Digital Twin-Assisted Adaptive Preloading for Short Video Streaming

Shengbo Liu*, Wen Wu*, Shaofeng Li*, Tom H. Luan†, and Ning Zhang‡
*Frontier Research Center, Peng Cheng Laboratory, Shenzhen, China
†School of Cyber Science and Engineering, Xi'an Jiaotong University, Xi'an, China
¶Department of Electrical and Computer Engineering, University of Windsor, Windsor, Canada
Email: {liushb, wuw02, lishf}@pcl.ac.cn, tom.luan@xjtu.edu.cn, ning.zhang@uwindsor.ca

***Abstract*—We propose a digital twin-assisted adaptive preloading scheme to reduce bandwidth waste as well as enhance user quality of experience (QoE) for short video streaming. Though preloading video content can reduce rebuffering and improve user QoE, non-sequential playback of short videos induced by user swipe can result in substantial bandwidth wastage in mobile networks. To tackle this problem, we first model the short video streaming system and carry out preloading threshold analysis. We then construct a digital twin-assisted adaptive preloading framework for short video streaming. By collecting and analyzing the user's historical throughput and tracking swipe timing information, a throughput prediction model and a probabilistic model can be constructed to accurately predict future throughput and user swipe behavior, respectively. Utilizing the predicted information and real-time running status data from a short video application, we design a preloading strategy to enhance bandwidth efficiency while achieving high user QoE. Simulation results demonstrate the effectiveness of our proposed scheme compared with the state-of-the-art schemes.**

I. Introduction

Nowadays, video streaming occupies around 70% of total mobile data traffic, which is expected to reach 80% by 2028.[1] In recent years, short video platforms, e.g., TikTok, YouTube Shorts, and Instagram Reels, have become phenomenal applications [1]. The total number of short video users worldwide has exceeded one billion in 2022 [2].

In short video streaming, users can swipe the screen to skip from the current video to the next one in the video recommendation list, allowing them to easily search for their interested videos. Due to channel condition fluctuations in wireless networks [3], it is crucial to preload video content on mobile devices to facilitate smooth playback and avoid rebuffering. Considering the user's swipe behavior in short video streaming, it is imperative to preload both the current video and the following ones in the recommended list, which can effectively reduce rebuffering during video playback. However, some preloaded video content may be unwatched and discarded due to users' frequent video swipes, resulting in significant bandwidth waste [2]. Recent studies indicate that only 30.92% of videos are watched in their entirety [4]. Moreover, nearly 45% of the downloaded video is ultimately discarded [5]. Such bandwidth waste not only escalates the cost of mobile data traffic for users but also incurs operational expenses for short video service providers [2]. Consequently, an adaptive preloading scheme is essential for short video streaming, aiming to reduce bandwidth waste while enhancing user quality of experience (QoE).

Recent works have studied short video preloading to achieve the aforementioned objective. Nguyen *et al.* considered the network throughput condition and then dynamically adjusted buffer sizes for the current video and the following ones in the video recommendation list [5]. Zhang *et al.* presented a novel preloading mechanism that dynamically preloads the recommended videos to maximize playback smoothness and minimize bandwidth waste [6]. However, these works do not fully consider the impact of varying bitrates on system performance. Zhou *et al.* introduced a probabilistic model of user retention to adaptively control the buffer size for reducing bandwidth waste [7]. Zhang *et al.* proposed a learning-based approach to capture the characteristics of past network conditions and train adaption models for reducing data usage [2]. Nevertheless, these works may not fully exploit user-specific information for buffer control and bitrate adaptation.

To leverage user-specific information and enhance the performance of short video streaming, we introduce the promising digital twin (DT) technology [8], [9]. DT is a virtual representation of a physical entity that enables real-time synchronization between the digital model and the actual entity [10]. By applying DT technology in the short video streaming scenario, we can create a virtual representation of a user, capturing his/her throughput traces and swipe preferences. Analysis of this user-specific historical and real-time data facilitates more precise traffic and swipe predictions for buffer control and bitrate adaptation, thereby effectively reducing bandwidth waste while ensuring high user QoE. Consequently, it is worthwhile to explore a DT-based solution by dynamically adjusting the preloading strategy, tailored to the user's characteristics and real-time network conditions.

In this paper, we propose a DT-assisted adaptive preloading framework for short video streaming, aiming to reduce bandwidth waste and enhance user QoE. We first model the short video streaming system and analyze the relationship between the achievable throughput and the encoding bitrate of a short video. Then, we introduce a DT framework for short video streaming, consisting of four functional blocks:

[1]https://www.ericsson.com/en/reports-and-papers/mobility-report/dataforecasts/mobile-traffic-forecast

data pool, throughput prediction, user swipe prediction, and preloading strategy. In particular, the user behavior prediction block constructs a probabilistic model based on historical video swipe information to predict the user's viewing behavior. More importantly, the preloading strategy block is responsible for making effective preloading decisions via the designed algorithms, including adaptively adjusting buffer sizes for the current video and the ones in the video recommendation list and selecting the bitrates for the upcoming video contents. Extensive simulation results demonstrate the effectiveness of our solution in reducing bandwidth waste and enhancing user QoE, outperforming the state-of-the-art schemes.

The main contributions of this paper are summarized as follows:

- We propose a DT framework for short video streaming, which establishes a throughput model and a probabilistic model to predict the user's future throughput and video swipe event, respectively.
- We design an adaptive short video preloading strategy, which can dynamically adjust the buffer threshold and adaptively select the bitrates for the current video and the next ones in the video recommendation list.

II. System Model and Problem Formulation

We consider the scenario in which multiple users watch videos through short video applications over mobile networks. The users' digital twins (UDTs) are constructed and stored in the edge server which is collected to the base station. A general short-form video streaming system is depicted in Fig. 1. In this system, a user initiates video requests directed toward a content delivery network (CDN) server, which serves as a repository for short videos. Each video transmitted to the user's device is segmented into small chunks, featuring consistent playback duration but varying bitrate levels. In addition to the current video being played, the videos in the recommendation queue can also be preloaded. Upon receipt of video chunks from the CDN server, the user's device stores them in its buffer. When watching a video, the user can swipe away the current video to the next video in the recommendation queue. To ensure a satisfactory QoE, the user can decide which chunk to preload as well as its bitrate to accommodate the fluctuating network conditions and random user swipe timing.

A. Short Video Streaming Model

We consider a viewing session that starts when a user opens the short video application and ends when the user closes the application. During the viewing session, the user watches a list of short videos $\mathcal{V} = \{V_1, V_2, ..., V_N\}$ recommended by the CDN server in sequential order. Video V_i is divided by K_i chunks with an identical duration T_0. Each chunk is encoded by different bitrate levels. The short video player can download chunk $v_{i,k}$ encoded by bitrate $r_{i,k} \in \mathcal{R}$, where $\mathcal{R}$ denotes the set of all available bitrates.

The video chunks are downloaded into a *playback buffer*, which stores the downloaded but unwatched video content. To ensure smooth playback, the preloaded video content contains the current video and the subsequent videos in the recommended list. Let $B(t) \in [0, B_{max}]$ be the *buffer occupancy* at time t, i.e., the play time of the video chunks left in the buffer. The *buffer size* B_{max} depends on the service provider, as well as the player. Let $B_i(t) \in [0, B_i^{th}]$ be the buffer occupancy for video V_i at time t, which is the play time of the video left in the buffer. The *buffer threshold* B_i^{th} for video V_i is determined by the policy of the player. The buffer occupancy $B(t)$ evolves as the chunks are downloaded, and the videos are played. Thus, we have $B(t) = \sum_{i=i_c}^{N_p} B_i(t)$, where i_c is the current video index, and N_p denotes the maximum number of videos that have preloaded chunks at time t. Here, we assume that the preloaded chunks for the current video will be discarded once swiping to the next one. Let $B_{i,k} = B_i(t_{i,k})$ denote the buffer occupancy for video V_i when the player starts to download chunk $v_{i,k}$. If the player immediately starts to download chunk $v_{i,k+1}$ as soon as chunk $v_{i,k}$ is downloaded, the buffer dynamics can then be formulated as

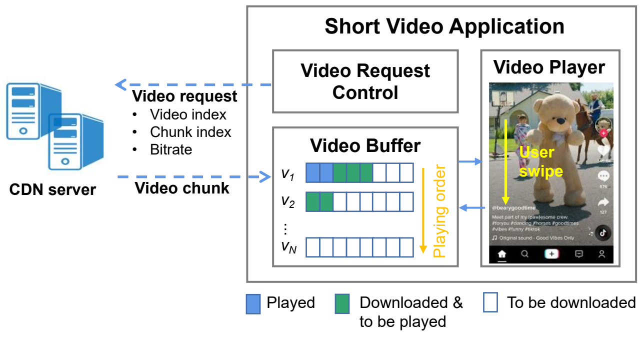


Fig. 1. Overview of a short video streaming system.

$$B_{i,k+1} = max\{B_{i,k} - \frac{r_{i,k}T_0}{C_{i,k}}, 0\} + T_0, \tag{1}$$

where $C_{i,k}$ is the average download speed during downloading chunk $v_{i,k}$. Let $C(t)$ denote the network throughput at time t. Then, we have

$$C_{i,k} = \frac{1}{t_{i,k+1} - t_{i,k}} \int_{t_{i,k}}^{t_{i,k+1}} C(t)dt. \tag{2}$$

B. Problem Formulation

As discussed in [1], user QoE is mainly determined by the video bitrate, the bitrate variation, and the rebuffering delay. We define the QoE of video V_i by a weighted sum of the aforementioned factors

$$QoE_i = \omega_1 \cdot \sum_{k=1}^{K_i^s} r_{i,k} - \omega_2 \cdot \sum_{k=1}^{K_i^s - 1} |r_{i,k+1} - r_{i,k}| \\ - \omega_3 \cdot \sum_{k=1}^{K_i^s} max\{\frac{r_{i,k}T_0}{C_{i,k}} - B_{i,k}, 0\}, \tag{3}$$

where K_i^s denotes the number of watched chunks of video V_i before swiping the video; ω_1, ω_2, and ω_3 are non-negative weighting parameters related to the video bitrate, bitrate variations, and the rebuffering time, respectively. Considering that users may prioritize different factors, the weighting parameters can be customized based on their individual preferences.

The bandwidth consumed by downloading video V_i is

$$BW_i = \sum_{k=1}^{K_i^s + B_i^s/T_0} r_{i,k} \cdot T_0, \tag{4}$$

where B_i^s denotes the buffer occupancy for video V_i when swiping to video V_{i+1}. Then, the bandwidth waste caused by video swipes can be formulated as

$$BW_w = \sum_{i}^{N} \sum_{k=K_i^s+1}^{B_i^s/T_0} r_{i,k} \cdot T_0. \tag{5}$$

To balance the impacts of bandwidth usage and user QoE on the system performance, we define a *utility function* U, which is given by

$$U = \sum_{i}^{N} (QoE_i - \omega_4 \cdot BW_i), \tag{6}$$

where ω_4 denotes the weighting parameter related to the bandwidth usage. To minimize the bandwidth usage while improving the user QoE, we formulate the following problem

$$\max_{B_i^{th}, r_{i,k}} \; U \tag{7}$$

$$\text{s.t. } B_{i,k} \in [0, B_i^{th}], B_i^{th} \in (0, B_{max}), \tag{7a}$$

$$r_{i,k} \in \mathcal{R}. \tag{7b}$$

The ultimate goal of this paper is to reduce bandwidth usage and improve the user QoE concurrently. According to Eq. (4), the bandwidth usage is closely related to the buffer occupancy when swiping video. Since the buffer occupancy is limited by the buffer threshold, a smaller buffer threshold can significantly reduce the bandwidth usage. However, Eq. (3) shows that the buffer occupancy can efficiently reduce the rebuffering delay by handling the network scenario with a period of low throughput. A larger buffer threshold can effectively reduce the rebuffering delay. Therefore, the buffer threshold setting plays a crucial role in improving the utility function defined in Eq. (6), as well as the bitrate adaption which has been investigated in many works [1], [7]. In the face of the vagaries of network throughput and user swipe timing, it is difficult to design an optimal strategy of real-time buffer control and bitrate adaption.

C. Buffer Threshold Analysis

As shown in Eq. (3), the rebuffering delay for chunk $v_{i,k}$ is denoted by $max\{\frac{r_{i,k}T_0}{C_{i,k}} - B_{i,k}, 0\}$, which illustrates that the buffer occupancy is closely associated with the throughput of downloading the chunk. Obviously, the optimal buffer threshold for video V_i can be formulated as

$$\arg\min_{B_i^{th}} \; BW_i \tag{8}$$

$$\text{s.t. } max\{\frac{r_{i,k}T_0}{C_{i,k}} - B_{i,k}, 0\} = 0, \tag{8a}$$

$$B_{i,k} \in [0, B_i^{th}], B_i^{th} \in (0, B_{max}). \tag{8b}$$

It is easy to derive the optimal buffer threshold B_i^{th*}, i.e., $B_i^{th*} = \frac{r_{i,k}T_0}{C_{min}}$, where C_{min} denotes the minimum throughput downloading any chunk in video V_i.

To find out the relationship between the network throughput and the bitrate, we first consider a worst-case scenario where a user continuously scrolls the screen to swipe the videos once watching the first chunk. In this scenario, we assume that video playback begins only when the number of buffered chunks reaches an initial threshold denoted as B_0, where $B_0 \geq 1$. To prevent rebuffering events while watching the current video or transitioning to the next video, it is necessary to download the subsequent chunk of the current video and B_0 chunks of the next video while the user is viewing the first chunk. Let C_{min}^{worst} represent the minimum achieved throughput that can sufficiently handle this worst-case scenario. Consequently, C_{min}^{worst} must satisfy the following equation

$$C_{min}^{worst} \cdot T_0 = r_{i,B_0+1} \cdot T_0 + \sum_{j=1}^{B_0} r_{i+1,j} \cdot T_0. \tag{9}$$

Thus, we have $C_{min}^{worst} = r_{i,B_0+1} + \sum_{j=1}^{B_0} r_{i+1,j}$.

When the achievable throughput falls below the minimum bitrate R_{min}, a rebuffering event may occur. This implies that the download rate is slower than the video's playback rate, resulting in the inability to ensure the smooth playback of the current video. However, if the achievable throughput satisfies $C \in [R_{min}, C_{min}^{worst}]$, there is room for designing the preloading strategy to prevent rebuffering events.

III. Digital Twin-Assisted Adaptive Preloading Framework

In this section, we present a novel framework called Digital Twin-Assisted Adaptive Preloading (DTAAP), which aims to reduce bandwidth usage while improving user QoE. As depicted in Figure 2, the DTAAP framework leverages the user's DT to generate a personalized preloading decision by utilizing private user data. The DTAAP framework comprises four functional blocks, i.e., data pool, throughput prediction, user behavior prediction, and preloading strategy. The Data pool block periodically collects and stores user data related to throughput prediction, user behavior prediction, and preloading decisions [2]. Next, we will introduce the other three functional blocks in detail.

[2]The data upload typically consumes only a few bytes per second on average, exerting minimal impact on bandwidth usage. Moreover, we can leverage the latest data protection techniques, such as homomorphic encryption, differential privacy, among others, to enhance user privacy [11].

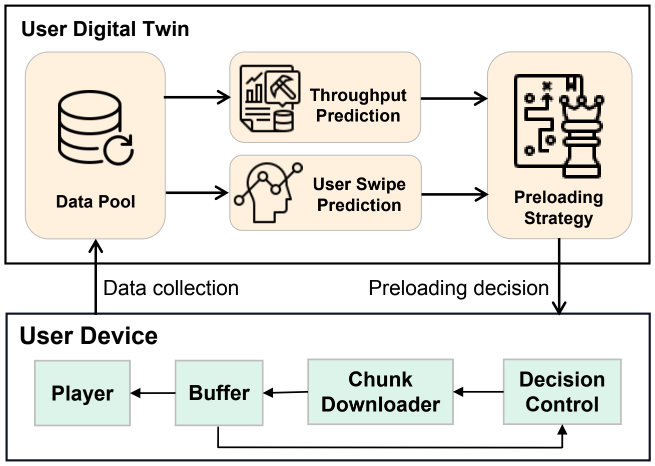


Fig. 2. The proposed DTAAS framework.

A. Throughput Prediction

Accurate throughput prediction is critical in making effective video preloading decisions. Short video applications are often used in diverse wireless access scenarios, such as open-air spaces, indoor locations, or on a moving bus. Each scenario presents distinct throughput characteristics, as discussed in [12]. Moreover, the dynamics of wireless channel and network traffic load contribute to rapid fluctuations in a user's achieved throughput. Nevertheless, by utilizing the throughput traces provided by the user, the user's DT can effectively identify the network environment using the technology proposed in [12]. Once the network scenario is determined, the next step is to design a model that can predict future throughput accurately. One challenge in predicting throughput is the rapid variation experienced during the duration of playing a video chunk, making it difficult to capture this characteristic. To address this issue, we focus on predicting the average throughput over chunk-level duration. Additionally, we can leverage buffer control and bitrate adaptation techniques to mitigate the impact of prediction error. Therefore, we use the following model to predict future throughput $\hat{C} = \alpha_1 \cdot C_{ave} + \alpha_2 \cdot C_{last}$, where C_{ave} denotes the average throughput, C_{last} denotes the throughput of downloading last chunk, α_1 and α_2 are two weighting parameters.

B. User Behavior Prediction

Considering that videos within the same category may have different durations, we use X pieces of user trace data $Tr_x(k_x, K_x)$ to represent the scrolling behavior of the x-th trace. Here, k_x denotes the viewing chunk number when swiping the video, and K_x represents the total number of chunks in the video. To build a swiping ratio distribution model with 1% resolution, we employ Algorithm 1. The resulting probability distribution of the model is denoted by a vector $\mathbf{S}$. Using this model, we can determine the probability of swiping the video at a specific chunk during playback. For

Algorithm 1 State transition model construction algorithm.

```
Input: User traces $Tr_x(k_x, K_x)$.
Output: The probability distribution vector $\mathbf{S}$.
 1: Initialize a vector $\mathbf{S} = [s_1, s_2, ..., s_{100}]$;
 2: for $x$ in $[1, X]$ do
 3:     if $\lceil \frac{100(k_x-1)}{K_x} \rceil \neq \lceil \frac{100k_x}{K_x} \rceil$ then
 4:         for $y$ in $[\lceil \frac{100(k_x-1)}{K_x} \rceil + 1, \lceil \frac{100k_x}{K_x} \rceil]$ do
 5:             $S[y] = S[y] + \frac{1}{X(\lceil \frac{100k_x}{K_x} \rceil - \lceil \frac{100(k_x-1)}{K_x} \rceil)}$;
 6:         end for
 7:     else
 8:         $S[\lceil \frac{100k_x}{K_x} \rceil] = S[\lceil \frac{100k_x}{K_x} \rceil] + 1/X$;
 9:     end if
10: end for
11: return $\mathbf{S}$;
```

instance, for a video i consisting of K_i chunks, the probability of swiping the video at chunk $v_{i,k}$ can be calculated by

$$p_{i,k} = \sum_{j=\lceil \frac{100(k-1)}{K_i} \rceil + 1}^{\lceil \frac{100k}{K_i} \rceil} S[j]. \tag{10}$$

Furthermore, the total swipe probability of all chunks in the video V_i is 1, i.e., $\sum_{k=1}^{K_i} p_{i,k} = 1$.

It is worth emphasizing that different categories of videos may have distinct state transition models. By matching the video category with the corresponding model, we can extract valuable information that aids in making informed preloading decisions, including buffer control and bitrate selection. This ensures that the preloading strategy is tailored to the characteristics of each video type, leading to effective and efficient preloading. From the established state transition model, we can extract the following valuable insights

- *Minimum Viewing Duration* K_{min}: K_{min} represents the minimum number of video chunks that a user typically watches before swiping the video ($K_{min} \geq 1$);
- *Short viewing Duration* K_{short}: K_{short} represents swiping video with a high probability P_{th}^{short} after watching only a few chunks ($K_{short} \leq max\{0.1K_i, 5\}$), i.e., the minimum value that satisfies $\sum_{j=1}^{K_{short}} p_{i,j} > P_{th}^{short}$.
- *Long Viewing Duration* K_{long}: K_{long} represents swiping video with a high probability P_{th}^{long} after watching nearly the whole video ($K_{long} \geq max\{0.9K_i, K_i - 5\}$), i.e., the minimum value that satisfies $\sum_{j=K_{long}}^{K_i} p_{i,j} > P_{th}^{long}$.

C. Preloading Strategy

The preloading strategy for short video streaming is distinct from that of conventional long video streaming because it involves preloading both the current video and the next videos in the recommended list. The primary goal of the preloading strategy is to minimize bandwidth waste while improving the user QoE concurrently. The proposed preloading strategy, outlined in Algorithm 2 and Algorithm 3, focuses on two essential aspects: buffer control and bitrate adaptation. We first determine the buffer thresholds for the current video and the next ones in the recommended list using the throughput

Algorithm 2 DT-assisted adaptive preloading algorithm.

```
Input: The chunk viewing probability $p_{i,k}$; The number of preloading videos $N_p$; Sleep time $T_{sleep}$.
 1: Update the average throughput $C_{ave}$, the predicted throughput $\hat{C}$;
 2: Update $B_{i_c}^{th}$ and $B_{i_n}^{th}$ with $\hat{C}$ and $C_{ave}$, respectively;
 3: if $B_{i_c} < B_{i_c}^{th}$ then
 4:     Derive $r_{i_c,k}^{*}$ with Algorithm 3;
 5:     Download the current video $v_{i_c}$ with $r_{i_c,k}^{*}$;
 6: else
 7:     for $j$ in $1, N_p - 1$ do
 8:         if $B_{i_c+j} < B_{i_n}^{th}$ then
 9:             Derive $r_{i_c+j,k}^{*}$ with Algorithm 3;
10:             Download the video chunk $v_{i_c+j,k}$ with $r_{i_c+j,k}^{*}$;
11:         else if $j == N_p - 1$ then
12:             Sleep for $T_{sleep}$ time;
13:         end if
14:     end for
15: end if
```

state and user swipe information. Subsequently, we select the appropriate bitrate for preloading the video, aiming to enhance the user QoE.

1) Buffer Control: We consider separate buffer thresholds for the current video and the next videos in the recommended list. The threshold of preloaded chunks for the current video, denoted as $B_{i_c}^{th}$, is determined by balancing the need to reduce rebuffering and bandwidth waste. The value of $B_{i_c}^{th}$ depends on the user's viewing behavior and network throughput C, i.e.,

$$B_{i_c}^{th} = \begin{cases} max\{B_i^{th*}, B_{i_c}^{min}\} + \lceil \frac{K_{long}}{K_i} \rceil, & C \geq C_{min}^{worst} \\ max\{B_i^{th*}, B_{i_c}^{min}\} - \lceil \frac{K_{short}}{K_i} \rceil, & \text{otherwise} \end{cases},$$

where $B_{i_c}^{min}$ denotes a minimum buffer threshold for the current video.

For the videos in the recommended list, the threshold of preloaded chunks denoted as $B_{i_n}^{th}$, is determined to ensure smooth playback and reduce rebuffering when swiping to the next video. The value of $B_{i_n}^{th}$ depends on the user's viewing behavior and network conditions and is given by

$$B_{i_n}^{th} = \begin{cases} max\{B_i^{th*} - \lceil \frac{K_{min}}{K_{short}} \rceil, K_{min}, 2\}, & C \geq C_{min}^{worst} \\ max\{B_i^{th*}, K_{min}, 2\}, & \text{otherwise} \end{cases}.$$

2) Bitrate Adaption: To address the challenge of adaptive bitrate selection in response to fluctuating network conditions, we have devised an adaptive bitrate control scheme that focuses on improving user QoE. The scheme dynamically adjusts the bitrate of the video chunk to be downloaded, taking into account the available network bandwidth and the buffer state. Algorithm 3 outlines the steps of this scheme. The basic idea is to adaptively adjust the bitrate according to the network throughput and buffer state. This method can effectively reduce bitrate variations, which degrades the user QoE performance. Furthermore, the average throughput is used to match the bitrate for the subsequent recommended videos instead of the real-time throughput. Thus, the bitrate variations of the next videos can be also reduced.

Algorithm 3 DT-assisted bitrate adaption algorithm.

```
 1: if Preload the current video then
 2:     if A rebuffering event occurs then
 3:         Reduce the bitrate $r_{i_c,k}^{*}$ according to $\hat{C}$;
 4:     else if $\hat{C} < r_{i_c-1,k}$ and $B_{i_c} < B_{i_c}^{th}/2$ then
 5:         Reduce the bitrate $r_{i_c,k}^{*}$ according to $\hat{C}$;
 6:     else if $\hat{C} > r_{i_c-1,k}$ and $B_{i_c} > B_{i_c}^{th}/2$ then
 7:         Increase the bitrate $r_{i_c,k}^{*}$ according to $\hat{C}$;
 8:     end if
 9:     return $r_{i_c,k}^{*}$;
10: else
11:     for $j$ in $1, N_p - 1$ do
12:         Match the bitrate $r_{i_c+j,k}^{*}$ for the recommended videos $v_{i_c+j}$ with $C_{ave}$;
13:     end for
14:     return $r_{i_c+j,k}^{*}$;
15: end if
```

IV. Performance Evaluation

A. Experimental Setting

We use the trace-driven simulator provided in [13] to evaluate the proposed DTAAP scheme, which simulates the user's playing and swipe behavior by sampling the offline video retention rate table and the throughput traces. In this simulator, there are 4 players in the recommendation list, i.e., 5 players in total. Each video is encoded into 3 representations on different bitrates (i.e., 750kbps, 1200kbps, and 1850kbps) and further cropped into chunks with $T_0 = 1$ second. The test cases are divided into 4 network scenarios: 1) High bandwidth, 2) Medium bandwidth, 3) Low bandwidth, and 4) Mixed bandwidth. For each case, we sample 50 playback traces, multiplied by 20 network traces to evaluate the performance. In accordance with [7], the weighting parameters for the user QoE and the bandwidth usage ω_1, ω_2, ω_3, and ω_4 are respectively configured as follows:s 1, 1, 1.85, and 0.5.

The proposed DTAAP is compared with the following reference schemes:

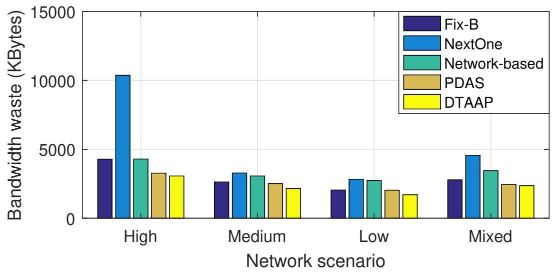
- **Fix-B**: This scheme has a fixed buffer size for the current video and the next videos in the recommended list;
- **NextOne**: This scheme preloads the next video until the current video is fully downloaded [5];
- **Network-based**: This scheme dynamically adjusts the buffer sizes for the current video and the next recommended videos based on the predicted throughput [5];
- **PDAS**: This scheme leverages the user retention probability to build a probabilistic model, which is then utilized to control the maximum buffer size and the bitrate [7].

B. Simulation Results

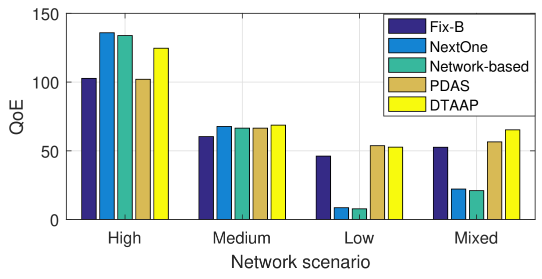
The utility results of our proposed DTAAP and the reference schemes are shown in Table I. It can be observed that the DTAAP consistently achieves the best performance across all network scenarios. This is because our DTAAP strikes a balance between QoE performance and bandwidth efficiency. With dynamic buffer threshold settings adapted to the network condition and user viewing preference, our

TABLE I
UTILITY PERFORMANCE ACHIEVED ON FOUR NETWORK SCENARIOS

Schemes	High	Medium	Low	Mixed
Fix-B	26.79	8.24	3.13	-1.92
NextOne	22.71	-16.78	-74.25	-67.91
Network-based	43.87	-17.35	-74.79	-64.65
PDAS	30.62	15.69	11.46	3.34
DTAAP	**44.78**	**16.62**	**11.41**	**7.29**



(a) Bandwidth waste



(b) QoE

Fig. 3. Performance of the proposed scheme and reference schemes.

DTAAP can achieve less bandwidth waste than other schemes. Figure 3 (a) presents the bandwidth waste incurred by the five schemes evaluated. It is evident that our proposed DTAAP consistently achieves the lowest bandwidth waste across all four network scenarios. The figure also highlights the trend of increasing bandwidth waste as the available bandwidth decreases. This is because selecting higher bitrates in favorable network conditions leads to larger bandwidth waste.

Figure 3 (b) depicts the QoE performance achieved by the five schemes. It is evident that our proposed DTAAP consistently achieves high QoE values across all four network scenarios. Notably, in the medium and mixed bandwidth scenarios, our DTAAP outperforms the other schemes, attaining the highest QoE levels. Furthermore, in the low bandwidth scenario, our DTAAP achieves the second-best performance, closely approaching the best-performing scheme. These findings underscore the effectiveness of our DTAAP in providing superior user experiences, particularly in challenging network conditions. Additionally, it is observed that the QoE performance gradually degrades as the available bandwidth decreases. In comparison to the results depicted in Fig. 3 (a), it is noteworthy that although the NextOne and Network-based schemes achieve higher QoE performance than our DTAAP, they also exhibit higher bandwidth waste. In particular, the NextOne scheme shows significant bandwidth waste due to their full preloading of the current video. This highlights the trade-off between QoE performance and bandwidth efficiency. While these schemes may provide better user experiences, they come at the cost of increased bandwidth waste.

V. CONCLUSION

In this paper, we have investigated the adaptive preloading problem in short video streaming with the objective of enhancing user QoE and reducing bandwidth waste. We have proposed a novel digital twin-assisted adaptive preloading framework, which can capture user network dynamics and swipe patterns to enable adaptive video preloading. Extensive simulations demonstrate that the proposed scheme can effectively enhance user QoE and reduce bandwidth waste. The proposed DTAAP framework can effectively and efficiently utilize user-specific information to enhance the performance of short video streaming in dynamic network environments.

ACKNOWLEDGMENT

This work was supported in part by the Peng Cheng Laboratory Major Key Project under Grants PCL2023AS1-5, PCL2021A09-B2, and by the Natural Science Foundation of China under Grant 62201311.

REFERENCES

[1] Z. Li, Y. Xie, R. Netravali, and K. Jamieson, “Dashlet: Taming swipe uncertainty for robust short video streaming,” in *Proc. USENIX Symp. Netw. Syst. Des. Implement.*, 2023, pp. 1583–1599.
[2] G. Zhang, J. Zhang, K. Liu, J. Guo, J. Y. Lee, H. Hu, and V. Aggarwal, “DUASVS: A mobile data saving strategy in short-form video streaming,” *IEEE Trans. Serv. Comput.*, vol. 16, no. 2, pp. 1066–1078, 2022.
[3] W. Wu, N. Chen, C. Zhou, M. Li, X. Shen, W. Zhuang, and X. Li, “Dynamic RAN slicing for service-oriented vehicular networks via constrained learning,” *IEEE J. Sel. Areas Commun.*, vol. 39, no. 7, pp. 2076–2089, Jul. 2021.
[4] Y. Zhang, Y. Liu, L. Guo, and J. Y. B. Lee, “Measurement of a large-scale short-video service over mobile and wireless networks,” *IEEE Trans. Mobile Comput.*, vol. 22, no. 6, pp. 3472–3488, 2023.
[5] D. Nguyen, P. Nguyen, V. Long, T. T. Huong, and P. N. Nam, “Network-aware prefetching method for short-form video streaming,” in *Proc. IEEE MMSP Workshops*, 2022, pp. 1–5.
[6] 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 *Proc. IEEE VCIP*, 2020, pp. 13–16.
[7] C. Zhou, Y. Ban, Y. Zhao, L. Guo, and B. Yu, “PDAS: Probability-driven adaptive streaming for short video,” in *Proc. ACM MM*, 2022, pp. 7021–7025.
[8] X. Shen, J. Gao, W. Wu, M. Li, C. Zhou, and W. Zhuang, “Holistic network virtualization and pervasive network intelligence for 6G,” *IEEE Commun. Surveys Tuts.*, vol. 24, no. 1, pp. 1–30, 2022.
[9] Y. Wang, Z. Su, S. Guo, M. Dai, T. H. Luan, and Y. Liu, “A survey on digital twins: Architecture, enabling technologies, security and privacy, and future prospects,” *IEEE Internet Things J.*, vol. 10, no. 17, pp. 14 965–14 987, 2023.
[10] X. Huang, M. Li, W. Wu, C. Zhou, and X. Shen, “Digital twin-assisted collaborative transcoding for better user satisfaction in live streaming,” in *Proc. IEEE ICC*, 2023, pp. 4051–4056.
[11] B. Mao, J. Liu, Y. Wu, and N. Kato, “Security and privacy on 6G network edge: A survey,” *IEEE Commun. Surveys Tuts.*, vol. 25, no. 2, pp. 1095–1127, 2023.
[12] C. Qiao, G. Li, Q. Ma, J. Wang, and Y. Liu, “Trace-driven optimization on bitrate adaptation for mobile video streaming,” *IEEE Trans. Mobile Comput.*, vol. 21, no. 6, pp. 2243–2256, 2020.
[13] 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 *Proc. ACM MM*, 2022, pp. 7084–7088.