, "Liveclip: towards intelligent mobile short-form video streaming with deep reinforcement learning," in *Proceedings of the 30th ACM Workshop on Network and Operating Systems Support for Digital Audio and Video*, 2020, pp. 54–59.

[13] N. T. Phong, T. T. Huong, P. N. Nam, T. C. Thang, and D. Nguyen, "Joint preloading and bitrate adaptation for short video streaming," *IEEE Access*, vol. 11, pp. 121 064–121 076, 2023.

[14] X. Zuo, Y. Li, M. Xu, W. T. Ooi, J. Liu, J. Jiang, X. Zhang, K. Zheng, and Y. Cui, "Bandwidth-efficient multi-video prefetching for short video streaming," in *Proceedings of the 30th ACM International Conference on Multimedia*, ser. MM '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 7084–7088. [Online]. Available: https://doi.org/10.1145/3503161.3551584

[15] G. Appel, *The moving average convergence-divergence trading method: advanced version*. Scientific Investment Systems, 1985.

[16] [Online]. Available: https://github.com/AItransCompetition/Short-Video-Streaming-Challenge

[17] G. Appel and E. Dobson, *Understanding MACD*. Traders Press, 2007, vol. 34.

[18] S. J. Wright, "Numerical optimization," 2006.

[19] M. Licciardello, "Understanding, reconstructing and optimising adaptive bitrate video streaming," Ph.D. dissertation, ETH Zurich, 2022.