Network Agile Preference-Based Prefetching for Mobile Devices

JunZe Han[1], Xiang-Yang Li[1,2], Taeho Jung[1], Jumin Zhao[3], Zenghua Zhao[4]
[1]Department of Computer Science, Illinois Institute of Technology, USA
[2] School of Software and TNLIST, Tsinghua University
[3]College of Information Engineering, Taiyuan University of Technology, China
[4]Department of Computer Engineering, Tianjin University, China
Email: jhan20@hawk.iit.edu, xli@cs.iit.edu, zhaojumin@tyut.edu.cn, zenghua@tju.edu.cn

***Abstract*—For mobile devices, communication via cellular networks consumes more energy than via WiFi networks, and suffers an expensive limited data plan. On the other hand, as the coverage and the density of WiFI networks are smaller than those of the cellular networks, users cannot purely rely on WiFi to access the Internet. In this work we present a behavior-aware and preference-based approach to prefetch news webpages for the user to visit in the near future, by exploiting the WiFi network connections to reduce the energy and monetary cost. We first design an efficient preference learning algorithm to keep track of the user's changing interests, and then by predicting the appearance and durations of the WiFi network connections, our prefetch approach optimizes when to prefetch to maximize the user experience while lowing the prefetch cost. Our prefetch approach also exploits the idle period of WiFi connections to reduce the tail-energy consumption. We implement our approach in iPhone and our extensive evaluations show that our system achieves about 60% hit ratio, saves about 50% cellular data usage, and reduces the energy cost by 7%.**

I. Introduction

According to a recent study [1], mobile web browsing is growing significantly and is expected to surpass the desktop web browsing by 2015. Web browsing and news reading account for a large proportion of the time and the data used by smartphones: smartphone users with a data plan spend on average 300 minutes per month browsing the web, which is comparable to mobile voice usage [2]. A global smartphone study [2] shows that web browser is the single most popular data application which accounts for 54% of data application face time and 50% of data volume for smartphones; and reading news makes up for 68% of the user's web browsing time, which is one of the most frequent activities [3].

Typically there are two networking access approaches: through cellular network and through WiFi network. Although the cellular network is almost ubiquitous and its coverage seems not to be a problem, the limited data plan, higher price and more energy consumption [4], [5] make the smartphone users prefer using the WiFi network. It is thus natural to switch to WiFi network connection whenever possible. In order to reduce energy and monetary cost while not sacrificing the user experience, in this work we present a seamless transparent solution that automatically prefetches contents and switches the connection between cellular and WiFi networks. To make the content prefetching work for mobile devices, a number of challenges must be addressed. In general, we need to know when to prefetch, and what contents to prefetch such that the user's experience is not deteriorated and the overall energy cost and monetary cost is reduced. In this work we propose a prefetching approach that is able to predict what webpages the user will visit more likely in the near future and prefetch them via WiFi network at an appropriate time, such that when the user indeed wants to visit these webpages later on, the user can directly access them from the prefetched buffer instead of accessing them via cellular network.

A large number of previous works on web prefetching [6], [7], [8], [9] are based on the URLs of webpages visited in the past. However, for news prefetch, simple URL-based approach does not work because newly posted news is usually assigned a new unpredictable URL. Thus it is hard to predict what kind of news a user will be interested in purely based on the URLs of the news. In this work, we present a prefetching approach that combines the keyword-based and URL-based approaches. Based on the websites and sections visited in the past, *e.g.*, cnn.com, cnn.com/World, and the keywords in the news, we design an efficient preference learning algorithm to keep track of the user's changing interests for news so that we can prefetch appropriate contents using WiFi.

On the other hand, many web prefetch techniques focus on short-term prefetching [7], [10], [11], which prefetches the webpages to be visited within a few seconds to a few minutes at most. However, in this paper we focus on the long-term prefetching instead which prefetches the webpages much earlier, *e.g.*, half an hour to several hours, before the user visits the webpages. A motivating example for the long-term prefetching is as follows. Assume that a user Bob goes to work at 8:00 am in the morning by train and he is used to read news on his way to the office. Assume there is no WiFi connection on the train and Bob has a WiFi connection at home. By profiling the news types and frequently visited webpages by Bob on the train, we can prefetch these contents for Bob when he is about to leave home for work. Although the contents prefetched might be obsolete or invalided in the future, we assume that the user's experience will not drop by reading the news prefetched about an hour ago.

The research of Li is partially supported by NSF CNS-1035894, NSF ECCS-1247944, NSF ECCS-1343306, NSF CMMI 1436786, National Natural Science Foundation of China under Grant No. 61170216, No. 61228202.

For the long-term mobile prefetching described above, another difficulty for prefetching is to decide when to prefetch the contents. Several challenges make the decision of prefetch timing difficult: (1) if a user has a WiFi network connection now, we need to predict when the user will move outside the coverage region of the WiFi network; (2) even if we know when the WiFi network connection will be lost, we need to estimate the time needed for prefetching the contents so that the prefetched contents are mostly up-to-date when the user reads it; (3) the prefetching activities should not interfere the user's regular network usage activities (otherwise, the user experience will be deteriorated).

In summary, the main contributions of this work are as follows. We design an efficient keyword-based and URL-based preference learning algorithm to predict the news webpage that the user will be interested in. We also present a network connection prediction approach that can start contents prefetching automatically. In addition, we propose a prefetch scheduling algorithm to exploit idle time of the connection to prefetch the webpages. We implemented the prototype system on iPhone and conducted extensive evaluations on the performance of our system. The experimental results show that our system achieves about 60% hit ratio, saves about 50% cellular data usage, and reduces the energy cost by 7%.

II. Related Work

A. Desktop Prefetch

Web prefetching has been studied for a long time. The primary objective of prefetching technique in desktop computer is to reduce the latency of webpage loading and increase the hit ratio of the prefetched webpages.

1) Short-Term Prefetching: Short-term prefetching technique aims at predicting the webpages to be visited immediately say the next one or two webpages. The Markov model has been exploited in the short-term prefetching to predict next webpages to be visited by analyzing the past visit sequence [6], [8], [12]. In HTML5 [13] and Firefox browser [11], a strategy "link prefetching" for website optimization is introduced: the browser can prefetch specified webpages based on the prefetching hints provided by the webpage. It utilizes browser's idle time to download or prefetch documents that the user might visit in the near future. However both Markov-based and link-based approaches predict and prefetch webpages that are closely related to the current webpage being viewed and webpages that are likely to be visited in the following a new web requests; while our purpose is to prefech webpages based on the user's preference and behavior and we make prediction for further future, *e.g.*, next a few hours.

2) Long-Term Prefetching: Long-term prefetching is a technique designed for large proxies and content distribution networks to reduce client's access latency by storing recently referenced content closer to the users [14], [15], [16], [17]. Instead of making prefetching decisions on the recent history of a client, long-term prefetching make predictions based on global object access patterns to identify a collection of valuable objects to replicate to caches and content distribution servers. The basic idea of long term prefetching is to calculate the most popular domains and most popular objects in those domains, and then a web proxy can prefetch those objects to reduce clients' access latency. Thus without global webpage access information, we cannot adapt the long-term prefetching into personal prefetching for our problem.

B. Mobile Prefetch

Compared to desktop prefetching techniques, the prefetching techniques for mobile device need to be energy and data efficient. Several prefetching schemes [18], [19], [20] have been proposed which make prefetch decision by considering the power consumption, data access rate, data update rate, and data size. Specially, Higgins *et al.* [20] presented a cost-benefit analysis to decide when to prefetch based on the performance such as latency reduction, the cost of energy and monetary cost or data usage. Some prefetching schemes [9], [21], [22] also take advantage of the users' spatiotemporal access patterns for web contents. Lymberopoulos *et al.* [9] presented a prefetching scheme that predicts what webpages a user is likely to request as well as when these requests are likely to occur. Parate *et al.* [21] proposed an approach to predict which app will be used next and when it will be used, and then prefetch application content to fast app launch. Kamaraju *et al.* [22] presented a context-aware delivery paradigm that prefetches video contents by exploiting locations and times in which the networks experience excess of resources. However none of the prior prefetching schemes are designed for news webpage prefetching and make prefetch decisions based on the network condition prediction. In particularly, as news webpages are frequently updated and require high freshness, it is desirable that we can delay the prefetching as late as possible, which requires that we can predict the future network condition. Thus priors prefetching cannot be directly adapted to news webpage prefetching.

III. Problem Formulation and Challenges

A. Problem Formulation

Given the user's visited webpages and the past network conditions, we want to prefetch news webpages via WiFi network that the user is likely to visit when the WiFi network is not available. Specially, we aim at achieving the following objectives for the prefetch system:

1) Our system should prefetch the webpages automatically by carefully exploiting the idle network connections to keep the prefetching activities transparent from the end-users while not interfering the users' regular browsing activities.
2) Our system should predict the network connections, *i.e.*, when the user will have a WiFi network access and when the user will leave the current WiFi network.
3) Based on the network prediction, the user's browsing behavior and preference model, we should carefully schedule the prefetching jobs such that the latest news that are likely to be read are prefetched before the user leaves the current WiFi network coverage.

Observe that a user still needs to use cellular network connections when reading news webpages not yet prefetched. An intermediate goal here is to reduce such cellular network connections as much as possible. The ultimate goal is to carefully decide what contents to be prefetched at what time

such that the overall energy consumption and the paid network access are reduced.

B. Challenges

1) News Preference Issue: How to learn the user's preference is a challenging problem. Learning the preference by the URLs visited by the user is not accurate for news prefetching, as a user always reads newly posted news with unpredictable new URLs. Keywords extraction has also been widely used for understanding webpages [23], [24] and webpage prefetching [25], [26]. Unfortunately these techniques cannot be directly applied here since analyzing the whole news webpages word by word will cost a large amount of energy and might not find the keywords in which the user are really interested. Besides, the keyword list maintaining is also challenging since the keyword list will keep growing as the time goes on and some keywords will become obsolete.

2) Performance Issue: Fast loading of a webpage is what the user desires, so the prefetching is supposed not to interfere the user's normal browsing considering the limited bandwidth of the mobile device. It is better that we prefetch contents when there are no regular networking activities: prefetching should be interleaved among regular data transmission. In order to exploit the period after the data transmission is completed, we need to estimate when the next regular web request will occur and how long a prefetching job will take as a user often experiences various networking speeds in different networking environments (*e.g.*, time, location, and APs).

3) Energy Issue: Due to the limited lifetime of the battery of mobile devices, energy consumption is an important consideration for the mobile prefetching approach. It is well-known that cellular network connection consume more energy than WiFi [5], thus an energy-saving strategy is to visit and prefetch webpages via WiFi connection. However, although the WiFi connection is energy saving compared to the cellular network, both WiFi and cellular network waste some energy when the data transmission is completed due to the "tail energy" [27], [28]. The mobile device is at high powering setting when transmitting data via WiFi or cellular network, and after the mobile device completes the transmission it will remain in the high powering setting for a period of time, which costs more energy than in normal setting. In order to exploit the tail energy to prefetch and further save energy, we need to learn the user's behavior such as the time separation between two normal consecutive networking activities and how long the network connection is idle.

On the other hand, prefetching useless news that the user will not read in the future also wastes a certain amount of energy. Though the more we prefetch, the higher possibility we prefetch all the contents to be accessed, but the more energy are consumed. Thus we need to balance the amount of the webpages to prefetch and prefetching accuracy.

IV. System Design

A. System Architecture

Fig. 1 illustrates our system architecture. In our system, the preference learning module learns the user's preference and predicts what webpages the user will visit based on the

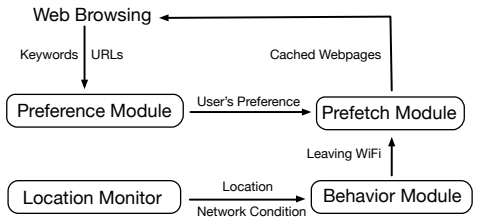


Fig. 1: System Architecture

preference model. Location monitor module runs in the background to keep track of when the user enters and leaves certain Wifi coverage areas. Based on the historical log provided by the location module, the behavior learning module will predict when the user is about to leave the WiFi coverage area and trigger the prefetching module to start prefetching. If the prefetching module is triggered to start, it will search for the webpages that are likely to be visited in the future based on the prediction results provided by the preference learning module. After extracting the URLs of the webpages to be visited, the prefetching module then schedules these prefetching jobs according to the user's browsing behavior provided by the behavior learning module.

To predict the webpages the user will visit in the future, we need to learn the user's preference based on the news the user has read. Besides, in order to effectively schedule the prefetching jobs, we also need to learn the user' s browsing behavior. Table I summarizes the data we collect to learn the user's preference and behavior.

TABLE I: Data Collected for Preference and Behavior Learning

Learning	Data
Browsing Behavior	Time interval between two web requests, Enter time, Leave time of the webpages
Browsing Preference	Keywords in the title, visited URLs

B. Network Environment Prediction

The first challenge to be solved in our system design is to decide when to prefetch webpages for the user. Although the user wants to read the latest news, but for most users they will be also interested in the news posted several minutes or hours ago. So prefetch the "old" news is reasonable and meaningful to the user. But we still want the news prefetched to be relatively new, so we begin the prefetching as late as possible while ensuring the prefetching can be finished before the user leaves the WiFi. To do this, we need estimate how long the user will stay in the coverage area of the current WiFi network. For most of the people, the daily schedule and weekly schedule do not change a lot and hence it is possible to do such network coverage prediction.

Monitoring the network connection all the time sometimes may give us misleading information about how long the user stays in the WiFi area, because it is possible that in certain locations in the WiFi coverage area, the WiFi access is temporarily unavailable. Thus instead of monitoring the network connection, we monitor the user's current location to estimate when he or she leaves a certain area. The location module is available in most of the mobile devices that can help us to monitor when the user enters of leaves a certain area.

We observe that the duration when a user stays in a network depends on the network he connects or the location of the network, and the time he enters the networks. Thus, to precisely estimate how long a user will stay in a certain area, we will keep track the following information $\langle t_i, d_i, L_i, SSID_i \rangle$, where t_i is the time the user entered the coverage area of the network with SSID_i at location L_i, and the user stayed in this network for a duration of d_i. We round time t_i to hours. Based on the collected networking access data, we then predict when the user will leave the current Wifi network.

As a result, for the same WiFi network we have multiple time duration records for the same entering time. We use a probability-based algorithm to estimate the time the user will leave the current network, such that we can finish the prefetch in time. Let T_i^t be the collection of time the user stays in the WiFi coverage area of $SSID\ i$ after entering at time t, where d is the day (*e.g.*, Monday, Tuesday, *etc.*) and the hour when the user enters the WiFi coverage. Let $p_i^t(d)$ be the probability that the mobile device is connected to this network for over time d after entering at time t. We estimate the time that the user will stay in the current WiFi coverage as

$$\text{argmin}\{d \mid p_i^t(d) > \delta\},$$

where δ is a threshold value (chosen as 0.5 in our implementation).

Given the estimated staying time, it is still necessary to choose an appropriate time during this period to prefetch webpages. We seek to find the time to start the prefetching as late as possible, while at the same time to ensure that the prefetching can be finished in time. Given the estimated staying time d, number of webpages to prefetch l and the average fetch time f, in order to finish prefetching these webpages before the WiFi coverage is not available, we need to start to prefetch no later than $d - lf$.

C. Webpage Access Prediction

1) Preference Learning: News title is always concise, informative and highly related to the content of the news, and the user is always guided by the keywords in the title. Thus by extracting keywords from the titles we can effectively learn the user's preference and predict what news the user will be interested in. Note that users are interested in different keywords to different degree, we quantify the keyword interest by assigning each keyword w with a interest weight $q(w)$. Each time the keyword w appears in the title of the news read by the user, we increase its weight $q(w)$ by a constant c_q

Since people's interests will change as time goes on, newly appearing keywords in the title play more important roles in learning the user's preference than old ones. To deal with the issue of interest changing, we use a simple time decay function to keep reducing the keyword's interest weight as time goes on. The decay function works as follows:

1) In each time period t, the keyword w's weight $q(w)$ in the keyword list will be reduced to $q(w)(1-\delta)$, where $1-\delta$ is the decay rate.
2) The keyword w will be removed if its interest weight $q(w)$ is less than the threshold ϵ.

Not only the decay function helps us capture the user's current interests, but also contributes to reducing the size of the keyword list. To effectively maintain the keyword list, we use the heap data structure to store the keywords. In every time period we will check whether the root's interest weight is less than ϵ, if so we remove it from the keyword list and heapify the keyword list, and then check the new root's interest weight until the root's weight is larger or equal to ϵ.

In our keyword maintaining approach, the keyword w might be removed from the keyword list and later be added again. Thus w's weight $q(w)$ is only accumulated and decayed since the latest time it is added to the keyword list and hence $q(w)$ is less than the "actual" weight. However in our approach, if keyword is removed from the keyword when weight $q(w)$ is smaller that ϵ, the actual weight is also less than a constant of the removal threshold ϵ. Formally, for each keyword w in time interval between time i and n we define a weight function

$$f_i^n(w) = \sum_{j=i}^{n} x_j(w)(1-\delta)^{n-j},$$

where $x(w) \in \{0,1\}$ indicates whether w appears at time t_j. In each time slot, if $f_i^n(w) < \epsilon$, where time i is the first time word w appears since the last removal, the word w will be removed from the keyword list. Then we have the following theorem

Theorem 1. *If a word w is removed from the keyword list at time slot n, then the weight $f_1^n(w) \leq \frac{\epsilon}{1-\epsilon}$.*

Proof: Assume that a word w is added into the keyword list and then removed for n times. Afterwards, at time slot a_i word w is added into the list again and at time slot r_i removed from the list. Let $v_i(w) = f_{a_{i-1}}^{r_{i-1}}$ and $t_i = r_i - a_i$, then we have

$$\begin{aligned} f_1^n(w) &= \sum_{i=1}^{n} v_i(w)(1-\delta)^{\sum_i^n t_i} \\ &\leq \epsilon \sum_{i=1}^{n} (1-\delta)^{\sum_i^n t_i} = \frac{\epsilon}{1-\epsilon} \end{aligned}$$

This finishes the proof. ■

Since word w is added at time a_i and removed at time r_i, $(1-\delta)^{t_i}$ must be less than ϵ and hence $\sum_{i=1}^{n}(1-\delta)^{\sum_i^n t_i} \leq \frac{1}{1-\epsilon}$.

However, learning the user's preference only based on keywords is no enough. We observe that the user's preference on news is also related to sections of the news website where the news is posted. For example, if a user always visits the sports section, we can infer that he might be interested in sports and hence the news in this section will be read by the user with higher probability. Thus we also keep track of URLs of the sections and subsections visited frequently by the user such that we can prefetch the news from these sections. The news website is always organized as a tree structure as shown in Figure 2 and each section always has a fixed URL. Similarly to the keywords, we assign a interest weight $t(s)$ to the URL of each section s. We keep track of the times the section s is visited by the user and each time the user visits the section s, we increase the weight $t(s)$ by a constant c_t. Besides, the

weights of URLs are also decayed as time goes on using the same approach for the keyword.

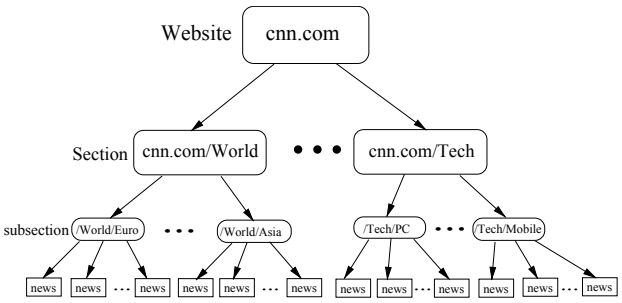


Fig. 2: Website Architecture

With the interest weights of keywords and URLs of the sections, we define the interest weight for each news according to the keywords appearing in the headline and the section where the news is posted. The higher the weight is the more possible the user will read it. Assume that keywords $w_1, ..., w_n$ appear in the news h's title, and the weight of the section where the news is posted is $t(s)$, then we set h's weight as

$$w(h) = \sum_{i=1}^{n} q(w_i) + t(s)$$

2) News Searching: If the prefetching module is about to start prefetching, it will search for the news in the sections frequently visited by the user.

Given the news searching results, we need to decide which news webpages to prefetch. To make prefetching decision, we compare the news's weight with a prefetch threshold, if it is greater than the threshold, we then add the URL of the news webpage to the prefetch queue. In order to set the value of the prefetch threshold appropriately, we keep track of the weight of each news the users has read and set the threshold as follows. Let p be the expected probability we set that the prefetched webpages will be accessed, n be the total number of webpages the user visited and $v(w)$ be the number of webpages whose weight is above w, then we set the threshold weight s as

$$\text{argmin}\{w \mid v(w) > pn\}.$$

3) Scheduling Prefetching: Given the prediction result from preference learning module and behavior learning module, we know the time when the user will leave the current WiFi coverage area and the collection of URLs to prefetch. Due to the limited bandwidth of the mobile devices, prefetching should not interfere user's normal web browsing as aforementioned. Observing that the network connection is idle between two web requests, we exploit the interval between two web requests to prefetch webpages, as shown in Figure3. Moreover, exploiting the idle interval to prefetch also leverages the tail energy.

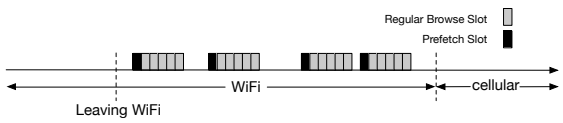


Fig. 3: Prefetch Scheduling

Before scheduling webpage prefetching, we need to learn how long it takes for fetching a webpage. Our extensive evaluations (downloading more than 300 webpages) at three different locations show that the fetch times for webpages do not have a large variance. For simplicity, in this work, we use the mean fetch time as the estimated time for prefetching any news webpages.

When scheduling prefetching jobs, we divide the time domain into time-slots of fixed length, and assume that fetching one webpage costs one time-slot. Formally, let $a(u) = [a_0, a_1, ..., a_n] \in \{0,1\}^n$ be the webpage access sequence by the user, and $p(u) = [p_0, p_1, ..., p_m] \in \{0,1\}^n$ be the prefetch sequence. In order to not interfere the user' s normal browsing, for each time t, we should have $a_t + p_t \leq 1$. To fully exploit the idle time when the user is browsing news, we need to maximize $\sum_{i=1}^{n} a_i + \sum_{j=1}^{m} p_j$.

In our system, we maintain a queue of the webpages to prefetch. We first sort the webpages in prefetch queue in decreasing order of the weights such that we could first prefetch the webpages that are more likely to be accessed in case that we do not have enough time to prefetch all the webpages in the prefetching queue. Each time the webpage loading is finished and the network connection is idle, prefetching module will schedule a batch of webpages in the prefetch queue. In particular, let the average idle time interval the user spend on reading one webpage be d and the average fetch time for prefetching a webpage be t, then we schedule d/t jobs from the prefetch queue after each normal webpage request.

We specify a maximum size for the prefetching cache and apply the same time decay function to the interest weights of prefetched webpages in cache. When the cache is full and we have webpages to prefetch in the prefetch queue, we will remove the cached webpages with interest weights lower than those in the prefetch queue.

4) Visit Un-prefetched Webpages Without WiFi: Despite our personalized preference learning algorithm, it is unavoidable that the user will visit webpages which have not been prefetched in the cache when out of the WiFi coverage. In order to further reduce the energy consumption, for the above cache-miss cases we leverage the peer-to-peer (P2P) connection to provide temporary network access. Several prior works [29], [30] have proposed solutions for a mobile device who has access to the Internet via cellular connection to share its network access with nearby devices via WiFi connection, such as Bluetooth and WiFi Direct, which are more energy efficient than cellular connection. Moreover, through connections with nearby users, it is also possible to find other users sharing similar interests and preferences [31], [32], [33] and to get webpages prefetched by those users via P2P connection.

To exploit nearby mobile devices' network accesses and prefetched contents, the first step is to quickly discover and build connections with other devices in the vicinity. We design a simple and efficient method for quickly building connections based on the protocols proposed in [34]. The user first sets a number k of connections to be built and then our system will ask nearby devices to ACK with their MAC addresses so that the connections can be built. The basic steps are as follows.

Step-1: The user's device broadcasts a message including the size m of a frame which is a set of continuous time slots.

Step-2: A nearby device who received the message sends an ACK message including its MAC address randomly at one of the m slots.

Step-3: If the user has obtained enough MAC addresses from nearby devices, it will send out an END messages; otherwise repeat step 1 and 2.

Since collisions will happen when multiple devices ACK simultaneously, we can receive at most one ACK per slot, thus we need at least $\Theta(k)$ slots to connect to k neighbor devices. Let n be the number of nearby devices, based on the Hoeffding inequality, our theoretical analysis shows that by carefully choosing the frame size we can achieve the optimal latency and we have the following lemma.

Lemma 2. *When $n \in [a_1 k, a_2 k]$ for constants $1 < a_1 < a_2$, the optimal frame size is $m = n$ and the latency to discover k neighbors is $\Theta(k)$ w.h.p..*

V. System Implementation and Evaluation

We implemented the prototype system in iPhone 4 with iOS 5.1.1. In order to monitor when the user enters and leaves the WiFi coverage area, we use a service "region monitoring" provided by iOS[35]. In iOS 4.0 and later, applications can use region monitoring to be notified when the user crosses geographic boundaries. We use this feature to keep track of the time the user enters and leaves certain WiFi areas. When the user enters a WiFi coverage area for the first time, the user needs to register this area with its SSID in our system and the radius of the WiFi area is set to be 100 meters. Before the system starts to prefetch webpages, it will send a notification to the user to ask for the permission. In addition, we also allow the user to start prefetching manually.

A. Browsing Behavior Learning

1) Web Request Interval and Web Fetch Time: We first present the user's browsing behavior statistics when using our system. We have three student users use the prototype system for two weeks. Everyday they browse news webpages using our system for about half an hour. Figure 4 shows the time intervals between two web requests when the user browses news webpages. For 65% of the time intervals, the lengths of the intervals are less than 1.5 second. These short intervals usually appear on the way the user goes to the destination section after entering the news website, because the user only stays on intermediate webpages for a short time. These short intervals also appear when the user switches from one section to another section. For the rest 35% time intervals, the average length is 61 seconds, which is the average time the user spends on reading one piece of news.

Figure 5 plots the time of fetching webpages via WiFi network. As we can see, for most of the web requests, the time of fetching a webpage via WiFi network is short. It takes less than 1 second to fetch a webpage for over 94% of the web requests. This is because that news webpages designed for the mobile devices usually simply consists of words and one or two pictures. Thus the size of a webpage is small and hence it takes very little time to fetch a webpage via the high speed WiFi connection.

2) Keyword Maintaining: Figure 6 shows the interest weights of the keywords that have appeared in the title of news read by the user. In our experiment, we set the removal threshold to 0.2 and the decay period to 3 minutes. Here we define the **compression rate** as the proportion of the number of the keywords maintained in the keyword list to the total keywords that have appeared in news titles. Our experiment shows that we achieve the compression rate of 81%

B. Performance

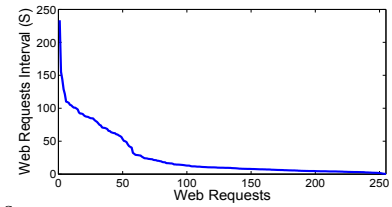
1) Network Condition Prediction: In our experiment, the student users' daily schedules are almost fixed and hence the time the users stay in a certain WiFi coverage does no vary a lot. As mentioned above, our system will send a notification before the user's estimated leaving time and ask for the permission to start to prefetch. We assume that the network prediction is correct if the user agrees the prefetch request and we achieve accuracy rate of 80% in a week for one user.

2) Hit Ratio and Waste Ratio: Prefetching systems are often evaluated in terms of the **hit ratio** and **waste ratio**. Hit ratio refers to the proportion of the number of prefetched webpages that are accessed by the user to the total requested webpages; waste ratio refers to the proportion of the number of undesired prefetched webpages to the total prefetched webpages. Figure 7 plots the hit ratio of our system. As we can see the hit ratio is relatively stable and around to 60% on average. Although at the beginning phase, our system does not learn the user's preference precisely, we achieve about 90% hit ratio. The reason for the high hit ratio at the beginning phase is that the prefetch threshold is low due to the low weights of the news the user read and large amount of webpages are prefetched. Besides, at the beginning phase, the weight of the section is dominating when make prefetch decisions and most of the news in the user's favorite section are prefetched. Thus we can still achieve high hit ratio at the beginning phase.

Figure 8 plots the waste ratio of our system. The waste ratio here is calculated for each batch of prefetched webpages. Compared with the hit ratio, the waste ratio continuously decreases. As time goes on, the user is more likely to access to the webpages from the sections with large weight and also to access the news with titles containing keywords with large weight. Thus the prefetch threshold become larger and less webpages are prefetched. However since our system learn the user's preference more precisely, the webpages prefetched are more likely to be accessed by the subscriber and the waste keeps decreasing.

In our prototype system, we remove prefetched webpages and the clean the prefetch cache everyday. Figure 9 plots the everyday's cache usage in 9 days. Similarly to the waste ratio, the cache size is relatively large at the beginning. As the system learns user's preference more precisely and the prefetch threshold become larger, less webpages are prefetched and the cache size keeps decreasing. For the last 5 days the size of the prefetch cache is about 11 MB per day.

3) Energy and Data Consumption: We use the tool *instruments*[36] provided in MacOS to evaluate the energy consumption. The energy consumption in iPhone is divided into 20 levels in *instruments*. In our experiment, we browse the



s

Fig. 4: Web Request Interval

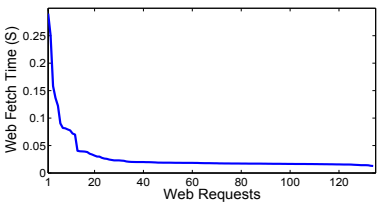


Fig. 5: Webpage Fetch Time

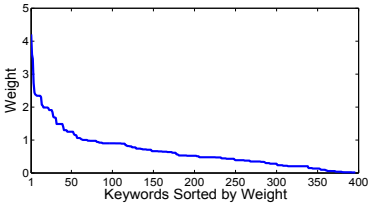


Fig. 6: Keyword Weight

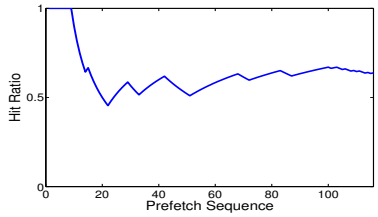


Fig. 7: Hit Ratio

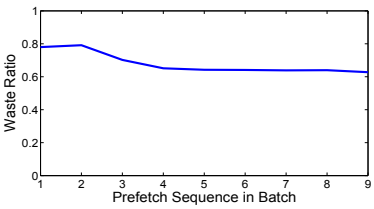


Fig. 8: Waste Ratio

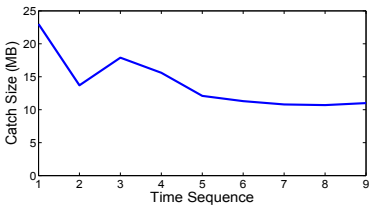
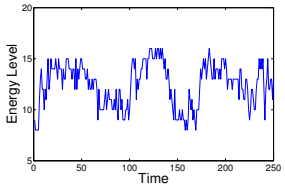


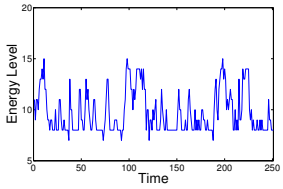
Fig. 9: Cache Size

same webpages using our system with prefetch feature enabled and disabled respectively. In this set of experiments we first run the system for about 10 minutes via WiFi connection and then via cellular connection for 20 minutes, such that when the prefetch feature is enabled our system will prefetch webpages via WiFi network in the first 10 minutes.

Figure 10(a) and Figure 10(b) plot the energy consumption when the user browses webpages via WiFi network with prefetch feature enabled and disabled respectively. In both figures the peaks appear when the browser loading webpages, and after loading the webpage the energy level decreases to a low level. As we can see, there are some time slots where the energy level is higher after webpage loading, this is caused by scrolling the webpage which makes energy level increase due to display contents change. With prefetch feature enabled, the device stays at high energy level for longer time than the one with prefetch feature disabled. This is because our system is prefetching webpages in the background. However, as we can see that even with prefetch feature disabled, the energy level does not fall to a low energy level instantly due to the effect of tail energy. The average energy level with prefetch feature enabled and disabled is 12.5 and 9.6.



(a) Prefetch Enabled



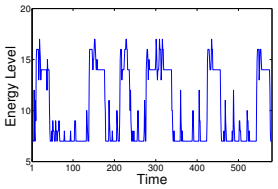
(b) Prefetch Disabled

Fig. 10: Energy Consumption via WiFi Network

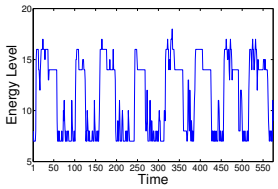
Figure 11(a) and Figure 11(b) show the energy consumption when our system is connected via cellular network. In Figure 11(a) after loading a webpage from prefetch cache, the energy level falls instantly to the low level. However, in Figure 11(b), after fetching a webpage via cellular network, although the energy level falls to a lower level, it still higher than that when the webpage is loaded from the prefetch cache. The average energy level when when prefetch feature is disabled is about 11.8 while the energy level is 10.2 when prefetch feature enabled.

Let $e(p) = e_w(p) + e_c(p)$ and $e(n) = e_w(n) + e_c(n)$ be the energy cost when the prefetch feature enabled and disabled, where e_w and e_c is the energy cost via WiFi connection and cellular connection respectively. We then calculate e_p/e_n as the energy cost reduction. The result shows that our system consume less energy when prefetch enabled than that when disabled and we achieve about 7% of the energy reduction.

During the data usage evaluation, we browsed 100 webpages with prefetch feature enabled and disabled respectively. When the prefetch feature is enabled, 30% of pages are not prefetched and all the other webpages are prefetched via Wifi network, which consumed 2 MB cellular data. When the prefetch feature is disabled, all of the webpages are fetched via the cellular network and the cellular data usage is 5 MB, which is over 2 times of the one with prefetch feature enabled.



(a) Prefetch Enabled



(b) Prefetch Disabled

Fig. 11: Energy Consumption via Cellular Network

VI. Discussion

There are several interesting questions left for our future research. First of all, as most of existing popular web browsers support plugins and provide open APIs, we plan to implement our prototype system as a plugin, such that there is no need for users to switch to new browsers. Besides, we can also take advantage of the cloud service to perform fine-grained analysis on the user's preference and behavior [37], [38] by securely uploading user's visiting history and daily schedule, to further improve the prediction accuracy. Furthermore, by carefully considering the connection or channel quality [39], [40] it is possible to optimize the communication cost and further reduce the energy consumption.

VII. Conclusion

In this paper we designed a network-agile preference-based prefetching method for mobile devices. We implemented our method in iPhone and conducted extensive evaluations on the performances of our methods. Our evaluations show that our prefetching based approach is able to reduce the cellular network access by about 50% and reduce the energy cost by about 7%.

References

[1] K. Mateus, "Web usage prediction: When mobile and desktop collide," http://www.mequoda.com/articles/new-media-trends/web-usage-prediction-when-mobile-and-desktop-collide.

[2] "Applications capture already half of mobile internet traffic," http://www.zokem.com/2010/09/applications-capture-already-half-of-mobile-internet-traffic/.

[3] "Adobe mobile experience survey: What users want from media, finance, travel & shopping," http://www.synergetechsolutions.com/whos-nearme.aspx, 2010.

[4] W. Lehr and L. W. McKnight, "Wireless internet access: 3G vs. WiFi?" *Telecommunications Policy*, vol. 27, no. 5, pp. 351–370, 2003.

[5] N. Balasubramanian, A. Balasubramanian, and A. Venkataramani, "Energy consumption in mobile phones: a measurement study and implications for network applications," in *ACM SIGCOMM*, 2009.

[6] R. R. Sarukkai, "Link prediction and path analysis using markov chains," *Computer Networks*, vol. 33, no. 1, pp. 377–386, 2000.

[7] X. Jin and H. Xu, "An approach to intelligent web pre-fetching based on hidden markov model," in *Decision and Control, 2003. Proceedings. 42nd IEEE Conference on*, vol. 3, pp. 2954–2958.

[8] M. Deshpande and G. Karypis, "Selective markov models for predicting web page accesses," *ACM Transactions on Internet Technology (TOIT)*, vol. 4, no. 2, pp. 163–184, 2004.

[9] D. Lymberopoulos, O. Riva, K. Strauss, A. Mittal, and A. Ntoulas, "Pocketweb: instant web browsing for mobile devices," in *ACM ASPLOS*, 2012.

[10] X. Dongshan and S. Junyi, "A new markov model for web access prediction," *Computing in Science & Engineering*, vol. 4, no. 6, pp. 34–39, 2002.

[11] "Link prefetching," https://developer.mozilla.org/en-US/docs/Web/HTTP/Link_prefetching_FAQ.

[12] F. Khalil, J. Li, and H. Wang, "Integrating recommendation models for improved web page prediction accuracy," in *Proceedings of the thirty-first Australasian conference on Computer science*, 2008.

[13] "Html5 specification," http://www.w3schools.com/html5/default.aspLL.

[14] E. P. Markatos and C. E. Chronaki, "A top-10 approach to prefetching on the web," in *Proceedings of INET*, vol. 98, 1998, pp. 276–290.

[15] S. W. Shin, B. H. Seong, and D. Park, "Improving world-wide-web performance using domain-top approach to prefetching," in *The Fourth International Conference/Exhibition on High Performance Computing in the Asia-Pacific Region, 2000.* IEEE.

[16] A. Venkataramani, P. Yalagandula, R. Kokku, S. Sharif, and M. Dahlin, "The potential costs and benefits of long-term prefetching for content distribution," *Computer Communications*, vol. 25, no. 4, pp. 367–375, 2002.

[17] B. Wu and A. D. Kshemkalyani, "Objective-optimal algorithms for long-term web prefetching," *Computers, IEEE Transactions on*, vol. 55, no. 1, pp. 2–17, 2006.

[18] L. Yin, G. Cao, C. Das, and A. Ashraf, "Power-aware prefetch in mobile environments," in *22nd International Conference on Distributed Computing Systems, 2002*. IEEE, pp. 571–578.

[19] H. Song and G. Cao, "Cache-miss-initiated prefetch in mobile environments," *Computer Communications*, vol. 28, no. 7, pp. 741–753, 2005.

[20] B. D. Higgins, J. Flinn, T. Giuli, B. Noble, C. Peplin, and D. Watson, "Informed mobile prefetching," in *ACM MobiSys*, 2012.

[21] 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 *ACM UbiComp*, 2013.

[22] P. Kamaraju, P. Lungaro, and Z. Segall, "A novel paradigm for context-aware content pre-fetching in mobile networks," in *Wireless Communications and Networking Conference (WCNC)*. IEEE, 2013.

[23] W.-t. Yih, J. Goodman, and V. R. Carvalho, "Finding advertising keywords on web pages," in *ACM WWW*, 2006.

[24] D. W. Embley, D. M. Campbell, Y. S. Jiang, S. W. Liddle, D. W. Lonsdale, Y.-K. Ng, and R. D. Smith, "Conceptual-model-based data extraction from multiple-record web pages," *Data & Knowledge Engineering*, vol. 31, no. 3, pp. 227–251, 1999.

[25] C.-Z. Xu and T. I. Ibrahim, "Towards semantics-based prefetching to reduce web access latency," in *Applications and the Internet, 2003. Proceedings. 2003 Symposium on*. IEEE, 2003, pp. 318–325.

[26] C.-Z. Xu and T. I. Ibrahim, "A keyword-based semantic prefetching approach in internet news services," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 16, no. 5, pp. 601–611, 2004.

[27] A. Balasubramanian, R. Mahajan, and A. Venkataramani, "Augmenting mobile 3G using WiFi," in *ACM MobiSys*, 2010.

[28] H. Liu, Y. Zhang, and Y. Zhou, "Tailtheft: Leveraging the wasted time for saving energy in cellular communications," in *Proceedings of the sixth international workshop on MobiArch*. ACM, 2011, pp. 31–36.

[29] L. Keller, A. Le, B. Cici, H. Seferoglu, C. Fragouli, and A. Markopoulou, "Microcast: cooperative video streaming on smartphones," in *ACM MobiSys*, 2012.

[30] N. Do, C.-H. Hsu, and N. Venkatasubramanian, "Crowdmac: a crowd-sourcing system for mobile access," in *Proceedings of the 13th International Middleware Conference*, 2012.

[31] Z. Li, C. Wang, S. Yang, C. Jiang, and X.-Y. Li, "Lass: Local-activity and social-similarity based data forwarding in mobile social networks," *IEEE TPDS*, 2014.

[32] X. Zhang, Z. Yang, Z. Zhou, H. Cai, L. Chen, and X.-Y. Li, "Free market of crowdsourcing: Incentive mechanism design for mobile sensing," *IEEE TPDS*, 2014.

[33] L. Zhang, J. T. Li, Xiang-Yang, and Y. Liu, "Message in a sealed bottle: Privacy preserving friending in social networks," in *IEEE Transaction on Mobile Computing. Oct*, 2014.

[34] J. Han and X.-Y. Li, "Pickup game: Acquainting neighbors quickly and efficiently in crowd," in *IEEE MASS*, 2014.

[35] "Regionmonitor," https://developer.apple.com/library/ios/documentation/userexperience/conceptual/LocationAwarenessPG/RegionMonitoring/RegionMonitoring.html.

[36] "Instruments," http://developer.apple.com/library/mac/#documentation/DeveloperTools/Conceptual/InstrumentsUserGuide/Introduction/Introduction.html.

[37] C. Bo, L. Zhang, X.-Y. Li, Q. Huang, and Y. Wang, "Silentsense: silent user identification via touch and movement behavioral biometrics," in *ACM MobiCom*, 2013.

[38] C. Bo, X. Jian, X.-Y. Li, X. Mao, Y. Wang, and F. Li, "You're driving and texting: detecting drivers using personal smart phones by leveraging inertial sensors," in *ACM MobiCom*, 2013.

[39] B. Li, P. Yang, J. Wang, Q. Wu, S. Tang, X.-Y. Li, and Y. Liu, "Almost optimal dynamically-ordered channel sensing and accessing for cognitive networks," *IEEE Transactions on Mobile Computing*, 2013.

[40] Y. Zhou, X.-Y. Li, F. Li, M. Liu, Z. Li, and Z. Yin, "Almost optimal channel access in multi-hop networks with unknown channel variables," in *IEEE ICDCS*, 2014.