Study on Dynamic Adaptive Bitrate Selection Algorithm for Mobile Streaming Media

Taoshen Li\textsuperscript{1,2}, Zhihui Ge\textsuperscript{2}, and Jichang Chen\textsuperscript{1,*}

\textsuperscript{1}Nanning University, School of Information Engineering, 530199 Nanning, China
\textsuperscript{2}Guangxi University, School of Computer, Electronics and Information, 530004 Nanning, China

Abstract. An energy-aware dynamic adaptive Bitrate selection algorithm (EDABS) was proposed to solve the problem of rate switching frequency and low energy utilization efficiency while receiving streaming media data. According to the monitored network bandwidth and streaming media coding rate, the algorithm adaptively adjusted the lower limit threshold of the terminal cache area, so that the size of the cache space was more fit the current bandwidth and streaming media coding rate. And then, it set a dynamic security boundary and response delay factor, and adaptively selected the next target video block's code rate. The experimental results show that proposed algorithm can better reduce rate switching frequency and improve the utilization efficiency of the energy.

1 Introduction

Dynamic adaptive streaming over HTTP (DASH) is a widely used standard for adaptive video streaming over HTTP, which can provide users with better quality of experience (QoE)[1]. In DASH, the dynamic adaptive bitrate selection algorithm is the core technology to enables high quality streaming of media content over the Internet delivered from conventional HTTP web servers. A good adaptive bitrate algorithm not only continually adjust the bitrate of the video segments that are downloaded and rendered to the user, but also is responsive to user and network events and can be used in demanding scenarios such as low-latency live streaming[2].

With the popularization of wireless applications in mobile terminals, more attention has been paid to the energy saving of mobile multimedia streaming transmission. In order to save the power consumption of network and equipment, it is necessary to adjust the rate dynamically according to the resource load in the actual operation of network system[3-4]. In [5], a general energy consumption model of data transmission and reception was established, and a rate adaptive algorithm based on energy perception was proposed. In [6], the relationship between data transmission rate and power consumption rate was obtained by linear fitting method, and then the current remaining power was integrated to select the appropriate streaming media coding rate. [7] applied the optimal resource allocation technology to the energy consumption optimization problem, and proposed a method to allocate scarce resources according to the terminal hardware and software capabilities. However, this algorithm may cause that some single terminal cannot get the required resources all the time, and the fairness is not considered properly. In [8], it was pointed out...
that dynamic adjustment of cache size can obtain energy consumption benefits, but no specific implementation scheme is given. [9] proposed an energy-aware DASH delivery framework over LTE networks (EDASH) to jointly optimize the network throughput, users' QoE and UEs' energy efficiency. In general, the above energy consumption model and energy consumption optimization strategy do not consider the limited capacity of the user device cache. While focusing on energy saving, they have to reduce the user's QoE.

In fact, the energy efficiency of mobile devices is an important factor affecting user experience. However, the existing dynamic adaptive bitrate algorithms of WiFi network rarely consider the problem of terminal energy consumption. This paper mainly studies the energy consumption of wireless port of mobile terminal when receiving streaming media data, and proposes a new energy-aware bitrate selection algorithm, which aims to reduce the energy consumption of mobile terminal (mobile phone) as much as possible on the premise of providing users with good QoE.

2 Description of algorithm

2.1 Design idea of algorithm

From the existing mobile streaming media transmission system, the server and the client are connected through WLAN or cellular mobile data network. Therefore, while designing of dynamic adaptive rate selection algorithm, we mainly consider three factors related to wireless channel and terminal characteristics, such as the limitation of battery power, the influence of dynamic changes of network throughput, and the cache management of client video playback. Therefore, we propose an energy-aware dynamic adaptive bitrate selection algorithms (EDABS) in this paper. The design idea of the algorithm is as follows: (1) the next target video block's bit rate is selected adaptively according to the current cache status, video rate history and TCP throughput history of the mobile terminal; (2) the lower limit thresholds of the cache area of the wireless port when opening and closing are adjusted dynamically to reduce the rate switching frequency. At the same time of satisfying QoE, the algorithm will make the sleep time of the wireless interface as long as possible, so as to save the energy consumption of the terminal battery and extend the endurance time of the terminal. The specific implementation of the algorithm is described below.

2.2 Predicting available bandwidth value

In the actual operation of video system, the instantaneous rate of media stream changes in real time. In this paper, weighted average prediction algorithm is used to calculate the average rate of media stream in the next measurement period. The formula is as follows:

\[ \overline{A}_{n+1} = (1-\delta)\overline{A}_n + \delta \overline{A}_n \]  

(1)

where, \( \overline{A}_n \) is the variable that denotes the historical rate of media stream in the measurement period, and records the average value of the data receiving rate in the last \( num \) measurement periods; \( \overline{A}_n \) is the average rate of media stream in this measurement period, \( \overline{A}_{n+1} \) is the average rate in the next period; \( \delta (0 \leq \delta < 1) \) is the weighting coefficient. In this paper, \( num = 6, \delta = 0.25 \).
2.3 Lower threshold management of cache

It is assumed that the upper threshold value of the mobile terminal's cache is fixed, that is, a fixed size cache space is allocated at the beginning of the session, and the cycle frequency, the size of the download content block and the download duration are determined by the lower threshold value. Obviously, the lower the threshold value, the lower the frequency of cache data reload events, and the less the average power consumption. However, if the lower threshold is set too low, there is an increased risk of playback pauses. Therefore, a trade-off between saving energy and reducing the risk of cache underflow is needed when selecting the lower cache threshold value. In this paper, the variable $k (k \in (0, 1])$ is defined to represent the buffer underflow risk, and it satisfies the following formula:

$$k = \sqrt{\frac{D \lambda}{B_{\text{low}}}}$$  \hspace{1cm} (2)

where, $D$ is the average playback rate, $\lambda$ is the delay between the opening of the port and the actual read data in the cache, and $B_{\text{low}}$ is the lower threshold. When $k > 1$, the value of $k$ is regarded as 1. The larger the value of $k$, the more probability the cache underflow is. According to the results of many experiments, we divide $k$ into four levels to represent different risk levels: $k \in (0, 0.3]$ is safety level, $k \in (0.3, 0.5]$ is slight risk level, $k \in (0.5, 0.8]$ is moderate risk level, and $k \in (0.8, 1]$ is serious risk level. For different risk levels, different adjustment factors are used to prevent buffer underflow. The adjustment of the lower threshold should consider the fluctuation of TCP throughput and the rate of cache consumption. The adjustment formula is as follows:

$$B_{\text{low}}^{\text{new}} = B_{\text{low}}^{\text{old}} + \omega \times (B_{\text{up}} - B_{\text{low}}^{\text{old}})/D$$  \hspace{1cm} (3)

$$\omega = \frac{1}{\text{num}} \sum_{k=\text{num}}^{n} \frac{\overline{A}(k) - \overline{A}(k-1)}{\overline{A}(k-1)}$$  \hspace{1cm} (4)

where, $B_{\text{low}}^{\text{new}}$ is the calculated new lower threshold value, $B_{\text{low}}^{\text{old}}$ is the lower threshold value of the previous round; $\omega$ is the increment factor (can be positive or negative), which indicates the fluctuation of TCP throughput of last $\text{num}$ video block cycles and is used to adjust the increase or decrease of threshold value; $B_{\text{up}}$ is the upper threshold value of cache; $D$ is the average playback rate.

2.4 Bitrate adaptive adjustment

To maintain continuous video playback without interruption, the requested video rate $V_{(k+1)}$ cannot be higher than the available TCP throughput (i.e. available bandwidth) during video download. At the same time, in order to maximize the video quality, the requested video rate should be as close to the TCP throughput as possible. Therefore, we need to estimate TCP throughput in real time $\overline{A}(k+1)$. Among the existing methods, the based prediction method based on simple history has high accuracy[10].

Because the wireless link is more unstable than the wired link, the actual download rate of the next video block may be significantly different from the TCP throughput predicted based on historical data. If the target video rate is directly set to the predicted TCP throughput, once the prediction of the TCP throughput is too optimistic, the video data to input into the buffer will not keep up with the playback speed, and then the playback will be stuck. To solve this problem, we insert a dynamic safe margin (DSM) between the target video rate $V_{(k+1)}$ and the predicted TCP throughput $\overline{A}(k+1)$, and sets the target video rate to be slightly less than the predicted TCP throughput.

The setting of DSM is related to the fluctuation of TCP throughput. If the increment factor $\omega$ is large, the DSM will be set to be larger to ensure that the selected code rate of the
algorithm is conducive to the smooth play of the terminal, and vice versa. The DSM\((k+1)\) is calculated as follows:

\[ DSM(k+1) = 1 - \frac{20 + 75e^\omega}{100e^\omega} \]  

(5)

It can be seen from equation (5) that the value of DSM \((k+1)\) changes with \(\omega\), and the value range is \([0.05, 0.25]\). If the TCP throughput hardly fluctuates in last num video block periods, i.e. \(\omega=0\), then the value of DSM \((k+1)\) is 0.05. If the TCP throughput fluctuates violently, the value of DSM \((k+1)\) is 0.25.

After determining the TCP throughput prediction value and DSM \((k+1)\), the following formula can be used to calculate the target video bit rate:

\[ V(k+1) = \tilde{A}(k+1) \times (1 - DSM(k+1)) \]  

(6)

Because the DASH system only provides a limited discrete video rate \(\{V_i\}\), it is necessary to quantify the calculated target video rate. In the specific processing, the maximum video bitrate less than \(V(k+1)\) is selected as the final target video bitrate from the limited discrete bit rate to be selected. The calculation formula is as follows:

\[ \hat{V}(k+1) = \underset{\{V_i\}; V(k+1)}{\text{max}} V_i \]  

(7)

In the design and implementation of the algorithm, we do not directly use the maximum available video bitrate as the bitrate of the target video block, but consider the buffer video time and the latest video bitrate comprehensively. Our purpose is to enable the algorithm to adjust the video speed adaptively in a smooth way, and avoid the playback interruption caused by the empty buffer.

### 2.5 Switching delay factor

In order to ensure that the actual TCP throughput can support a higher video bit rate, keep the video playing smoothly and avoid frequent switching rate, we set a switching delay factor \(m\) to determine the smoothness and adaptability of the EDABS algorithm. In the specific implementation process, even if the video bitrate provided by network prediction is higher than the current video bitrate, it is not to immediately increase the video bitrate, but to wait for \(m\) times of prediction results. If the continuous \(m\) times prediction results are higher than the current video bitrate, the video bitrate will be increased; otherwise, the original bitrate will remain unchanged. Obviously, the larger the value of \(m\) is, the stronger the ability of EDABS algorithm to smooth the change of TCP throughput is, but the slower the algorithm responds to the sudden increase of bandwidth. The smaller the value of \(m\) is, the faster the EDABS algorithm can respond to a higher or lower rate received by TCP, which can increase or decrease the video rate rapidly. However, the algorithm is also more vulnerable to the impact of TCP throughput estimation errors. In EDABS algorithm, the value of \(m\) will change dynamically with the increasing trend of TCP throughput. In order to avoid unreasonable rate switching caused by a data exception, we use the following formula to calculate the average value of \(m\) value obtained three times as the final switching delay factor:

\[ m(k) = \begin{cases} 
\Delta A(k+1) = A(k+1) - A(k) \\
4 & \Delta A(k+1) \in [0.4A(k), A(k)] \\
8 & \Delta A(k+1) \in [0.2A(k), 0.4A(k)] \\
13 & \Delta A(k+1) \in [0, 0.2A(k)] \\
22 & \text{otherwise.} 
\end{cases} \]  

(8)
2.6 Description of algorithm

Suppose that $B_{up}$ and $B_{low}$ respectively represent the upper and lower thresholds of mobile terminal buffer, $B(k+1)$ is the amount of data to be cached in the next video block period, $\hat{V}(k+1)$ is the predicted video rate higher than the current one, and DSM $(k+1)$ is the dynamic security boundary. The pseudo code of EDABS algorithm is described as follows:

1: if $B(k+1) < \frac{(B_{up} - B_{low})}{2}$ then
2: $V(k+1) = Q(A(k)(1 - DSM(k+1)))$;
3: return;
4: else
5: $V(k+1) = Q(A(k+1)(1 - DSM(k+1)))$;
6: if $\hat{V}(k+1) \geq V(k)$ then
7: Counter++;
8: if Counter $> m$ then
9: $V(k+1) = \hat{V}(k+1)$;
10: Counter = 0;
11: return;
12: end if
13: else
14: Counter = 0;
15: end if
16: end if
17: $V(k+1) = V(k)$;
18: if $B(k+1) - B_{low} > 0$ then /* Sleep, save energy */
19: $I = B(k+1) - B_{low}$;
20: else
21: $I = 0$;
22: end if
23: Idle(I);
24: return;

In the EDABS algorithm, the rate increase, rate reduction and the sleep of wireless terminal are considered respectively. In order to avoid video carton, as long as the amount of cached data is lower than the expected value $(B_{up} - B_{low})/2$, the video bit rate is immediately lowered. To ensure the smooth play of video, the algorithm uses the parameter $m$ to determine the degree of smoothness and adaption. In algorithm, in order to avoid buffer overflow, as long as the current mobile terminal's buffer is higher than the upper limit threshold of the buffer, the wireless interface will be in sleep mode. During the sleep of wireless terminal, the video will continue to play, and there is no HTTP block download request. Therefore, the algorithm can achieve considerable energy-saving effect by reasonable scheduling.

3 Experiment and performance analysis

3.1 Experimental environment settings

The experimental platform is windows7 + VMware 9.0 + Ubuntu 12.04, the network simulation software is NS3.19. In experiment, the server and the client are connected by node 1 and node 2. The server, node 1 and node 2 are connected by wired link, the link
bandwidth is 6Mbps, and they do not constitute a communication bottleneck. The client and node 2 are connected by wireless link, and the wireless link bandwidth fluctuates randomly between 2.5mbps. The server stores 8 kinds of files with encoding rate (1.6mbps, 1.8mbps, 2.0mbps, 2.2mbps, 2.4mbps, 2.6mbps, 2.8mbps, 3.0mbps) in advance, and the client buffer size is set to 40MB. In order to get closer to the real network situation, the exponential flow generator (EXP) is added to simulate the background flow in the network. At the same time, the constant flow generator (CBR) is used to control the available bandwidth of the client. EXP always exists in the whole simulation experiment, and CBR only exists in a certain period of time. The research results of [5] show that there is a positive correlation between speed and power consumption in a certain range. Therefore, the energy consumption model of the experiment in this paper is as follows:

\[ E(t) = t_{on} \times P_{data} + t_{off} \times P_{nodata} + E_{switch} \]  

\[ P_{data} = \alpha \times v + \beta \]  

where, \( \alpha \) is the correlation coefficient, \( \beta \) is a constant. At the rate \( v \in (1.2 \text{Mbps}, 6 \text{Mbps}) \), \( \alpha \) is 0.8, \( \beta \) is 0.125.

The purpose of the experiment is to test the validity of EDABS algorithm. The comparison algorithm is the traditional greedy rate adaptive algorithm. The performance evaluation indexes are the rate switching frequency and the power consumption of the same size streaming media. The time of simulation experiment is 600s.

3.2 Experimental results and analysis

3.2.1 Switching delay factor \( m \)

Firstly, the influence of switching delay factor \( m \) on switching performance in EDABS algorithm is investigated. In experiment, \( m \) is set 4, 8, 13 and 22 respectively, and other conditions are the same. The performance evaluation indexes are switching frequency \( \psi \) and bandwidth utilization \( \Phi \). The \( \psi \) is the ratio of the number of video rate changes in the simulation time to the simulation time, which represents the fluctuation of bitrate selection. For convenience, we compare the rate switching frequency in the same time in experiment. The \( \Phi \) is the ratio of average video bit rate to average TCP throughput. The larger the value of \( \Phi \) is, the better the bandwidth utilization is.

Figure 1 illustrates the influence of the switching delay factor \( m \) on the algorithm. It can be seen from the figure that the smoothing parameter \( m \) plays a decisive role in the smoothness and adaptive degree of the algorithm. The larger \( m \) is, the less the number of bitrate switching is, the smaller the switching frequency \( \psi \) is, the stronger the algorithm’s ability to smooth the change of TCP throughput is, but the slower the algorithm’s response to sudden bandwidth increase is, the lower the bandwidth utilization is. The smaller \( m \) is, the faster the algorithm can respond when the TCP throughput increases. If the selected video rate is increased, the bandwidth utilization rate is high, but at the same time, the video rate switching frequency will be high and sometimes there is a direct conversion from a higher bit rate to a lower bit rate, resulting in a large fluctuation in the video viewing experience obtained by users. In addition, the smaller \( m \) is, the more vulnerable it is to TCP throughput estimation errors. Therefore, the value of \( m \) should be balanced between low switching frequency and high bandwidth utilization. In order to avoid unreasonable rate switching caused by a data exception, the later experiments use equations (8) and (9) to calculate the average value of the \( m \) obtained three times as the final switching delay factor.

Figure 2 shows the influence of different switching delay factor \( m \) on switching frequency and bandwidth utilization ratio. Obviously, the smaller \( m \) is, the greater the
frequency of rate switching is, the more sensitive the EDARS algorithm is to bandwidth changes, and the higher the corresponding bandwidth utilization is.

Figure 1. The influence of switching delay factor $m$ on algorithm.

Figure 2. The influence of switching delay factor $m$ on switching frequency and bandwidth utilization ratio.

3.2.2 Performance comparison of algorithm

Figure 3 shows the TCP throughput and selected video rates of the traditional greedy adaptive algorithm and EDARS algorithm. When the time is between 250s-450s, TCP throughput of the traditional greedy adaptive algorithm is concentrated around 2.6Mbps, and the video rate is relatively stable. At other times, the video rate fluctuates with changes.
in TCP throughput. Especially during 450s-600s, the throughput is more dispersed, the video rate jitter is frequent, and even there are some direct jumps between lower video code rates and higher video code rates. The experimental results show that the greedy adaptive algorithm has a high fit degree to the video bit rate switching and the current network throughput changes, and is more sensitive to network bandwidth changes. Even if the bandwidth fluctuates slightly in the short term, it will trigger the rate selection algorithm to adjust the rate and select the video bit rate suitable for the current bandwidth conditions. This frequent switching of video rates is detrimental to the user's QoE, and frequent rate switching will not only generate a large amount of signaling flow, but will also aggravate the power consumption of terminal devices.

Compared with the traditional greedy adaptive algorithm, the TCP throughput of the EDARS algorithm is more consistent with the selected video rate, the switching frequency is greatly reduced, and there is little direct rate switching between the lower video bit rate and the higher video bit rate. The reason for this result is mainly determined by the strategy of algorithm design. The traditional greedy adaptive strategy is aimed at adapting to changes in TCP throughput, and its rate adjustment is based on the predicted value of TCP throughput. Once the predicted value of TCP throughput changes, the video rate needs to be adjusted accordingly. By dynamically setting the safety boundary DSM and response delay factor $m$, the EDARS algorithm only adjusts the rate when ensuring that the TCP throughput is indeed steadily increasing, and video rate adjustment does not change much.

![Figure 3](image3.jpg)

(a) The greedy adaptive algorithm  
(b) The EDARS algorithm

**Figure 3.** TCP throughput and selected video rates of two algorithms.

Figure 4 is the terminal energy consumption comparison of the EDARS algorithm and traditional greedy adaptive algorithm. From the experimental results, the EDARS algorithm improves the energy utilization efficiency and delays the battery power consumption rate, and its terminal energy utilization rate can be increased by up to 6-7%.

![Figure 4](image4.jpg)

**Figure 4.** The energy consumption comparison of the EDARS algorithm and the greedy adaptive algorithm.
From the above experimental results and performance analysis, it can be seen that the EDARS algorithm can reduce the frequency of video rate switching under the premise of ensuring smooth playback, so that the algorithm can maintain the video rate in a relatively stable state and then obtain considerable energy saving effect. It should be pointed out that the energy-saving benefit of the algorithm comes at the expense of some available bandwidth.

4 Conclusion

This paper discusses the energy saving of wireless port of mobile terminal under the environment of HTTP transmission protocol, and proposes a dynamic adaptive rate selection algorithm based on energy consumption awareness (EDARS). The algorithm considers the user's video experience and terminal energy consumption in a comprehensive manner, and can adaptively select the bit rate of the next target video block according to the terminal's cache status, TCP throughput history, and video bit rate history. Compared with the traditional greedy adaptive algorithm, the EDARS algorithm reduces the video rate switching frequency. Under the premise of ensuring the user QoE, the algorithm can slow down the terminal battery consumption as much as possible, and extend the terminal's battery life. The network simulation experiment results on NS3 illustrate the effectiveness of the EDARS algorithm.

References

1. M. Seufert, S. Egger, M. Slanina, T. Zinner, T. Hoßfeld, P. Tran-Gia, IEEE Commun. Sirv. Tutor. 17, 469-492 (2015).
2. K. Spiteri, R. Sitaraman, D. Sparacio, ACM T. Multim Compt. 15, 1-29 (2019).
3. E. Sillasto, L. Wang, J. Manner, IEEE/ACM Int. Conf. Green Comput. Commun. 9-16 (2010).
4. H. Han, J. Yu, H. Zhu, Y. Chen, J. Yang, G. Xue, Y Zhu, M Li. Proc. ACM Conf. Embedded Networked Sens. Syst. 1-15 (2013).
5. M. O. Khan, V. Dave, Y. C. Chen, O. Jensen, L. Qiu, A. Bhartia, S. Rallapalli, Proc. Int. Symp. Mobile Ad Hoc Networking Comput. 217-226 (2013).
6. G. B. Tian, Y. Liu. Int. Packet Video Workshop, 1-8 (2013).
7. S. Khan, D. Schroeder, A. E. Essaili, E. S. Steinbach, IEEE Wireless Commun. Networking Conf. 2354–2359 (2014).
8. M. A. Hoque, M. Siekkinen, J. K. Nurminen, M. Aalto, S. Tarkoma, Pervasive Mob. Compt. 16, 96-114 (2015).
9. Song J, Cui Y, Li Z, Y. Bao, L. Zhang, Y. Zhang, Int. Conf. Comput. Commun. Networks, 1-9 (2016).
10. Z. Li, A. C. Begen, L. Gaehm, Y. Shan, B. Osler, D. Oran, Proc. 5th ACM Multim. Syst. Conf. 248-258 (2014).