Noname manuscript No.
(will be inserted by the editor)

Optimizing Power Consumption of Mobile Devices for Video Streaming over 4G LTE Networks

Jingyu Zhang[1] · Zhi-Jie Wang[2] · Zhe Quan[3] · Jian Yin[2] · Yuanyi Chen[1] · Minyi Guo[1]

zhangzhang@sjtu.edu.cn, cszhijwang@yahoo.com, quanzhe@hnu.edu.cn, issjyin@mail.sysu.edu.cn, cyyxz@sjtu.edu.cn, guo-my@cs.sjtu.edu.cn

Received: date / Accepted: date

Abstract While the video streaming technique and 4G LTE networks are developing rapidly, the advancement of the battery technology, however, is relatively slow. Existing works have paid much attention to understanding the power consumption in general 4G LTE networks, while few attention has been paid to reducing the power consumption for online video streaming in 4G LTE networks. Our previous work based on the real experimental platform showed that, the saving room in the network part is large, the transmission pattern and the number of RRC tails could be promising for optimizing the power consumption. In this paper, we attempt to develop efficient optimization strategies to save the energy cost of mobile devices for online video streaming over 4G LTE networks. This problem is clearly important and also interesting, while it is challenging, due to various reasons such as the uncertainty of users' behavior modes. To alleviate the challenges, we suggest a self-adaptive method that can allow us to adjust various parameters dynamically and efficiently, so as to achieve a relatively small energy consumption. We give the rigorous theoretical analysis for the proposed method, and conduct extensive experiments to validate its effectiveness. The experimental results show that the proposed method consistently outperforms the classical method as well as other competitors adapted from existing methods.

[1] Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, China
[2] Guangdong Key Laboratory of Big Data Analysis and Processing, School of Data and Computer Science, Sun Yat-Sen University, Guangzhou, China
[3] College of Computer Science and Electronic Engineering, Hunan University, Changsha, China

Keywords Power efficiency · online video streaming · user behavior · 4G LTE networks · mobile devices

1 Introduction

In recent several years, the mobile industry has been growing rapidly [1, 2]. Video streaming as one of the most important big data applications has received much attention in mobile industry [2–5]. In 2016 video dominates the mobile traffic and takes 50% of the total traffic (around 4.25 ExaBytes) [2]. It is predicted that video will account for around 75% of mobile data traffic (around 34.5 ExaBytes) in 2022 [2]. In addition, mobile devices are arousing the increasing demand of video streaming, since users can experience video streaming services on a mobile broadband network with different mobile devices (e.g., mobile PCs, smartphones) [2]. Recently video streaming has already been applied into 4G LTE networks [6–8], and particularly the video streaming technique and 4G LTE networks have been developing at an amazing speed [1, 3].

Although the video streaming technique and 4G LTE networks are developing rapidly, the advancement of the battery technology, however, is relatively slow [9]. Existing works have paid attention to understanding the power consumption in general 4G LTE networks [10–13], while few attention has been paid to reducing the power consumption for online video streaming over 4G LTE networks. Clearly, mobile users usually care about the battery capacity of mobile deceives when they watch videos. Our previous work [14] based on the real experimental platform showed that, the saving room in the network part is large, and meanwhile the transmission pattern and the number of *radio resource control* (RRC) tails could be promising for optimizing the power consumption. Yet, no targeted algorithms and/or techniques are discussed for saving the energy cost of mobile devices (for online video streaming over 4G LTE networks). In this paper, we focus on video streaming in 4G LTE networks, and attempt to develop efficient optimization strategies to address this issue.

There are many challenges needing to be addressed: (1) Intuitively, one can use a larger buffer, it shall immediately bring us the benefit in terms of energy consumption. Yet, imagine if the behavior of a user u is in the "unstable" mode (simply speaking, this model refers to that, user tends to drag the process bar of the video player), video segments (downloaded in the buffer prefetching phase) are usually being cleaned without watching them; this shall incur serious bandwidth waste (note: this waste actually also consumes a lot of energy). Then, how to control the buffer size efficiently? (2) Our previous work [14] has shown that the number of RRC tails could be promising for optimizing the power consumption. An immediate idea is to merge multiple *segment fetching sessions*[1] together. Essentially, existing techniques (see e.g., [8]) can be directly used to merge the segment fetching sessions, whereas they ignore users' behaviors. That is, the "merged" video segments could be discarded without watching them, incurring serious bandwidth waste (again, this actually also consumes much power). Then, how to merge the video segment fetching sessions wisely? (3) One could argue that existing works (see e.g., [6]) have suggested different methods for different users' behavior modes. Yet, the users' behavior modes are assumed to be available beforehand in their work. In the usual case, we could not know users' behavior modes beforehand. Then, how to reduce the power consumption when users' behavior modes are unavailable beforehand? (A more detailed analysis is covered in Section 4.)

The challenges above highlight the need for developing more competitive approaches. To this end, this paper suggests a self-adaptive method. The central idea of our method is to exploit the continuous video watching time to predict users' behavior modes, and then dynamically adjust corresponding parameters, so as to consume less energy. The difficulty in designing our method is to construct a systematic model that can "balance" various parameters. These parameters together with our strategies cooperatively contribute to the reduction of power consumption. To summarize, we made the following main contributions.

1. We reveal the major challenges in reducing the power consumption for online video streaming over 4G LTE networks. These challenges highlight the need for studying more competitive methods.
2. We propose a self-adaptive method to save power consumption. Our idea could be also useful for addressing other issues related to energy consumption over video streaming.
3. We give the rigorous theoretical analysis for our proposed method. The analysis verifies the feasibility and effectiveness of our method, from the theoretical perspective.
4. We also conduct extensive experiments to demonstrate the effectiveness of our proposed method. Compared against other competitors, the performance of our method is closest to that of the "ideal" prototype.

In the subsequent section, we review previous works related to ours. Section 3 introduces some basic concepts. Section 4 formally describes our problem and reveals the major challenges. Section 5 presents our proposed method and gives the rigorous theoretical analysis. Experimental results are covered in Section 6, and we conclude this paper in Section 7.

2 Related work

In this section we review previous works by categorizing them into three categories.

Power consumption in wireless networks and/or in IoTs. In the literature a lot of effort has been made to study the power/energy consumption in wireless networks [15–21]. For example, Zhu *et al.* [16] presented the design and implementation of a house-build experimental platform, for the energy management and exploration on wireless sensor networks. In [17] an online algorithm with low computational complexity and deployment overhead was proposed for multi-channel wireless networks. Wang *et al.* [15] presented an analytical formula for estimating power consumption in large-scale wireless sensor networks. In [20] a distributed algorithm for increasing the lifetime of wireless sensor networks was presented. A recent survey [22] covered the energy-efficiency oriented traffic offloading in wireless network, and another survey [23] discussed *general* power control issues in wireless sensor networks. Most of prior works in these literature focused on energy consumption of *sensors*, and assumed to be in the "general" wireless networks environment. These works

[1] When a user u enjoys the online video streaming service, the video streaming player downloads the video segments periodically. This implies that many *downloading sessions* shall be involved in this process; and usually the "downloading session" is called "segment fetching session".

did not investigate the power consumption characteristics for video streaming in 4G LTE networks. In contrast, the concern of this paper is to reduce the power consumption for *online video streaming* over *4G LTE* networks.

On the other hand, many papers [24–30] studied the power consumption in *internet of things* (IoTs). For example, Balasubramanya *et al.* [25] proposed a refined DRX mechanism to reduce the power consumption in IoTs. In [24] a power control and resource block allocation scheme was proposed for reducing the power consumption in IoTs. Also, most of works in these literature did not cover the power consumption for online video streaming, whereas it is the focus of this paper.

Power consumption related to online video streaming. There are a large bulk of works [31,32,6,33–36] that studied the power consumption related to online video streaming. For example, Ukhanova *et al.* [32] presented an analysis on the power consumption of video data transmission in 3G mobile wireless networks. In [31] a seamless high-quality HTTP adaptive streaming algorithm was presented, which considers the energy consumption of a mobile device over heterogeneous wireless networks. Hoque *et al.* [33] proposed a download scheduling algorithm based on crowd-sourced watching statistics to save energy in wireless video streaming. In [35] the problem of resource allocation for video multicast in 4G wireless systems was addressed. Sheu *et al.* [36] discussed scalable-video multicast for WiMAX relay networks. In [37] a hybrid-stream model was proposed to improve the data processing and network overload for video analysis. Compared with these works, our work is different from theirs in several points at least: (i) these works did not cover the LTE network environments while our paper mainly focuses on the power consumption related to video streaming in *4G LTE networks*; and (ii) these works did not address the challenge related to the uncertainty of users' behavior modes, while it is one of focuses in this paper.

Power consumption in LTE networks. We also realize that many researchers studied the power consumption in LTE networks [10,11,38,12,13,39]. For example, Deng *et al.* [38] realized that, keeping the mobile device's radio in the "Active" mode is power-consuming. Their method reduces the energy consumption based on this observation, and their study is in the 3G LTE wireless network environment. In [11] a deployment tool was developed for optimizing the power consumption. Huang *et al.* [12] studied the network performance of many types of mobile networks (including LTE networks). In [13] a novel energy efficient MAC scheme for LTE-Advanced networks was proposed. In summary, these works focused on power consumption in the "general" LTE networks; the specific online video scenarios are not discussed, and so they are clearly different from our work.

Besides, there are many other related works, e.g., *network architectures for data or video delivery* [40,41], *video/data transmission and processing* [42–46], *live video streaming delivery optimization* [47], *online mobile services* [48,49], *user-perceived Quality-of-Experience in Internet video applications* [50,51]. These works are also different from ours, and could be complementary to our work.

3 RRC states, video streaming, and buffer working scheme

In this section, we review some basic concepts, for ease of understanding the rest of the paper.

RRC states. In 4G LTE networks there are two main *radio resource control* RRC states: RRC idle and RRC connected [52]. When there is no mobile data transmission in LTE networks, the RRC state is in the *idle mode*. Usually, before transferring any application data, user equipments (e.g., smartphones) shall change from the RRC idle mode to RRC connected mode immediately. Yet, after finishing the transmission, even if no new data transmission happens, the mobile client still stays at the RRC connected state within a time duration. The status in such a time interval refers to the *RRC tail*. Usually, the RRC connected mode consumes much more power than the RRC idle mode. Note that, in an online video session, there may have many transitions in terms of RRC state. As pointed out in our previous study [14], the number of RRC tails makes impact on the power consumption of mobile devices.

Online video streaming and timing scheme. Video streaming is one of mainstream video playing mechanisms. The most common implementation of video streaming is the HTTP-based online video streaming, in which the video file (from the video source) shall be split into many short segments (or chunks), and encoded to the desired delivery format [53]. Note that, it uses a *media representation description* (MPD) file to describe the video chunks information (such as the accessible segments and corresponding timings). The video chunks are carefully encoded without gaps, and so the video chunks can be played back as a seamless video.

For the video streaming, a widely used timing scheme involves two levels [38,12,53,14]: (i) the video streaming traffic level, see the bottom of Figure 1; and (ii) the RRC level, see the top of Figure 1. At level (i), there is an interactive process between the mobile client

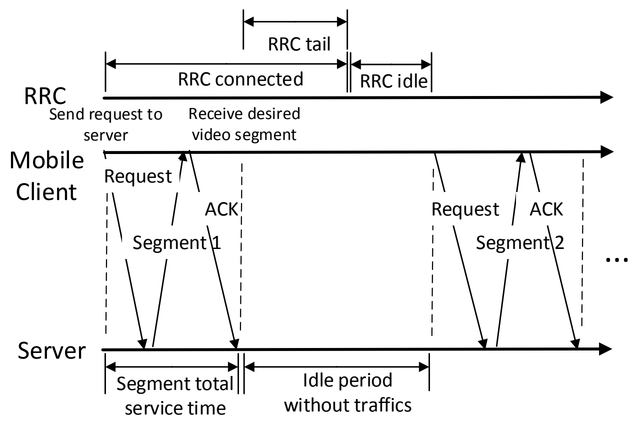


Fig. 1 Timing scheme of video streaming.

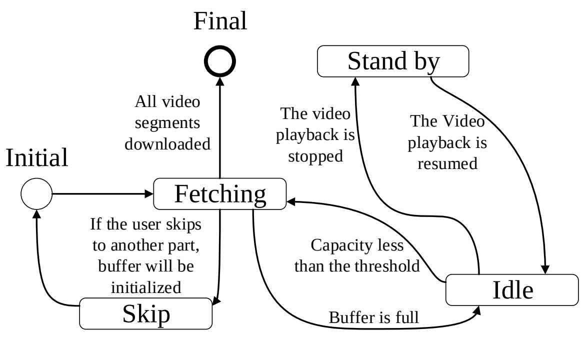

Final
Stand by
All video segments downloaded
The video playback is stopped
The Video playback is resumed
Initial
Fetching
If the user skips to another part, buffer will be initialized
Capacity less than the threshold
Idle
Skip
Buffer is full

Fig. 2 Working scheme of video buffer.

and the server. The mobile client sends periodically the video segment request to the video server. Once the video server receives such a request, it sends back the requested segment to the mobile client. When the requested segment is transmitted successfully, the mobile client sends the ACK message to the video server, and then enters the idle period, in which no network traffic is produced. On the other hand, at the level (ii), the RRC state is in the RRC connected state during the segment transmission. As mentioned earlier, after finishing the segment transmission, the mobile client still stays at this state within a time duration (before entering the RRC idle state mode).

Buffer working scheme. To avoid the video playing delay, the *video buffer* is usually employed for video segments downloading and storage [7,8]. Figure 2 illustrates a general picture for the working scheme of the video buffer.

Initially, a user starts a video streaming session, and the video buffer enters into the "fetching" state. In this state, the video segments (from the HTTP video server) are continuously downloaded into the buffer. When the buffer is full, the buffer transits to the "idle" state. In the idle state, if the user pauses the video playing, the buffer shall shift to the "stand by" state. In contrast, when the video playing is resumed, the state transits back into the idle state. Note that, in the idle state the video segments that have been played back, shall be removed. Once the size of the buffered video is below a specific threshold, the idle state shall shift back to the fetching state, and new video segments are to be downloaded into the buffer.

Remark that, if the user skips to another part of the video (e.g., when he/she wants to watch the latter part of a video), the buffered video segments shall be discarded (since the limit of buffer size); and the buffer shall be filled with new video segments.

4 Problem definition and analysis

In this section we start by formulating the settings of our problem, followed by the formal definition of our problem, and finally we analyze the problem and reveal the major challenges.

Problem settings. Given a 4G LTE network $\mathbb{N}$, assume that a user u with a mobile device (e.g., smartphone) d, on which a video player p is deployed. In addition, assume that the video segments Γ are encoded and stored on an internet-accessible video server s. Let n be the number of total video segments, and f_s be the file size of a *video segment* (i.e., a chunk). Also, assume that the user u requests the video segments on the video server s via $\mathbb{N}$, using mobile device d. The video segments are downloaded and stored in the video buffer b of mobile device d. Without loss of generality, assume that the maximum number of video segments contained in the buffer is b_m, and the buffer threshold used to trigger another segment fetching session is b_t. The downloaded video segments Γ' shall be played back using the video player p (deployed on the mobile device d). For the online video user u, assume that one of two behavior modes below could be involved:

– *Stable mode.* In this mode, the user neither skips to another part of the current video, nor quits the current video. In other words, the user is likely to watch video for a relative long time.
– *Unstable mode.* In this mode, the user tends to skip to another part of the current video, or quits the current video playing even if the current video does not finish. In other words, the continuous time for watching video is relative short. (Once the "skip" happens, the buffer shall clean the current buffered video segments and restart.)

Note that, in the entire video playing process, the video buffer b involves two phases:

– *Buffer prefetching.* In this phase, a set of consecutive video segments are prefetched and filled into the video buffer b. Usually, when the user skips to another part of the current video (or, the user starts a new video), the buffer is in this phase, and it lasts for some time, say τ_{bp}.

– *Buffer feeding*. In this phase the mobile client requests the video segments, based on the specific demand. Usually, when the video is playing back, the buffer downloads the “requested” video segments periodically.

Without loss of generality, let $\overline{v}_p$ (reps., $\overline{v}_f$) be the average download speed in the buffer prefetching (resp., feeding) phase. In addition, let ζ be the total energy consumption of the mobile device for network activities, ζ_o be the average power consumption when playing back the video *offline*, and $\zeta_{\tau_{(\cdot)}}$ be the average power consumption during $\tau_{(\cdot)}$ (e.g., $\zeta_{\tau_{ri}}$ denotes the average power consumption during the RRC idle τ_{ri}). For ease of reference, the notations used frequently are summarized in Table 1.

Table 1 Notations used frequently

Item	Description
ζ	the total energy consumption of d for network activities
ζ_o	the average power consumption when playing back video offline
$\zeta_{\tau_{(\cdot)}}$	the average power consumption during $\tau_{(\cdot)}$
n	the number of total video segments
n_i	the number of video segments of the ith part in unstable mode
b_m	the maximum number of video segments contained in b
b_t	the threshold to trigger video segment fetching session
$\overline{v}_p$	the average download speed in buffer prefetching phase
$\overline{v}_f$	the average download speed in buffer feeding phase
f_s	the file size (of a *video segment*, i.e., chunk)
τ_{bp}	the buffer prefetching time
τ_s	the total service time of a segment
τ_i	the idle period with no traffic
τ_{rt}	the RRC tail time duration
τ_{ri}	the RRC idle time duration
τ_{sd}	the playing back time duration of a video segment

Problem statement. As mentioned earlier, a user u could be one of two behavior modes: *stable mode* and *unstable mode*. For these two modes, the energy consumption are different. Observe that, in the usual case the behavior mode of a user u is unknown beforehand. In what follows, we first formulate the energy consumption for these two modes, and then present our problem formally.

First, consider the *stable mode*, one can have the following:

$$\begin{aligned}\zeta =&\int_0^{\tau_{bp}} \zeta_{\tau_{bp}}\Delta_t + \frac{n-b_m}{b_m-b_t}(\int_0^{\tau_s}(\zeta_{\tau_s}-\zeta_o)\Delta_t + \int_0^{\tau_i}(\zeta_{\tau_i}-\zeta_o)\Delta_t)\\ =&\frac{b_m}{\overline{v}_p}\zeta_{\tau_{bp}} + \frac{n-b_m}{b_m-b_t}(\frac{f_s}{\overline{v}_f}(\zeta_{\tau_s}-\zeta_o) + (\tau_{sd}-\frac{f_s}{\overline{v}_f})(\zeta_{\tau_i}-\zeta_o))\end{aligned} \tag{1}$$

where the part before the first “+” refers to the power consumption in the buffer prefetching phase, while the “remaining” part refers to the energy consumption in the buffer feeding stage. In particular, the latter is composed of two sub-parts: (i) the energy consumption of network transmission; and (ii) the energy consumption of network idle period (see the part after the second “+” in Equation 1). Notice that, Step 2 in the above equation essentially is utilizing the video segment size and the average speed at the corresponding stage to replace the integral operation. Hereafter, we process it in the same way, unless otherwise stated.

Observe that τ_i equals the sum of τ_{rt} and τ_{ri} (c.f., Figure 1). Thus, Equation 1 can be rewritten as follows.

$$\begin{aligned}\zeta =&\frac{b_m}{\overline{v}_p}\zeta_{\tau_{bp}} + \frac{n-b_m}{b_m-b_t}(\frac{f_s}{\overline{v}_f}(\zeta_{\tau_s}-\zeta_o) + \tau_{rt}(\zeta_{\tau_{rt}}-\zeta_o)\\ &+ (\tau_{sd}-\frac{f_s}{\overline{v}_f}-\tau_{rt})(\zeta_{\tau_{ri}}-\zeta_o))\end{aligned} \tag{2}$$

We proceed to consider the *unstable mode*. In this mode, users tend to skip to another part of the video. Assume, without loss of generality, that the number of skips is $i-1$; naturally, video playing shall be divided into i parts. Denote by n_i the number of video segments in the ith part of video playing. Then, one can have

$$\begin{aligned}\zeta =&i\times\int_0^{\tau_{bp}} \zeta_{\tau_{bp}}\Delta_t + \frac{n_1+\cdots+n_i-b_m\times i}{b_m-b_t}(\int_0^{\tau_s}(\zeta_{\tau_s}-\zeta_o)\Delta_t\\ &+\int_0^{\tau_i}(\zeta_{\tau_i}-\zeta_o)\Delta_t)\\ =&i\times\frac{b_m}{\overline{v}_p}\zeta_{\tau_{bp}} + \frac{n_1+\cdots+n_i-b_m\times i}{b_m-b_t}(\frac{f_s}{\overline{v}_f}(\zeta_{\tau_s}-\zeta_o)\\ &+(\tau_{sd}-\frac{f_s}{\overline{v}_f})(\zeta_{\tau_i}-\zeta_o))\end{aligned} \tag{3}$$

Similarly, the above equation can be further rewritten as follows (since $\tau_i = \tau_{rt} + \tau_{ri}$, recall Figure 1).

$$\begin{aligned}\zeta =&i\times\frac{b_m}{\overline{v}_p}\zeta_{\tau_{bp}} + \frac{n_1+\cdots+n_i-b_m\times i}{b_m-b_t}(\frac{f_s}{\overline{v}_f}(\zeta_{\tau_s}-\zeta_o)\\ &+\tau_{rt}(\zeta_{\tau_{rt}}-\zeta_o) + (\tau_{sd}-\frac{f_s}{\overline{v}_f}-\tau_{rt})(\zeta_{\tau_{ri}}-\zeta_o))\end{aligned} \tag{4}$$

Then, one can construct the followings:

$$\zeta = \begin{cases} \models \ of\ Eq.\ 2, & \text{if the stable mode} \\ \models \ of\ Eq.\ 4, & \text{if the unstable mode}\end{cases}$$

where $\models$ denotes the right-hand of an equation. Specifically, our goal is to obtain a small value in terms of ζ.

Problem analysis. To understand the hardness of our problem, this section analyzes it in detail, and reveals the major challenges.

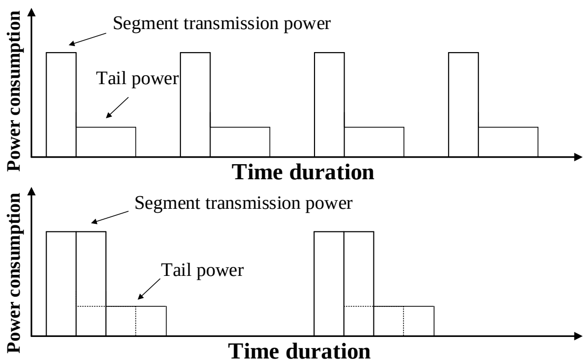


Fig. 3 Illustration of merging segment fetching sessions.

▷ *Buffer size*. Intuitively, to minimize ζ, one may decrease the value of $\frac{n-b_m}{b_m-b_t}$ (resp., $\frac{n_1+\cdots+n_i-b_m\times i}{b_m-b_t}$) for Equation 2 (resp., Equation 4). This can be achieved by setting b_m (resp., b_t) to a large (resp., small) value. Since b_m is depended on the buffer size, one can obtain a larger value of b_m by increasing the buffer size.

Essentially, the larger buffer size can allow more video segments to be downloaded in the buffer prefetching phase; this way, the number of subsequent segment fetching sessions (in the buffer feeding phase) is naturally reduced. On the other hand, the buffer prefetching phase (compared with the buffer feeding phase) can usually achieve higher speed for segments downloading [14], so a larger buffer size also makes positive contribution on saving the entire energy consumption.

Although a larger buffer size is with many merits, a too large buffer size incurs also troubles. Imagine if the behavior of a user u is in the unstable mode, video segments downloaded in the buffer prefetching are usually being cleaned without watching them; this shall incur serious bandwidth waste (note: this waste actually also consumes a lot of energy). In this case, a small buffer size could reduce the bandwidth waste, whereas it shall incur more power consumption if it is too small. Recently, [7] proposed a dynamic cache management strategy, which can adjust the buffer size according to the user's watching logs. This method, however, could be invalid for new videos, since no watching logs are available. Also, this method cannot work well when the behavior of a user is in the unstable mode. To this step, a natural question raises: *how to control the buffer size efficiently*?

▷ *The number of RRC tails*. Our previous study [14] has shown that the number of RRC tails could be promising for optimizing the power consumption. As we know, each segment fetching session is followed by an RRC tail. To decrease the number of RRC tails, an obvious method is to merge more segment fetching sessions together (c.f., Figure 3). This can be achieved by using the push technology mentioned in [8]. However, such a method ignores user's behavior. Imagine if the behavior of a user u is in the unstable mode, the "merged" video segments could be discarded without watching them, incurring serious bandwidth waste (again, this actually also consumes much power). Then, *how to merge the video segment fetching sessions wisely such that the number of RRC tails can be reduced while keeping the bandwidth waste low*?

▷ *Different behavior modes*. From the above analysis, one can easily realize that, in the stable mode a larger buffer size and merging more segment fetching sessions are favourable; in contrast, in the unstable mode a smaller buffer size and merging few segment fetching sessions are beneficial to the reduction of the power consumption. Essentially, [6] suggested different methods for different users' behavior modes, in which the users' behavior modes are assumed to be available beforehand. In the usual case, we could not know users' behavior modes beforehand. Then, *how to reduce the power consumption when users' behavior modes are unavailable beforehand*?

5 Our solution

To address the challenges mentioned earlier, we propose a self-adaptive mechanism for saving the power consumption. Our method provides a unified framework for different behavior modes; also, it can adjust the buffer size and merge video segment fetching sessions dynamically and flexibly. In what follows, we first describe our method at a high level, and then discuss important components in detail.

Overview. Generally speaking, our method consists of two main steps. First, a set of parameters are initialized and then it prefetches a set of video segments to fill the buffer. Our key contribution in the first step is the design of various parameters; these parameters together with our strategies (developed in the second step) shall collaboratively contribute to the reduction of energy consumption. Second, we dynamically adjust the parameters based on our proposed strategies. Our strategies shall guide us to fetch remaining video segments flexibly and wisely in the video segment feeding phase. The central idea of our method is to exploit the continuous video watching time to predict users' behavior modes, and then dynamically adjust corresponding parameters, so as to consume less energy. The intuition behind our idea is that, if a user watches the video for a relative long time, it is more likely to continuously watch the whole video without skips.

Self-adaptive mechanism. The self-adaptive mechanism needs to adjust several parameters according to various scenarios. The difficulty in designing this method is to construct a systematic model that can "balance" various parameters.

One of important elements is b_t, which is the threshold to trigger video segment fetching (recall Table 1). For the conventional online video player, if the buffer b is not full, another video segment fetching session shall be triggered. That is, the default value of b_t is equal to b_m-1, where b_m refers to the maximum number of video segments contained in b (recall Table 1). Instead, in our design we adjust b_t dynamically. Let $\mho$ ($\in [1, +\infty)$) be a predefined threshold used to differentiate the *warm-up stage*[2]. We use a "diminishing" policy for adjusting the threshold b_t. In other words, the longer the continuous online video watching time is, the smaller b_t will be set. This implies that, in the rest of segment fetching session, more video segments shall be fetched/downloaded. Specifically, our diminishing policy is as follows.

$$b_t = \begin{cases} b_t - \left\lfloor \frac{\frac{u_t}{\tau_{sd}} - (b_m - b_t)(b_m - b_t - 1)}{(b_m - b_t) \times 2} \right\rfloor, & if\ (b_m - b_t) < \mho \\ b_t - 1, & otherwise \end{cases} \tag{6}$$

where $\frac{u_t}{\tau_{sd}}$ is used to compute the total number of continuously viewed video segments; $(b_m - b_t) \times 2$ implies we use two segment fetching sessions as the evaluation unit in the warm-up stage; $(b_m - b_t)(b_m - b_t - 1)$ refers to the number of viewed video segments before the latest change on b_t, and it is obtained based on the following:

$$\begin{aligned} 2 \times \sum_{i=1}^{b_m - b_t - 1} i &= 2 \times \frac{(1 + (b_m - b_t - 1)) \times (b_m - b_t)}{2} \\ &= (b_m - b_t)(b_m - b_t - 1) \end{aligned} \tag{7}$$

In summary, Equation 6 reflects that, if $b_m - b_t < \mho$, it is in the warm-up stage. This implies that it could be hard to predict the user behavior. In this case, b_t is decreased after more segment fetching sessions are finished. In contrast, if $b_m - b_t \geq \mho$, the possibility to be in stable mode is increasing gradually. In this case, b_t is decreased after one segment fetching session is finished.

Note that, there is an issue needing to be emphasized. That is, when b_t is smaller and smaller, more and more video segments are fetched in a single segment fetching session. In this case, if users skip to another part of the video. This shall incur the significant waste, regardless of energy or bandwidth. To alleviate this issue, we introduce a parameter $\top$, which denotes the upper bound of the number of video segments in a single segment fetching session. In other words, when $b_m - b_t = \top$, we shall not further use the "diminishing" policy.

Table 2 Main notations used in our algorithm

Notation	Description
u_t	the user continuous watching time
b_s	the video buffer size
Θ	the threshold to trigger the adjustment of b_s
b'_m	the initial maximum number of video segments contained in the buffer b
$\mho$	the threshold to determine the warm-up stage
$\top$	the upper bound of the number of video segments in a single segment fetching session
Δ	the size of each adjustment
n_e	the total times of enlarging buffer
n_e^i	the total times of enlarging buffer for the ith playing part in the unstable mode
n_c	the number of buffered video segments

Another important element is the buffer size b_s. For the conventional online video player, it is usually set to a fixed value. Imagine if the number of video segments in a single segment fetching session is larger than $b_m - b_t$. In this case, the system could not work well. To alleviate this dilemma, in our design we adjust b_s self-adaptively. Let Θ ($\in (0, 1)$) be a threshold used to trigger the adjustment of b_s, and Δ^* be the size of each adjustment. Specifically, when $\frac{b_t}{b_m} \leq \Theta$, we set $b_s = b_s + \Delta^*$.

Note that, when we enlarge b_s, b_m (i.e., the maximum number of video segments contained in b) is naturally enlarged. In this case, the number of segments in the subsequent segment fetching session could sharply increase, incurring negative results such as the waste of bandwidth and energy (the reason is similar to our previous analysis). We remedy this trouble by enlarging the value of b_t. Specifically, only if b_s is set to $b_s + \Delta^*$, we immediately set $b_t = b_t + \Delta$, where $\Delta = \frac{\Delta^*}{\tau_{sd}}$. Notice that, τ_{sd} refers to the total service time of a segment (a.k.a., the playing back time duration of a video segment, recall Table 1).

Algorithm. The pseudo-codes of our method are shown in Algorithm 1. For ease of reference, the main notations are summarized in Table 2. In general, Line 1 serves as the initialization of parameters. Line 2 is the buffer prefetching phase; a lot of video segments are prefetched and then are filled into the buffer. Lines 3-17 are used to process the buffer feeding phase. More specifically, once a video segment has been played back, we remove it from the buffer, and accumulate the continuous watching time u_t (Lines 4-5). When n_c is smaller than the threshold b_t, we starts a new segment fetching session (Lines 6-12). Clearly, if the remaining video segments are null, nothing shall be fetched (Lines 7-8). In contrast, we need to fetch video segments through

[2] When one plays back an online video, the video may last for a relative long time. In this paper, the warm-up stage refers to the early stage of the video being played back. In the warm-up stage, the number of video segments in a single segment fetching session is relatively small.

a new segment fetching session. Note that, there are two cases: Lines 9-10 are used to process the case when the buffer size needs to be adjusted, while Lines 11-12 are used to process the case when the buffer size does not need to be adjusted. Finally, lines 13-17 are used to update b_t based on Equation 6. We remark that, if the user skips to another part of the video or starts a new video, the algorithm restarts the above process.

Algorithm 1: SA

```
1  initialize parameters ;
2  prefetch $b_m$ video segments, and set $n_c = b_m$ ;
3  while $n_c \neq 0$ do
4      if a video segment has been played back then
5          set $n_c = n_c - 1$, $u_t = u_t + \tau_{sd}$;
6      if $n_c <= b_t$ then
7          if the remaining video segments are null then
8              continue // go to Line 3 ;
9          if $\frac{b_t}{b_m} \leq \Theta$ // need to adjust buffer then
10             set $b_s = b_s + \Delta^*$, $b_t = b_t + \Delta$,
               $b_m = b_m + \Delta$, and fetch $\Delta + (b_m - b_t)$
               video segments;
11         else
12             fetch $b_m - b_t$ video segments;
13     if $b_m - b_t < \top$ then
14         if $b_m - b_t < \mho$ // warm-up stage then
15             update $b_t$ based on the 1st item in Eq. 6 ;
16         else
17             update $b_t$ based on the 2nd item in Eq. 6;
```

Theoretical analysis. In what follows, we examine the effectiveness of our proposed method, from theoretical perspective. As mentioned earlier, our method works for both stable and unstable modes. For *ease of exposition*, we first analyze the stable mode, and then examine the unstable mode.

$\triangleright$ *Stable mode.* Let b'_m be the initial maximum number[3] of video segments contained in the buffer b. Denote by N_m the number of downloaded video segments in the last segment fetching session. It can be computed as

$$\begin{aligned} N_m =& [n - b'_m - \Delta \times n_e - 2(\mho - 1) - (\top - \mho - 1)] \mod \top \\ =& [n - b'_m - \Delta \times n_e - \mho - \top + 3] \mod \top \end{aligned} \tag{8}$$

where "$2(\mho - 1)$" denotes the total downloaded video segments before $(b_m - b_t)$ reaches $\mho$, and "$(\top - \mho - 1)$" denotes the total downloaded video segments when $(b_m - b_t)$ is between $\mho$ and $\top$. Let N be the number of downloaded video segment after $(b_m - b_t)$ reaches $\top$. It can be computed as

$$N = \left\lfloor \frac{n - b'_m - \Delta \times n_e - \mho - \top + 3}{\top} \right\rfloor \tag{9}$$

Let $\overline{v}_f^k$ (resp., ζ_f^k) be the average download speed for (resp., the power consumption paid for) the fetching session that fetches k segments. Similar to Equation 2, one can have the following.

$$\begin{aligned} \zeta_f^k =& \frac{f_s \times k}{\overline{v}_f^k}(\zeta_{\tau_s} - \zeta_o) + \tau_{rt}(\zeta_{\tau_{rt}} - \zeta_o) \\ &+ (\tau_{sd} - \frac{f_s \times k}{\overline{v}_f^k} - \tau_{rt})(\zeta_{\tau_{ri}} - \zeta_o) \end{aligned} \tag{10}$$

Let n_e be the total times of enlarging buffer in the buffer feeding phase, and ζ_a be the additional energy paid for each buffer enlarging process. Based on the equations above, we can compute the "improved" energy consumption in the stable mode as follows.

$$\begin{aligned} \zeta =& \frac{b'_m}{\overline{v}_p} \times \zeta_{\tau_{bp}} + n_e \times \zeta_a + 2 \times \sum_{k=1}^{\mho-1} \zeta_f^k \\ &+ \sum_{k=\mho}^{\top-1} \zeta_f^k + N \times \zeta_f^{\top} + \zeta_f^{N_m} \end{aligned} \tag{11}$$

In the above equation, the part before the second "+" refers to the power consumption in the buffer prefetching phase and buffer enlarging process. Note that, both these phases have power-efficient traffic; in this case, the larger the segments to be downloaded, the more power-efficient [54,14,55]. It is not hard to realize that this part corresponds to the part before the first "+" in Equation 2. One can observe that the number of segments contained in this part (c.f., Equation 11) is $(b'_m + n_e \times \Delta)$. It is larger than[4] b_m in Equation 2. So, our method is more power-efficient in this part.

We next consider the part after the second "+" in Equation 11. This part refers to the power consumption in the buffer feeding phase, which could produce the RRC tails. Note that, the smaller the number of RRC tails, the less energy shall be consumed [14]. It is not hard to see that this part corresponds to the part after the first "+" in Equation 2. One can observe that the number of RRC tails in this part (c.f., Equation 11) is $(\mho + \top + N - 2)$. It is smaller than[5] $\frac{n-b_m}{b_m-b_t} = (n - b_m)$ in Equation 2. This is because the proposed method merges many single segment fetching sessions together. On the other hand, when many single segment fetching sessions are merged, the segments to be downloaded in a

[3] Here the term "initial" maximum number refers to the value before enlarging the buffer.

[4] Note that here b_m is essentially equal to b'_m.

[5] Remark that, in the traditional implementation, $b_m - b_t = 1$, and so $\frac{n-b_m}{b_m-b_t} = (n - b_m)$.

single session turns more, this shall achieve more power-efficient traffic (as we discussed earlier). Combing these reasons, our method is also more power-efficient in this part. To summarize, our proposed method achieves the improvement in the stable mode, viewed from the theoretical perspective.

$\triangleright$ *Unstable mode.* Recall Section 4, the total number of user skips is assumed to be $(i-1)$. Without loss of generality, assume that the buffer size b_s will be enlarged n_e^i times for the ith video playing part. Denote by N_m^i the number of downloaded video segments in the last fetching session, in terms of the ith video playing part. Then, We have

$$\begin{aligned} N_m^i =& (n_i - b'_m - \mathit{\Delta} \times n_e^i - 2(\mho - 1) - (\top - \mho - 1)) \mod \top \\ =& (n_i - b'_m - \mathit{\Delta} \times n_e^i - \mho - \top + 3) \mod \top \end{aligned} \tag{12}$$

Denote by N^i the number of downloaded video segment after $(b_m - b_t)$ reaches $\top$, in terms of the ith video playing part. We have

$$N^i = \left\lfloor \frac{n_i - b'_m - \mathit{\Delta} \times n_e^i - \mho - \top + 3}{\top} \right\rfloor \tag{13}$$

Based on the equations above, we can compute the "improved" energy consumption in the unstable mode as follows.

$$\begin{aligned} \zeta =& \sum_{k=1}^{i} (\frac{b'_m}{\overline{v}_p} \times \zeta_{\tau_{bp}} + n_e^i \times \zeta_a) + \sum_{k=1}^{i} (2 \times \sum_{k=1}^{\mho - 1} \zeta_f^k \\ &+ \sum_{k=\mho}^{\top - 1} \zeta_f^k + N^i \times \zeta_f^\top + \zeta_f^{N_m^i}) \end{aligned} \tag{14}$$

As similar as Equation 11, the part before (resp., after) the second "+" in Equation 14 corresponds to the part before (resp., after) the first "+" in Equation 4. Also, the former (resp., latter) part refers to the power consumption in the buffer prefetching phase and buffer enlarging process (resp., in the buffer feeding phase).

For the first part, in each single segment fetching session, the number of segments contained in the session is $(b'_m + n_e^i \times \mathit{\Delta})$, larger than b_m in Equation 4. Thus, our method is more power-efficient in this part (the reason is same to our analysis mentioned before). On the other hand, for the second part, in each single segment fetching session, the total number of RRC tails in the buffer feeding phase is $(\mho + \top + N^i - 2)$, which is smaller than the corresponding number $\frac{n_i - b_m}{b_m - b_t} = (n_i - b_m)$ in Equation 4. Thus, our method is also more power-efficient in the latter part (again, the reason is same to our previous analysis). To summarize, in the unstable mode the proposed method also achieves the improvement, viewed from the theoretical perspective.

6 Experiments

In this section, we first discuss the settings of our experiments, and then cover the detailed experimental results.

Experimental settings. To investigate the power consumption of mobile devices for online video streaming over 4G LTE networks, we deploy a systematic platform, which is composed of three major components: (i) Samsung Galaxy S5, one of the most popular smartphones; (ii) a professional *mobile power meter* — Monsoon power monitor [56]; and (iii) DASH-IF player, the official DASH industry forum reference and production video streaming player (http://dashif.org/software). Besides, some other auxiliary components including *wireshark* (https://www.wireshark.org), *QXDM* (https://www.qualcomm.com), are also used in our experiments. Here *Wireshark* is used for the traffic information collection, and *QXDM* is used for the RRC state information collection. In addition, the video server is built with *Jetty* (http://www.eclipse.org/jetty).

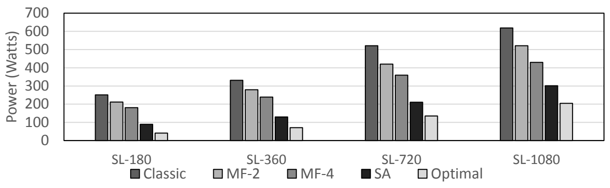
We cover the experimental results of four sample videos: (i) SL-180; (ii) SL-360; (iii) SL-720, and (iv) SL-1080, respectively. The length of each video is 600 seconds, and each video contains 300 video segments (i.e., chunks). Each video segment lasts for about 2 seconds. The resolutions of these videos are 180p, 360p, 720p, and 1080p, respectively. These videos are available at http://www.digitalprimates.net. Remark that we also tested other videos; those results are similar to our findings, omitted for saving space. The experimental parameters $\mathit{\Delta}$, $\mho$, $\top$, and $\mathit{\Theta}$ can be adjusted manually. We test these parameters offline, and empirically set to 5, 4, 10 and 50% respectively, in order to achieve the good performance. In addition, the default value of b_s is set to 15.

We test both the stable and unstable modes. When a user skips to watch another part of the video, we assume the "skip duration"[6] is uniformly distributed from 20 to 50 seconds. If a skip goes over the length of a video, we assume the user quits the video before ending. When there are user skips, all methods skip at the same time. In our tests, zero to three skips are used for our performance comparison. The "zero skips" means there is no user skips during the video playing (i.e., in stable mode).

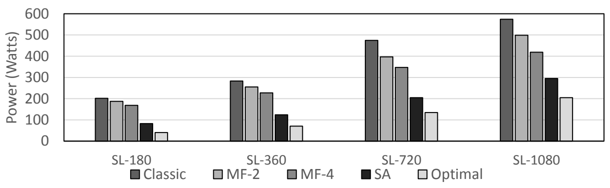
To evaluate the performance of our proposed method (known as SA, for short), we compare it against four competitors. We shortly introduce them as follows.

$\triangleright$ *Classic.* This is the most classic method for online video streaming [57]. When the buffer is not full,

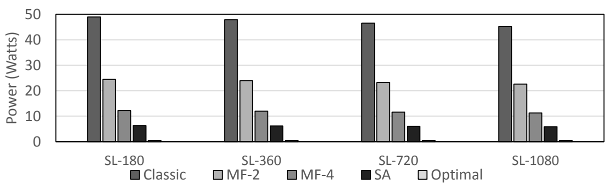
[6] Here the skip duration refers to the step length of each skip, when a user drags the process bar.



(a) Overall power consumption



(b) Transmission power consumption



(c) RRC tail power consumption

Fig. 4 Power consumption comparison. Note that, the "Optimal" method mainly serve as a reference.

another video segment fetching session starts immediately in the buffer feeding phase. That is, the classic method fetches the video segments one by one in a linear fashion in the buffer feeding phase.

▷ *MF-2*. This method was mentioned in [8]. Its basic idea is to merge two *segment fetching sessions* into one. This method can directly reduce the number of RRC tails. Note that, an obvious characteristic of this method is to merge the segment fetching sessions in a fixed manner.

▷ *MF-4*. This method is a variant of MF-2. Compared with MF-2, the major difference is that, it merges four segment fetching sessions into one.

▷ *Optimal*. This method downloads all video segments in a single segment fetching session, and then plays back the video *offline*. This method obtains the minimum power consumption. To some extent, this method can be also viewed as a variant of MF-2. Essentially, it is more extreme in the segment fetching session, compared with MF-4. Note that, this method is an "ideal" prototype, yet it is impractical in the real online video streaming application. Here it mainly serves as a reference.

We remark that the approach mentioned in [6] turns the wireless interface *off* (resp., *on*) when a segment fetching session *finishes* (resp., *starts*). This approach takes much additional time and energy to restore the network connection; particularly, it could harm the user experience, since other applications (e.g., Facebook, Twitter, WeChat) running on the mobile device could not work. Furthermore, the approach in [7] adjusts the buffer size according to users' log records of video watching time. In the context of our concern, we assume no such information is available. In view of these, we did not compare with these two approaches.

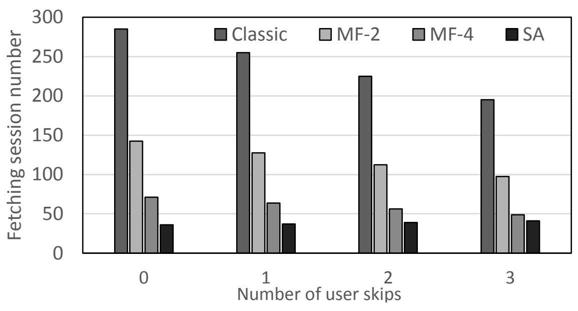


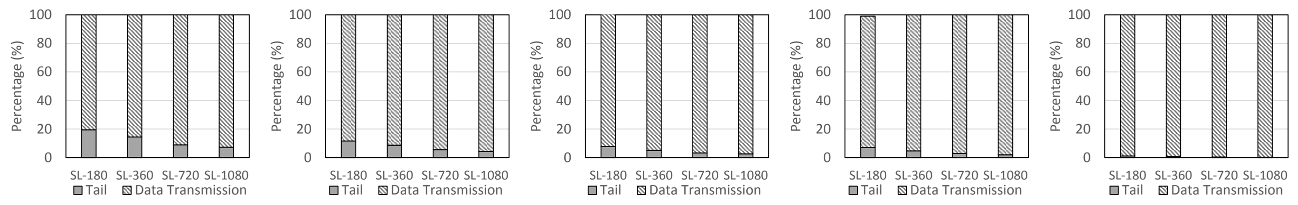
Fig. 5 The number of segment fetching sessions.

Experimental results. We first cover the experimental results without user skips, and then discuss the experimental results with user skips.

▷ *Without user skips*. Figure 4(a) shows the overall power consumption. We can see from this figure that, the performance of our proposed method (i.e., SA) is closest to that of "Optimal", and SA outperforms Classic by up to 62% for the power consumption of mobile devices. Also, we can see that both MF-2 and MF-4 can improve the power consumption of mobile devices, while they are still inferior than SA. This could be due to that they do not merge enough segment fetching sessions together. (See Figure 5, in which the number of segment fetching session are shown.) Note that, no matter how long the user watches the video, the buffer parameters are constant for these two methods. In contrast, SA collects the continuous video watching time, and adaptively adjust the buffer parameters (e.g., the threshold b_t, the buffer size b_s). As a result, the more video segment fetching sessions can be merged, saving the power consumption.

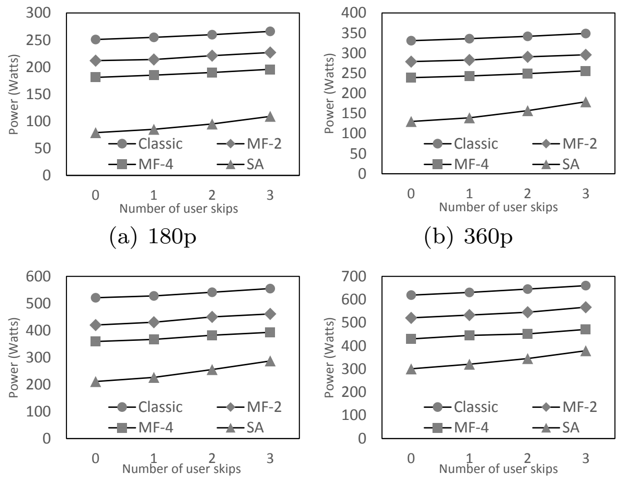
Figure 4(b) shows the power consumption for downloading the video segments (i.e., the transmission power consumption). We can see that SA consumes less power than classic, MF-2 and MF-4. This implies that SA can achieve more power-efficient traffics than other three approaches.

Figure 4(c) compares the power consumption of RRC tails. We can see that the power consumptions of RRC tails are almost the same for different *resolutions*. This is because, for different resolutions the numbers of segment fetching sessions are almost the same. Compared with Classic, MF-2 and MF-4 can significantly decrease the number of RRC tails, therefore they consume less energy. Note that SA outperforms both MF-2 and MF-



(a) Classic (b) MF-2 (c) MF-4 (d) SA (e) Optimal

Fig. 6 The power consumption percentage in the stable mode: the data transmission vs. the RRC tails.



(a) 180p (b) 360p

(c) 720p (d) 1080p

Fig. 7 Impact of user skips. Note that, it is essentially in the stable mode when the number of skips is 0.

4. This is because it can decrease more RRC tails, based on the user behaviors. As we expected, the power consumption of RRC tails is extremely tiny for the Optimal method. This is because it almost does not contain RRC tails.

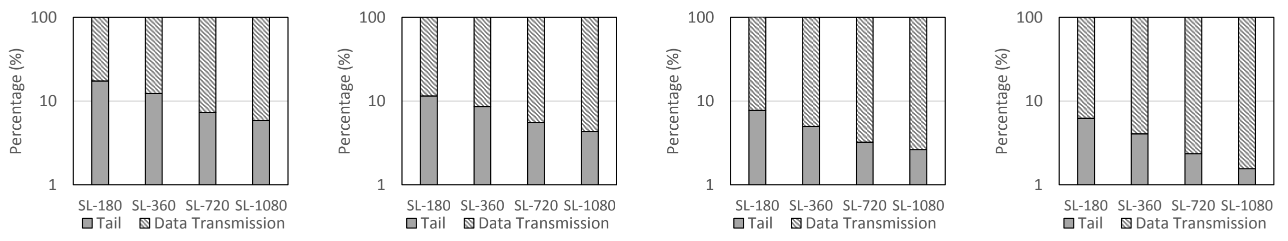
Figure 6 shows the power consumption percentage for data transmission and RRC tails. From Figure 6(e), we can see that almost no RRC tails exist in optimal. In addition, when the resolution turns smaller, the percentage of RRC tails is decreased gradually for all methods. Instead, the percentage of data transmission is increased gradually. And for all resolutions, data transmission occupies the larger percentage than RRC tails. In addition, we can also see that SA (cf., Figure 6(d)) keeps the power consumption of RRC tails in a lower percentage than Classic, MF-2 and MF-4 (cf., Figures 6(a), 6(b) and 6(c) respectively). This implies that SA makes the total power consumption more dedicated to video segment transmissions, and thus it is more efficient than the other three approaches.

▷ *With user skips.* We compare the performance of four methods here. Note that, in the unstable mode Optimal can not work; we here do not discuss it. Figure 7 shows the impact of user skips. It can been seen that the more the number of user skips is, the more power consumption will be paid. This is because, if a user skips to another part of the video, another additional buffer prefetching phase will be paid. Particularly, in this case the undesired buffered video segments will be removed from the buffer, without watching them; yet these video segments have already consumed (additional) power previously. In addition, we can see that, with the increase of user skips, the power consumption of SA is increased, and the increase speed is greater than Classic, MF-2 and MF-4. This is because, in the unstable mode SA can not stay in the most power-efficient stage for a long time. However, from the experimental results, we can still see that the overall performance of SA is still better than Classic, MF-2 and MF-4. This further demonstrates the effectiveness of our proposed approach. Again, we can see from Figure 5 that, SA can achieve a smaller value in terms of the number of fetching sessions, even if there are user skips. This could be the main reason why SA outperforms the other three methods.

Figure 8 shows the power consumption percentage for data transmission and RRC tails, when user skips happen. (Note that, for ease of observation, we use the logarithmic scale for the percentage). One can see from the figure that, SA also has the lower percentage in terms of RRC tail power consumption. This further demonstrates that, SA makes the total power consumption more dedicated to video segment transmissions. Therefore, SA is more power-efficient than other three approaches in the unstable mode, too.

7 Conclusion

In this paper, we analyzed the major challenges in reducing the power consumption for online video streaming over 4G LTE networks; and proposed a self-adaptive method that addresses these challenges efficiently. The central idea of our method is to exploit the continuous video watching time to predict users' behavior modes, and then dynamically adjust corresponding parameters, so as to achieve a relatively small power consumption.



(a) Classic (b) MF-2 (c) MF-4 (d) SA

Fig. 8 The power consumption percentage in the unstable mode: the data transmission vs. the RRC tails.

We provided a rigorous theoretical analysis for our proposed method. Also, we experimentally showed that the proposed method consistently outperforms the classical method as well as other competitors adapted from existing methods.

Acknowledgements We would like to acknowledge the editors and anonymous reviewers for their instructive suggestions. Also, we gratefully acknowledge the warm help of Prof. Chunyi Peng and Prof. Sheng Wei, who have offered us valuable advices. This work was sponsored by the National Basic Research ("973") Program of China (2015CB352403), the National Natural Science Foundation of China (61261160502 and 61272099), the Scientific Innovation Act of STCSM (13511504200), and the EU FP7 CLIMBER project (PIRSES-GA-2012-318939).

References

1. GSMA global mobile economy report 2015. http://gsmamobileeconomy.com/global/.
2. Ericsson mobility report: on the pulse of the networked society. http://www.ericsson.com/mobility-report, 2016.
3. Mohammad Reza Zakerinasab and Mea Wang. A cloud-assisted energy-efficient video streaming system for smartphones. In *IWQoS*, pages 1–10, 2013.
4. Wei Xiang, Gengkun Wang, Mark Pickering, and Yongbing Zhang. Big video data for light-field-based 3D telemedicine. *IEEE Network*, 30(3):30–38, 2016.
5. Xiaoyan Guo, Yu Cao, and Jun Tao. SVIS: Large scale video data ingestion into big data platform. In *DASFAA Workshops*, pages 300–306, 2015.
6. Wenjie Hu and Guohong Cao. Energy-aware video streaming on smartphones. In *INFOCOM*, pages 1185–1193, 2015.
7. Xin Li, Mian Dong, Zhan Ma, and Felix C. A. Fernandes. Greentube: power optimization for mobile videostreaming via dynamic cache management. In *ACM Multimedia Conference*, pages 279–288, 2012.
8. Sheng Wei, V. Swaminathan, and Mengbai Xiao. Power efficient mobile video streaming using HTTP/2 server push. In *MMSP*, pages 1–6, 2015.
9. Mike Williams. Why are mobile phone batteries still so crap. http://www.techradar.com/news/phone-and-communications/mobile-phones/why-are-mobile-phone-batteries-still-so-crap–1162779.
10. Hauke Holtkamp, Gunther Auer, Samer Bazzi, and Harald Haas. Minimizing base station power consumption. *IEEE Journal on Selected Areas in Communications*, 32(2):297–306, 2014.
11. Margot Deruyck, Emmeric Tanghe, David Plets, Luc Martens, and Wout Joseph. Optimizing LTE wireless access networks towards power consumption and electromagnetic exposure of human beings. *Computer Networks*, 94:29–40, 2015.
12. Junxian Huang, Feng Qian, Alexandre Gerber, Z Morley Mao, Subhabrata Sen, and Oliver Spatscheck. A close examination of performance and power characteristics of 4G LTE networks. In *MobiSys*, pages 225–238, 2012.
13. Reema Imran, Mutaz Shukair, Nizar Zorba, and Christos Verikoukis. An energy saving strategy for LTE-A multiantenna systems. *Mobile Networks and Applications*, 20(5):692–700, 2015.
14. Jingyu Zhang, Zhi-Jie Wang, Song Guo, Dingyu Yang, Gan Fang, Chunyi Peng, and Mingyi Guo. Power consumption analysis of video streaming in 4G LTE networks. *Wireless Networks*, doi: 10.1007/s11276-017-1519-9, 2017.
15. Hui Wang, H Eduardo Roman, Liyong Yuan, Yongfeng Huang, and Rongli Wang. Connectivity, coverage and power consumption in large-scale wireless sensor networks. *Computer Networks*, 75:212–225, 2014.
16. Nanhao Zhu and Athanasios V Vasilakos. A generic framework for energy evaluation on wireless sensor networks. *Wireless Networks*, 22(4):1199–1220, 2016.
17. Peng Li, Song Guo, and Jiankun Hu. Energy-efficient cooperative communications for multimedia applications in multi-channel wireless networks. *IEEE Transactions on Computers*, 64(6):1670–1679, 2015.
18. Lin Xiang, Xiaohu Ge, Cheng-Xiang Wang, Frank Y Li, and Frank Reichert. Energy efficiency evaluation of cellular networks based on spatial distributions of traffic load and power consumption. *IEEE Trans. on Wireless Communications*, 12(3):961–973, 2013.
19. Andrea Lupia and Floriano De Rango. Evaluation of the energy consumption introduced by a trust management scheme on mobile ad-hoc networks. *Journal of Networks*, 10(4):240–251, 2015.
20. Metin Tekkalmaz and Ibrahim Korpeoglu. Distributed power-source-aware routing in wireless sensor networks. *Wireless Networks*, 22(4):1381–1399, 2016.
21. Krishnan Narendran, R. M. Karthik, and Krishna M. Sivalingam. Iterative power control based admission control for wireless networks. *Wireless Networks*, 22(2):619–633, 2016.
22. Xianfu Chen, Jinsong Wu, Yueming Cai, Honggang Zhang, and Tao Chen. Energy-efficiency oriented traffic offloading in wireless networks: a brief survey and a learning approach for heterogeneous cellular networks. *IEEE Journal on Selected Areas in Communications*, 33(4):627–640, 2015.
23. Nikolaos A. Pantazis and Dimitrios D. Vergados. A survey on power control issues in wireless sensor

networks. *IEEE Communications Surveys and Tutorials (COMSUR)*, 9(1-4):86–107, 2007.

24. Vijeth J. Kotagi, Rahul Thakur, Sudeepta Mishra, and Chebiyyam Siva Ram Murthy. Breathe to save energy: Assigning downlink transmit power and resource blocks to LTE enabled IoT networks. *IEEE Communications Letters*, 20(8):1607–1610, 2016.
25. Naveen Mysore Balasubramanya, Lutz Lampe, Gustav Vos, and Steve Bennett. DRX with quick sleeping: A novel mechanism for energy-efficient IoT Using LTE/LTE-A. *IEEE Internet of Things Journal*, 3(3):398–407, 2016.
26. Naveen Mysore Balasubramanya, Lutz Lampe, Gustav Vos, and Steve Bennett. Low SNR uplink CFO estimation for energy efficient IoT using LTE. *IEEE Access*, 4:3936–3950, 2016.
27. Juan Luo, Di Wu, Chen Pan, and Junli Zha. Optimal energy strategy for node selection and data relay in WSN-based IoT. *Mobile Networks and Applications*, 20(2):169–180, 2015.
28. Antonino Orsino, Giuseppe Araniti, Leonardo Militano, Jesus Alonso-Zarate, Antonella Molinaro, and Antonio Iera. Energy efficient IoT data collection in smart cities exploiting D2D communications. *Sensors*, 16(6), 2016.
29. Kun Wang, Yihui Wang, Yanfei Sun, Song Guo, and Jinsong Wu. Green industrial Internet of Things architecture: An energy-efficient perspective. *IEEE Communications Magazine*, 54(12-Supp):48–54, 2016.
30. Luigi Atzori, Antonio Iera, and Giacomo Morabito. The Internet of Things: A survey. *Computer Networks*, 54(15):2787–2805, 2010.
31. Yunmin Go, Oh Chan Kwon, and Hwangjun Song. An energy-efficient HTTP Adaptive Video Streaming with networking cost constraint over heterogeneous wireless networks. *IEEE Trans. on Multimedia*, 17(9):1646–1657, 2015.
32. Anna Ukhanova, Evgeny Belyaev, Le Wang, and Søren Forchhammer. Power consumption analysis of constant bit rate video transmission over 3G networks. *Computer Communications*, 35(14):1695–1706, 2012.
33. Mohammad Ashraful Hoque, Matti Siekkinen, and Jukka K Nurminen. Using crowd-sourced viewing statistics to save energy in wireless video streaming. In *MOBICOM*, pages 377–388, 2013.
34. Yousef O Sharrab and Nabil J Sarhan. Aggregate power consumption modeling of live video streaming systems. In *MMSys*, pages 60–71, 2013.
35. Ya Ju Yu, Pi Cheng Hsiu, and Ai Chun Pang. Energy-efficient video multicast in 4G wireless systems. *IEEE Transactions on Mobile Computing*, 11(10):1508–1522, 2012.
36. Jang Ping Sheu, Chien Chi Kao, Shun Ren Yang, and Lee Fan Chang. A resource allocation scheme for scalable video multicast in WiMAX relay networks. *IEEE Transactions on Mobile Computing*, 12(1):90–104, 2013.
37. Kun Wang, Jun Mi, Chenhan Xu, Qingquan Zhu, Lei Shu, and Der Jiunn Deng. Real-time load reduction in multimedia big data for mobile Internet. *ACM Trans. on Multimedia Computing Communications & Applications*, 12(5s):76:1–76:20, 2016.
38. Shuo Deng and Hari Balakrishnan. Traffic-aware techniques to reduce 3G/LTE wireless energy consumption. In *CoNEXT*, pages 181–192, 2012.
39. Li-Ping Tung, Ying-Dar Lin, Yu-Hsien Kuo, Yuan-Cheng Lai, and Krishna M Sivalingam. Reducing power consumption in LTE data scheduling with the constraints of channel condition and QoS. *Computer Networks*, 75:149–159, 2014.
40. Kun Wang, Yihui Wang, Deze Zeng, and Song Guo. An sdn-based architecture for next-generation wireless networks. *IEEE Wireless Communications*, 24(1):25–31, 2017.
41. Kun Wang, Heng Lu, Lei Shu, and Joel J. P. C. Rodrigues. A context-aware system architecture for leak point detection in the large-scale petrochemical industry. *IEEE Communications Magazine*, 52(6):62–69, 2014.
42. Siripuram Aditya and Sachin Katti. Flexcast: graceful wireless video streaming. In *MOBICOM*, pages 277–288, 2011.
43. Kun Wang and Yue Yu. A query-matching mechanism over out-of-order event stream in iot. *International Journal of Ad Hoc & Ubiquitous Computing*, 13(3/4):197–208, 2013.
44. Xiaoqi Yin, Abhishek Jindal, Vyas Sekar, and Bruno Sinopoli. A control-theoretic approach for dynamic adaptive video streaming over HTTP. In *SIGCOMM*, pages 325–338, 2015.
45. Kun Wang, Yun Shao, Lei Shu, and Guangjie Han. Ldpa: a local data processing architecture in ambient assisted living communications. *IEEE Communications Magazine*, 53(1):56–63, 2015.
46. Kun Wang, Linchao Zhuo, Yun Shao, Dong Yue, and Kim Fung Tsang. Toward distributed data processing on intelligent leak-points prediction in petrochemical industries. *IEEE Trans. on Industrial Informatics*, 12(6):2091–2102, 2016.
47. Matthew K Mukerjee, David Naylor, Junchen Jiang, Dongsu Han, Srinivasan Seshan, and Hui Zhang. Practical, real-time centralized control for CDN-based live video delivery. In *SIGCOMM*, pages 311–324, 2015.
48. Yun Shao, Kun Wang, Lei Shu, Song Deng, and Der Jiunn Deng. Heuristic optimization for reliable data congestion analytics in crowdsourced ehealth networks. *IEEE Access*, 4:9174–9183, 2016.
49. Kun Wang, Yun Shao, Lei Shu, and Chunsheng Zhu. Mobile big data fault-tolerant processing for ehealth networks. *IEEE Network*, 30(1):36–42, 2016.
50. Florin Dobrian, Vyas Sekar, Asad Awan, Ion Stoica, Dilip Joseph, Aditya Ganjam, Jibin Zhan, and Hui Zhang. Understanding the impact of video quality on user engagement. *ACM SIGCOMM Computer Communication Review*, 41(4):362–373, 2011.
51. Athula Balachandran, Vyas Sekar, Aditya Akella, Srinivasan Seshan, Ion Stoica, and Hui Zhang. Developing a predictive model of quality of experience for internet video. *ACM SIGCOMM Computer Communication Review*, 43(4):339–350, 2013.
52. 3GPP TR 25.813: Radio interface protocol aspects (V7.1.0), 2006.
53. Thomas Stockhammer. Dynamic adaptive streaming over HTTP –: standards and design principles. In *MMSys*, pages 133–144, 2011.
54. Yan Zhang, N. Ansari, Mingquan Wu, and H. Yu. Afstart: An adaptive fast TCP slow start for wide area networks. In *ICC*, pages 1260–1264, 2012.
55. In Kwan Yu and Richard Newman. TCP slow start with fair share of bandwidth. *Computer Networks*, 55(17):3932–3946, 2011.
56. Monsoon power monitor. http://www.msoon.com/LabEquipment/PowerMonitor/.
57. Alex Zambelli. IIS smooth streaming technical overview. *Microsoft Corporation*, 3:40, 2009.