Toward Cost-Effective Mobile Video Streaming via Smart Cache With Adaptive Thresholding

Di Wu, *Member, IEEE*, Jian Huang, Jian He, Min Chen, *Senior Member, IEEE*, and Guoqing Zhang

***Abstract*—Mobile video streaming has become the leading contributor to the explosive growth of mobile Internet traffic. As a key system parameter of mobile video apps, cache threshold plays an important role in regulating the downloading behavior of a mobile device, and directly affects the efficacy of mobile video streaming. In this paper, we first conduct a series of well-designed experiments to understand the mechanism of cache management in the Android OS and investigate the impact of cache threshold on the performance of mobile video streaming. The experiments show that the current static configuration of cache thresholds in the Android OS cannot well balance the tradeoff between cost incurred by unconsumed content and user quality of experience. To achieve cost-effective mobile video streaming, this paper further proposes a control-theoretic cache management algorithm called smart cache with adaptive thresholding (SCAT), which can intelligently tune cache thresholds to satisfy user preferences. The complexity of cache management and optimization can be decreased extensively by the SCAT strategy, facilitating its easy integration with the OS kernel codes. Finally, we implement and evaluate our proposed scheme on the real Android platform and the experimental results verify that our proposed scheme achieves significant gain over other alternative approaches. Compared to the Android's default scheme, SCAT achieves over 40% reduction of unconsumed content cost and nearly 30% reduction of freezing duration in the low-bandwidth network environment.**

***Index Terms*—Mobile video streaming, multimedia broadcasting, cache management, cache threshold, control theory.**

I. Introduction

In RECENT years, the prevalence of mobile devices leads to the rapid growth of the mobile Internet traffic. Cisco reported in [1] that the global mobile data traffic has increased to 1.5 Exabytes per month by the end of 2013, nearly 81 percents higher than that in 2012. With wide deployments of 3G/4G/WiFi networks, more and more people prefer to watch videos with mobile devices, such as smartphones, tablets, pads, etc [2]. The report also shows that 53% of the mobile traffic is generated by video, and it is expected that over two-thirds of the world's mobile data traffic will be video in 2018.

In reality, the quality of mobile video streaming is highly impacted by user mobility and the fluctuation of wireless channels. To mitigate the effect of stochastic channel fluctuation, a commonly adopted approach is to cache a certain amount of video data before playback. By data prefetching, it is feasible to reduce the possibility of playback freezing. For cache management, two cache thresholds are widely used to regulate the prefetching behavior of a client, namely, *high threshold* and *low threshold*. Normally, the high threshold is set as the maximum size of memory space allocated for video cache, and the low threshold is set as the minimum amount of fresh content in the cache to avoid playback freezing.

In spite that video caching can help improve user experience, it is at the cost of downloading extra data bytes before the actual video playback. As pointed by the measurement work in [3] and [4], mobile users tend to abort more frequently than PC users during the viewing process. When a user aborts viewing, the extra downloaded bytes will be wasted with no gain. For mobile users with the 3G/4G connectivity, who are charged by their data usage, such cost of unconsumed content is directly translated into monetary cost for mobile users. Even for mobile users with a flat pricing plan, the unconsumed content would cause additional energy consumption for mobile devices.

Previous work (e.g., [5], [6]) mostly focused on improving the quality of user experience, but paid less attention on the problem of unconsumed content. In [7], He *et al.* proposed a novel buffer management strategy called CBM (cost-aware buffer management) to address the problem of unconsumed video content. CBM uses a fixed buffer size and intelligently calculates the number of video chunks to be downloaded at the beginning of each slot to achieve a good balance between sunk cost and user experience. However, the CBM strategy requires the modification of the internal buffer manager itself, which makes it not easy to be integrated into current mobile platforms seamlessly. Different from above, we focus our attention on cache thresholds, which are the key parameters of a cache management algorithm. We aim to design a smart cache management algorithm by tuning cache thresholds and take both viewing quality and unconsumed

Manuscript received March 25, 2015; revised June 18, 2015; accepted July 23, 2015. Date of publication September 17, 2015; date of current version December 5, 2015. This work was supported in part by the National Science Foundation of China under Grant 61272397, Grant 61174152, and Grant 61331008, in part by the Guangdong Natural Science Funds for Distinguished Young Scholar under Grant S20120011187, in part by the Basic Research Program 973 of China under Grant 2011CB302505, and in part by the Innovation Team from Guangdong Scientific Bureau, China, under Grant 201001D0104726115. *(Corresponding author: Di Wu.)*

D. Wu and J. Huang are with the Department of Computer Science, Sun Yat-sen University, Guangzhou 510006, China, and also with the SYSU-CMU Shunde International Joint Research Institute, Foshan 528300, China (e-mail: wudi27@mail.sysu.edu.cn; huangj77@mail2.sysu.edu.cn).

J. He is with the Department of Computer Science, University of Texas at Austin, Austin, TX 78712-1757 USA (e-mail: jianhe@cs.utexas.edu).

M. Chen is with the School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China (e-mail: minchen@ieee.org).

G. Zhang is with the Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China (e-mail: gqzhang@ict.ac.cn).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TBC.2015.2465173

content cost into account. Moreover, our design will not use any proxy or predicted information to avoid implementation complexity.

In this paper, we first build a dedicated testbed to measure various user-related metrics of mobile video streaming under different configurations of cache thresholds. We aim to better understand the mechanism of cache management in the real Android platform and study the impacts of cache thresholds on mobile video streaming. Our measurement results point out the drawback of the current static configuration of cache thresholds, and show that there exists a clear cost-QoE (Quality of Experience) tradeoff when varying the cache thresholds. To achieve cost-effective mobile video streaming, we further exploit control theory to design a simple yet efficient cache management algorithm called *SCAT (Smart Cache with Adaptive Thresholding)*, which can reduce both unconsumed content cost and freezing duration by simply tuning cache thresholds. Finally, we validate the effectiveness of our proposed algorithm via extensive simulations. In summary, our main contributions in this paper can be listed as follows:

- We conduct a detailed measurement study to measure the unconsumed content and freezing duration of mobile video streaming when varying cache thresholds in the Android platform. Our measurements show that, the amount of unconsumed content incurred by user viewing abortion will grow with the increase of either high or low threshold. The freezing duration can be reduced significantly by increasing the low threshold. It is also interesting to observe that there exists a clear tradeoff between unconsumed content cost and freezing duration when varying cache thresholds. It implies that the current static scheme has a large room for improvement.
- We propose an online cache management algorithm called *SCAT* to adjust cache thresholds dynamically according to user preferences. Our algorithm is based on control theory and can guide the cache towards the target state by simply tuning high and low thresholds. Our proposed algorithm can well balance the tradeoff between unconsumed content cost and freezing duration. The implementation of our proposed algorithm incurs little modification to the Android media framework and can be easily integrated with the OS kernel codes.
- We implement our algorithm on the Android OS and compare with other alternative schemes by real measurements. Our experimental results show that, compared with the Android's default scheme, our proposed SCAT algorithm can reduce over 40% unconsumed content cost and about 30% freezing duration in the low-bandwidth environment. Our algorithm also outperforms another two alternatives.

The remainder of this paper is organized as follows: Section II surveys related works in the field. In Section III, we evaluate the impacts of cache thresholds on mobile video streaming. In Section IV, we propose a control-theoretic cache management algorithm to tune cache thresholds intelligently, which can achieve the balance between unconsumed content cost and freezing duration. In Section V, we describe the implementation and evaluation of our proposed algorithm and compared with other schemes. Section VI concludes the whole paper and discusses our future work.

II. Related Work

Due to the rapid adoption of mobile devices, mobile video streaming has attracted a lot of attention. One thread of research focused on the measurements of mobile video streaming applications. Erman *et al.* [8] measured the traffic characteristics of video streaming in the cellular networks. In [9]–[13], the authors investigated the component breakdown of energy consumption of mobile devices during data transfer and video streaming. Finamore *et al.* [3] and Li *et al.* [14] analyzed viewing behaviors of mobile users and observed frequent abortion of mobile users during video viewing. Liu *et al.* [15] measured several key properties of mobile video streaming systems, including the distribution of mobile devices, video length distribution, etc. Staelens *et al.* [16] focused on assessing the user-perceived quality of viewing long-duration video sequences with tablet devices.

Another thread of research is on the design of new streaming protocols and algorithms for better user experience [17] and resource utilization [18]. Liu *et al.* [19] compared the architecture and performance of four typical mobile video streaming protocols, including RTSP streaming, Pseudo streaming, Chunk-based streaming and P2P streaming. In [20], BlueStreaming was proposed to achieve energy-efficient streaming by increasing the probability that the WNIC (Wireless Network Interface Controller) can be switched to the power-saving mode. In [21], van der Schaar and Shankar studied the effectiveness of cross-layer optimization to meet the QoS requirements of wireless multimedia streaming. Mastronarde and van der Schaar [22] proposed to use online reinforcement learning for multimedia buffer control. In [23], Ding *et al.* studied the use of proxying to achieve a more efficient PSM (Power-Saving Mode). Bhatia *et al.* [24] proposed a scheme to opportunistically exploit slow time-varying channels to increase the capacity of video streaming in mobile networks.

Cache management is an important component to realize smooth playback of mobile video streaming. Liu *et al.* [5], [25] compared cache management mechanisms on different mobile platforms, and found that iOS devices tend to download aggressively to improve user experience. Li *et al.* [6] designed and implemented a cache management algorithm called *GreenTube*, which can optimize power consumption of a mobile device by adjusting the high threshold dynamically. The *GreenTube* algorithm aims to minimize the download time for mobile devices to save energy consumption. Jia *et al.* [26] proposed a cooperative content fetching strategy to increase the quality of video delivery to mobile users. In [27], a frame selection scheme for video transcoding is proposed to enhance mobile video communication via frame complexity analysis. Ukommi *et al.* [28] designed a media bitrate adaptation scheme for mobile video services based on the content characteristics. Xu *et al.* [29] introduced an ant-inspired mini-community-based video sharing solution for

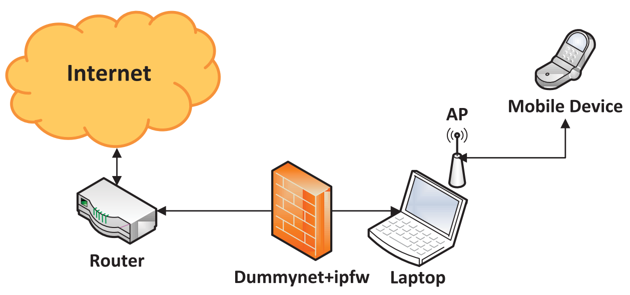


Fig. 1. Components of our measurement testbed.

improving the VoD (Video on Demand) services in wireless mobile networks.

All our previous and current research work focus on having a better understanding of mobile video streaming and thus being able to provide a more cost-effective mobile video streaming service. To this purpose, we have conducted a series of measurement studies to understand the internal mechanism and characteristics of mobile video streaming. In [30], we investigated power consumption of mobile video streaming apps under various network conditions. We measured power consumption under different bandwidth settings, packet loss rates, and propagation delays. In [31], we conducted a measurement study to measure the cost and quality of mobile video streaming when varying cache thresholds, and reported some preliminary measurement results. This paper extends our previous work in [30] and [31] by further proposing a simple yet practical approach called SCAT, which adjusts cache thresholds intelligently to achieve cost-effective mobile video streaming. Different from the work in [7], SCAT does not need to make any changes to the underlying buffer management FSM. In addition, we implement SCAT on the real Android platform and evaluate its performance in a real testbed, which has not been done in [7].

III. Understanding the Impacts of Cache Thresholds on Mobile Video Streaming

Cache thresholds are of great significance to the performance of video streaming. In this section, we conduct a series of measurement studies to understand the mechanism of cache management in the Android OS and examine the impacts of different cache threshold settings on mobile video streaming.

A. Measurement Methodology

To conduct our experiments, we build a real testbed as shown in Fig. 1, which consists of a mobile device, a laptop with the WiFi interface, and a router with the Internet connection.

The mobile device used in our experiment is an HTC G8 smartphone installed with the Android OS. The usage of the Android OS in our experiments is due to its popularity and open-source nature. The Android smartphone was installed with a number of mobile video streaming apps for testing. As YouKu [32] is the largest video streaming service provider in China, we adopt the YouKu app as the default video streaming application in the experiments and the default video streaming rate is set as 300 Kbps. In our measurements, to understand their impacts on mobile video streaming, we adjust the setting of high and low thresholds and evaluate the corresponding quality measures, such as freezing duration, the amount of unconsumed content, and so on.

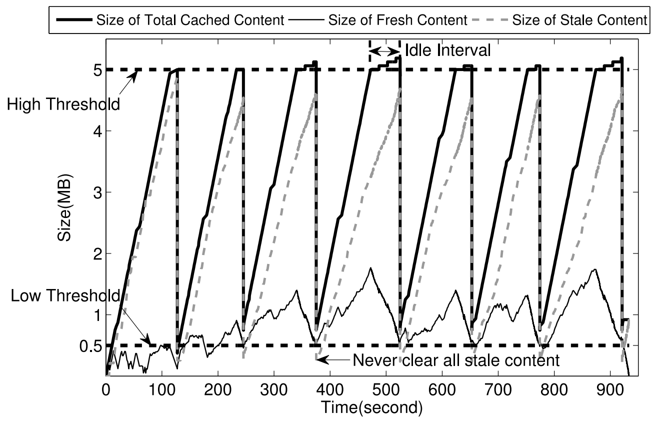


Fig. 2. Evolution of Android cache state (available bandwidth: 400 Kbps).

By enabling the AP (Access Point) mode of the WiFi interface, we configure the laptop as a wireless router, which is further connected to the Internet via our campus network. The smartphone is connected with the laptop-based wireless router for the Internet access. To emulate the properties of wide-area networks (e.g., packet loss, propagation delay, available bandwidth), we install DummyNet [33] and ipfw [34] on the laptop to control data transmission to the smartphone. In addition, we also install Wireshark [35] on the laptop to capture packet-level traces on the wireless interface that connects the smartphone with the laptop.

In the Android media framework, the underlying Android media player is called when playing a local or remote video file. For YouKu, its logs contain all the information of video playback, including starting time, pause time, freezing duration, etc. In the default settings, video logs generated by the media framework are debug-level system files, which cannot be directly accessed by upper-level applications. To access the log files, we raise the priority level of YouKu's log files explicitly. We write a simple monitor program to automate the task of log collection on the Android platform. To keep track of cache status, we insert extra codes into the Android media framework. The generated cache logs can be retrieved using the Android SDK tools.

B. Evolution of the Android Cache State

In the following section, we use our testbed to investigate the evolution of cache state, and study the impacts of different configurations of cache thresholds.

We first investigate the dynamics of cached contents for mobile video streaming. We define *fresh content* as the amount of video data downloaded but not viewed in the cache, and *stale content* as the amount of video content in the cache that has been viewed. Fig. 2 shows the evolution of cache state when the available bandwidth is set as 400Kbps. The high and low thresholds are set as 5MB and 512KB respectively.

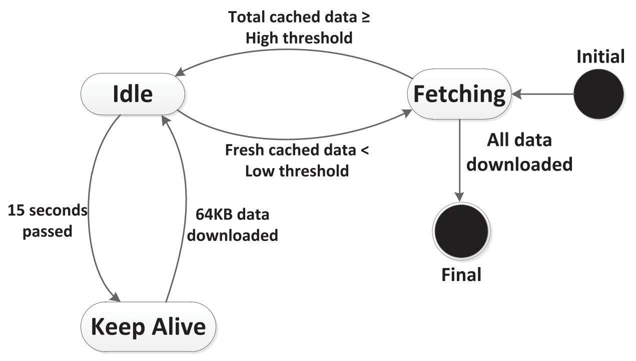


Fig. 3. Finite state machine of the Android HTTP data source provider.

We find that, the client stops fetching more content when the total amount of cached content reaches the high threshold. Once the amount of fresh content drops below the low threshold, the fetching process is started again. In the meanwhile, most of the stale content is cleaned out the cache[1].

To better understand the evolution of Android cache state, we examine the source code of the Android media framework. We find that, when the Android media framework is called to handle a video streaming request, its data source provider [36] follows the *FSM (Finite State Machine)* shown in Fig. 3.

The FSM contains five states, namely, *Initial, Fetching*, *Idle*, *Keep-Alive* and *Final*. Initially, when the video playback is started by the viewer, the data source provider enters into the *Fetching* state and downloads video chunks continuously from the HTTP server. When the total amount of cached data reaches a high threshold, it moves to the *Idle* state. No data transmission occurs in the *Idle* state. However, a keep-alive mechanism will be activated in the idle state. The data source provider enters into the *Keep-alive* state every 15 seconds and downloads 64KB video data to keep the TCP connection alive. After that, it will return back to the *Idle* state. Once the amount of the fresh cached data falls below a low threshold, the state machine moves to the *Fetching* state and starts another round of bulk data transmission.

C. *Cost Incurred by Unconsumed Content*

The viewing pattern of mobile users is quite different from that of PC users [3], and mobile users tend to abort more frequently than PC users during viewing videos, which will result in the sunk cost of unconsumed video data (i.e., fresh content) in the cache.

We define *wasted content* as the amount of unconsumed content in the cache when a mobile user aborts video viewing. To analyze the amount of wasted content under different configurations of cache thresholds, we write a shell script to simulate the abortion behavior of viewers and measure the average amount of wasted content due to user abortion.

[1]By analyzing the Android source code, we observe that a small portion of the most recent stale content will be kept in the cache all the time. Thus, the amount of stale content will never become zero, except in the initial phase of video playback.

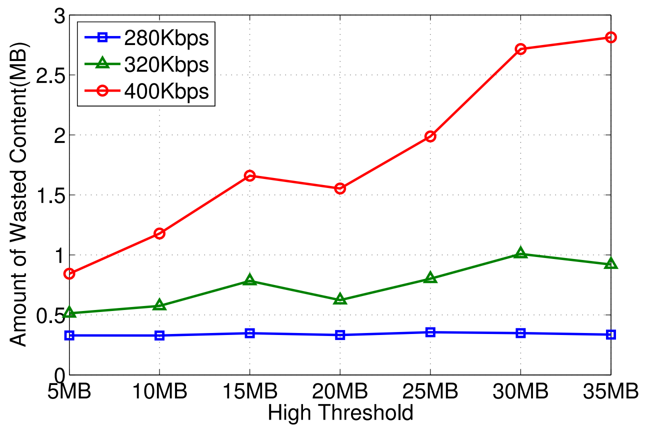


Fig. 4. Average amount of wasted content when varying high threshold.

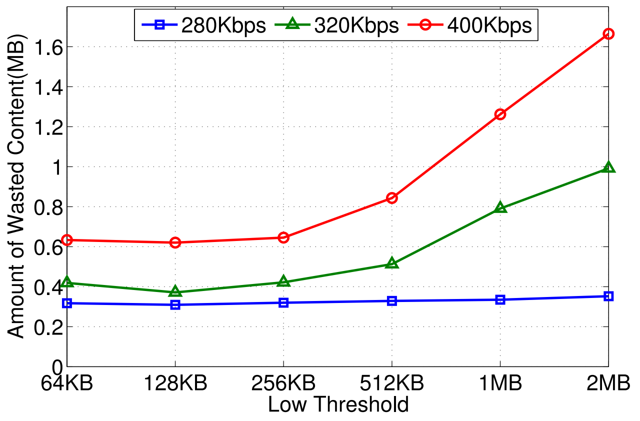


Fig. 5. Average amount of wasted content when varying low threshold.

Fig. 4 illustrates how the average amount of wasted content changes when varying high threshold under different bandwidth settings. When the bandwidth is higher than the video playback rate, the increase of high threshold will make the problem of unconsumed content be more serious. Even under the same high threshold, there is significant difference in terms of the amount of wasted content under different bandwidth settings. For instance, the amount of wasted content under 400Kbps bandwidth is almost 2.8 times of that under 320 Kbps bandwidth when the high threshold is 35MB. However, if the available bandwidth is smaller than the video playback rate (e.g., 280Kbps bandwidth), there is not much difference for different high thresholds. The main reason is that, when the bandwidth is insufficient to support the video playback rate, the amount of fresh content will be smaller than the low threshold most of the time. In this case, the change of high threshold brings little impact on the amount of wasted content.

We also study the impact of low threshold on the cost of fresh content. Fig. 5 shows similar phenomena as that in Fig. 4. When the bandwidth is insufficient to support the playback rate, the amount of unconsumed content is insensitive to the setting of low threshold. We can clearly observe that there is almost no change for the curve under the 280Kbps bandwidth. When the bandwidth is higher than the video playback rate (e.g., 400 Kbps), the increase of low threshold directly translates into a higher amount of wasted content.

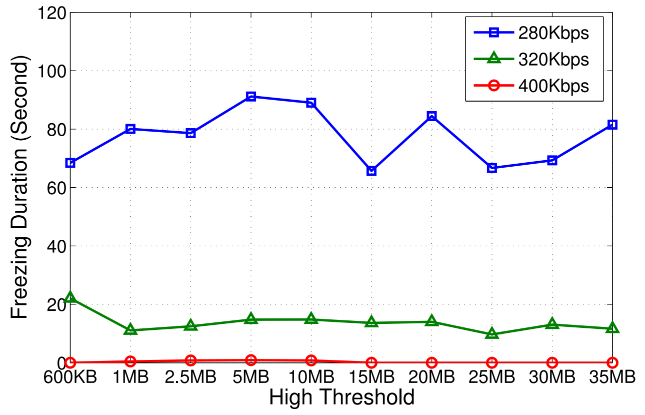


Fig. 6. Freezing duration when varying high threshold.

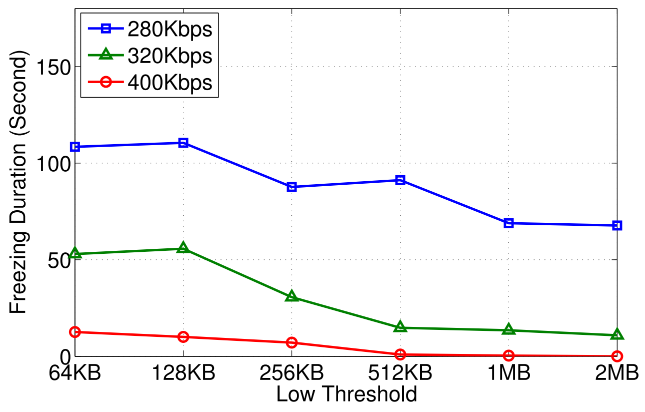
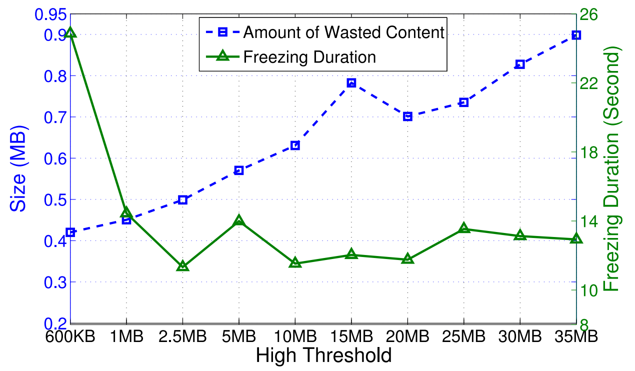
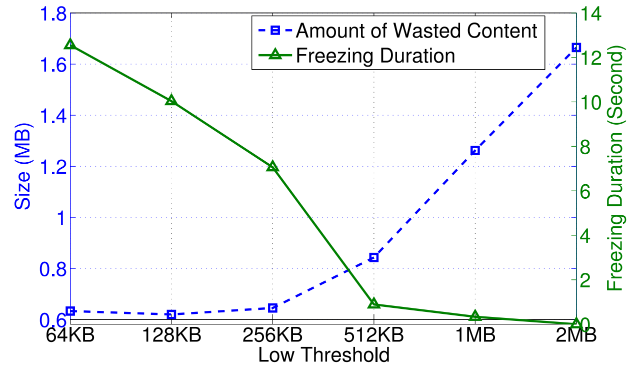


Fig. 7. Freezing duration when varying low threshold.



(a) Varying high threshold (network bandwidth: 320Kbps)



(b) Varying low threshold (network bandwidth: 400Kbps)

Fig. 8. Cost-QoE tradeoff when varying high and low thresholds.

D. Impacts on User Quality-of-Experience

In this section, we study how the configuration of cache thresholds impacts the user Quality-of-Experience (QoE). In terms of user QoE, we focus more on freezing duration. In the viewing process, when fresh content in the cache is exhausted, the video player has to enter into a freezing state.

We first vary the value of high threshold and plot the impacts on freezing duration in Fig. 6. In the figure, the value of freezing duration is the accumulated freezing duration in the whole video viewing process, instead of the duration of a single freezing period. In the viewing process, the user may encounter multiple short freezing periods. It is found that, the impact of varying high threshold is different under different bandwidth settings. When the bandwidth is set as 320Kbps, the increase of high threshold can help to reduce freezing duration in a certain degree. For the high-bandwidth environment (i.e., 400Kbps), we cannot gain more reduction of freezing duration by increasing the high threshold. When the bandwidth is insufficient (i.e., 280Kbps), the impact brought by changing high threshold does not show a clear trend.

Next, we plot the impacts of low threshold on freezing duration in Fig. 7. From the figure, we can observe that the increase of low threshold has significant impacts on the reduction of freezing duration. Under different bandwidth settings, the impacts brought by varying low thresholds show a similar trend. The reason lies in that a higher value of low threshold can trigger the mobile app client to enter into the *Fetching* state earlier and thus reduce the possibility that the buffer will run out of fresh content for video playback. However, when the low threshold is higher than 1 MB, the further increase of low threshold brings little impact on the reduction of freezing duration.

E. Tradeoff Between Wasted Content and User Experience

From the results in the previous sections, we can observe that the configuration of cache thresholds has significant impacts on the unconsumed content cost and user QoE. Therefore, we need to carefully configure cache thresholds to minimize the unconsumed content cost and improve the user QoE in the meanwhile.

Fig. 8 illustrates the evolution of unconsumed content and freezing duration when varying high and low thresholds under different bandwidth settings. From the figure, we can observe the existence of a clear cost-QoE tradeoff in some bandwidth settings. In Fig. 8(a), in spite that the increase of high threshold can decrease the freezing duration to a certain extent, but the gain has a diminishing effect. On the contrary, the amount of unconsumed content increases almost linearly in the meanwhile. Therefore, it is not a good idea to set a very large high threshold. We also find that the tradeoff under low bandwidth is not as pronounced as that under high bandwidth.

For the configuration of low threshold, we observe that there is a similar tradeoff between unconsumed content cost and freezing duration in Fig. 8(b). With the increase of low threshold, the freezing duration can be decreased at the same time. However, the effect becomes less significant when the low threshold is higher than 1 MB. We also observe that the increase of low threshold incurs more wastage.

However, the current cache management scheme used in the Android platform is static, and the settings on high and low thresholds are pretty ad-hoc. The designer also does not take the above tradeoff into account. The main contribution of our work is to empirically study the impacts of cache thresholds and show the tradeoff curve explicitly. Our measurement results are valuable for the design of a smart cache management mechanism that can minimize the cost induced by unconsumed content while still respecting a certain level of user experience.

IV. Design of Smart Cache Control Algorithm With Adaptive Thresholding

In this section, we propose a control-theoretic cache management scheme to carefully balance the tradeoff between the unconsumed content cost and the user QoE. Instead of changing the entire FSM of the Android media framework, we consider to optimize cache management by intelligently tuning cache thresholds dynamically. The reconfiguration of cache thresholds is much easier to implement and brings marginal impact to the OS kernel codes.

A. Problem Formulation

In a typical scenario of mobile video streaming, a user (or mobile client) receives video streams from the Internet via a wireless interface (e.g., 3G/4G, WiFi, WiMax). Assume that the video is streamed to the mobile client with a constant streaming rate r.

Consider a time-slotted system and each time slot lasts for τ seconds. Define $H(k)$ and $L(k)$ as the high threshold and low threshold of the playback cache at time slot k respectively. Also assume that $H(k)$ and $L(k)$ are constant within one time slot but can vary across time slots. Denote $b(k)$ as the amount of fresh content in the cache at the end of time slot k, which changes with the downloading and consumption of video data.

The unconsumed content cost caused by user abortion is determined by the amount of unconsumed content when aborting the viewing. Assume that the video playback will continue until the end of the current time slot, if a user aborts viewing in the middle of a time slot. Let w_k be the amount of wasted content in the cache when a user aborts viewing in time slot k. As the real abortion occurs at the end of time slot k, the value of w_k equals $b(k)$, which is the amount of fresh content at the end of time slot k. For the user QoE, we focus more on the fluency of video playback. Let f_k be the duration of freezing period in time slot k. We aim to minimize the value of f_k in each time slot to guarantee smooth playback.

From the perspective of a mobile client, it prefers to keep the amount of fresh content in the cache as low as possible, so as to minimize the cost of unconsumed content. In the meanwhile, it is also important to maintain a good user QoE level during video playback. However, these two objectives are conflicting with each other.

Considering the tradeoff between the unconsumed content cost and user QoE, we can define a generic utility function $\Phi(\cdot)$ to capture the impact of user preferences on the unconsumed content cost and user QoE. For example, users with WiFi connections care less about the unconsumed content cost, while users with 3G/4G connections care more about the unconsumed content cost. Such diversity can be reflected by defining different utility functions for different types of users. In this paper, we define the utility function $\Phi(\cdot)$ as a convex function of w_k and f_k. Generally, the value of the utility function decreases convexly with the increase of w_k or f_k. This property captures the fact that the marginal utility will increase more significantly when w_k or f_k becomes smaller.

Our objective is to maximize user utility $\Phi(\cdot)$ at each time slot k by intelligently tuning high and low thresholds (namely $H(k)$ and $L(k)$) so as to regulate the downloading behavior of a mobile client. The problem of cache management can then be transformed into finding the optimal threshold configuration strategy $(H^*(k), L^*(k))$ in each time slot k that can maximize the utility function $\Phi(\cdot)$. The problem formulation can be formally defined as below:

$$(H^*(k), L^*(k)) = \arg \max_{H(k),L(k)} \Phi(w_k, f_k) \tag{1}$$

Actually, w_k (or f_k) can be regarded as a function of $H(k)$, $L(k)$, and the available bandwidth. The value of w_k or f_k depends on the setting of $H(k)$ and $L(k)$, and the changes of the available bandwidth in time slot k. However, the time-varying nature of network conditions makes it difficult to obtain the accurate information of available download bandwidth at the beginning of each time slot.

B. Algorithm Design

Our problem formulation shows that the cache state and playback state in the previous time slot can be exploited as feedback signals for cache management. The value of w_k and f_k can be easily obtained at the end of each time slot k. Therefore, we can design an online cache control algorithm based on control theory [37] to optimize both unconsumed content cost and user QoE and achieve cost-effective mobile video streaming..

A user can specify a targeted control objective on the unconsumed content cost and user QoE, (w^*, f^*), in which w^* is the maximum tolerable cost of unconsumed content and f^* is the maximum tolerable duration of freezing time in each time slot. Normally we can set w^* as a small constant value and f^* as zero for fluent video playback. The amount of cached fresh content at the end of each time slot (namely $b(k)$) actually reflects the distance to the targeted control objective. w_k and f_k in the previous time slot can be the feedback signal for closed-loop control.

According to our measurement results, the cost of unconsumed data is more sensitive to the setting of high threshold, while the freezing duration is largely determined by the setting of low threshold. Therefore, by controlling the update of high threshold and low threshold, we can approach the targeted control objective (w^*, f^*) gradually.

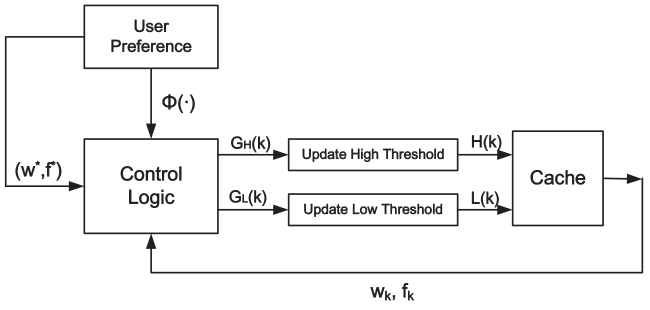


Fig. 9. Illustration of the online cache control algorithm.

Let $\Delta_w(k)$ and $\Delta_f(k)$ be the marginal utility incurred by the difference between the current state and the targeted state, which are defined as below:

$$\Delta_w(k) = \Phi(w_k, f_k) - \Phi(w^*, f_k) \tag{2}$$

$$\Delta_f(k) = \Phi(w_k, f_k) - \Phi(w_k, f^*) \tag{3}$$

We define two separate control factors $G_H(k)$ and $G_L(k)$, where ψ_h and ψ_l are two constants which determine the smoothness of two control factors.

$$G_H(k) = \frac{2e^{\psi_h \cdot \Delta_w(k)}}{1 + e^{\psi_h \cdot \Delta_w(k)}} \tag{4}$$

$$G_L(k) = \frac{2e^{\psi_l \cdot \Delta_f(k)}}{1 + e^{\psi_l \cdot \Delta_f(k)}} \tag{5}$$

The update of high threshold and low threshold can be governed by the following controllers:

$$H(k+1) = G_H(k) \cdot H(k) \tag{6}$$

$$L(k+1) = G_L(k) \cdot L(k) \tag{7}$$

For a better understanding, we plot the operation of our proposed online cache control algorithm **SCAT** (Smart Cache with Adaptive Thresholding) in Fig. 9. Initially, a user will specify the targeted control objective (w^*, f^*) and the utility function $\Phi(\cdot)$, which serve as the input for the control logic module. Combined with the feedback signal w_k (i.e., $b(k)$) from the cache and the freezing duration f_k in the past time slot, the control logic module can determine how to update the control factors $G_H(k)$ and $G_L(k)$, and then determine high threshold and low threshold accordingly. The new threshold values will be sent to the cache module for updating.

The detailed description of our proposed algorithm is given in **Algorithm 1**. Note that, SCAT updates thresholds at the beginning of each time slot. The execution of SCAT will not stop until the abortion of video viewing. The time complexity of SCAT is low, as the derivation of control factors ($G_H(k)$, $G_L(k)$) is not complicated. More specifically, its time complexity is determined by the derivation of the value of the utility function at each time slot. When updating cache thresholds, it is necessary to calculate the utility values corresponding to four different combinations of inputs. Since the cache size and freezing time can be directly obtained without further computation, the time complexity of SCAT at each time slot equals to four times of the time complexity to evaluate the utility function.

Algorithm 1 SCAT: Smart Cache With Adaptive Thresholding

```
Input:
    targeted state $(w^*, f^*)$;
    video playback rate $r$;
    time slot length $\tau$;
    user utility function $\Phi(\cdot)$.
1:  Initialize: $k = 0, b(0) = 0$.
2:  Initialize: $H(0), L(0)$ as the default values in the OS;
3:  repeat
4:      Obtain the value of $w_k$ from the cache monitor, $w_k = b(k)$;
5:      Obtain the value of freezing duration $f_k$ from the cache monitor;
6:      Use $w_k$ and $f_k$ as the feedback signals to calculate control factors $G_H(k)$ and $G_L(k)$.
7:      Derive high threshold and low threshold according to $H(k+1) = G_H(k) \cdot H(k)$ and $L(k+1) = G_L(k) \cdot L(k)$.
8:      Update high and low thresholds of the cache;
9:      Wait for a time slot $\tau$;
10:     Increase the slot index: $k = k + 1$.
11: until Quit video viewing.
```

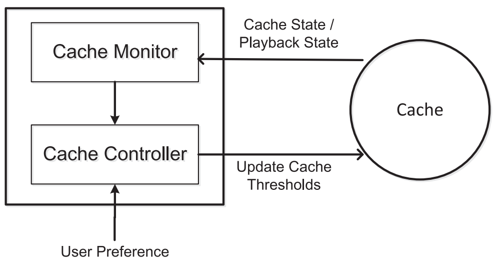


Fig. 10. Implementation of SCAT modules.

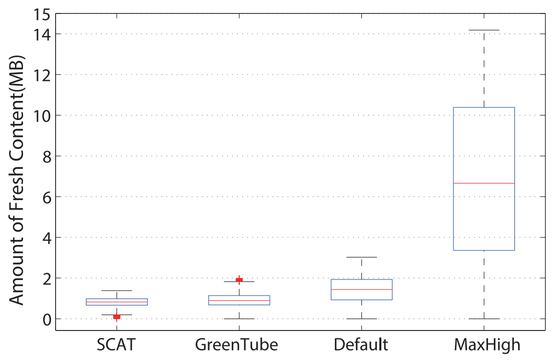
V. Implementation and Performance Evaluation

In this section, we describe how to implement our proposed algorithm in the Android platform and conduct a set of experiments to evaluate the effectiveness of our proposed cache management algorithm.

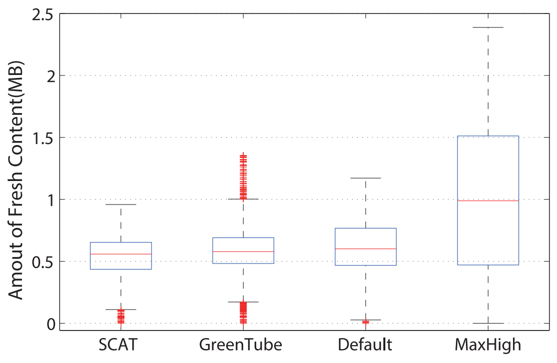
A. Implementation on the Android OS

Our proposed *SCAT* algorithm can be easily implemented on the Android OS. The software architecture of our implementation is shown in Fig. 10.

Only two lightweight modules need be added into the default Android media library: one is a *Cache Monitor* module, which keeps tracking the cache state (e.g., the amount of fresh content) and the playback state (e.g., freezing duration) in each time slot; another is a *Cache Controller* module, which dynamically adjusts the high and low thresholds of the cache according to the feedback information provided by the *Cache Monitor*. As our modules can be integrated into the Android media library, any mobile video streaming application can easily use our proposed cache management scheme. It is not necessary to make any modification of the upper-level applications. Users can choose whether to use our controller by



(a) 600Kbps



(b) 320Kbps

Fig. 11. Distribution of the amount of fresh content across all time slots in the viewing process.

simply changing the configuration file, where a user needs to specify his preference on w^*, f^* and $\Phi(\cdot)$.

The user preference is used to configure the parameters of the *Cache Controller* module. The *Cache Monitor* module sends the feedback information of the cache state to the *Cache Controller* module every time slot. During the execution, the *Cache Controller* will periodically derive and update the values of cache thresholds based on the recent feedback signals and user preference. Our algorithm can quickly approach the targeted control objective. Note that, our implementation on the Android platform can also be easily extended to other mobile platforms (e.g., iOS).

B. Experimental Settings

We reuse the testbed shown in Fig. 1 to perform experiments, which consists of a mobile smartphone with the Android OS, a laptop with the WiFi interface and a router for Internet connection. The current video streaming platform does not support DASH-based streaming. The video stream is encoded with the popular H.264 codec. DummyNet is used to control the available bandwidth to the mobile smartphone. The default video streaming rate is set as 300 Kbps, which is the same as that used in our previous measurements. In addition, Youku app is still used for performance evaluation. Note that we have not made any changes to Youku app itself. In default, the values of w^* and f^* are set as 300KB and 0. Similar to previous work [7], we can define the utility function as a convex function like $\Phi(w_k, f_k) = \frac{1}{1+w_k} + \alpha \cdot \frac{1}{1+f_k}$, where α is a weight parameter and set to 1 in default. With the increase of either w_k or f_k, the utility of a user will be decreased. Note that other convex functions that have the similar properties can also be adopted.

To better understand the pros and cons of our approach, we compare our algorithm with three other cache management schemes, namely:

- **Default**, which uses the default setting of cache thresholds in the Android OS. In default, the low and high thresholds are set as 0.5 MB and 5 MB respectively.
- **MaxHigh,** which sets the high threshold as the maximum, so that the client will not stop fetching data until the video has been completely downloaded. The low threshold is fixed and set as the default value.
- **GreenTube,** which is a cache management scheme proposed in [6]. GreenTube dynamically adjusts the high threshold to achieve power efficiency of video streaming applications on mobile devices. GreenTube also optimizes user experiences at the same time since it prefers to keep more fresh content in its cache to prolong the WNIC idle duration before viewing abortion.

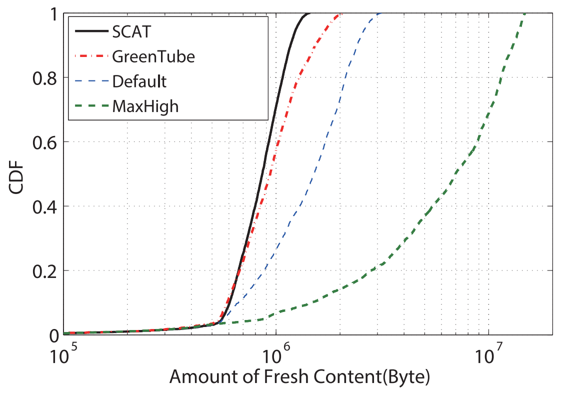
For the comparison, we also implement *MaxHigh* and *GreenTube* in the Android media library.

C. On the Cost of Unconsumed Content Incurred by Viewing Abortion

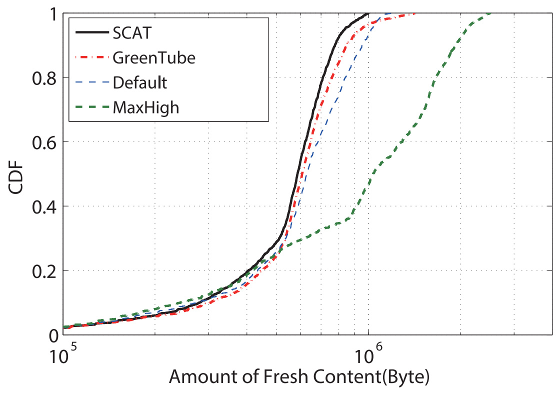
For each run of our experiment, we limit the available bandwidth to the smartphone by DummyNet and keep monitoring the evolution of the playback cache, especially the amount of fresh content in each time slot. The amount of fresh content determines the unconsumed content incurred by viewing abortion. The experiments are repeated multiple times under different bandwidth settings, and we collect all the log information in the viewing process.

Fig. 11 shows the distribution of the amount of fresh content across all time slots under different bandwidth settings. In the high-bandwidth scenario (i.e., 600 Kbps), it is unsurprising to see that *MaxHigh* buffers the highest amount of fresh content in the cache, as the client keeps fetching new contents. SCAT achieves the lowest level of fresh content in the cache, which is about half of that under the default Android setting. GreenTube caches a slightly higher amount of fresh content than that of SCAT. When the available bandwidth decreases to 320 Kbps, in spite that SCAT still has the lowest cache level, the difference with other schemes is not as significant as that under 600 Kbps. Even when using the *MaxHigh* scheme, the average amount of fresh content is only around 1 MB. The reason is because that the accumulation of fresh content is slow under a low-bandwidth scenario.

To examine the evolution of fresh content in the cache, we keep monitoring the amount of fresh content across all time slots. We plot the cumulative distribution of the amount of



(a) Network Bandwidth: 600Kbps



(b) Network Bandwidth: 320Kbps

Fig. 12. CDF of the amount of fresh content across all the time slots.

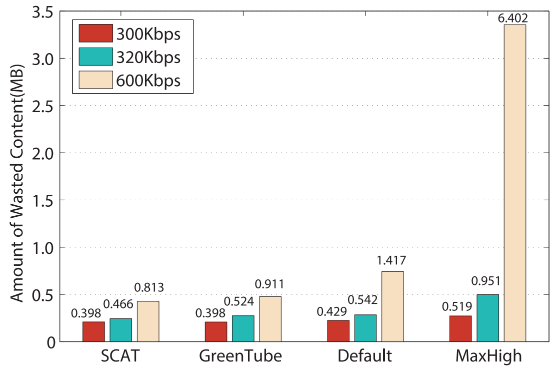


Fig. 13. Average amount of wasted content under different schemes.

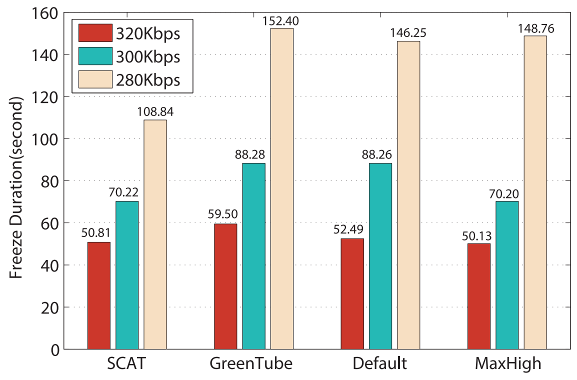


Fig. 14. Average freezing duration under different schemes.

fresh content across all time slots in Fig. 12. In the high-bandwidth case (see Fig. 12(a)), the amount of fresh content cached by *MaxHigh* is much higher than that of other three alternatives in most of time slots. The *MaxHigh* and *Default* schemes cache more than 1 MB fresh content over 90% and 70% of time respectively. On the contrary, the amount of fresh content cached by *SCAT* is over 1 MB in less than 30% of time. *GreenTube* achieves comparable performance to that of *SCAT*, and caches over 1 MB fresh content in around 40% of time. In the low-bandwidth case (see Fig. 12(b)), *MaxHigh* still caches the highest amount of fresh content, and the other three schemes cache comparable amount of fresh content. Our proposed *SCAT* scheme performs slightly better than *GreenTube* and *Default*.

To better evaluate the effectiveness of different schemes, we simulate user viewing abortion behaviors and record the amount of unconsumed content. In order to reduce the variance, we run the experiment multiple times under different bandwidth settings and plot the average amount of wasted content in Fig. 13. The results show the clear advantage of *SCAT* under the high-bandwidth scenario (i.e., 600 Kbps). *SCAT* reduces the amount of unconsumed content due to viewing abortion by 87.3% compared with *MaxHigh*, 42.6% compared with *Default* and 10.7% compared with *GreenTube*. It is because that *SCAT* will lower the high threshold to reduce the possible unconsumed content once the amount of fresh content is high, while static schemes like *Default* or *MaxHigh* will always accumulate too much fresh content in the cache due to the unreasonable high threshold. In addition, *SCAT* also outperforms *GreenTube*, as *GreenTube* only minimizes unconsumed content at its expected abortion time point.

D. On the User Experience During Viewing

In this section, we study user experience under different cache management schemes. Especially, we focus on freezing occurrence in the viewing process. We conduct the experiments under different bandwidth settings multiple times and collect all the information of freezing events.

With collected logs, we further plot the average freezing duration under different schemes in Fig. 14. When the available bandwidth is high enough (e.g., higher than 400Kbps), no freezing occurs for all the four schemes in the viewing process. Thus, it is not very meaningful to compare different schemes under the high-bandwidth environment. We only conduct comparisons under bandwidth settings close to the video playback rate (namely, 320Kbps, 300Kbps and 280Kbps in our experiments). When the bandwidth drops to 280Kbps, which is lower than the video playback rate, the average freezing duration of all schemes increases greatly. However, compared to other alternatives, our proposed SCAT algorithm still achieves the lowest freezing duration. The reduction

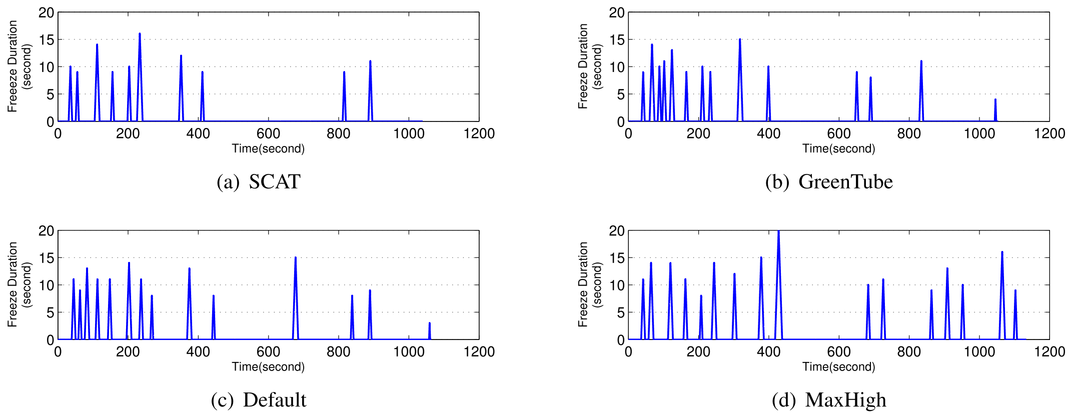


Fig. 15. Freezing occurrence when using different schemes under 280 Kbps.

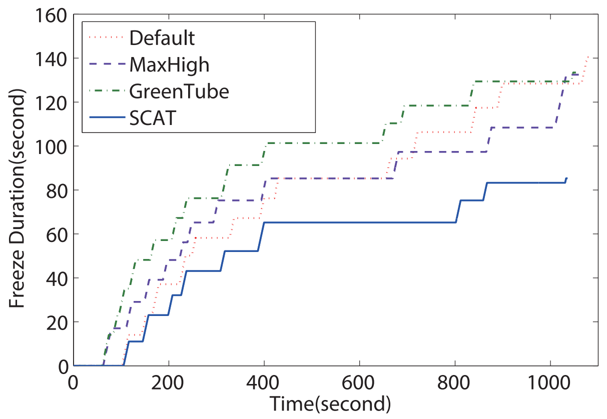


Fig. 16. Cumulative freezing duration under 280 Kbps.

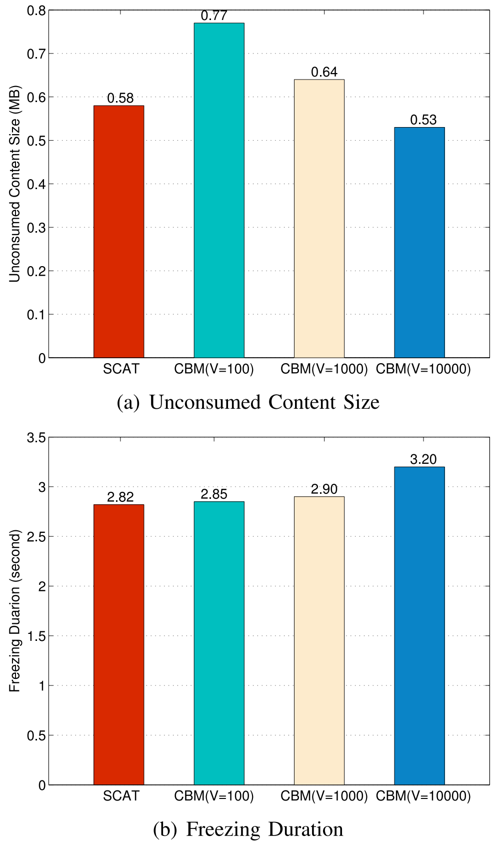


Fig. 17. Comparison between SCAT and CBM.

of freezing duration is around 30% compared to other three schemes. *GreenTube* even performs worse than *Default* and *MaxHigh*. With the increase of available bandwidth, the freezing duration of all four schemes can be reduced. The benefit brought by using SCAT is not that significant. However, our SCAT algorithm still performs close to the best one (i.e., *MaxHigh*) in terms of minimizing freezing duration.

Fig. 15 further shows the details of freezing occurrence during the viewing process. When the bandwidth is insufficient to support the video playback rate, freezing occurs very frequently no matter which scheme is used. Especially in the first 400 seconds, as the amount of fresh content is low and there is no enough bandwidth to accumulate more content, the video player has to enter into the freezing state more frequently. However, there is still a bit of difference among four schemes. After 400 seconds, the number of freezing events can be reduced when using SCAT.

To better understand the effects of our SCAT algorithm, we look closely at one round of experiments and plot the cumulative freezing duration in Fig. 16. In the beginning phase, there is a buffering period for initial playback start-up, which is not considered as freezing in our calculation. From the figure, we can observe that SCAT incurs the shortest cumulative freezing duration among all four schemes.

We have also conducted a set of simulation-based experiments to compare SCAT with CBM. The results in Fig. 17 show that SCAT can achieve similar performance as that of CBM (note that V is a tunable parameter of CBM algorithm). In the experiments, the streaming rate is 300kbps and the download rate is 30% higher than the streaming rate.

In summary, our results point out that our proposed SCAT algorithm can achieve a good balance between the unconsumed content cost and user QoE. By using a control-theoretic approach, the SCAT algorithm can reduce the unconsumed content cost and mitigate freezing occurrence simultaneously.

VI. Conclusion

Cache management has significant impacts on the cost and performance of mobile video streaming. In this paper, we first built a dedicated testbed to measure how cache thresholds affect various user-related metrics of mobile video streaming applications, such as unconsumed content cost due to viewing abortion, freezing duration, etc. From our results, we identified an implicit tradeoff between unconsumed content cost and user QoE when varying the threshold. Motivated by our findings, we further proposed a control-theoretic cache management algorithm called SCAT, which can achieve cost-effective mobile video streaming by dynamic adjustment of cache thresholds. We also implemented and evaluated our proposed algorithm on the Android OS. Our results show that, our proposed SCAT can save the cost of unconsumed content compared with other schemes and significantly shorten the freezing duration in the low-bandwidth environment. In the future work, we will extend our algorithm to take more metrics related to user experiences into account. The extension is still based on adaptive cache threshold tuning to ensure the implementation be simple and practical. In the experiments, we will also evaluate more measures related to user-perceived quality, such as startup delay, playback delay, etc.

References

[1] (2013). *Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update*. [Online]. Available: http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white_paper_c11-520862.html

[2] (2015). *Mobile/Tablet Operating System Market Share*. [Online]. Available: http://www.netmarketshare.com/operating-system-market-share.aspx?qprid = 8&qpcustomd = 1

[3] A. Finamore, M. Mellia, M. M. Munafò, R. Torres, and S. G. Rao, "YouTube everywhere: Impact of device and infrastructure synergies on user experience," in *Proc. Internet Meas. Conf. (IMC)*, Berlin, Germany, 2011, pp. 345–360.

[4] A. Rao *et al.*, "Network characteristics of video streaming traffic," in *Proc. CoNEXT*, Tokyo, Japan, 2011, Art. ID 25.

[5] Y. Liu, F. Li, L. Guo, B. Shen, and S. Chen, "Effectively minimizing redundant Internet streaming traffic to iOS devices," in *Proc. IEEE INFOCOM*, Turin, Italy, 2013, pp. 250–254.

[6] X. Li, M. Dong, Z. Ma, and F. C. Fernandes, "GreenTube: Power optimization for mobile videostreaming via dynamic cache management," in *Proc. 20th ACM Int. Conf. Multimedia*, Nara, Japan, 2012, pp. 279–288.

[7] J. He, Z. Xue, D. Wu, D. O. Wu, and Y. Wen, "CBM: Online strategies for cost-aware buffer management over mobile video streaming," *IEEE Trans. Multimedia*, vol. 16, no. 1, pp. 242–252, Jan. 2014.

[8] J. Erman, A. Gerber, K. Ramadrishnan, S. Sen, and O. Spatscheck, "Over the top video: The gorilla in cellular networks," in *Proc. ACM SIGCOMM Conf. Internet Meas. Conf.*, Berlin, Germany, 2011, pp. 127–136.

[9] N. Balasubramanian, A. Balasubramanian, and A. Venkataramani, "Energy consumption in mobile phones: A measurement study and implications for network applications," in *Proc. ACM SIGCOMM Conf. Internet Meas. Conf. (IMC)*, Chicago, IL, USA, 2009, pp. 280–293.

[10] A. Pathak, Y. Hu, and M. Zhang, "Where is the energy spent inside my app? Fine grained energy accounting on smartphones with Eprof," in *Proc. EuroSys*, Bern, Switzerland, 2012, pp. 29–42.

[11] Y. Liu, F. Li, L. Guo, and S. Chen, "A measurement study of resource utilization in Internet mobile streaming," in *Proc. ACM NOSSDAV*, Vancouver, BC, Canada, 2011, pp. 33–38.

[12] Y. Xiao, R. S. Kalyanaraman, and A. Yla-Jaaski, "Energy consumption of mobile YouTube: Quantitative measurement and analysis," in *Proc. IEEE 2nd Int. Conf. Next Gener. Mobile Appl. Serv. Technol.*, Cardiff, U.K., 2008, pp. 61–69.

[13] F. Qian *et al.*, "Profiling resource usage for mobile applications: A cross-layer approach," in *Proc. 9th Int. Conf. Mobile Syst. Appl. Serv.*, Washington, DC, USA, 2011, pp. 321–334.

[14] Y. Li, Y. Zhang, and R. Yuan, "Measurement and analysis of a large scale commercial mobile Internet TV system," in *Proc. ACM SIGCOMM Conf. Internet Meas. Conf. (IMC)*, Berlin, Germany, 2011, pp. 209–224.

[15] Y. Liu, F. Li, L. Guo, B. Shen, and S. Chen, "A server's perspective of Internet streaming delivery to mobile devices," in *Proc. INFOCOM*, Orlando, FL, USA, 2012, pp. 1332–1340.

[16] N. Staelens *et al.*, "Subjective quality assessment of longer duration video sequences delivered over HTTP adaptive streaming to tablet devices," *IEEE Trans. Broadcast.*, vol. 60, no. 4, pp. 707–714, Dec. 2014.

[17] J. Yang, H. Hu, H. Xi, and L. Hanzo, "Online buffer fullness estimation aided adaptive media playout for video streaming," *IEEE Trans. Multimedia*, vol. 13, no. 5, pp. 1141–1153, Oct. 2011.

[18] S. Mohapatra, R. Cornea, N. Dutt, A. Nicolau, and N. Venkatasubramanian, "Integrated power management for video streaming to mobile handheld devices," in *Proc. 11th ACM Int. Conf. Multimedia*, Berkeley, CA, USA, 2003, pp. 582–591.

[19] Y. Liu, L. Guo, F. Li, and S. Chen, "An empirical evaluation of battery power consumption for streaming data transmission to mobile devices," in *Proc. ACM Multimedia*, Scottsdale, AZ, USA, 2011, pp. 473–482.

[20] Y. Liu, F. Li, L. Guo, and S. Chen, "BlueStreaming: Towards power-efficient Internet P2P streaming to mobile devices," in *Proc. ACM Multimedia*, Scottsdale, AZ, USA, 2011, pp. 193–202.

[21] M. van der Schaar and S. Shankar, "Cross-layer wireless multimedia transmission: Challenges, principles, and new paradigms," *IEEE Wireless Commun.*, vol. 12, no. 4, pp. 50–58, Aug. 2005.

[22] N. Mastronarde and M. van der Schaar, "Online reinforcement learning for multimedia buffer control," in *Proc. ICASSP*, Dallas, TX, USA, 2010, pp. 1958–1961.

[23] N. Ding *et al.*, "Realizing the full potential of PSM using proxying," in *Proc. Infocom*, Orlando, FL, USA, 2012, pp. 2821–2825.

[24] R. Bhatia, T. Lakshman, A. Netravali, and K. Sabnani, "Improving mobile video streaming with link aware scheduling and client caches," in *Proc. IEEE INFOCOM*, Toronto, ON, Canada, 2014, pp. 100–108.

[25] Y. Liu, F. Li, L. Guo, B. Shen, and S. Chen, "A comparative study of android and iOS for accessing Internet streaming services," in *Passive and Active Measurement*. Berlin, Germany: Springer, 2013, pp. 104–114.

[26] S. Jia, C. Xu, J. Guan, H. Zhang, and G.-M. Muntean, "A novel cooperative content fetching-based strategy to increase the quality of video delivery to mobile users in wireless networks," *IEEE Trans. Broadcast.*, vol. 60, no. 2, pp. 370–384, Jun. 2014.

[27] C.-H. Yeh, S.-J. F. Jiang, C.-Y. Lin, and M.-J. Chen, "Temporal video transcoding based on frame complexity analysis for mobile video communication," *IEEE Trans. Broadcast.*, vol. 59, no. 1, pp. 38–46, Mar. 2013.

[28] U. Ukommi, H. K. Arachchi, S. Dogan, and A. Kondoz, "Content-aware bitrate adaptation for robust mobile video services," in *Proc. IEEE Int. Symp. Broadband Multimedia Syst. Broadcast. (BMSB)*, London, U.K., 2013, pp. 1–4.

[29] C. Xu, S. Jia, L. Zhong, H. Zhang, and G.-M. Muntean, "Ant-inspired mini-community-based solution for video-on-demand services in wireless mobile networks," *IEEE Trans. Broadcast.*, vol. 60, no. 2, pp. 322–335, Jun. 2014.

[30] J. Ma, X. Deng, Y. Liu, and D. Wu, "Power consumption of mobile video streaming under adverse network conditions," in *Proc. IEEE Int. Conf. Commun. China (ICCC)*, Xi'an, China, 2013, pp. 106–111.

[31] J. Huang, D. Wu, and J. He, "Demystifying the magic of cache thresholds in the Android media framework," in *Proc. Int. Conf. Wireless Commun. Signal Process. (WCSP)*, Hefei, China, 2014, pp. 1–6.

[32] (2015). *Youku Homepage*. [Online]. Available: http://mobile.youku.com/index/

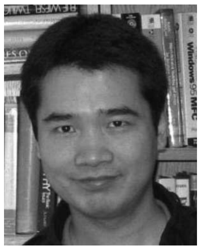
[33] (2015). *The Dummynet Project Homepage*. [Online]. Available: http://info.iet.unipi.it/∼luigi/dummynet/

[34] (2015). *The IPFIREWALL*. [Online]. Available: http://www.freebsd.org/doc/e_US.ISO8859-1/books/handbook/firewalls-ipfw.html

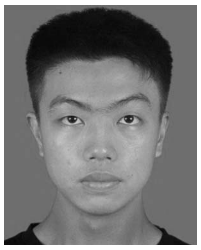
[35] (2015). *Wireshark Homepage*. [Online]. Available: http://www.wireshark.org/

[36] (2010). *An Overview of Stagefright Player*. [Online]. Available: http://freepine.blogspot.sg/2010/01/overview-of-stagefrighter-player.html

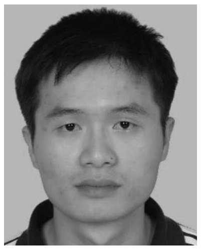
[37] E. B. Lee and L. Markus, *Foundations of Optimal Control Theory*. New York, NY, USA: Wiley, 1967.



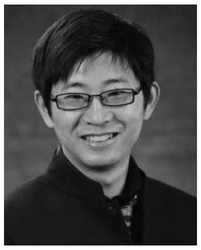
Di Wu (M'06) received the B.S. degree from the University of Science and Technology of China in 2000, the M.S. degree from the Institute of Computing Technology, Chinese Academy of Sciences in 2003, and the Ph.D. degree in computer science and engineering from the Chinese University of Hong Kong in 2007. He is a Professor and the Associate Department Head of the Department of Computer Science, Sun Yat-sen University, Guangzhou, China. From 2007 to 2009, he was a Post-Doctoral Researcher with the Department of Computer Science and Engineering, Polytechnic Institute of NYU, advised by Prof. K. W. Ross. His research interests include multimedia communication, cloud computing, peer-to-peer networking, Internet measurement, and network security. He was a co-recipient of the IEEE INFOCOM 2009 Best Paper Award. He has served as an Editor for the *Springer Journal of Telecommunication Systems*, the *Journal of Communications and Networks*, *Springer Peer-to-Peer Networking and Applications*, *Wiley Security and Communication Networks*, and the *KSII Transactions on Internet and Information Systems*, and a Guest Editor of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY. He has also served as the MSIG Chair of the Multimedia Communications Technical Committee in the IEEE Communications Society from 2014 to 2016, the TPC Co-Chair of the IEEE GLOBECOM-CCSNA 2014, the Chair of CCF YOCSEF-Guangzhou from 2014 to 2015, and a member of the Council of China Computer Federation.



Jian Huang received the B.S. degree from Sun Yat-sen University in 2012. He is currently pursuing the master's degree with the Department of Computer Science, Sun Yat-sen University, Guangzhou, China, under the supervision of Prof. D. Wu. His research interests include cloud computing, data center networking, content distribution, and software defined networking.



Jian He received the B.S. and M.S. degrees from Sun Yat-sen University in 2011 and 2014, respectively. He is currently pursuing the Ph.D. degree with the Department of Computer Science, University of Texas at Austin. His research interests include content distribution networks, data center networking, green networking, and network measurement.



Min Chen (M'08–SM'09) is a Professor with the School of Computer Science and Technology, Huazhong University of Science and Technology. He was an Assistant Professor with the School of Computer Science and Engineering, Seoul National University (SNU) from 2009 to 2012. He was the Research and Development Director with Confederal Network Inc., from 2008 to 2009. He was a Post-Doctoral Fellow with SNU for one and a half years and the Department of Electrical and Computer Engineering, University of British Columbia for three years. His research focuses on Internet of things, big data, machine to machine communications, body area networks, e-healthcare, mobile cloud computing, *ad hoc* cloudlet, cloud-assisted mobile computing, ubiquitous network and services, and multimedia transmission over wireless network. He has over 180 paper publications, including 85 SCI papers. He was a recipient of the Best Paper Award from the IEEE ICC 2012 and the Best Paper Runner-Up Award from QShine 2008. He is the Chair of the IEEE Computer Society Special Technical Communities on Big Data. He serves as an Editor or an Associate Editor for *Information Sciences*, *Wireless Communications and Mobile Computing*, *IET Communications*, *IET Networks*, the *Wiley International Journal of Security and Communication Networks*, the *Journal of Internet Technology*, the *KSII Transactions on Internet and Information Systems*, and the *International Journal of Sensor Networks*. He is the Managing Editor for IJAACS and IJART. He is a Guest Editor of the IEEE NETWORK and the *IEEE Wireless Communications Magazine*. He is the Co-Chair of the IEEE ICC 2012—Communications Theory Symposium and the IEEE ICC 2013—Wireless Networks Symposium. He is the General Co-Chair of the IEEE CIT-2012 and Mobimedia 2015. He is the General Vice Chair for Tridentcom 2014. He is a Keynote Speaker for CyberC 2012 and Mobiquitous 2012. He is a TPC Member of the IEEE INFOCOM 2013 and 2014.

Guoqing Zhang received the Ph.D. degree from the Institute of Computing Technology, Chinese Academy of Sciences, where he is currently a Principal Investigator. His research interests include information networks and network science, especially topology analysis of the Internet, online social networks and big data, as well as topology-aware optimization and application.