DUASVS: A Mobile Data Saving Strategy in Short-Form Video Streaming

Guanghui Zhang, Jie Zhang, *Member, IEEE*, Ke Liu, Jing Guo, Jack Y. B. Lee, *Senior Member, IEEE*, Haibo Hu, *Senior Member, IEEE*, and Vaneet Aggarwal, *Senior Member, IEEE*

Abstract—Fueled by the emerging short video applications (e.g., TikTok), streaming short-form videos nowadays is ubiquitous among mobile users. During the viewing, one common action is to scroll the screen to switch videos, which is a handy operation for the viewers to quickly search for content of interest. However, our empirical measurements reveal that frequent video switching can result in nearly half of the mobile data quota being used for transferring the video data that is never watched. This problem is called data loss in this work. Given the immense cost of the network infrastructure, such a high proportion of data loss is financially tremendous to both mobile users and streaming vendors. To tackle the problem, this study proposes a novel system called Data Usage Aware Short Video Streaming (DUASVS), where a new Integrated Learning is used to capture the characters of past network conditions and then trains intelligent adaptation models to reduce data loss and save data usage. Extensive evaluations show that DUASVS is able to save 70.7%∼83.2% of mobile data usage without incurring any QoE degradation. Moreover, the system exhibits strong robustness, performing consistently over a wide range of network environments as well as video streaming sessions.

Index Terms—Short video streaming, mobile network, data usage, quality-of-experience, video reliability

1 Introduction

Mobile video streaming has seen tremendous growth in the past decade and is now a mainstream application on the mobile Internet. Beginning with the delivery of professionally authored video content (e.g., movies, news), a new trend in recent years was the streaming of user-generated videos to share personal lives. This trend drove the explosive advances of some short-form video applications such as TikTok [1], Douyin [2], and Kwai [3], which have been downloaded over billions of times globally in the past few years [4], [5], [6].

One common operation from these short video applications is known as scrolling the screen. Specifically, with so many genres of videos, it is not surprising that not all the application-recommended videos satisfy the taste of viewers. If a viewer is not interested in the video content currently being played, he/she can scroll the screen at any time to switch to the next one. For viewers, scrolling the screen is such a handy action to promptly search for the content of interest. However, for the ongoing video sessions, if the videos are switched before being completely played back, the already downloaded but unwatched part would be discarded by the player, such that the bandwidth consumed in transferring them would be wasted. At first glance, such bandwidth wastage may not appear to be a severe problem. However, previous studies [4], [5], [6] as well as our empirical measurements (see Section 3) found that it is common for the viewers to switch the short videos frequently during the viewing, which results in an average of 44.2% mobile data losses.

In fact, the cause of the data loss is rooted in the architecture of the short video streaming system. First, it adopts the HTTP-based protocol that transfers data as fast as TCP allows. Whenever the HTTP/TCP throughput is higher than the video bitrate, the video player will keep fetching the short videos following a playlist and then buffer them locally. While such buffering mechanism can effectively avoid the playback rebuffering events as the buffered video

- *Guanghui Zhang is with the Department of Computer Science, Hong Kong Baptist University, Kowloon, Hong Kong and also with the Centre for Advances in Reliability and Safety (CAiRS), Pak Shek Kok, NT, Hong Kong. E-mail: ghzhang@link.cuhk.edu.hk.*
- *Jie Zhang is with the Department of Electronic Engineering and information Science (EEIS), University of Science and Technology of China (USTC), Hefei, Anhui 230026, China and also with the State Key Laboratory of Acoustics, Institute of Acoustics, Chinese Academy of Sciences, Beijing 100190, China. E-mail: jzhang6@ustc.edu.cn.*
- *Ke Liu is with the State Key Laboratory of Computer Architecture, Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China and also with the University of Chinese Academy of Sciences (UCAS), Beijing 101408, China. E-mail: liuke@ict.ac.cn.*
- *Jing Guo is with the Department of Computing, The Hong Kong Polytechnic University, Kowloon, Hong Kong. E-mail: guojing_jane@foxmail.com.*
- *Jack Y. B. Lee is with the Department of Information Engineering, The Chinese University of Hong Kong, Shatin, NT, Hong Kong. E-mail: yblee@ie.cuhk.edu.hk.*
- *Haibo Hu is with the Department of Electronic and Information Engineering, The Hong Kong Polytechnic University, Kowloon, Hong Kong and also with PolyU Shenzhen Research Institute, Kowloon, Hong Kong. E-mail: haihaibo.hu@polyu.edu.hk.*
- *Vaneet Aggarwal is with the School of Industrial Engineering, Purdue University, West Lafayette, IN 47907 USA and also with the School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47907 USA. E-mail: vaneet@purdue.edu.*

Manuscript received 25 July 2021; revised 3 January 2022; accepted 5 February 2022. Date of publication 10 February 2022; date of current version 10 April 2023.
This work was supported in part by the Centre for Advances in Reliability and Safety (CAiRS) Admitted under Grants AIR@InnoHK Research Cluster, in part by the National Natural Science Foundation of China under Grant 62101523, and in part by the Fundamental Research Funds for the Central Universities.
(Corresponding author: Jie Zhang)
Digital Object Identifier no. 10.1109/TSC.2022.3150012

data is able to absorb the bandwidth fluctuation, it would lead to a large amount of mobile data being lost in the presence of frequent video switching. Second, unlike the conventional video-on-demand (VoD) platforms (e.g., YouTube) adopting the bitrate adaptive scheme (c.f. DASH [7]), the short video streaming system encodes the video contents into a fixed low bitrate version (approximately 1 Mbps according to our measurement). In practice, however, such a bitrate version is typically lower than the available HTTP/TCP throughput, and, as a result, the data loss problem is exacerbated.

The mobile data loss can result in substantial financial losses due to the expensive operating costs of the network infrastructure. First, a previous study [8] reported that, the streaming vendors typically spent hundreds of millions of dollars on the CDN bandwidth annually, which means that the 44.2% of mobile data loss (measured in Section 3) wastes at least tens of millions of dollars each year, which is far from trivial. Second, today's mobile data service purchased by users typically has a hard data quota, e.g., 10 GB per month [35]. If data usage exceeds the given quota, users have to purchase additional data at a higher price. In terms of the 44.2% mobile data loss, users in fact consume nearly half of their data quota for transferring the video data that is never watched.

One conventional method to reduce the mobile data loss was to limit the amount of video prefetch data (i.e., the data in the buffer) [9], [10], [11]. Intuitively, if a video player merely prefetches 1s of video data at most, then in the worst case only the 1s worth of data will be lost upon the video switching. However, the video prefetch exists for an important reason – to buffer video data such that the playback can be sustained during the periods of poor network condition. Too little prefetch data will likely lead to more rebuffering events and thus significant QoE degradation, which is an even bigger problem than the data loss [16]. Therefore, a grand challenge posed is how to effectively reduce the mobile data loss while keeping the QoE intact.

This work tackles this challenge by developing a novel system called Data Usage Aware Short Video Streaming (DUASVS), which saves the mobile data usage through the dynamically adaptive control for video prefetch and bitrates. Specifically, our empirical study on real-world streaming traces showed that, in streaming short videos, the network conditions experienced differ significantly across different viewers, which suggests that training one single adaptation model for all the viewers is unlikely to achieve optimal performance [26]. Built upon this insight, we developed the novel Integrated Learning in DUASVS to first segregate the network conditions into different categories, and then automatically generate/train a specialized adaptation model for each category to determine the video bitrate as well as the prefetch threshold for each video. At runtime, whenever a viewer starts a video session, the system will match him/her to a particular category based on his/her past network condition and then apply the corresponding adaptation model for use in online streaming. In this way, DUASVS can enhance the system strength to effectively reduce the data loss in various streaming environments.

Extensive evaluations showed that DUASVS is able to save 70.7%~83.2% of mobile data usage without incurring any QoE degradations, which significantly outperforms the state-of-the-art algorithms in current short video streaming. Moreover, DUASVS exhibits strong robustness, performing consistently across a wide range of networks as well as video sessions. Last but not least, DUASVS can be readily deployed into the real streaming platform, offering an immediate and practical solution to the data loss problem.

The rest of the paper is organized as follows: Section 2 reviews the background and related work; Section 3 conducts an empirical study to measure the mobile data loss; Section 4 presents the design of DUASVS; Section 5 evaluates the performance of DUASVS and compares it to the state-of-the-art algorithms; Section 6 discusses the implementation and deployment of DUASVS, and Section 7 summarizes the study and outlines some future works.

2 Background and Related Works

In the short video streaming system, the streaming server learns the interests of viewers from their previous access records and then pushes relevant genres of videos to them. The videos are downloaded one by one following a playlist and then are played back in a full-screen manner on the client's mobile devices. If the viewer is not interested in the video content currently being played, he/she can switch the video at any time. Since these short video applications (e.g., TikTok, Douyin) were merely launched a couple of years ago, only a few studies have focused on them.

The earliest work was by Chen *et al*. [4] who conducted a measurement study on Douyin [2] and revealed many problems. For instance, they found that a considerable proportion of videos pushed by the recommended system fail to attract the interest of viewers, such that these uninterested videos would be switched/terminated rapidly. In addition, the short videos were encoded with only one fixed bitrate version around 1 Mbps without adopting the bitrate adaptation scheme like DASH. However, while this study pointed out many limitations for the current short video streaming, the authors did not propose any methods to address them.

In another study, He *et al*. [5] found that frequent video switching in watching short videos is a very common behavior among the viewers (consistent with our findings in Section 3). They measured that such behaviors can rapidly drain the buffered video data, resulting in substantial playback rebuffering events. To this end, they proposed a new strategy called LiveClip, which tries to predict the viewer's future switching behavior and estimate the viewing duration of the subsequent videos (i.e., when the videos will be switched). Based on this, their system will then determine the downloading sequence for the follow-up videos to maintain a safe buffer occupancy to avoid rebuffering.

Ran *et al*. [6] developed SSR which breaks the current encoding mechanism for short-form videos (i.e., using one fixed bitrate version) and turns to encode the video content into multiple bitrate versions to perform bitrate adaptation. This system also tries to predict the viewer's future switching behavior, and then allocates high-level bitrate to the videos with longer predicted engagement time and low bitrate to that will be rapidly switched. As a result, the video quality experienced by the viewers can be substantially improved under the limited bandwidth resources.

However, both the systems by He *et al.* [5] and Ran *et al.* [6] rely heavily on the estimation for the viewing duration of the subsequent videos, and their estimation is merely based on the viewer's past behavior. According to our investigation, this is far from enough to achieve high prediction accuracy, as when the viewers switch the videos is also influenced by many other factors, such as their interests in the video content, playback smoothness, and so on [30]. Therefore, while these two systems can somewhat improve the QoE performance, they cannot effectively tackle the data loss problem investigated in this study.

In the conventional VoD streaming, there are some studies to save data usage by limiting the client buffer capacity. For example, Yarnagula *et al.* [9] proposed SARA, which limits the amount of data in the buffer with a pre-defined 20s buffer threshold. The methodology is that when the client buffer occupancy reaches the buffer threshold, the client request for downloading the next segment will be delayed until the buffer occupancy falls below the threshold. Similarly, Chen *et al.* [10] proposed an algorithm that also adopts the buffer threshold but set the value to 30s. In another study by Huang *et al.* [11], the solution is slightly different. They proposed to use the Lyapunov optimization theory to design a dynamic buffer allocation strategy and the goal is to save the total data usage for all the mobile users served by one base station. Although limiting the buffered data is also in principle effective in reducing the data loss in short video streaming, as discussed in Section 1, it can lead to far more rebuffering events, thus degrading the QoE performance significantly.

In comparison, to our knowledge, the proposed DUASVS is the first system so far that can thoroughly tackle the data loss problem in short video streaming while keeping the QoE intact. An early version of DUASVS is proposed in our conference paper [12], where the system is called WAS. This study extends the conference version in three significant aspects.

First, to reduce the data loss, the early version WAS employs two tunable parameters to control the video prefetch and the bitrate selection respectively. It extracts the correlation between the data loss and QoE by capturing the features from the past network environment to reduce the adverse impacts on QoE. However, our investigation showed that the two tunable parameters in WAS are not expressive enough to cover all the features in the past environments such that the resulting performance is inevitable sub-optimal. Therefore, in this work, we turn to using deep reinforcement learning A3C [13] to learn from the past environments, and then train deep neural networks to express the learned features. More details will be elaborated in Section 4.1∼4.2. Moreover, we conducted experiments in Section 5 to quantify the performance improvement of DUASVS over WAS.

Second, the early version WAS suffers from deployment issues in practice. Specifically, as the network condition may differ significantly across different video clients, WAS's parameter optimization is designed as per-viewer basis to match different network environments. Therefore, as the number of viewers (i.e., video clients) increases, the computational overhead increases dramatically. This undoubtedly occupies a large amount of computational resource when the client scales and thus would hinder the large-scale deployment on the real streaming platforms.

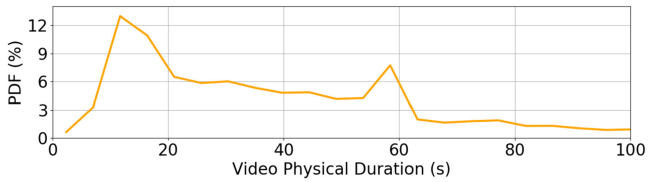

Fig. 1. Probability Density Function (PDF) of video physical duration.

In comparison, in DUASVS, we developed Integrated Learning which first segregates the network conditions into a limited number of categories, and then trains the specialized adaptation model for each category to determine the video bitrate as well as the prefetch threshold for each video. At runtime, the system will match each video session to a particular category based on their network condition and then apply the corresponding adaptation model for use in the online streaming. In this way, DUASVS not only achieves a much lower training overhead, but also has strong robust performance over various streaming environments. More details for the Integrated Learning are provided in Sections 4.3 and 4.4.

Third, the scope of the experiments and performance evaluations has been expanded substantially in this work. While our earlier work already employs TCP throughput trace data for experiments and performance evaluation, this work further expands the scope to incorporate five independent trace sources (see Section 5.1), which include 3G, 4G/LTE, as well as Wi-Fi networks. In addition, these network traces were captured from different locations and served by multiple service providers. The far broader scope of the evaluation environments enables us to gain a better understanding for the behaviors and performance of the streaming algorithms (see Section 5.3).

3 Empirical Study

In this work, we collaborated with an anonymous (short video) streaming vendor who provided us with a streaming trace dataset collected from their commercial video servers. This dataset records three months of video access log over 6 million video sessions from 120 thousand viewers (we call these data as video property trace in the rest of this paper). We have publicized part of formatted trace data in [31], and we will gradually publicize the whole dataset after obtaining the permission from the streaming vendor. Based on the dataset, we conducted an empirical study to investigate the current commercial setup of the short video streaming platform.

3.1 Commercial Setup

Fig. 1 plots the PDF distribution for the video physical duration. We observed that most of the videos (∼82%) in the dataset last less than 60s, which is significantly shorter than that in the conventional VoD services (e.g., YouTube) [15]. Furthermore, there are two prominent peaks at 13s and 57s in the figure, which is because the streaming vendor preconfigured two video duration thresholds (15s and 60s) with the purpose of restricting the maximum physical duration for the videos captured/uploaded from the clients.

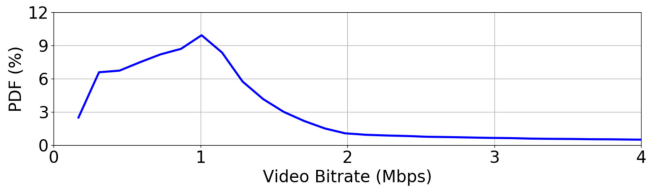

Fig. 2. Probability Density Function (PDF) of video bitrate.

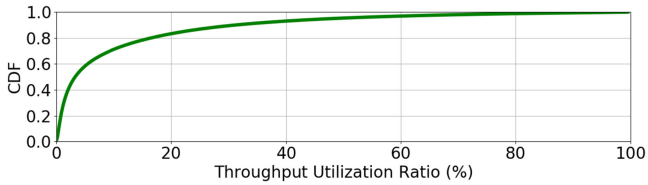

Fig. 3. Cumulative Distribution Function (CDF) of throughput utilization ratio.

Next, we investigated the video encoding bitrate. In Fig. 2, we plotted the PDF distribution for the actual bitrate over different videos. We observed that the bitrate of most videos in the dataset is lower than 2 Mbps, typically surrounding 1 Mbps. This suggests that the videos are encoded with a fixed bitrate version (as opposed to the bitrate adaptive streaming like DASH [7]), where the target encoding bitrate is approximately set at 1 Mbps. With regard to adopting such encoding configurations, the streaming vendor has two practical considerations. On one hand, since the short video applications are merely applicable to mobile devices with limited screen size, the video quality perceived by the viewer is typically higher than that on the PC monitors. On the other hand, the adopted bitrate is generally lower than today's available TCP throughput, so that playback rebuffering can be largely avoided. Therefore, the resulting viewing experience from the adopted encoding bitrate (i.e., about 1 Mbps) is acceptable to most viewers.

However, in fact, such configurations are far from optimal, which can be reflected by the throughput utilization ratio κ_i, defined as

$$\kappa_i = r_i / X_i \tag{1}$$

where r_i is the mean bitrate of video i and X_i is the mean measured throughput during downloading video i. Fig. 3 plots the CDF distribution for κ_i across all the videos. To our surprise, more than 80% of the streamed videos only utilize less than 20% of the available throughput. In this case, the video player often prefetches/buffers a large number of videos prior to its playback schedule. While the buffered video data can compensate for the bandwidth fluctuations to avoid rebuffering, it would lead to a large amount of mobile data loss upon frequent video switching. We will investigate this problem in detail next.

3.2 Mobile Data Loss Measurement

Data loss is caused by the downloaded video data not being watched but discarded by the video player when the video is switched. Therefore, to quantify the data loss, we need to measure the proportion of each video being watched and downloaded [30].

The viewing ratio θ_i is defined as the ratio for the video watched duration V_i to the total video physical duration L_i in video i, i.e.,

$$\theta_i = V_i / L_i \tag{2}$$

Similarly, we define download ratio τ_i that is the ratio for video downloaded duration D_i to the total video physical duration L_i, i.e.,

$$\tau_i = D_i / L_i \tag{3}$$

In Table 1, we calculate the average of the two ratios over all the videos, respectively. It is evident that a significant proportion of videos are switched/terminated early, with an overall average viewing ratio of 54.4%. In comparison, the download ratio is much higher with an overall average of 97.1%. Furthermore, we plot the CDF distributions for the two ratios in Fig. 4 where we can observe a huge gap between the two CDF curves. All these results reveal a fact that a large amount of downloaded video data is not played but discarded by the player, so the mobile data consumed in transferring them is ultimately lost.

Next, we measured the daily amount of data loss from the perspective of the streaming vendor side (their servers on average stream 216 million short videos every day). The data loss amount is calculated from the difference between the video data downloaded and viewed:

$$W_k = \sum_{\forall d_{k,i,j}>0} d_{k,i,j} - \sum_{\forall v_{k,i,j}>0} s_{k,i,j} \frac{v_{k,i,j}}{l_{k,i,j}} \tag{4}$$

where W_k is the data loss at day k, $d_{k,i,j}$, $s_{k,i,j}$, $l_{k,i,j}$, $v_{k,i,j}$ are the video data downloaded, segment size, full segment duration, and viewed segment duration for the segment j in video i, respectively. Furthermore, we compute the data loss ratio, denoted by R, from

$$R = 1 - \sum_{\forall v_{k,i,j}>0} s_{k,i,j} \frac{v_{k,i,j}}{l_{k,i,j}} \Bigg/ \sum_{\forall d_{k,i,j}>0} d_{k,i,j} \tag{5}$$

We summarize the data loss measurement results in Table 1 as well where the results for the videos longer and shorter than 30s are presented separately. There are three

TABLE 1
Video Switching and Data Loss

Video duration	Mean viewing ratio (%)	Mean download ratio (%)	Data loss ratio (%)	Daily data loss amount (Petabyte)
<30s	70.1	99.0	29.6	1.05
≥30s	48.1	96.2	50.1	3.27
All	54.4	97.1	44.2	4.32

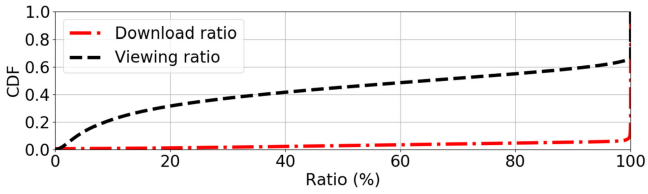

Fig. 4. Cumulative Distribution Function (CDF) of video download ratio and viewing ratio.

observations: First, the most noticeable one is the 44.2% data loss ratio, which is such a tremendous proportion indicating nearly half of the video data delivered being lost. Second, on average, 4.32 Petabytes of mobile data are lost every day. Given the pricing of Amazon CDN [17], such an amount of data loss can consume the streaming vendor nearly $100 million annually which is a striking cost. Third, the longer videos (i.e., $\geq$30s) exhibit more substantial data loss than their shorter counterparts (i.e., <30s), e.g., 50.1% vs. 29.6%. This is because the longer videos are generally switched relatively earlier by the viewers (reflected by the viewing ratios 48.1% vs 70.1%), resulting in a larger amount of prefetch video data being discarded.

Considering the mobile data loss is directly caused by the viewer's decision on the video switching, so we next measured the switching behavior of the individual viewers. We randomly picked three viewers (A, B, and C) from the dataset and calculated their viewing ratio and data loss ratio over a period of six weeks. The results are shown in Table 2. We can see that the three viewers exhibit quite different viewing behaviors as well as the resultant data loss. For example, viewer C's viewing ratio is relatively lower while the data loss is the highest. On the contrary, viewer B has the highest viewing ratio but losses the least amount of data. In comparison, in the view of the time series, all the three viewers' viewing ratios and data loss ratios are relatively consistent from week to week, which suggests that there are no significant changes in their individual behaviors.

From the results in Table 2, we conjecture that the network throughput is likely to have similar characters, because the network condition is also the key factor for the degree of the data loss. To this end, in Table 3, we measured the weekly mean throughput and throughput coefficient-of-variation (CoV) from the same three viewers. As expected, both the mean throughput and CoV differ dramatically across different viewers but exhibit a high degree of consistency over the six weeks. This consistency can be presumably explained by the fact that the geographic location of each viewer is relatively fixed when streaming the short videos day by day (as observed from the dataset), so each of them has a high probability to be served by the same Wi-Fi hotspots or cellular base stations that typically have similar quality of service [15]. Overall, the results offer an insight that the significant network condition differences over different viewers and the potential network environment changes due to the geographic location change should be considered in the system design.

TABLE 2
Viewing Ratio and Data Loss Ratio of Three Viewers Over Six Weeks

Viewers	Two ratios (%)	Weeks 1	2	3	4	5	6
A	VR	57.5	56.7	53.1	60.8	56.7	61.4
	DLR	32.1	34.9	38.2	30.4	34.9	30.1
B	VR	76.7	70.2	78.1	79.4	75.7	78.2
	DLR	22.7	28.9	21.4	20.2	23.3	21.6
C	VR	32.7	31.2	33.0	37.9	31.3	34.7
	DLR	66.5	68.2	65.5	61.6	67.3	63.2

* *"VR" denotes Viewing Ratio (%); "DLR" is Data Loss Ratio (%).*

TABLE 3
Network Conditions of Three Viewers Over Six Weeks

Viewers	Throughput metrics	Weeks 1	2	3	4	5	6
A	MT (Mbps)	5.56	4.11	5.19	7.33	6.86	7.10
	CoV	0.83	0.92	0.74	0.77	0.95	0.99
B	MT (Mbps)	22.6	32.9	23.2	19.6	31.8	26.8
	CoV	0.44	0.39	0.53	0.69	0.59	0.40
C	MT (Mbps)	18.0	18.7	19.2	13.1	10.3	14.2
	CoV	0.37	0.41	0.22	0.41	0.30	0.39

* *"MT" is Mean Throughput (Mbps); "CoV" denotes throughput variations quantified by coefficient-of-variation.*

4 Data Usage Aware Short Video Streaming

Inspired by the findings from Section 3, we develop a novel system called Data Usage Aware Short Video Streaming (DUASVS) to help the users save the data usage in streaming the short-form videos. DUASVS incorporates two key modules: 1) *Integrated Learning* (see Section 4.3) that is to train a set of candidate data-saving aware adaptation models (the structure of the model is described in Section 4.1$\sim$4.2), and 2) *Online Model Selection* (see Section 4.4) that is to select/pick out the most appropriate trained model for use in the streamed videos online.

4.1 Data Saving Mechanisms

Since the mobile data loss is primarily attributed to the downloaded but unwatched video data, the most intuitive solution is to control and limit the video prefetch. Based on this insight, we define a prefetch threshold $\boldsymbol{\beta}_i$ to function in video i. The mechanism is to limit the actual prefetch duration B_i within $\boldsymbol{\beta}_i$:

$$B_i = \min\{\beta_i, L_i\} \tag{6}$$

where L_i is the video physical duration of video i. Once the prefetch duration reaches B_i, the video player will shift to fetching video i+1, and the rest part of video i will be downloaded when it starts to be played back.

As discussed, however, too small a $\boldsymbol{\beta}_i$ will likely lead to substantial rebuffering events, especially in poor network conditions. Therefore, we develop an adaptation model (in Section 4.2) to dynamically tune the value of $\boldsymbol{\beta}_i$ based on the specific network environment. To facilitate the design of the adaptation model, we discretize $\boldsymbol{\beta}_i$ into several candidate values, denoted by

$$\Sigma = \{\beta_{i,0}, \beta_{i,1}, \ldots, \beta_{i,Z-1}\} \tag{7}$$

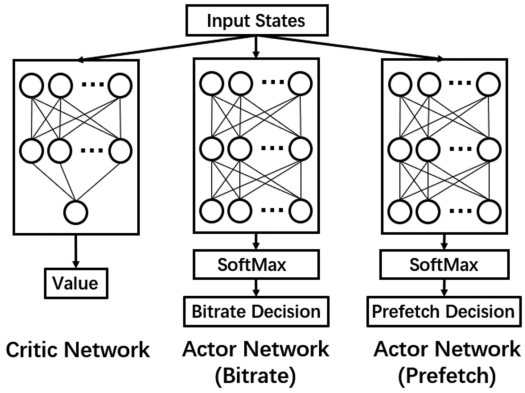

Fig. 5. The adaptation model of DUASVS.

The adaptation model will be introduced in Section 4.2.

Furthermore, as discussed in Section 3, the fixed low encoding bitrate for the short videos is the catalyst for the substantial data loss, so another intuitive method is to make the video bitrate match the TCP throughput so as to slow down the video prefetch and buffering process (intuitively assuming the bitrate is always exactly equal to the available throughput, videos do not need to be prefetched while no rebuffering events will occur).

To this end, we turn to encode the videos into multiple bitrate versions:

$$\Pi = \{r_{i,0}, r_{i,1}, \ldots, r_{i,H-1}\} \tag{8}$$

and then develop an adaptation model (see Section 4.2) to automatically determine the bitrate level to enable the size of the requested video segments to match the changing throughput. In this way, not only the data loss can be reduced, but video quality can be improved.

4.2 Data-usage Aware Adaptation Model

Built upon the two mechanisms in Section 4.1, we define the structure of the adaptation models for the data loss control. Since the learning process of DUASVS is to employ A3C [13] (a state-of-the-art deep reinforcement learning technique), the adaptation model is an Actor-Critic neural network. The structure is illustrated in Fig. 5, which contains three neural networks, namely, Actor Network (Prefetch), Actor Network (Bitrate), and Critic Network. The descriptions for their input states and output actions are summarized in Table 4. We will elaborate on them in the following.

Moreover, it is worth noting that the reason for adopting A3C is its widespread application in the design of the video streaming algorithms [5], [6], [18], [19], [20] as well as the fact that it can achieve quite good performance. Nevertheless, there are still many other candidate machine learning techniques to choose from, such as Q-learning [21], DQN [22], Policy Gradient [23], and so on. We have also implemented them into our cases and compared them to A3C in Section 5.

Input States. On requesting each short video, e.g., video i, the adaptation model takes input states S_i into its neural networks, which contain five independent states: *State 0* is to detect the current network condition. It is a vector of measured throughput where the mean throughput during downloading each of the past K segments is formed as a sample, i.e., $\{c_{i,0}, c_{i,1}, \ldots, c_{i,K\text{-}1},\}$. *State* 1 is the current whole buffer occupancy b_i (at the timepoint on requesting video i) which alerts the potential playback rebuffering events. *State* 2 is the size of the video at the H encoding bitrate levels, denoted by $\{s_{i,0}, s_{i,1}, \ldots, s_{i,H\text{-}1}\}$, which indicates the actual bitrate variation [24]. *State* 3 is the video physical duration, denoted by L_i, which is to incorporate the viewer behaviors in watching different durations of videos (c.f. Section 3.2). *State* 4 is the determined bitrate for the last requested video i-1, denoted by $r_{i\text{-}1}$, which works as a prompter for the video quality fluctuations.

TABLE 4
Inputs and Outputs of the Actor-Critic Network

Input states S_i	State 0	The measured TCP throughput in past K segments, $\{c_{i,0}, c_{i,1}, \ldots, c_{i,K\text{-}1},\}$.
	State 1	Current buffer occupancy b_i.
	State 2	The size of the video at different bitrate levels, $\{s_{i,0}, s_{i,1}, \ldots, s_{i,H\text{-}1}\}$.
	State 3	Video physical duration L_i.
	State 4	The video bitrate of the last requested video $r_{i\text{-}1}$.
Output	Actor network (Prefetch)	The probability of prefetch threshold $\boldsymbol{\beta}_{i,z}$ being selected at state S_i, i.e., $P(\boldsymbol{\beta}_{i,z} \mid S_i)$, $z = 01,\ldots,Z\text{-}1$.
	Actor network (Bitrate)	The probability of bitrate level $r_{i,h}$ being selected at state S_i, i.e., $P(r_{i,h} \mid S_i)$, $h = 01,\ldots,H\text{-}1$.
	Critic network	The estimated value at state S_i to calculate TD error, denoted by $v(S_i)$.

Actor-Critic Network and Output Actions. The neural network structure shown in Fig. 5 is defined as a combination of one-dimensional CNNs and fully connected networks to build the deterministic policy for the prefetch threshold as well as the video bitrate. Specifically, the Actor Network (Prefetch) specifies the mapping from any certain states to the prefetch threshold. Upon receiving the input state S_i, it will determine an action, i.e., the prefetch threshold, for the next video, where the decision is based on the deterministic policy function represented by the neural network. Its output is a SoftMax activation function [25] to normalize and formulate the probability distribution over different candidate values of prefetch threshold, i.e., (7), denoted by

$$P(\beta_{i,z}|S_i), z = 0, 1, \ldots, Z-1 \tag{9}$$

representing the probability of prefetch threshold $\boldsymbol{\beta}_{i,z}$ being selected upon state S_i. The final decision will be randomly made based on the probability distribution.

The Actor Network (Bitrate) has exactly the same structure as the Actor Network (Prefetch) except for the output:

$$P(r_{i,h}|S_i), h = 0, 1, \ldots, H-1 \tag{10}$$

which denotes the probability of bitrate level $r_{i,h}$ being selected at the input state S_i. Similarly, the system will then randomly choose a candidate bitrate from (8) based on the probability distribution.

The two actor networks determine the actions for the next video, while the Critic network outputs a value, i.e., $v(S_i)$, which is used to calculate the TD (Temporal Difference) error [13], [37] based on the observed rewards. The TD error

will then be fed back to the neural networks to further promote the training for achieving better performance.

4.3 Integrated Learning

Once the structure of the adaptation model is defined, it does not require any manual tuning anymore, but its internal neuron weights will be optimized through offline training. Specifically, the adaptation model will be exposed to the training streaming environment and the actions (i.e., (9) and (10)) will be determined based on the observed states (i.e., S_i). The neuron weights will then be evolved according to the resultant reward computed from past experiences. After the training, a fixed mapping between the input states and the outputs actions will be formed in the adaptation model, which is the deterministic policy function. At runtime, the adaptation model will be used for online streaming where the real streaming environment will provide the input states and then the adaptation model will determine the action automatically based on the deterministic policy function.

We have introduced the input states and the output actions of the adaptation model in Section 4.2, and in this section, we will introduce the definition of the reward function and how to build the streaming environment to train the adaptation model.

Reward Function. After each action decision, the video with the determined bitrate level will be requested and streamed in the streaming environment, and meanwhile, the prefetch duration of each video will be limited within the determined prefetch threshold. The reward can then be calculated through the resulting streaming performance to express the goodness of the past action decisions. Therefore, the definition of the reward function will directly guide the system's training direction.

In order to enable the adaptation model with data usage saving awareness as well as maintaining high QoE, we define the reward function by combining QoE function U (as a positive utility) and data loss amount W (as a penalty utility) into a unified utility function to make the problem become a utility-maximization problem, i.e.,

$$F = U - \mu \times W \tag{11}$$

where U can be any format of conventional QoE functions, e.g., [18], [24], [27], typically including video bitrate, rebuffering, etc. (a sample of the QoE function is shown by (22) in Section 5.1), and $\boldsymbol{\mu}$ is the penalty weight for the data loss W.

In practice, however, reducing the data loss may affect/impair the QoE (e.g., too short a prefetch duration may incur rebuffering), but due to the complexity of the network environment, it is unclear how to appropriately normalize the utilities between the two metrics in (11) to avoid the QoE impairment. Therefore, we adopted a different approach where we let the penalty weight $\boldsymbol{\mu}$ to be tunable to control the performance tradeoff. For instance, a smaller value of $\boldsymbol{\mu}$ will train an adaptation model with better QoE performance but less significant data usage saving, and vice versa. Based on this principle, we define a total of P values of $\boldsymbol{\mu}$, i.e., $\{\boldsymbol{\mu}_p \,|\, p = 01,\ldots,P\text{-}1\}$, to generate P different reward functions:

$$F_p = U - \mu_p \times W, p = 0, 1, \ldots, P-1 \tag{12}$$

which will be used to train p different adaptation models separately. We will introduce the training later.

Network Condition. To speed up the training, DUASVS employs virtual streaming [18], [30], [32] to emulate the streaming environment, where the adaptation model is executed in a simulated streaming environment replayed by TCP throughput trace data captured from real networks [28], [29] and video property trace introduced in Section 3. Specifically, for downloading each video, the download time is first calculated based on the determined video bitrate as well as the throughput. Then the video duration will be accumulated to the buffer occupancy to simulate the completion of the video downloading. For simulating the video playback, the simulator maintains a playback buffer and the consumption of the buffered data is affected by the viewing duration (i.e., the time consumed before the viewer switching the video) and the video physical duration [5]. During the whole streaming process, the simulator keeps tracking and recording each rebuffering event (i.e., the scenarios where the video download time is longer than the current buffer occupancy) for post-analysis.

One key insight from Section 3 is that the network conditions differ significantly across different viewers (see Table 3), which suggests that training one single adaptation model for all the viewers is unlikely to achieve optimal performance [26]. To this end, in this work, DUASVS segregates the network environments into different classes and then trains the specialized adaptation model for each class.

According to our measurement upon the throughput trace data as well as the experimental results from the previous studies [26], [32], two network features, namely, mean throughput and throughput variation (quantified by coefficient-of-variation (CoV)), can well reflect different types of network conditions, so we use them to segregate the networks. We denote the mean throughput of video i as X_i, that can then be mapped to the discrete throughput level x_i through a linear quantization policy:

$$x_i = \min(\lfloor X_i/\Delta_0 \rfloor, M-1) \tag{13}$$

where Δ_0 is the quantization step size and M is the maximum number of the mean throughput class. Similarly, the throughput CoV can be quantified by:

$$y_i = \min(\lfloor Y_i/\Delta_1 \rfloor, N-1) \tag{14}$$

where Y_i is the throughput CoV of video i, Δ_1 is the quantization step size, and N (e.g., $= 5$) is the total CoV class number. Therefore, with the throughput level x_i and CoV level y_i, the trace data of all the videos, i.e., E_i, $i = 01,\ldots,I$, can be divided into $M \times N$ network classes:

$$\begin{aligned} C_{m,n} = \{E_i | x_i = \langle m \rangle, y_i = \langle n \rangle, \forall i\}, \\ m = 0, 1, \ldots, M-1, n = 0, 1, \ldots, N-1 \end{aligned} \tag{15}$$

Algorithm Training. Finally, with the set of network classes, i.e., $\{C_{m,n} |\ m = 01,\ldots,M\text{-}1,\ n = 01,\ldots,N\text{-}1\}$ and reward functions, i.e., $\{F_p |\ p = 01,\ldots,P\text{-}1\}$, DUASVS will execute the offline training:

$$A_{m,n,p} = G(C_{m,n}, F_p),$$
$$m = 0, 1, \ldots, M-1, n = 0, 1, \ldots, N-1, p = 0, 1, \ldots, P-1 \quad (16)$$

where function $G(.)$ represents the learning algorithm A3C [13] and $A_{m,n,p}$ is a set of trained adaptation models in which each member is an Actor-Critic neural network (the structure is defined in Section 4.2). Note that A3C was originally proposed by Google DeepMind, and in this work, we adopted the same training workflow as their original study (i.e., [13], which includes the detailed pseudocode).

As discussed, reducing the data loss may affect/impair the QoE. Thus, to avoid the QoE degradation, we will further filter the trained adaptation models to keep the actual QoE achieved at the same level as that without DUASVS (e.g., the current commercial setup) [15], [30]. Specifically, we denote the original QoE (i.e., unaffected by DUASVS) as U_{ori}. The system will then sequentially filter the adaptation models trained with different reward functions $\{F_p \mid p = 01,\ldots,P\text{-}1\}$ (c.f. (12)) with the objective of minimizing the amount of data loss and at the same time restricting the actual achieved QoE larger than or equal to U_{ori},

$$\begin{aligned} &\min_p W_{m,n,p} \\ &\text{s.t. } U_{m,n,p} \geq U_{ori}, \\ &m = 0, 1, \ldots, M-1, n = 0, 1, \ldots, N-1 \end{aligned} \quad (17)$$

where $W_{m,n,p}$ and $U_{m,n,p}$ are the data loss amount and QoE achieved by the corresponding adaptation models, i.e., $A_{m,n,p}$. In practice, all the metrics in (17), e.g., $W_{m,n,p}$, $U_{m,n,p}$ and U_{ori}, can be measured/recorded in the training because it is virtual streaming. Finally, the filtered adaptation models will be used for online streaming:

$$\Lambda = \left\{ A^*_{m,n} | m = 0, 1, \ldots, M-1, n = 0, 1, \ldots, N-1 \right\} \quad (18)$$

The filtering for the adaptation model in (17) only targets avoiding any degradation of QoE performance. In practice, however, different viewers can have different habits and interests, so they may have very different preferences for the data saving and QoE. For example, some viewers expect to achieve more significant data saving even at the cost of some QoE, while some cannot afford to lose any bit of QoE. To this end, we further modified the system and proposed an enhanced version of DUASVS that offers explicit options for the viewers to choose. The details are elaborated in Appendix A.1, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TSC.2022.3150012.

4.4 Online Model Selection

The Integrated Learning (introduced in Section 4.3) trains/outputs a set of candidate adaptation models, i.e., (18), where different members are specialized for different network conditions, i.e., (15). Thus, at runtime, the system's major task is to select out the most appropriate model for use in the online streamed videos.

As discussed in Section 3, the network environment over different viewers varies greatly. In addition, even for a specific viewer, when its geographical location changes, the network environment can be different. Therefore, the system needs to select the specialized adaptation model for each video session (a video session denotes one process where a viewer uses the short video app to stream short videos from beginning to end, and it typically contains multiple streamed videos). However, how to judge the network condition for a video session is a significant challenge because while the throughput series can be known ahead in the training as the throughput trace data are given, it cannot be obtained before streaming the actual videos online.

To address the challenge, we proposed a method to *estimate* the network condition of the current video session online. Specifically, during streaming the session, the videos are requested consecutively one by one, and the requesting time interval is typically low due to the video physical duration being short. Thus, the network condition can be well expressed by the throughput series in downloading the initial a few videos, e.g., $i = 01,\ldots,K\text{-}1$, and the mean throughput X can be estimated by

$$X = avg\left(\left\{ \frac{s_i}{t_i} \middle| i = 0, 1, \ldots, K-1, \forall t_i > 0 \right\} \right) \quad (19)$$

where s_i and t_i are the video size and download time of the initially requested video i, and function $avg(.)$ is to compute the arithmetic mean (a previous study [34] measured that arithmetic mean is outstanding for the throughput estimation). Similarly, the throughput CoV can be calculated by

$$Y = std\left(\left\{ \frac{s_i}{t_i} \middle| i = 0, 1, \ldots, K-1, \forall t_i > 0 \right\} \right) \Big/ X \quad (20)$$

where function $std(.)$ is to compute the standard deviation of the throughput. At last, the estimated throughput features X and Y will then be discretized by the a linear quantization policy, i.e., (13) and (14), to determine the adaptation model.

5 Performance Evaluation

In this section, we evaluate DUASVS's effectiveness in addressing the data loss problem and compare its performance to the state-of-the-art streaming algorithms.

5.1 Experiment Setup

To evaluate the performance of the streaming algorithms in realistic streaming settings, we employed trace-driven simulations where the simulator executes the algorithms over a simulated streaming environment reproduced by trace data. There are two kinds of trace data adopted, namely TCP throughput trace and video property trace.

TCP throughput trace emulate the network conditions, which were captured from real production networks. To cover different kinds of network properties, we employed five independent trace sources [28], [29] and their statistics are summarized in Table 5. Among them, #1∼#3 are captured from cellular networks (3G and 4G/LTE), #4 and #5 are from Wi-Fi networks. In the rest of the paper, unless stated otherwise, the throughput trace used for evaluation consists of all the data from #1∼#5. Video property trace include video viewing duration and video physical duration, which are extracted from the commercial dataset

TABLE 5
Statistics of Five TCP Throughput Trace Sources

Features	#1	#2	#3	#4	#5
Mean throughput (Mbps)	5.97	1.21	10.1	3.12	4.43
Variation (CoV)	0.44	0.83	0.52	0.77	0.58
Network type	3G	3G	LTE	Wi-Fi	Wi-Fi
Collection location	L1	L2	L3	L4	L5
Service provider	S1	S2	S3	S4	S5

TABLE 6
Prefetch and Bitrate Settings of the Algorithms

Algorithms	Prefetch threshold (s)	Bitrate version (Mbps)
DUASVS	2∼60 (tunable)	0.2∼8.0 (tunable)
DUASVS-$\boldsymbol{\beta}$	2∼60 (tunable)	1.0
WAS	2∼60 (tunable)	0.2∼8.0 (tunable)
CA	N/A	1.0
LiveClip	N/A	1.0

introduced in Section 3. On the whole, these two kinds of trace data cover a total of 6 million videos over 120 thousand viewers. In this work, 50% of the data were used for training and the remaining 50% were for testing (c.f. Appendix A.2, available in the online supplemental material).

We implemented a total of five streaming algorithms for the evaluation. A summary for the algorithm settings is listed in Table 6.

1) CA – the current commercial setup. Its detailed configurations have been introduced in Section 3.
2) DUASVS – the algorithm proposed in this work. Its adaptation model tunes the prefetch threshold (see (7)) within 2s $\sim$ 60s and the available bitrate (see (8)) within 0.2 Mbps $\sim$ 8.0 Mbps. In Integrated Learning, the throughput traces are classified by the bin size $\Delta_0 = 1$ Mbps and $\Delta_1 = 0.2$, and the network class numbers are $M = 10$ and $N = 5$ (see (13) and (14)). A total of 14 penalty weights are adopted in (12) ranging from 0.01 to 1.8. U_{ori} in (17) is equivalent to the QoE achieved by CA. For the learning process of A3C, the entropy weight [13] is randomly initialized within 1∼5, then is linearly degraded in a gradual manner until ultimately reaches 0.01 after 100000 iterations.
3) DUASVS-$\boldsymbol{\beta}$ – a variant of DUASVS. In contrast to DUASVS, DUASVS-$\boldsymbol{\beta}$ only has the Actor Network (Prefetch) to tune the prefetch threshold $\boldsymbol{\beta}$ and it is not available to tune the bitrate. The motivation to evaluate this variant is to quantify the efficacy of controlling the video prefetch in DUASVS.
4) WAS [12] – the earlier version of DUASVS proposed in our conference paper. Its differences from DUASVS have been described in Section 2.
5) LiveClip [5] – an existing short video streaming algorithm, which dynamically adjusts the download sequence for the follow-up videos based on predicting when the viewer will switch the videos. LiveClip can neither tune the video prefetch nor the bitrate.

For the performance metric, in addition to quantifying the data loss using the metrics defined by (4)∼(5), we also compute the data saving ratio by comparing the differences in the amount of data loss between CA (i.e., the commercial setup) and other algorithms:

$$\varsigma = \sum_{\forall i} (\widehat{W}_i - W_i) \Big/ \sum_{\forall i} \widehat{W}_i \tag{21}$$

where $\widehat{W}_i$ is the data loss produced by CA in video i, and W_i is that by other algorithms. Moreover, the QoE performance is evaluated using an existing QoE function proposed by Mao *et al*. [18]:

$$U = \sum_{i=1}^{I} \log\left(\frac{r_i}{r_{\min}}\right) - \sum_{i=1}^{I-1} \left|\log\left(\frac{r_{i+1}}{r_{\min}}\right) - \log\left(\frac{r_i}{r_{\min}}\right)\right| - 2.66 \times \sum_{i=1}^{I} \sigma_i \tag{22}$$

where r_i is the bitrate selected for video i, r_{min} is the minimum available bitrate 0.2 Mbps (see Table 6), the logarithmic function log(.) is to quantify the video quality, $\boldsymbol{\sigma}_i$ is the rebuffering duration during downloading video i, and I is the total number of videos. The QoE function in (22) is the default one in the evaluation, and we also test some other QoE functions in Section 5.4.

5.2 Data Loss and QoE Performance

We first evaluate the system's effectiveness in reducing data loss as well as its impacts on QoE. Table 7 compares the QoE performance, the daily amount of data loss, data loss ratio, and data saving ratio over the five streaming algorithms. It is not surprising that CA, i.e., the current commercial setup, incurs the largest amount of data loss among all the algorithms due to its naïve configurations.

The performance of LiveClip is also far from satisfactory. Although it is specifically designed for short video streaming, the QoE obtained is even 6.11% worse than CA. In addition, while LiveClip can save 7.63% of mobile data, such a ratio is too trivial to solve the data loss problem. The root reason is that LiveClip only concerns the prediction of the short-term viewing behavior, such as the viewing duration of the subsequent a few videos, but this is heavily influenced by

TABLE 7
Data Loss and QoE Performance Over Five Streaming Algorithms

	QoE	Daily data loss amount (Petabyte)	Data loss ratio (%)	Data saving ratio (%)
DUASVS	0.82	0.99	12.1	77.1
DUASVS-$\boldsymbol{\beta}$	0.81	3.06	35.7	29.2
WAS	0.82	1.81	20.7	58.4
CA	0.81	4.32	44.2	0
LiveClip	0.76	3.99	42.1	7.63

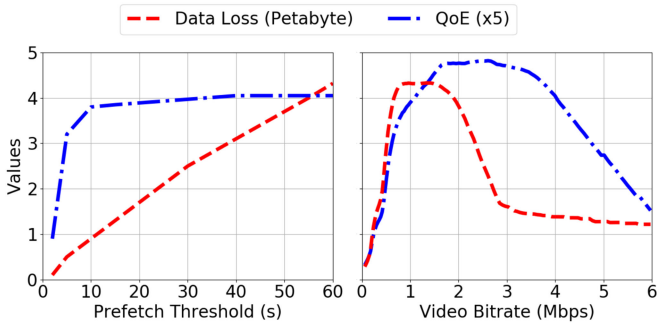

Fig. 6. The effectiveness of the two data saving mechanisms (the left chart is for prefetch threshold, and the right is for bitrate).

the viewer's interest in the video content, rebuffering events, and many other factors, resulting in the low accuracy of the prediction.

In comparison, the rest three algorithms, DUASVS, DUASVS-$\boldsymbol{\beta}$, and WAS, do not work from the perspective of viewer behavior prediction, but make decisions by analyzing real network conditions. As a result, the three algorithms are able to achieve more significant data saving ratios (29.2%∼77.1%) while maintaining the QoE intact.

Nevertheless, there are also huge performance differences among the three algorithms. Most remarkably, DUASVS achieves the lowest data loss, and the data saving ratio reaches 77.1%, outperforming all other algorithms substantially. DUASVS-$\boldsymbol{\beta}$ only tunes the prefetch threshold without adapting the video bitrate, such that it only saves 29.2% of mobile data. This result well reflects the effectiveness of the two data saving mechanisms proposed in Section 4.1, and we will further quantify their efficacy later.

For WAS, although both of the two data saving mechanisms are incorporated, its data saving ratio is still merely 58.4% that is much worse than DUASVS. This demonstrates that compared to purely tuning the parameters (i.e., WAS), the neural network model adopted in DUASVS can better express the features of the streaming environment, enabling a fuller release of the efficacy potential of the two mechanisms.

To uncover the principle of the two data saving mechanisms (Section 4.1), we thus quantify their efficacy separately. Specifically, we disabled one mechanism in DUASVS and tuned the other one to see its impacts on the data loss and QoE. The left chart of Fig. 6 shows the results for tuning the prefetch threshold. We observed that as the prefetch threshold decreases from 60s to 40s, the QoE keeps constant while the data loss is reduced from 4.32 to 3.11 Petabyte. Then from 40s to 10s, both the data loss and QoE decrease linearly, but the dropping slope of data loss is much larger than QoE. This is why DUASVS-$\boldsymbol{\beta}$ can reduce the data loss at the cost of little or even no QoE loss.

The right chart is for tuning the video bitrate. Compared to the performance of the fixed bitrate setting, i.e., under the bitrate around 1 Mbps, changing the bitrate not only can reduce the data usage but can also further improve the QoE. Thus, by jointly using the two mechanisms, the potential QoE improvement can provide a margin to reduce the data loss while avoiding the QoE degradation. As a result, DUASVS can achieve much more significant performance than DUASVS-$\boldsymbol{\beta}$.

TABLE 8
The Efficacy of Data Usage Saving Over Five Throughput Trace Sources

Algorithms		Data loss amount (PB)		Data saving ratio (%)
		CA	DUASVS	
Throughput trace sources	#1	5.01	1.02	79.6
	#2	3.95	0.91	76.7
	#3	5.24	1.04	80.2
	#4	4.14	0.92	77.8
	#5	4.07	0.96	76.4

5.3 DUASVS Over Various Streaming Scenarios

The Integrated Learning (introduced in Section 4.3) trains a set of adaptation models where different ones are specialized for different network conditions. In this section, we will evaluate its contribution to the system's robustness over the various network environments.

In Table 8, we compare the data loss performance of DUASVS and CA over the five throughput trace sources. It is evident that DUASVS achieves a substantial data saving ratio across all the sources, ranging from 76.4% to 80.2%. Moreover, compared to CA, DUASVS achieves a more consistent data loss amount across different sources, although the data loss amount is still slightly larger at the sources with higher mean throughput (e.g., #1 and #3). This is inevitable as the bitrate adaptation model selects higher video bitrate at higher throughput levels and hence the larger video segment size would naturally lead to more data loss. On the whole, DUASVS exhibit strong robustness performance over different network environments.

To deeply investigate the rationale, we further evaluated the dynamics of the data saving mechanisms with respect to different throughput trace sources. We list in Table 9 the prefetch and bitrate decisions made by the adaptation model under the five sources, where the decisions are quantified by the mean prefetch threshold and the throughput utilization (defined by (1)), respectively. It is clear that both the prefetch and bitrate decisions vary substantially across the five sources. For example, in the trace sources with more stable and higher throughput (e.g., #1 and #3), the prefetch threshold is tuned to a smaller value while the throughput utilization is larger. This indicates that DUASVS is able to exploit the (better) network condition with abundant throughput to increase the algorithm's bitrate selection aggressiveness while shortening the duration of prefetch data, so that the data loss can be largely reduced in the case of frequent video switching. On the contrary, in the trace sources with poor network

TABLE 9
The Decisions From DUASVS's Adaptation Model Across Five Throughput Trace Sources

Network characters and adaptation model decisions	Throughput trace sources				
	#1	#2	#3	#4	#5
Throughput (Mbps)	5.97	1.21	10.1	3.12	4.43
Coefficient of Variation (CoV)	0.44	0.83	0.52	0.77	0.58
Mean prefetch threshold (s)	10.8	17.3	10.2	15.4	13.9
Throughput utilization (%)	61.2	42.5	66.8	49.1	55.7

TABLE 10
Data Saving Ratio (%) Across QoE Functions

QoE Functions	QoE_1 [18]	QoE_2 [18]	QoE_3 [24]	QoE_4 [27]
DUASVS	77.1	79.8	83.2	70.7
DUASVS-β	29.2	33.5	32.1	25.2
WAS	58.4	66.7	60.3	51.3
LiveClip	7.63	7.63	7.63	7.63

TABLE 11
Data Saving Ratio Achieved By DUASVS with Different Learning Technologies

Learning Techniques	A3C [13]	Q-learning [21]	DQN [22]	Policy Gradient [23]
Saving Ratio (%)	77.1	46.5	64.1	60.5

conditions (e.g., #2), the prefetch threshold is tuned to be larger and the throughput utilization is kept lower to avoid playback rebuffering and QoE degradation.

Overall, these results suggest that using the trace data from a sufficiently wide spectrum of network conditions in the Integrated Learning, DUASVS is able to effectively save the data usage over a broad range of network environments.

5.4 Sensitivity Analysis

To see if the above observations are consistent under different QoE metrics, we evaluate the algorithms with three more QoE functions, i.e., $QoE_2 \sim QoE_4$ [18], [24], [27], in addition to the QoE_1 defined by (22). The data saving ratios under the four QoE functions are summarized in Table 10. We observed very similar results obtained over different QoE functions, where DUASVS saves 70.7%∼83.2% of the mobile data usage and outperforms all other algorithms consistently. It is worth noting that the performance of LiveClip is exactly the same over different cases, because its training does not incorporate any QoE function.

In this work, we employed A3C [13] to train the adaptation model for DUASVS, while there are many other candidate machine learning techniques to choose from, e.g., Q-learning [21], DQN [22], Policy Gradient [23], and so on. To this end, we implemented DUASVS with other learning techniques to see the performance. The results are shown in Table 11 where we compare the data saving ratio over different learning techniques. We can see that DUASVS trained with A3C performs the best and outperforms other three approaches by 20.3% $\sim$ 65.8%. Q-learning is the worst performing one due to its rather simple learning model (i.e., a tabular form [21]) that limits the choice of the state space. Despite that DQN and Policy Gradient have much better performance than Q-learning, there is still a large gap to A3C. Moreover, as opposed to the multi-core training in A3C [13], all the comparison learning techniques can only exploit one single CPU core, so their training convergence speed is much slower.

6 Real Implementation and Deployment

Considering the significant differences in the configuration and capacity of the mobile devices (e.g., CPU, memory, etc.) [36], DUASVS is best implemented as a server-side streaming system. Specifically, the streaming servers (e.g., CDN) can be easily extended to record the streaming trace data used for the offline training when it delivers the video content to the clients. With the recorded trace data, the training can then be conducted through the virtual streaming (as introduced in Section 4).

After the training, the trained adaptation models will be delivered to a specialized adaptation server to decide the prefetch threshold and video bitrate for each video at runtime. Then, the adaptation server is to send the bitrate and prefetch decision to the client via the streaming metadata (e.g., the m3u8 playlist in Apple's HLS protocol [14]), and the client will request the corresponding video content from the conventional streaming server. We have implemented a real prototype for DUASVS to validate its feasibility in the practical streaming platform. The details are described in Appendix A.3, available in the online supplemental material.

Moreover, another practical issue is that although DUASVS employs Integrated Learning to train a set of adaptation models for different network environments and switches the trained model online whenever the network conditions change, it is conceivable that the offline training may still need to be re-conducted when encountering completely new network environments that have not been incorporated into the training, e.g., the future 6G networks with a much higher bandwidth limit. Nevertheless, we argue that the frequency of the re-training will be very low as long as the Integrated Learning can be exposed to a sufficiently wide spectrum of network environments. Overall, DUASVS offers an immediate and practical scheme for the current short video streaming platform.

7 Summary and Future Work

This work investigates the mobile data saving topic in today's short video streaming. In the empirical study, we found that the frequent video switches can incur significant mobile data loss. To this end, we developed the novel system DUASVS to tackle this problem, which not only can substantially save the data usage by 70.7%∼83.2%, but also avoid any adverse impacts on QoE. Therefore, it offers a practical tool to help the viewers and the streaming vendor save the mobile data usage in streaming the short-form videos.

This study is only the first step in this direction and there are many potential problems to be investigated in the future. First, as we discussed, predicting the viewer switching behavior is a very difficult topic as the behavior is affected by many factors. Nevertheless, if we can obtain some insights from these influential factors and then develop a predictor to achieve accurate prediction for the future viewing duration, then data loss can be completely avoided. This could be a fruitful direction for future work.

Second, while the proposed DUASVS has been evaluated in various streaming environments, it has not yet been tested and validated through a large-scale deployment. We are currently in discussions with the streaming vendor about the deployment issues, and after their permission, we will deploy DUASVS into their commercial servers that have millions of daily visitors. We believe that the far broader scope

of the evaluation environments will enable us to gain a better understanding for the system's behaviors and performance.

Acknowledgments

The authors wish to thank the associate editor and the anonymous reviewers for their insightful comments in improving this paper.

References

[1] TikTok. [Online]. Available: https://www.tiktok.com
[2] Douyin. [Online]. Available: https://www.douyin.com
[3] Kwai. [Online]. Available: http://www.kwai.com/
[4] Z. Chen, Q. He, and Z. Mao, "A study on the characteristics of douyin short videos and implications for edge caching," in *Proc. ACM Turing Celebration Conf. - China*, 2019, pp. 1–6.
[5] J. He, M. Hu, and Y. Zhou, "LiveClip: Towards intelligent mobile short-form video streaming with deep reinforcement learning," in *Proc. 30th ACM Workshop Netw. Oper. Syst. Support Digit. Audio Video*, 2020, pp. 54–59.
[6] D. Ran, Y. Zhang, W. Zhang, and K. Bian, "SSR: Joint optimization of recommendation and adaptive bitrate streaming for short-form video feed," in *Proc. IEEE Int. Conf. Mobility Sens. Netw.*, 2020, pp. 418–426.
[7] T. Stockhammer, "Dynamic adaptive streaming over HTTP: Standards and design principles," in *Proc. ACM Conf. Multimedia Syst.*, 2011, pp. 133–144.
[8] L. Chen, Y. P. Zhou, and D. M. Chiu, "Smart streaming for online video services," *IEEE Trans. Multimedia*, vol. 17, no. 4, pp. 485–497, Apr. 2015.
[9] H. K. Yarnagula, P. Juluri, S. K. Mehr, V. Tamarapalli, and D. Medhi, "QoE for mobile clients with segment-aware rate adaptation algorithm (SARA) for DASH video streaming," *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 15, no. 2, pp. 1–23, Jun. 2019.
[10] X. Chen, T. Tan, and G. Cao, "Energy-aware and context-aware video streaming on smartphones," in *Proc. IEEE Int. Conf. Distrib. Comput. Syst.*, 2019, pp. 861–870.
[11] G. Huang, W. Gong, and B. Zhang, "An online buffer-aware resource allocation algorithm for multiuser mobile video streaming," *IEEE Trans. Veh. Technol.*, vol. 69, no. 3, pp. 3357–3369, Mar. 2020.
[12] G. Zhang, K. Liu, H. Hu, and J. Guo, "Short video streaming with data wastage awareness," in *Proc. IEEE Int. Conf. Multimedia Expo*, 2021, pp. 1–6.
[13] V. Mnih, A. P. Badia, M. Mirza, and A. Graves, "Asynchronous methods for deep reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 1928–1937.
[14] "Apple HTTP live streaming," Jul. 2021. [Online]. Available: https://developer.apple.com/streaming/
[15] Y. Liu and J. Y. B. Lee, "Post-streaming rate analysis - a new approach to mobile video streaming with predictable performance," *IEEE Trans. Mobile Comput.*, vol. 16, no. 12, pp. 3488–3501, Dec. 2017.
[16] Y. Lin and H. Shen, "AutoTune: Game-based adaptive bitrate streaming in cloud-based hybrid VoD systems," *IEEE Trans. Serv. Comput.*, vol. 12, no. 4, pp. 519–533, Jul./Aug. 2019.
[17] AmazonCDN Pricing. [Online]. Available: https://aws.amazon.com/cloudfront/pricing/?nc1=h_ls
[18] H. Mao, R. Netravali, and M. Alizadeh, "Neural adaptive video streaming with pensieve," in *Proc. ACM SIGCOMM*, 2017, pp. 197–210.
[19] S. Sengupta, N. Ganguly, S. Chakraborty, and P. De, "HotDASH: Hotspot aware adaptive video streaming using deep reinforcement learning," in *Proc. IEEE 26th Int. Conf. Netw. Protoc.*, 2018, pp. 165–175.
[20] C. Wang, J. Guan, T. Feng, N. Zhang, and T. Cao, "BitLat: Bitrate-adaptivity and latency-awareness algorithm for live video streaming," in *Proc. 27th ACM Int. Conf. Multimedia*, 2019, pp. 2642–2646.
[21] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Mach. Learn.*, vol. 8, no. 3-4, pp. 279–292, May 1992.
[22] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015.
[23] T. Lillicrap *et al.*, "Continuous control with deep reinforcement learning," *Comput. Sci. Mach. Learn.*, Sep. 2015, *arXiv:1509.02971*.
[24] G. Zhang, R. Ngan, and J. Lee, "EmuStream - An end-to-end platform for streaming video performance measurement," *IEEE Access*, vol. 8, pp. 669–680, 2019.
[25] C. M. Bishop, *Pattern Recognition and Machine Learning*. Berlin, Germany: Springer, 2006.
[26] G. Zhang and J. Lee, "Ensemble adaptive streaming - A new paradigm to generate streaming algorithms via specializations," *IEEE Trans. Mobile Comput.*, vol. 19, no. 6, pp. 1346–1358, Jun. 2020.
[27] X. Yin, A. Jindal, V. Sekar, and B. Sinopoli, "A control-theoretic approach for dynamic adaptive video streaming over HTTP," in *Proc. ACM SIGCOMM*, 2015, pp. 325–338.
[28] "Mobile throughput trace data," [Online]. Available: https://github.com/-Zhang-Guanghui-Nick/TCPtrace/releases/tag/TCP-trace
[29] H. Riiser, P. Vigmostad, C. Griwodz, and P. Halvorsen, "Commute path bandwidth traces from 3G networks: Analysis and applications," in *Proc. ACM Multimedia Syst. Conf.*, 2013, pp. 114–118.
[30] G. Zhang, K. Liu, H. Hu, V. Aggarwal, and J. Lee, "Post-streaming wastage analysis - A data wastage aware framework in mobile video streaming," *IEEE Trans. Mobile Comput.*, to be published, doi: 10.1109/TMC.2021.3069764.
[31] "Video property trace data," [Online]. Available: https://github.com/Zhang-Guanghui-Nick/TCPtrace/releases/tag/Short-video-trace
[32] Z. Akhtar, Y. S. Nam, and R. Govindan, "Oboe: Auto-tuning video abr algorithms to network conditions," in *Proc. ACM SIGCOMM*, 2018, pp. 44–58.
[33] "An improved version of dummynet," [Online]. Available: https://github.com/mclab-cuhk/netmap-ipfw
[34] Y. Liu and J. Lee, "An empirical study of throughput prediction in mobile data networks," in *Proc. IEEE GLOBECOM*, 2015, pp. 1–6.
[35] "Mobile data plan of China mobile in Hong Kong," [Online]. Available: https://www.hk.chinamobile.com/en/corporate_infomation/Service_Plans/
[36] D. Wang *et al.*, "Adaptive wireless video streaming based on edge computing: Opportunities and approaches," *IEEE Trans. Serv. Comput.*, vol. 12 no. 5, pp. 685–697, Sep./Oct. 2019.
[37] H. V. Seijen, A. R. Mahmood, P. M. Pilarski, M. C. Machado, and R. S. Sutton, "True online temporal-difference learning," *J. Mach. Learn. Res.*, 2016, vol. 17, no. 1, pp. 5057–5096.

Guanghui Zhang received the MS degree in electronic science and technology from Peking University in 2016 and the PhD degree in information engineering from The Chinese University of Hong Kong in 2020. He is currently a research assistant professor with the Department of Computer Science, Hong Kong Baptist University. From 2020 to 2021, he was a postdoctoral fellow with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, and with the Centre for Advances in Reliability and Safety, The Hong Kong Polytechnic University. His research interests include networking system, multimedia system, and machine learning.

Jie Zhang (Member, IEEE) was born in Anhui Province, China, in 1990. He received the BSc (with Hons.) degree in electrical engineering from Yunnan University, Yunnan, in 2012, the MSc degree (with Hons.) in electrical engineering from Peking University, Beijing, China, in 2015, and the PhD degree in electrical engineering from the Delft University of Technology, Delft, The Netherlands, in 2020. He is currently an assistant professor with the National Engineering Laboratory for Speech and Language Information Processing, Faculty of Information Science and Technology, University of Science and Technology of China, Hefei, China. His current research interests include multi-microphone speech enhancement, sound source localization, binaural auditory, speech recognition, and speech processing over wireless (acoustic) sensor networks. He was the recipient of the Best Student Paper Award for his publication at the 10th IEEE Sensor Array and Multichannel Signal Processing Workshop (SAM) in Sheffield, U.K.

Ke Liu received the BEng and PhD degrees in information engineering from the Chinese University of Hong Kong, Hong Kong, in 2008 and 2013, respectively. From 2017 to 2018, he was a postdoc scholar with the School of Industrial Engineering, Purdue University, IN, USA. He is currently an associate professor with the Advanced System Group under the Key Laboratory of Computer System and Architecture, Institute of Computing Technology, Chinese Academy of Science, Beijing, China, where he participated in the research of protocol optimization and cloud computing.

Jing Guo received the master's degree in electronics and communications engineering from the Harbin Institute of Technology in 2017. She is currently a research assistant with the Department of Computing, The Hong Kong Polytechnic University. From 2017 to 2021, she was a software engineer with Bank of Communications, China. Her research interests include communications and networking, video streaming, and machine learning.

Jack Y. B. Lee (Senior Member, IEEE) received the BEng and PhD degrees in information engineering from the Chinese University of Hong Kong, Shatin, Hong Kong, in 1993 and 1997, respectively. He is currently an associate professor with the Department of Information Engineering, Chinese University of Hong Kong. His research interests include multimedia communications systems, mobile communications, protocols, and applications. He specializes in tackling research challenges arising from real-world systems. He works closely with the industry to uncover new research challenges and opportunities for new services and applications. Several of the systems research from his lab have been adopted and deployed by the industry.

Haibo Hu (Senior Member, IEEE) is currently an associate professor with the Department of Electronic and Information Engineering, Hong Kong Polytechnic University and the programme leader of BSc (Hons.) in information security. He has authored or coauthored more than 80 research papers in refereed journals, international conferences, and book chapters. His research interests include cybersecurity, data privacy, Internet of Things, and adversarial machine learning. As a principal investigator, he was the recipient of more than 20 million HK dollars of external research grants from Hong Kong and mainland China as of year 2020. He was in the organizing committee of many international conferences, such as ACM GIS 2021, 2020, IEEE ICDSC 2020, IEEE MDM 2019, DASFAA 2011, DaMEN 2011, 2013 and CloudDB 2011, and in the programme committee of dozens of international conferences and symposiums. He was the recipient of a number of titles and awards, including the IEEE MDM 2019 Best Paper Award, WAIM Distinguished Young Lecturer, ICDE 2020 Outstanding Reviewer, VLDB 2018 Distinguished Reviewer, ACM-HK Best PhD Paper, Microsoft Imagine Cup, and GS1 Internet of Things Award.

Vaneet Aggarwal (Senior Member, IEEE) received the BTech degree in electrical engineering from the Indian Institute of Technology, Kanpur, India, in 2005, and the MA and PhD degrees in electrical engineering from Princeton University, Princeton, NJ, USA, in 2007 and 2010, respectively. From 2010 to 2014, he was a senior member of the Technical Staff Research with AT&T Labs Research, Bedminster, NJ, USA. He was an adjunct assistant professor with Columbia University, NY, from 2013 to 2014, and an adjunct professor with IISc Bangalore, India, from 2018 to 2019. He is currently a faculty of Purdue University, West Lafayette, IN, USA. His current research interests include communications and networking, video streaming, cloud computing, and machine learning. He was the recipient of the Princeton University's Porter Ogden Jacobus Honorific Fellowship in 2009, AT&T Vice President Excellence Award in 2012, AT&T Senior Vice President Excellence Award in 2014, 2017 Jack Neubauer Memorial Award recognizing the Best Systems Paper published in *IEEE Transactions on Vehicular Technology*, and 2018 Infocom Workshop HotPOST Best Paper Award. He is on the editorial board of *IEEE Transactions on Communications*, *IEEE Transactions on Green Communications and Networking*, and *IEEE/ACM Transactions on Networking.*