Socially-Driven Learning-Based Prefetching in Mobile Online Social Networks

Chao Wu, *Student Member, IEEE*, Xu Chen, *Member, IEEE*, Wenwu Zhu, *Fellow, IEEE*, and Yaoxue Zhang

***Abstract*— Mobile online social networks (OSNs) are emerging as the popular mainstream platform for information and content sharing among people. In order to provide the quality of experience support for mobile OSN services, in this paper, we propose a socially-driven learning-based framework, namely *Spice*, for the media content prefetching to reduce the access delay and enhance mobile user's satisfaction. Through a large-scale data-driven analysis over real-life mobile Twitter traces from over 17 000 users during a period of five months, we reveal that the social friendship has a great impact on user's media content click behavior. To capture this effect, we conduct the social friendship clustering over the set of user's friends, and then develop a cluster-based Latent Bias Model for socially-driven learning-based prefetching prediction. We then propose a usage-adaptive prefetching scheduling scheme by taking into account that different users may possess heterogeneous patterns in the mobile OSN app usage. We comprehensively evaluate the performance of Spice framework using trace-driven emulations on smartphones. Evaluation results corroborate that the Spice can achieve superior performance, with an average 80.6% access delay reduction at the low cost of cellular data and energy consumption. Furthermore, by enabling users to offload their machine learning procedures to a cloud server, our design can achieve up to a factor of 1000 speed-up over the local data training execution on smartphones.**

***Index Terms*— Mobile computing, online social network, multimedia applications, quality of experience.**

I. Introduction

THE past decade has witnessed the wide penetration of online social networks (OSNs) such as Facebook and Twitter into our daily lives. With the pervasivity and popularity of wireless communication such as WiFi and LTE, more and more users are accessing OSN services on mobile devices via wireless connection. It is reported that nowadays 68% of the OSN service consumptions occur on mobile devices [2], and on average a mobile user spends 2 hours and 25 minutes per day using OSN services, accounting for more than 20% of the overall mobile traffic [3].

Besides serving as the platform for social interaction, OSN is emerging as the mainstream channel for information and content sharing. For instance, over 52% and 47% of the users get news from Twitter and Facebook, respectively [4]. Moreover, a significant part of the shared content contains media files such as images and videos, which typically have much larger data size than that of the text content in users' posts. The increasing popularity and ubiquity of such media content in OSN calls for a mobile-friendly design in order to provide QoE support for mobile devices.

A key factor of degrading the mobile user's satisfaction in consuming rich OSN media content the access delay (service latency). On one hand, limited network bandwidth, high wireless connection establishment latency and long roundtrip time of data transmission (varying from 3 seconds to 10 seconds or more [5]) would impair the real-time responsiveness of users' daily social media usages, in particular when users try to access media files in social posts/tweets. On the other hand, time-varying network quality and sporadic network availability cause fluctuating connection and intermittent access. This would also incur excessive latency overhead for their social interaction engagement in OSNs.

To address this issue, an intriguing and promising approach is to leverage prefetching, i.e., to download the media content prior to user's consumption whenever possible [6]. A key challenge to exploit the benefit of prefetching is the precise prediction of media content download behavior. Achieving accurate content prediction can help to prefetch the most relevant content items which will be consumed by the user in the near future with high probability. This is useful to significantly reduce the access delay and meanwhile saving both energy and data traffic consumption by avoiding excessive content prefetching.

To boost the prediction accuracy of media content prefetching in OSNs on mobile devices, a very recent study in [7] proposed a framework of EarlyBird. The key idea is, by mining the user's OSN usage pattern, to integrate tweet training features (e.g., image embedded or not, the specified recipient) into the linear regression model for prediction. A key drawback of the proposed approach in [7] is that it does not provide sufficient consideration for social influence among the users (i.e., social interaction patterns), which plays a critical role in media content consumption in OSNs [8]. Intuitively, if a tweet with an image is sent from her close friend rather than some

Manuscript received December 19, 2015; revised October 8, 2016; accepted March 8, 2017; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor L. Qiu. This work was supported in part by the International Science and Technology Cooperation Program of China under Grant 2013DFB10070 and in part by the Tsinghua University Initiative Scientific Research Program under Grant 20161080066. Preliminary results of this paper has been presented in [1].

C. Wu and W. Zhu are with the Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China (e-mail: cwu12@mails.tsinghua.edu.cn; wwzhu@mail.tsinghua.edu.cn).

X. Chen is with the School of Data and Computer Science, Sun Yat-Sen University, Guangzhou 510275, China (e-mail: chenxu35@mail.sysu.edu.cn).

Y. Zhang is with the School of Information Science and Engineering, Central South University, Changsha 410012, China (e-mail: zyx@csu.edu.cn).

Digital Object Identifier 10.1109/TNET.2017.2681121

acquaintance with infrequent contact, then she would click the image with a high probability (see Table II).

Motivated by this insight, in this paper we propose a novel framework of *Spice*, which utilizes the unique characteristics of social interactions among users in OSNs for mobile media prefetching. To this end, we leverage the tools of socially-driven data mining and cluster-based machine learning, to infer a user's potential interest in media content consumption based on her history content usage pattern and social friendship preference. Moreover, we combine such inference, app usage pattern, and network environment, to jointly execute the whole prefetching decision and scheduling procedure.

Specifically, we collect user traces from Twidere, an Android Twitter app which has close to 500,000 downloads on Google Play [9] and over 17,000 users consented to report usage data to us (see Section II-C). This enables us to conduct a data-driven analysis and design, and a realistic trace-driven performance evaluation (see Section VI). First of all, through the large-scale data analysis, we reveal that the social friendship (i.e., the social interaction strength among users in OSN) has a critical impact on the user's tweet click behavior. Based on this observation, we then conduct the social friendship clustering to classify a user's social friends into different groups with different levels of importance. Accordingly, we next develop a cluster-based Latent Bias Model (LBM) to estimate her likelihood of media content clicking. In order to guide the media tweet prefetching in an energy and cellular data efficient manner, we judiciously design an adaptive scheduling scheme, accounting for the fact that different users may show significantly different habits/patterns in the mobile OSN app usage. In addition, our design also enables Spice users to offload their data training tasks for machine learning to a cloud server, in order to combat the high energy consumption and long processing latency when executing these tasks locally on the smartphones.

We summarize the major contributions of this paper as follows:

- We collect a large set of real-life mobile Twitter traces from over 17,000 Twidere users during a period of five months, and reveal the great impact of social friendship on their media content click behavior through data-driven analysis.
- We conduct social friendship clustering over the set of user's friends, and then accordingly develop the cluster-based LBM approach for socially-driven prefetching prediction. Trace-driven emulation shows that our proposed approach achieves an average prediction accuracy of 84.5%, which significantly outperforms the linear regression approach using tweet training features only.
- We develop a usage-adaptive prefetching scheduling scheme to account for heterogeneous users' mobile app usage pattern. In particular, we partition the horizon of the whole minutes of day into several period zones and tune different prefetching frequencies for different zones adaptively.
- We comprehensively evaluate the performance of the Spice framework using trace-driven emulations on smartphones. Evaluation results show that an average Spice user can reduce her access delay by 80.6% at the low cost of cellular data and energy consumption, which is a significant improvement over the benchmark approaches. Moreover, by enabling users to offload machine learning procedures to a cloud server, we can achieve a speed-up of a factor of 1000 over the local execution on smartphones.

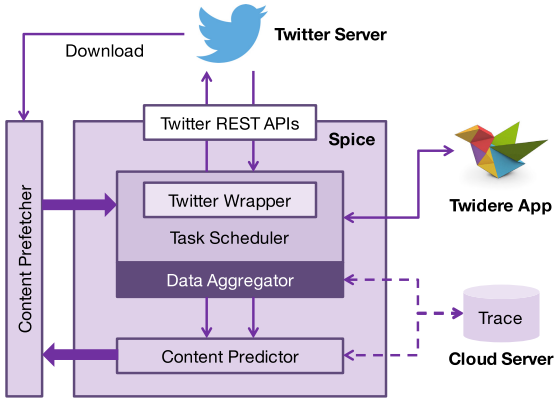


Fig. 1. Spice architecture.

The rest of this paper is organised as follows: Section II outlines system overview of Spice, and describes how we conduct data collection through the Twidere app. Section III analyses the social interaction among users and reveals its impact on the user's tweet click behavior. Section IV proposes the socially-driven learning-based prefetching prediction mechanism. Section V tackles the problem of prefetching scheduling. Section VI conducts the trace-driven emulation evaluation on smartphones. Section VIII reviews the related work, and Section IX concludes this paper.

II. Spice Overview

A. Spice Architecture

We now introduce the system architecture of Spice for media content prefetching in mobile OSNs. As illustrated in Fig. 1, Spice works in a user-centric manner (i.e., implemented on a user's mobile device), and collects traces about all tweets on the user's feed when accessing Twitter with the Twidere app [9]. These traces were retrieved using the Twitter REST API [10], located in the Twitter Wrapper, which is controlled by the *Task Scheduler* component to periodically query for new tweets on her newsfeed (see Section V).

Then the retrieved tweets and user information are passed to the *Data Aggregator* component. To ensure the user privacy, text content in tweets are not recorded and the anonymization of all personal data-related fields will carried out before directly storing the data on the mobile device. Later, the locally stored data is uploaded to the cloud server only for further analysis when the mobile device is charging and connecting with WiFi.

The Data Aggregator also passes the received information to the *Content Predictor* component, where the learning-based content prediction model is trained for predicting the likelihood whether she would click the media in a new tweet. Specifically, this predictor would take the user's new tweets

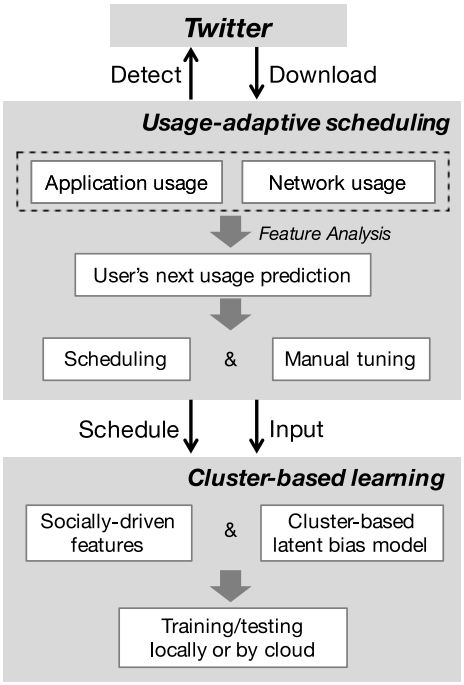


Fig. 2. Logical workflow of the Spice mobile media prefetching system.

and the relevant features of these tweets as an input to a machine learning model, in order to identify the relevant media content (e.g., image files) contained in these tweets as the prefetch candidates. These media files are then to be prefetched by the *Content Prefetcher* component. Note that, to speed up the whole process, we offload the machine learning procedure to a cloud server. When such a cloud server is not available, we can carry it out on the mobile device locally.

B. Logic Workflow of Spice

We then show the Logic workflow of Spice framework in Fig. 2 to illustrate how Spice works in more details when fresh media contents are going to be prefetched. As what we described above, Spice works in a user-centric manner and is implemented at user side to serve as a middleware intelligent library between the content context and user's prefetching requirements. A mobile app of OSNs, e.g., Twitter, Facebook, or WeChat etc, can interact with Spice with single third-party API, judiciously rank social media files based on the result of fully learning with one user's network utilities, app usage activeness, and context- or social-based preference.

Specifically, the Logic workflow of Spice consists of the following two components, i.e., usage-adaptive scheduling and cluster-based learning. The goal here is to judiciously decide when should the prefetching task be invoked, and then intelligently use a learning-based mechanism to guide what social media files should be prefetched. In particular, we conclude the whole prefetching mechanism as:

- **Learning**. As Spice is a socially-driven implementation for mobile media content prefetching. It is very important to take care of the social friendship influence, context preference, and OSN media attributes. Towards this target, we develop a socially-driven learning-based algorithm which would be impacted by the social friendship and context features. We also elaborate how significant the learning-based mechanism to show the effectiveness and correctness of our algorithm (see Section III and Section IV).
- **Scheduling**. In Spice, we define the prefetching task should be not only automatic but also usage-adaptive, which leads to critical cellular data flow and battery efficiency requirements on a preliminary that media files' loading delay can be guaranteed (see Section V). Namely, it is very important to decide when to prefetch according to user's profile, network usage preference, application usage activeness, and the manual tuning factors. We will elaborate this strategy later.

TABLE I
DATA COLLECTED FROM TWIDERE APP ON GOOGLE PLAY

Events	Content
App Launch and Close	Timestamp
Network Availablity	Timestamp, connection pattern
Tweets Click	Timestamp, tweet's attributes, participators
Embedded Media Preview	Timestamp, tweet ID, preview URL
Media Click	Timestamp, tweets ID list, link URL
Coarse Location	Timestamp, coordinates
Others	Tweets' favorite, retweet, and publish

C. Data Collection

As mentioned above, we collect data traces from the users using Twidere app. This is because, although Twitter's contents are publicly available, information about when, how, and where they access these social streams are not available in particular in the mobile environment. Therefore, we collected a large set of usage data from Twidere users[1] who agreed to provide their information to us anonymously.

As the aim is to enable intelligent prefetching by identifying the tweets that the user is most interested in, a set of tweet attributes are collected as well. To this end, the Twitter Wrapper tracks the user interaction information (e.g., retweet, favourite, or mention) of the individual tweets. The source of a tweet is also recorded by identifying whether the tweet is obtained from a direct friend or propagated through friends of others' friends. Furthermore, with the consent from the user, the Twidere app enables us to keep track of her activity events when reading the tweets, e.g., watching, liking, or commenting along the timeline. The collected trace items are shown in Table I.

During five months from March 2015 to July 2015, we have collected data traces from more than 17,000 users from all over the world with a diverse demographic composition. Fig. 3 shows the geographical distribution of users in our collected dataset. The users refreshed 238 million tweets and clicked 9.6 million of them in total. Also, 72% of the

[1]Twidere discloses the usage statistics information during its installation and update. Usage statistics only comes in effect after users choose to opt in. Up to now, there are around 23% of users grant us permissions, which indicates user privacy-awareness and the effectiveness of the privacy disclosure. In addition, we only keep the user data in a numerical and anonymous style. Last, twitter contents and social graph are publicly available information.

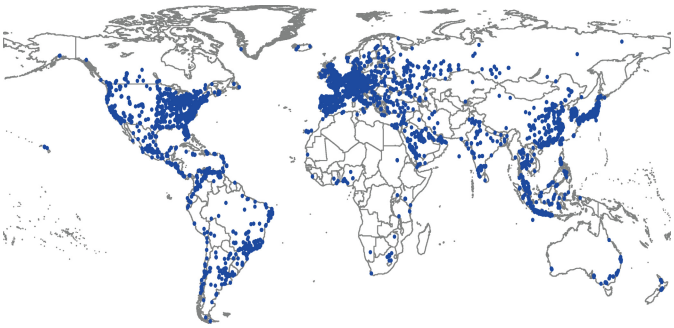


Fig. 3. Geographical distribution of users in the collected dataset.

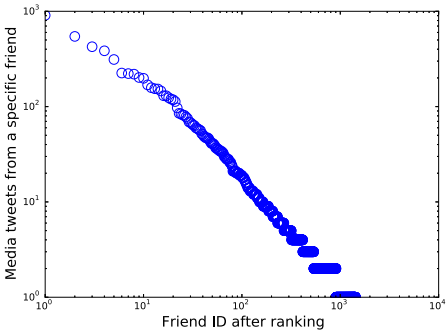


Fig. 4. Number of clicked media tweets from user's friends in log-log scale.

clicked tweets contain media content (e.g., image). The volume and diversity of data reflects the real-life behavior of the mobile social app users, which is crucial for understanding and predicting media content in mobile social application network traffic. In our prediction algorithm, we are primarily interested in tweets with media content because the plaintexts of the tweets are simply short character strings, which are of very small size and which would not play a significant role for prefetching design. Therefore, unless otherwise specified, we mainly consider media tweets with images,[2] and target at prefetching them in an appropriate manner.

III. The Impact of Social Friendship

In this section, we first conduct a data-driven analysis on the social interaction among users and reveal its impact on the user's tweet click behavior.

The generation and re-share of a tweet on Twitter is simple: any user who generates or re-shares it will become a new host of the tweet content. Users can fetch these contents from their direct friends in the social network. Intuitively, the social relationships and interactions among a user and her friends have a significant impact on the Twittering behavior. A user may treat different friends differently, and interact with some close friends frequently, while having little contact or response with some unfamiliar friends on Twitter [11], [12].

In Fig. 4, we plot the number of one real-life user's clicked tweets with media content (i.e., media tweets) from her friends (i.e., social neighbors) on Twitter in the log-log scale.

[2]Note that the Twidere app currently just supports few video formats and the total number of video sharing is very limited, thus we mainly focus on image prefetching. We will extend our framework to the case with videos for other mobile OSNs in a future work.

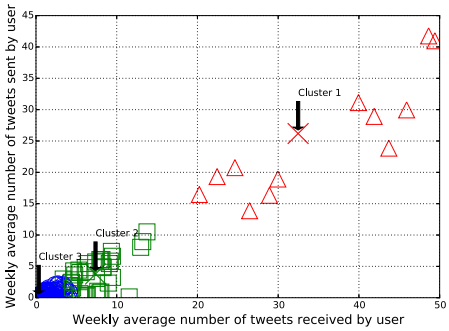


Fig. 5. Social friendship clustering.

We rank the set of friends in descending order according to the number of tweets sent by them. We observe a strong power law phenomenon, i.e., almost 85% of the tweets are from only a few friends (less than 5%), and most other friends have little contribution. This demonstrates that friendship (or social interaction strength) plays a critical role on shaping her usage behavior on Twitter. Thus, it motivates us to design the socially-driven prefetching mechanism by leveraging the social friendship among users.

With this insight, we further quantify the impact of social friendship on the user's media tweet click behaviors. To proceed, we first carry out the social friendship clustering. The intuition is that in reality a user typically has very close relationships with a small set of people (e.g., family member, close friends), and is familiar with a group of people (e.g., colleagues). For many other people, the user would have little contact with them. With this observation, we conduct the friendship clustering using the commonly-adopted K-Means clustering algorithm [13]. As illustrated in Fig. 5, we utilize the number of tweets received from a specific friend and the number of tweets sent by the user to that friend as the clustering features, and cluster the set of her friends into three types: close friends (i.e., Cluster 1 in Fig. 5), familiar friends (i.e., Cluster 2 in Fig. 5), and acquaintances with infrequent contacts (i.e., Cluster 3 in Fig. 5). We will elaborate the impact of the number of friendship clusters later.

After the social friendship clustering, we then explore the impact of friendship on the user's media click behavior when accessing the media tweets. Table II summarizes the media click probability under different scenarios. In total, we observe that the user clicks the media file with a probability of 0.64 (0.28, 0.13), when the media tweet is sent by a close (familiar, unfamiliar) friend, respectively. This again show that social friendship has a significant impact on user's media click behavior. We further explore the impact of social friendship on user's media click behavior with different features. For example, for the network feature in Table II, if the user is on WiFi and the media tweet is sent by a close friend, then she would click the media file with a probability of 0.71. However, if the tweet is sent by a familiar friend, the click probability would decrease to 0.19. As another example, for the interaction feature, if the media tweet has been favored by a close friend, then the user would click the media file with a high probability of 0.98.

Inspired by the observation above that the social friendship has a significant impact, we then develop a socially-driven

TABLE II
MEDIA CLICK PROBABILITY AMONG MEDIA TWEETS WITH DIFFERENT FEATURES

		close (%)	familiar (%)	unfamiliar (%)
Total Statistics	*Media click*	64.36	28.49	13.22
Network Features	*On WiFi*	71.26	19.69	1.97
	On cellular	49.43	13.66	1.47
Interaction Features	*Published by*	33.96	13.46	32.44
	Mentioning	33.96	13.46	9.01
	Favored by	98.99	97.55	0.52
	Retweeted by	97.66	9.57	0.06
	Replied by	50.00	28.49	23.40

scheme for the prefetching prediction based on machine learning.

IV. LEARNING ALGORITHM FOR SOCIALLY-DRIVEN PREFETCHING PREDICTION

In this section we propose the learning-based socially-driven prefetching prediction mechanism as follows.

A. Social Friendship Clustering

As discussed above, we first carry out the social friendship clustering process to partition the set of the user's friends into several friendship clusters. Different clusters represent different levels of social interactions between the user and her friends. Specifically, we utilize the number of tweets received from a specific friend and the number of tweets sent by the user to that friend as the clustering features, and use the commonly-adopted K-Means clustering algorithm [13] to carry out the social friendship clustering. In the following, we denote the number of friendship clusters as K, and the set of clusters as $\mathcal{K}$. For example, we can set $K = 3$ and $\mathcal{K} = \{$*close friends, familiar friends, unfamiliar friends*$\}$. Note that the number of friendship clusters K generates a great impact on the accuracy of the proposed prefetching prediction mechanism later. Intuitively, when K is too small, the impact of the social friendship is not well leveraged, which would impair the predition accuracy. When K is too large, the number of training parameters increases significantly due to the large number of friendship clusters. As a result, in order to train the prediction mechanism well, more fine-grained data traces with more detailed information with respect to all the friendship clusters are needed. However, in practice it is extremely difficult to obtain such fine grained data traces since many friends interact infrequently and reveal insufficient information. According to our experience (as illustrated in Fig. 6), typically the number of clusters $K = 3$ can provide a good balance and achieve the best prediction accuracy. This is also consistent with our daily life observation that people tends to classify their friends into three types: close friends, familiar friends, and acquaintances (with infrequent contact).

B. Tweet Training Features

After the social friendship clustering, we identify the set of important tweet training features for building up the learning model for prefetching prediction. As shown in Table II, we found that two types of features are critical: network and interaction features. For the network features, whether the user would click the media file depends on the network environment[3] (on WiFi or cellular). For the interaction features, the events that the media tweet is published by, mentioning, favored by, retweeted by, or reply to by a friend are also taken into account. In the following, we denote the number of training features as F, and the set of all tweet training features as $\mathcal{F}$.

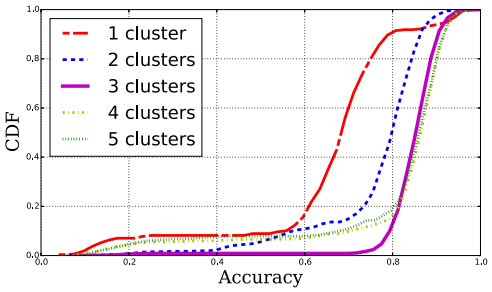


Fig. 6. Cluster-based LBM algorithm (with the average prediction accuracy of 65.54%, 77.93%, 84.51%, 80.38%, 80.03% for $1, 2, \ldots, 5$ clusters, respectively).

C. Cluster-Based Latent Bias Model

We then propose the learning model for prefetching prediction. Our algorithm design is based on the Latent Bias Model (LBM) [15] that aims to utilize proper bias terms to capture the importance of different features for prediction. Here we extend the standard LBM to our case with friendship clustering, and develop the cluster-based LBM approach for socially-driven prefetching prediction.

Specifically, based on the social friendship clustering, we define $b_{f,k}$ as the cluster-based bias term to stand for the case that a media tweet is sent by a friend in the friendship cluster k and contains the feature f. Moreover, for a given media tweet c, we first introduce an indicator function $I^c_{f,k} \in \{0, 1\}$ such that $I^c_{f,k} = 1$ if the media tweet c is sent by a friend in the friendship cluster $k \in \mathcal{K}$ and contains the feature $f \in \mathcal{F}$; $I^c_{f,k} = 0$ otherwise. Then, we define the media click score for media tweet c as follows:

$$\gamma_c = \alpha + b_0 + \sum_{f \in \mathcal{F}} \sum_{k \in \mathcal{K}} b_{f,k} I^c_{f,k}, \tag{1}$$

where α is the user's average click rate across all historical media tweets, and b_0 is the overall bias term for the user. Note that, in our system, there are different values of either α or b_0 for all friends of a specific user. In general, a higher media click score γ_c implies a higher probability that the user will click the media file in the media tweet c.

Then, the critical task is to train the cluster-based LBM, i.e., learn the proper bias terms in (1) in order to well capture the user's media click behavior. Suppose that the set of historical user data trace (i.e., historical media tweet set of the user) is

[3]Note that in this paper we utilize the widely-used Markov Chain model in [14] to predict user's network environment during a prefetching prediction slot.

denoted as $\mathcal{C}$. For each media tweet $c \in \mathcal{C}$, we have the ground truth $y_c \in \{1, -1\}$ such that $y_c = 1$ means that the user clicks the media file in the tweet c and $y_c = -1$ otherwise. Next, to quantify the discrepancy between the prediction based on the media click score γ_c and the desired ground-truth output y_c, we adopt the widely-used Logic loss measure, i.e.,

$$\mathcal{L}(\gamma_c, y_c) = \log[1 + \exp(-\gamma_c y_c)]. \quad (2)$$

Thus, we learn the proper bias terms to minimize the total loss across the historical data trace $\mathcal{C}$, i.e., $\sum_{c\in\mathcal{C}} \mathcal{L}(\gamma_c, y_c)$. Following the common practice in machine learning, in order to avoid overfitting, we also impose L_2 regularization into minimization. That is, we minimize:

$$\mathcal{O} = \sum_{c\in\mathcal{C}} \mathcal{L}(\gamma_c, y_c) + \lambda \left(||b_0||^2 + \sum_{f\in\mathcal{F}} \sum_{k\in\mathcal{K}} ||b_{f,k}||^2 \right), \quad (3)$$

where λ is the regularization parameter to be manually tuned.

Since the objective function in (3) is convex, we can apply the first-order condition and derive the gradients as

$$\frac{\partial \mathcal{O}}{\partial b_0} = -\sum_{c\in\mathcal{C}} \left(\frac{\exp(\gamma_c y_c)}{1 + \exp(\gamma_c y_c)} \right) y_c + 2\lambda b_0, \quad (4)$$

$$\frac{\partial \mathcal{O}}{\partial b_{f,k}} = -\sum_{c\in\mathcal{C}} \left(\frac{\exp(\gamma_c y_c)}{1 + \exp(\gamma_c y_c)} \right) y_c I^c_{f,k} + 2\lambda b_{f,k}. \quad (5)$$

Similar to many machine learning studies, we adopt the Stochastic Gradient Descent (SGD) method [16] to learn optimal bias terms [16]. The key idea is to utilize the data samples to iteratively update the gradient as follows:

$$b_*^{t+1} = b_*^t - \epsilon_t \frac{\partial \mathcal{O}}{\partial b_*^t}, \quad (6)$$

where b_*^t denotes a given bias term at the t-th iteration and $0 < \epsilon_t < 1$ is the smoothing factor for updating. As shown in [16], as long as a small enough ϵ_t is used, the SGD method converges to the optimum.

After learning the optimal bias terms from the historical data trace $\mathcal{C}$, we then rank the set of new coming media tweet $\mathcal{H}$ for prefetching prediction during a given time slot (e.g., every 15 minutes). Specifically, for each media tweet $c \in \mathcal{H}$, we compute its media click score γ_c using the learned bias terms. Accordingly, we rank the new media tweets in the set $\mathcal{H}$. Finally, as illustrated in Fig. 10, we determine the number of media tweets to be prefetched, based on the statistics of the average number of media tweet clicks during that specific time slot (e.g., weekend or weekday, and which slot).

Fig. 6 shows the performance of the proposed cluster-based LBM algorithm with different number of friendship clusters. It depicts the cumulative probability distribution (CDF) of the prediction accuracy for all the testing tweets. As discussed before, when the number of friendship clusters equals $K = 3$, it achieves the best performance with the average prediction accuracy of 0.845.[4] As the benchmark, we also implement the linear regression (LR) algorithm using tweet training features only [7]. Fig. 7 shows that the LR approach can only achieve

[4]We have elaborated the reason that $K = 3$ clusters offers the best performance in Section IV-A.

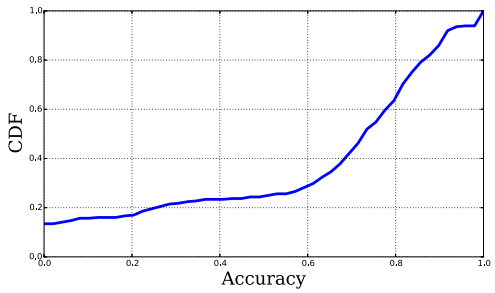


Fig. 7. LR algorithm (with the average prediction accuracy of 63.82%).

the average prediction accuracy of 0.638, which is even slightly worse than standard LBM without social friendship clustering (i.e., the case with one cluster in Fig. 6). This demonstrates the efficiency of the proposed cluster-based LBM approach. The gain of cluster-based LBM mainly stems from the fact that the defined cluster-based bias terms can well capture the impact of social friendship on user's click behavior.

V. Usage-Adaptive Prefetching Scheduling

After studying the prefetching prediction, we next consider the prefetching scheduling problem of Spice.

A. Observations and Measurements

Intuitively, different users may show feature significantly different habits/patterns in the mobile social app usage. Motivated by this phenomenon, we schedule the prefetching to be adaptive to a user's usage pattern. For example, in Fig. 8 and Fig. 10, through tracing a user's app usage along different hours of a day during the period of two months, we observe that the user shows different patterns for weekday and weekend. For instance, the user tends to use the app more frequently from the time period from 7:00 to 12:59 on weekdays. Furthermore, if we partition the horizon of a day into 4 zones: from 01:00 to 06:59 (midnight), from 07:00 to 12:59 (morning), from 13:00 to 18:59 (afternoon) and from 19:00 to 00:59 (night), the user has different app usage frequencies during different zones. Note that, as we elaborate on in Section VII-B, such 4 time zone scheme can efficiently cover most user activities already. Thus, we can schedule Spice for media tweet prefetching with different frequencies with respect to different specific zones, based on the statistics of the average app usage frequencies in different zones. For instance, for the user in Fig. 8, we invoke Spice every 1 hour in the first zone, in the second zone Spice will be scheduled every 20 minutes on weekdays and every 30 minutes on weekends, in the third zone Spice will be invoked every 15 minutes on weekdays and every 20 minutes on weekends, and in the fourth zone we invoke Spice every 15 minutes on both weekdays and weekends.

To reduce the scheduling complexity and to achieve superior performance, in this paper we equally partition the day horizon into 4 zones and accordingly set different prefetching frequencies in the experiments. Our evaluation experience shows that for our collected dataset more fine-grained day horizon partition do not increase the performance of Spice significantly. In Fig. 9, we compare the prefetching scheduling

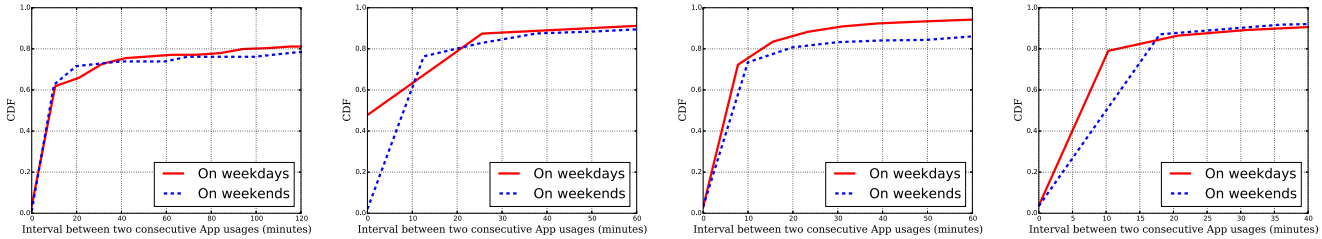


Fig. 8. Distribution of time interval between two consecutive app usages in 4 zones. (a) 01:00 to 06:59 (midnight). (b) 07:00 to 12:59 (morning). (c) 13:00 to 18:59 (afternoon). (d) 19:00 to 00:59 (night).

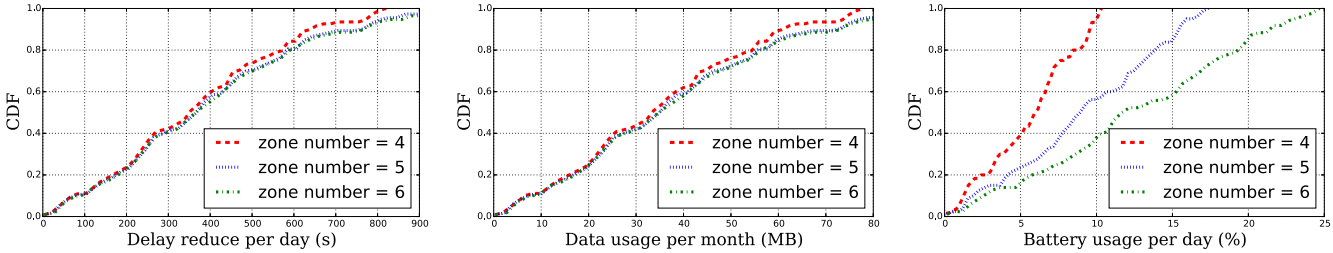


Fig. 9. Performance comparison of 3 different horizon partition setups in terms of its impact on delay reduction, cellular data usage, and energy consumption. (a) Delay reduction. (b) Cellular data consumption. (c) Energy consumption.

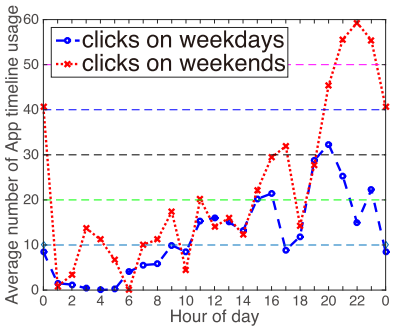


Fig. 10. Statistics of a user's media tweet consumption across hours of day.

settings of Spice with different numbers of zones for the day horizon partition. We focus on the primary metrics of delay reduction per day, monthly cellular data usage, and energy consumption per day. We observe that increasing the number of zones from 4 to 5 and 6 makes little improvement in terms of the access delay reduction. However, in addition to the increase of the scheduling complexity, it also results in increased monthly cellular data consumption of Twidere by 2.3 MB and 3.1 MB, respectively. What's more, daily battery usage climbs from 5.3% to 9.3% and 12.7% respectively. This is due to the fact that increasing the number of zones for the day horizon partition tends to result in more (or even excessive) frequent prefetching scheduling during some busy periods, and hence leads to high energy consumption. This demonstrates the effectiveness of the 4 zone setup for our case. Nevertheless, for other cases (with different OSN datasets) the number of zones for day horizon partition can be changed adaptively.

Note that to further reduce the cost of cellular data and energy consumption, we can decrease the number of media tweets to be prefetched during a scheduling slot t. Let N_t denote the average number of media tweet clicks during the prefetching slot t. We then introduce a control factor $\delta > 0$ such that we can globally control the number of prefetched media tweets in a slot as $\lceil \delta N_t \rceil$. In this case, we can set different control factor δ to reduce the cost of cellular data and energy consumption as needed. Obviously, there is a clear trade-off between the access delay reduction and the cost reduction. A smaller control factor δ reduces the cost of cellular data and energy consumption by reducing the number of prefetched media tweets, which would lead to a smaller user's access delay reduction. Due to the complicated and stochastic usages of the mobile OSN app by the user, it is difficult to characterize such trade-off in the theoretical manner. We hence investigate the trade-off effect in Section VI through extensive emulation-driven evaluation.

Furthermore, since WiFi coverage is pervasive and always available at regular locations such as home and workplace, we can also implement the solution of Spice with WiFi only such that we allow the scheduled prefetching to download media files only when the user is on WiFi. Evaluation results in Section VI show that Spice with WiFi only is still very efficient. With respect to the case without prefetching, it can achieve 58.4% access delay reduction at the cost of a small increase of daily battery usage.

B. Usage-Adaptive Scheme in Spice

The unique features of social relationship and mobile app usage brings an interesting but challenging work of

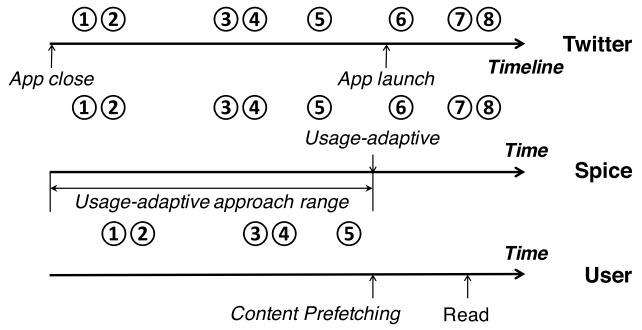


Fig. 11. Methodology of the usage-adaptive scheduling scheme in Spice.

scheduling the Spice prefetching task. In particular, prior work [6], [7], [17] have extensively studied the users' selectiveness when using a mobile OSN app. Moreover, as Fig. 8 and Fig. 10 illustrate, through analysing real-life usage trace of a certain Twidere users, we find that they usually consume social content with: 1) social selectiveness, especially for the embedded media content (see Table II); 2) time-sensitivity; 3) diverse access pattern; and 4) battery-sensitivity .

Towards addressing these systematic issues, we design Spice framework with an intelligent scheduling mechanism which adaptively leverages the app usage feeds. It integrates a complementary scheduling strategies, i.e., *usage-adaptive* that best leverage content freshness, channel conditions, and usage preference. Fig. 11 illustrates how Spice operates as the middleware between Twitter and user. The first timeline (timeline of the "Twitter Timeline" [18]) shows when the tweets with media contents (e.g., from 1 to 8 in Fig. 11) are posted. The second shows the Spice operations: a usage-adaptive scheme to predict when is the next real-life usage (see Section V-C). Links and media files in a tweet is prefetched by one of these three mechanisms respectively according to the user habits, with respect to different user's social selectiveness, time-sensitivity, and battery-sensitivity requirements.[5] The third timeline shows the user activity, where prefetched contents in tweets are consumed at this moment. Note that all of this three approaches only refers to the scheduling frequency, while as discussed before, this has little impact on the delay reduction and cellular data consumption. Thus, we adjust the three components to fit to user-specific battery consumption budgets, as discussed in detail next (see Section VI).

C. Prediction Model for Usage-Adaptive Scheduling

The trade-off between delay reduction and energy consumption motivate us further develop an aggressive but intelligent scheduling mechanism, namely, the usage-adaptive approach. As existing work has extensively studied network-condition-aware transmission/prefetching, which base on techniques such as predicting and leveraging favorable network conditions, using WiFi hotspots, piggy-backing with other transmissions, and batching [19]–[21], the usage-adaptive approach can be more energy effective than the former two while benefits user the same delay reduction as opportunistic approach at the same time. In particular, we define θ_t as the probability of a user accesses twitter at a specific moment (hour or minute) t. Note that θ_t varies significantly over time and across users, as illustrated in Fig. 8. We adjust how aggressive the Spice prefetching works based on this access probability θ: the click-through score for each tweet generated by content predictor is weighted using θ_t so that we prefetch more aggressively during active moments, and vise versa.

The usage-adaptive prefetcher runs when the internetwork is available. Inspired by [7], in this paper, if user cares about the cellular data flow or smartphone's battery, the prefetcher could be setup to only come into effect when a WiFi network is connected. It wakes up periodically (30 minutes in our preliminary setting and would be adaptively tuned) to scan the fresh social media and waits for the next piggy-backing context for downloading; i.e., data transmissions initiated by other applications. Upon detecting such a transmission, the prefetcher checks-in with the Twitter server for the fresh tweets. Then, smartphone notifies the Twitter server with the likelihood tweet accessed by the user since the last check-in which are then removed from the user's queue. The click-through score of each tweet with media files in the queue is weighted by $S_0 = S * \theta_t$. The tweets with weighted click-through score S_0 higher than a prefetching ratio threshold are then compressed and sent to the user device in one batch. Again, the threshold depends on energy budget, and is coupled with the threshold of the former two approaches, which are determined in Section V-A. We note that a special case is when the user accesses the application on WiFi. In this case, we set $\theta_t = 1$ and use the usage-adaptive threshold for prefetching (as it will not consume any cellular data).

Moreover, by leveraging the extensively studied work, e.g., [22]–[24], we further exploit the set of probabilities obtained by the usage-adaptive approach in order to provide a dynamic scheduling priority for the general applied mobile app case. With such a shifting from manual setting to priority inference, Spice can be more flexible to wake up the prefetching task. To achieve it, we apply two functions for inferring the dynamic prefetching scheduling: 1) For the n-th highest probabilities app launch, and 2) For the app whose probability has increased the most from its previous inference.

- n-th highest probabilities app launch:

$$\mathcal{P} = \{App_i \big| \mathcal{P}(App_i \big| C_i) \geq \mathcal{P}(App_a \big| C_a)\} \tag{7}$$

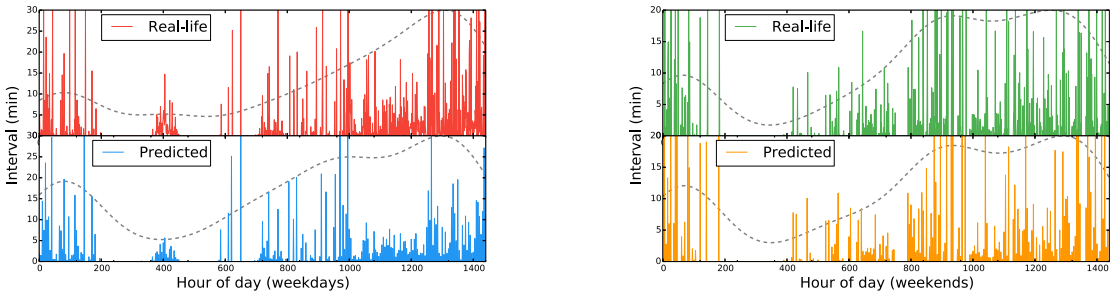
  where a refers to the app whose probability $\mathcal{P}(App_a \big| C_a)$ is n-th highest.
- Probability increasing:

$$\mathcal{P} = \arg\max_i (\mathcal{P}(App_i \big| C_i) - \mathcal{P}(App_i \big| C_{i,t-1})) \tag{8}$$

  where $\mathcal{P}(App_i \big| C_{i,t-1})$ is the previous probability of app i inferred for context $C_{i,t-1}$.

Specifically, we investigate the provided function with mobile social app as a study case in Fig. 12. We observe that it achieves an average 73% accuracy for predicting when and

[5]Note that in Fig. 11, we take both WiFi and cellular network access are available for instance, while it can be specifically applied for the WiFi only scenario.



(a) (b)

Fig. 12. Comparison among real app usage and proposed app usage prediction scheme along with hour of weekdays and weekends, respectively. (a) Real-life and predicted social app usage in weekdays. (b) Real-life and predicted social app usage in weekends.

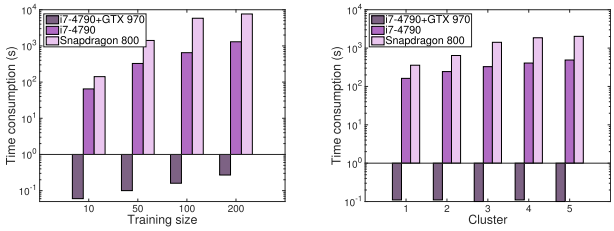
how long the user starts accessing Twitter. This again demonstrates the usage-adaptive could be more intelligent to call for a prefetching task according to predict when is user's next app access when comparing with benchmark approach in [7]. In addition, these functions can be applied to the selection timing of dynamic prefetching tasks i.e., recommending the applications that are necessary to but not have to be waken up; and further determining which of them must be waken up at a specific time.

VI. Evaluation

In this section, we conduct the trace-driven emulation evaluation on smartphones to investigate the performance of Spice. To evaluate the cluster-based LBM algorithm on a large-scale dataset, we run a trace-driven evaluation for 1000 long-term users such that each user keeps active for a long and consecutive period of at least 60 days with detailed trace records. The emulator runs on a Google Nexus 5 Android phone (with the CPU type of Qualcomm Snapdragon 800) connected to China Mobile TD-LTE cellular network and also has access to a campus WiFi network. The emulator reads and replays the usage events collected from real-life users, including connecting to or disconnecting from WiFi networks, accessing Twitter, and opening media files in tweets.

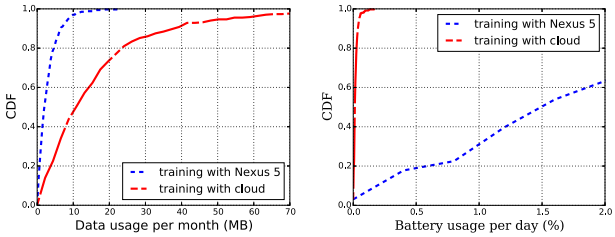
A. Comparison of Different Data Processing Approaches

The key component of Spice is to implement the cluster-based LBM algorithm for the data training (i.e., learning the optimal bias terms from the user data traces). In this paper we consider three data processing approaches: 1) processing on smartphone, i.e., we conduct the data processing on Google Nexus 5 phone locally; 2) processing on ordinary sever, i.e., we offload the data processing to the Intel i7-4790 CPU based server; 3) processing on a cloud cluster (e.g., using Spark), i.e., we offload the data processing to the cloud cluster server.[6] The key motivation here is to leverage the strong parallel computing power to speed up the data processing. In Fig 13, we evaluate the average time overhead of these three data processing approaches, by setting different training sizes and friendship cluster sizes in the learning algorithm. We find



(a) (b)

Fig. 13. Time overhead of different data processing approaches. (a) Different training sizes. (b) Different cluster sizes.



(a) (b)

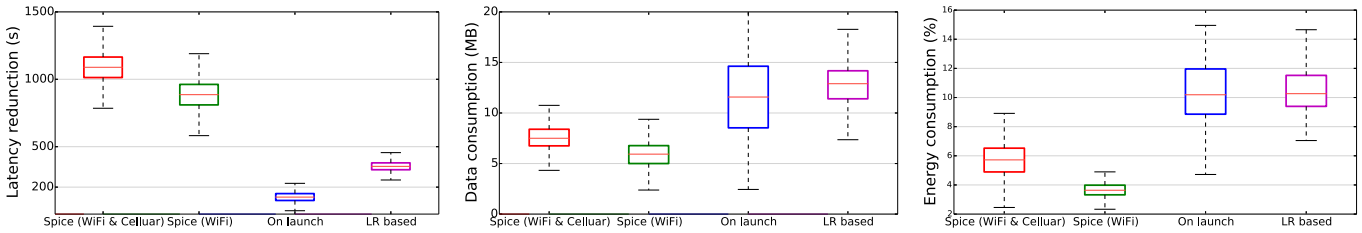
Fig. 14. The data and energy usage on smartphone for cluster-based LBM training. (a) Cellular data usage for training (3.12MB for mobile v.s. 15.14MB for cloud on average). (b) Battery usage for training (2.14% for mobile v.s. 0.02cloud on average)

that the cloud approach can significantly decrease the time overhead for data processing, by a factor of 1000.

We also evaluate the cellular data usage and energy consumption of these data processing approaches when a user allows the data training with cellular connectivity. Note that from the perspective of a smartphone, the cellular data usage and energy consumption are the same for both the server and cloud cluster approaches. Thus, we only compare the cloud cluster approach with the smartphone approach. The results are shown in Fig. 14. We observe that the cloud approach consumes less energy (0.02% of total battery usage per day), while leading to a higher cellular data usage (15 MB per month.)[7] If a user is more sensitive to energy consumption, she should choose the cloud approach. Otherwise, the user can choose to

[6]For simplicity, in this paper, we utilize NVIDIA GTX 970 GPU and Intel i7-4790 CPU to achieve the strong parallel computing power.

[7]The smartphone approach consumes 3 MB per month since additional data/information from Twitter server is needed for the training.



(a) (b) (c)

Fig. 15. Performance comparison among five prefetching strategies in terms of delay reduction, data usage, and energy consumption. (a) Benefit of delay reduction. (b) Cellular data consumption. (c) Energy consumption.

process the data locally on the smartphone. Furthermore, since the user's behavior tends to be stable, we can aggregate the training data for a while (e.g., one week), and carry out data training weekly by offloading the training to the cloud only when the user is on WiFi while charging, in order to save both cellular data and energy usage.

B. Comparison of Different Prefetching Strategies

After the data training and user budgets have been conducted, we finally evaluate the performance of the Spice framework running on the smartphones for media content prefetching. As illustrated in Fig. 6, the cluster-based LBM algorithm in Spice is very efficient, and achieves an average prediction accuracy of 0.845. Upon comparison, the linear regression (LR) algorithm using tweet training features [7] can only achieve an average prediction accuracy of 0.638. In the following, we also compare different prefetching strategies given as:

- Spice with WiFi and cellular: We implement the Spice framework using both WiFi and cellular connections, i.e., we allow media prefetching when the user is on either WiFi or cellular.
- LR-based prefetching: We replace the cluster-based LBM algorithm by the LR algorithm in the Spice framework using both WiFi and cellular connections.
- Spice on launch: We run the cluster-based LBM algorithm in Spice for media prefetching only when the user launches the Twidere app. Otherwise, we do not schedule the media prefetching.
- Spice with WiFi only: We run the cluster-based LBM algorithm in Spice for media prefetching using WiFi connection only, i.e., we allow the scheduled prefetching to download media files only when the user is on WiFi.

For the evaluation of each prefetching strategy, the same user events are replayed, and media links are fetched using the WebView component which is provided by the Android framework. WebView provides a notification when the media file downloading has finished, which is used to calculate the loading delay of a media link. Aiming at a fair comparison, here we do not impose the battery and data budget constraints among all the prefetching strategies.

We compare different prefetching strategies in terms of the delay reduction per day (with respect to the case without any prefetching), cellular data usage of Twidere with prefetching

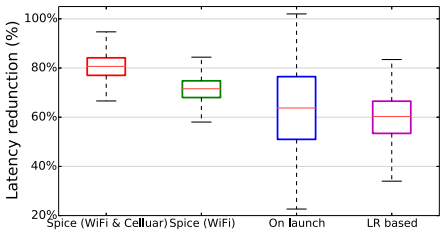


Fig. 16. Benefit of latency reduction (in %).

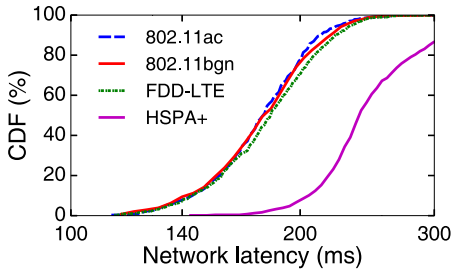


Fig. 17. RTT responsiveness to different network environments.

(which includes the cellular traffic for both prefetching by Spice and on-demand fetching by the user), and energy consumption of the app per day (i.e., the percentage of the fully-charged battery capacity). The results are shown in Fig. 15 and 16. We see that Spice with WiFi and cellular achieves the best performance in terms of delay reduction, which can achieve 80.6% (1089s) reduction per day. In terms of the overhead, Spice with WiFi and cellular uses 7.5 MB cellular traffic per month and 5.7% battery usage per day on average. Moreover, Fig. 17 measures the RTT responsiveness to four mainstream network environments. It significantly demonstrate the effectiveness of Spice since the RTT are average 198ms (17% of each service latency in Fig. 17). Upon comparison, LR-based prefetching can only achieve 60% (354s) delay reduction with 12.9 MB cellular traffic per month and 10.2% battery usage per day on average.[8] This is due to the fact that LR-based prefetching has a much lower prediction accuracy than that of the cluster-based LBM in Spice. This demonstrates the efficiency of the proposed Spice framework. Furthermore,

[8]In [7], LR achieves the promised performance with strict usage budgets. Otherwise, it consumes more energy and cellular data.

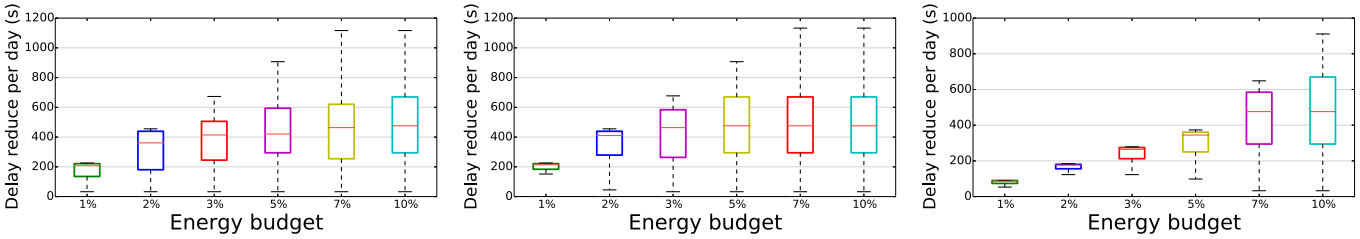


Fig. 18. Comparison among user's different daily battery budget in 3 different network access environments. (a) WiFi and cellular network. (b) With WiFi only access. (c) With cellular only access.

we observe that Spice with WiFi only achieves the second best performance for the delay reduction, with a 71.2% (886s) reduction per day. Spice with WiFi only achieves the lowest cost among all the prefetching strategies, with 5.9 MB per month and 3.6% battery usage per day on average. As per the evaluation result, compared with the case without prefetching, Spice with WiFi only consumes less cellular data traffic with an increase of around 2.1% battery usage per day on average. This is since the WiFi coverage is pervasive and always available at regular locations such as home and workplace, and Spice has a high prefetching prediction accuracy to download the most relevant media files in advance on WiFi. When a user is not sensitive with the cellular data usage, we can utilize Spice with WiFi and cellular to achieve a significant delay reduction. If the user is concerned about the cellular data usage, we can then adopt Spice with WiFi only, which can still achieve a superior performance with a low overhead.

C. Comparison of Different User Budget Constraints

We next evaluate how the battery and data budget constraints impact the performance of Spice. Specifically, we evaluate the delay reduction by Spice such that the budget constraints of battery usage and data consumption are satisfied by turning the control factor δ in the scheduling as discussed in Section V. Similar to previous emulation evaluations, we replay the usage trace by taking account into different network access environments as follows:

- Spice with both WiFi and cellular network access, i.e., we tuning user budgets and evaluate how Spice performs when both WiFi and cellular network are available.
- Spice with WiFi access only: We conduct the emulation evaluation with different user budgets only when in a WiFi available environment.
- Spice with cellular access only: We replace the network access permission from WiFi with cellular network.

We first evaluate how battery budget impacts Spice. As described in Fig. 18, we consider 6 categories of the battery budget constraints, which are given in terms of the usage percentage of a fully-charged battery capacity, i.e., $1\%, 2\%, 3\%, 5\%, 7\%,$ and 10%. We observe that the battery budget impacts Spice strictly. Specifically, a budget of $5\%(3\%, 7\%)$ battery budget per day can achieve almost the best performance when in both WiFi and cellular access (only WiFi access, only cellular available) network environment, and benefits user with $420(415, 414)$ seconds delay reduce.

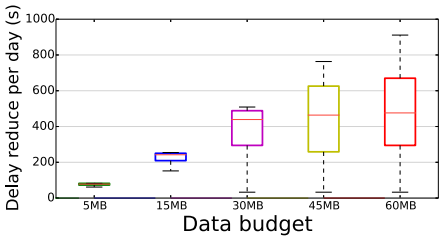


Fig. 19. Comparison among user's different monthly data budget.

On this basis, we next evaluate the influence of data budget.[9] As Fig. 19 shows, the data budget also plays a critical role in Spice. Moreover, a threshold of 30MB data budget per month can satisfy the most media content prefetching requirements when comparing with the previous battery budget evaluations (with an average 435 seconds daily delay reduction against 420 seconds). Furthermore, with the popularization of luxury size but inexpensive data plan [25], it is worthwhile for most users to embrace the benefits of Spice with no data budget constraints. Also, as mentioned above, when user is sensitive to cellular data consumption, it can adopt the solution of Spice with WiFi.

VII. Discussion and Future Work

A. Impact of Usage Time

Spice is a first step towards solving social media prefetching problem by predicting real usage along a line of social friendship impact. Our algorithm performance demonstrates that Spice is promising when integrates the key features of network and social interactions.

However, it may happen that in practice that Spice is trained insufficiently with user. Motivated by this fact, in Fig. 20, we further investigate the performance of Spice for media content prefetching, by considering three categories of users, namely, heavy users (the top 5% most active users that publish and subscribe average 251.5 tweets per day), normal users (the top 20% active users with average 121.6 tweets per day),

[9]Note that from the perspective of each prefetching procedure, the battery budget is the same for all the three network access environments and makes no impact on each time of media content prefetching. Thus, we only compare delay reduce in the both WiFi and cellular are available environment.

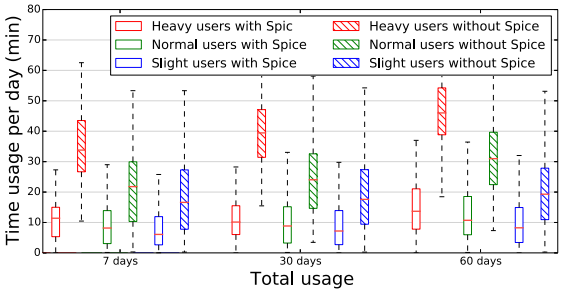


Fig. 20. Benefits of Spice with respect to no media content prefetching case.

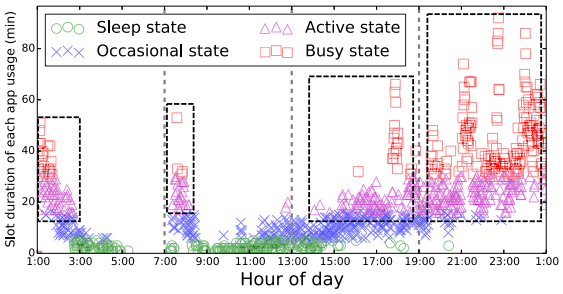


Fig. 21. Analysis of a user's 60 days Twitter state distribution law along with hour of day.

and slight user (i.e., the top 60% active users with average 36.7 tweets per day). In addition, we conduct this emulation evaluation with the selected users during 7 days', 30days', and 60 days' real usage as the training periods, respectively.

Fig. 20 depicts the CDF of the prediction accuracy of all the testing twitter media content prefetching for different user categories. Similar to the accuracy evaluation above, we adopt the 3 friendship cluster and 60 days use as a case, it achieves a performance with an average prediction accuracy of 0.845, with a performance of 10% (21%, 37%) delay reduction for slight (normal, heavy) users, with respect to the case with no Spice prefetching procedure. This shows the effectiveness and significance of the Spice system. Along another line, we also investigate the prefetching performance along with different training periods, i.e., 7 days usage, 30 days usage, and 60 days usage. We observe that the performance of Spice slightly improves as the increase of the length of the training period. This is mainly due to the fact that Spice is trained with more fine-grained data as the training period length increases.

B. More Fine-Grained Day Horizon Partitioning

To explore the effectiveness of our scheduling design in Section V, we have adopted the proposed 4 time zone partition approach for an instance. Nevertheless, people tends to exhibit complicated behavior patterns in terms of mobile social app usage in every single day. To comprehensively study this effect, we further analyze the user behavior patterns by first defining the following states as:

- *Busy:* While a user remains in the busy state, she will constantly use the Twitter for more than 30 minutes.
- *Active:* While a user remains in the active state, she keeps using the Twitter for more than 15 minutes but less than 30 minutes.
- *Occasional:* While a user remains in the occasional state, she keeps using the Twitter for more than 5 minutes but less than 5 minutes.
- *Break:* While a user remains in the break state, she keeps using the Twitter for less than 5 minutes.

In Fig. 21, we carry out data-driven measurement on a user's state distribution along with hour of a day. Based on this preliminary, we then outline the law of user behavior in each time zone. More specifically, most effective user behaviors, i.e., busy and active state, can be located in 3 or 4 time zones. This also conforms to our daily experience that people usually define their time into 4 horizons as aforementioned. It further demonstrate the effectiveness of 4 time zone horizon. As for a future work, we can tuning the threshold of scheduling frequency in according with user behavior adaptively.

VIII. Related Work

In this section, we describe prior work in mobile prefetching and in socially-driven network analysis, and highlight the key differences of Spice from the existing studies.

A. Mobile Prefetching

For the mobile prefeteching, [6] presents the Informed Mobile Prefetching (IMP) framework as a prefetching scheduling library that a mobile app is able to link to control the energy and cellular data consumption. In IMP, a strong assumption is that the whole procedure works on the basis that mobile apps provide precise prediction information through mining users' content usage pattern. Ravindranath *et al.* [26] illustrate that inappropriate prefetching can be worthless to mobile users. They adopt Procrastinator to decide whether prefetching tasks should be invoked by considering different constraints, including the network environment (on WiFi or cellular), the user's data plan, and battery life. Note many related works in the literature (e.g., [6], [26]–[29]) target at designing mobile prefetching mechanisms of generic purpose, which can be used for different types of mobile apps. Similar to our work, a recent study in [7] considers the media content prefetching in mobile OSN services, which adopt the linear regression model for prediction by utilizing the tweet training features through mining the user's OSN usage pattern. Along a different line, motivated by the insight that social friendship plays a critical role on users' media tweet click behavior, in this paper we propose a novel socially-driven learning-based prefetching prediction based on the generalized cluster-based Latent Bias Model.

B. Socially-Driven Network Analysis

For the socially-driven network analysis, [30]–[32] identify the social graphical structure as a key influence on the interactions of users with social ties using Flickr dataset. A number of recent papers addresses the problem of computing influence in Twitter-like networks and finding leader users whose tweets are influential. Reference [33] and [34] detect the influential users by applying the PageRank ranking algorithm based on the number of retweets among users, and [35] utilizes

the user attributes such as number of friends, number of followers and past influence of seed users. Reference [36] proposes a variation of PageRank algorithm, accounting for topic specific ranking to measure the influence. Our work does not aim at finding users who are influential directly. Instead, we incorporate the feature that the different social friends make significantly different impact on a user's likelihood behaviors on media tweet consumption. Reference [37] proposes a tree-based algorithm to mine user-friend graphs to discover strong friends of a user. In contrast to our work, [37] does not consider how to utilize the social friendship structure to facilitate the information and content sharing among users in particular under the rich media content.

IX. Conclusion

Aiming at designing an intelligent mobile prefetching mechanism, in this paper we first identified the unique features of user's social behavior in OSN, and then proposed a novel framework of Spice based on the cluster-based LBM learning mechanism for prefetching prediction. We also developed an adaptive prefetching scheduling scheme by mining user's mobile OSN app usage pattern. We further evaluated the performance of Spice through trace-driven emulation on smartphones. Evaluation results corroborate that the proposed Spice approach can achieve superior performance with a significant access delay reduction at the low cost of cellular data and energy consumption. Moreover, our design enables users to offload their machine learning procedures to a cloud server, and achieves a speed-up of up to 1000 over the local execution on smartphones.

Note that in this paper we propose the Spice framework by using Twitter as a case study. Nevertheless, the proposed techniques can be applied to other OSNs as well. For instance, by integrating the Spice prefetching mechanism, Spice could benefit the *Moment* module (which contains rich media content for information sharing among friends) of WeChat [38], a popular mobile OSN service with 600 million active users. Moreover, we will consider a comprehensive design to integrate the prefetching techniques enabled by Spice at the mobile side with cloud computing techniques at the content server side in a synergetic manner.

Acknowledgment

The authors would like to thank the anonymous reviewers for their insightful comments; B. Yang for the his help of the emulation experiments; N. Li for his selfless and significant work on the Twidere application. Also, we would like to thank all the voluntary data providers for their kind help on this paper.

References

[1] C. Wu *et al.*, "Spice: Socially-driven learning-based mobile media prefetching," in *Proc. 35th Annu. IEEE Int. Conf. Comput. Commun. (IEEE INFOCOM)*, Apr. 2016, pp. 1–9.

[2] A. Lella, A. Lipsman, and K. Dreyer. *U.S. Digital Future in Focus*, accessed on Apr. 2014. [Online]. Available: http://www.comscore.com/Insights/Presentations-and-Whitepapers/2014/2014-US-Digital-Future-in-Focus

[3] S. Kemp. *Digital, Social & Mobile Worldwide in 2015*, accessed on Jan. 2015. [Online]. Available: http://wearesocial.net/blog/2015/01/digital-social-mobile-worldwide-2015

[4] J. Holcomb, J. Gottfried, and A. Mitchell. *News Use Across Social Media Platforms*, accessed on Nov. 2013. [Online]. Available: http://www.journalism.org/2013/11/14/news-use-across-social-media-platforms/

[5] D. Chu, A. Kansal, J. Liu, and F. Zhao, "Mobile apps: It's time to move up to CondOS," in *Proc. 13th USENIX Conf. Hot Topics Oper. Syst.*, 2011, p. 16.

[6] B. D. Higgins *et al.*, "Informed mobile prefetching," in *Proc. 10th Int. Conf. Mobile Syst., Appl., Services*, 2012, pp. 155–168.

[7] Y. Wang, X. Liu, D. Chu, and Y. Liu, "EarlyBird: Mobile prefetching of social network feeds via content preference mining and usage pattern analysis," in *Proc. 16th ACM Int. Symp. Mobile Ad Hoc Netw. Comput.*, 2015, pp. 67–76.

[8] D. Rout, K. Bontcheva, D. Preoţiuc-Pietro, and T. Cohn, "Where's@ wally?: A classification approach to geolocating users based on their social ties," in *Proc. 24th ACM Conf. Hypertext Social Media*, 2013, pp. 11–20.

[9] K. Makice, *Twitter API: Up and Running: Learn How to Build Applications with the Twitter API*, Sebastopol, CA, USA: O'Reilly Media, 2009.

[10] R. McCreadie, I. Soboroff, J. Lin, C. Macdonald, I. Ounis, and D. McCullough, "On building a reusable Twitter corpus," in *Proc. 35th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, 2012, pp. 1113–1114.

[11] M. E. J. Newman, "Modularity and community structure in networks," *Proc. Nat. Acad. Sci. USA*, vol. 103, no. 23, pp. 8577–8582, 2006.

[12] P. H. Wright, "Interpreting research on gender differences in friendship: A case for moderation and a plea for caution," *J. Social Pers. Relationships*, vol. 5, no. 3, pp. 367–373, 1988.

[13] A. Ahmad and L. Dey, "A k-mean clustering algorithm for mixed numeric and categorical data," *Data Knowl. Eng.*, vol. 63, no. 2, pp. 503–527, 2007.

[14] A. J. Nicholson and B. D. Noble, "BreadCrumbs: Forecasting mobile connectivity," in *Proc. 14th ACM Int. Conf. Mobile Comput. Netw.*, 2008, pp. 46–57.

[15] L. Hong, R. Bekkerman, J. Adler, and B. D. Davison, "Learning to rank social update streams," in *Proc. 35th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, 2012, pp. 651–660.

[16] L. Bottou, "Stochastic gradient descent tricks," in *Neural Networks: Tricks of the Trade*. Berlin, Germany: Springer, 2012, pp. 421–436.

[17] P. Ferreira, M. McGregor, and A. Lampinen, "Caring for batteries: Maintaining infrastructures and mobile social contexts," in *Proc. 17th Int. Conf. Human-Comput. Interact. Mobile Devices Services*, 2015, pp. 383–392.

[18] D. Shamma, L. Kennedy, and E. Churchill, "Tweetgeist: Can the Twitter timeline reveal the structure of broadcast events," *CSCW Horizons*, pp. 589–593, 2010.

[19] L. Li, C. Chen, W. Yu, Y. Wang, and X. Guan, "Demo: An efficient and reliable wireless link for mobile video surveillance systems," in *Proc. 16th ACM Int. Symp. Mobile Ad Hoc Netw. Comput. (MobiHoc)*, New York, NY, USA, 2015, pp. 409–410. [Online]. Available: http://doi.acm.org/10.1145/2746285.2764934

[20] L. Sang, A. Arora, and H. Zhang, "On exploiting asymmetric wireless links via one-way estimation," in *Proc. 8th ACM Int. Symp. Mobile Ad Hoc Netw. Comput.*, 2007, pp. 11–21.

[21] T. Issariyakul, E. Hossain, and A. S. Alfa, "End-to-end batch transmission in a multihop and multirate wireless network: Latency, reliability, and throughput analysis," *IEEE Trans. Mobile Comput.*, vol. 5, no. 9, pp. 1143–1155, Sep. 2006.

[22] C. Shin, J.-H. Hong, and A. K. Dey, "Understanding and prediction of mobile application usage for smart phones," in *Proc. ACM Conf. Ubiquitous Comput.*, 2012, pp. 173–182.

[23] Z.-X. Liao, Y.-C. Pan, W.-C. Peng, and P.-R. Lei, "On mining mobile apps usage behavior for predicting apps usage in smartphones," in *Proc. 22nd ACM Int. Conf. Conf. Inf. Knowl. Manage.*, 2013, pp. 609–618.

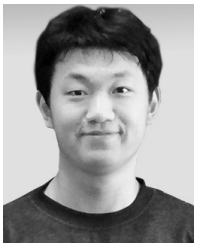
[24] W. Yuan and K. Nahrstedt, "Energy-efficient soft real-time cpu scheduling for mobile multimedia systems," *ACM SIGOPS Oper. Syst. Rev.*, vol. 37, no. 5, pp. 149–163, 2003.

[25] *AT&T Data Planner*, accessed on Dec. 2016. [Online]. Available: http://www.att.com/att/planner/

[26] L. Ravindranath, S. Agarwal, J. Padhye, and C. Riederer, "Give in to procrastination and stop prefetching," in *Proc. 12th ACM Workshop Hot Topics Netw.*, 2013, p. 7.

[27] A. Parate, M. Böhmer, D. Chu, D. Ganesan, and B. M. Marlin, "Practical prediction and prefetch for faster access to applications on mobile phones," in *Proc. ACM Int. Conf. Pervasive Ubiquitous Comput.*, 2013, pp. 275–284.

[28] J. Morse and J. Grubb, "Prefetching content based on a mobile user profile," U.S. Patent 2006 0 277 271, Dec. 7, 2006.
[29] Y. Zhang, K. Guo, J. Ren, Y. Zhou, J. Wang, and J. Chen, "Transparent computing: A promising network computing paradigm," *Comput. Sci. Eng.*, vol. 19, no. 1, pp. 7–20, 2017.
[30] A. Anagnostopoulos, R. Kumar, and M. Mahdian, "Influence and correlation in social networks," in *Proc. 14th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2008, pp. 7–15.
[31] C. Wu *et al.*, "Affective contextual mobile recommender system," in *Proc. ACM Multimedia Conf. (MM)*, New York, NY, USA, 2016, pp. 1375–1384. [Online]. Available: http://doi.acm.org/10.1145/2964284.2964327
[32] X. Chen, B. Proulx, X. Gong, and J. Zhang, "Exploiting social ties for cooperative D2D communications: A mobile social networking case," *IEEE/ACM Trans. Netw.*, vol. 23, no. 5, pp. 1471–1484, Oct. 2015.
[33] H. Kwak, C. Lee, H. Park, and S. Moon, "What is Twitter, a social network or a news media?" in *Proc. 19th Int. Conf. World Wide Web*, 2010, pp. 591–600.
[34] M. Cha, H. Haddadi, F. Benevenuto, and P. K. Gummadi, "Measuring user influence in Twitter: The million follower fallacy," in *Proc. ICWSM*, 2010, vol. 10. nos. 10–17, p. 30.
[35] E. Bakshy, J. M. Hofman, W. A. Mason, and D. J. Watts, "Everyone's an influencer: Quantifying influence on Twitter," in *Proc. 4th ACM Int. Conf. Web Search Data Mining*, 2011, pp. 65–74.
[36] J. Weng, E.-P. Lim, J. Jiang, and Q. He, "TwitterRank: Finding topic-sensitive influential twitterers," in *Proc. 3rd ACM Int. Conf. Web Search Data Mining*, 2010, pp. 261–270.
[37] J. J. Cameron, C. K.-S. Leung, and S. K. Tanbeer, "Finding strong groups of friends among friends in social networks," in *Proc. IEEE 9th Int. Conf. Dependable, Autonomic Secure Comput. (DASC)*, Dec. 2011, pp. 824–831.
[38] *WeChat Application for Smartphone*, accessed on Mar. 2017. [Online]. Available: http://en.wikipedia.org/wiki/wechat



Chao Wu received the B.Eng. degree in software engineering from Southeast University, Nanjing, China, in 2012. He is currently pursuing the Ph.D. degree with the Department of Computer Science and Technology, Tsinghua University, Beijing, China. His current research interests include mobile computing systems and networked systems with a focus on fog computing, operating system, and wireless communication and applications.

Xu Chen received the Ph.D. degree in information engineering from The Chinese University of Hong Kong in 2012. He was a Post-Doctoral Research Associate with Arizona State University, Tempe, USA, from 2012 to 2014, and a Humboldt Fellow with the University of Göttingen, Germany, from 2014 to 2016. He is currently a Professor with the School of Data and Computer Science, Sun Yat-sen University, Guangzhou, China. He received the 2014 IEEE INFOCOM Best Paper Runner-Up Award and the 2014 Hong Kong Young Scientist Runner-Up Award.

Wenwu Zhu received the Ph.D. degree in electrical and computer engineering from the New York University Polytechnic School of Engineering, New York, NY, USA, in 1996. He is currently a Professor with the Computer Science Department, Tsinghua University, Beijing, China. His current research interests include multimedia cloud computing, social media computing, multimedia big data, and multimedia communications and networking. He has been serving as Editor-in-Chief for the IEEE TRANSACTIONS ON MULTIMEDIA since 2017. He served on various editorial boards, including serving as Leading Editor of Computer Networks and Distributed Computing for the *Journal of Computer Science and Technology*; a Guest Editor of the PROCEEDINGS OF THE IEEE, the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY, and the IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS; and an Associate Editor of the IEEE TRANSACTIONS ON MOBILE COMPUTING, the IEEE TRANSACTIONS ON MULTIMEDIA, and the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY. He was a recipient of the Best Paper Award at ACM Multimedia 2012, the Best Paper Award of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY in 2001, and four other international Best Paper Awards.

Yaoxue Zhang received the B.S. degree from the Northwest Institute of Telecommunication Engineering, China, and the Ph.D. degree in computer networking from Tohoku University, Japan, in 1989. He joined the Department of Computer Science, Tsinghua University, China. He was a Visiting Professor with the Massachusetts Institute of Technology and University of Aizu, in 1995 and 1998, respectively. His major research interests include computer networking, operating systems, ubiquitous computing, big data driven service optimization, and mobile networks. He serves as an Associate Editor of the *Chinese Journal of Computers* and *Chinese Journal of Electronics*, and a Guest Editor of the IEEE TRANSACTIONS ON COMPUTERS. He is currently a Fellow of the Chinese Academy of Engineering and a Professor of Computer Science and Technology with Tsinghua University, China. He is the President of Central South University, China.