A User Mobility-Based Rate Adaptation Approach for Dynamic HTTP Streaming

Renkang Chen\textsuperscript{1,\textast}, Zeping Li\textsuperscript{2,\textast\textdagger}, Huawei Yang\textsuperscript{3,\dagger}, Yangshang Xue\textsuperscript{4,\dagger}

\textsuperscript{1,\textsuperscript{2,\textsuperscript{4}}}College of Computer Science and Technology, Guizhou University, Guiyang, China
\textsuperscript{3}College of Big Data Application and Economics, Guizhou University of Finance and Economics, Guiyang, China
\textsuperscript{\textast}renkangchen@gmail.com, \textsuperscript{\dagger}yang-hw2005@163.com
\textsuperscript{\textast\textdagger} Corresponding author: zpli1@gzu.edu.cn
\textsuperscript{\dagger}xueyangshang@163.com

Abstract—It is common for users to use smartphones to watch streaming videos when they are walking, taking a bus, or taking a subway. However, the current streaming rate adaptation algorithms are mostly based on the available network throughput or the player buffer occupancy, and the user mobility state is less considered. In order to improve the user experience quality under the mobile streaming media, we propose a user mobility-based rate adaptation approach, which classifies the user mobility state using the random forests algorithm and adopts different rate decision under different mobility states. The experimental results show that compared with popular approaches, our approach reduces the re-buffer events by 24\% on average, and reduces the data traffic consumption by 19\% on average.

1. INTRODUCTION

According to [1], in 2017, mobile video traffic of global Internet users accounted for more than 60\% of total mobile traffic, and the annual growth is expected to reach 54\%, and mobile video traffic will account for 78\% of total traffic by 2021. Watching streaming videos on smartphones has become a choice for most users to pass the boring time when they are walking, taking a bus, or taking a subway. According to the characteristics of human vision, the user's sensitivity to video quality vary from different mobility states. If the client selects high-quality video when the user is less sensitive to the video quality, it will cause unnecessary bandwidth waste and traffic consumption. Meanwhile, the network conditions are unstable in mobile streaming media, so, problems such as re-buffer, also referred as stall problem, and frequent bitrate switching are more likely to occur. How to ensure the quality of user experience (QoE) in different mobile states is still a challenging problem.

In this paper, we propose a user Mobility-Based Rate Adaptation algorithm for Dynamic HTTP Streaming (MBRA). MBRA uses the random forests algorithm to classify the user's mobility state, and adopts different bitrate decisions according to the mobility state. Like most adaptation algorithms, the MBRA algorithm we proposed is designed based on DASH (Dynamic Adaptive Streaming over HTTP), which is an international standard for streaming media transmission [2]. We implemented our mobility state classification algorithm on the Scikit-Learn platform and deployed it on the Flask server. The
MBRA algorithm was implemented on Google’s open source player Exoplayer. We have carried out various experiments, and our experimental results show that compared with existing methods, MBRA can effectively reduce re-buffer event and data traffic consumption.

The remainder of this paper is organized as follows: in Section 2, we first explain our method of data collection for user mobility classification, then describe the user mobility state recognition algorithm based on random forests, last, introduce the bitrate adaptation algorithm based on user mobility state. Section 3 reports the experiments we conducted. Finally, the last section summarizes this paper and points out future work.

2. Proposed Methods

2.1. Data collection

We use the Android sensor framework to access and obtain the data of two sensors, the accelerometer and the gyroscope. Specifically, using SensorManager and implement the interface exposed by SensorEventListener. Accelerometer is a sensor that can measure acceleration. It has three axes, namely x-axis, y-axis, and z-axis, which can measure acceleration values in three directions respectively. Note that the sensor contains the value of acceleration of gravity. For the convenience of calculation, we use a high-pass filter to filter out the influence of gravity. Gyroscope is a device used to sense and maintain direction. It also has three axes, namely the x axis, the y axis, and the z axis, which can measure the angular velocity values in the three directions respectively.

According to the basic theorem of space vectors, let O as the origin point in the space rectangular coordinate system, and module of any vector $\overline{OP}$ in the space can be expressed as

$$|\overline{OP}| = \sqrt{x^2 + y^2 + z^2}$$  (1)

where x, y, z are the vector $\overline{OP}$ projection on the X axis, Y axis, and Z axis. According to this theorem, we can use the data on the three direction axes of the sensor into a spatial signal vector and find its module. Define the acceleration value $A$ and angular velocity $W$ of the three-axis synthesis as:

$$A = \sqrt{a_x^2 + a_y^2 + a_z^2}$$  (2)

$$W = \sqrt{\omega_x^2 + \omega_y^2 + \omega_z^2}$$  (3)

We designed an Android application to collect sensor data in different mobility states, and store it as a log file in CSV format, named as the corresponding mobility state. We define the following 4 types of user mobility states, which are, still (STILL), walking (WALK), taking a bus (BUS), and taking a subway (SUBWAY). Figure 1 and Figure 2 show the change curves of A and W under different states.

![Fig. 1. A change curve under different states.](image-url)
According to Figure 1 and Figure 2 above, we can conclude that the STILL state (0~200s in figures) is most stable with small fluctuations, whether in A or W. On the contrary the WALK state (200~400s in figures) is the most unstable, which has frequent fluctuations and large amplitude compare other states. The BUS (400~600s in figures) and SUBWAY (600~800s in figures) states are relatively stable, but there are still fluctuations in a certain range, and notice that the fluctuations of A and W in the BUS state are more frequently than those in the SUBWAY state.

2.2. Mobility state classification algorithm based on Random Forests

Random forests (RF) is an ensemble learning strategy composed of multiple tree-structured classifiers \( \{ h(x,\Theta_k) , k=1... \} \), where \( \{ \Theta_k \} \) are independent identically distributed random vectors and each tree casts a unit vote for the most popular class at input \( x \) [3]. The mobility state classification algorithm based on RF is described in Algorithm 1.

**Algorithm 1. RF for Mobility State Classification**

| Precondition: | train set \( S \), features \( F \), and number of decision trees \( N \) |
|---------------|--------------------------------------------------------------------------|
| function      | RandomForest\((S,F)\) | H ← 0 |
| for \( i \in 1,...,N \) do | \( S_i \leftarrow \text{Bootstrap}(S) \) | \( h_i \leftarrow \text{DecisionTreeBuild}(S_i,F) \) | \( H \leftarrow H \cup \{ h_i \} \) |
| end for | return \( H \) | end function |

function DecisionTreeBuild\((S,F)\)

while not leaf node do

// if \( f \ll F \)

\( f \leftarrow \text{a small random subset of } F \)

Split on min Gini feature in \( f \)

end while

return decision tree

end function

The working principle of the algorithm is as follows: for each tree in the forest, we randomly select \( n \) samples from the original sample data set \( S \) with replacement to generate a new training sample set \( S_i \).
Each sample has $F$ attributes, when each node of the tree needs to be split, we randomly select $f$ attributes from these $F$ attributes, satisfying the condition $f \ll F$. Here we use $\sqrt{F}$, which is a popular choice. Then, from these $f$ attributes, a certain strategy is adopted to select one attribute as the split attribute of the node. The strategy we use here is Gini impurity, which can be arrived using equation (4). And generating n trees to form the random forest. Finally, according to the Bagging voting principle, the one with the most classification results of the tree is classified as the final decision result of the random forest.

$$I_c(n) = 1 - \sum_{i=1}^{C} (p_i)^2$$ (4)

We collected 5 minutes of data under each state, and the work frequency is 10Hz. By analysing the characteristics of the two sensor data in different mobility states, we select 8 features: the number of three-axis synthetic acceleration peaks and valleys($\text{num-pv}$), the maximum peak-valley difference($\text{max-pvd}$), the minimum peak-valley difference($\text{min-pvd}$), the maximum three-axis synthesized acceleration($\text{max-acc}$), the minimum three-axis synthesized acceleration($\text{min-acc}$), and the average value of three-axis synthesized acceleration($\text{avg-acc}$), the maximum and minimum slope of the three-axis synthetic angular velocity change($\text{max-avs,min-avs}$) within two seconds (the window size is 2 seconds). That is $F: \{\text{num-pv}, \text{max-pvd}, \text{min-pvd}, \text{max-acc}, \text{min-acc}, \text{avg-acc}, \text{max-avs}, \text{min-avs}\}$.

We use Python to process the original data, and get the data containing each feature value and label column, $S$. When training the model, we used 70% of the total sample as the training set, and the remaining 30% as the test set. The predictive performance of random forests has been proved to converge as more trees are added [3]. Therefore, setting more trees is no risk, and we choose $N$ as 100. Regarding the number of attributes to be considered for each node, because random forests perform well even with a small sample of attributes, the popular choice is the square root or logarithm of the number of attributes. As mentioned earlier, we chose the former one. Finally, we implemented it on the Scikit-Learn platform.

In the classification scenario of this paper, accuracy is not an appropriate performance evaluation indicator, so instead, we use the Receiver Operating Characteristic Area Under the Curve (ROC AUC), and we perform category prediction and prediction Probability to calculate ROC AUC (using metrics in Scikit-Learn), the result shows that the final test ROC AUC of our algorithm is 0.86, which means that our model achieves a good prediction result, and we have also drawn the ROC curve to evaluate the model (Figure 3).

![ROC Curve](image.png)

**Fig. 3.** ROC Curves.

2.3. User mobility-based rate adaptation algorithm

At present, mainstream rate adaptation strategies can be roughly divided into the following two categories: throughput-based adaptation algorithms and buffer-based adaptive algorithms. The essence of the throughput-based adaptation algorithm is to select the highest possible bitrate according to the currently estimated available network throughput [4]. The buffer-based adaptation algorithm is different from the throughput-based adaptation algorithm. This type of algorithm only uses occupancy of the player buffer as a feedback signal, and keeps the buffer occupancy at a required level [5]. Based on the above two rate adaptation algorithms and the study of mobility state classification based on RF, we propose a mobile streaming media bitrate adaptation algorithm, MBRA. Algorithm 2 shows the workflow of MBRA.
Algorithm 2. MBRA

Precondition: throughput prediction $Rate_{avail}$, player buffer occupancy features $BF_{now}$, sensor data feature $F_{now}$

1. function MBRA($Rate_{avail}$, $BF_{now}$, $F_{now}$)
2. 
3. if player is startup status then
4.     $B_{next} \leftarrow \max\{B_i; B_i \leq Rate_{avail}\}$
5.     return $B_{next}$
6. else
7.     $B_{now} \leftarrow B_{next}$
8.     $MobilityState \leftarrow MobilityState(F_{now})$
9.     if $MobilityState$ is WALK then
10.        $B_{next} \leftarrow \max\{B_i; B_i \leq Rate_{avail}\} \times 0.6$
11.     end if
12.     if $MobilityState$ is BUS then
13.        $B_{next} \leftarrow \max\{B_i; B_i \leq Rate_{avail}\} \times 0.8$
14.     end if
15.     if $MobilityState$ is SUBWAY then
16.        $B_{next} \leftarrow \max\{B_i; B_i \leq Rate_{avail}\} \times 0.9$
17.     end if
18.     if $MobilityState$ is STILL then
19.        $B_{next} \leftarrow \max\{B_i; B_i \leq Rate_{avail}\} \times 1.0$
20.     end if
21.     if $B_{next} > B_{now}$ and $BF_{now} < minBF$ then
22.        $B_{next} \leftarrow B_{now}$
23.     end if
24.     if $B_{next} > B_{now}$ and $BF_{now} > maxBF$ then
25.        $B_{next} \leftarrow B_{now}$
26.     end if
27. end else
28. return $B_{next}$
29. end function

As shown in Algorithm 2, given the current bandwidth prediction value $Rate_{avail}$, which can be obtained using some bandwidth prediction methods. We use an existing method [4] here. The current buffer occupancy $BF_{now}$ of the player is also needed. And the sensor data feature value $F_{now}$ used for mobility state classification. The algorithm will select the bitrate $B_{next}$ of the next video segment from the list of available bitrates.

If the player is in the startup stage, then select the largest one that meets the current network throughput conditions from the available bitrates as the initial bitrate. If the player is already in the playback stage, we first use the trained model to identify the user mobility state based on the acquired accelerometer and gyroscope sensor data. According to our user research, when walking, users need to pay more attention to the surroundings. Therefore, they are less sensitive to video quality. So a coefficient less than 1 is adopted to appropriately reduce the video bitrate. When taking a bus or subway, the shaking of the vehicle will cause the user to be less sensitive to the video quality too. At the same time, the rapid movement of the vehicle will cause the network throughput to be unstable. Usually, we found that the subway is running more smoothly than bus, so we adopt different coefficients for them. When the user is in a still state, we choose the highest video bitrate that is currently suitable for the current network throughput, as user is particularly concerned about the video quality at this time.
Note that in two cases, the algorithm does not perform bitrate switching. The first case is that the next bitrate selected is greater than the current one, but the buffer length is less than the minimum value of the buffer for switching to a higher bitrate (the default $minBF$ is 10s). The other case is that the next bitrate selected is less than the current bitrate, while the length of the buffer is greater than the maximum value of the buffer switched to a lower bitrate (the default $maxBF$ is 25s). The purpose of this is to avoid frequent bitrate switching as much as possible and reduce re-buffer events.

3. EXPERIMENTS AND RESULTS

3.1. Experimental Setup
The experimental platform uses Nginx server as the streaming media server which runs on Ubuntu 20.04. The model for user mobility status classification is deployed on the Flask server, and an API service is provided for the client to call. We implemented the MBRA algorithm on Google's open source media player Exoplayer (v2.11.0). The experimental video we chose is the open source video "Big Buck Bunny" [6]. The original video dataset has 20 different bitrate levels. In the experiment, 10 different bitrate levels are selected, and each bitrate level has 150 segments, each segment is 4 seconds in length.

3.2. Performance Evaluation
Although different researchers may use different evaluation indicators when evaluating the performance of rate adaptation algorithms, generally speaking, several key elements such as initial delay, re-buffer events, smoothness of rate switching, and average video quality are usually considered.

Let $\Omega$ be all the optional bitrates, the player can select the video with the bitrate $B_n \in \Omega$ to play when playing the nth video block. Let $L(B_n)$ be the size of the n-th video block with a code rate of $B_n$, the download time of the n-th video block can be expressed as $\frac{L(B_n)}{C_n}$, where $C_n$ represents the download rate when downloading the n-th video block. Let $q(B_n)$ denote the video quality when the bitrate is $B_n$. Let the total video block be $N$.

**Initial Delay:** the initial buffering time of the player. Generally, the player will buffer a certain length of video (less than the maximum buffer length) in the initial stage before playing it to avoid re-buffering events in the initial stage. Suppose the initial buffer duration set by the player is $L$ and the initial bitrate is $B_{init}$, then the initial delay is

$$ID = \frac{(L \times B_{init})}{C_{init}}$$

(5)

**Re-buffer Event:** For each video block $n$, after the start of playback, if the download time of the video block is higher than the playback buffer length (such as $BF_n$), a re-buffer event will occur. The total number of re-buffer event is

$$RB = \sum_{n=1}^{N} \mathrm{sig}\left(\frac{L(B_n)}{C_n} - BF_n\right)$$

(6)

where

$$\mathrm{sig}(x) = \begin{cases} 1, & x > 0 \\ 0, & \text{otherwise} \end{cases}$$

(7)

**Average Rate Switching:** the size of the video bitrate level change when switching from one video block to another video block

$$S = \frac{1}{N-1} \sum_{n=1}^{N-1} |q(B_{n+1}) - q(B_n)|$$

(8)

**Average Video Quality:** Average block video quality of all video blocks

$$Q = \frac{1}{N} \sum_{n=1}^{N} q(B_n)$$

(9)

In addition to the above 4 elements, we also considered the traffic data usage caused by video playback during the experiment comparison, which can be obtained via the system's traffic management software.
3.3. Experimental Results
We compared the performance of the RB algorithm [4], the BBA-0 algorithm [5] and the MBRA algorithm in four state of STILL, WALK, BUS, and SUBWAY from initial delay, re-buffer, switching times, average quality and data usage. In order to facilitate comparison, we use the ratio of the value to the maximum value of the item, instead of using the original data.

(a) STILL State Performance Comparison. 
(b) WALK State Performance Comparison. 
(c) BUS State Performance Comparison. 
(d) SUBWAY State Performance Comparison. 

Fig. 4. Performance comparison of three algorithms under different mobility states.

It can be seen from Figure 4 that BBA-0 has a better performance on initial delay and average video quality as it initially selects a lower bitrate and carry out a more aggressive bitrate selection strategy during playback, however, frequent rate switching results in poor rate switching smoothness, and aggressive rate selection strategies often lead to re-buffer events. As RB performs bitrate switching only based on the current network throughput, in BUS and SUBWAY states, compared with MBRA, the bitrate switching are more frequent and there are more re-buffer events. MBRA appropriately reduces the bitrate level according to different mobile states. Compared with RB and BBA-0, the re-buffer events are reduced by 24% on average. At the same time, the data traffic consumption is reduced by 19% on average. Although the average video quality of MBRA is generally lower than that of RB and BBA-0, our user research shows that users cannot clearly distinguish the videos transmitted by the RB and MBRA algorithms.

4. CONCLUSIONS 
In this paper, we studied the video streaming under different users mobility state on the smartphone. We propose an algorithm for user mobility state classification based on RF, and apply it to the mobile streaming media bitrate adaptation decision, and propose a bitrate adaptation algorithm based on user mobility state. Various experiments were carried out, and the experimental results show that our approach can effectively reduce video re-buffer events and unnecessary data traffic consumption. As future work, we plan to take more user mobility states into consideration and optimize the performance of user mobility state classification model.
ACKNOWLEDGMENT
This work was financially supported by the National Natural Science Foundation of China (61462014) and the Natural Science Foundation of Guizhou Science & Technology Department (S2017JP00401185093).

REFERENCES
[1] Cisco.(2018)Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2018-2023 White Paper.https://www.cisco.com/c/en/us/solutions/collateral/service-provider/.
[2] Stockhammer, Thomas. "Dynamic adaptive streaming over HTTP-- standards and design principles." In Proceedings of the second annual ACM conference on Multimedia systems, pp. 133-144. 2011.
[3] Breiman, Leo. "Random forests." Machine learning 45, no. 1 (2001): 5-32.
[4] Liu, Chenghao, Imed Bouazizi, and Moncef Gabbouj. "Rate adaptation for adaptive HTTP streaming." In Proceedings of the second annual ACM conference on Multimedia systems, pp. 169-174. 2011.
[5] Huang, Te-Yuan, Ramesh Johari, Nick McKeown, Matthew Trunnell, and Mark Watson. "A buffer-based approach to rate adaptation: Evidence from a large video streaming service." In Proceedings of the 2014 ACM conference on SIGCOMM, pp. 187-198. 2014.
[6] Lederer, Stefan, Christopher Müller, and Christian Timmerer. "Dynamic adaptive streaming over HTTP dataset." In Proceedings of the 3rd multimedia systems conference, pp. 89-94. 2012.