Researches on TCP Throughput Prediction and Adaptive Bitrate Control in Mobile Network

モバイルネットワークにおける TCP スループット予測と適応レート制御に関する研究

February 2019

Waseda University

Graduate School of Fundamental Science and Engineering
Department of Computer Science and Communications Engineering
Research on Image Information

Bo WEI

魏　博

Abstract

The number of mobile applications is increasing rapidly, which has resulted in massive mobile network traffic. In order to guarantee high quality of service/experience (QoS/QoE), such as providing high quality video without playback freezing, it is important to control bitrate adaptively through predicting future throughput distribution with high accuracy. An accurate throughput prediction can contribute a lot to improving QoS/QoE. In order to solve this problem, transmission control protocol (TCP) throughput prediction methods are proposed in this thesis.

First, TCP throughput prediction methods using statistics and machine learning are proposed. A novel approach is proposed which utilizes hidden Markov model (HMM) and Gaussian mixture model (GMM) to deal with historical time series of throughput and judge fluctuation factor with total variance when predicting future throughput. Besides, an advanced model which is history-based throughput prediction method utilizing time series analysis and machine learning techniques for mobile network communication is proposed. This method is called the hybrid prediction with the autoregressive model and hidden Markov model (HOAH). Different from existing methods, HOAH uses support vector machine (SVM) to classify the throughput transition into two classes and predicts the TCP throughput by switching between the autoregressive model (AR Model) and the Gaussian mixture model-hidden Markov model (GMM-HMM).

Second, TCP throughput prediction method using neural networks for mobile network is proposed which is named as TRUST (**Th**r**ou**ghput prediction based on L**ST**M). TRUST has two stages: user movement pattern identification and throughput prediction. In the prediction stage, the long short-term memory (LSTM) model is employed for TCP throughput prediction. TRUST takes all the communication quality factors, sensor data and scenario information into consideration. Field experiments are conducted to evaluate TRUST in various scenarios. The results indicate that TRUST can predict future throughput with higher accuracy than the conventional methods, which decreases the throughput prediction error by maximum 44% under the moving bus scenario.

Third, a trace-based emulation environment is established and adaptive bitrate control method using throughput prediction is proposed. Dynamic adaptive video streaming over HTTP (DASH) is widely studied and adopted in modern video players for ensuring user quality of experience (QoE) since QoE directly affects the revenue. Basically, the algorithms need to be tested with large-scale deployment. However, it is not always possible in academic research. We established a video transmission system with DASH which enables replicable trace-based emulations. The emulation enables us to compare different methods under the artificially same condition with limited experiments. We also proposed a new adaptive bitrate (ABR) control method which incorporating both prediction and buffer occupancy information named decision map method (DMM). DMM creates both aggressive and conservative mechanisms to handle different network conditions. The emulation results demonstrate that the DMM can achieve better performance in QoE than conventional methods, showing the efficiency of the DMM algorithm.

In conclusion, this thesis proposes TCP throughput prediction methods, and experiments are conducted to evaluate the performances of these proposals for adaptive bitrate control. The results conclude that the proposals can predict throughput accurately and are effective for adaptive bitrate control. Meanwhile, we propose a new ABR method incorporating both prediction and buffer occupancy information which improves the performance further.

Acknowledgement

This thesis has been written with the help and support of people around me. I would like to express my gratitude to them.

First and foremost, I would like to thank my supervisor, Professor Jiro Katto, for his continuous guidance and enormous help in my research work at Waseda University. He has always provided me useful suggestions and helped me to improve my research. Professor Katto talks with me about my research each week, which broadens my knowledge and provide me very useful ideas. Moreover, Professor Jiro Katto taught me how to become a good researcher.

I would like to offer my special thanks to Professor Fumiaki Maehara, I took his course and discussed with him about research problems I met in the work which benefit me a lot. He has provided practical and valuable advices on my study. I would particularly like to thank Dr. Kenji Kanai for his constructive suggestions and constant encouragement.

I would like to thank Mr. Takeuchi and other members in Katto Laboratory, Waseda University. They work together with me and only through the team work, I can achieve good research results.

Finally, I sincerely thank my family and friends for their support, encouragement and love.

Bo Wei

25th February 2019

Table of Contents

1 Introduction

1.1 Background

It is important to ensure high Quality of Service (QoS) and Quality of experience (QoE) for users in wireless networks. As shown in Figure 1.1, the number of mobile applications is increasing rapidly, which has resulted in massive mobile network traffic. In order to guarantee high QoS/QoE (such as support higher quality video without playback freezing), it is important to control bitrate adaptively by predicting future throughput distribution with high accuracy. An accurate throughput prediction can contribute a lot to improving QoS/QoE as reported in [1], [2].

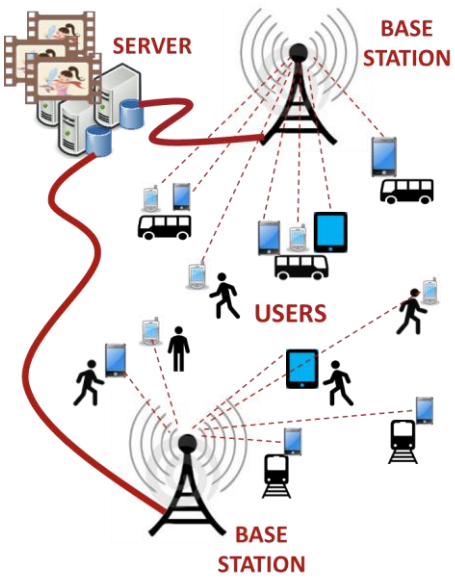


Figure 1.1 The rapid increasing of mobile network traffic.

Although there have been many researches for throughput prediction, it is still a challenging topic especially in mobile networks due to the fact that few methods have taken user moving mode and the fluctuation characteristic of session into consideration. For example, current methods adopt one model to predict throughput for the whole session which is different to adaptively fit to the throughput variation during one session. Therefore, existing prediction methods are not fit for moving user scenarios,

which may lead to large prediction error. Moreover, no effective prediction model is universally accepted by all researchers for adaptive bitrate control. To solve the problem of lacking an accurate measurement of network parameters (e.g. received signal strength and future number of wireless users) for many throughput prediction methods, we propose throughput prediction models to predict the future throughput. The application of predicted throughput for adaptive bitrate control can be categorized into two kinds: 1) Calculating the amount of possible future data to be delivered and combining buffer control for adaptive bitrate control, such as in [1], [3]. 2) Controlling the bitrate directly with the predicted throughput of future time [4], [5].

The accuracy of throughput prediction is critical as depicted in [6]. Zou et al. [2] held that the combination of throughput prediction and buffer occupancy or stability function could perform better than the existing methods. It reduces the gap to 4%, which indicates that cellular operators and content providers improve their video QoE largely by the prediction of available bandwidth and shares it via application programming interface (API). Yin et al. in [1], [7] showed that QoE decrease with the increase of throughput prediction error when controlling the bitrate of video streaming. Moreover, even the 0.1 prediction error difference affects the performance of bitrate control and QoS/QoE for many applications such as Internet Protocol Television (IPTV). Therefore, transmission control protocol (TCP) throughput must be predicted accurately.

In order to achieve this goal, we first propose TCP throughput prediction methods using statistics and machine learning. A novel approach utilizes hidden Markov model (HMM) with Gaussian mixture model (GMM) to deal with historical time series of throughput and judge fluctuation factor with total variance when predicting future throughput. History data are separated into various groups according to the value range as observations. Then data are clustered into different classes with multi-component Gaussian mixture model and defined as hidden states. After obtaining state sequence and observation sequence, we train data with hidden Markov model and calculate the most probable state transition path. Then we judge fluctuation (stationary or non-stationary) of current time according to the fluctuation factor calculated by former state transition path. Finally, we predict throughput of following time with corresponding method: linear regression for stationary process and locally weighted linear regression for non-stationary process. Besides this, a history-based throughput prediction method that utilizes time series analysis and machine learning techniques for mobile network communication is proposed. This method is called the hybrid prediction with the autoregressive model and hidden Markov model (HOAH). Different from existing

methods, HOAH uses support vector machine (SVM) to classify the throughput transition into two classes, and predicts the TCP throughput by switching between the autoregressive model (AR Model) and the Gaussian mixture model-hidden Markov model (GMM-HMM). We conduct field experiments to evaluate the proposed method in seven different scenarios. The results show that HOAH can predict future throughput effectively and decreases the prediction error by a maximum of 55.95% compared with other methods.

Second, we propose a TCP throughput prediction using neural networks which is named TRUST. This method has two stages: user movement pattern identification and throughput prediction. In the prediction stage, the long short-term memory (LSTM) model is employed for TCP throughput prediction. TRUST takes all the communication quality factors, sensor data and scenario information into consideration. Field experiments are conducted to evaluate TRUST in various scenarios. The results indicate that TRUST can predict future throughput with higher accuracy than the conventional methods, which decreases the throughput prediction error by maximum 44% under the moving bus scenario.

Third, in order to validate the prediction methods and provide reliable video streaming with efficient wireless resource, adaptive bitrate control using throughput prediction is explored. Meanwhile, a trace-based emulation system is established to exam different methods using limited experiments. Dynamic adaptive video streaming over HTTP (DASH) is widely studied and adopted in modern video players for ensuring user QoE since QoE directly affects the revenue. In DASH, adaptive bitrate control is a key part for achieving high quality of service and QoE when transmitting video streaming. The ultimate goal of adaptive bitrate control is to maximize video bitrate while minimizing rebuffering events and duration. However, this task is non-trivial since the network condition is not always stable. The choice of higher bitrate may cause frequent video freezing which annoying the user while choosing lower bitrate may give worse experience. Therefore, throughput prediction plays an important role in helping select the proper bitrate of video dynamically. We implement the proposal into DASH-JS [8] and evaluate the performance of our approach. The DASH-JS structure is modified and extended for flexible purposes.

Basically, the algorithms need to be tested with large-scale deployment. However, it is not always possible in academic research. In this paper, we establish a replicable trace-based emulation environment and we study the influence of different prediction methods on adaptive video streaming. The emulation enables us to compare different methods under the artificially same condition, with limited experiment. The

quality metrics such as average bitrate, the number of rebuffering events, the duration of rebuffering, etc. are examined. The results indicate that a good prediction can provide better user QoE and that predicted throughput is effective for bitrate selection, thus provide highly-reliable mobile video streaming with high bitrate and avoid rebuffering event.

In order to further improve the QoE, we propose a new adaptive bitrate (ABR) control method named decision map method (DMM). In this method, the buffer occupancy information is considered simultaneously with the prediction result. DMM creates both aggressive and conservative mechanisms to handle different network condition. The emulation results demonstrate that the DMM can achieve better performance in QoE than conventional methods, showing the efficiency of the DMM algorithm.

1.2 Thesis organization

This thesis consists of six chapters as follows.

Chapter 1 introduces the background and motivation of this research. As throughput prediction plays a significant role for adaptive bitrate in mobile network, we focus on this topic and construct several prediction models.

In Chapter 2, we introduce related works of TCP throughput prediction methodology for wireless network, user movement pattern recognition, adaptive bitrate control, MPEG-DASH, and QoS/QoE optimization.

Chapter 3 depicts the TCP throughput prediction methods using statistics and machine learning. A novel approach is proposed utilizing HMM with GMM to deal with historical time series of throughput and judge fluctuation factor with total variance when predicting future throughput. Based on the model, a history-based throughput prediction method that utilizes time series analysis and machine learning techniques for mobile network communication is proposed. This method is called the hybrid prediction with the autoregressive model and hidden Markov model (HOAH).

Chapter 4 presents the TCP throughput prediction using neural network. The method uses measured throughput, received signal strength indicator (RSSI), Cell ID and other parameters via neural network to predict future throughput. Results show the method can predict throughput effectively.

Chapter 5 proposes a trace-based emulation system and an adaptive bitrate control method. A replicable trace-based emulation environment is established and the influence of different prediction methods on adaptive video streaming is studied. The

results indicate that a good prediction can provide better user QoE. In order to further improve the QoE, a new ABR method is proposed in which the buffer occupancy is considered simultaneously. The emulation results demonstrate that DMM can achieve better performance in QoE than conventional methods.

Chapter 6 gives overall conclusions and future work.

Figure 1.2 shows the relationship of the research described in Chapter 3, 4 and 5. As shown in the figure, in order to predict future TCP throughput, we construct throughput prediction models in Chapter 3 and 4. Based on the throughput prediction methodologies, the adaptive bitrate technique is put forward to MPEG-DASH to ensure high QoS/QoE.

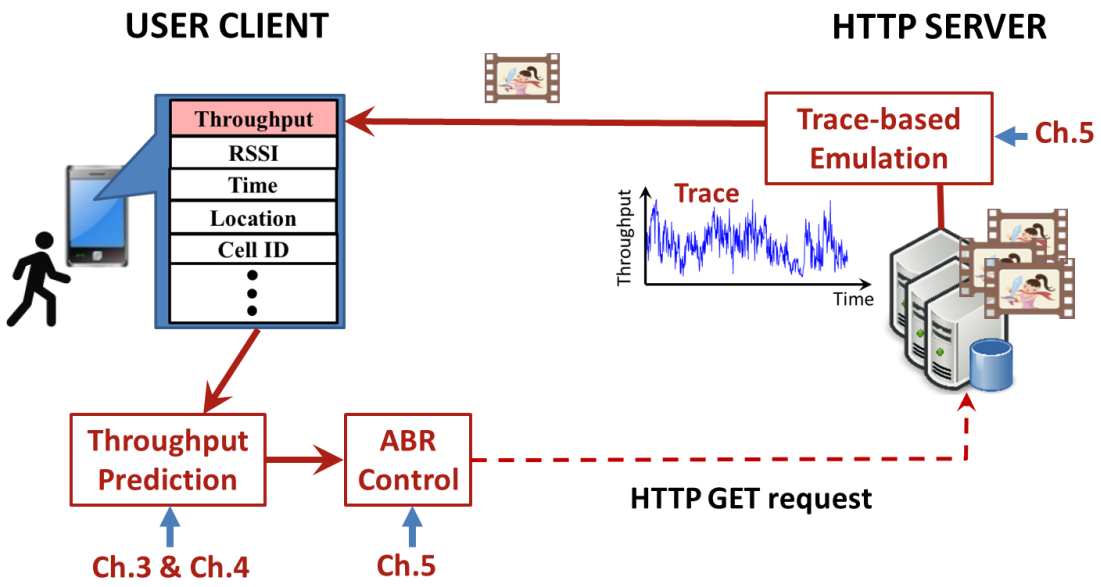


Figure 1.2 Structure and relationship of the research chapter in this thesis.

2 Related work

2.1 Wireless communication

Wireless communication mainly uses wireless techniques such as radio wave, magnetic, light and sound to transmit information. It brought large convenience for human life. One of the important applications of wireless communication is constructing connection for wireless network.

Wireless network is the kind of computer network that uses wireless connection to transmit information between network nodes. There are many types of wireless network, such as wireless LAN, wireless WAN, wireless ad hoc network and mobile network.

There are many technologies for wireless communication. Some of them are introduced in the following.

High speed downlink packet access (HSDPA) is an enhanced version of 3rd mobile communications protocol in the high-speed packet access family. It allows networks based on the universal mobile telecommunications system to have higher data speed and capacity. HSDPA decreases latency and the round trip time for applications. In this thesis, we use the public HSDPA dataset as part of the data to construct our proposal.

Long-term evolution (LTE) is a standard developed by 3rd Generation Partnership Project for high speed wireless communication in mobile network. LTE can realize higher transmission speed, lower delay and higher bandwidth efficiency. Moreover, LTE supports various carrier bandwidths from 1.4 MHz to 20 MHz and supports both frequency division duplexing and time-division duplexing.

Wi-Fi is a technique to realize the connection of terminals such as PC and mobile phone to the internet base on IEEE 802.11 in wireless local area network. The coverage area of Wi-Fi can be 20 meters indoors and larger ranges outdoors.

There are many important characteristics in wireless communication. Throughput is the actual data being delivered in a unit time of the transmission link. Transmission control protocol (TCP) is one of the main internet protocol. It provides a reliable and robust communication service with the internet protocol (IP) together. As the data we collected or utilized are mainly TCP throughput, we take these data as the

main objective for our methods. Received signal strength indicator (RSSI) indicates the power of received radio signal, which represents the relative signal strength. Cell ID is the unique number of the base station used to deliver the signal in the mobile network.

2.2 User movement pattern recognition

User movement pattern recognition can contribute to the progress of many researches. For example, it can provide the information of the situation a client is in, thus can deliver the information of environment change to the server side which the client is communicate with via the internet. It also has significant meaning for the topic of human trace prediction, geography information prediction and traffic management of urban transportation.

Many researches [9-15] have been conducted for the movement pattern recognition of user. In [9], the author uses global positioning system (GPS) measurements to identify the movement pattern. While based on [10], [14], [15], the data collection of GPS information is not available in some areas such as in the subway station which is usually located underground.

Thus the authors in [10], [11], [14] propose methods to utilize other sensor data instead of GPS information. User movement patterns include transportation mode and human activity. Transportation modes contain walk, travel by bus and etc. [11-16]. The patterns of human activity consist of standing, sitting, and etc. [17-21]. In [12], different machine learning methods, such as k-NN, SVM, RF and decision tree, are adopted to identify transportation modes using the accelerometer and gyro sensors. In [14], different statistical characteristics including mean, variance, and standard deviation in frequency domain are utilized to identify various transportation modes, such as walk, running, and traveling by bike and bus. The k-NN, SVM, and RF are adopted. The accelerometer sensors deployed on the mobile phone and wearable equipment are applied to identify human activity in [17-20]. Deep learning techniques such as CNN are used to identify the human activity [20], [21].

2.3 Throughput prediction

With the increase of mobile applications, massive mobile network traffic has been emerging. An accurate throughput prediction can contribute a lot to improving QoS/QoE as reported in [1], [2]. Moreover, throughput prediction is an important part for anticipatory network based on [22]. Throughput prediction is challenging especially

in mobile networks due to the fluctuation characteristic of session [23]. It plays an essential role in adaptive bitrate control [24-26]. The performance of bitrate control and QoS/QoE will be influenced by even 0.1 prediction error in many applications such as IPTV. Therefore, the accurate TCP throughput prediction is significant [27], [28].

According to [29], the existing TCP throughput prediction methods can be divided into two types: formula-based approach and history-based approach. Formula-based approaches employ mathematical functions to calculate the future throughput with other observable network parameters, such as round-trip time (RTT), packet loss rate, and TCP window size. For example, Mathis et al. [30] demonstrated a methodology to predict the bandwidth for TCP implementations with congestion avoidance algorithm in various situations. The model is effective in predicting shared bandwidth, but the assumption is strict and does not consider the timeout-driven behavior. Padhye et al. [31] proposed a model to compute the TCP flow throughput as a function of RTT, packet loss rate, TCP congestion window size, and TCP retransmission timeout. Although the model considers the TCP timeouts, this method is sensitive to RTT fluctuation, which sometimes leads to large prediction errors. Floyd et al. [32] extended Mathis's work and proposed an equation-based congestion control for unicast applications. Experiments verified that the method could provide better performance over a wide range of timescales for unreliable applications. However, this approach is not robust against network parameter estimation errors.

On the other hand, history-based approaches utilize methodologies such as time-series analysis to predict future TCP throughput. In this approach, the observed throughput data are treated as time-series data, and future throughput transitions are predicted by various methodologies. For example, Vazhkudai et al. [33] mentioned several predictors, such as average value, median value, and autoregressive integrated moving average (ARIMA). He et al. [29] also mentioned similar linear predictors, such as moving average (MA), exponential weighted moving average (EWMA), and Holt Winters (HW), which has proved that history-based approaches could achieve higher accuracy compared with formula-based approaches. Swany et al. [34] proposed a novel method to predict the throughput with cumulative distribution functions (CDFs) of history time series. Experiment results concluded that the method could predict future throughput with acceptable prediction errors. Yoshida et al. [25] raised a stochastic method combining the stationary and non-stationary Gaussian models to predict the possible value range. The method performs remarkably well, but does not consider the user moving scenarios.

All of the above-mentioned prediction methods fail to consider the user moving

mode and network characteristics such as communication qualities in mobile network. To fill these gaps, we conduct the research in this thesis.

2.4 Adaptive bitrate control

2.4.1 Overview

It is important to provide video streaming with high QoE for user which becomes more and more essential, since the user QoE is directly related to service provider's revenue [35, 36]. In order to maximize the QoE, the basic requirement is providing contents with higher video quality (or bitrate) and fewer rebuffering duration. Since the network condition is not always stable, transmitting contents with constant bitrate may result in troubles. Suppose the highest streaming quality is always chosen under an environment with inadequate bandwidth, the rebuffering events may occur very frequently. Then, the user may be upset and quit the video session, resulting in less chance of promoting commercial contents such as advertisement. However, if the possible network throughput is not fully exploited, the video may be streamed with a relatively low quality, which may also damage the user QoE and decrease the user engagement. Therefore, the adaptive bitrate control should be involved during the video streaming to choose proper video bitrate dynamically via trading off between video quality and rebuffering.

There exist several adaptive streaming protocols such as Adobe HTTP Dynamic Streaming [37], Apple HTTP Live Streaming [38], and Microsoft Smooth Streaming [39]. Recent years, dynamic adaptive video streaming over HTTP (DASH) is studied worldwide as a unifying standard [40]. In DASH protocol, the video contents are divided into short chunks and encoded at different bitrate levels. Then the client player can request the segment chunks with proper bitrate successively and dynamically according to the network condition. The algorithm for selecting download bitrate is called adaptive bitrate (ABR) algorithm. The ABR algorithm employs the network condition logs (such as throughput, buffer state and etc.) which monitoring in the client side to decide the bitrate of the latter downloading chunks. The purpose is maximizing the video quality while reducing rebuffering. The basic ABR control method is rate-based (RB) [41-43]. The RB algorithm selects the next downloading chunk by estimating the future throughput. The development of ABR method is still ongoing since it started in only recent few years. RB algorithm may perform badly if the prediction is inaccurate.

In the ABR algorithms which involves prediction, the throughput prediction

method is basically chosen as harmonic mean of the former several measurements. The impacts of different throughput prediction methods are not discussed. In this thesis, we mainly focus on the influence of different prediction methods on the RB adaptive control algorithm. In order to evaluate the methods with limited experiments, we established a replicable trace-based emulation environment. Since there are rarely literatures discussing such subject, our work may give a deeper insight into the effect of the throughput prediction. Note that, this thesis is not aiming to develop an algorithm which can defeat other state-of-the-art ABR methods. This is a future research topic. But we evaluate how the prediction methods will influence the performance of the adaptive bitrate control. Then some design guidance of ABR algorithm can be achieved, giving directions of future work. The quality metrics such as average bitrate, the number of rebuffering events, the duration of rebuffering, initial delay, bitrate switches are calculated and compared.

2.4.2 MPEG-DASH

MPEG-Dynamic Adaptive Streaming over HTTP (DASH) [44] is an adaptive bitrate streaming technique which realizes the video streaming with high quality provided to user through the internet from the HTTP server. As the technology is developed by Moving Picture Experts Group (MPEG), it is also named MPEG-DASH.

MPEG-DASH standard is designed for HTTP streaming. In MPEG-DASH, the media source of various resolutions is encoded at different bitrate. The media content is divided into various segments. Media Presentation Description (MPD) contains the information of media content components and segments such as resolution, bitrate, start time and end time. Figure 2.1 shows the structure of MPD file. The client can choose the proper bitrate based on the network condition. An example of dynamic bitrate adaptation for multimedia content is shown in Figure 2.2.

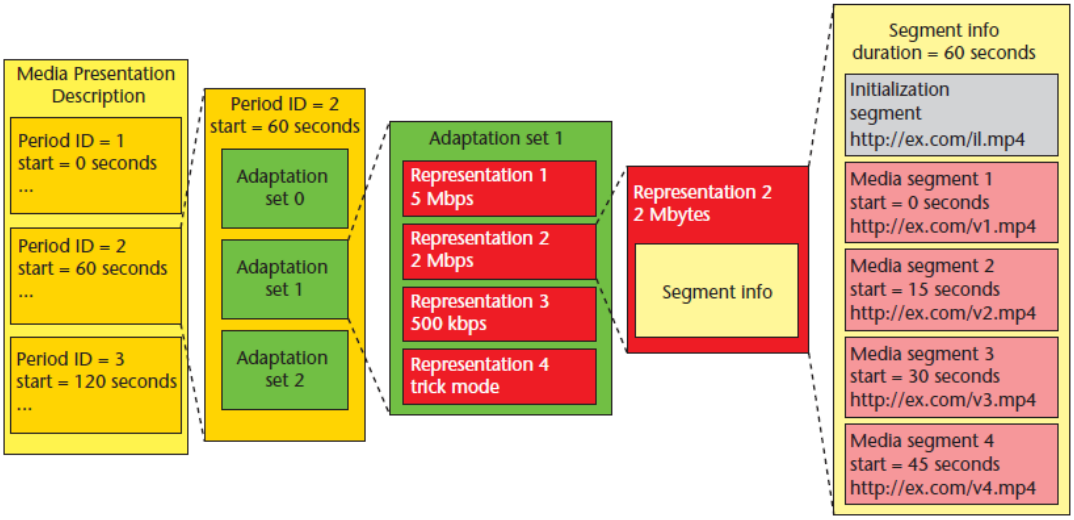


Figure 2.1 Structure of Multimedia Presentation Description (MPD) file [44].

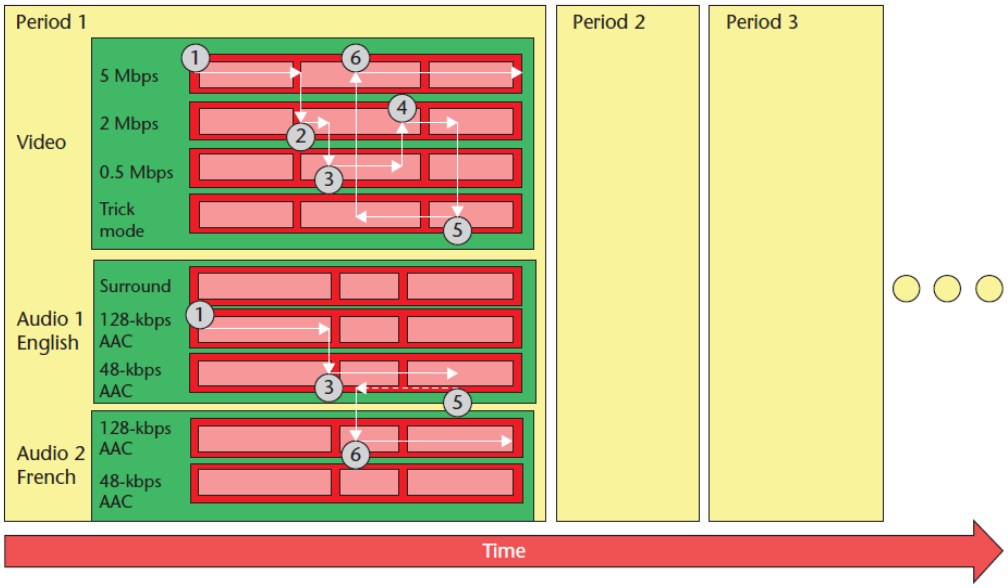


Figure 2.2 Example of dynamic bitrate adaptiation in MEPG-DASH [44].

DASH-JS [43] is a JavaScript-based MPEG-DASH client. It achieves a DASH client which is flexible and independent of browser by the Media Source API of Google's Chrome browser. It is robust in the real-world environment and has the best performing adaption algorithms. Moreover, it realizes the best application for the playback of media content for MPEG-DASH.

2.5 QoS and QoE

Quality of service (QoS) represents the performance of service for computer network, which is from the viewpoint of media provider [45]. The parameters of QoS include throughput, bitrate, packet loss, delay and etc. QoS can be utilized as the quality metric of a service. High QoS usually means providing high performance media with high bitrate, low latency and low bitrate error. The QoS of mobile network is very complicated as the communication is easily to be affected by many factors such as the transmission mechanism and environment.

Quality of experience (QoE) is used to measure the user's experience and feeling of the communication service. It depicts the whole experience of the client side to the service of the received information. The QoE generates from QoS. The difference of them are that QoS focuses on the service parameters and much relates to the media and the communication network, while QoE concentrates on the subjective perception from the user's point of view. Based on [46], the QoE factors include human influence, system influence and context influence as shown in Table 2.1.

Table 2.1 QoE Factors [46].

Human Influence	1. Low-level processing (visual and auditory acuity, gender, age, mood) 2. Higher-level processing (cognitive processes, socio-cultural and economic background, expectations, needs and goals, other personality traits, etc.)
System Influence	1. Content-related 2. Media-related (encoding, resolution, sample rate, etc.) 3. Network-related (bandwidth, delay, jitter, etc.) 4. Device-related (screen resolution, display size, etc.)
Context Influence	1. Physical context (location and space) 2. Temporal context (time of day, frequency of use, etc.) 3. Social context (inter-personal relations during experience) 4. Economic context 5. Task context (multitasking, interruptions, task type) 6. Technical and information context (relationship between systems)

QoE is important for the design of a system especially for video service. Subjective quality evaluation and objective evaluation methods are used to obtain the QoE of the communication service. Subjective quality evaluation needs lots of volunteers to offer their evaluation results, thus it is very time consuming. The objective evaluation method uses a stable function and related parameters to calculate the QoE. It can save more time and labor force, though may be not as accurate as the subjective quality evaluation which is actually from user's evaluation.

The main purpose of network management is providing high QoS and QoE for the media transmission. Adaptive bitrate control is the main technique in network management. In this thesis, QoE is adopted as the measurement which is aimed to be optimized for the adaptive bitrate control methods.

2.6 Summary

As mentioned in the above parts, throughput prediction plays an important role for adaptive bitrate control in mobile network. The realization of adaptive bitrate control can ensure high QoS/QoE for video streaming. All of these motivate the research in this thesis. We construct TCP throughput prediction methods in this thesis, and conduct field experiments to evaluate the methods. Meanwhile, the performances of the adaptive bitrate control method utilizing proposed methods are evaluated.

3 TCP throughput prediction using time series analysis by statistics and machine learning*

Throughput prediction is one of the promising techniques to improve the quality of service (QoS) and quality of experience (QoE) of mobile applications. To address the problem of predicting future throughput distribution accurately during the whole session, which can exhibit large throughput fluctuations in different scenarios, especially scenarios of moving user, we propose the history-based throughput prediction methods that utilize time series analysis and machine learning techniques for mobile network communication. The throughput prediction method utilizing hidden Markov model (HMM) with Gaussian mixture model (GMM) is proposed to deal with history time series of throughput and judge fluctuation factor with total variance when predicting future throughput. Further, a developed version of this method called the hybrid prediction with the autoregressive model and hidden Markov model (HOAH) is proposed. Different from existing methods, HOAH uses support vector machine (SVM) to classify the throughput transition into two classes and predicts the transmission control protocol (TCP) throughput by switching between the autoregressive model (AR Model) and the Gaussian mixture model-hidden Markov model (GMM-HMM).

3.1 Time series analysis and forecast

3.1.1 Time series

A time series is a series of samples in time order. Generally a time series is a sequence obtained continuously in stable time interval which is a series of discrete data. Time series are widely used in statistics, pattern recognition, econometrics, signal processing, communication engineering, weather forecasting and other areas. In the history based method for throughput prediction in the former chapter, throughput data is generally being considered as time series. Thus in this research, we consider the TCP throughput measurements as time series. Figure 3.1 shows an example of time series.

*This chapter is adapted from the work published in [49], [63].

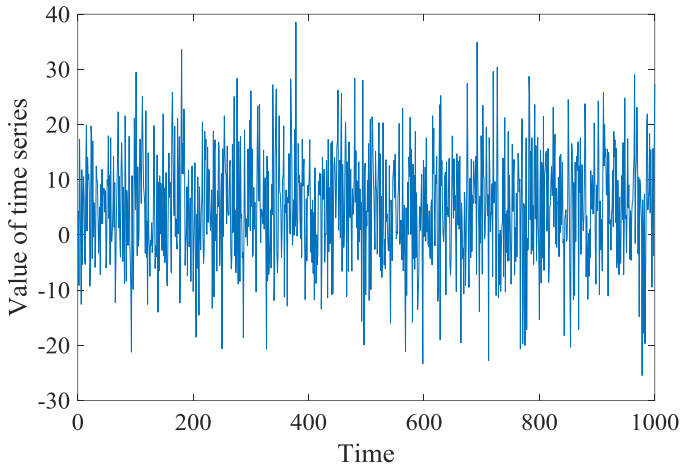


Figure 3.1 Example of time series.

3.1.2 Time series analysis techniques

Time series analysis aims at extracting the characteristics of time series data. The analysis methods include frequency-domain methods and time-domain methods. As the throughput data is temporal-spatial measurements which the most important characteristic is value, thus we focus on time-domain methods.

In this research, statistical analysis includes stationarity analysis, correlation and partial correlation, mean value and variance.

1. Stationarity analysis

Stationarity analysis plays an important role in the constructed prediction model in this thesis. Stationarity process is one kind of stochastic process, the unconditional joint probability distribution of which does not change with time. [47] Thus parameters such as mean value also do not change over time.

Wide-sense stationarity is commonly used in signal processing. In this case, the mean value keep stable, and the autocorrelation only depends on the time interval between the two points. In this thesis, wide-sense stationary is utilized.

Unit root test tests whether a time series variable is non-stationary and whether it shows a unit root. Augmented Dickey–Fuller test (ADF) is one of the most popular unit root test methods. There are three models in the ADF test. In the thesis, we utilize the three models in ADF test to identify the data stationarity situation and to predict future data.

2. Correlation and partial correlation

Correlation is used to represent the statistical association. Commonly it is adopted to show how close the linear relationship between two variables.

Autocorrelation describes the correlation between a time series of throughput data and the time lagged copy of itself. It can show the similarity between samples which appear in different time periods. Partial correlation gives the partial correlation of a time series of throughput data with its own shorter lagged values. Both of these are being used in the throughput prediction methods.

3. Mean and variance

There are many kinds of mean values in mathematics including the arithmetic mean (AM), geometric mean (GM), harmonic mean (HM), which all reflects the average of samples.

Variance is the expectation of the squared standard deviation of the variables from its mean value. It shows how far the values are distributed from the average value.

3.1.3 Time series forecast techniques

The time series forecast techniques utilize former data of the time series to predict future data. The former data is adopted to construct a model which covers the characteristics of the time series, then the model is used to forecast following data. The popular methods are techniques such as regression, moving average, Kalman filtering, linear prediction, curve fitting, machine learning etc.

3.2 Throughput data collection and analysis

First, we illustrate the throughput data collection, and analyze the characteristics of measurements.

An Android application developed by our laboratory is utilized to collect communication measurements. The application is developed by Java through android studio. The measurements include throughput data, packet size, RSSI, Cell ID, global positioning system (GPS) data, interface, battery status and other sensor data. The collected data are saved in the cellphone and can be sent to the user through email. We simply utilize GPS to get the location information. Figure 3.2 shows the screen when the application is working.

Figure 3.2 Screen of the application when it is working.

The throughput measurements can be visualized by google API. Figure 3.3 shows the visualization of throughput data with location information. The red color represents the throughput value is high, while the green color represents a low value of throughput data.

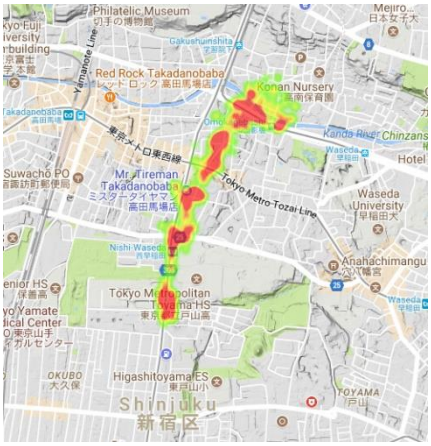


Figure 3.3 Visualization of throughput data with location information.

The throughput data are collected in different scenarios. The scenarios are shown in Table 3.1 and 3.2. We define one scenario as the situations which share the same scenarios properties including time, location, user moving mode and interface.

Besides the data collected by ourselves, we also use a public dataset which contains logs from TCP streaming sessions in Telenor's 3G/HSDPA mobile wireless network in Norway. [48]

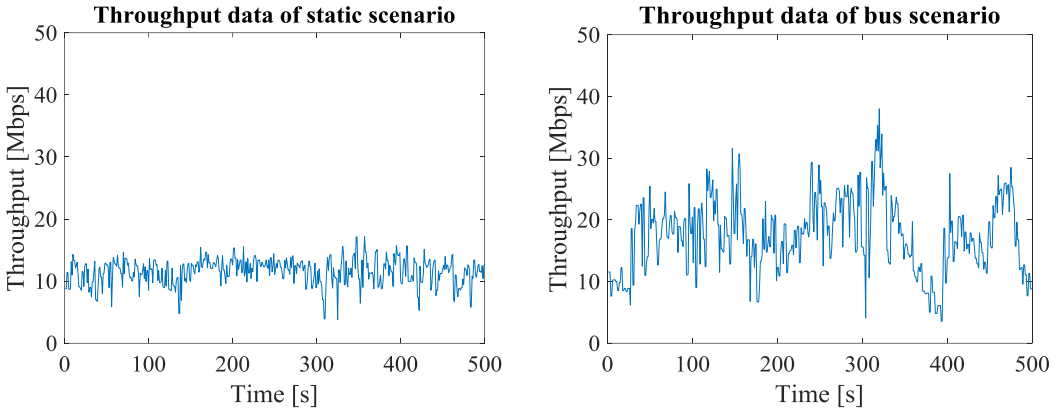
Table 3.1 Scenario properties.

Scenario Property	Details
Time	Morning, Afternoon, Evening
Location	Lab, One stable trace, etc.
User moving mode	Subway, Bus, Train, Walk, Static
Interface	LTE, Wi-Fi

Table 3.2 Scenario properties of public dataset.

Scenario Property	Details
Unix timestamps	number of seconds since 1970-01-01
Location	Lab, One stable trace, etc.
User moving mode	Subway, Bus, Train, Walk, Static
Interface	HSDPA

Throughput data of different scenarios in Table 3.3 are shown in Figure 3.4. In different scenarios, throughput data show different performances. The interval of the collection is 1 second. We have also collected the measurements with the intervals of 0.5 s, 1 s, 2 s, etc. The data are actual measurements collected by our software in Tokyo.



(a) (b)

Figure 3.4 Throughput data of static and bus scenario.

Table 3.3 Scenario information of the data above.

Scenario	Interface	Location	Time
static	LTE	Nishi-waseda campus	afternoon
bus	LTE	Nishi-waseda to Waseda campus	afternoon

Statistical characteristics of throughput measurements in different scenarios show very different performances. We give a brief analysis of the characteristics of throughput data in various scenarios. Figure 3.5 shows the characteristics of the collected throughput data mentioned above in static and bus scenario.

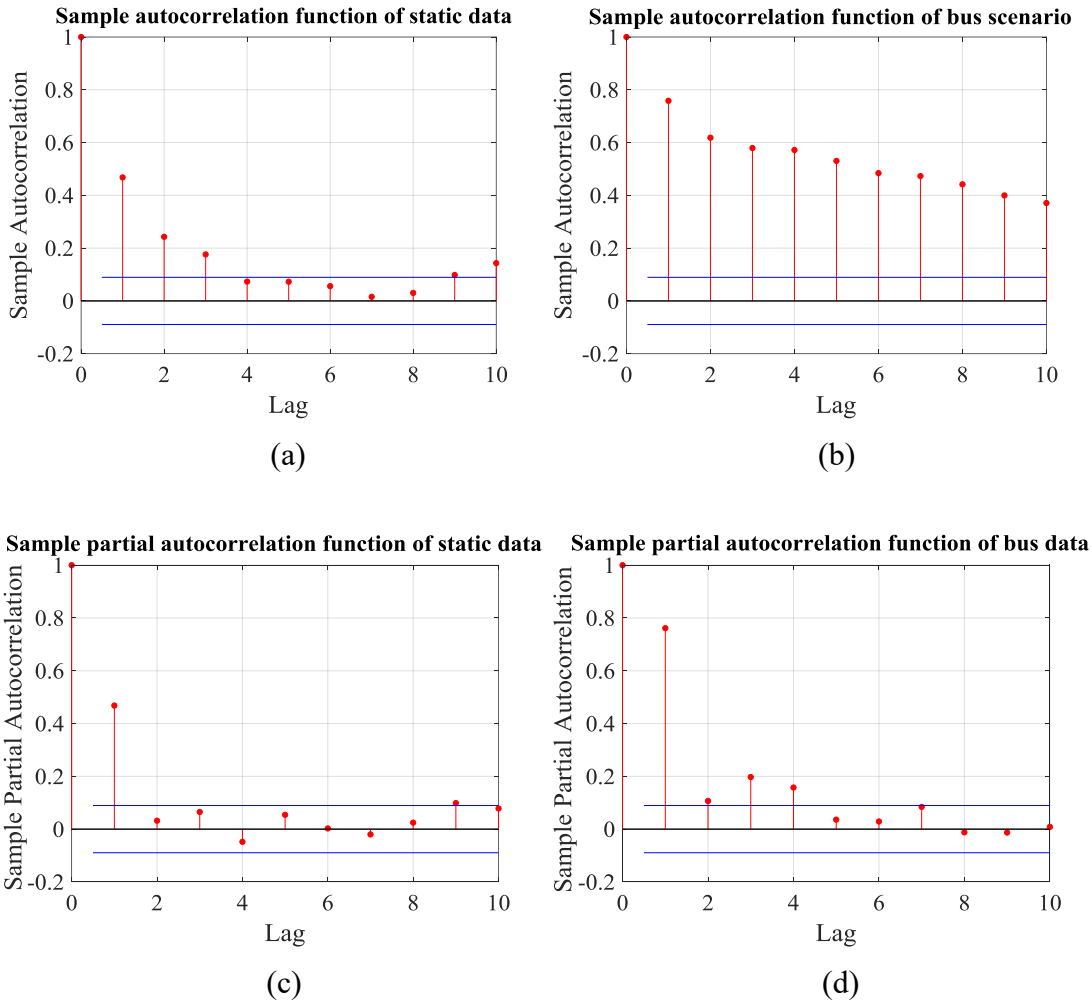


Figure 3.5 The statistical characteristics of throughput data.

1. Autocorrelation and partial autocorrelation

The autocorrelation and partial autocorrelation of the throughput data in static and bus scenario are shown in Figure 3.5. The autocorrelation of static data in Figure 3.5(a) tails off and the partial autocorrelation in Figure 3.5(c) cuts off, thus the data follow AR model. The autocorrelation of bus data in Figure 3.5(b) tails off and the partial autocorrelation in Figure 3.5(d) tails off, thus the data follow ARMA model.

2. Mean and variance

The mean value of the static data is 11.544 Mbps, while the mean value of bus data is 17.497 Mbps. The average throughput of bus is much higher than the static scenario. The variance of static data is 4.368 $Mbps^2$, and the variance of bus data is 35.820 $Mbps^2$. We can conclude that the data of bus scenario is much more distributed than static scenario.

3.3 TCP throughput prediction using GMM-HMM

3.3.1 GMM and HMM

A Gaussian mixture model is a probabilistic model which consists of finite number of Gaussian distributions. All the values in the dataset are assumed to follow the Gaussian mixture model which is a continuous probability distribution. Each Gaussian component has its own parameters such as expectation, variance and component efficient. The probability density function of Gaussian distribution is:

$$f(y \mid \mu_i, \sigma_i) = \frac{1}{\sqrt{2\pi\sigma_i^2}} \exp\left[-\frac{(y-\mu_i)^2}{2\sigma_i^2} \right] \tag{3.1}$$

where y is variable, μ_i is expectation, σ_i is variance. The probability density function of Gaussian distribution is:

$$g(y \mid \omega_i, \mu_i, \sigma_i) = \sum_{i=1}^{m} \omega_i f(y \mid \mu_i, \sigma_i) \tag{3.2}$$

where ω_i ($0 \le i \le m$) denotes a mixture weight of *i-th* Gaussian density component in GMM. GMM use expectation maximization algorithm to obtain the parameters.

Hidden Markov Model (HMM) is a statistical Markov model and the system being modeled is assumed to be a Markov process. The hidden states of HMM is not visible, while the output dependent on the states can be observed. The output is also called observation. Each state has a probability distribution over the possible

observations. The type of HMM being considered is like this: the hidden states are discrete, while the observations can be discrete or continuous and generally will be assumed follow Gaussian distribution. The parameters of HMM include transition probability and emission probability. The transition probability decides the hidden state at next time t+1 is chosen given the hidden state at time t.

3.3.2 Throughput prediction using GMM-HMM

3.3.2.1 Whole structure of proposal

In this part, we elaborate the proposed method: history-based throughput prediction with hidden Markov model in [49]. The method mainly consists of two parts, identification of throughput fluctuation and throughput prediction. As GMM can be used to cluster data according to frequency, we use GMM to obtain clusters and define them as different hidden states. HMM is adopted to dig out the pattern how hidden factor decides the observations, as forward-backward algorithm can produce the most likely model for the state and observation sequence, and Viterbi algorithm could find the most possible state transition path of throughput instead of the real throughput value which might contain outliers and sudden change to avoid interruption when we judge current fluctuation. By applying GMM-HMM, the new method could separate fluctuation of data into stationary and non-stationary accurately. Then, we use proper means to predict future throughput in different conditions effectively. The related notations used in HMM are shown in Table 3.4.

Table 3.4 Notations used in HMM.

Notation	Meaning
m	number of states in the model
M	number of distinct observations of each state
T	length of observation sequence
$O = (O_1,...,O_T)$	observation sequence
$Q = (Q_1,...,Q_T)$	state sequence
$A = \{a_{ij}\}$	A is state transition matrix, a_{ij} is transition probability from state i to j
$B = \{b_i(O_t)\}$	B is observation emission matrix, $b_i(O_t)$ is when state is i, probability of observing O_t
$\pi = \{\pi_i\}$	π is prior probability matrix, π_i is the beginning probability of being in state i
$\lambda = (\pi,A,B)$	HMM model with parameter
$S_i\,(1 \le i \le m)$	Current State is i
$r_k\,(1 \le k \le M)$	Observation k (value range of throughput data)

The whole process of history-based throughput prediction with hidden Markov model is shown in Figure 3.6. History data are separated into *M* groups according to the value range as observations. Then data are clustered into *m* classes with *m*-component Gaussian mixture model and defined as hidden states. After we have state sequence and observation sequence we train data with hidden Markov model, and calculate the most probable state transition path. Then we judge fluctuation (stationary or non-stationary) of current time according to the fluctuation factor calculated by former state transition path. Finally, we predict throughput of following time with corresponding means: linear regression for stationary and locally weighted linear regression for non-stationary.

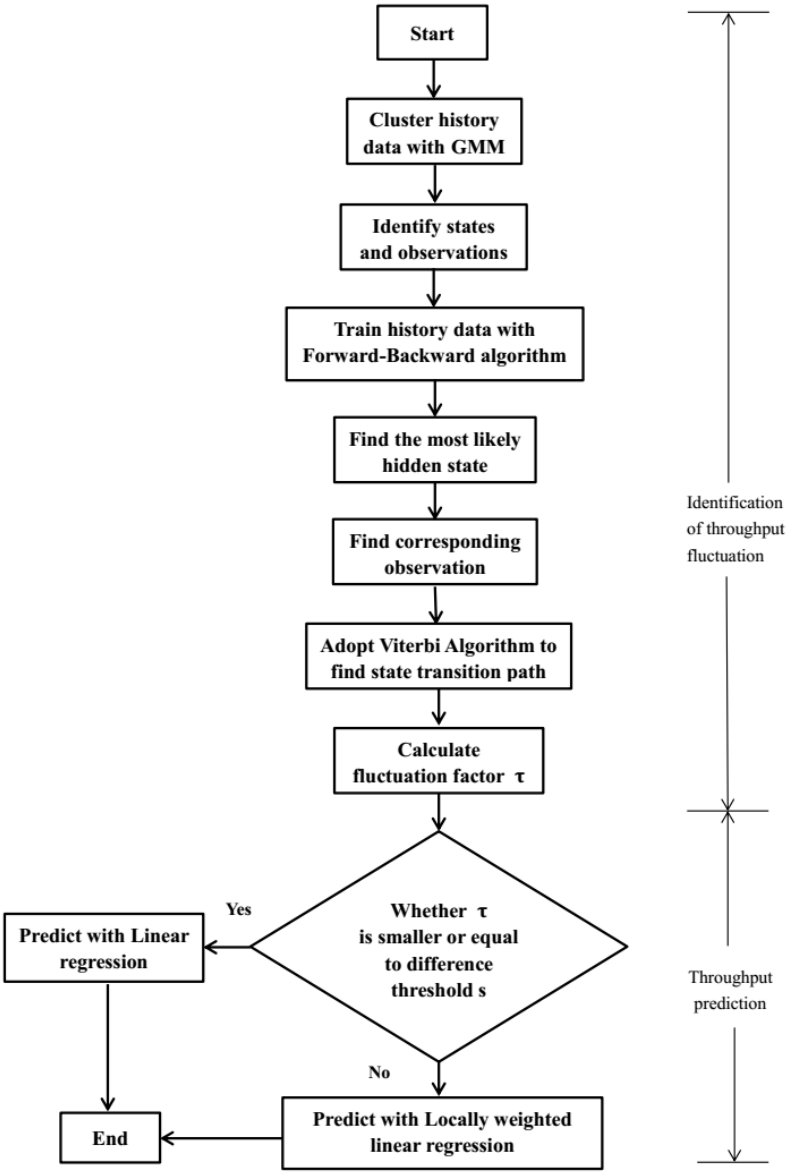


Figure 3.6 Process of throughput prediction with HMM.

3.3.2.2 Gaussian mixture model

Gaussian mixture model (GMM) is a reliable classification tool in many applications such as pattern recognition [50-52]. In the part of generating states of this work, GMM is adopted to cluster throughput data into different classes according to distribution of data. The parameters of GMM for the classes can be obtained by training data with standard expectation maximization algorithm. The classes are defined as distinct states which demonstrate that data follow different distribution structures of different expectation and standard variance. States are value ranges in which the data appear with largest probability. The probability density function (PDF) of GMM is a weighted sum of *m*-component Gaussian densities given by the following equations:

$$p(x \mid \omega_i, \mu_i, \sigma_i) = \sum_{i=1}^{m} \omega_i g(x \mid \mu_i, \sigma_i) \tag{3.3}$$

$$0 \le \omega_i \le 1, \sum_{i=1}^{m} \omega_i = 1 \tag{3.4}$$

$$g(x \mid \mu_i, \sigma_i) = \frac{1}{\sqrt{2\pi\sigma_i^2}} \exp[-\frac{(x-\mu_i)^2}{2\sigma_i^2}] \tag{3.5}$$

where ω_i ($0 \le i \le m$) is a mixture weight of corresponding PDF of Gaussian distribution in GMM. μ_i and σ_i are the expectation and standard variance of the *i*-th Gaussian density component. $\psi(x,i)$ is defined as the occurrence probability of throughput data *x* to which the *i*-th Gaussian density component contributes as shown in Equation (3.6). And, the component which contributes to the largest probability can be considered as the state(*i*) that data *x* belongs to.

$$\psi(x,i) = \omega_i g(x \mid \mu_i, \sigma_i) \tag{3.6}$$

The number of Gaussian mixture model components is decided by historical data. Throughput with the value fluctuating slightly will be clustered by two Gaussian mixture models while data with large fluctuation will be clustered by three or more Gaussian mixture models.

3.3.2.3 Hidden Markov model

Hidden Markov model (HMM) is a finite state machine with state transitions and observations, which is widely applied to temporal pattern recognition such as speech, handwriting recognition, and DNA sequence analysis [53-55]. It provides a probabilistic approach for modelling the time series of states and observations. In this work, HMM is proposed as a predictor being used to find the hidden law of throughput variation, as HMM could not only construct the connection of hidden factors and corresponding observations it decides, but also build the transition probability between hidden factors. Thus, we could obtain the most probable state transition path, with which we could know the real current fluctuation status. This is much more accurate than analyzing the real change of throughput value directly which might be affected by outliers and sudden changes. First, the forward-backward algorithm is employed to find the proper model and parameters. Then, Viterbi algorithm is applied to find the most probable state transition path. After this, current fluctuation factor with total variance is calculated and the fluctuation of current time is judged. Finally, throughput is predicted with a corresponding way based on fluctuation condition. This work meets the assumptions of HMM.

We define the classes of GMM as hidden states S_i ($0 \le i \le m$) in HMM and divide the throughput historical data into M ranges according to value range with definition of M observations r_k ($0 \le k \le M$). Thus, we obtain the state sequence Q, and observation sequence O. Then, we train state and observation sequence with the forward-backward algorithm to generate the most probable hidden Markov model $\lambda = (\pi, A, B)$. After predicting next state and corresponding observation, we train the observations with Viterbi algorithm to find the most likely sequence of hidden states for the given hidden Markov model and observation sequence. Then, we get the state transition path and calculate fluctuation factor of current time.

a. Forward-backward algorithm

After defining the state and observation sequence, we use forward-backward algorithm to generate the most probable HMM parameters from given sequences.

b. Viterbi algorithm

Viterbi algorithm is used to find the most likely sequence of hidden state sequences when given the hidden Markov model and observation sequence. It uses backtracking array to obtain the most probable hidden state sequence. After above process, the state path we need is the backtracking array. Thus we obtain the state transition path.

3.3.2.4 Fluctuation factor

We apply mean square of total variation [56] to judge the fluctuation of state transition path. After obtaining the most probable state transition path by Viterbi algorithm, we calculate the difference e_t between adjacent states of former N states (including current state) and the fluctuation factor τ (standard difference) as:

$$\tau = \sqrt{\frac{e_1^2 + e_2^2 + \ldots + e_N^2}{N}} \tag{3.7}$$

Then, we compare the value of τ with parameter s, called difference threshold, which can be assigned different value according to various situations and requirements by user. When $\tau \leq s$, the state transition shows low-frequency and low-width, we define current fluctuation as stationary, and we apply linear regression to predict future throughput with former N_1 throughput data; when $\tau > s$, the state transition shows high-frequency or several high-width, we define current fluctuation as non-stationary, and use locally weighted linear regression to predict future throughput with former N_2 throughput data (N_2 is usually smaller than N_1). This means that for non-stationary condition, we use local regression to smooth data by giving suitable weight to data points. Because we think predicted value is only reflected by latest data largely without distant ones.

Figure 3.7 State transition path.

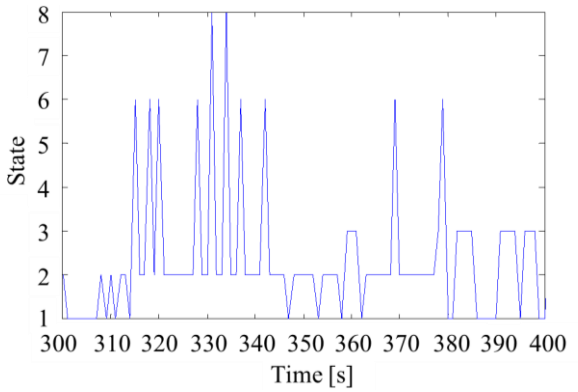


Figure 3.8 Simple example of state transition path.

3.3.2.5 Linear regression and locally weighted linear regression

After judgement of above steps, if fluctuation of state transition path is stationary, we use linear regression to predict throughput. Linear regression is a popular method modeling the relationship between a dependent variable and one or more independent variables, which has many practical uses [57-59]. In this work, linear regression is applied to predict throughput when data fluctuate slightly. It fits the predictive model with former N_1 throughput values x_i. After developing the model and obtaining parameter θ_0 and θ_i, we use the known fitted model to predict current throughput x_t as $\hat{x}_t$. Prediction function is shown as follows:

$$\hat{x}_t = h_\theta(t) = \theta_0 + \theta_i \cdot t, \quad 1 \le i \le N_1 \tag{3.8}$$

$$J(\theta_0, \theta_i) = \sum_{i=1}^{N_1} (h_\theta(t_i) - x_i)^2 \tag{3.9}$$

where θ_0 is constant term, θ_i is the relative parameter of t, and $\hat{x}_t$ is predicted future throughput. We fit θ to minimize evaluation function and obtain parameters and to predict x_t.

If the fluctuation is non-stationary, we use locally weighted linear regression to predict throughput. The method combines linear regression model in a N_2-nearest-neighbor-based one to smooth data. It addresses conditions in which the classical procedures do not perform well or cannot be effectively applied with undue labor caused by outliers, which disturb predicting future data accurately. The smoothed value is determined by neighboring data points defined within the limit. We give larger weight to nearer data, thus the predicted throughput will not be affected by the distant data at all. The weights are calculated by following equation:

$$v_i = \left(1 - \left| \frac{t - t_i}{d(t)} \right|^3 \right)^3, \quad 1 \le i \le N_2 \tag{3.10}$$

$$J(\theta_0, \theta_i) = \sum_{i=1}^{N_2} v_i (h_\theta(t_i) - x_i)^2 \tag{3.11}$$

In Equation (3.11), we fit θ to minimize evaluation function $J(\theta_0,\theta_i)$ and obtain parameters and to predict x_t, where v_i is weight, $\hat{x}_t$ is the predicted value, x_i are the

nearest neighbors, and *d(t)* is the distance along the time axis from x_t to the most furthest value in the range.

3.3.3 Model validation

3.3.3.1 Experiment environment

We use the developed application to collect throughput history data every second from the HTTP server via LTE offered by NTT DOCOMO. Figure 3.9 shows the experiment environment.

Figure 3.9 Experiment Environment.

We obtain throughput history data in the afternoon and evening with various cases of user mobility. In the static situation, a user keeps static when collecting data in the laboratory. While in dynamic situation, a mobile user collects data when walking from the station of Takadanobaba to Nishi-waseda campus or sitting on moving subway between stations. We ensure same data transmission path for each case. Time and user mobility of six cases are shown in Table 3.5. The time interval of data series is 1 s. We train former 850-second historical data to predict next future 100-second throughput. Then, we analyze prediction results and evaluate prediction accuracy. We compare the accuracy of the predicted throughput with three other prediction methods: stochastic model, linear regression and locally weighted liner regression.

Table 3.5 Experiment cases.

Cases	Environment
Case 1	Afternoon, walking
Case 2	Afternoon, static
Case 3	Afternoon, subway
Case 4	Evening, static
Case 5	Evening, walking

3.3.3.2 Fluctuation identification

Figure 3.10 shows the experiment result of clustering data with GMM in case1. Inside GMM, there are six Gaussian components used as hidden states. And they superimpose on each other largely from 10 Mbps to 20 Mbps, which means many throughput data distribute intensively during this value range. We can conclude from the result that majority of data value appear between 5 Mbps to 35 Mbps. Figure 3.11 is identification of throughput fluctuation. The upper dots represent non-stationary status and lower ones show stationary status. Some throughput data from 910 s to 930 s are non-stationary as they perform larger fluctuation. Figure 3.11 indicates the method could identify different fluctuation statuses effectively.

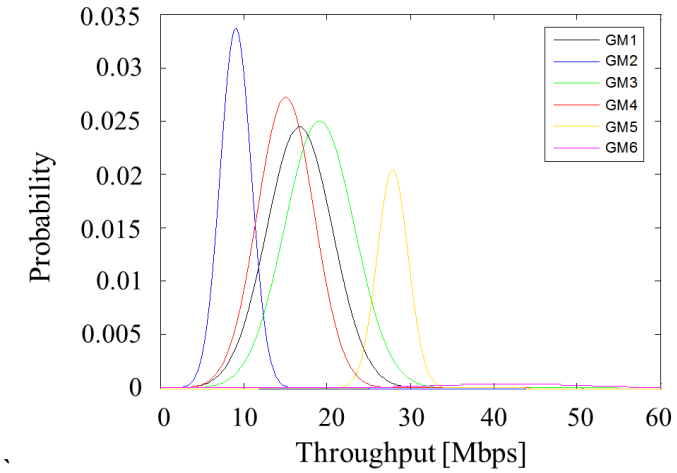


Figure 3.10 Clustering data with GMM.

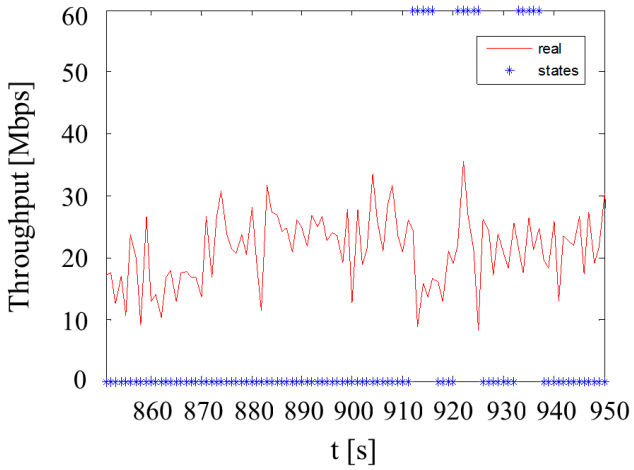


Figure 3.11 Identification result of throughput fluctuation.

3.3.3.3 Accuracy comparison and analysis

Similar to the reference [29], we use relative prediction error R_t and root mean squared relative error (*RMSRE*) with following equations to evaluate accuracy as calculated in Equation (3.12) and (3.13), respectively. We adjust related parameters in different situations to achieve high accuracy and meet various requirements of users.

$$R_t = \frac{|x_t - \hat{x}_t|}{x_t} \tag{3.12}$$

$$RMSRE = \sqrt{\frac{1}{n}\sum_{t=1}^{n} R_t^2} \tag{3.13}$$

The cumulative distribution function (CDF) of R_t is shown in Figure 3.12. As the limitation of space, we only show case 1 out of the six cases in our experiment. Figure 3.12 shows the prediction result of case1 utilizing four methods. From the figure, we can see that initially the value of R_t is almost the same for all methods, while they separate when R_t equals to about 0.2. From 40%, our proposed method is better than the linear regression prediction. The new method could divide data fluctuation into stationary and non-stationary accurately and apply suitable prediction methods for each condition. We can conclude that for 90% of the predicted 100 s throughput data, R_t of the proposed method is the smallest among the four methods. The smaller R_t is, the more accurate the predicted value is. Moreover, for 100% predicted data, our method has the smallest R_t for all forecasting data. In other word, value of the largest R_t is the smallest, which proves the new method is effective.

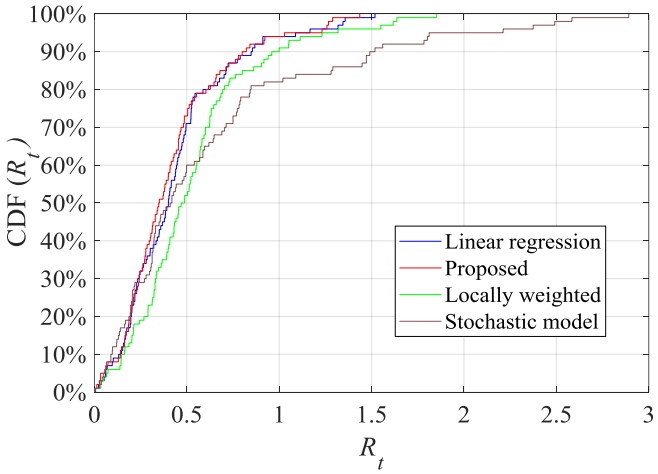


Figure 3.12 CDF of R_t for all predictions with four methods.

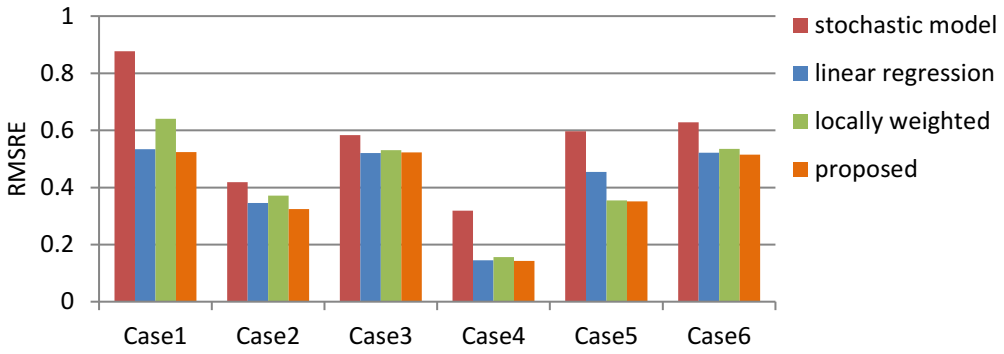


Figure 3.13 *RMSRE* of throughput prediction with four methods.

Figure 3.13 shows *RMSRE*s of forecasting with the four methods in six cases. *RMSRE* represents the prediction accuracy of all cases by single figure. We can see that, generally, the four prediction methods achieve higher accuracy when users are static compared with moving users. The proposed method predict future throughput effectively not only in situations of static users, but also in moving user cases. This is because new method could identify fluctuation of throughput data into stationary and non-stationary accurately with GMM-HMM and adopting proper means for each condition. By assigning proper values to relative parameters, the proposed method could predict future throughput accurately in different situations to satisfy users.

3.4 TCP throughput prediction using AR and GMM-HMM

3.4.1 Prediction system

Researchers have shown that user mobility scenario can be identified in [12], [60-62] which means that the user's communication scenario can be obtained for mobile networks. These researches support the premise for constructing a prototype system to predict throughput. Based on this premise, the non-trivial task is undertaken in this part. We focus on throughput prediction in different situations individually for a specific scenario and propose the TCP throughput prediction method using the hybrid prediction with the autoregressive model and hidden Markov model (HOAH). Figure 3.14 shows the overview of our work in [63].

To obtain a deep understanding of the throughput data variation and characteristics, large amount of throughput measurements in various scenarios are needed. We use the developed Android software application to collect throughput data per second of other network interfaces such as LTE and Wi-Fi, because the public

dataset [48] only contains the network interface of high-speed downlink packet access (HSDPA). According to our previous work [64], the throughput transition characteristics drastically change when the user changes the mobility route. Therefore, when testing the data in user mobility scenario, each scenario means one mobility route. Based on the conclusion of [29], the tested throughput data are considered coming from the same transmission path for each scenario.

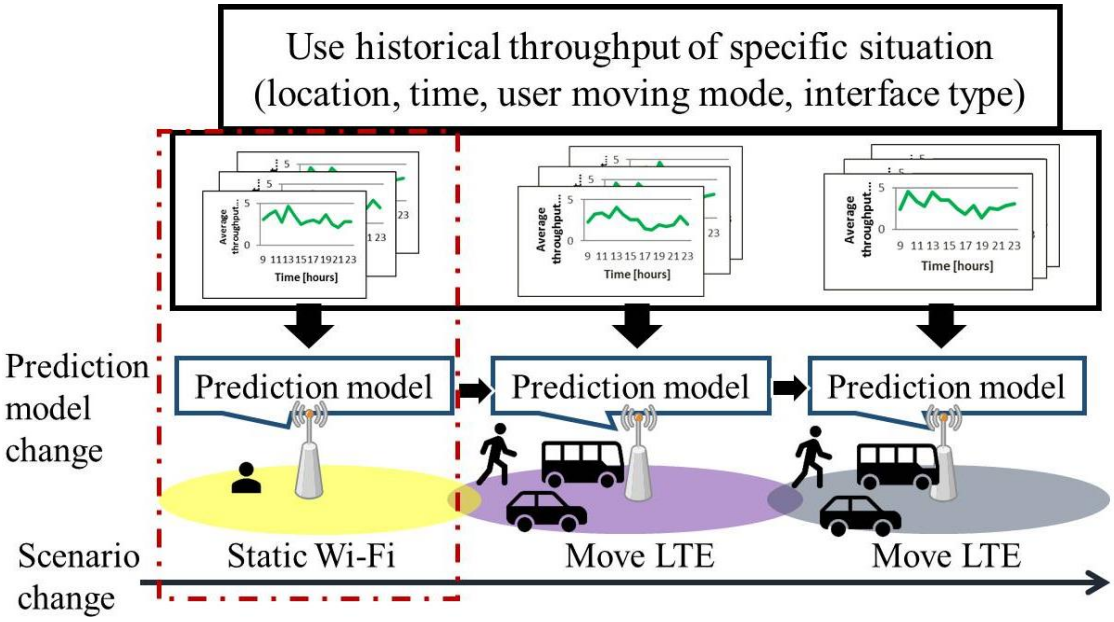


Figure 3.14 Complete structure of the prediction system.

3.4.2 Data analysis

We collect a large volume of continuous throughput data in many video sessions in different scenarios by one user, e.g. different network interface types, user behaviors, location and time are shown in Table 3.6.

Table 3.6 Scenario properties.

Scenario Property	Details
Time	Morning, Afternoon, Evening
Location	Lab, One stable trace, etc.
User moving mode	Ferry, Bus, Train, Walk, Static
Interface	HSDPA, LTE, Wi-Fi

After analyzing the data, we obtain three observations:

1. The throughput measurements of sessions in the same scenario tend to perform similar throughput variation process. Figure 3.15 shows an example of two realizations in the same walk scenario using LTE. We can easily observe that the realizations have similar performances. Thus it is better to analyze the data and predict throughput in each specific scenario respectively.

2. For the neighbor segments inside one session, the variation of the process shows very large difference such as staying stable or changing very sharply between the segments. Thus, throughput is difficult to predict by using only one method or one model. When predicting throughput, the former situation is fit for using AR Model, while the latter one may be better to be solved by GMM-HMM. Inside one segment, the data have different statistical characteristics, which will show different relationships with the variation process of the next segment with distinct probability. As shown in the second realization of Figure 3.15, if the segment size is 50 s, the performances of the first and second segments (period from 1 s to 50 s and period from 51 s to 100 s) are very different, and the changes are extremely sharp and sudden. Moreover, the fifth and sixth segments (period from 201 s to 250 s and period from 251 s to 300 s) show similar performances, and the changes are slight.

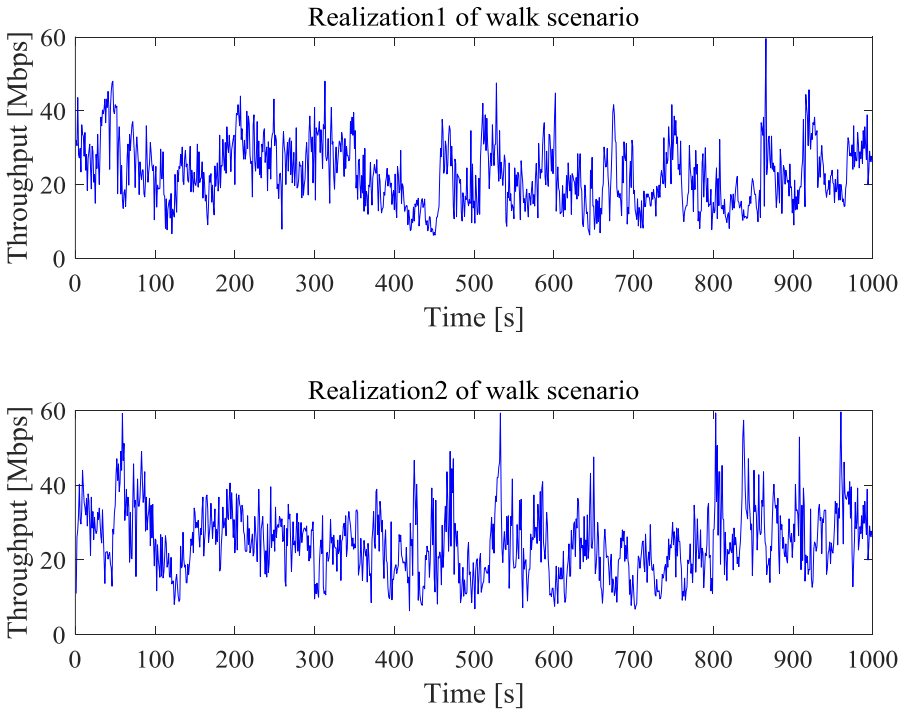


Figure 3.15 Two realizations of one walk scenario.

Therefore, it is suitable to use different methods to predict the throughput of the next segment. It is important to decide which method to use. In this paper, the choice is based on statistical characteristics of the current segment. The support vector machine (SVM) classifier is utilized to separate the segment into the correct class and predict with the corresponding method (AR or GMM-HMM).

3. The throughput of one session tends to have large fluctuation and shows multi-cluster characteristics, especially in moving user scenarios. Figure 3.16 shows an example of the clustering result for the mean value of a segment in the walk case. Two states can be observed in this trial, and the data have transition behavior from one state to another state. For other moving cases, such as bus and train, there are more state transitions for the mean value of the segments. Thus it is reasonable and necessary to adopt GMM-HMM for modeling state transition in HOAH.

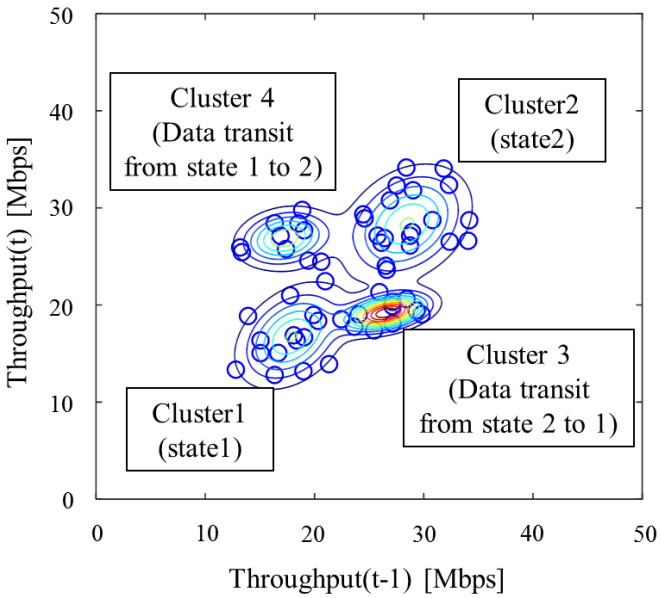


Figure 3.16 Example of the clustering segments' mean value for one realization in the walk scenario.

Although the two-class classification (AR and GMM-HMM) is used in this work, the results in the following sections demonstrate the advantage of the proposed method compared with the conventional methods where only one method is applied all the time. These results also show the necessity of choosing the proper model which proves the significance of this work.

3.4.3 Related techniques

3.4.3.1 SVM

In many scenarios, the data could not be divided into two classes with constant threshold values for two or higher dimension case. Thus, we use SVM to accomplish the classification task to assign a segment to two prediction models.

SVM is a kind of supervised learning model in machine learning, which is effective for classification [65], [66]. Many works, such as [67-69], have verified that SVM is more useful for time series classification than other machine learning methods. We use statistical characteristics of the data segment to classify data for properly choosing the prediction method. The features utilized in HOAH are shown in Table 3.7. These features are employed to depict the variation of the throughput data. Prediction does not work when the autocorrelation is around zero. However, note that this case rarely happens even if SVM can be trained.

The SVM maps the features of each segment into a higher dimension space by the Gaussian kernel function non-linearly and divides the segment into two classes. Since the transformation is nonlinear and the transformed space is high dimensional, the identification line is nonlinear in the original feature space.

Table 3.7 SVM features.

Feature	Meaning
Variance	Depicts how far the value of the throughput data is from the mean value.
Derivative	Describes both the direction and the steepness of data change.
Autocorrelation	Describes the correlation between a time series of throughput data and the time lagged copy of itself.
Partial autocorrelation	Gives the partial correlation of a time series of throughput data with its own shorter lagged values.

3.4.3.2 Basic AR model

In HOAH, if historical data is classified to *ar* class, it means that future data will probably follow the same behavior of the former period. Thus, the autoregressive model can be used to predict future data with short-term history samples accurately.

The autoregressive model is one of the most popular models to describe the performance of time series changing with time [70], [71]. HOAH applies the AR Model

as part of the basic model to predict throughput. HOAH attempts to predict the distribution of future throughput, which makes the mean value and variance important parameters. The basic AR(p) model is given by the following equation:

$$y_t = c + a_1 y_{t-1} + a_2 y_{t-2} + \ldots + a_p y_{t-p} + \varepsilon_t \tag{3.14}$$

where y_t is a sample of the time series, c is constant, a_i is the corresponding coefficient of y_i, and ε_t is the white noise error term that follows $N(0,1)$.

3.4.3.3 Augmented Dickey-Fuller test

In the AR Model of HOAH, the model is separated into three kinds based on the restraint of the coefficient, which is introduced in the Augmented Dickey–Fuller test. The theory of each model is given in the hypothesis test, which includes null hypothesis and an alternative hypothesis. Among the three models, Model 1 is the most basic autoregressive model with lag terms. And Model 2 has a drift term besides lag terms. Model 3 has lag term, drift term and trend term, indicating it is the most complicated one which can describe the most properties of data such as trend. Details are shown in Equation (3.15), (3.16) and (3.17).

Model 1:

$$y_t = a y_{t-1} + b_1 \Delta y_{t-1} + \ldots + b_p \Delta y_{t-p} + \varepsilon_t \tag{3.15}$$

Null hypothesis: $a = 1$

Alternative hypothesis: $a < 1$

Model 2:

$$y_t = c + a y_{t-1} + b_1 \Delta y_{t-1} + \ldots + b_p \Delta y_{t-p} + \varepsilon_t \ \left(c \neq 0\right) \tag{3.16}$$

Null hypothesis: $a = 1$

Alternative hypothesis: $a < 1$

Model 3:

$$y_t = c + dt + a y_{t-1} + b_1 \Delta y_{t-1} + \ldots + b_p \Delta y_{t-p} + \varepsilon_t \ \left(c, d \neq 0\right) \tag{3.17}$$

Null hypothesis: $a = 1$

Alternative hypothesis: $a < 1$

where y_t is the throughput time series, Δ denotes the differencing operator, Δy_i is the difference term of the adjacent data calculated as:

$$\Delta y_i = y_i - y_{i-1} \tag{3.18}$$

b_i is the coefficient of difference of the term Δy_{t-i}, a is the coefficient of y_{t-1} (data of the last second), p is the lag order of the autoregressive process. c is the drift coefficient, d is the deterministic trend coefficient, ε_t is an innovation process that follows a normal distribution with mean value of zero.

The ADF test is used to examine the stationarity for history throughput segments [72-74]. $a = 1$ means a unit root exists, and the data is non-stationary. Meanwhile, $a < 1$ denotes no unit root, and the data is stationary. Based on the ADF test, the proper AR model is selected.

Ordinary least squares (OLS) is used to calculate the estimated value of the coefficient for the corresponding model, and then use t statistic to evaluate the existence of the unit root. We take the models of the ADF test with $lag = 0$ as example because this is the easiest case of the ADF test. After the test, we compare the calculated value of the t statistic in Equation (3.19) with a known value, which is being calculated by Monte Carlo method. If t is larger than or equal to the compared value, the null hypothesis will be accepted, meaning the process is non-stationary. If t is smaller than the compared value, the null hypothesis will be rejected, while the alternative hypothesis will be accepted, which means that the process is stationary.

$$t = \frac{\hat{a} - 1}{Se(\hat{a})} \tag{3.19}$$

3.4.3.4 GMM and HMM

If the test result shows that the data belong to the *hmm* class, the future throughput may not have similar features with the short term historical data. Therefore, HOAH adopts the Gaussian mixture model and the hidden Markov model to predict the distribution of future throughput.

GMM and HMM are widely used in signal processing, which could analyze and recognize information [75], [76]. In HOAH, we utilize the GMM-HMM to predict the future throughput when future throughput may not have the similar features with the historical segment. This combination is to find the distribution of the future throughput,

which has the largest probability to appear. HOAH constructs a model of multi-Gaussian and Markov chain to reveal the hidden law of historical data and use the calculated distribution with the largest probability to appear as the distribution of the predicted throughput.

1. Gaussian mixture model

The Gaussian mixture model is a popular classification tool, which has been applied to many areas, such as pattern recognition [77], [78].

In the proposed model, GMM is applied to generate states based on the value features of data by training history throughput. If the historical data of throughput shows large fluctuation, the number of Gaussian mixture components will be set at four or more. Meanwhile, for historical data with small fluctuation, the number of the Gaussian distribution will be two or three.

2. Hidden Markov model

The hidden Markov model is a kind of Markov model that can be employed to model the stochastic system changing with random variables. HMM plays an important role in temporal pattern recognition, such as speech, gesture recognition, and DNA sequence analysis [54], [79].

In HOAH, HMM is adopted as a predictor to recognize the performance of throughput variation with time because it could describe the relationship between time series and its characteristics by using hidden states and observations. As a member of the dynamic Bayesian network, HMM could connect adjacent variables, implying that it is possible to calculate the hidden variable of the current time with internal regression and variable of former time. In HMM, the hidden states control the appearance of observations, and the observation at each time is visible while the states are invisible. The hidden states represent the combination of many factors, such as the number of users sharing the same channel and the condition of the bottleneck link.

3.4.4 Proposed model: HOAH

3.4.4.1 Structure of HOAH

Based on the AR Model and GMM-HMM, we demonstrate the hybrid prediction model named HOAH, the hybrid prediction with the autoregressive model and hidden Markov model. This method is a novel combination of GMM-HMM and AR-based model by using SVM as a classifier. It can predict the distribution of future throughput with proper method. The basic concept of HOAH is shown in Figure 3.17. We also adopt the sliding window to refresh the training segment.

We consider the data from the same scenario to be one dataset *DS*(*v*) = {data | scenario *v* (time, location, interface, moving mode)}. Among the data, the HOAH separates the dataset *DS*(*v*) into training data *TRDS*(*v*), validation data *VDS*(*v*), and test data *TDS*(*v*). *TRDS*(*v*) and VDS(v) are used to set the parameters, such as the number of states *m* in GMM-HMM and *lag* = *p* of the AR model. *TRDS*(*v*) is also used to obtain the SVM classifier. *TDS*(*v*) is utilized to evaluate the HOAH by comparing prediction results with other methods. The SVM classifier is trained by the training dataset of the assumed scenario and is adopted to predict throughput of the same scenario.

After assigning a specific value to the historical segment length and prediction segment length according to the user, the HMM and AR Model is used to predict the data in *VDS*(*v*) by training other data in *TRDS*(*v*) separately. We adjust the parameter in these two methods until the lowest prediction error is achieved. Thus, we determine the number of states *m* in HMM and *lag* = *p* of the AR model. Then, HOAH starts to construct the classifier.

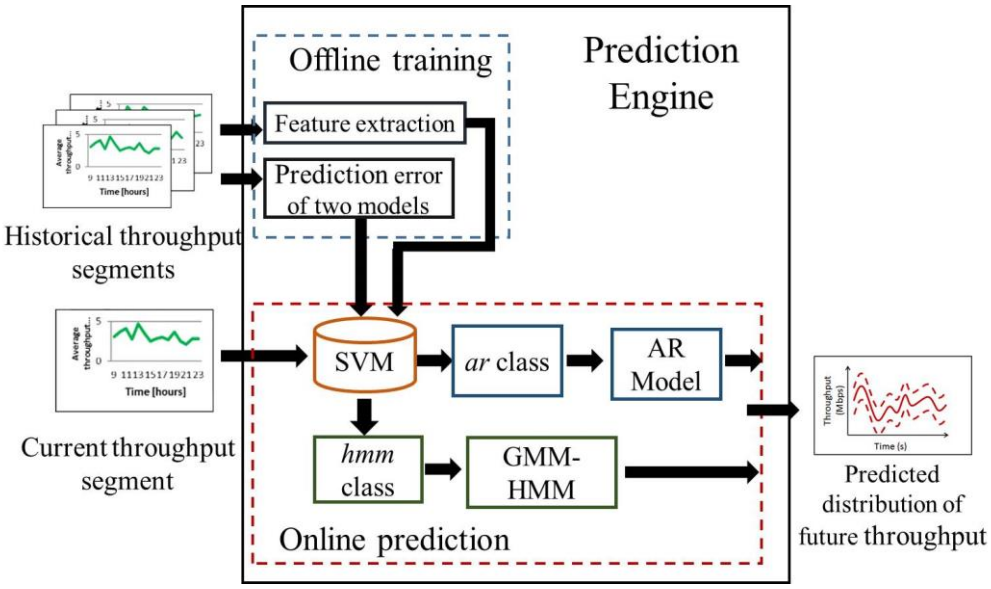


Figure 3.17 Basic concept of HOAH.

3.4.4.2 Segment classification with SVM

In HOAH, we construct the SVM by supervised learning with the dataset *TRDS*(*v*). It is also used to classify the segment data into *ar* class and *hmm* class to realize the model switching strategy. We train the SVM with the statistical features of historical segments and the prediction error of each method (AR Model and GMM-

HMM) of the historical segment. The 4-fold cross validation is applied to determine the parameters of SVM, such as *lag* = *h* for autocorrelation. The details are as follows:

The construction of classifier includes three steps.

1. HOAH decides and calculates the features of the segment in *TRDS*(*v*). The statistical features include autocorrelation (*acf*), partial autocorrelation (*pacf*), variance (*var*), and derivative (*de*). We calculate the features for each segment. Thus, we obtain the sequences of all the features. The function of each feature is calculated as follows:

Autocorrelation (lag = *h*):

$$acf_h = \frac{\sum_{t=1}^{T-h}(y_t - \bar{y})(y_{t+h} - \bar{y})}{\sum_{t=1}^{T-h}(y_t - \bar{y})^2} \tag{3.20}$$

Partial autocorrelation of *lag* = *h**, denoted by $pacf_{h*}$, is the autocorrelation between y_t and y_{t+h*} after removing the linear dependence of y_t on y_{t+h*-1} through y_{t+h*}

Variance:

$$var = \frac{1}{N_{var}} \sum_{i=1}^{N_{var}} \left(\frac{y}{\bar{y}} - 1 \right)^2 \tag{3.21}$$

Derivative:

$$de = \frac{y_{t_2} - y_{t_1}}{t_2 - t_1} \tag{3.22}$$

2. We predict the throughput of the next segment with the current segment using the AR model and HMM separately, and record the better model of the lower prediction error for each segment in *TRDS*(*v*). The root mean squared relative error, *RMSRE* is used which can be calculated as:

$$RMSRE = \sqrt{\frac{1}{n} \sum_{t=1}^{n} \left(\frac{y_t - \bar{y}_t}{y_t} \right)^2} \tag{3.23}$$

where y_t is the actual value and $\bar{y}_t$ is the predicted mean value at time *t*. For example, if one segment has the prediction *RMSRE* of 0.1 by the AR Model and 0.2 by the HMM, the segment will be classified into class *ar*, instead of class *hmm*, which means that the proper prediction model for this segment is the autoregressive Model. Thus, we have sequences of the prediction error of models.

3. From the above processes, we obtain the sequences of features and errors. Then, we adopt the supervised learning to develop the two-class classifier by SVM with a Gaussian kernel. Moreover, we use the 4-fold cross validation to decide the parameter for the classifier and choose the parameters with the lowest prediction error.

3.4.4.3 Throughput prediction

After dividing current segment into *ar* class or *hmm* class with SVM classifier, proper prediction model is utilized by HOAH to predict the distribution of throughput for next segment. The average processing time for one prediction is about 0.2 s using a computer with Intel Xeon CPU E3-1226 v3 and 8 GB memory.

1. Prediction with Autoregressive Model

If the segment is classified to *ar* class, HOAH will apply the autoregressive model to predict the future throughput. Inside the autoregressive model, the ADF test which is one of the famous unit root tests is employed. The flow of stationarity test by ADF is shown in Figure 3.18, where three models are adopted. If the history throughput data belong to stationary process, HOAH will predict throughput with the corresponding ADF test model. However, if the history throughput data does not belong to any stationary process, it will be differenced and the stationarity test will be repeated until the stationary state is achieved, and the corresponding prediction model is applied.

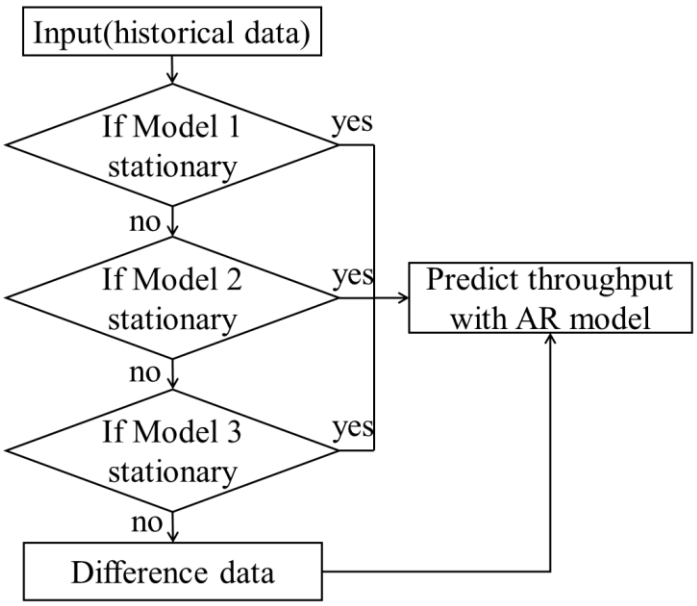


Figure 3.18 Stationarity test in the AR Model.

In the AR Model, the predicted throughput distribution follows the Gaussian distribution as expressed in Equation (3.24):

$$f\left(y_t,t\right)=\frac{1}{\sqrt{2\pi V(y_t)}}\exp\left[-\frac{\left(y_t-E(y_t)\right)^2}{2V(y_t)}\right] \tag{3.24}$$

where $E(y_t)$ and $V(y_t)$ denote the expectation and variance of the throughput series y_t. The expectation and variance are calculated by corresponding model from Equations (3.15)-(3.17) using the historical throughput data iteratively. For the situation with *lag* = 0, the general formulas for the models are shown as follows:

Model 1:

$$E\left(y_t\right)=a^t y_0 \tag{3.25}$$

$$V\left(y_t\right)=\frac{1-a^{2t}}{1-a^t}\sigma_\varepsilon{}^2 \tag{3.26}$$

Model 2:

$$E\left(y_t\right)=a^t y_0+\frac{1-a^t}{1-a}\cdot c \tag{3.27}$$

$$V\left(y_t\right)=\frac{1-a^{2t}}{1-a^t}\sigma_\varepsilon{}^2 \tag{3.28}$$

Model 3:

$$E\left(y_t\right)=a^t y_0+\frac{1}{1-a}\cdot dt-\frac{a\left(1-a^t\right)}{\left(1-a\right)^2}\cdot d+\frac{1-a^t}{1-a}\cdot c \tag{3.29}$$

$$V\left(y_t\right)=\frac{1-a^{2t}}{1-a^t}\sigma_\varepsilon{}^2 \tag{3.30}$$

where a, c, d and σ_ε are the model coefficients. y_0 is the last throughput of the historical data. Finally, the maximum/minimum predicted values ($E_{max/min}$) are calculated by using Equation (3.31):

$$E_{\max/\min}=E(y_t)\pm\alpha\cdot V(y_t) \tag{3.31}$$

where α can be changed to control the size of the predicted range.

2. Prediction with GMM-HMM

If the segment is categorized into *hmm* class, HOAH will adopt the GMM-HMM to predict the future throughput. We introduce the GMM-HMM in this

subsection. A basic idea of this model is shown in Figure 3.19. In the model, historical segments are used as training data. The GMM-HMM models throughput transitions from the training data. After completing the training process, the future prediction distribution of the throughput is calculated with the current segment. Then, the window slides to the next segment which consists of newly collected data with a step, such as 10 s. The historical segment is expanded with a longer length, and the model is updated.

First, in the training process, all historical throughput data are clustered by the GMM and classified into m number of Gaussian distribution expressed as Equation (3.32). Each Gaussian component can be calculated by using the standard expectation maximization algorithm. The Gaussian component is expressed as Equation (3.34) and has different weights (ω_i).

$$g(y \mid \omega_i, \mu_i, \sigma_i) = \sum_{i=1}^{m} \omega_i f(y \mid \mu_i, \sigma_i) \tag{3.32}$$

$$0 \le \omega_i \le 1, \sum_{i=1}^{m} \omega_i = 1 \tag{3.33}$$

$$f(y \mid \mu_i, \sigma_i) = \frac{1}{\sqrt{2\pi\sigma_i^2}} \exp\left[-\frac{(y-\mu_i)^2}{2\sigma_i^2} \right] \tag{3.34}$$

$$\psi(y,i) = \omega_i f(y \mid \mu_i, \sigma_i) \tag{3.35}$$

where ω_i ($0 \le i \le m$) denotes a mixture weight of i-th Gaussian density component in GMM. μ_i, σ_i, and $\psi(y,i)$ represent expectation, standard deviation, and the probability that the i-th Gaussian density component provides for the existence of the throughput data y. An example of data classification with GMM is shown in Figure 3.20.

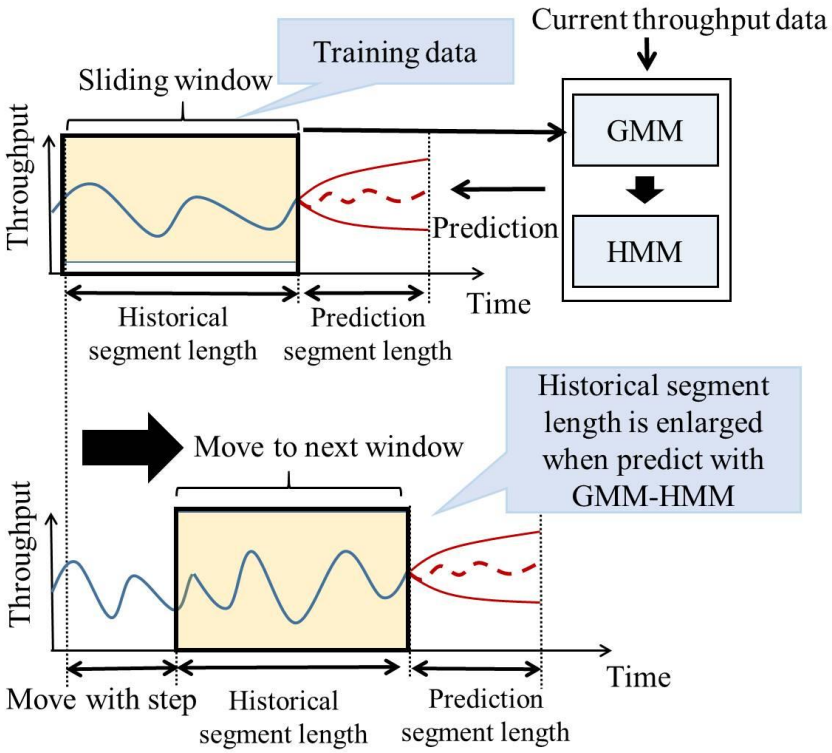


Figure 3.19 Basic concept of GMM-HMM.

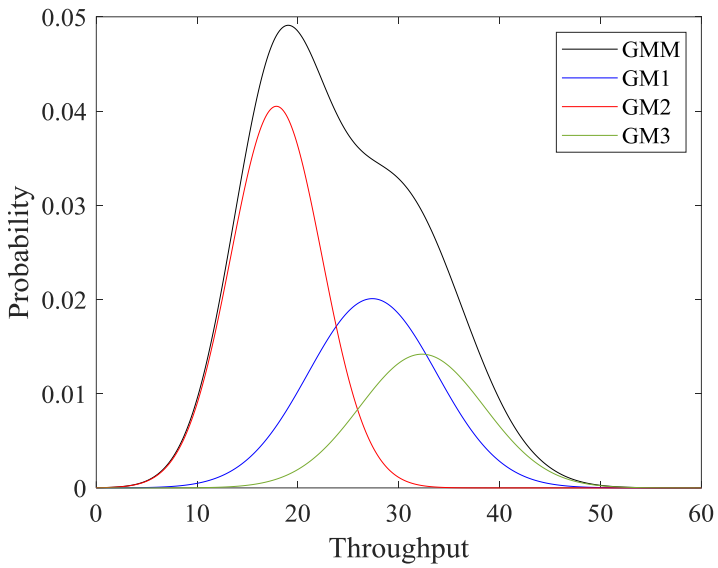


Figure 3.20 An example of clustered throughput data with GMM (m = 3).

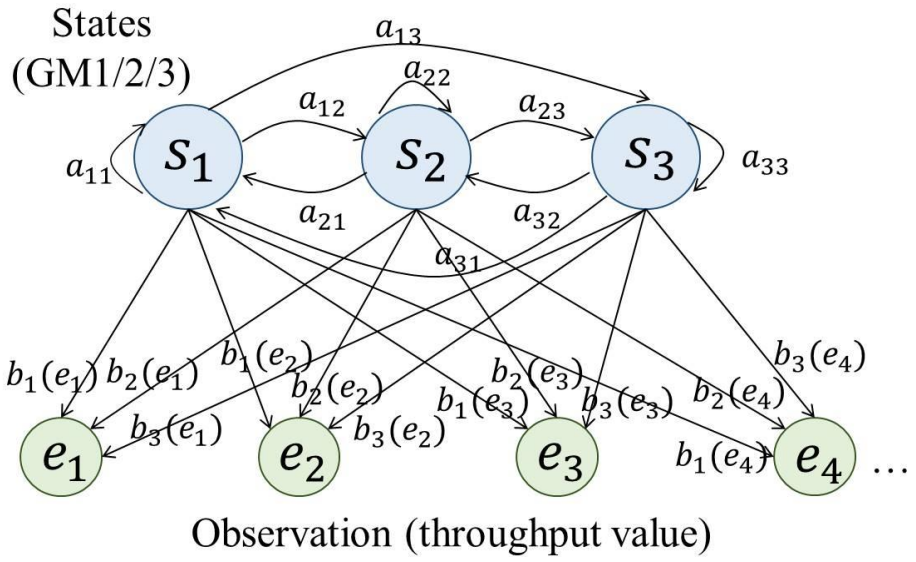


Figure 3.21 An example of throughput transition model with HMM. (s_i: use Gaussian component as a hidden state, e_k: historical throughput value as an observation).

Next, HOAH models the throughput transition pattern with HMM. After defining the states with GMM, the throughput data are considered as observations. The definition for the parameters of HMM is shown in Table 3.8, and an example of HMM is shown in Figure 3.21. After clustering all historical data with GMM, the Gaussian components (e.g., GM1/2/3 in Figure 3.20) are defined as hidden states (s_i) and the historical throughput value is defined as an observation (e_k) as shown in Figure 3.21. The hidden states represent the whole effect of factors, such as the number of customers sharing the same bottleneck in the channel. Then, we calculate the corresponding hidden state for each observation using Equation (3.35). Thus, we obtain an observation sequence and a hidden state sequence. By using these sequences, we calculate A_{tr} and B_{em} as shown in Table 3.8 and obtain hidden Markov model $\lambda = (\pi, A_{tr}, B_{em})$.

Table 3.8 Definition of parameters for HMM.

Parameter	Definition
m	Number of states in HMM
$O = (O_1, \ldots, O_T)$	Observation sequence
$Q = (Q_1, \ldots, Q_T)$	Hidden state sequence
$A_{tr} = \{a_{ij}\}$	A_{tr} is a state transition matrix, a_{ij} is the transition probability from the state i to j
$B_{em} = \{b_i(e_k)\}$	B_{em} is an observation emission matrix, $b_i(e_k)$ is when the state is i and the probability of observing e_k

Parameter	Definition
$\pi = \{\pi_i\}$	A_{tr} is a state transition matrix, a_{ij} is the transition probability from the state i to j
$\lambda = (\pi, A_{tr}, B_{em})$	HMM model with parameters
$s_i\ (1 \le i \le m)$	Current hidden state is i
e_k	Observation e_k (value inside throughput data range)

Finally, we predict the distribution of future throughput by utilizing the state transition probability in HMM as shown in Figure 3.22. We calculate the probability of each hidden state at the final time of historical data using the forward algorithm and assume that the probabilities are the initial state probability π from which we start to predict. Then, we calculate the probability of each hidden state $p_t(i)$ in future time with the current state probability $\pi = p_0(i)$ and state transition probability matrix $A_{tr} = \{a_{ij}\}$ as shown in Equation (3.36). After obtaining each hidden state probability, the expectation and variance are calculated by using the distribution of each hidden state (i.e. each Gaussian component) as shown in Equation (3.37) and (3.38). Thus, the predicted distribution can be obtained by using the calculated expectation and variance. Furthermore, the maximum/minimum predicted values $E_{max/min}$ are calculated by using Equation (3.31).

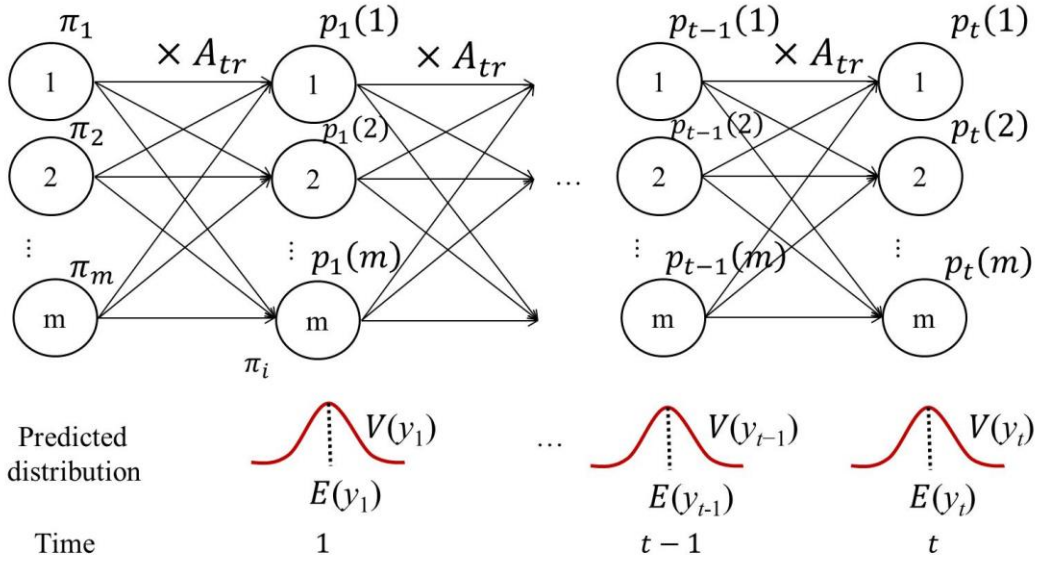


Figure 3.22 Process of prediction with HMM.

$$
\begin{aligned}
&[\pi_1, \pi_2, \ldots \pi_m] \times \begin{bmatrix} a_{11} & \ldots & a_{1m} \\ \vdots & \ddots & \vdots \\ a_{m1} & \ldots & a_{mm} \end{bmatrix} \\
&= \left[P(Q_1 = s_1 | \lambda), P(Q_1 = s_2 | \lambda), \ldots P(Q_1 = s_m | \lambda) \right] \\
&= \left[p_1(1), p_1(2), \ldots p_1(m) \right] \\
&\ldots \\
&\left[p_{t-1}(1), p_{t-1}(2), \ldots p_{t-1}(m) \right] \times \begin{bmatrix} a_{11} & \ldots & a_{1m} \\ \vdots & \ddots & \vdots \\ a_{m1} & \ldots & a_{mm} \end{bmatrix} \\
&= \left[P(Q_1 = s_1 | \lambda), P(Q_1 = s_2 | \lambda), \ldots P(Q_1 = s_m | \lambda) \right] \\
&= \left[p_t(1), p_t(2), \ldots p_t(m) \right]
\end{aligned}
\tag{3.36}
$$

$$
E(y_t) = \sum_{i=1}^{m} p_t(i) \cdot \mu_i \tag{3.37}
$$

$$
V(y_t) = \sum_{i=1}^{m} p_t(i) \cdot \sigma_i^2 + \sum_{i=1}^{m} p_t(i) \cdot \mu_i^2 - \mu_t^2 \tag{3.38}
$$

where π_i represents the initial state probability of the hidden state i, $p_t(i)$ represents the probability of state i at time t, a_{ij} represents the transition probability from state i to state j. μ_i is the expectation of the i-th Gaussian component. σ_i is the standard deviation of the i-th Gaussian component.

3.4.5 Evaluation

3.4.5.1 Experiment environment

Experiments are conducted to evaluate HOAH. We use the Android application developed by our lab to collect wireless throughput data in different scenarios, and we assume that the throughput data come from users who enjoy video streaming. In the experiment, the application observes and records throughput by downloading a 600 KB video segment per second from a content server. Since we assume MPEG-DASH, of which segments consist of 2-second video contents and rate control is carried out per segment, we calculate throughputs per second. Thus, we observe the LTE and Wi-Fi

throughput as shown in Figure 3.23. One user terminal which is Galaxy S4 (Android 4.2.2) connects with the server through LTE provided by the major public cellular carrier in Japan, or Wi-Fi access point 1 (AP1) that adopts 5 GHz IEEE 802.11n located in our laboratory, or Wi-Fi access point 2 (AP2) that uses 5 GHz IEEE 802.11n located in Nishi-waseda campus. The content server performs as an HTTP streaming server and is located in our laboratory. The DASH-JS framework is used [8]. To evaluate HOAH in scenarios of HSDPA, we use the public dataset in [48].

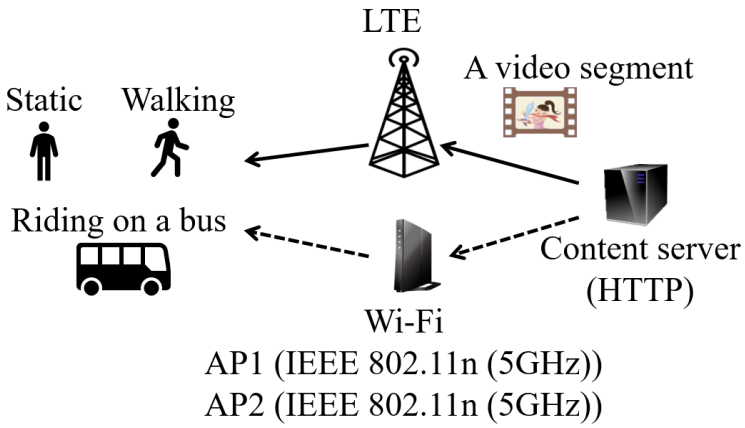


Figure 3.23 Experiment environment.

We observe wireless throughput in seven different scenarios as shown in Table 3.9. In this table, we change the wireless interface (LTE and Wi-Fi), the user moving mode (static, walking, and riding on a bus), and time (morning (10:20 a.m.), afternoon (15:00 p.m.) and evening (22:30 p.m.)). In static scenarios (LTE and Wi-Fi), a user stays static in the laboratory in the morning, afternoon and evening. In the walking case (LTE), the user walks from Nishi-waseda station to Zoshigaya station in the evening as shown in Figure 3.24(a). It takes approximately 23 min on a one-way trip. In the bus case (LTE), the user travels by a bus from Nishi-waseda campus to Waseda campus of Waseda University in the afternoon as shown in Figure 3.24(b). It takes approximately 10 min on a one-way trip. We conducted 5 trials for each scenario. The scenarios of HSDPA case are shown in [48].

Table 3.9 Experiment scenarios.

Scenario	Interface	User moving mode	Time	Evaluation time (s)	Number of days
Static case	LTE	Static	Afternoon 15:00 p.m.	2000	5

Scenario	Interface	User moving mode	Time	Evaluation time (s)	Number of days
Walk case	LTE	Walking	Evening 22:30 p.m.	1000	5
Bus case	LTE	Riding on a train	Afternoon 15:00 p.m.	600	5
Ferry	HSDPA	Moving ferry	unknown	1000	5
Train	HSDPA	Moving train	unknown	1500	5
Static case	Wi-Fi (AP1)	Static	Evening 22:30 p.m.	2000	5
Static case	Wi-Fi (AP2)	Static	Morning 10:20 a.m.	2000	5

Wireless throughput data are observed and recorded per second. We predict distributions of future throughput with HOAH and compare the prediction accuracy of the baseline methods demonstrated in [25] and [29].

(a) Walking case (b) Bus case

Figure 3.24 Maps of the two moving routes.

3.4.5.2 Evaluation metric

Although HOAH predicts the throughput distribution, we focus on the predicted mean value $\bar{y}_t$ of the throughput because from the viewpoint of accuracy evaluation, the comparison between the mean throughput and the actual throughput can be straightforward and comprehensive. Therefore, same as [80] we evaluate the accuracy of HOAH with relative prediction error R_t and *RMSRE* between the actual

value y_t and predicted mean value $\bar{y}_t$ at time t. *RMSRE* is calculated by Equation (3.23) and R_t is calculated by:

$$R_t = \frac{|y_t - \bar{y}_t|}{y_t} \tag{3.39}$$

The smaller *RMSRE* indicates the smaller prediction error a method has (i.e. higher accuracy), and vice versa. The variance of the predicted distribution is also a considerable parameter when applied in actual application.

For the parameter configuration of HOAH, we divide the dataset *DS*(v) into training data *TRDS*(v), validation data *VDS*(v), and test data *TDS*(v). HOAH obtains the model of proper parameters (e.g. number of states for HMM, order for the Autoregressive Model) with *TRDS*(v) and *VDS*(v), which could limit problems such as overfitting, to derive a more accurate model for HOAH. HOAH obtains the prediction accuracy of the testing data *TDS*(v) and compare with other methods. Inside HOAH, the SVM applies 4-fold cross validation to decide the value of the features such as the order of autocorrelation and partial autocorrelation.

3.4.5.3 Classification accuracy

Before introducing the results of the prediction accuracy, we analyze the classification result of the observed throughput by SVM in all scenarios. The classification accuracy of SVM in Table 3.10 is the probability that the method of AR or GMM-HMM with lower *RMSRE* is correctly chosen in different scenarios.

Table 3.10 Actual classification accuracy of seven scenarios.

Scenario	Accuracy	Number of prediction segment
Static case (LTE)	76%	25
Walk case (LTE)	93.10%	29
Bus case (LTE)	90%	10
Ferry case (HSDPA)	82.35%	17
Train case (HSDPA)	93.33%	15
Static case Wi-Fi (AP1)	88%	25
Static case Wi-Fi (AP2)	92%	25

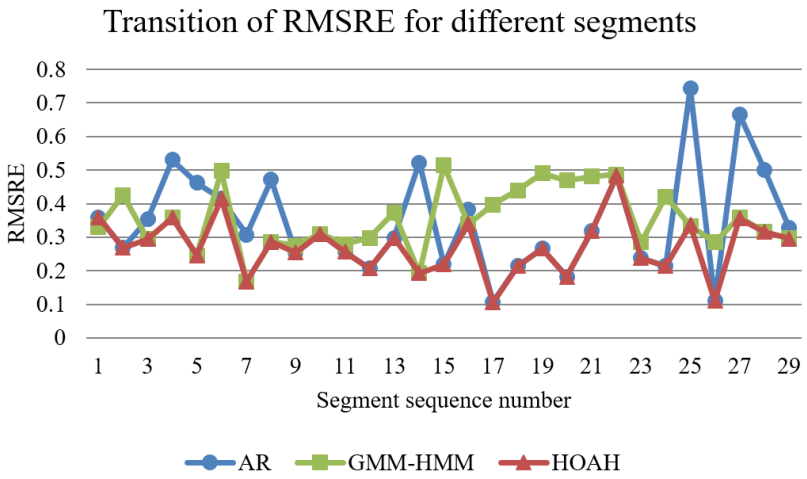


Figure 3.25 *RMSRE* for three methods in walk scenario (LTE).

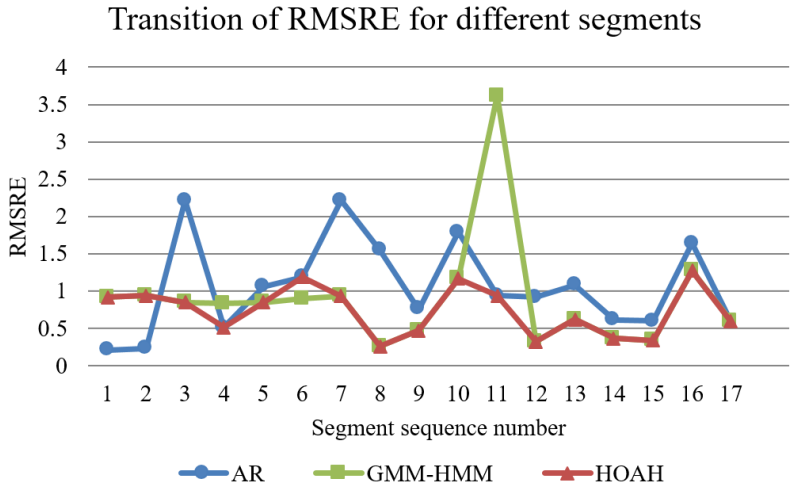


Figure 3.26 *RMSRE* for three methods in ferry scenario (HSDPA).

We can conclude from Table 3.10 that the actual classification accuracy of SVM is very high for all the scenarios. We take the walk scenario (LTE) and ferry scenario (HSDPA) as examples. Figure 3.25 shows the prediction error of different segments using three methods in the walk scenario and Figure 3.26 shows corresponding results in ferry scenario. We can see that the HOAH chooses the correct methods for 93.10% of all segments in walk scenario and can choose the correct method for 82.35% of all segments in ferry scenario.

3.4.5.4 Prediction accuracy

We evaluate and compare the prediction accuracy with CDF of R_t and *RMSRE* of seven methods, which are HOAH, AR, GMM-HMM, harmonic mean (HM), last sample (LS), moving average (MA) and stochastic model (Stochastic), in seven scenarios. Evaluation parameters are shown in Table 3.11. Moreover, the results are calculated by the average of multi-segment prediction results of all the trials in each scenario.

Table 3.11 Evaluation parameters.

Scenario	Historical segment length (s)	Prediction segment length (s)	Number of prediction segments
Static case (LTE)	30	20	25
Walk case (LTE)	30	20	29
Bus case (LTE)	30	20	10
Ferry case (HSDPA)	30	20	17
Train case (HSDPA)	30	20	15
Static case Wi-Fi (AP1)	30	20	25
Static case Wi-Fi (AP2)	30	20	25

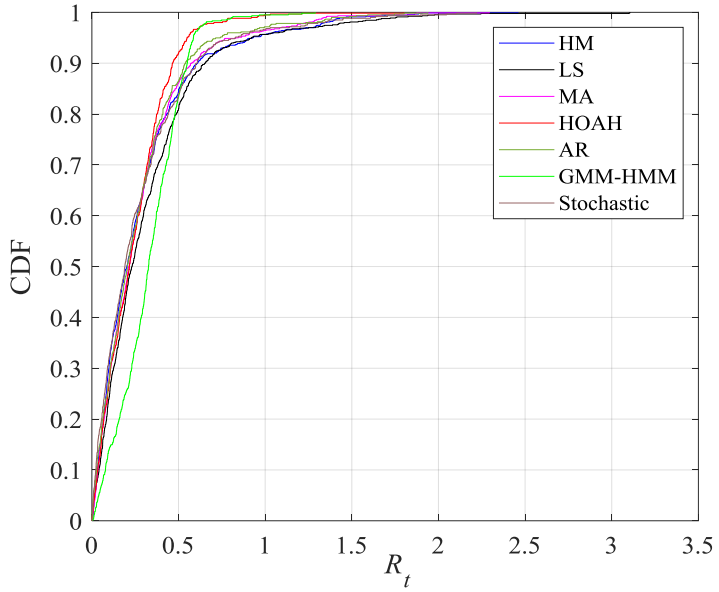


Figure 3.27 CDF of R_t in walk scenario using different methods when historical segment length is 30 s and prediction segment length is 20 s.

The walk scenario of LTE is taken as an example shown in Figure 3.27. The CDF curves show that 92% of the total predicted data with HOAH has the R_t that is smaller than 0.5, and the percentage of which is 6% larger than other methods. Moreover, R_t of 98.8% data is smaller than 0.783. These indicate that HOAH is more effective than other methods.

Figure 3.28 depicts the *RMSRE* of prediction with all methods in various scenarios. It can be seen that AR and GMM-HMM models outperform other conventional methods such as HM, LS and MA in most cases, demonstrating the feasibility of choosing these two methods for switching. Basically the AR model performs better in static cases where the throughput is more like stationary process. In the moving cases, GMM-HMM performs better because HMM can model the process with sharp change as a statistical transition between states. Table 3.12 shows the percentage of the decreased prediction error of HOAH compared with other methods. We can conclude that HOAH is effective for both static user scenarios and moving user scenarios. Especially for moving user scenarios, HOAH can reduce the prediction error by maximum 55.95% in the ferry scenario using HSDPA as shown in Table 3.12.

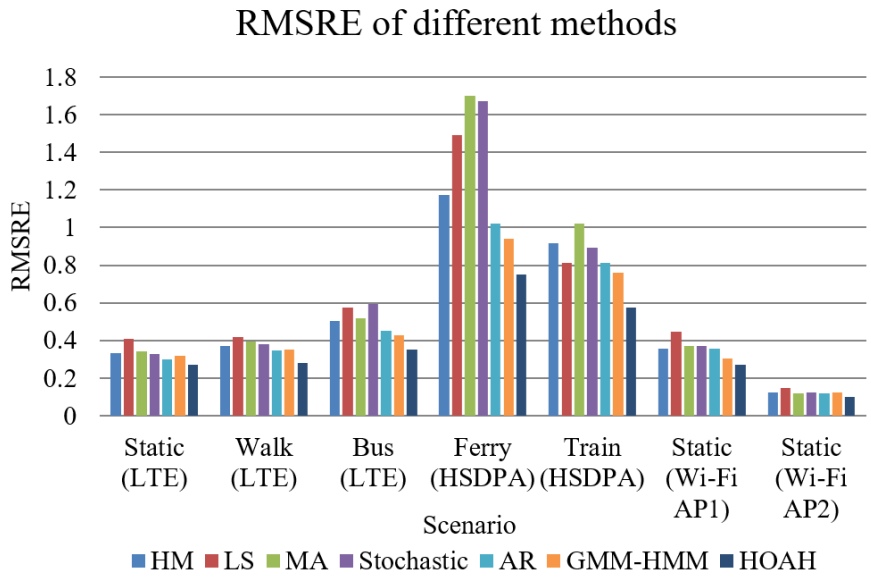


Figure 3.28 Average *RMSRE* results in all scenarios.

Table 3.12 Ratio of decreased prediction error when using HOAH compared with other methods.

Scenario	HM	LS	MA	Stochastic	AR	GMM-HMM
Static case (LTE)	18.09%	33.12%	20.25%	16.36%	8.86%	14.71%
Walk case (LTE)	23.98%	32.72%	28.47%	26.16%	18.09%	19.98%
Bus case (LTE)	30.26%	38.54%	31.58%	40.80%	21.58%	17.65%
Ferry case (HSDPA)	36.06%	49.78%	55.95%	55.16%	26.67%	20.40%
Train case (HSDPA)	37.38%	29.47%	43.73%	35.85%	29.54%	24.33%
Static case Wi-Fi (AP1)	24.25%	39.09%	26.38%	26.70%	24.01%	10.12%
Static case Wi-Fi (AP2)	18.43%	31.43%	14.69%	18.22%	14.38%	17.91%

3.4.5.5 Analysis of the effect of parameters

Next, we evaluate the effect of different historical segment length and prediction segment length on the prediction accuracy. In the evaluation, we take the walk (LTE) and ferry scenarios as examples.

1. Historical segment length

To evaluate the effect of historical segment length, we assign 20 s to the prediction segment length and compare the prediction error of HOAH with other baseline methods. The evaluation parameters are shown in Table 3.13, and the results are calculated by averaging the results of 29 or 17 prediction segments.

Table 3.13 Evaluation parameters.

Scenario	Historical segment length (s)	Prediction segment length(s)	Number of prediction segments
Walk case (LTE)	30-100	20	29
Ferry case (HSDPA)	30-100	20	17

Figure 3.29 shows the prediction error of different methods in walk scenario and Figure 3.30 shows the corresponding results in ferry scenario. From both figures, we can conclude that HOAH indicates lower prediction error than other baseline methods regardless of the historical segment length (30-100 s). Moreover, all methods seem to be not affected by the change of historical segment length.

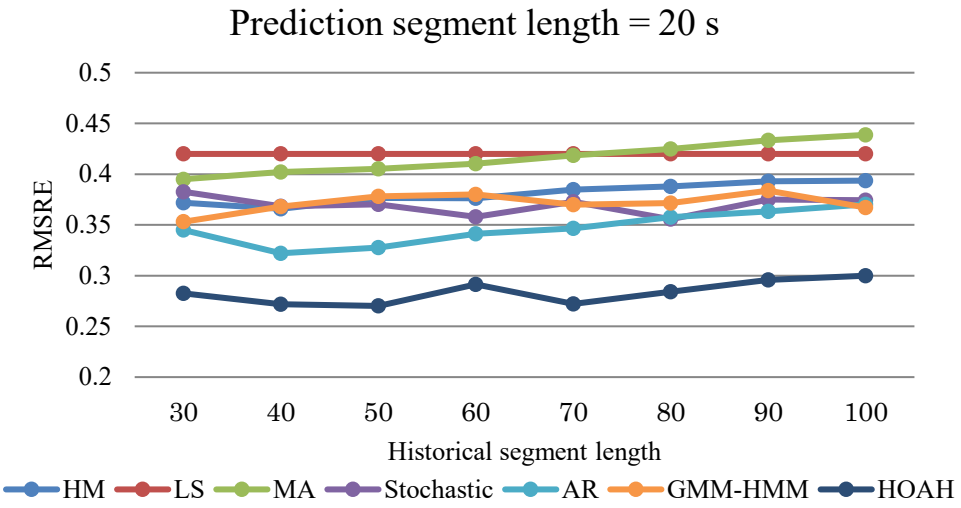


Figure 3.29 The influence of different historical segment length on average *RMSRE* in walk scenario (LTE).

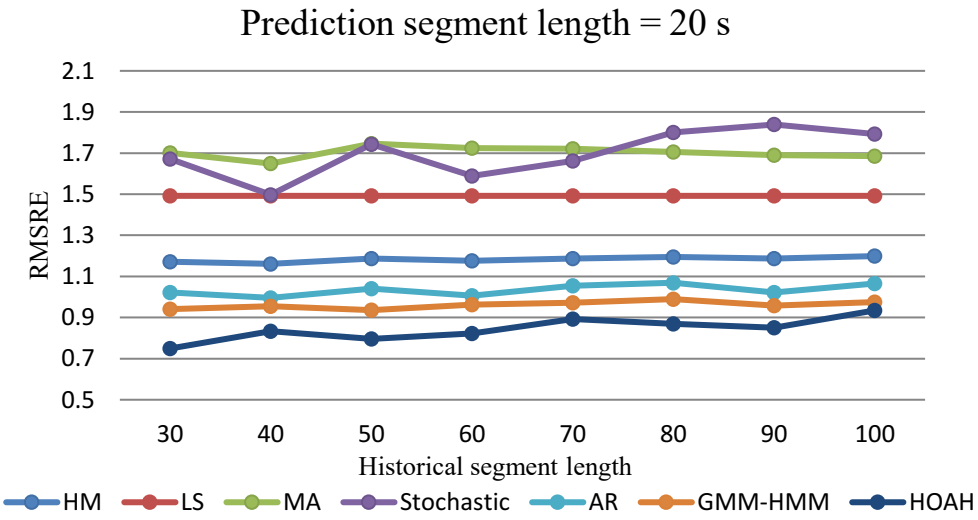


Figure 3.30 The influence of different historical segment length on average *RMSRE* in ferry scenario (HSDPA).

2. Prediction segment length

We also evaluate the prediction error of different prediction segment length. In the evaluation, we fix the historical segment length with 50 s and change the prediction segment length from 10 s to 100 s as shown in Table 3.14. The average *RMSRE* of different methods in the walk (LTE) scenario are shown in Figure 3.31, and the corresponding results in ferry scenario are shown in Figure 3.32.

Table 3.14 Evaluation parameters.

Scenario	Historical segment length (s)	Prediction segment length(s)	Number of prediction segments
Walk case (LTE)	50	10-100	5-58
Ferry case (HSDPA)	50	10-100	4-34

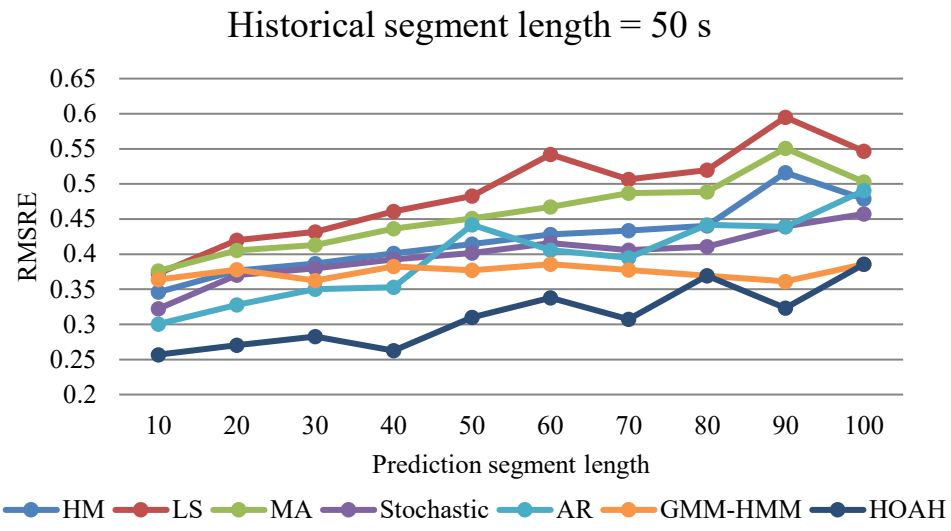


Figure 3.31 The influence of different prediction segment length on average *RMSRE* in the walk scenario (LTE).

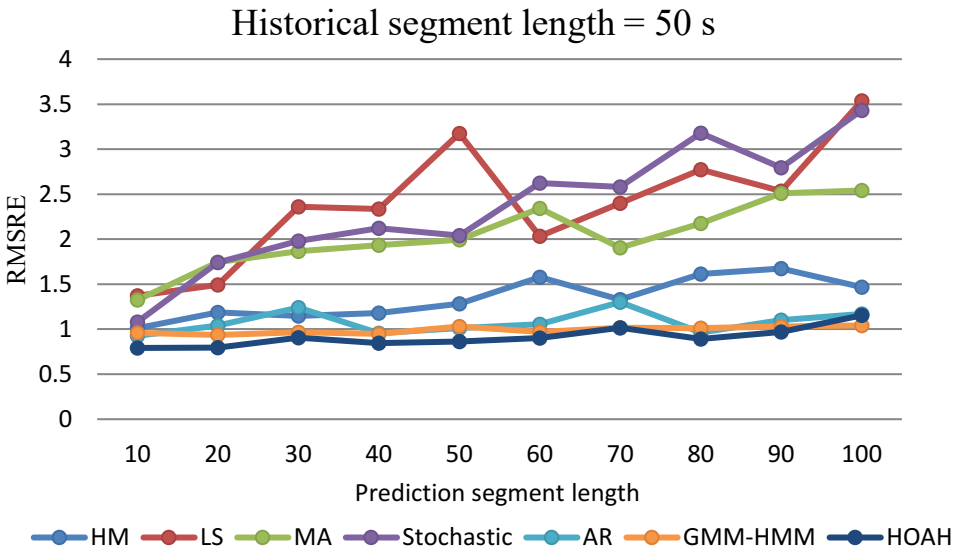


Figure 3.32 The influence of different prediction segment length on average *RMSRE* in the ferry scenario (HSDPA).

Both Figure 3.31 and Figure 3.32 imply that HOAH has lower prediction error than baseline methods. With the increase of prediction segment length, data in future time tend to have different performance with historical segment data. And future data become harder to be predicted because they may have less relationship with historical data. Therefore, all methods show larger *RMSRE*, which means larger prediction error with the increase of prediction segment length.

3.5 Summary

In Chapter 3, the time series analysis by statistics and machine learning techniques are utilized and studied for throughput prediction. A history-based throughput prediction method utilizing hidden Markov model for real time communication in mobile networks is proposed, which could judge data fluctuation of current time and adopt proper methods to predict future throughput for each condition. Then, we conduct experiment in different situations with different time and user mobility in mobile network to evaluate the proposed method. Results show that, compared with the conventional prediction methods, including linear regression, locally weighted linear regression and stochastic model, our proposed approach could achieve higher accuracy not only in static user situations but also in moving user situations. Based on this method, an advanced model called the hybrid prediction with the autoregressive model and hidden Markov model, HOAH is proposed. The novel model combines the autoregressive model and hidden Markov model by SVM classifier. HOAH could predict throughput with the proper model during the whole session by switching between two models. Moreover, it takes user moving mode into consideration and can decrease prediction error in various scenarios. We conduct experiments to evaluate the method and compare it with conventional methods. Experiments showed that HOAH could decrease the prediction error by 55.95% and achieve higher accuracy than conventional methods.

In the future, the inclusion of human moving behaviors into the classification step will be considered and integrated into our prediction method. We will also continue to improve the prediction accuracy and apply HOAH to control the bitrate adaptively for mobile network communication to guarantee the QoS/QoE for video delivery.

4 TCP throughput prediction using neural networks[†]

*Throughput prediction is essential for ensuring high quality of service and quality of experience for video streaming transmissions. However, current methods are incapable of accurately predicting throughput in mobile networks, especially for moving user scenarios. Therefore, we propose a TCP throughput prediction method: **T**hro**u**ghput prediction based on L**ST**M (TRUST) using machine learning for mobile networks. TRUST has two stages: user movement pattern identification and throughput prediction. In the prediction stage, the long short-term memory (LSTM) model is employed for TCP throughput prediction. TRUST takes all the communication quality factors, sensor data and scenario information into consideration. Field experiments are conducted to evaluate TRUST in various scenarios. The results indicate that TRUST can predict future throughput with higher accuracy than the conventional methods, which decreases the throughput prediction error by maximum 44% under the moving bus scenario.*

4.1 Neural networks

The neural networks use units to model the neurons in a biological brain and communicate with each other by transmit real numbers. The output number of artificial neuron is calculated by the non-linear function of the sum of its inputs and with related adjustable weights. Generally neural networks have many layers, and different layers do various calculation to the inputs. The output of the last layer is the final result. Neural networks are widely used in many areas such as signal processing, computer vision, speech recognition, and AI related products.

4.2 User movement pattern recognition

Throughput data in different scenarios have different characteristics, thus it is important to identify and classify data of same scenario. User movement pattern is one important factor of the scenario information, so recognize user movement pattern is an important task.

[†]This chapter is adapted from the work published in [90].

4.2.1 Classification

The basic process for movement pattern recognition is classifying the data to the corresponding class. Thus, the problem can be generally considered as classification problem. Classification is to find the class that an observation belongs to, which is a topic of pattern recognition. In the area of machine learning, supervised learning is adopted to construct the classifier, which can be used to classify the new input to the correct category. In supervised learning, the characteristics of data are used as features for the basis of model construction. Different kinds of methods are considered for classification problem. Machine learning techniques are useful tools for this topic. Thus, in the next session, we will introduce some machine learning methods that being used in this thesis.

4.2.2 Machine learning techniques

There are many machine learning techniques can be utilized to solve the problem of classification, such as Naïve Bayes classifier, support vector machine, k-nearest neighbor and neural networks. In this thesis, we only try some of the methods and give a discussion later. The machine learning techniques are k-nearest neighbors (*k*-NN), support vector machine (SVM), and random forest (RF).

SVM adopts supervised learning to model and solve the problem of classification and regression. It is popular to work as classifier not only for linear classification, but also for nonlinear classification. By using kernel function, SVM can map the feature data to a high-dimensional feature space, and construct a hyper-plane to solve the nonlinear classification problem. The common kernel function utilized are:

Polynomial function:

$$k(\vec{x}_i, \vec{x}_j) = k(\vec{x}_i \cdot \vec{x}_j)^d \tag{4.1}$$

Gaussian radial basis function:

$$k(\vec{x}_i, \vec{x}_j) = \exp\left(-\frac{1}{2\sigma^2} \left\| \vec{x}_i - \vec{x}_j \right\|^2 \right), \quad \text{for } \frac{1}{2\sigma^2} > 0 \tag{4.2}$$

k-NN is an algorithm that classify the new event based on similarity measurement with the *k* nearest neighbors. Supervised learning is adopted to construct the *k*-NN model. By calculating the vote of the nearest *k* neighbors, the classifier output

the class that the new event belongs to. The choice of *k* has a large effect on the classification result. The cross validation can be used to help chose the best value for *k*. RF is a kind of ensemble learning that can be used for classification. Supervised learning is also adopted to generate the tree structure for classification.

4.2.3 Experiment result

In our former research [81], [82], *k*-NN, SVM and RF are adopted for user movement pattern recognition. First, the data of throughput, RSSI, and Cell ID are collected in the following scenarios: a) One user utilizes one mobile phone to collect the data at 15:00 with different moving patterns; b) One user utilizes three mobile phones to collect data during 5:00-23:00 with different moving patterns.

Table 4.1 Dataset definitions.

Data set	Throughput	RSSI	Cell ID	Accelerometer
1				○
2	○			
3		○		
4			○	
5	○	○		
6		○	○	
7	○		○	
8	○	○	○	
9	○	○	○	○

The datasets 1-9 are shown in Table 4.1, from which we can observe that the combination of four features are developed. Scikit-learn [83] is adopted to conduct the experiments. 75% data are used for training the model, and 25% are used as test data. Experiment results for the two scenarios are shown in Figure 4.1 and Figure 4.2. The results indicate that the communication quality can be adopted to identify transportation mode and the results of the three machine learning methods show little difference. Moreover, Cell ID is the most effective factor as the experimental area is fixed in the evaluations for each scenario of specific movement pattern. The contribution of RSSI is higher than throughput because the throughput is affected by many factors such as RSSI and the condition of the base station being utilized which is complex. While the factors which affect RSSI simply includes moving speed and distance between the user and the signal source.

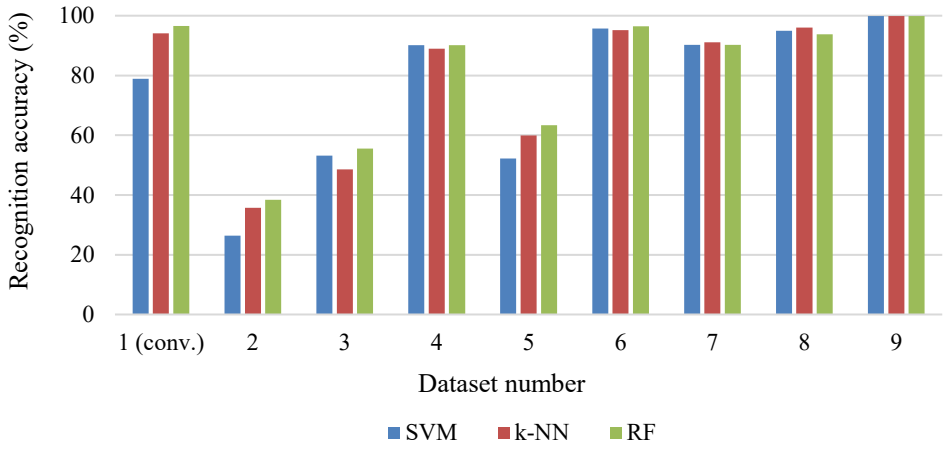


Figure 4.1 Comparison of the classification results using different datasets and methods for scenario a. [81]

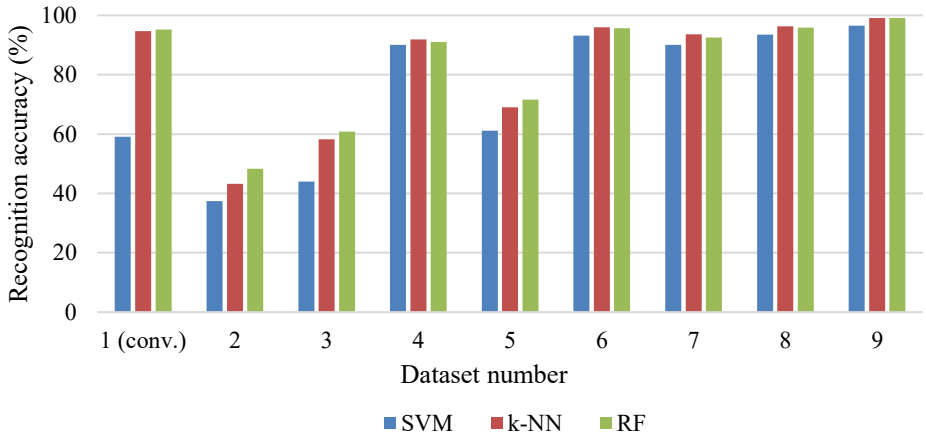


Figure 4.2 Comparison of the classification results using different datasets and methods for scenario b. [81]

4.3 Throughput prediction using TRUST

4.3.1 Data acquisition and analysis

4.3.1.1 Data acquisition setup

We use the Android application developed by our laboratory to record the data, which are assumed from video streaming being enjoyed by a user. In the application, the throughput is observed by downloading video segments with the size of 500-1200 KB

to user terminal (Galaxy S4 of Android 4.2.2) from the content server located in our laboratory via Long-term evolution (LTE) provided by the major public cellular carrier in Japan. DASH-JS [8] is adopted and the content server serves as an HTTP streaming server. The throughput, RSSI, Cell ID, time, and location, as well as sensor data are collected every second.

4.3.1.2 Characteristics of the measured data

Figure 4.3 presents examples of measured actual throughput in walk scenario and static scenario. The throughput data are derived from two days with same time (10:00 PM) and same moving route or position. A comparison of throughputs from walk scenario of two days elucidates that the throughputs show similar behavior in the same scenario. By contrast, comparing throughputs from walk and static scenarios, the throughputs show different behaviors in different scenarios.

Figure 4.4 shows the throughput and corresponding RSSI and Cell ID in walk and static scenarios. It can be observed that the throughput, RSSI, and Cell ID in a moving user scenario have larger fluctuation and change more frequently than those of a static user scenario. Since the throughput behaves in a different manner in different scenarios, the prediction should be considered separately for all scenarios. Meanwhile, as other measured data such as RSSI have relationship with the throughput, they should also be included in the prediction.

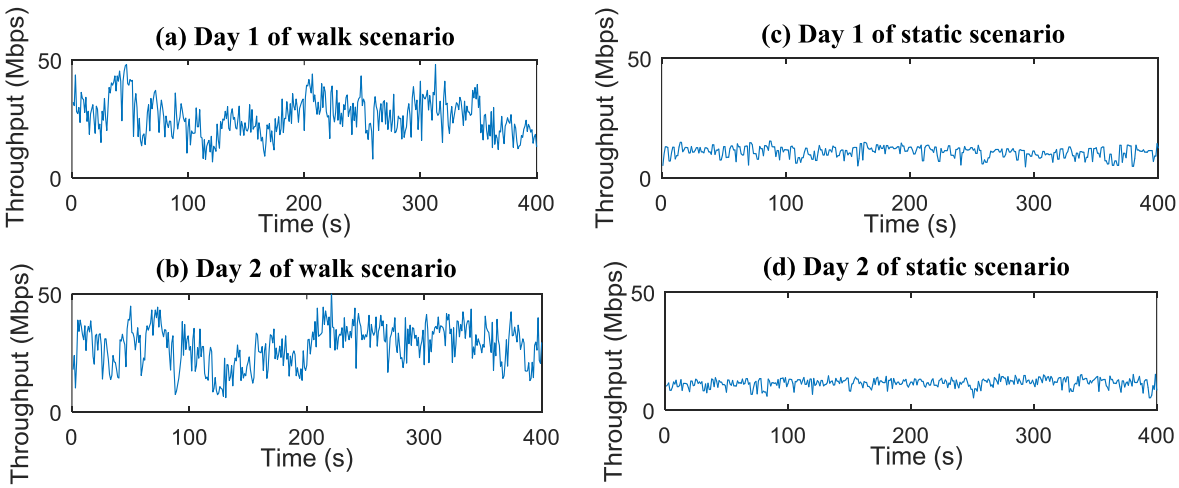


Figure 4.3 Throughput data of walk scenario from (a) Day 1, (b) Day 2 and static scenario from (c) Day 1, (d) Day 2.

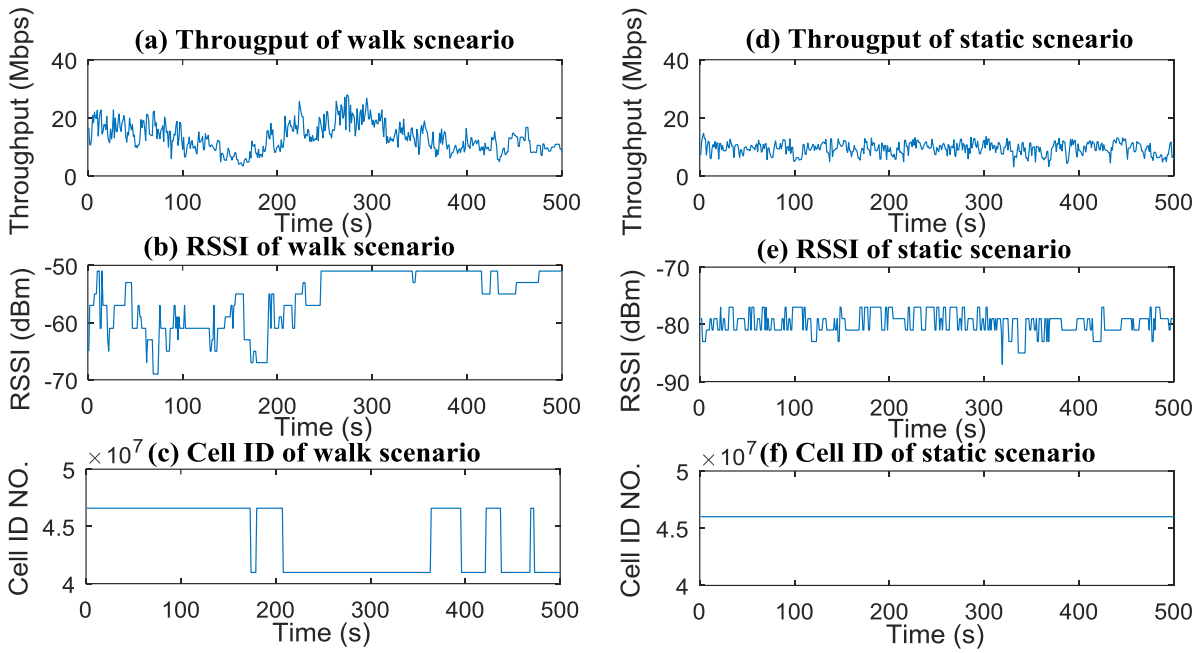


Figure 4.4 Communication quality factors in walk and static scenario: (a), (d) throughput, (b), (e) RSSI, (c), (f) Cell ID.

4.3.1.3 Preprocessing of the measured data

As shown in Figure 4.4, since the orders of the measured data are different, it is essential to process the data before applying them to the prediction method. In addition to the data shown above, a couple of sensor data can also be captured simultaneously during usage. To change them into similar order, the following preprocessing procedures are carried out:

1) Throughput (TH): Normalize the data TH with the maximum value (TH_{max}) in the historical measurements as: $TH_{new} = TH/TH_{max}$.

2) RSSI: As the value is basically in the range (-90,-50), the values are rescaled to (0, 1) as: $RSSI_{new} = (RSSI + 90)/40$.

3) Cell ID: The ID value in this work is basically in the range (3×108, 5×108), the values are rescaled as: $Cell\ ID_{new} = (Cell\ ID – 3×108)/2×108$.

4) The device collects position information such as latitude, longitude and orientations, which are saved as degree. The range of latitude is from (-90, 90) and it is rescaled by $LAT_{new} = (LAT + 90)/180$. The longitude is from (-180, 180) and it is rescaled by $LONG_{new} = (LONG + 180)/360$. The orientations are from (0, 360) and they are rescaled by: $ORI_{new} = ORI/360$.

5) The accelerations are also collected. These data are rescaled by dividing the acceleration of gravity: $ACC_{new} = ACC/9.8$.

Besides the measured data, the mean value, minimum value, maximum value, variance, and standard deviation of the throughput are calculated and considered as additional inputs for TCP throughput prediction.

4.3.2 Throughput prediction methodology

4.3.2.1 The structure of TRUST

TRUST, a two-stage machine learning-based TCP throughput prediction method is proposed in this paper. As shown in Figure 4.5, the throughput, RSSI, Cell ID, time, and location, as well as other sensor data are collected. The data are assumed from a user enjoying mobile services such as video streaming in various scenarios. Each kind of data is regarded as an input feature for prediction. First, the features are used to identify the user movement pattern. Then, prediction is performed, and finally, the predicted throughput can be used for smart rate control.

4.3.2.2 User movement pattern identification

In [81], we proposed a user movement pattern identification method using communication quality factors and sensor data via machine learning. Experiments were conducted and the results demonstrated that the user movement pattern can be recognized with high accuracy. In this paper, the movement identification is employed as the first stage of TRUST. According to the identification results, the prediction is conducted using the corresponding LSTM model which is trained with datasets from the same scenario.

4.3.2.3 Throughput prediction mechanism

Recurrent neural network (RNN) is widely used for problems such as natural language processing in which the next word is heavily dependent on the previous words [84], [85]. As the throughput data in the future are considered to have close relationship with the historical data, we adopt the RNN technique to deal with this time series prediction challenge [86]. Compared with the basic RNN model, LSTM can eliminate exploding and vanishing gradient problems [86]. Therefore, it can be used to address both long- and short-term prediction problems. In this paper, the neural network (NN)

model based on the LSTM structure is constructed. Figure 4.6 illustrates the architecture of the proposed model. $\mathbf{H}^{l}_{t'}$ represents the hidden states of layer l at timestep t'. The number of hidden states is defined as d and the number of layers is defined as L. d ranges from 2 to 100 and L ranges from 1 to 3. We define the input of the network as $\mathbf{x}_{t'}$ and the output as y_t. In the proposed model, the input $\mathbf{x}_{t'}$ are the historical throughput and other features such as RSSI and Cell ID, whereas the output y_t is a scalar only containing the future throughput. The size of $\mathbf{x}_{t'}$ is $N\times1$ where N denotes the number of features being used. n is the input length and m is the prediction length on time dimension. Here, n and m are not necessarily identical. m can be smaller than n. $\mathbf{W}$ and $\mathbf{U}$ are model parameter matrices. The task is to determine a NN model that approximates the relationship between $(\mathbf{x}_1, \mathbf{x}_2,\ldots, \mathbf{x}_n)$ and $(y_1, y_2,\ldots, y_m)$.

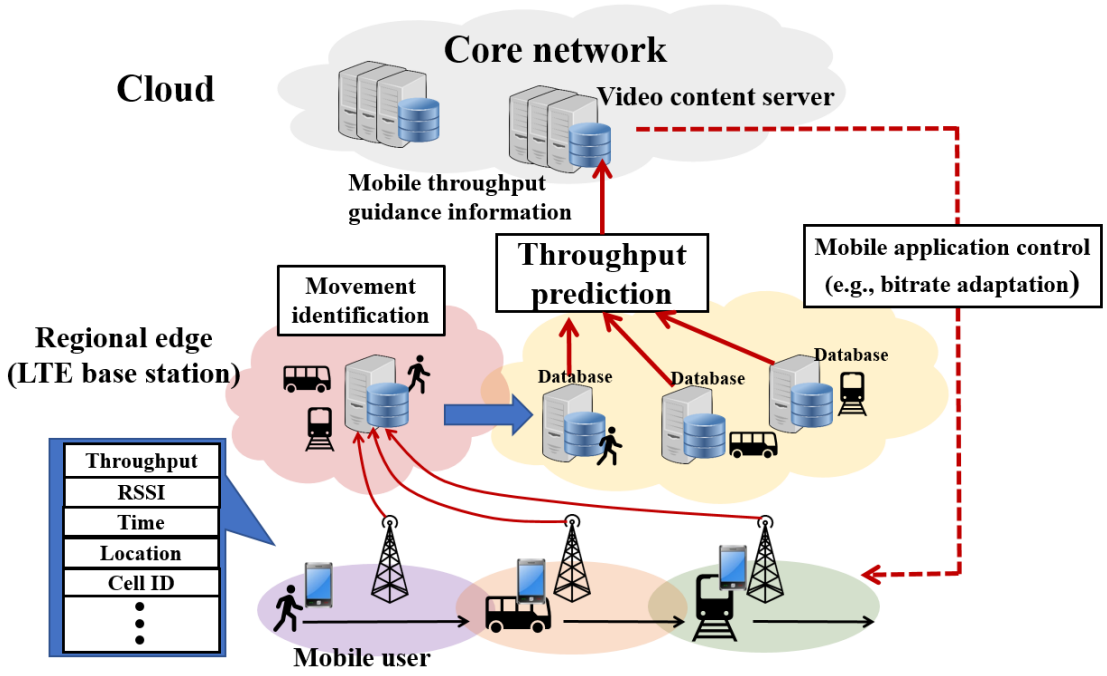


Figure 4.5 Schematic diagram of TRUST.

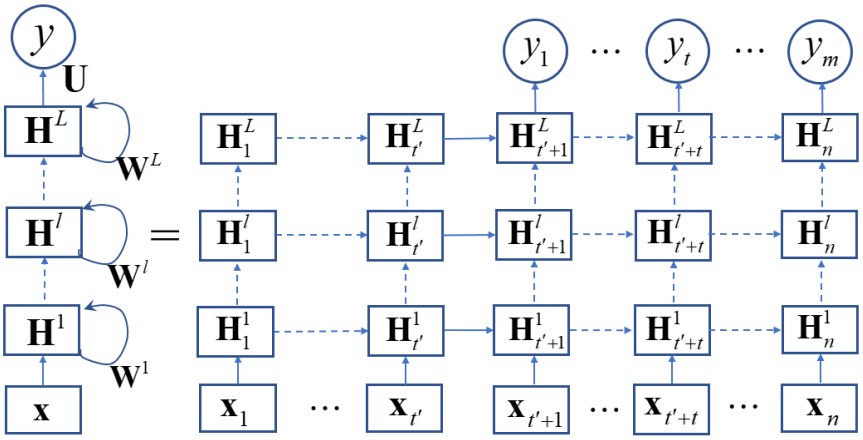


Figure 4.6 Structure of the proposed prediction model.

The LSTM cell, whose structure is shown in Figure 4.7, forms the basic cell in each layer. In the LSTM cell, a memory cell $\mathbf{C}_{t'}$ is introduced to hold the information from the past. Three gates, forget gate $\boldsymbol{f}$, input gate $\boldsymbol{i}$, and output gate $\boldsymbol{o}$, are used to manipulate the information flow. The formulas to update gates, memory cell, and hidden states are as follows:

$$\begin{pmatrix} \boldsymbol{i} \\ \boldsymbol{f} \\ \boldsymbol{o} \\ \boldsymbol{g} \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ tanh \end{pmatrix} \left(\mathbf{W} \cdot \begin{pmatrix} \mathbf{H}_{t'-1} \\ \mathbf{x}_{t'} \end{pmatrix} + \mathbf{b} \right) \tag{4.3}$$

$$\mathbf{C}_{t'} = \boldsymbol{f} \odot \mathbf{C}_{t'-1} + \boldsymbol{i} \odot \boldsymbol{g} \tag{4.4}$$

$$\mathbf{H}_{t'} = \boldsymbol{o} \odot tanh(\mathbf{C}_{t'}) \tag{4.5}$$

where $\mathbf{W}$ is the weight matrix, $\mathbf{b}$ is the bias vector, σ is the sigmoid function, and $\odot$ is the point-wide multiplication. The sizes of the gates and memory cell are the same as the hidden states. Therefore, $\mathbf{W}$ is a matrix with the size of $(4d)\times(d+N)$. The size of $\mathbf{b}$ is $(4d)\times 1$. Note that Figure 4.7 is the representation of an LSTM cell in the first hidden layer. If the LSTM cell is in the deeper layer, $\mathbf{x}_{t'}$ should be replaced with the hidden states from the former layer and the size of $\mathbf{W}$ becomes $(4d)\times(2d)$.

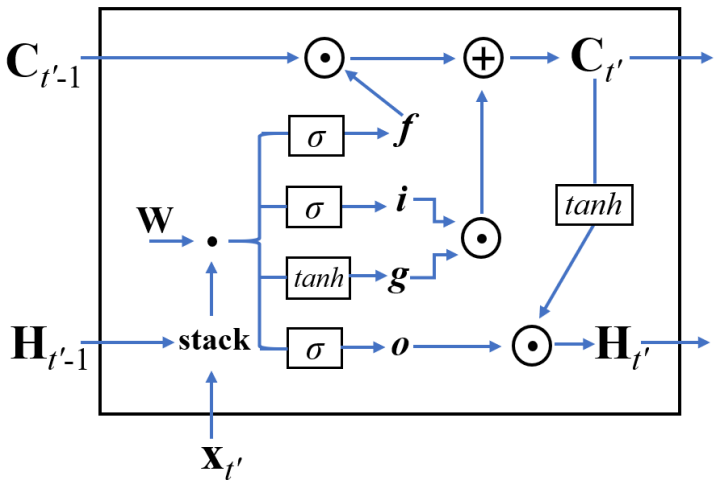


Figure 4.7 Structure of the LSTM cell.

While updating, the forget gate $\boldsymbol{f}$ determines how much the former information to be retained based on the input data. Meanwhile, the input gate $\boldsymbol{i}$ determines how much new information to be added. By adding the gated former memory cell $\mathbf{C}_{t'-1}$ and the new information, the memory cell for the current timestep $\mathbf{C}_{t'}$ is calculated. Then the current memory cell and the output gate are combined to calculate the current hidden states.

For throughput prediction, NN is used as a regression model in which the output should be certain numbers instead of classifications. Therefore, the output y is calculated as follows:

$$y = \mathbf{U} \cdot \mathbf{H}^{L} + b_o \tag{4.6}$$

where $\mathbf{U}$ is the weight matrix and b_o is the corresponding bias for output layer.

In the model training for LSTM, each training example is a (($\mathbf{x}_1$, $\mathbf{x}_2$,…, $\mathbf{x}_n$), (y_1, y_2,…, y_m)) pair. The mean squared error (MSE) between the predicted output $\hat{y}_t$ and real output y_t is used as the cost function which is written as:

$$J(\mathbf{W}^{1},...,\mathbf{W}^{L},\mathbf{b},\mathbf{U},b_o) = \frac{1}{s \cdot m} \sum_{k=1}^{s} \sum_{t=1}^{m} \left(\hat{y}_t^k - y_t^k \right)^2 \tag{4.7}$$

where k is the example index and s is the number of training examples. Figure 4.8 illustrates the preparation of the training datasets. The procedure is as follows:

1) The features' values in former n timesteps are extracted as the input ($\mathbf{x}_1$,

$\mathbf{x}_2,\ldots, \mathbf{x}_n$) of the training example.

2) The latter m timesteps of the throughput (Feature1) are extracted as the output ($y_1, y_2,\ldots,y_m$).

3) Putting ($\mathbf{x}_1, \mathbf{x}_2,\ldots, \mathbf{x}_n$) and ($y_1, y_2,\ldots,y_m$) pair together, a training example is completed.

4) The time window slides in m timesteps, and step 1 to step 3 are repeated until the end of the current data file.

5) The measured data file is changed, and step 1 to step 4 are repeated.

The training examples are fed into the LSTM model and the model parameters are optimized to minimize the prediction error defined in Equation (4.7). The optimization method used here is Adam [87]. The learning rates range from 0.0001 to 0.001 for different networks. The maximum number of training epoch is 2000. A patience factor P and a minimum improvement factor δ are defined to terminate the training. δ is the improvement of J between two epochs. If there are continuous P epochs where the improvement is less than δ, the training will be terminated. Here, P is set as 100 and δ is set as 0.01%.

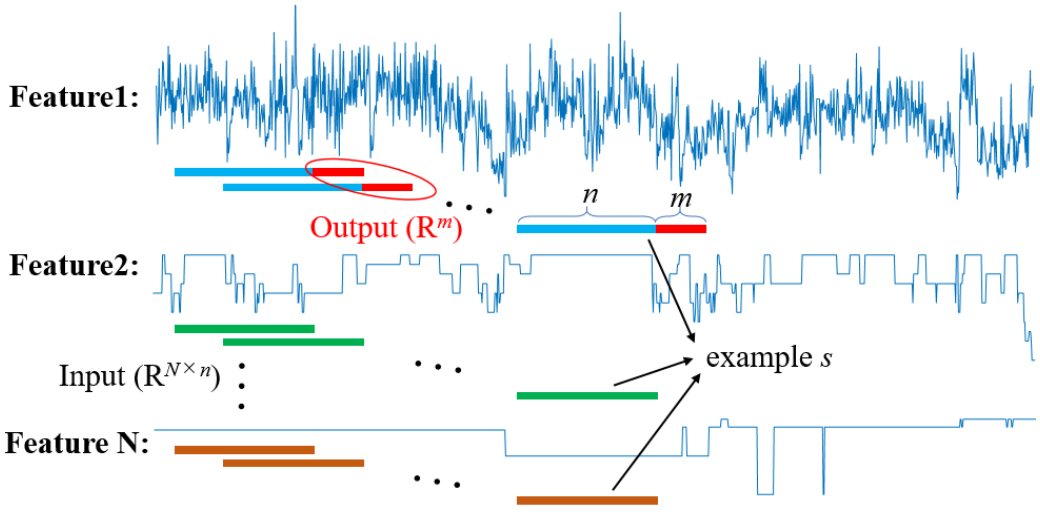


Figure 4.8 Preparation of the training datasets for the LSTM model.

Algorithm 1 briefly summarizes the entire procedure of TRUST. When the prediction is to be carried out, the historical data in the former n seconds are preprocessed. Then the movement pattern is identified. According to the identification result, the throughput in the future m seconds is predicted using the corresponding LSTM model. Note that m is the length for predicting once. The prediction is repeated every m seconds and T is the total prediction length.

Algorithm 1 Two-stage Machine Learning-Based Prediction Scheme

Input: Measured communication quality factors and sensor data
$\mathbf{x} = \{TH, ..., ACC\}$
Output: Predicted throughput $\hat{y}$

for $pred = 1 \rightarrow T/m$ **do**
  for $t' = (pred-1) \times m - n \rightarrow (pred-1) \times m - 1$ **do**
    Preprocess $\mathbf{x}_{t'}$
  end for
  Movement Identification using $\{ \mathbf{x}_{t'} \}$
  for $t = (pred-1) \times m + 1 \rightarrow pred \times m$ **do**
    Prediction using corresponding LSTM model :
    $\hat{y}_t \leftarrow \mathbf{U} \cdot \mathbf{H}^L + b_o$
  end for
end for
return $\hat{y}$

4.3.3 Evaluation

We conduct field experiments to evaluate the throughput prediction method TRUST. We collect data under four different scenarios and predict TCP throughput using TRUST. The results are compared with other methods.

4.3.3.1 Experiment environment

The moving routes or positions of static, walk, and bus scenarios are shown in Figure 4.9(a), and the moving route of train is demonstrated in Figure 4.9(b). In the static scenario, the user remains static in the laboratory located in Nishi-waseda campus in the afternoon. Regarding the walk scenario, the user walks from Nishi-waseda campus to Zoshigaya station in the evening, which takes approximately 21 min for a one-way trip. As for the bus scenario, the user travels on a moving bus from Waseda campus to Nishi-waseda campus in the afternoon, which takes about 10 min for a one-way trip. In the train scenario, the user travels on a moving train from Shimosa-Nakayama station to Shinjuku station in the morning. The datasets used in static, walk, bus, and train scenarios include about 173800 s from 14 days, 29400 s from 22 days, 4260 s from 6 days, and 29500 s from 14 days, respectively. In the evaluation, 80% data are used for model training and 20% for prediction performance test.

(a) Static/walk/bus scenario (b) Train scenario

Figure 4.9 Maps of moving routes in static, walk, bus, and train scenarios.

4.3.3.2 Evaluation metrics

Similar to [29], [88] and [89], the accuracy of the TCP throughput prediction method is evaluated by the relative prediction error R_t and normalized root mean squared error (*NRMSE*) between the predicted value $\hat{y}_t$ and actual value y_t. R_t and *NRMSE* are calculated by:

$$R_t = \frac{|\hat{y}_t - y_t|}{y_t} \tag{4.8}$$

$$NRMSE = \frac{\sqrt{\left(\sum_{t=1}^{T}(\hat{y}_t - y_t)^2\right)/T}}{\left(\sum_{t=1}^{T} y_t\right)/T} \tag{4.9}$$

where T is the total prediction length. The cumulative distributed function (CDF) of R_t is employed to assess the distribution of the prediction error. The *NRMSE* is used to assess the prediction error relative to the mean throughput, enabling the comparison between data with different orders. The smaller *NRMSE* represents the higher accuracy.

4.3.3.3 Prediction performance

We compare the prediction results of TRUST with other seven methods in the four scenarios, respectively. The methods are arithmetic mean (AM) [89], harmonic mean (HM) [89], last sample (LS) [88], MA [29], HMM [88], hybrid model of

autoregressive model and HMM (Hybrid) [63] and stochastic model (Stochastic) [25].

To prove the necessity of user movement pattern identification, we conduct prediction using different datasets and compare the *NRMSE*. The results are shown in Table 4.2. In the table, "All Data" indicates that the datasets for training are the mixed data from all scenarios and a common LSTM model is trained. The prediction is conducted using the common model for all scenarios without movement identification. By contrast, "Data with Same Scenario" indicates that the training dataset is derived from the same scenario and the prediction is conducted using the corresponding LSTM model based on the identification result. For each scenario, the prediction length m is assigned 200 s, 100 s, 20 s, and 5 s. The input length n is set the same as m. These time lengths are selected to assess the performance in long- and short-term prediction. The total prediction lengths T for static, walk, bus, and train scenarios are 34000 s, 5800 s, 800 s, and 5800 s, respectively. The hidden state number d and layer number L of the LSTM models for each scenario are tuned separately to achieve the best prediction accuracy. It can be concluded from Table 4.2 that the prediction error *NRMSE* when using corresponding dataset after identification is smaller than that when using the "All Data". This can be explained by use of the analysis in Section 4.3.1.2. As shown in Figure 4.3 and Figure 4.4, the measured data from the same scenario have similar performance, whereas those from different scenarios have different behavior. When the prediction model is trained using mixed data, finding an optimal solution to fit for all types of data behaviors is challenging. Therefore, it is essential to perform the first-stage identification, and then, use the model trained with data from the same scenario for prediction.

Table 4.2 *NRMSE* Comparison using different training dataset.

Prediction Length m	Training Dataset	*NRMSE*			
		Static	Walk	Bus	Train
200 s	All Data	0.305	0.348	0.386	0.456
	Data with Same Scenario	0.188	0.304	0.264	0.355
100 s	All Data	0.251	0.372	0.308	0.376
	Data with Same Scenario	0.2	0.326	0.249	0.36
20 s	All Data	0.221	0.293	0.26	0.383
	Data with Same Scenario	0.196	0.259	0.223	0.361
5 s	All Data	0.231	0.188	0.254	0.353
	Data with Same Scenario	0.179	0.157	0.217	0.316

Figure 4.10 illustrates the CDF of R_t in the moving bus scenario, where the x-axis represents the value of R_t and the y-axis represents the corresponding CDF of different methods. The prediction length m is 200 s. The CDF result using TRUST with non-preprocessed data (raw) is also shown. We can observe that the prediction accuracy could be extremely poor if the data are not preprocessed to the same order. This result demonstrates the importance of data preprocessing. Regarding the results, 80% predicted data has relative error under 0.294 in TRUST. This is the best among all the methods. Meanwhile, TRUST has 91% predicted data whose R_t is smaller than 0.4. This data ratio is larger than other methods. Similar results can be observed in the short-term prediction where the prediction length m is 20 s as shown in Figure 4.11. There are 80% predicted data whose relative error is under 0.264 in TRUST, which is the best among all the methods. Meanwhile, TRUST has 85.5% predicted data whose R_t is smaller than 0.3. This data ratio is larger than other methods.

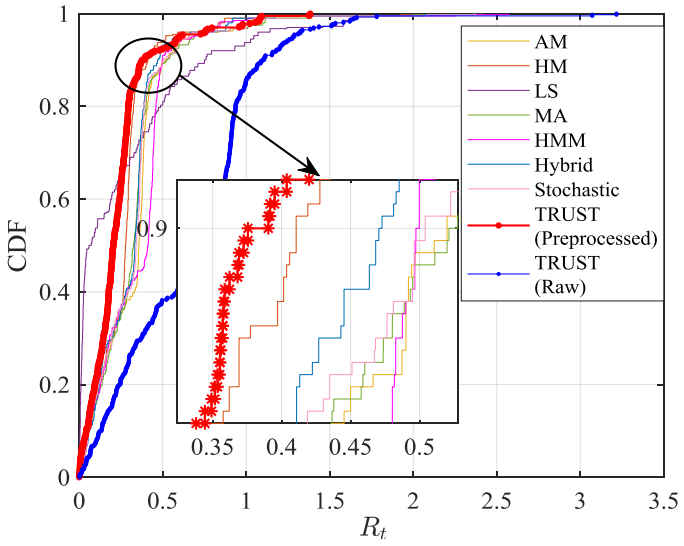


Figure 4.10 CDF of R_t by different methods in the bus scenario when the prediction length m is 200 s.

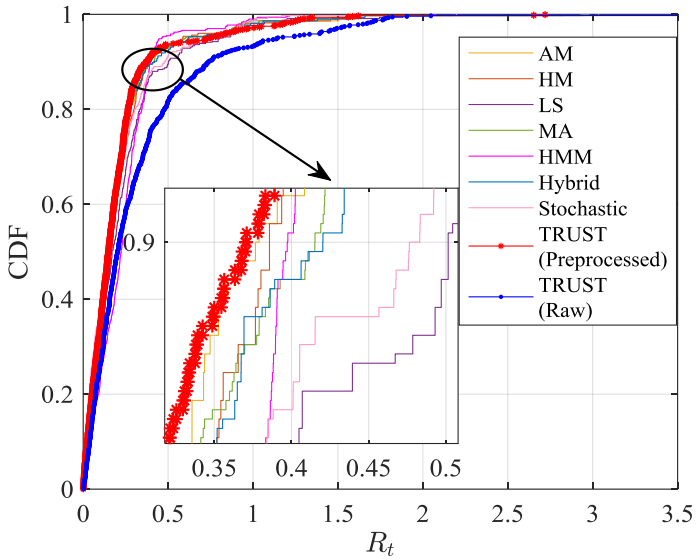


Figure 4.11 CDF of R_t by different methods in the bus scenario when the prediction length m is 20 s.

Figure 4.12 shows the *NRMSE* results of different methods in various scenarios. It can be concluded that, generally the prediction errors are smaller in static scenario than those in moving scenarios since the throughput fluctuation is smaller in static scenario. In the 200-second prediction, the commonly used LS method performs worst in all scenarios because LS only reflects the information in the last timestep. TRUST performs the best in the 200-second prediction because it is based on the LSTM model that considers the effect of long-term information. Similar results can also be observed in the 100-second prediction. In short term prediction, prediction errors are smaller than in long-term prediction, especially under the walk scenario. Similarly, TRUST also performs the best among all the methods. The results imply that TRUST can achieve the lowest prediction error compared with the conventional methods, both in long- and short-term prediction. Throughput prediction errors can be decreased by a maximum of 44% under the moving bus scenario. For other scenarios, the prediction errors are decreased by maximum 38%, 40%, and 34% under static, walk, and train scenarios, respectively.

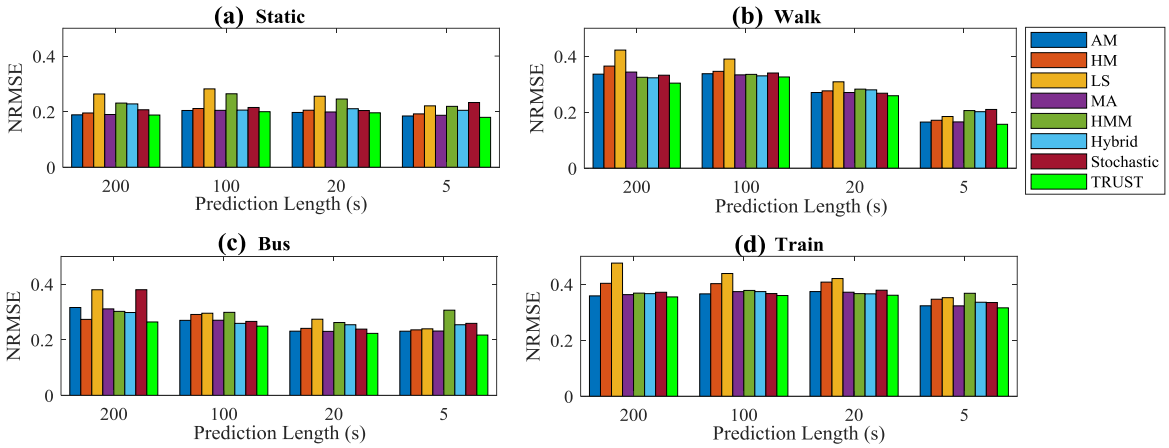


Figure 4.12 *NRMSE* comparison of different methods with various prediction lengths in (a) Static scenario, (b) Walk scenario, (c) Bus scenario, and (d) Train scenario.

4.4 Summary

In chapter 4, we propose TRUST, a machine learning-based TCP throughput prediction method to predict the future throughput for mobile networks. In this method, the user movement pattern is first identified with communication quality factors and sensor data. Based on the identification result, the LSTM model trained with corresponding dataset is used for prediction. For throughput prediction, the communication quality factors; throughput characteristics such as mean value, minimum value, maximum value, variation, and standard deviation; together with scenario information are jointly employed. The input data are preprocessed by a set of data preprocessing methods. Field experiments are conducted for evaluating the prediction method. The results indicate the importance of data preprocessing and user movement pattern identification before prediction. Furthermore, the proposed method effectively predict long- and short-term TCP throughput in different scenarios, and decrease the prediction error by maximum 44% in the moving bus scenario.

For further research, we aim to collect more data of wider areas and improve the prediction accuracy. Moreover, we intend to implement throughput prediction into actual mobile service for video streaming by proposing a new adaptive bitrate control strategy that considers future throughput transitions to ensure high quality of service.

5 Adaptive bitrate control with QoE maximization using throughput prediction ‡

Dynamic adaptive video streaming over HTTP (DASH) is widely studied and adopted in modern video players for ensuring user quality of experience (QoE) since QoE directly affects the revenue. In DASH, adaptive bitrate control is a key part for achieving high quality of service and QoE when transmit video streaming. The ultimate goal of adaptive bitrate control is to maximize video bitrate while minimize rebuffering events and duration. However, this task is non-trivial since the network condition is not always stable. The choice of higher bitrate may cause frequent video freezing which annoying the user while choosing lower bitrate may give worse experience. Therefore, throughput prediction plays an important role in helping select the proper bitrate of video dynamically. Basically, the algorithms need to be tested with large-scale deployment. However, it is not always possible in academic research. In this chapter, we established a video transmission system with DASH which enables replicable trace-based emulations. The emulation enables us to compare different methods under the artificially same condition, with limited experiment. The quality metrics such as average bitrate, the number of rebuffering events, the duration of rebuffering, etc. are examined. The results indicate that a good prediction can provide better user QoE in rate-based adaptive bitrate (ABR) method. In order to further improve the QoE, the buffer occupancy needs to be considered simultaneously. We also proposed a new ABR method which incorporating both prediction and buffer occupancy information named decision map method (DMM). DMM creates both aggressive and conservative mechanisms to handle different network conditions. The emulation results demonstrate that the DMM can achieve better performance in QoE than conventional methods, showing the efficiency of the DMM algorithm.

5.1 Video transmission system with DASH

There exist several adaptive streaming protocols such as Adobe HTTP

‡This chapter is adapted from the work submitted in [91].

Dynamic Streaming [37], Apple HTTP Live Streaming [38], and Microsoft Smooth Streaming [39]. Recent years, dynamic adaptive video streaming over HTTP (DASH) is studied worldwide as a unifying standard [40]. In DASH protocol, the video contents are divided into short chunks and encoded at different bitrate levels. Then the client player can request the segment chunks with proper bitrate successively and dynamically according to the network condition. The algorithm for selecting download bitrate is called adaptive bitrate (ABR) algorithm. The ABR algorithm employs the network condition logs (such as throughput, buffer occupancy and etc.) which monitoring in the client side to decide the bitrate of the latter downloading chunks. The purpose is maximizing the video quality while reducing rebuffering.

5.1.1 DASH system structure

As shown in Figure 5.1, the basic DASH system structure consists of a HTTP server and a DASH client side which communicates with the content server and plays the video. The video content is encoded at different bitrates and stored in the server orderly. Here in the DASH context, different bitrate versions are named representations. The contents are then divided into short chunks for example each chunk includes 2-second video playback time. For different representations, although the video qualities are various, the start time and end time of each chunk are aligned. Therefore, the chunks in different representations can be concatenated and played smoothly only via the chunk order, enabling the dynamical choice of video quality during streaming. All the information about the video content and the representation details are written in the Media Presentation Description (MPD) file. The MPD documents the number of representation for the video content, the encoding bitrates, the URLs of the chunks and etc. By parsing the MPD file, the client side can obtain full knowledge of the contents.

The DASH client side mainly consists of a MPD parser module, ABR control module, HTTP client module, video buffer and the media player. During the streaming, the MPD file is firstly requested and downloaded by the client and parsed to get the information about the video contents. Then the ABR control module will determine which representation to select for the next video chunk and tell the decision to the HTTP client. The HTTP client then generates a request and communicates with the server to get the corresponding video chunk. After completion of the current chunk, the content will be stored into the video buffer and the ABR control module will repeat the download selection for the next chunk. After enough video buffer is filled, the media player will begin to playback the video.

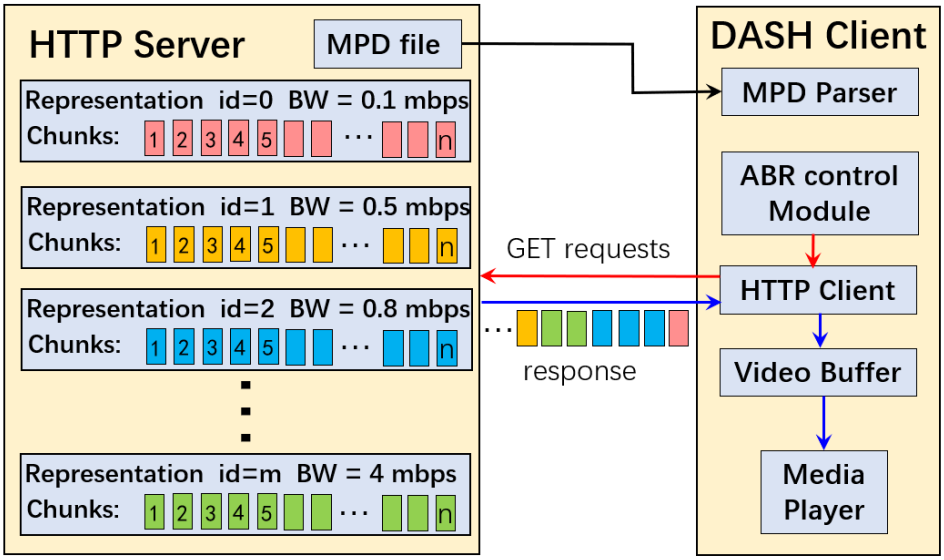


Figure 5.1 The DASH system structure.

5.1.2 DASH client framework

5.1.2.1 Basic architecture

A JavaScript-based DASH client is adopted in this study which is originally developed by ITEC [8], [43]. This basic structure of the framework is shown in Figure 5.2 [43]. The Google Chrome Media Source API is used to play the video.

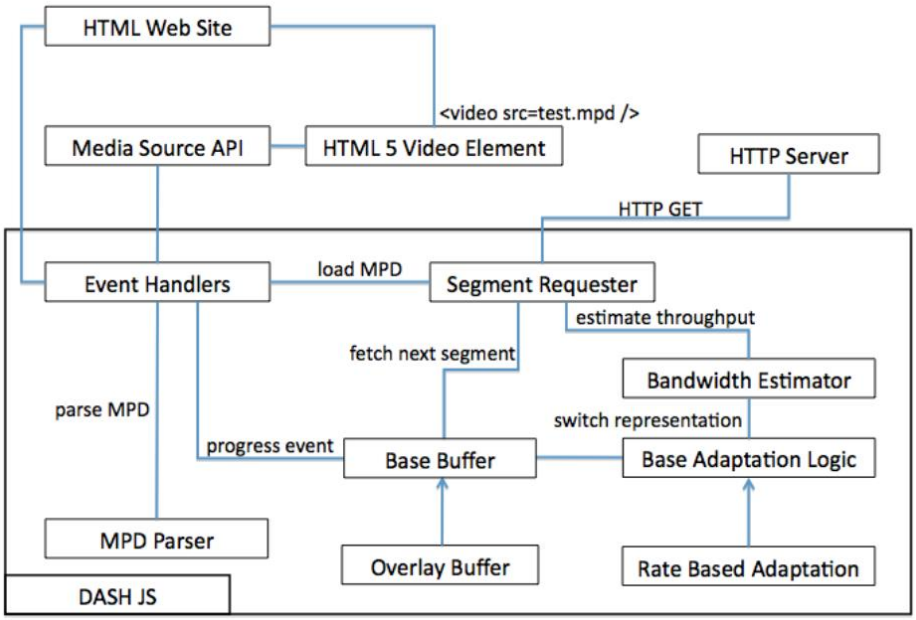


Figure 5.2 The architecture of the basic DASH client framework [43].

The working flow of this structure is as follows. First, when the global object `dashPlayer` is created, the download of the MPD file is triggered. When the download is completed, the `MPDparser()` is called to analyze the full information of the video contents to be downloaded. Then the `bandwidth`, `rateBasedAdaptation` and `mediaSourceBuffer` are created to estimate the bandwidth, to switch the representation which to be downloaded for the next chuck, and to store the downloaded contents. After this, the `MediaSource()` object is created and attached to the HTML video element. When the `MediaSource()` open event is listened, the monitoring of the buffer state will be triggered and the downloading and streaming of the video content will start. During the streaming, the buffer state is monitored with a preset interval, such as every 100 ms. When the buffer state is lower than a threshold (defined as `criticalLevel` here), the HTTP request will be generated and sent by the `XMLHttpRequest()` object. The download request is continued until the buffer state reaches the preset maximum (defined as `bufferSize.maxseconds` here). Each time the download of a certain video segment is completed, the average downloading speed will be calculated and then the estimation of the bandwidth for the next segment is made. Based on the estimation, the bitrate is adaptively selected, and the next HTTP request is sent. This process is repeated to the end of the video session.

Since the buffer of the `MediaSource()` class is not accessible, an `overlayBuffer` object is created which keeps a record of the left play duration in the media API, which is indicated by variable `mediaElementBuffered`. This object does not store any data, and the data is stored in the object `baseBuffer`. The `baseBuffer` is ring buffer in which two pointers, add pointer and get pointer, are used to refer the buffer index to be extracted and to be written in, respectively. When the play duration left in the media API is less than a threshold, the `drain()` function will be triggered to extract data from the `baseBuffer` as shown in Figure 5.3. The data gotten from the server is also stored in `baseBuffer`.

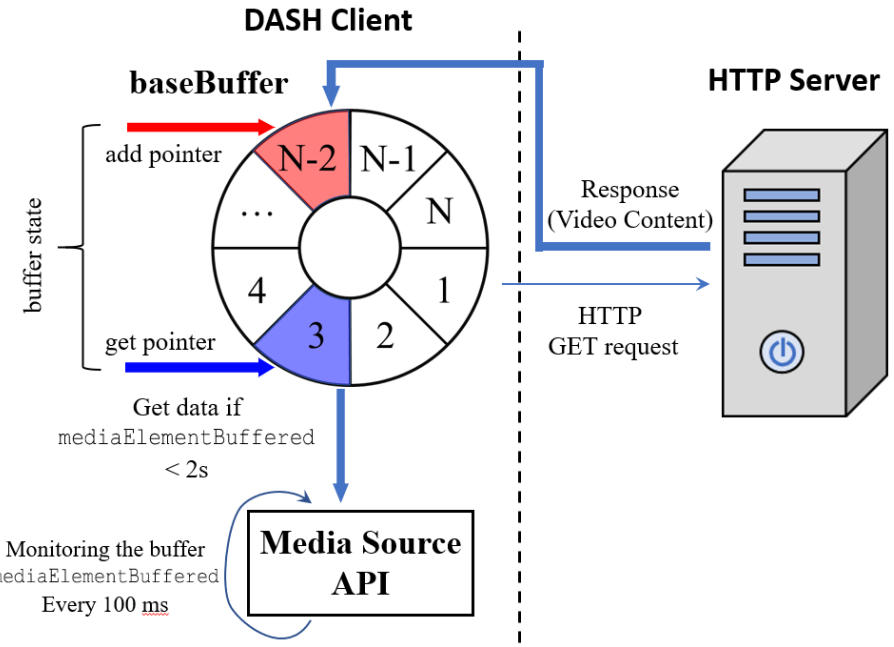


Figure 5.3 The buffer structure of the framework.

5.1.2.2 Modifications and extensions

In deploying the DASH framework, we find some problems and make modifications to original version. Besides, we also add some modules in order to extend the functionality and analyze the results.

A) Real-time plot functions and metrics

First, we modify the plot module and add new functions to show the results during the video session. In the original version, only the estimated bandwidth, selected bitrate and the current playback time are shown. The relationship between the current playback time (black bar) and the estimated bandwidth (red line) is not clear. Nor, the time axis is not clear because the metrics and scales are not shown. As shown in Figure 5.4, in the modification, we correct the time axis and add the scale to show the playback time duration. The actual average download throughput is added to the plot as the green line. The estimated bandwidth, the representation rate and the actual throughput are plotted every segment since the video contents are divided into 2-second chunks. The last segments in estimated bandwidth and representation rate are plotted as dash line since this segment is not downloaded yet. This plot is updated upon the estimation is made or the download is completed.

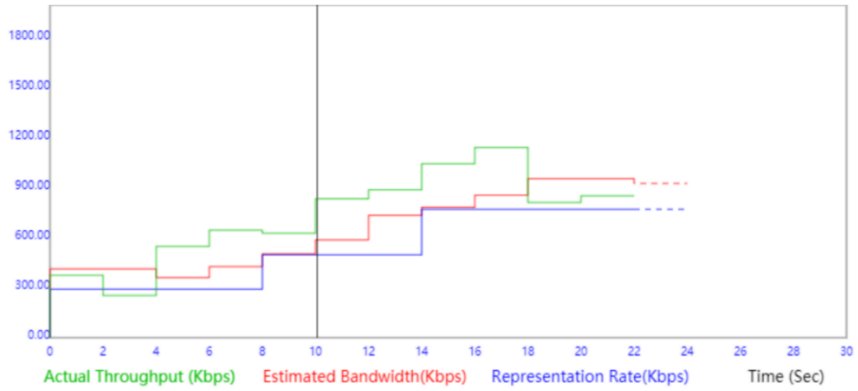


Figure 5.4 Plot of the playback information of the video session.

Besides the plot of the playback information, we add another plot to show the buffer occupancy and the download information in the real time. As shown in Figure 5.5, the black line is the buffer occupancy in real time and the green squares are the download information. The height of each square stands for the average downloading speed of the segment and the length stands for the consumed time in downloading. The area of the square is the size of the segment. Different from the plot in Figure 5.4, this time axis is consistent with the real time world. Even when the video session is rebuffered or paused, the plot is updated constantly. As can be seen in the example in Figure 5.5, when the video session encounters the running out of buffer and rebuffering, the lasting time of such event is recorded. After the video session, these logs can be exploited to analyze the performance of adaptive control algorithms. This plot is updated every time the buffer is monitored, and the download is completed.

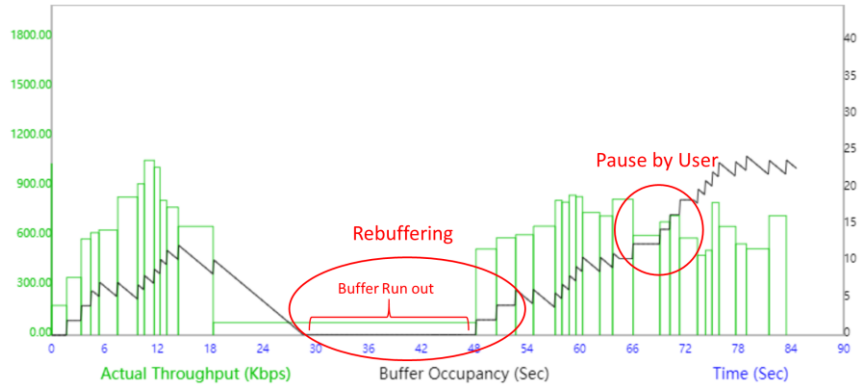


Figure 5.5 The plot of the real-time buffer occupancy and downloading information.

B) Bandwidth estimation module

Second, we add some bandwidth estimation methods to the current module. In the original version, the bandwidth estimation for the next segment is made based on the average download speed of the current segment and the former estimation. However, this kind of estimation only considers the average value and the lasting time of downloading is not considered. If the downloading time of the current segment is very long, the influence of the former ones should not be included. Here, we propose a method that take the downloading time into consideration. As shown in Figure 5.6, we use the former *hisSize* seconds data to estimate the bandwidth as follows:

$$C_{pred} = \frac{\sum_{n=1}^{k} \int_{t_{begin}[n]}^{t_{end}[n]} \{f(t) \cdot bps[n]\}}{\sum_{n=1}^{k} \int_{t_{begin}[n]}^{t_{end}[n]} f(t)} \tag{5.1}$$

where $t_{begin}[n]$ and $t_{end}[n]$ are the start and end timestamps of downloading the former nth segment. k is number of the historical segments involved in calculation. C_{pred} stands for predicted capacity, which is the estimated bandwidth. $f(t)$ is a function to weight the contribution of the historical download speed. Basically, the information in the past should have fewer impact on the estimation.

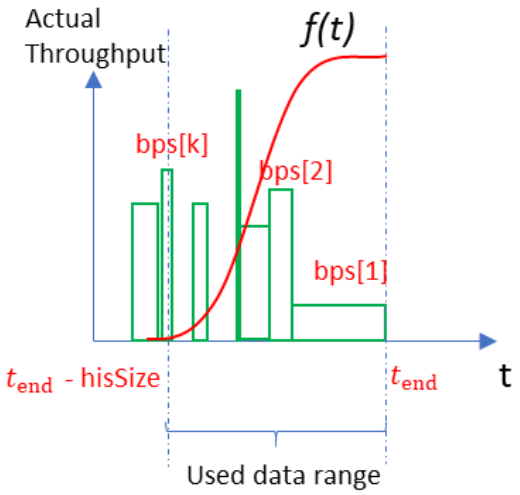


Figure 5.6 The calculation of bandwidth estimation considering lasting time.

For the simplest case, we can choose $f(t) = 1$ to only consider the lasting time as the weight as:

$$C_{pred} = \frac{\sum_{n=1}^{k} \{(t_{end}[n] - t_{begin}[n]) \cdot bps[n]\}}{\sum_{n=1}^{k} (t_{end}[n] - t_{begin}[n])} \tag{5.2}$$

Besides, we also add another throughput calculation and prediction mechanism into the framework. The original download speed measurement is done by segment, which means when the download of a certain segment is over, the calculation will be carried out for an average throughput. However, in the former chapters, the throughput data are measured every second and the prediction is also made in second. Therefore, we add an additional module to calculate the throughput. This is realized by the event handler `onProgress()` in the `XMLHttpRequest()`. During the data transferring, the `onprogress` event is fired periodically and the throughput can be calculated by making use of the data loaded and the timestamp information which contains in the `onprogress` event. With the measurement of throughput in second, the prediction methods mentioned in the former chapters can be integrated into the framework.

C) Bitrate switch logic

Third, we correct original bitrate switch logic which is problematic. In the original version, the rate-based adaptive bitrate control method is used, which chooses the maximum bitrate that is lower than the estimated bandwidth. When the estimated bandwidth is smaller than the lowest encoding bitrate, no action is taken, and the selection is not triggered. The bitrate is kept the same as the former segment. However, since the estimated bandwidth is even lower than the lowest encoding bitrate, the former selection may be aggressive for this time. Therefore, we correct the bitrate choice as the lowest one in this case.

D) Buffer logic and buffer strategy

Fourth, some logics of the buffer are corrected. In the original version, the buffer fill state only considers the data stored in the `baseBuffer` when the running out is judged and the data already pushed into the Media Source Extension (MSE) is not counted. Even though the metric of the base buffer is zero, there are still data not played yet in the MSE. Therefore, there is sometimes false judgement of exhaustion of buffer. This may affect the statistics of rebuffering events in the post-processing. Here, we re-define the buffer occupancy as the summation of the buffer in the `baseBuffer` and MSE to judge whether the rebuffering events occurs. Another amendment is made to the logic

of triggering the download. In the original version, when we want to keep the buffer level in a certain degree, the critical level and maximum size is set the same. However, the difference between maximum buffer fill state and the minimum one is duration of 2 segments. This is caused by a logic flaw that hinder the download when the buffer occupancy drops below the critical level for the first time. After the modification, the buffer can be kept in a certain level where the difference between maximum and minimum buffer fill state is one segment duration.

We also extend the buffer strategy. We add a new factor B_{start} which indicates the buffer occupancy for when to start the video session in the initial stage. In the original version, the video will start after the buffer reaches the critical level. However, if the critical level is large, it will take a long time to start the session which will annoy the user and may cause quit. On the other hand, the critical level cannot be set as small value because there is danger to encounter rebuffering events. Since the initial waiting time is a very essential factor for quality of experience (QoE), it should be able to adjust independently. This factor is also applied when the rebuffering event occurs. Basically, it is expected the video will start very soon in the first stage or restart in the rebuffering event. Therefore, this factor is always set as a relatively small value.

In the original version, the buffer strategy is determined by two factors, the maximum buffer occupancy B_{max} and critical level $B_{critical.}$ These two factors can allow the buffer acts as the “Long on-off” or “Short on-off” behavior. However, these two strategies are not flexible enough to both ensure good QoE and memory consumption. Assuming that in the “Long on-off” case as shown in Figure 5.7, it can be seen that during the OFF phase, there is no data transferring between the server and client. This will be dangerous because the communication condition is unknown. If the condition becomes bad during OFF phase, there is a danger to choose bitrate aggressively based on the relatively old record, which may result in rebuffering. On the other hand, assuming that in the “Short on-off” case as shown in Figure 5.8, it can be seen the buffer is kept in a relatively stable high level to avoid rebuffering. However, this strategy may consume unnecessary resource, if the communication condition is quite well. There is no need to keep the buffer in a high level.

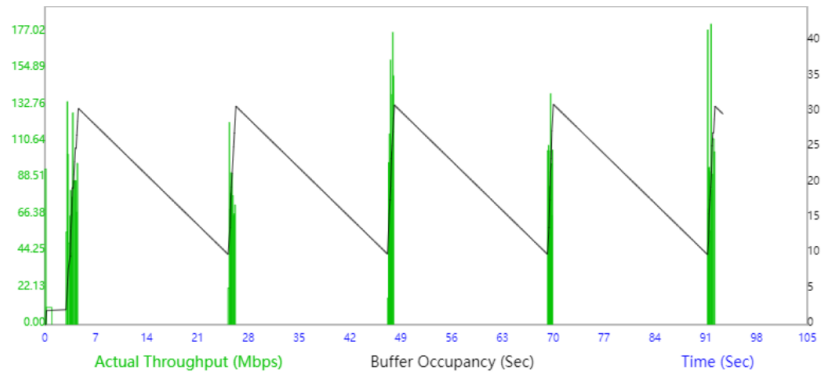


Figure 5.7 The "Long on-off" buffer strategy.

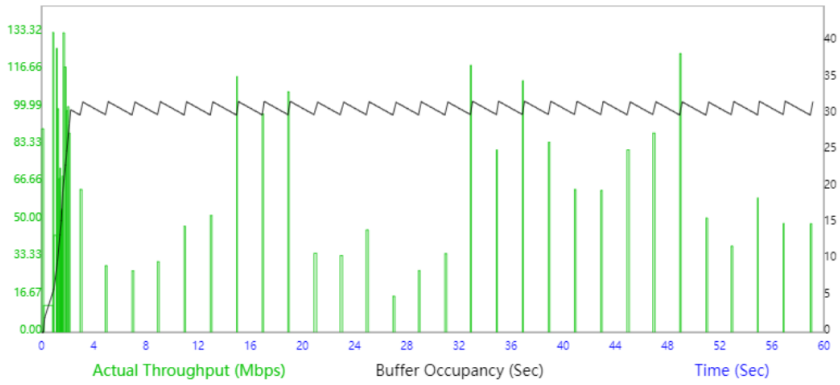


Figure 5.8 The "Short on-off" buffer strategy.

Therefore, we proposed a variable buffer strategy in which the buffer level is adjustable based on the current communication condition. The proposed strategy is shown in Figure 5.9. The buffer level is adjusted as a function of the ratio of predicted throughput and bitrate. In this method, four parameters are set, B_{upper}, B_{lower}, $Ratio_{upper}$, and $Ratio_{lower}$, which are the buffer level upper limit, buffer lower limit, ratio upper limit, and ratio lower limit, respectively. When the communication condition is expected to be good enough, that is the ratio between prediction throughput and bitrate is larger than $Ratio_{upper}$, the buffer level can be set as B_{lower}. When the condition is not good, that is the ratio is lower than the $Ratio_{upper}$, the buffer level should be set as B_{upper}. When the ratio is between the upper and lower limits, the buffer level is a function of the ratio. $B_{critical} = f(C_{pred}/R_{next})$. Here, we show the linear, conservative, aggressive mapping relationships between the buffer level and the ratio in black, red, blue lines, respectively. Figure 5.10 shows an example of the buffer occupancy dynamics using the proposed buffer strategy. Here the linear mapping function is used and the B_{upper}, B_{lower}, $Ratio_{upper}$,

and $Ratio_{lower}$ are set as 40 s, 20 s, 8 s and 4 s, respectively. As can be seen, in the stage of red circle, the throughput is not very large. Therefore, the buffer occupancy is constantly increasing to the upper buffer limit. However, during the buffer increasing, the communication condition is turning good as shown in the stage of blue circle. Therefore, the buffer occupancy stops climbing and starts to decrease. When the throughput keeps in a good condition as shown in the stage of black circle, the buffer level also keeps at a low level. This figure demonstrates that using the proposed buffer strategy, the buffer occupancy can be adjusted adaptively according to the communication condition between the server and the client. By properly setting the buffer mapping function, the strategy can help save resource as well as avoid rebuffering events.

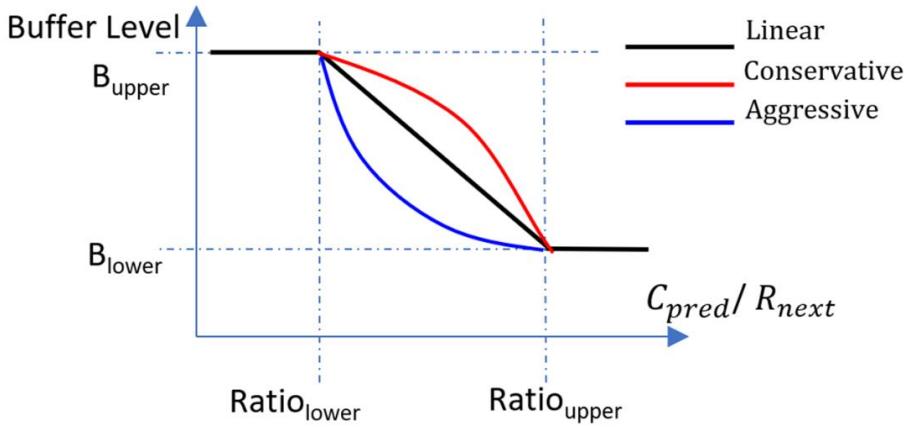


Figure 5.9 The illustration of the variable buffer strategy.

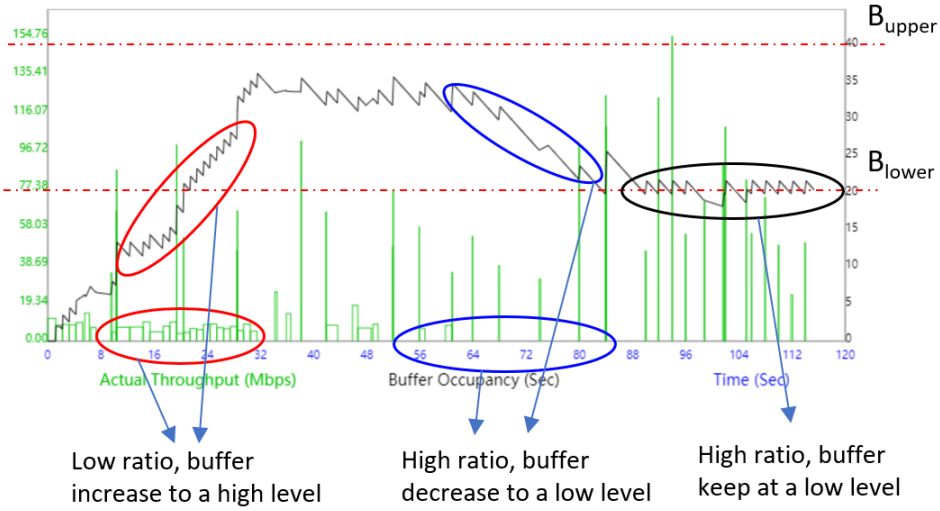


Figure 5.10 The buffer occupancy using the variable buffer strategy.

E) Logs storing module

Finally, we add a new module to store all the logs for post-processing and analysis, such as QoE. These logs include the buffer occupancy, throughput measurement, bandwidth estimation, bitrate selection and the data transfer logs in the `onProgress` event. All the logs contain the value and the corresponding time stamp. Note that the throughput measurement has two timestamps, which are the start time and end time of the transferring of a certain segment. By analyzing these logs, the performance of different methods and algorithm can be compared.

5.1.3 Trace-based HTTP server

5.1.3.1 The concept of trace-based server

The development of ABR algorithms is still ongoing and a widely accepted method is not achieved yet. Since the network in real world is always dynamic, we cannot evaluate the algorithms under two completely same network conditions. Therefore, to validate the actual efficiency of the algorithms, basically large-scale deployment in real network environment is needed via video streaming providers. Then, the data containing logs from millions of video sessions can be analyzed statistically. However, it is not always the case for academic researchers to obtain such large-scale data. As an alternative, the trace-based emulation is employed for evaluation. Under the artificially same network condition, the effect of different algorithms can be compared using limited deployment. Here, we developed the a trace-based server.

The implementation of our emulation is shown in Figure 5.11. The purpose is to replicate the throughput between the server and client as the same as a given throughput trace. Since the network condition in the real world cannot be fully controlled, we build a virtual emulation environment of the network. In this environment, the server is built on the same computer as the client using the local host 127.0.0.1. By using the local host, we can regard the delay time of the data transfer to be small enough (< 1 ms) that can be ignored. The `node.js` is used to build the HTTP server. The choice of this structure is because it gives full control of the data transferring, such as when to respond, what to respond. In order to constrain the throughput between the server and client, the response of the HTTP request is manipulated and delayed intentionally. Therefore, from the viewpoint of the client side, the throughput is changing dynamically since it doesn't know what happens in the server side.

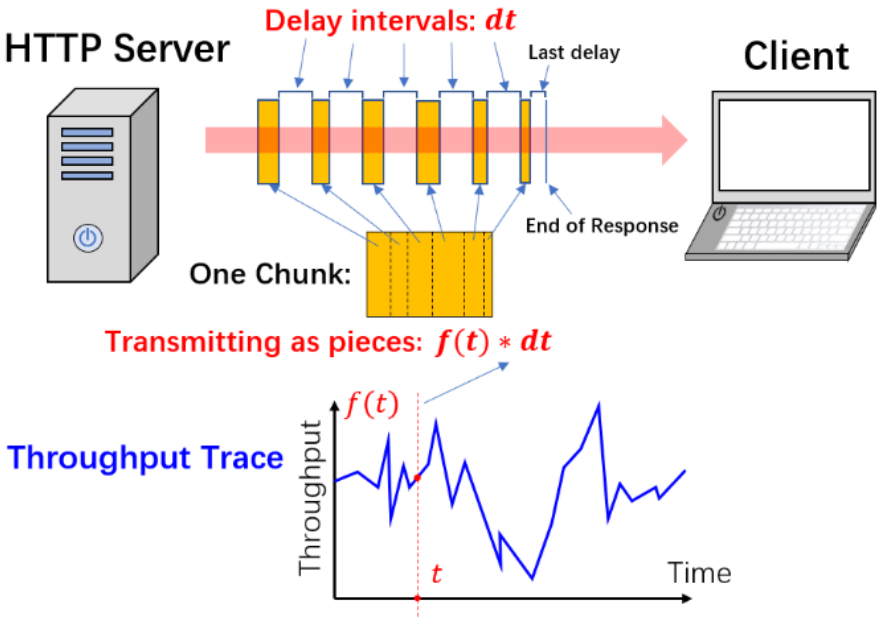


Figure 5.11 The implementation of the trace-based emulation.

5.1.3.2 The algorithm of the trace-based server

The procedure of the manipulation is as shown in algorithm I. The throughput trace is prepared in the server side which can be referred freely. The trace is stored every second. When the server catches the GET request from the client, it firstly judge whether this request is for video contents or other contents such as html file or JavaScript files. If the request is for other files, the data are responded back to the client immediately. When the request for the first segment of the video content is captured, an initial time stamp t_{init} is created which is regarded as a baseline for calculating the time delay. When the requests for video contents are captured, the size of the requested chunk is analyzed. Normally, the chunk data should be sent back to client immediately. However, in order to shape the throughput according to the designated trace, the chunk data is divided and transmitted as pieces with artificial delay intervals *dt*. After the request is coming, the elapsed time from the initial time stamp is calculated as $t = t_1 - t_{init}$. Then the sending size is determined by $S_{sent} = f(t)*dt$, where $f(t)$ is the throughput value at t in the prepared trace table and *dt* is the delay interval. *dt* is chosen as 100 ms here. After sending, the server waits for *dt* time (or 100 ms here) and then repeats the elapsed time calculation $t = t_{n-1} - t_{init}$ and sending size determination according to the new $f(t)$. After sending the last piece of the chunk, the delay interval *dt* should be recalculated since more data can be sent within *dt*. The delay is calculated by

Algorithm I: Trace-based Server Implementation

Input: the requests captured from the client, response interval: dt, the prepared throughput trace $f(t)$
Output: the responding data D_{sent}, the last sent data D_{last}, the end signal of response *respEnd*

```
dt = 100 ms;
while a GET request comes do
    analyze the request and get the file type filetype;
    get the requested file content filecontent;
    if filetype ~= .m4s then
        D_last = filecontent[1:end];
        send respEnd;
    end
    else filetype == .m4s then
        if requesting first chunk then
            record the initial time stamp t_init = getTime();
        end
        n = 1;
        t_n = floor(getTime() - t_init);
        the start index of the sending data: ST = 1;
        S_sent = dt*f(t_n);
        while (ST + S_sent - 1) < size(filecontent) do
            D_sent = filecontent[ST : S_sent];
            ST = ST + S_sent;
            n++;
            t_n = floor(getTime() - t_init);
            S_sent = dt*f(t_n);
            wait for dt;
        end
        D_last = filecontent[ST : end];
        wait for D_last/ f(t_n);
        send respEnd;
    end
end
```

$S_{last}/f(t)$, where S_{last} is the size of the last piece and $f(t)$ is the throughput at the time sending the last piece. After the last delay, the end signal of the response is triggered. Note that the trace data are usually stored every second for many datasets. In implementation, t is floored to integer second since the unit in the time stamps is millisecond.

5.1.3.3 Trace-based emulation validation

In order to validate whether the trace-based emulation is successfully implemented, and the virtual network environment is reproducible, the test video streaming is conducted. The used content is BigBuckBunny. In the test, the bitrate adaptive bitrate switch is disabled, and the Client is set to request the content encoded in 515 kbps constantly. Figure 5.12 shows the results of the trace-based emulation. The blue stairs plot is the throughput trace used to shape the sending sequence of the data on server side. The red one and black one are the throughputs recorded and calculated on the client side in two tests. As can be seen, the results of the two tests are the same. And the shapes of the measured throughputs in the client side are the same as prepared trace in the server side. Figure 5.13 shows the buffer occupancy logs of the two tests. As can be seen, the buffer occupancy logs are also identical. These results demonstrate that our trace-based emulation is successful in shaping the throughput between server and client, and the emulated network condition can be replicated, allowing us to compare the algorithms quantitatively with limited experiments.

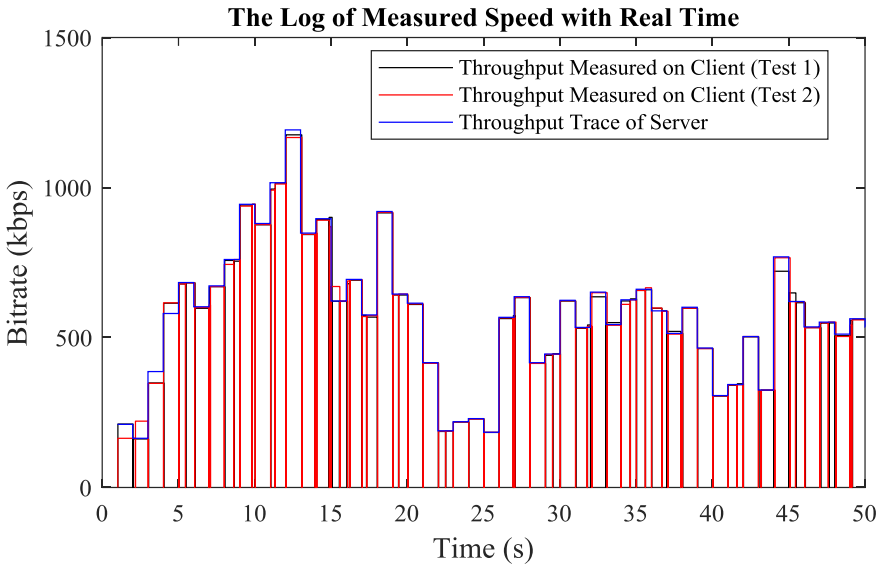


Figure 5.12 The prepared trace applied on server and the corresponding measured throughputs on client side in two tests.

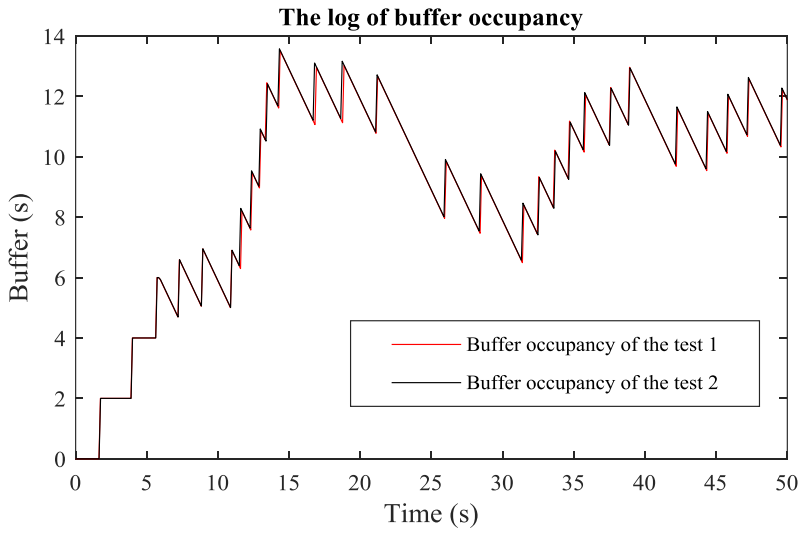


Figure 5.13 Buffer occupancy of the two tests.

5.2 Implementation of throughput prediction in DASH

The default throughput estimation method in the original DASH client is based on the average throughput of one segment. However, these measurements cannot be directly used in the prediction methods proposed in the previous chapters since these methods assume that the throughput data are calculated every second. In the extended version, the throughput measurements by second are available taking the advantage of the `onprogress` event. In our previous work, we have proposed and implemented several methods in prediction. The methods are arithmetic mean (AM) [89], harmonic mean (HM) [89], last sample (LS) [88], MA [29], HMM [88], stochastic model (Stochastic) [25], hybrid model of autoregressive model and HMM (Hybrid) [63] and throughput prediction based on LSTM (TRUST) [90]. To incorporate the prediction results by the aforementioned algorithms into the emulation, a new prediction trace is applied in the client side. This prediction trace is created off-line which is related to the throughput trace on server side. Since the network condition is reproducible with regard to the prepared trace, we can just simply change the prediction trace on the client side by different prediction algorithms. The rate-based (RB) adaptive bitrate control method is employed where the bitrate selection is decided based on the prediction value. Here, the value can be obtained from the prediction trace table applied in the client side. In order to compare the effect of these methods, during evaluation, only the prediction method is changed, and other settings are kept the same such as the buffer strategy.

5.3 Evaluation, analysis, and discussion

5.3.1 Setup and QoE metrics

In evaluation, the video content is encoded into 14 versions from 100 kbps to 4000 kbps. Total length of the video is about 598 seconds. Each chunk contains 2-second video. The B_{upper}, B_{lower}, $Ratio_{upper}$, and $Ratio_{lower}$ are set to 40 s, 20 s, 8 s, and 4 s, and B_{start} is set to 6 second. The video streaming experiments are conducted using different prediction methods under the same network condition shaped by the trace on the server side. The buffer occupancy log and the choice of bitrate for each chunk are recorded for post analysis.

In order to analyze the performance of each algorithm, some factors are considered in the QoE calculation, which are the initial delay T_{init}, number of rebuffering N_{rebuf}, rebuffering duration T_{rebuf}, the average bitrate R_{ave}, and the switch frequency of bitrate. T_{init}, N_{rebuf}, and T_{rebuf} are extracted from the buffer log. R_{ave} and switch frequency are extracted from the bitrate choice log. The five factors are analyzed here as metrics for performance assessment. Additionally, the formula used in [1] is adopted as the primary QoE metrics, which is:

$$QoE = \sum_{n=1}^{N} q(R_n) - \mu T_{rebuf} - \mu_s T_{init} - \sum_{n=1}^{N-1} \left| q(R_{n+1}) - q(R_n) \right| \tag{5.3}$$

where $q(R_n)$ represents the relationship between bitrate and user perceived quality. N is the total number of chunks. T_{rebuf} and T_{init} are the total rebuffering time and initial delay. μ and μ_s are the corresponding penalties. The last term on the left stands for the penalty of bitrate switch. The linear form $q(R_n) = R_n$ is considered here. The μ and μ_s are chosen as the maximum bitrate. Note that there are also a lot of other QoE definitions which takes the factors with different weights. Therefore, using different QoE metrics, the performance judgement may change. Since the official QoE definition is not available yet, we use the most widely used one. Of course, one can even give their own equation for QoE calculation if proper explanation is provided.

5.3.2 Performance evaluation

The throughput traces implemented in server are chosen from the open dataset *Mobile dataset* (*HSDPA*) [48]. Figure 5.14 shows a selected trace from HSDPA. This trace is measured on the ferry. As can be seen, there is a period the network condition is

extremely bad which is almost cutoff. The results of different methods are shown in Table 5.1. From the total QoE, the LSTM performs best among all the methods. For the individual metric, it can be seen that LSTM makes a conservative choice in the initial phase, resulting in a small delay. As for the average bitrate, DASH-original, LS and Stochastic has a relatively high score. However, this aggressive choice leads to longer rebuffering duration. DASH-original even has 5 rebuffering events. The log of buffer occupancy using different prediction methods in Ferry Trace is shown in Figure 5.15. As can be seen that, the buffer occupancy of the DASH-original, LS and Stochastic methods are always at a relative low level. This causes danger of the frequent rebuffering events. Although the number of rebuffering events is not included in Equation (5.3), the more events may damage the user experience more with the same rebuffering duration. On the other hand, although LSTM performs best from the viewpoint of the total QoE, it still can be improved. During the whole session, the buffer occupancy is sometimes at a relatively high level. For these periods, the bitrate can be selected more aggressively.

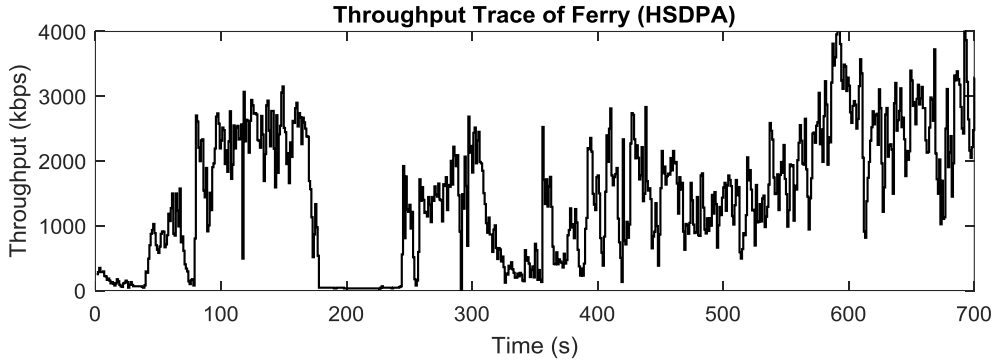


Figure 5.14 The throughput trace of Ferry (HSDPA).

Table 5.1 The results of Ferry Trace.

Methods	Metrics				
	QoE	R_{ave} (kbps)	T_{init} (s)	T_{rebuf} (s)	N_{rebuf}
LSTM	140.7	1042.1	2.4	32.4	1
Hybrid	118.6	1044.2	5.5	32.7	2
LS	84.0	1313.4	6.2	48.7	1
MA	100.5	1050.5	6.2	40.4	2
Stochastic	104.1	1270.8	5.9	46.3	2
AM	130.2	1011.6	6.2	30.6	3
HM	75.2	844.1	5.0	31.2	2
HMM	53.5	818.7	4.9	30.7	2
Original	92.8	1458.8	5.6	68.5	5

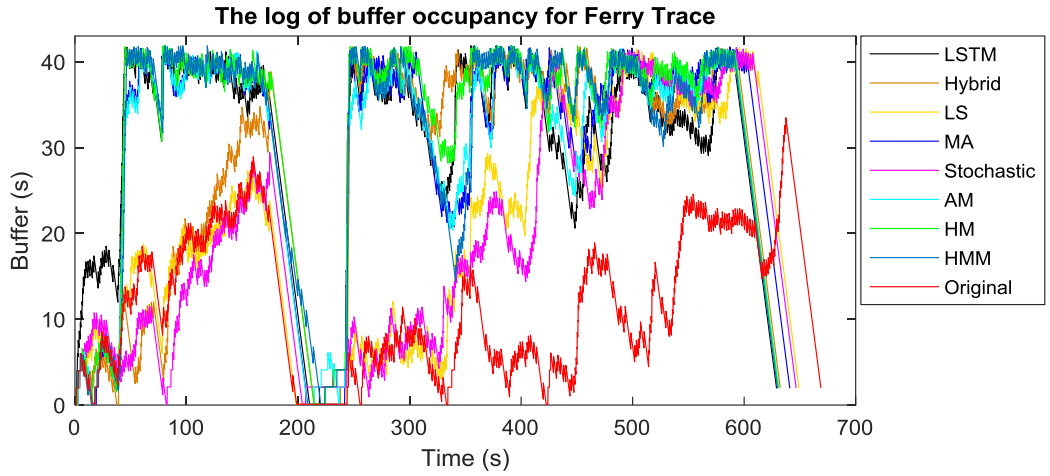


Figure 5.15 The log of buffer occupancy using different prediction methods in Ferry.

Figure 5.16 shows another selected trace which is measured on the bus. As can be seen, the average throughput is relatively higher than that in Ferry trace. The average quality of the video transmission should be higher. However, there are still some sudden degradation of network condition such as around the 150 s and 350 s. These areas need to be handled well otherwise could cause rebuffering events. Table 5.2 shows the QoE results of different methods. From the total QoE, it can be confirmed that LSTM still performs best among others. The original method has the tendency to be excessively aggressive. Therefore, this aggressive strategy gains the highest average bitrate in sacrifice of rebuffering events and time. This situation also can be observed from the former trace. The log of buffer occupancy using different prediction methods in Ferry Trace is shown in Figure 5.17. As can be seen, the buffer occupancy of LSTM also at a relatively high level. There is still space to improve the adaptive bitrate control method.

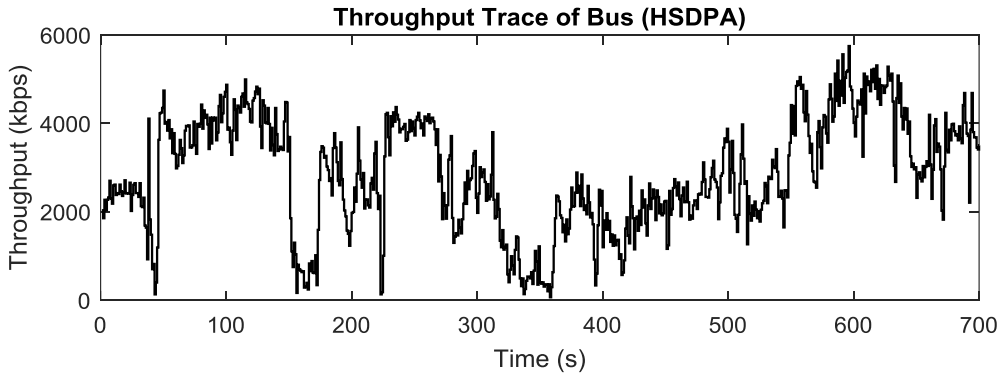


Figure 5.16 The throughput trace of Bus (HSDPA).

Table 5.2 The results of Bus Trace.

Methods	Metrics				
	QoE	R_{ave} (kbps)	T_{init} (s)	T_{rebuf} (s)	N_{rebuf}
LSTM	612.0	2263.3	6.3	0.0	0
Hybrid	581.9	2163.1	3.7	0.0	0
LS	558.6	2364.6	6.5	0.0	0
MA	604.3	2199.2	4.0	0.0	0
Stochastic	553.9	2240.3	5.9	0.0	0
AM	599.6	2209.5	6.4	0.0	0
HM	507.7	1928.9	6.0	0.0	0
HMM	490.9	1996.9	4.1	0.0	0
Original	561.7	2648.4	14.1	34.1	4

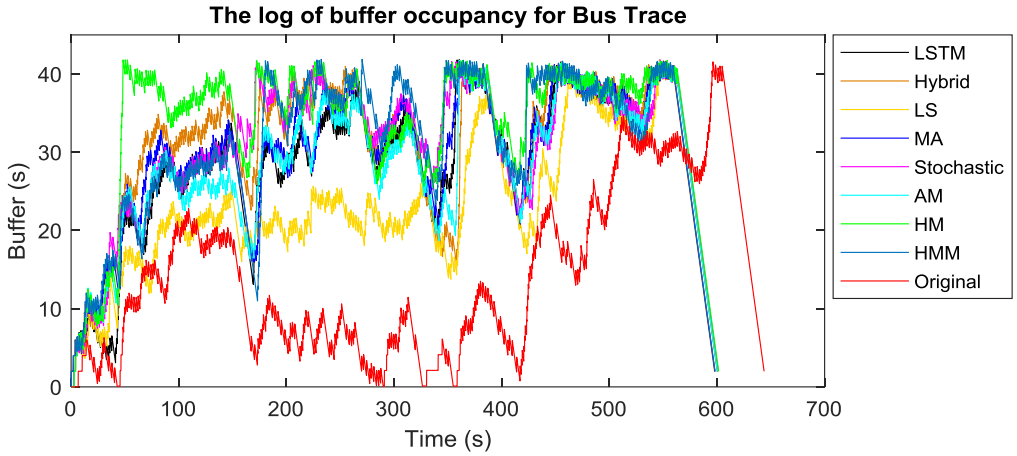


Figure 5.17 The log of buffer occupancy using different prediction methods in Bus.

Figure 5.18 shows another trace from Tram [48]. As can be seen, the average throughput is relatively lower than that in Ferry trace. There is a period the network condition is bad but no cutoff. The results of different methods are shown in Table 5.3. In this case, the LSTM still has the better QoE score than other prediction methods. However, the original performs best this time. This aggressive strategy wins in achieving the highest average bitrate and successfully avoids rebuffering events. The log of buffer occupancy using different prediction methods in Tram trace is shown in Figure 5.19. As can be seen, the buffer occupancy during the streaming is at a relatively high level, which means the bitrate selection is somehow conservative. This is considered to be the reason that the LSTM is defeated by the original method. Since the ABR algorithm used here is rate-based, the chosen bitrate is always below the prediction. Therefore, the bitrate selection can be improved while reducing the buffer occupancy by taking advantage of the buffer occupancy information.

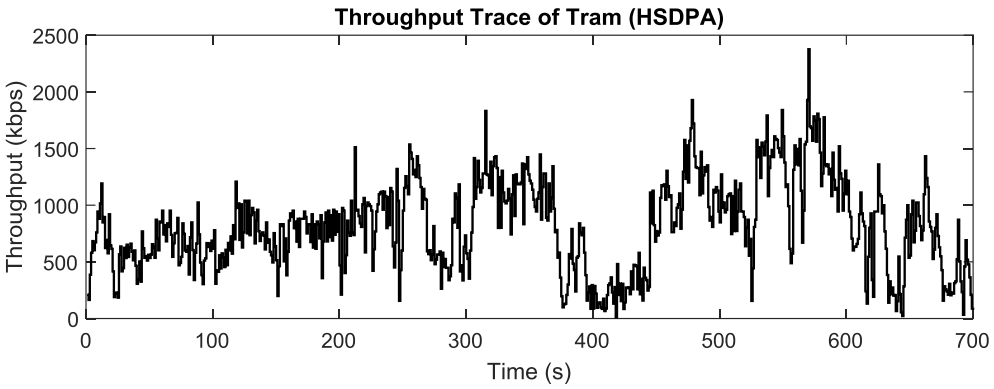


Figure 5.18 The throughput trace of Tram.

Table 5.3 The results of Tram Trace.

Methods	Metrics				
	QoE	R_{ave} (kbps)	T_{init} (s)	T_{rebuf} (s)	N_{rebuf}
LSTM	170.9	698.4	4.2	0.0	0
Hybrid	161.9	656.4	4.5	0.0	0
LS	136.8	664.0	3.2	0.0	0
MA	159.7	639.8	4.3	0.0	0
Stochastic	154.8	692.7	5.6	0.0	0
AM	160.8	654.5	5.2	0.0	0
HM	143.7	584.9	3.7	0.0	0
HMM	132.8	594.6	3.9	0.0	0
Original	181.9	716.8	3.9	0.0	0

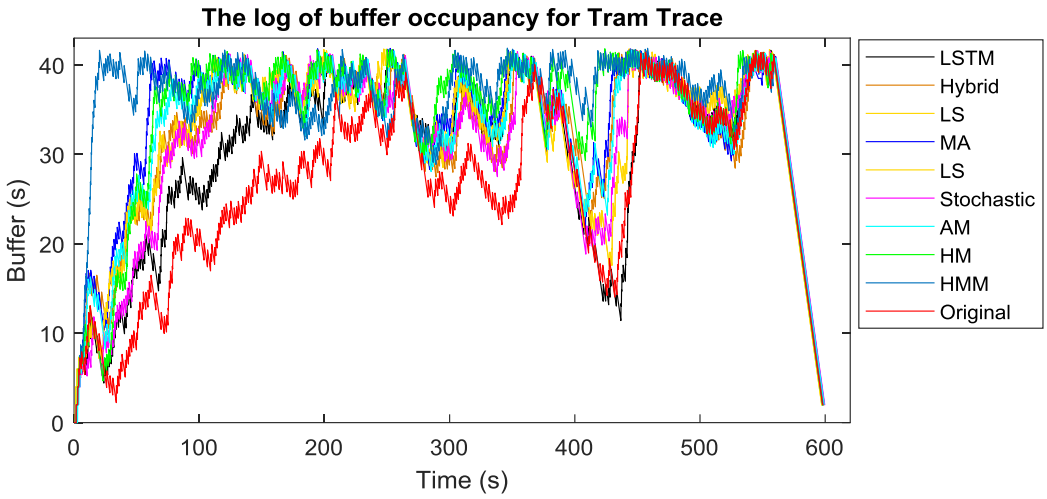


Figure 5.19 The log of buffer occupancy using different methods in Tram.

5.3.3 Discussion

It can be seen from Section 5.3.2, the accuracy of prediction will influence the performance of adaptive streaming a lot. The overestimate of throughput may make the selection too aggressive, resulting in the danger of rebuffering while underestimate of the throughput may decrease the user perceived quality. Furthermore, a good prediction is not enough to ensure high QoE. Since the rate-based algorithm always chooses the bitrate lower than the prediction throughput, the buffer occupancy may stay in a high level during the normal streaming. Figure 5.19 shows that the average buffer occupancy is about 60%~70% of B_{max}. Actually, the bitrate selection can be more aggressive if the buffer occupancy is high. Comparing the results of the LSTM and original methods in Tram case, it can be found that a good QoE is related to higher average bitrate and a lower buffer occupancy under the same network condition. Therefore, an additional term can be included in the ABR method. In the initial phase, overestimate could be a disaster since long waiting time may annoy the user a lot and may let the user give up the video session. This will cause a dramatic loss of the revenue of service provider. Therefore, in the future adaptive strategy design, the selection needs to be conservative in initial phase in order to establish the streaming as soon as possible.

5.4 Decision map method for adaptive bitrate control

5.4.1 Aggressive decision

From the evaluation results in Section 5.3.2 and the discussion in Section 5.3.3, it is obvious that there is needs to design new adaptive bitrate control algorithm not only based on the throughput prediction but also the buffer occupancy information since the rate-based ABR tends to be too conservative. If the throughput is the same as the bitrate, the buffer occupancy should keep the same because the downloaded video duration can balance the consuming time for downloading. If the throughput is lower than the bitrate, the buffer occupancy should decrease since it consumes more time in downloading. When the current buffer occupancy is large, we can choose the bitrate aggressively.

We proposed a decision map method with aggressive mechanism (DMM-A) for adaptive bitrate control incorporating both prediction and buffer occupancy information as shown in Figure 5.20. In this map, the x axis is the current buffer occupancy B_{cur} and y axis is an addition term ΔT_{DL} named extra downloading time (EDT). ΔT_{DL} is calculated using the following equation:

$$\Delta T_{DL} = [(R_{ind+1} / C_{pred}) - 1] \cdot T_{seg} \tag{5.4}$$

where C_{pred} is the throughput prediction using LSTM, $R_{ind+1}$is the bitrate one rank higher than the rate-based choice. T_{seg} is the duration of one segment. This term is used to estimate the possible extra downloading time when choosing the bitrate larger than the throughput prediction. B_{upper} is the maximum buffer occupancy and B_{agg} is a threshold for deciding when to be aggressive. ΔT_{upper} is another threshold in the EDT axis to judge the aggressive action. The area with red dots is the aggressive area and that with green dots is the normal area. The aggressive area is where the buffer occupancy is relatively high and the EDT is not very large.

The red dot area can be determined whether aggressively or conservatively. For simplicity and neutrality, it is drawn as a linear relationship here. The B_{agg} is chosen as 20 s which is the same as the lower limit of buffer occupancy. The smaller B_{agg} is, the more aggressive the map is. ΔT_{upper} is set as 2 s here since we expect one segment time is the largest tolerance for aggressive decision. The larger ΔT_{upper} is, the more aggressive the map is. The boundary between normal area and aggressive is a line from (B_{agg}, 0) to (B_{upper}, ΔT_{upper}). If the (ΔT_{DL}, B_{cur}) falls in the aggressive area, the bitrate will be chosen as one rank higher than the rate-based decision. By using this decision map, the choice of bitrate can be more aggressive than rate-based method. It can be expected, a higher average bitrate can be achieved.

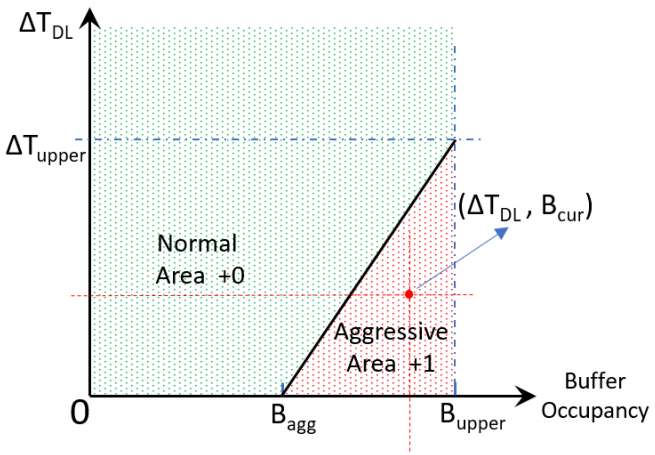


Figure 5.20 The illustration of the decision map with aggressive mechanism (DMM-A).

Table 5.4 shows the results of DMM-A with LSTM and original method. The corresponding buffer occupancy log is shown in Figure 5.21. As can be seen that, from the point view of total QoE, there seems to be no improvement than even LSTM. The score is almost the same. This is caused by a rebuffering event around 450 s. However, except for the rebuffering drawback, the buffer occupancy stays relatively stable near

20 s and the average bitrate is much higher than other two methods. DMM-A is indeed an aggressive method. But this method can be considered as a "controlled aggressive" case since the buffer occupancy is monitored. If the drawback of rebuffering can be solved, the performance of the DMM-A is expected to be more outstanding.

Table 5.4 The results of Tram Trace.

Methods	Metrics				
	QoE	R_{ave} (kbps)	T_{init} (s)	T_{rebuf} (s)	N_{rebuf}
LSTM	170.9	698.4	4.2	0.0	0
Original	181.9	716.8	3.9	0.0	0
DMM-A	169.0	800.0	4.6	6.4	1

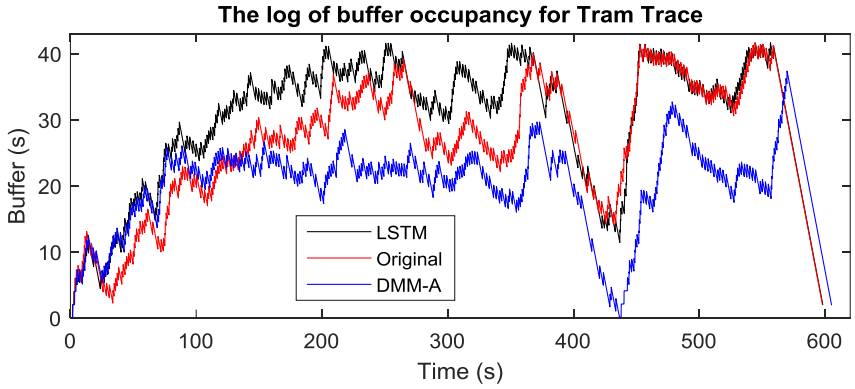


Figure 5.21 The log of buffer occupancy using different methods in Tram.

5.4.2 Aggressive decision with conservative mechanism

As discussed in the former section, it is necessary to deal with the possible rebuffering event in the DMM-A since the aggressive decision is made intentionally. Here, we extend the DMM-A with an additional conservative mechanism as shown in Figure 5.22. We just name this extended method as DMM since it involves both aggressive and conservative areas. In DMM, besides the division of normal and aggressive areas, the conservative area is added which is shown as blue dots. This area is determined by two thresholds, B_{con1} and B_{con2}. When the buffer occupancy is within the conservative area, no matter what the throughput prediction is, the conservative action should be taken immediately to avoid rebuffering events. B_{con1} and B_{con2} are set to indicate the emergency level of the situation. If the B_{cur} is lower than B_{con1} but higher than B_{con2}, the bitrate will be chosen as one rank lower than the rate-based decision. If

the B_{cur} is lower than B_{con2}, this is considered to be an extremely dangerous situation, therefore the bitrate will be chosen as two rank lower than the rate-based decision. The whole procedure of this method is shown in Algorithm II. Here, B_{con1} and B_{con2} are set as 10 s and 5 s, respectively. It can be expected that rebuffering events can be avoided using this conservative mechanism. Meanwhile, this conservative decision will also be applied to the initial period when the user starts the video session. It is expected this conservative decision can help reduce initial delay since there is no buffer at the beginning.

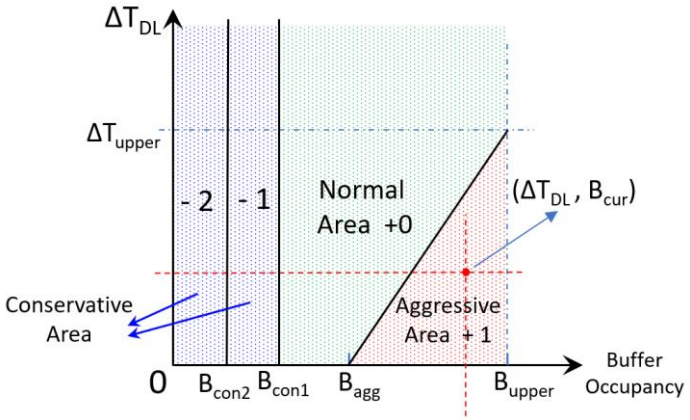


Figure 5.22 The illustration of the decision map method (DMM) with both aggressive and conservative areas.

Table 5.5 shows the results of DMM with LSTM, original, and DMM-A method. The corresponding buffer occupancy log is shown in Figure 5.23. As can be seen that, from the viewpoint of total QoE, the DMM performs best among all the methods. Besides the total QoE, the average bitrate is improved by 9% compared with the original method which was the best. Furthermore, the initial delay is reduced as expected thanks to the conservative action at the beginning. As shown in Figure 5.24(a), the choice of bitrate is much lower than the predicted throughput at the initial stage. The rebuffering event is also avoided during the bad network condition period. As show in Figure 5.24(b), the choice of bitrate is also very conservative since the buffer occupancy becomes low caused by the bad condition. This strategy helps the video session survive and play on without rebuffering. These results demonstrate that the DMM method can improve the QoE significantly in adaptive video transmission.

Algorithm II: Adaptive Bitrate Control using Decision Map Method

Input: throughput prediction: C_{pred} using LSTM, current buffer state: B_{cur}
Output: selected bitrate: R_{sel}

set thresholds: B_{agg}, B_{con1}, B_{con2}, ΔT_{upper};
if $C_{pred} < R_{\min}$ **then**
  $R_{sel} = R_{\min}$;
end
else
  $ind = 0$; initialize the selected rate index
  for each bitrate R_i in encoding rates $\{R\}$ **do**
    if $R_i < C_{pred}$ && $R_i > R_{i-1}$ **then**
      $ind = i$;
    end
  end
  decide whether take conservative strategy
  if $B_{cur} <= B_{con2}$ && $ind > 1$ **then**
    $ind = ind - 2$;
  end
  elseif $B_{cur} < B_{con1}$ && $B_{cur} > B_{con2}$ && $ind > 0$ **then**
    $ind = ind - 1$;
  end
  decide whether take aggressive strategy
  $\Delta T_{DL} = (R_{ind+1}/C_{pred} - 1) * T_{\text{seg}}$;
  if $(B_{cur} - B_{agg} / B_{upper} - B_{agg}) > (\Delta T_{DL} / \Delta T_{upper})$ && $R_{ind} < R_{max}$ **then**
    $ind = ind + 1$;
  end
  $R_{sel} = R_{ind}$
end

Table 5.5 The results of Tram Trace.

Methods	Metrics				
	QoE	R_{ave} (kbps)	T_{init} (s)	T_{rebuf} (s)	N_{rebuf}
LSTM	170.9	698.4	4.2	0.0	0
Original	181.9	716.8	3.9	0.0	0
DMM-A	169.0	800.0	4.6	6.4	1
DMM	195.4	779.8	3.2	0	0

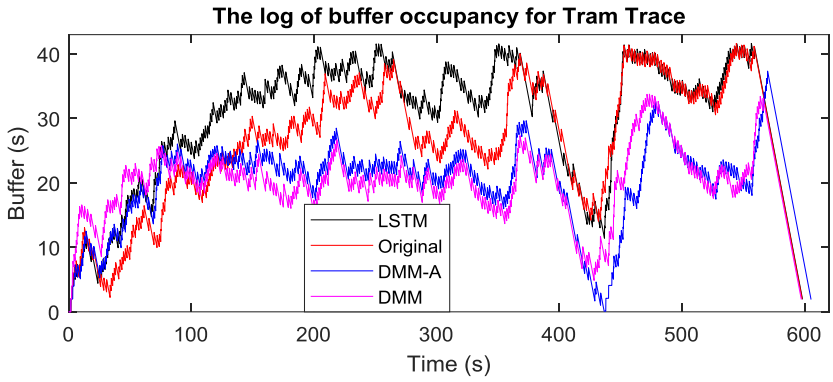


Figure 5.23 The log of buffer occupancy using different methods in Tram.

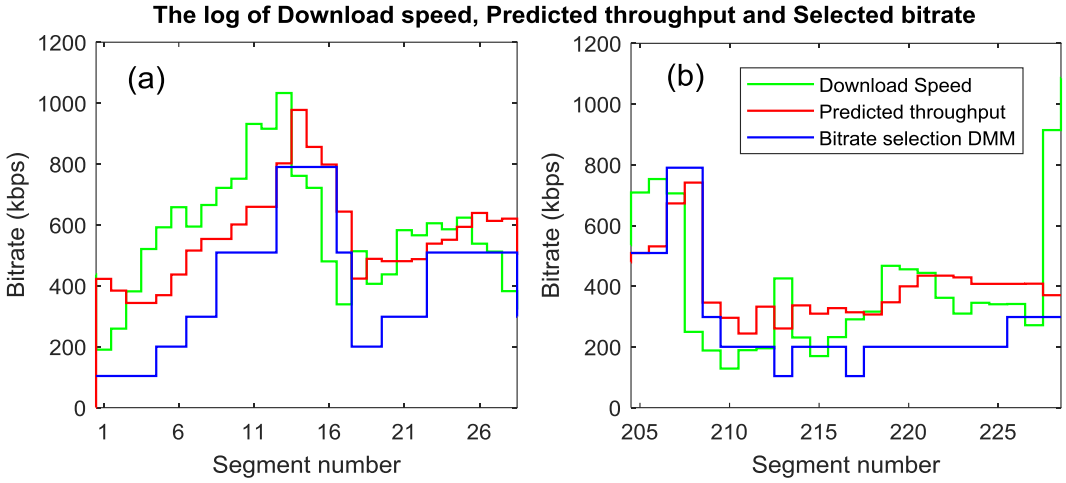


Figure 5.24 The log of bitrate status using DMM in Tram. (a) Initial stage, (b) Bad network condition period.

5.4.3 DMM performance verification in other traces

The performance of the DMM method is also verified in other traces. Table 5.6 shows the results of DMM with LSTM and original method in Ferry Trace. The corresponding buffer occupancy log is shown in Figure 5.25. As can be seen that, from the viewpoint of total QoE, the DMM performs best among all the methods. The average bitrate is improved significantly compared with the LSTM rate-based method. Meanwhile, the rebuffering time does not increase. It is expected that the DMM can also reduce the rebuffering event. However, in the ferry trace, there is a period when the network is suddenly cut off. Therefore, even the bitrate is chosen as the lowest one, the rebuffering event is not avoided. In such situation, other information should be

considered for preparation of sudden network cutoff. For example, there may be no-signal area in the ferry route. Based on the designated route, we can expect to know when we will enter the no-signal area. Then, the contents can be downloaded more than $B_{\max}$ before the network cutoff to go through the no-signal area.

Table 5.6 The results of Ferry Trace.

Methods	Metrics				
	QoE	R_{ave} (kbps)	T_{init} (s)	T_{rebuf} (s)	N_{rebuf}
LSTM	140.7	1042.1	2.4	32.4	1
Original	92.8	1458.8	5.6	68.5	5
DMM	185.9	1225.0	2.8	32.6	1

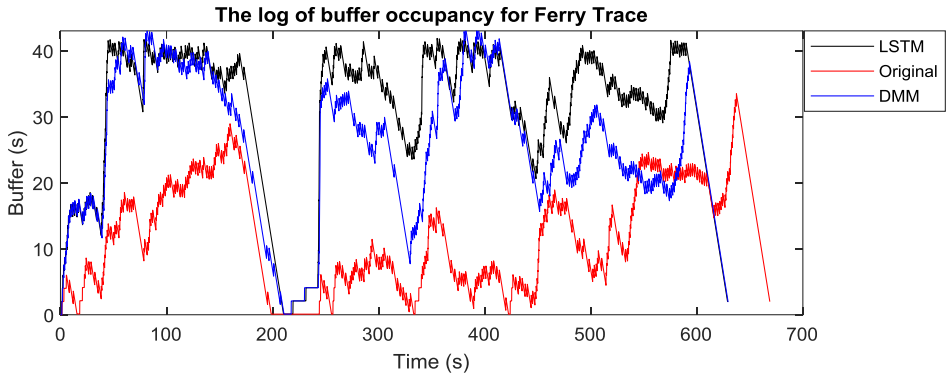


Figure 5.25 The log of buffer occupancy using different methods in Ferry.

Table 5.7 shows the results of DMM with LSTM and original method in Bus Trace. The corresponding buffer occupancy log is shown in Figure 5.26. As can be seen that, from the viewpoint of total QoE, the DMM performs best among all the methods. The average bitrate is improved significantly compared with the LSTM rate-based method. The initial delay is also reduced thanks to the conservative mechanism. From these results, it can be concluded that the DMM can significantly improve the QoE performance in DASH compared with conventional methods.

Table 5.7 The results of Bus Trace.

Methods	Metrics				
	QoE	R_{ave} (kbps)	T_{init} (s)	T_{rebuf} (s)	N_{rebuf}
LSTM	612.0	2263.3	6.3	0.0	0
Original	561.7	2648.4	14.1	34.1	4
DMM	653.6	2405.8	3.5	0.0	0

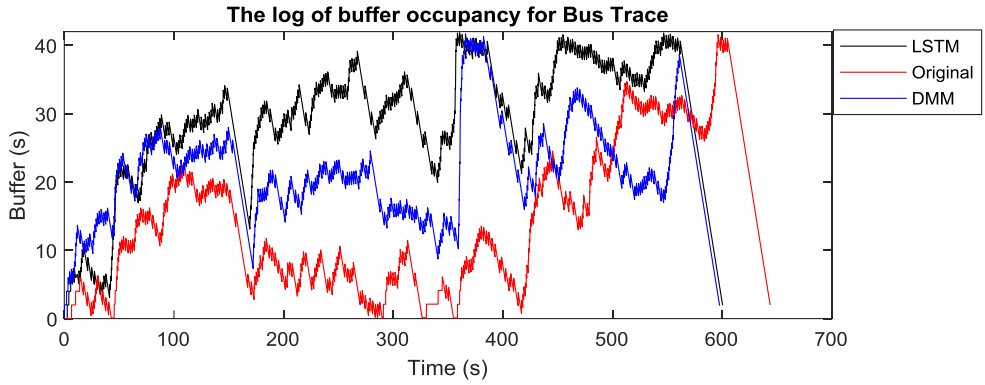


Figure 5.26 The log of buffer occupancy using different methods in Bus.

5.5 Summary

In this chapter, a video transmission system with DASH is established. The DASH-JS client is modified and extended for flexible usage. A trace-based server is built and proposed. The results demonstrate this server can create reproducible emulation environment according to the prepared trace, which allows the evaluation of algorithms effectively with limited experiments. We evaluate throughput prediction methods for adaptive bitrate control via trace-based emulation. The basic ABR strategy is rate-based method. By comparison of the results, it is found that the good throughput prediction is essential in achieving a good QoE performance. While, there are still space to further improve the QoE by incorporating the information of the buffer occupancy.

We also propose a new ABR algorithm name decision map method (DMM). This algorithm incorporates both throughput prediction and buffer occupancy information to make decision whether the aggressive or conservative bitrate selection is carried out. Through trace-based emulations in several traces, it is demonstrated that the DMM performs significantly well than conventional methods. The average bitrate is improved while no additional rebuffering event is encountered. Meanwhile, the initial delay is also reduced. The total QoE can be improved by 32.1% in the Ferry trace, showing the efficiency of the proposed ABR algorithm.

For further research, we will continue to improve the adaptive bitrate control algorithm for better handling different circumstances such as sudden network cutoff. We will also test the performance in more traces and deploy the DMM algorithm into real network environment.

6 Conclusions and future work

6.1 Conclusions

In order to provide fluent video streaming and improve the QoS and QoE for mobile users, throughput prediction methods and adaptive bitrate control model are proposed in this thesis.

In Chapter 3, throughput prediction methods using statistics and machine learning are proposed. An approach is proposed which utilizes HMM and total variance to evaluate the fluctuation of the former sequence, then uses linear prediction and locally weighted linear prediction to predict the future throughput. Based on this method, an advanced prediction model named the hybrid prediction with the autoregressive model and hidden Markov model (HOAH) is developed to predict TCP throughput. The method adopts support vector machine (SVM) as classifier, and switch between autoregressive model (AR model) and Gaussian mixture model-hidden Markov model (GMM-HMM) to predict future data. Evaluation shows the method can choose the proper prediction model correctly and predict throughput effectively.

In Chapter 4, a TCP throughput prediction method using long short-term memory (LSTM) model is proposed. The method is named throughput prediction based on LSTM (TRUST), which apply not only throughput measurements, but also other parameters as features to construct the neural network model. Field experiments are conducted to evaluate the method. Results show the method can decrease the prediction error by a maximum of 44% compared with conventional methods.

In Chapter 5, a trace-based emulation for Dynamic Adaptive Streaming over HTTP (DASH) is established to evaluate the effect of the throughput prediction methods on adaptive bitrate control. Results indicate a good prediction can contribute to good QoE performance. Moreover, a new adaptive bitrate control method named decision map method (DMM) is proposed. The evaluation results show that DMM can increase the average bitrate and avoid extra rebuffering event at the same time.

6.2 Future work

In the future, we will apply other methodologies such as reinforcement learning

to construct prediction model. Furthermore, parameters of lower layer of mobile network will be collected by developing new software and be adopted to predict future throughput. More measurements will be conducted to enlarge the database for improving the accuracy of the prediction model.

Meanwhile, more experiments will be conducted to evaluate the proposed ABR method. We will continue to improve the adaptive bitrate control algorithm for better handling different circumstances such as sudden network cutoff. We will also deploy the DMM algorithm into real network environment and test the performance. Based on the real-world deployment performance, other techniques will be utilized to improve the ABR method.

Bibliography

[1] X. Yin, A. Jindal, V. Sekar, and B. Sinopoli, "A control-theoretic approach for dynamic adaptive video streaming over HTTP," in *Proc. ACM SIGCOMM 2015*, pp. 325-338, 2015.

[2] X. K. Zou, J. Erman, V. Gopalakrishnan, E. Halepovic, R. Jana, X Jin, J. Rexford, and R. K. Sinha, "Can accurate predictions improve video streaming in cellular networks?" in *Proc. ACM HotMobile 2015*, pp. 57-62, 2015.

[3] H. Mao, R. Netravali, and M. Alizadeh, "Neural adaptive video streaming with pensieve," in *Proc. ACM SIGCOMM 2017*, pp. 197-210, 2017.

[4] J. Jiang, V. Sekar, and H. Zhang, "Improving fairness, efficiency, and stability in http-based adaptive video streaming with festive," *IEEE/ACM Transactions on Networking*, vol. 22, no. 1, pp. 326-340, Feb. 2014.

[5] J. Kua, G. Armitage, and P. Branch, "A survey of rate adaptation techniques for dynamic adaptive streaming over http," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1842-1866, third quarter 2017.

[6] K. Miller, A. Al-Tamimi, and A. Wolisz, "QoE-based low-delay live streaming using throughput predictions," *ACM Transactions on Multimedia Computing, Communications, and Applications*, vol. 13, no. 1, pp. 1–24, Jan. 2017.

[7] X. Yin, V. Sekar, and B. Sinopoli, "Toward a principled framework to design dynamic adaptive streaming algorithms over http," in *Proc. ACM HotNets 2014*, pp. 1-9, 2014.

[8] ITEC DASH-JS, [online]:

http://www-itec.uni-klu.ac.at/dash/?page_id=746.

[9] Y. Zheng, Y. Chen, Q. Li, X. Xie, and W. Ma, "Understanding transportation modes based on GPS data for web applications," *ACM Transactions on the Web*, vol. 4, no. 1, Jan. 2010.

[10] V. Manzoni, D. Maniloff, K. Kloeckl, and C. Ratti, "Transportation mode identification and real-time CO2 emission estimation using smartphones," *Massachusetts Institute of Technology*, Cambridge, MA, USA, Tech. Rep., 2010.

[11] P. Widhalm, P. Nitsche, and N. Brändie, "Transport mode detection with realistic smartphone sensor data," in *Proc. IEEE ICPR 2012*, pp. 573-576, 2012.

[12] A. Jahangiri and H. A. Rakha, “Applying machine learning techniques to transportation mode recognition using mobile phone sensor data,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 5, pp. 2406-2417, Oct. 2015.

[13] B. Nham, K. Siangliulue, and S. Yeung, “Predicting mode of transport from iPhone accelerometer data,” *Stanford University*, Stanford, CA, USA, Tech. Rep., 2008.

[14] H. I. Ashqar, M. H. Almannaa, M. Elhenawy, H. A. Rakha, and L. House, “Smartphone transportation mode recognition using a hierarchical machine learning classifier and pooled features from time and frequency domains,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 1, pp. 244-252, Jan. 2019 .

[15] X. Su, H. Caceres, H. Tong, and Q. He, “Online travel mode identification using smartphones with battery saving considerations,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 10, pp. 2921-2934, Oct. 2016.

[16] S. Wang, C. Chen, and J. Ma, “Accelerometer based transportation mode recognition on mobile phones,” in *Proc. IEEE APWCS 2010*, pp. 44-46, 2010.

[17] J. Suto, S. Oniga, C. Lung, and I. Orha, “Recognition rate difference between real-time and offline human activity recognition,” in *Proc. IEEE IoTGC 2017*, pp. 1-6, 2017.

[18] O. D. Lara and M. A. Labrador, “A survey on human activity recognition using wearable sensors,” *IEEE Communications Surveys & Tutorials,* vol. 15, no. 3, pp. 1192-1209, third quarter 2013.

[19] X. Su, H. Tong, and P. Ji, “Activity recognition using smartphone sensors,” *Tsinghua Science and Technology*, vol. 19, no. 3, pp. 235-249, Jun. 2014.

[20] Y. Chen and Y. Xue, “A deep learning approach to human activity recognition based on single accelerometer,” in *Proc. IEEE SMC 2015*, pp. 1488-1492, 2015.

[21] M. Gochoo, T. Tan, S. Huang, S. Liu, and F. S. Alnajjar, “DCNN-based elderly activity recognition using binary sensors,” in *Proc. IEEE ICECTA*, pp. 1-5, 2017.

[22] N. Bui, M. Cesana, S. A. Hosseini, Q. Liao, I. Malanchini, and J. Widmer, “A survey of anticipatory mobile networking: Context-based classification, prediction methodologies, and optimization techniques,” *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1790-1821, Apr. 2017.

[23] Y. T. Lin, E. M. R. Oliveira, S. B. Jemaa, and S. E. Elayoubi, “Machine learning for predicting QoE of video streaming in mobile networks,” in *Proc. IEEE ICC 2017*, pp. 1-6,

2017.

[24] K. Satoda, H. Yoshida, H. Ito, and K. Ozawa, “Adaptive video pacing method based on the prediction of stochastic TCP throughput,” in *Proc. IEEE GLOBECOM 2012*, pp. 1944-1950 2012.

[25] H. Yoshida, K. Satoda, and T. Murase, “Constructing stochastic model of TCP throughput on basis of stationarity analysis,” in *Proc. IEEE GLOBECOM 2013*, pp. 1544-1550, 2013.

[26] K. Nihei, H. Yoshida, N. Kai, D. Kanetomo, and K. Satoda, “QoE maximizing bitrate control for live video streaming on a mobile uplink,” in *Proc. IEEE ConTEL 2017*, pp. 91-98, 2017.

[27] B. Wei, M. Okano, K. Kanai, W. Kawakami, and J. Katto, “Throughput prediction using recurrent neural network model,” in *Proc. IEEE GCCE 2018*, pp. 107-108, 2018.

[28] B. Wei, W. Kawakami, K. Kanai, and J. Katto, “A history-based TCP throughput prediction incorporating communication quality features by support vector regression for mobile networks,” in *Proc. IEEE ISM 2017*, pp. 374-375, 2017.

[29] Q. He, C. Dovrolis, and M. Ammar, “On the predictability of large transfer TCP throughput,” in *Proc. ACM SIGCOMM 2005*, pp. 145-156, 2005.

[30] M. Mathis, J. Semke, J. Mahdavi, and T. Ott, “The macroscopic behavior of the TCP congestion avoidance algorithm,” in *Proc. ACM SIGCOMM 1997*, pp. 67-82, 1997.

[31] J. Padhye, V. Firoiu, D. F. Towsley, and J. F. Kurose, “Modeling TCP Reno performance: a simple model and its empirical validation,” *IEEE/ACM Transactions on Networking*, vol. 8, no. 2, pp. 133-145, Apr. 2000.

[32] S. Floyd, M. Handley, J. Padhye, and J. Widmer, “Equation-based congestion control for unicast applications,” in *Proc. ACM SIGCOMM 2000*, pp. 43-56, 2000.

[33] S. Vazhkudai, J. M. Schopf, and I. Foster. “Predicting the Performance of Wide Area Data Transfers,” in *Proc. IEEE IPDPS 2002*, pp. 1-10, 2002.

[34] M. Swany and R. Wolski, “Multivariate resource performance forecasting in the network weather service,” in *Proc. ACM/IEEE SC 2002*, pp. 1-10, 2002.

[35] F. Dobrian, V. Sekar, A. Awan, I. Stoica, D. Joseph, A. Ganjam, J. Zhan, and H. Zhang, “Understanding the impact of video quality on user engagement,” in *Proc. ACM SIGCOMM 2011*, pp. 362-373, 2011.

[36] Y. Liu, S. Dey, F. Ulupinar, M. Luby, and Y. Mao, “Deriving and validating user experience model for DASH video streaming,” *IEEE Transactions on Broadcasting*, vol. 61, no. 4, pp. 651-665, Dec. 2015.

[37] Adobe, “Adobe HTTP Dynamic Streaming (HDS),” 2016. [Online]: https://www.adobe.com/devnet/hds.html.

[38] Apple, “Apple HTTP Live Streaming,” 2016. [Online]: https://developer.apple.com/streaming/.

[39] Microsoft, “Microsoft Silverlight Smooth Streaming,” 2016. [Online]: https://www.microsoft.com/silverlight/smoothstreaming/.

[40] ISO/IEC, “ISO/IEC 23009-1:2014 Information Technology: Dynamic Adaptive Streaming over HTTP (DASH) Part 1: Media presentation description and segment formats,” 2014. [Online]. Available: https://www.iso.org/standard/65274.html.

[41] T. Huang, R. Johari, N. McKeown, M. Trunnel, and M. Watson, “A buffer-based approach to rate adaptation: Evidence from a large video streaming service,” in *Proc. ACM SIGCOMM 2014*, pp. 187-198, 2014.

[42] K. Spiteri, R. Urgaonkar and R. K. Sitaraman, “BOLA: Near-optimal bitrate adaptation for online videos,” in *Proc. IEEE INFOCOM 2016*, pp. 1-9, 2016.

[43] B. Rainer, S. Lederer, C. Muller, and C. Timmerer, “A seamless web integration of adaptive HTTP streaming,” in *Proc. IEEE EUSIPCO 2012*, pp. 1519–1523, 2012.

[44] I. Sodagar, “The MPEG-DASH standard for multimedia streaming over the internet,” *IEEE Multimedia*, vol. 18, no. 4, pp. 62-67, Oct. 2011.

[45] Z. Wang and J. Crowcroft, “Quality-of-service routing for supporting multimedia applications,” *IEEE Journal on Selected Areas in Communications*, vol. 14, no. 7, pp. 1228-1234, Sep. 1996.

[46] U. Reiter, K. Brunnström, K. D. Moor, M. C. Larabi, M. Pereira, A. Pinheiro, J. You, and A. Zgank, “Factors Influencing Quality of Experience,” *Quality of Experience*, Springer, Cham, pp. 55–74, 2014.

[47] P. A. Gagniuc, “Markov Chains: From Theory to Implementation and Experimentation,” John Wiley & Sons, 2017.

[48] HSDPA DATASET, [online]:

http://home.ifi.uio.no/paalh/dataset/hsdpa-tcp-logs/.

[49] B. Wei, K. Kanai, and J. Katto, “History-based throughput prediction with Hidden Markov Model in mobile networks,” in *Proc. IEEE ICMEW 2016*, pp. 1-6, 2016.

[50] D. A. Reynolds, “Speaker identification and verification using Gaussian mixture speaker models,” *Speech communication*, vol. 17, no. 1-2, pp. 91-108, Aug. 1995.

[51] Y. Huang, K. B. Englehart, B. Hudgins, and A.D.C. Chan, “A Gaussian mixture model based classification scheme for myoelectric control of powered upper limb prostheses,” *IEEE Transactions on Biomedical Engineering*, vol. 52, no. 11, pp. 1801-1811, Nov. 2005.

[52] H. Li, G. Hua, Z. Lin, J. Brandt, and J. Yang, “Probabilistic elastic matching for pose variant face verification,” in *Proc. IEEE CVPR 2013*, pp. 3499-3506, 2013.

[53] J. Gauvain and C. Lee, “Maximum a posteriori estimation for multivariate Gaussian mixture observations of Markov chains,” *IEEE Transaction on Speech and Audio Processing*, vol. 2, no. 2, pp. 291-298, Apr. 1994.

[54] J. Ajmera, I. McCowan, and H. Bourlard, “Speech/music segmentation using entropy and dynamism features in a HMM classification framework,” *Speech Communication*, vol. 40, no. 3, pp. 351-363, May 2003.

[55] C. E. Pertsinidou, and N. Limnios, “Viterbi algorithms for Hidden semi-Markov models with application to DNA analysis,” *RAIRO-Operations Research*, vol. 49, no. 3, pp. 511-526, Sep. 2015.

[56] C. A. Greenhall, D. A. Howe, and D. B. Percival, “Total variance, an estimator of long-term frequency stability [standards],” *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control*, vol. 46, no. 5, pp. 1183-1191, Sep. 1999.

[57] S. J. Rao, “Regression Modeling Strategies: With Applications to Linear Models, Logistic Regression, and Survival Analysis,” Journal of the American Statistical Association, vol. 98, no. 461, pp. 257-258, 2005.

[58] O. Fedotova, L. Teixeira, and H. Alvelos, “Software effort estimation with multiple linear regression: Review and practical application,” *Journal of Information Science and Engineering*, vol. 29, no. 5, pp. 925-945, 2013.

[59] I. Naseem, R. Togneri, and M. Bennamoun, “Linear regression for face recognition,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 11, pp. 2106-2112, Nov. 2010.

[60] S. Dabiri and K. Heaslip, “Inferring transportation modes from GPS trajectories using a convolutional neural network,” *Transportation Research Part C: Emerging Technologies*, vol. 86, pp. 360-371, Jan. 2018.

[61] Y. Qiao, Y. Cheng, J. Yang, J. Liu, and N. Kato, “A Mobility Analytical Framework for Big Mobile Data in Densely Populated Area,” *IEEE Transactions on Vehicular Technology*, vol. 66, no. 2, pp. 1443-1455, Feb. 2017.

[62] W. Kawakami, K. Kanai, B. Wei, and J. Katto, “Accuracy evaluations of human moving pattern using communication quality based on machine learning,” in *Proc. IEEE GCCE 2017*, pp. 1-2, 2017.

[63] B. Wei, K. Kanai, W. Kawakami, and J. Katto, “HOAH: A hybrid TCP throughput prediction with autoregressive model and hidden markov model for mobile networks,” *IEICE Transactions on Communications*, vol. E101-B, no. 7, pp. 1612–1624, 2018.

[64] S. Takenaka, K. Kanai, J. Katto, and T. Murase, “Green Video Delivery System using Moving Route Navigation and Playout Buffer Control,” in *Proc. IEEE CCNC 2017*, pp. 1-4, 2017.

[65] M. E. Ayadi, Moataz, M. S. Kamel, and F. Karray, “Survey on speech emotion recognition: Features, classification schemes, and databases,” *Pattern Recognition*, vol. 44, no. 3, pp. 572-587, 2011.

[66] M. N. Ayyaz, I. Javed, and W. Mahmood, “Handwritten Character Recognition Using Multiclass SVM Classification with Hybrid Feature Extraction,” *Pakistan Journal of Engineering and Applied Sciences*, vol. 10, pp. 57-67, 2012.

[67] U. Thissen, R. van Brakel, A.P. de Weijer, W.J. Melssen, and L.M.C. Buydens, “Using support vector machines for time series prediction,” *Chemometrics and Intelligent Laboratory Systems*, vol. 69, no. 1, pp. 35-49, Nov. 2003.

[68] N. I. Sapankevych and R. Sankar, “Time Series Prediction Using Support Vector Machines: A Survey,” *IEEE Computational Intelligence Magazine*, vol. 4, no. 2, pp. 24-38, May 2009.

[69] A. Kampouraki, G. Manis, and C. Nikou, “Heartbeat Time Series Classification with Support Vector Machines,” *IEEE Transactions on Information Technology in Biomedicine*, vol. 13, no. 4, pp. 512-518, Jul. 2009.

[70] L. E. Nieto-Barajas and F. A. Quintana, “A Bayesian Non-Parametric Dynamic AR Model for Multiple Time Series Analysis,” *Journal of Time Series Analysis*, vol. 37, no. 5, pp. 675-

689, 2016.

[71] G. Inoussa, H. Peng, and J. Wu, “Nonlinear time series modeling and prediction using functional weights wavelet neural network-based state-dependent AR model,” *Neurocomputing*, vol. 86, pp. 59-74, 2012.

[72] D. A. Dickey and W. A. Fuller, “Distribution of the estimators for autoregressive time series with a unit root,” *Journal of the American Statistical Association*, vol. 74, no. 366, pp. 427-431, Jun. 1979.

[73] A. W. Gregory and B. E. Hansen, “Residual-based tests for cointegration in models with regime shifts,” *Journal of Econometrics*, vol. 70, no. 1, pp. 99-126, Jan. 1996.

[74] O. P. Chimobi, “Government expenditure and national income: A causality test for Nigeria,” *European Journal of Economic and Political Studies*, vol. 2, no. 2, pp. 1-11, 2009.

[75] S. V. Vaseghi, “Advanced digital signal processing and noise reduction,” John Wiley & Sons, 2008.

[76] M. S. Crouse, R. D. Nowak, and R. G. Baraniuk, “Wavelet-based statistical signal processing using hidden Markov models,” *IEEE Transactions on Signal Processing*, vol. 46, no. 4, pp. 886-902, Apr. 1998.

[77] C. L. P. Lim, W. L. Woo, S. S. Dlay, and B. Gao, “Heartrate-dependent heartwave biometric identification with thresholding-based GMM–HMM methodology,” *IEEE Transactions on Industrial Informatics*, vol. 15, no. 1, pp. 45-53, Jan. 2019.

[78] D. Guo, W. Zhou, H. Li, and M. Wang, “Online early-late fusion based on adaptive HMM for sign language recognition,” *ACM Transactions on Multimedia Computing, Communications, and Applications*, vol. 14, no. 1, Jan. 2018.

[79] Z. Shi, C. Yang, M. Hao, X. Wang, R.D. Ward, and A. Zhang, “FuzzyID2: A software package for large data set species identification via barcoding and metabarcoding using hidden Markov models and fuzzy set methods,” *Molecular ecology resources*, vol. 18, no, 3, pp. 666-675, May 2018.

[80] D. Park, L. R. Rilett, and G. Han, “Spectral basis neural networks for real-time travel time forecasting,” *Journal of Transportation Engineering*, vol. 125, no. 6, pp. 515-523, Nov. 1999.

[81] W. Kawakami, K. Kanai, B. Wei, and J. Katto, “Machine learning based transportation modes recognition using mobile communication quality,” in *Proc. IEEE ICME 2018*, pp. 1-

6, 2018.

[82] W. Kawakami, K. Kanai, B. Wei, and J. Katto, “A highly accurate transportation mode recognition using mobile communication quality,” *IEICE Transactions on Communications*, 2019, in press.

[83] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and É. Duchesnay, “Scikit-learn: Machine learning in Python,” *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[84] T. Mikolov, S. Kombrink, L. Burget, J. Černocký, and S. Khudanpur, “Extensions of recurrent neural network language model,” in *Proc. IEEE ICASS 2011*, pp. 5528-5531, 2011.

[85] M. Sundermeyer, H. Ney, and R. Schlüter, “From feedforward to recurrent LSTM neural networks for language modeling,” *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 23, no. 3, pp. 517-529, Mar. 2015.

[86] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.

[87] D. P. Kingma and J. L. Ba, “Adam: A method for stochastic optimization,” in *Proc. ICRL*, pp. 1–41, 2015.

[88] Y. Sun, X. Yin, J. Jiang, V. Sekar, F. Lin, N. Wang, T. Liu, and B. Sinopoli, “Cs2p: Improving video bitrate selection and adaptation with data-driven throughput prediction,” in *Proc. ACM SIGCOMM 2016*, pp. 272-285, 2016.

[89] Y. Liu and J.Y. B. Lee, “An empirical study of throughput prediction in mobile data networks,” in *Proc. IEEE GLOBECOM 2015*, pp. 1-6, 2015.

[90] B. Wei, S. Wang, W. Kawakami, K. Kanai, and J. Katto, “TRUST: A TCP Throughput Prediction Method in Mobile Networks,” in *Proc. IEEE GLOBECOM 2018*, pp. 1-6, 2018.

[91] B. Wei, H. Song, S. Wang, K. Kanai, and J. Katto, “Evaluation of Throughput Prediction for Adaptive Bitrate Control using Trace-based Emulation,” *IEEE Access*, submitted.

Publication Lists

JOURNAL PAPERS

[1] W. Kawakami, K. Kanai, **B. Wei**, and J. Katto, "A highly accurate transportation mode recognition using mobile communication quality," *IEICE Transactions on Communications*, 2019, in press.

[2] K. Kanai, **B. Wei**, Z. Cheng, M. Takeuchi, and J. Katto, "Methods for adaptive video streaming and picture quality assessment to improve QoS/QoE performances," *IEICE Transactions on Communications*, July 2019. (invited)

[3] **B. Wei**, K. Kanai, W. Kawakami, and J. Katto, "HOAH: A hybrid TCP throughput prediction with autoregressive model and hidden markov model for mobile networks," *IEICE Transactions on Communications*, vol. E101-B, no. 7, pp. 1612–1624, 2018.

[4] **B. Wei**, H. Song, S. Wang, K. Kanai, and J. Katto, "Evaluation of Throughput Prediction for Adaptive Bitrate Control using Trace-based Emulation," *IEEE Access*, submitted.

INTERNATIONAL CONFERENCE PAPERS

[5] **B. Wei**, S. Wang, W. Kawakami, K. Kanai, and J. Katto, "TRUST: A TCP Throughput Prediction Method in Mobile Networks," in *Proc. IEEE GLOBECOM 2018*, pp. 1-6, 2018.

[6] **B. Wei**, M. Okano, K. Kanai, W. Kawakami, and J. Katto, "Throughput prediction using recurrent neural network model," in *Proc. IEEE GCCE 2018*, pp. 107-108, 2018.

[7] W. Kawakami, K. Kanai, **B. Wei** and J. Katto, "Machine learning based transportation modes recognition using mobile communication quality," in *Proc. IEEE ICME 2018*, pp. 1-6, 2018.

[8] **B. Wei**, W. Kawakami, K. Kanai, and J. Katto, "A history-based TCP throughput prediction incorporating communication quality features by support vector regression for mobile networks," in *Proc. IEEE ISM 2017*, pp. 374-375, 2017.

[9] W. Kawakami, K. Kanai, **B. Wei,** and Jiro Katto, “Accuracy evaluations of human moving pattern using communication quality based on machine learning,” in *Proc. IEEE GCCE 2017*, pp. 1-2, 2017.

[10] **B. Wei**, K. Kanai and Jiro Katto, “History-based throughput prediction with Hidden Markov model in mobile network,” in *Proc. IEEE ICMEW 2016*, pp. 1-6, 2016.

DOMESTIC CONFERENCE PAPERS

[11] 川上航・金井謙治・**Bo Wei**・甲藤二郎: “CNN を活用したモバイルアプリケーション利用時のユーザ移動状態推定の精度評価,” 信学ソ大、Sep. 2018.

[12] **Bo Wei**, Kenji Kanai, Wataru Kawakami, and Jiro Katto, “Throughput Prediction Method based on machine learning in Mobile Networks,” *IEICE Tech Report*, NS-2018-42, Jul. 2018.

[13] 川上航・金井謙治・**Wei Bo**・甲藤二郎: “通信品質を利用した CNN によるユーザ移動状態推定の精度評価,” 信学会 NS 研究会, NS-2018-65, Jul. 2018.

[14] **Bo Wei**, Kenji Kanai, Wataru Kawakami, and Jiro Katto, “Machine learning-based throughput prediction using communication quality in mobile networks,” 信学会 MoNA 研究会, Jan.2018.

[15] 川上航・金井謙治・**Wei Bo**・甲藤二郎: “通信品質を用いた機械学習に基づくユーザの移動状態推定,” 信学会 MoNA 研究会, Jan.2018.

[16] 川上航・金井謙治・**Wei Bo**・甲藤二郎: “モバイルセンシングと機械学習を用いた通信品質に基づくユーザ行動推定,” 信学会 CS 研究会, Jul.2017.

[17] **Bo Wei**, Kenji Kanai, and Jiro Katto, “Throughput prediction by combining Autoregressive Model and Hidden Markov Model,” In *IEICE General Conference 2017*, Mar. 2017.

[18] **Bo Wei**, Kenji Kanai, and Jiro Katto, “Performance evaluations of history-based throughput prediction with trend analysis for mobile network,” In *IEICE Society Conference 2016*, Sep. 2016. (awarded)

[19] **Bo Wei**, Kenji Kanai, and Jiro Katto, “Throughput prediction based on Hidden Markov Model in mobile network,” In *IEICE General Conference 2016*, Mar. 2016.

[20] **Bo Wei**, Kenji Kanai, Sakiko Takenaka, and Jiro Katto, “Throughput prediction based on stochastic model of mobile network,” In *IEICE Society Conference 2015*, Sep. 2015. (awarded)