Energy-Efficient Transmission Scheduling in Mobile Phones using Machine Learning and Participatory Sensing

Zaiyang Tang, Song Guo, *Senior Member, IEEE,* Peng Li, *Member, IEEE,* Toshiaki Miyazaki, *Senior Member, IEEE,* Hai Jin, *Senior Member IEEE,* and Xiaofei Liao, *Member IEEE*

***Abstract*—Energy efficiency is important for smartphones because they are powered by batteries with limited capacity. Existing work has shown that the tail energy of the 3G/4G network interface on a mobile device would lead to low energy efficiency. To solve the tail energy minimization problem, some online scheduling algorithms have been proposed, but with a big gap from the offline algorithms that work depending on the knowledge of future transmissions. In this paper, we study the tail energy minimization problem by exploiting the techniques of machine learning and participatory sensing. We design a client-server architecture, in which the training process is conducted in a server, and mobile devices download the constructed predictor from server to make transmission decisions. A system is developed and deployed on real hardware to evaluate the performance of our proposal. The experimental results show that it can significantly improve the energy efficiency of mobile devices while incurring minimum overhead.**

***Index Terms*—Energy Efficiency; Participatory Sensing; Machine Learning**

I. Introduction

Recent research has shown that a big portion of energy is consumed by 3G/4G network interface among all components of a mobile device [1]. A major reason is so called *tail energy* that the 3G/4G interface will consume energy by remaining some time in a high-power state after packet transmissions, accounting for about 60% of energy consumption in data transmission [2], [3]. For example, according to the 3GPP regulations [4], a Radio Resource Control (RRC) protocol of 3G includes three states: *IDLE*, *DCH*, and *FACH*, representing idle, dedicated and forward access channel respectively. State *DCH* provides high throughput and low delay for transmission at the cost of high power consumption. State *FACH* consumes about half power of *DCH* for sporadic data transmissions on shared channels with other devices. State *IDLE* consumes about 1% power of *DCH* [2]. The state switch is controlled by inactivity timers [5]. The 3G interface does not stay in *FACH* for a long time. The link must change to *DCH* upon arrivals of any data transmission request. After the completion of a data transfer, link keeps consuming energy by staying on *DCH* for about 5 seconds and then *FACH* for about another 5 seconds, before moving to the *IDLE* state. Another prevalent standard 3GPP2 [6] has similar problem [5], [7], which appears to be worse for LTE network [22].

To reduce tail energy, several transmission scheduling algorithms [2], [8] have been proposed to delay transmission requests such that they can form consecutive transmissions over 3G/4G interface. Compared with the default scheduling which transmits data immediately as they arrive, these algorithms have shown effective in improving energy efficiency. However, none of these online scheduling can achieve a similar performance as the offline algorithms that posses the knowledge of the whole transmission sequence.

In this paper, we apply the technique of machine learning (ML) to improve the energy efficiency of 3G/4G transmissions in mobile phones. It is motivated by the observation that requests belonging to an application usually show a certain pattern that can be extracted by machine learning, and be further used to improve the performance of transmission scheduling. A straightforward method to implement such a machine learning based approach is to let each mobile device individually create its own predictor. However, it suffers from the following two major weaknesses. First, to guarantee a certain level of prediction accuracy, the training process needs a large number of transmission records, which are not always available to mobile nodes due to the cold-start problem. Second, the training process is resource-hungry because of its high computational complexity and large storage occupation by training records.

To eliminate the above weaknesses, we propose a client-server architecture by exploiting the participatory sensing technique. In our architecture, a centralized server collects transmission records contributed by many mobile devices, and creates the predictor using machine learning algorithms. Then, each mobile device downloads the predictor from the server, and applies it to make transmission decisions. The main contributions of our paper are summarized as follows.

- We propose a novel approach to improve energy efficiency of mobile devices by exploiting the techniques of machine learning and participatory sensing. Specifically, we apply supervised learning methods to train a predictor, based on the historical transmission records, which will categorize future requests into two classes: the requests that should be immediately transmitted, and the requests that can be delayed for a period.
- We propose a client-server architecture to implement our proposed ML-based approach.

P. Li, S. Guo and T. Miyazaki are with the School of Computer Science and Engineering, The University of Aizu, Japan.

Z. Tang, H. Jin and X. Liao are with the School of Computer Science and Technology, Huazhong University of Science and Technology, China.

- A system is developed and deployed on real hardware to evaluate the performance of our proposals. The experimental results show that our proposed approach can effectively improve the energy efficiency of mobile devices.

The rest of the paper is organized as follows. Section 2 briefly reviews the related work. Section 3 describes system model and problem statement. The architecture is presented in Section 4, followed by the machine learning algorithm design in Section 5. The system implementation and evaluation are given in Section 6. Finally, Section 7 concludes this paper.

II. Related work

A. Energy consumption

The energy consumption problem in mobile phones has attracted significant attentions in recent years. The impact of different energy saving techniques [9], [5], [7] has been studied in 3G networks using analytical models. It has been also addressed for specific smartphone applications. Xiao et al. [10] measure the energy consumption for video streaming application in mobile phones using both WiFi and 3G. Nurminen et al. [11] measure the energy consumption for peer-to-peer application over 3G. Naredran et al. [12] present an infrastructure for measuring the precise energy consumption of operations using network (e.g., web browsing, email, blogging, news, and social networking).

To reduce tail energy, two approaches have been studied. One is on the tail time tuning. For example, Falaki et al. [13] show that setting the tail time to 4.5 seconds will significantly reduce energy consumption, while Chuah et al. [9] suggest to set it as 10 to 15 seconds so as to limit the probability of state promotion below 5%. Dynamic tail time tuning is addressed in [14], [5] by exploiting traffic patterns to make a better tradeoff between tail time and state promotion. The other approach is to aggregate small transmissions into large ones such that the occurrence of tails can be reduced. Balasubramanian et al. [2] develop a protocol called *Tail Ender* that reduces energy consumption of common mobile applications. Liu et al. [8] propose a dual queue scheduling algorithm to reduce the tail time. In this paper, we focus on the transmission scheduling which aggregates small transmissions into large ones to reduce tail energy.

B. Machine learning techniques in network traffic modeling

Machine learning (ML) techniques have a wide range of applications with four basic types of learning [15]: Classification (or supervised learning), Clustering (or unsupervised learning), Association, and Numeric Prediction. Machine learning techniques have been also intensively explored in network modeling such as traffic classification and prediction [16], [17], [18], [19], [20].

Broadly defined, ML is the process of exploiting and describing structural patterns in a supplied dataset. ML takes input in a form of dataset of instances, each instance referring to an individual, independent example of the dataset, and characterized by the features that quantitively measure its different aspects. The output is the description of the knowledge that has been learnt. How the specific outcome of the learning process is represented depends largely on the particular ML approach used. In network traffic prediction and classification, ML techniques focus on supervised learning or unsupervised learning (clustering) [21].

III. System model and problem statement

A. System model

In a cellular network with a number of smartphone users, we consider any user generates a sequence of uplink transmission requests denoted as $R = \{r_0, r_1, ..., r_n\}$. One and only one transmission will be scheduled in a unit time slot. For any $r_i \in R$, we use a_i and d_i to denote its arrival time-slot and the latest time-slot, at which it must be delivered due to a delay requirement, respectively.

Let $S = \{s_0, s_1, ..., s_n\}$ be an arbitrary feasible scheduling of R, i.e., the scheduled transmission slot s_i for request r_i satisfies $a_i \leq s_i \leq d_i$. We note that the delay of request r_i, defined as

$$\varphi_i = s_i - a_i, \tag{1}$$

places a critical role in energy efficiency because after a transmission. In 3G and LTE networks, the interface will stay at a high energy consumption state for a tail time τ until the next transmission. Suppose there are two successive data transmissions s_{i-1} followed by s_i with an interval of $s_i - s_{i-1}$. If the interval is long enough (i.e., $s_i - s_{i-1} - 1 > \tau$), the tail time is τ since the interface will turn to *IDLE* to save energy after τ. The energy saving will be obtained by reducing the total tail time T that can be calculated as:

$$T = \sum_{i=1}^{n} \min\left\{\tau, s_i - s_{i-1} - 1\right\}, \tag{2}$$

B. Problem statement

For a given sequence of requests, the objective is to find the transmission slot for each request r_i, i.e., the transmission delay φ_i, such that the total tail time is minimized. An offline scheduling finds φ_i with the knowledge of the whole request sequence R, while an online algorithm estimates φ_i by the knowledge of the requests that have arrived prior to r_i.

Consider a sequence of three requests with arrival times $\{a_1, a_2, a_3\}$ as shown in Fig. 1. Let t_0 denotes the starting time slot. Here the unit tail time after each transmission is τ. Without knowing the arrival time of request r_2, the online scheme transmits r_1 upon its arrival. In contrast, r_1 will be deferred until r_2 arrives by an offline scheduling, as long as the delay requirements are met. Therefore, the energy that would be wasted in the tail time of r_i is avoided.

While online scheduling can improve performance, offline scheduling is hardly applicable. This is because the transmission requests arrive randomly and their transmission deadline will be missed if the scheduling decisions are made after collecting a bunch of requests by an offline algorithm.

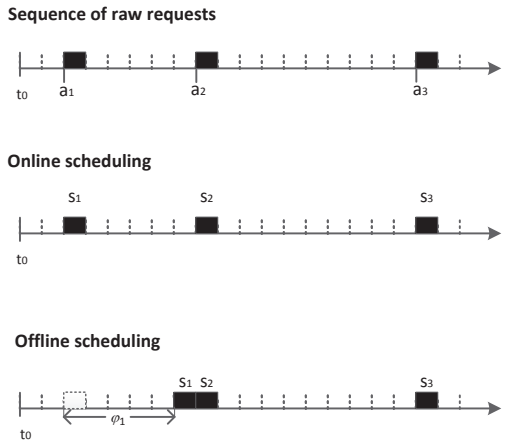


Fig. 1. Illustration of online and offline scheduling schemes.

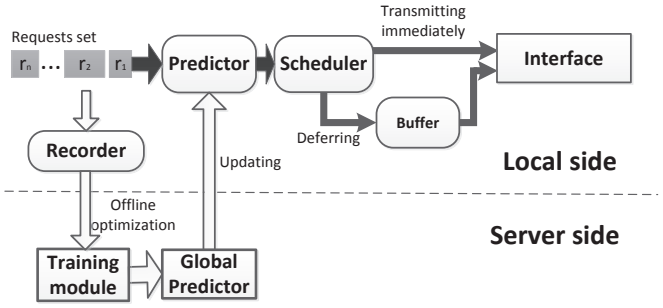


Fig. 2. Architecture.

To improve the performance of online scheduling, machine learning and participatory sensing are exploited in this paper. The system takes transmission requests from multiple users as inputs, over which the delay predictor is then trained. Based on the predicted delay parameters, our proposed online scheduling will show a much improved energy saving performance over the deterministic online scheduling.

IV. Architecture

To implement our machine learning and participatory sensing based solution, the system has a client-server based architecture. Initially, each client uploads its raw request sequence to the server periodically. The server collects requests from multiple clients and makes the training offline. After a predictor is generated on server and then deployed to clients, each client will use it to make online scheduling.

A. Client design

As shown in Fig. 2, the client part consists of five modules that are elaborated in the following.

Recorder. The recorder is responsible for logging the information of coming requests. When new requests come, their arrival time and transmission order will be recorded. Other information such as inter-arrival time can be obtained as well. These extracted features represented by a vector are then delivered to the predictor. The recorder is also responsible for communication with the server, by uploading the original request sets with feature vectors periodically.

Predictor. Once the global training is completed, each client synchronizes with the server and updates its local predictor, which will make transmission decision for each new request immediately. In prediction, the request vectors are the inputs and the delay parameters are the outputs. The predictor takes the feature vector generated from each coming request as input, and produces a numerical output, indicating the delay for that request. Note that the training phase is offline, while the prediction is made online in a timely manner, e.g., about 10^{-5}s.

Scheduler. The scheduler makes the decision whether transmitting the request immediately or deferring for a while according to the predicted delay parameter. For the request with a delay parameter greater than 0, the scheduler delivers request to the transmission interface. Otherwise, the scheduler put the request into the buffer.

Buffer. The buffer pool is responsible for caching the delayed requests. Each request in the buffer pool has a deadline. The system scans all buffered requests during each cycle, and the request whose deadline comes will be transmitted.

Interface. The interface module maintains the channel between the upper applications and 3G/4G NIC and guarantees the transmission order.

B. Server design

Training module. To relieve the computing overhead from clients, the training phase is conducted at the server. We apply a participatory sensing scheme that each client just uploads small chunks of transmission information to the server at a time. After collecting such data from multiple users, the global classifier will be trained by a supervised learning method. The basic idea is that an offline algorithm first calculates the expected delay based on the global knowledge, and then the system takes the feature vectors and the corresponding calculated delays as the inputs to train the predictor.

Global predictor. The predictor is an artificial neural network (ANN) with processing units connected by communication channels that have multiplicative weights. An input signal is multiplied by a weight and added onto other signals, the total sum of which is then applied to a generally non-linear activation function to calculate the output of each neuron. Once the training phase is done, the new predictor will be applied to the mobile devices.

V. Algorithms

We consider a sequence of communication requests, each denoted by $r_i\langle a_i, d_i\rangle$ with an arrival time slot a_i and deadline d_i. Our objective is to schedule these requests such that the tail time of 3G/4G interface can be significantly reduced. As discussed in Section 4, we use the solution of an offline scheduling algorithm to train a delay predictor that will be adopted by the online scheduling algorithm to make transmission decisions such that it will achieve a similar performance as the offline one. The involving algorithms are presented in this section below.

A. Offline scheduling

For supervised learning, a desired scheduling should be obtained as training input. Using a participatory sensing approach, we propose an offline scheduling with the knowledge of whole transmissions collected. To reduce the total tail time, it updates the scheduling for a given sequence of n requests $R = \{r_i\}$ iteratively. The description is given in Algorithm 1 as follows.

Initially, a feasible scheduling is generated randomly with a total tail time T calculated by function `SUM_TAILTIME()` using Equation (2). Then we use the descent method to find an updated scheduling with a reduced total tail time. The update will be iterated until no further improvement possible.

Algorithm 1 The offline scheduling algorithm based on descent method

1: **for** $i = 1$ to n **do**
2: set the transmission time of r_i at a random time slot s_i with $a_i \leq s_i \leq d_i$ and $s_i \neq s_j, \forall r_i \neq r_j \in R$;
3: **end for**
4: $S = \{s_1, s_2, .., s_n\}$;
5: $T =$ `SUM_TAILTIME`(S);
6: $\Delta T = \infty$;
7: **while** $\Delta T > 0$ **do**
8: **for** $i = 1$ to n **do**
9: **for** $k = a_i$ to d_i **do**
10: set a new transmission time slot $k \neq s_i$ for r_i;
11: $S_{ik} = \{s_1, s_2, .., s_i = k, ..., s_m\}$;
12: $\Delta T_{ik} = T -$ `SUM_TAILTIME`(S_{ik});
13: **end for**
14: **end for**
15: $\Delta T = \max_{(i,k)} \Delta T_{ik}$;
16: $S = S_{ik}$, where $(i, k) = \arg(\max_{(i,k)} \Delta T_{ik})$;
17: **end while**
18: **return** S

B. ANN-based training

We first need to determine the features of a given request sequence to be trained. A straightforward feature is the inter-arrival time, i.e., the time between two consecutive requests, which is also adopted in [2]. More information describing the transmission pattern can be exploited by historical data. In our study, we consider a number of candidate features denoted as $\mathbf{f}_i$ for each request r_i:

$$\mathbf{f}_i = \langle i, \Delta a_i, A_i, A_i^m, B_i, T_i \rangle, \tag{3}$$

where the features are explained as follows.

- Feature i is the sequence number.
- Feature Δa_i is the time interval between current request r_i and its predecessor r_{i-1}, i.e.,

$$\Delta a_i = a_i - a_{i-1}. \tag{4}$$

- Feature A_i denotes the average of time interval up to request r_i, i.e.,

$$A_i = \frac{1}{i} \sum_{j=0}^{i} \Delta a_j. \tag{5}$$

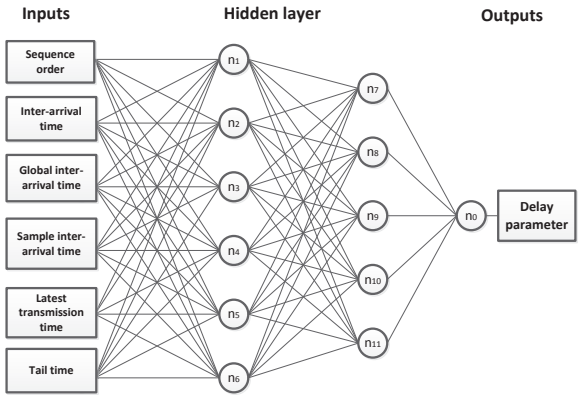


Fig. 3. The structure of MLP.

- Feature A_i^m is the average inter-arrival time of latest m requests, i.e.,

$$A_i^m = \frac{1}{m} \sum_{j=i-m+1}^{i} \Delta a_j, \ m \leq i. \tag{6}$$

- Feature B_i is the tail time of request r_i.
- Feature T_i is the total tail time till a_i.

Our objectives is to estimate a delay parameter φ, as close to the offline solution as possible, based on the given request feature vector $\mathbf{f}$. In the context of machine learning, it is equivalent to creating a function that maps the feature vector $\mathbf{f}$ to the delay parameter φ. Such relationship is usually complex and cannot be characterized by a simple linear function. In the following, we develop an ANN model to build the predictor.

Given a requests set $R = \{r_0, r_1, ..., r_n\}$, we list its feature vector set $\{\mathbf{f_0}, \mathbf{f_1}, ..., \mathbf{f_n}\}$ and the corresponding delay parameter set $\{\varphi_0, \varphi_1, ..., \varphi_n\}$ that is obtained by Algorithm 1. A supervised learning method is then used in the training phase with training set $\{\{\mathbf{f_0}, \varphi_0\}, \{\mathbf{f_1}, \varphi_1\}, ..., \{\mathbf{f_n}, \varphi_n\}\}$.

The training is conducted by multi-layer perception (MLP), consisting of three layers: inputs layer, hidden layer and outputs layer, and a number of nodes at each layer. As shown in Fig. 3, the inputs layer with six nodes represents the feature vector $\mathbf{f}$ in 6-tuple, and only one output node represents the delay parameter φ. It is generally believed that increasing hidden layers can reduce network error, while the network will become complex and prone to be over trained. A proper setting of hidden layers is usually accomplished by a trial-and-error approach, which will be studied in the experiment. During the supervised learning, instances of training pairs are presented to the neural network that will adapt its internal weighs to approximate the desired mapping.

C. Online scheduling

Once the predictor is developed, it accepts the request feature vector as input and estimates the corresponding φ immediately without collecting the knowledge of prospective requests.

The main idea is to transmit request r_i if its predicted deadline is reached. In other words, it will be transmitted

Feature vector						Delay parameter
Sequence number	Inter-arrival time	Global Inter-arrival time	Sample inter-arrival time	Latest transmission time	Tail time	
5	4174	1183	1573	4174	4174	5392
171	4803	1066	1949	4803	78211	591
333	105	954	1623	3077	116120	0

Fig. 4. An example of data uploaded from client to server.

Node	Threshold	<Connected node,Weight>
node_1	0.288	<input_1=-0.274>,<input_2=1.610>,<input_3=0.566>, <input_4=0.569>,<input_5=0.253>,<input_6=1.002>
node_7	-1.778	<node_1=-0.172>,<node_2=-0.234>,<node_3=-4.848>, <node_4=-0.642>,<node_5=-1.999>,<node_6,-1.365>
node_0	0.111	<node_7=2.728>,<node_8=2.731>,<node_9=2.717>, <node_10=2.722>,<node_11=2.716>

Fig. 5. An example of the classifier description.

upon arrival if parameter $\varphi_i = 0$ is obtained. Otherwise, the transmission will be postponed for a period time up to φ_i. In particular, if the deadline of a request with a positive predicted delay comes, this request and the remaining ones in the buffer will be all transmitted together.

D. Communication

The communication between clients and server is asynchronous and UDP-based. On the client side, the sequence of raw requests will be translated into feature vectors and sent to the server. As shown in Fig. 4, the data uploaded is a set of 7-tuple vectors which include both feature vectors $\mathbf{f}$ and delay parameters φ. The feature vector is defined in (3) and the delay parameter is attained via the offline scheduling algorithm.

Based on the participatory data from multiple users, an MLP will be trained based on supervised learning. The example given in Fig. 3 shows an MLP with two hidden layers, in which the feature vectors are the inputs and the corresponding delay parameters are the expected outputs. The six features, each represented by a block, connect to nodes from n_1 to n_6 in the first hidden layer, which then connect to nodes from n_7 to n_{11} in the second hidden layer similarly. Finally, all nodes in the second hidden layer connect to the output node n_0 which links to the expected outputs. Each node takes the outputs with different connection weights from previous layer nodes as inputs and generates the output via activation function with threshold. So the MLP can be expressed as a set of nodes with different connection weights and threshold, as shown in Fig. 5. Once the training phase is done, the server sends the MLP description data to the participatory devices. Based on the description, a local classifier can be built on the client side. The communication overhead is negligible because the size of description data if small.

VI. Experiments

A. Experiment environment

We first investigate the prediction performance using multi-layer perceptron. The proposed algorithms are implemented by using WEKA $3.6.10$ in a 32-bit computer with a core i5 processor (clock speed of 2.3 GHz) and 4GB RAM.

We also implement our online scheduling on *SAMSUNG Galaxy Wonder* smartphones installed with operating system of *Android 2.3.3*, which supports APIs showing the operating voltage and remaining battery capacity. We then evaluate the energy consumption of smartphones with different scheduling methods in different scenarios. To measure the energy consumption, we use external 3.8V DC power supply. To lower the communication overhead, the size of requests uploaded from each client to the server is limited to $n = 200$.

In our experiments, all request data are generated by mobile devices running different applications, e.g., web browser and Sina Weibo, which can tolerate a small user-specified delays [2].

TABLE I
Prediction performance under various MLP structrures

Structure	Correlation coefficient	RMSE (ms)	Time taken (s)
3	0.6869	1321.36	0.18
4	0.8011	1215.8316	0.22
5	0.8051	1228.68	0.25
6	0.8378	1087.1189	0.28
$6,1$	0.6996	1394.7887	0.3
$6,2$	0.8521	1047.5422	0.42
$6,3$	0.8451	1182.2626	0.65
$6,4$	0.8344	1263.409	0.69
$6,5$	0.8576	911.851	0.67
$6,6$	0.8109	919.6565	0.68

B. MLP construction

For a given training set with n instances (or requests), let $\widehat{\varphi}$ and φ be the delay of r_i predicted by our ANN model and calculated by the offline Algorithm 1, respectively. An evaluation criterion is called the root mean squared error (RMSE) defined as:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n} \left(\widehat{\varphi}_i - \varphi_i\right)^2}{n}}. \tag{7}$$

The lower value of RMSE, the more accurate the prediction is. Another evaluation criterion is the correlation coefficient (CC) which depicts the correlation between the delay parameter and the feature vector.

We first need to determine the structure of trained multi-layer perceptron. A list of candidate structures are shown in Table I, where the numbers indicate the number of nodes in each hidden layer. For example, "3" represents three nodes in a single hidden layer and "$6,3$" represents a multi-layer network with two hidden layers: 6 nodes in the first layer and 4 nodes in the second layer. To achieve a good accuracy-complexity tradeoff and to avoid the over-fitting problem, a network with one or two hidden layers is preferred. Applying the trial-and-error approach, we will adopt structure "$6,3$" because it has minimum RMSE and maximum CC as shown in Table I. The corresponding MLP structure is illustrated in Fig. 3.

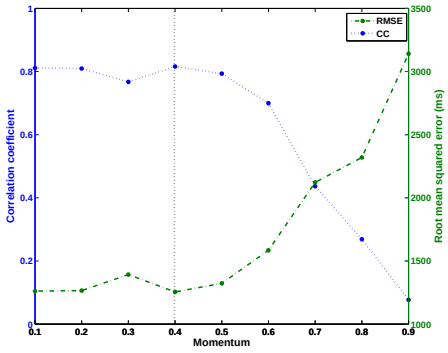


Fig. 6. Prediction accuracy under various settings of momentum.

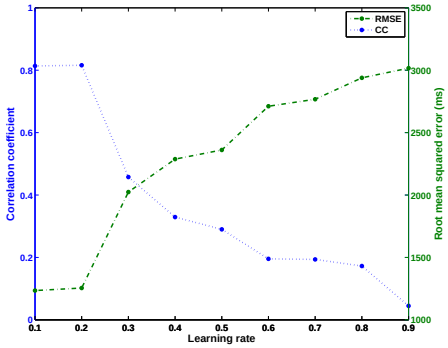


Fig. 7. Prediction accuracy under various settings of learning rate.

We then investigate the influence of other parameter settings in the training phase. There are two parameters with great impact on the prediction: learning rate and momentum, which represent the number of weight updates in each training and the momentum when weight updates, respectively. We test the performance under various combinations of these parameters and find that the system achieves the highest prediction accuracy at learning rate of 0.1 and momentum of 0.6. For example, under a fixed learning rate of 0.1, the network has the lowest RMSE and highest correlation coefficient when the value of momentum is equal to 0.4 as shown in Fig. 6. Under such momentum, the network has the lowest RMSE and highest correlation coefficient when the value of learning rate is equal to 0.1 as shown in Fig. 7.

TABLE II
FEATURE VECTOR

Label	Feature vector
1	$\langle i, \Delta a_i \rangle$
2	$\langle i, \Delta a_i, A_i \rangle$
3	$\langle i, \Delta a_i, A_i, A_i^m \rangle$
4	$\langle i, \Delta a_i, A_i, A_i^m, B_i \rangle$
5	$\langle i, \Delta a_i, A_i, A_i^m, B_i, T_i \rangle$

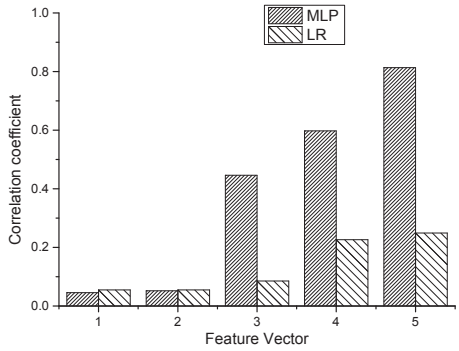


Fig. 8. Correlation coefficient under various feature combinations.

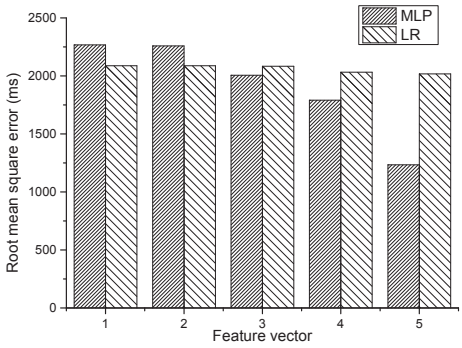


Fig. 9. Root mean square error under various feature combinations.

C. Performance of prediction

The performance of MLP will be evaluated by comparing to the linear regression (LR) method in predicting the delay parameters for the given request feature vectors.

As discussed in Section 5.2, each request r_i is expressed as a feature vector $\mathbf{f}_i$ with a number of candidate features. We first investigate the performance of prediction under various combinations of those features as shown in Table II. The corresponding performance in terms of CC and RMSE are shown in Fig. 8 and Fig. 9, respectively. It is illustrated that including more features leads to higher CC and lower RMSE. In particular, adding A_i^m significantly improves CC as shown in Fig. 8, where the feature vectors including A_i^m or not are labeled as 3-5 and 1-2, respectively.

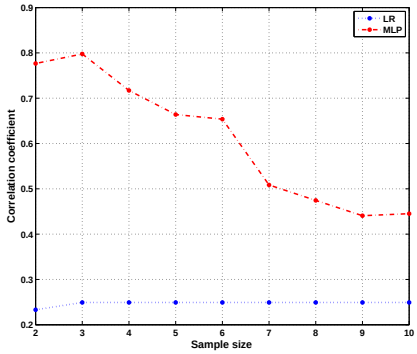


Fig. 10. Correlation coefficient vs m.

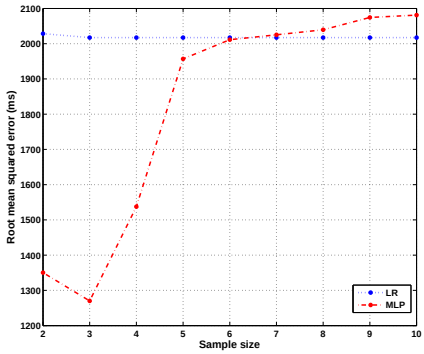


Fig. 11. Root mean square error vs m.

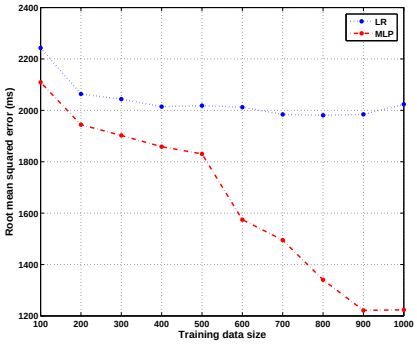


Fig. 13. Root mean square error under various training data sizes.

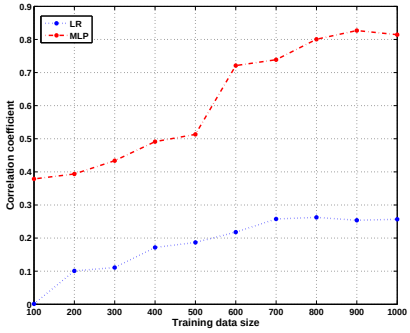


Fig. 12. Correlation coefficient under various training data sizes.

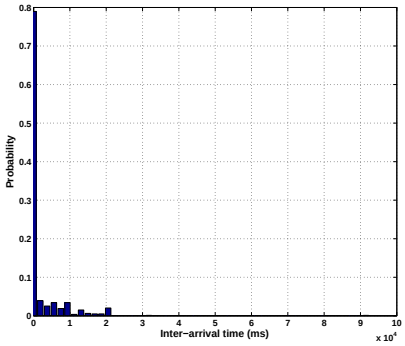


Fig. 14. Probability distribution of request inter-arrival time in web browsing.

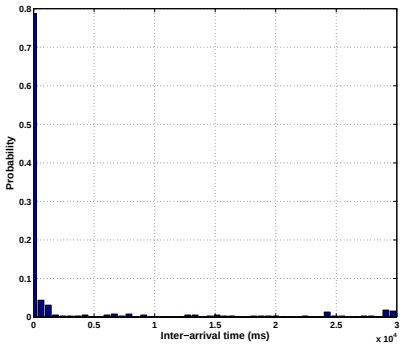


Fig. 15. Probability distribution of request inter-arrival time in Sina Weibo.

Recall that A_i^m reflects the average inter-arrival time of m most recent arrivals. When $m = 1$, A_i^m turns to Δa_i, and when $m = n$, A_i^m is equal to A_i. We show the performance of prediction by varying m from 2 to 10 in Fig. 10 and Fig. 11. The experimental results show that both maximum correlation coefficient and minimum RMSE are achieved when $m = 3$.

To validate our participatory sensing approach, we also show the prediction performance as a function of the training data size in Fig. 12 and Fig. 13 . The results clearly reveal that the performance can be significantly improved with the increasing number of request logs as the training inputs.

D. Performance of ML-based online scheduling

In the final set of experiment, we investigate the performance of a number of scheduling algorithms:

1) our proposed machine learning based online algorithm (ML-online),
2) the online algorithm that makes transmissions immediately upon arrivals (Default),
3) the online algorithm proposed in [2] that makes transmission decisions based on the inter-arrival time between current request and the prior request (TailEnder), and
4) the offline Algorithm 1 (OPT-offline).

We conduct the experiments for two applications: chrome browser (loading 10 web page), and *Sina Weibo* (getting 100 news). After converting these application traces into request sequences, we show the probability distribution of their inter-arrival times in Fig. 14 and Fig. 15, respectively. It can be concluded that, for both requests in web browsing and social application, the request inter-arrival time obeys the index distribution. In particular, the requests of *Weibo* show a much broader distribution than requests of web browsing. In each experiment, we run a transmission sequence of about 300 requests, generated by chrome and *Sina Weibo*.

The energy consumption as a function of maximum delay allowed for each transmission is illustrated in Fig. 16 and Fig. 17 for these two applications. The Default scheduling algorithm shows a horizontal line since the transmissions are always made immediately no matter how the delay is set. The offline algorithm always achieves the best performance because it makes use of the knowledge of whole transmission

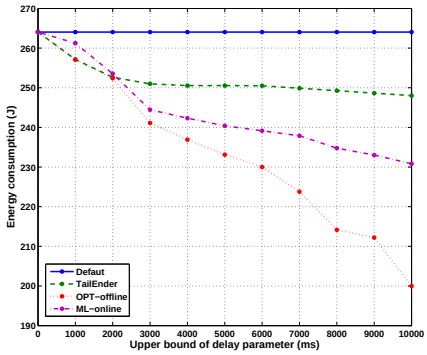


Fig. 16. Energy consumption in web browsing.

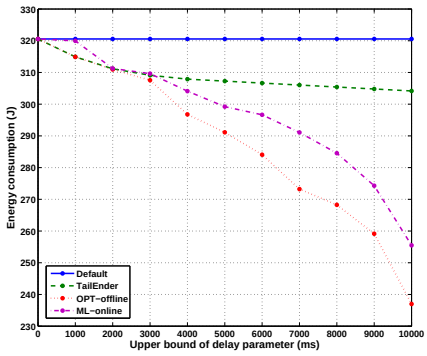


Fig. 17. Energy consumption in Sina Weibo.

requests. Better performance will be obtained if a longer delay is allowed. This is because longer delay promotes the probability of postponed transmissions with shortened total tail time. The TailEender algorithm improves the performance of Default only slightly by exploiting limited historical information. In contrast, our proposed ML-online can save significant energy compared to other online algorithms, especially when the delay requirement is large. For example, as shown in Fig. 17, ML-online saves energy by 20% and 17% over Default and TailEnder, respectively, when the maximum delay is 10 seconds.

The experimental results also validate our ML-based approach by showing its similar performance to the offline algorithm. For example, when a maximum 5 second delay is allowed, the ML-online algorithm consumes only 3% additional energy compared to the OPT-offline algorithm for both applications.

E. System overhead

To relieve the computing overhead from clients, the computation-intensive training process is performed on the server. To lower the participatory communication overhead, the size of requests uploaded from each client to the server is limited. In this section we evaluate the communication overhead by conducting experiments in three scenarios: web browsing, *Weibo* and hybrid, each with 30 minutes.

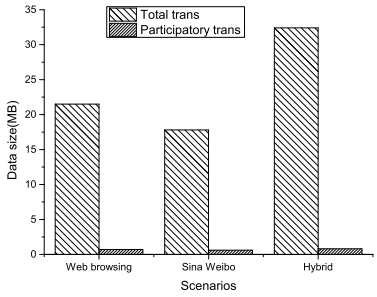


Fig. 18. The data size of total transmission and participatory transmission.

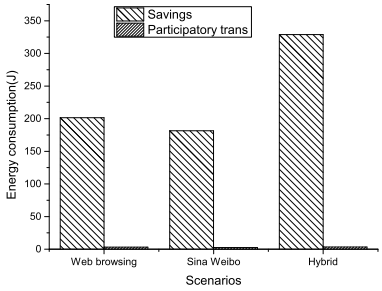


Fig. 19. The energy of savings and participatory transmission.

Fig. 18 shows the data size of total transmission and participatory transmission. For a round of participatory communication, the uploading data is less than 20KB and the classifier description data is less than 5KB. We also compare the energy savings by our approach and energy consumption of participatory communications. The average energy savings are 201.5J, 181J and 328.4J in three scenarios. However, the average energy consumptions of participatory communications, i.e., uploading and classifier updating, are only 3.1J, 2.6J and 3.5J, respectively. The results indicate that the overhead is negligible compared with the energy savings.

VII. Conclusion

Energy consumption is a critical problem in mobile device and the *tail energy* of 3G/4G transmission takes up a large percentage. In this paper, we investigate the tail energy reduction by exploiting transmission request features.

We propose a machine learning based online scheduling that reduces the tail energy by deferring transmissions. To validate our approach, we propose a client-server architecture and implement it on a real system. The experiments conducted on smartphones show that our method can effectively improve the energy efficiency of mobile devices. Our online scheduling using ML saves energy over the traditional online methods and runs similarly to the offline algorithm.

Acknowledgement

The work was mainly supported by Strategic Information and Communications R&D Promotion Programme (SCOPE No. 121802001) from the Ministry of Internal Affairs, Japan.

This paper was also partially supported by China National Natural Science Foundation under grant No. 61272408, 61322210, National High-tech Research and Development Program of China (863 Program) under grant No.2012AA010905, and Doctoral Fund of Ministry of Education of China under grant No. 20130142110048.

REFERENCES

[1] Haverinen H, Siren J, Eronen P, "Energy consumption of always-on applications in WCDMA networks," *Proc. IEEE Vehicular Technology Conference (VTC2007-Spring)*, pp. 964-968, 2007.

[2] Balasubramanian N, Balasubramanian A, Venkataramani A, "Energy consumption in mobile phones: a measurement study and implications for network applications," *Proc. ACM SIGCOMM conference on Internet measurement conference*, pp. 280-293, 2009.

[3] Sharma A, Navda V, Ramjee R, et al, "Cool-Tether: energy efficient on-the-fly wifi hot-spots using mobile phones," *Proc. ACM Emerging networking experiments and technologies*, pp. 109-120, 2009.

[4] Third generation partnership project (3gpp), http://www.3gpp.org.

[5] Lee C C, Yeh J H, Chen J C, "Impact of inactivity timer on energy consumption in WCDMA and cdma2000," *IEEE Wireless Telecommunications Symposium*, pp.15-24, 2004.

[6] Third generation partnership project2 (3gpp2), http://www.3gpp2.org.

[7] Yeh J H, Chen J C, Lee C C, "Comparative analysis of energy-saving techniques in 3GPP and 3GPP2 systems," *IEEE Trans. Vehicular Technology*, vol. 58, no. 1, pp. 432-448, 2009.

[8] Liu H, Zhang Y, Zhou Y, "Tailtheft: leveraging the wasted time for saving energy in cellular communications," *Proc. ACM 6th international workshop on MobiArch*, pp. 31-36, 2011.

[9] Chuah M, Luo W, Zhang X, "Impacts of inactivity timer values on UMTS system capacity," *Proc. IEEE Wireless Communications and Networking Conference (WCNC2002)*, vol. 2, pp. 897-903, 2002.

[10] Xiao Y, Kalyanaraman R S, Yla-Jaaski A, "Energy consumption of mobile youtube: Quantitative measurement and analysis," *Proc. IEEE Next Generation Mobile Applications, Services and Technologies (NGMAST'08)*, pp. 61-69, 2008.

[11] Nurminen J K, Noyranen J, "Energy-consumption in mobile peer-to-peer-quantitative results from file sharing," *Proc. IEEE Consumer Communications and Networking Conference (CCNC2008)*, pp. 729-733, 2008.

[12] Thiagarajan N, Aggarwal G, Nicoara A, et al, "Who killed my battery?: analyzing mobile browser energy consumption," *Proc. ACM World Wide Web*, pp 41-50, 2012.

[13] Falaki H, Lymberopoulos D, Mahajan R, et al, "A first look at traffic on smartphones," *Proc. ACM SIGCOMM conference on Internet measurement*, pp. 281-287, 2010.

[14] Qian F, Wang Z, Gerber A, et al, "Characterizing radio resource allocation for 3G networks," *Proc. ACM SIGCOMM conference on Internet measurement*, pp. 137-150, 2010.

[15] Witten I H, Frank E, *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2005.

[16] Frank J, "Artificial intelligence and intrusion detection: Current and future directions," *Proc. 17th national computer security conference*, vol. 10, pp. 1-12, 1994.

[17] McGregor A, Hall M, Lorier P, et al, "Flow clustering using machine learning techniques," *Passive and Active Network Measurement, Springer*, pp. 205-214, 2004.

[18] Bernaille L, Teixeira R, Akodkenou I, et al, "Traffic classification on the fly," *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 2, pp. 23-26, 2006.

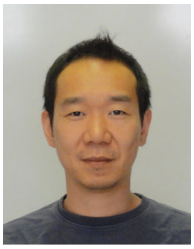
[19] Moore A W, Zuev D, "Internet traffic classification using bayesian analysis techniques," *ACM SIGMETRICS Performance Evaluation Review*, vol. 33, no. 1, pp. 50-60, 2005.

[20] Geurts P, El Khayat I, Leduc G, "A machine learning approach to improve congestion control over wireless computer networks," *Proc. IEEE International Conference on Data Mining (ICDM'04)*, pp. 383-386, 2004.

[21] Nguyen T T T, Armitage G, "A survey of techniques for internet traffic classification using machine learning," *IEEE Trans. Communications Surveys & Tutorials*, vol. 10, no. 4, pp. 56-76, 2008.

[22] J. Huang, F. Qian, A. Gerber, Z. M. Mao, S. Sen, and O. Spatscheck,A close examination of performance and power characteristics of 4G LTE networks, in *Proc. ACM. Mobisys*, 2012, pp. 225C238.

Zaiyang Tang received his BS degree from Huazhong University of Science and Technology, China, in 2011. He is a PhD candidate in Huazhong University of Science and Technology. His research interests include networking modeling, network virtualization, machine learning and recommendation system.



Song Guo (M'02-SM'11) received the PhD degree in computer science from University of Ottawa, Canada. He is currently a Full Professor at School of Computer Science and Engineering, the University of Aizu, Japan. His research interests are mainly in the areas of protocol design and performance analysis for computer and telecommunication networks. He received the Best Paper Awards at ACM IMCOM 2014, IEEE CSE 2011, and IEEE HPCC 2008. Dr. Guo currently serves as Associate Editor of IEEE Transactions on Parallel and Distributed Systems and IEEE Transactions on Emerging Topics in Computing. He is in the editorial boards of ACM/Springer Wireless Networks, Wireless Communications and Mobile Computing, and many others. He has also been in organizing and technical committees of numerous international conferences, including serving as a General Co-Chair of MobiQuitous 2013. Dr. Guo is a senior member of the IEEE and the ACM.

Peng Li received his BS degree from Huazhong University of Science and Technology, China, in 2007, the MS and PhD degrees from the University of Aizu, Japan, in 2009 and 2012, respectively. He is currently an Associate Professor at School of Computer Science and Engineering, the University of Aizu, Japan. His research interests include networking modeling, cross-layer optimization, wireless sensor networks, cloud computing, smart grid, performance evaluation of wireless and mobile networks for reliable, energy-efficient, and cost-effective communications. He is a member of the IEEE.

Toshiaki Miyazaki is a professor of the University of Aizu, Fukushima, Japan, and Dean of the Undergraduate School of Computer Science and Engineering. His research interests are in reconfigurable hardware systems, adaptive networking technologies, and autonomous systems. He received the B.E. and M.E. degrees in applied electronic engineering from the University of Electro-Communications, Tokyo, Japan in 1981 and 1983, and Ph.D. degree in electronic engineering from Tokyo Institute of Technology in 1994. Before joining the University of Aizu, he has been worked for NTT for 22 years, and engaged in research on VLSI CAD systems, telecommunications-oriented FPGAs and their applications, active networks, peer-to-peer communications, and ubiquitous network environments. Dr. Miyazaki was a visiting professor of the graduate school, Niigata University in 2004, and a part-time lecturer of the Tokyo University of Agriculture and Technology in 2003-2007. He is a senior member of IEEE, and a member of IEICE and IPSJ.

Hai Jin is a Cheung Kung Scholars Chair Professor of computer science and engineering at the Huazhong University of Science and Technology (HUST) in China. He is now Dean of the School of Computer Science and Technology at HUST. Jin received his PhD in computer engineering from HUST in 1994. In 1996, he was awarded a German Academic Exchange Service fellowship to visit the Technical University of Chemnitz in Germany. Jin worked at The University of Hong Kong between 1998 and 2000, and as a visiting scholar at the University of Southern California between 1999 and 2000. He was awarded Excellent Youth Award from the National Science Foundation of China in 2001. Jin is the chief scientist of ChinaGrid, the largest grid computing project in China, and the chief scientist of National 973 Basic Research Program Project of Virtualization Technology of Computing System. Jin is a senior member of the IEEE and a member of the ACM. Jin is the member of Grid Forum Steering Group (GFSG). He has co-authored 15 books and published over 400 research papers. His research interests include computer architecture, virtualization technology, cluster computing and grid computing, peer-to-peer computing, network storage, and network security. Jin is the steering committee chair of International Conference on Grid and Pervasive Computing (GPC), Asia-Pacific Services Computing Conference (APSCC), International Conference on Frontier of Computer Science and Technology (FCST), and Annual ChinaGrid Conference. Jin is a member of the steering committee of the IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid), the IFIP International Conference on Network and Parallel Computing (NPC), and the International Conference on Grid and Cooperative Computing (GCC), International Conference on Autonomic and Trusted Computing (ATC), International Conference on Ubiquitous Intelligence and Computing (UIC).

Xiaofei Liao received his Ph.D. degree in computer science and engineering from Huazhong University of Science and Technology (HUST), China, in 2005. He is now a professor in the school of Computer Science and Engineering at HUST. He has served as a reviewer for many conferences and journal papers. His research interests are in the areas of system software, P2P system, cluster computing and streaming services. He is a member of the IEEE and the IEEE Computer society.