Q-offload: Quality Aware WiFi Offloading with Link Dynamics

Yi Zhang*†, Jiliang Wang †, Yuan He †, Yanrong Kang *, Bo Li * and Yunhao Liu †
*Department of Computer Science and Engineering, Hong Kong University of Science and Technology
†School of Software and TNLIST, Tsinghua University
Email: {yzhangbh,ykangaa,bli}@ust.hk, {jiliang,he,yunhao}@greenorbs.com

***Abstract*—Driven by the proliferation of mobile applications, the conflict between data communication requirement and limited battery capacity is becoming sharp on modern smartphones. Offloading mobile traffic from cellular to WiFi is widely recognized as a viable solution to improve the energy efficiency. However, through extensive field experiments, we find WiFi offloading is not always energy efficient and even consumes more energy than cellular network due to link quality variation. In addition, we also observe that practical data transmission deadline requirement and link utilization allows scheduling of data traffic to time periods with good link quality. Accordingly, we propose Q-offload, the first attempt towards energy efficient WiFi offloading with link dynamics. In Q-offload, we propose an iterative framework to achieve energy efficient WiFi offloading by exploiting good link quality while not affecting user experience. We evaluate the performance of Q-offload through both trace-driven analysis and real-world experiments. The results show that it can achieve 33.5%∼55.7% energy efficiency improvement, compared with state-of-the-arts under different conditions.**

I. Introduction

Driven by the proliferation of mobile applications, mobile communication is playing an important role in our daily life. Reported in [1], global mobile data traffic in Q1 2014 has exceeded 2400 PB. As a typical and commonly-used network type, however, modern cellular network often falls short in providing energy efficient network access due to its relatively high power consumption [2].

On the other hand, WiFi [2], which is becoming a common network access method in our daily life, provides more energy efficient network access than typical modern cellular networks. Therefore, offloading mobile data through available WiFi [3]–[6] has been adopted as a viable solution in saving energy cost of data transmission. By exploiting the opportunistic WiFi connectivity in mobile environment, the energy consumption in data transmission can be decreased. Actually, most modern cellular phones have adopted such a mechanism as the default setting in which they switch from cellular network to WiFi when WiFi is available. Meanwhile, various approaches are proposed to improve the offloading efficiency, e.g., prefetching [7], delayed offloading [4], [8] and collaborative offloading [3], [9].

Most existing works on WiFi offloading, however, assumes that WiFi is always energy efficient and neglect an important fact that the energy efficiency of WiFi is dynamic, especially under mobile environment. This is because WiFi link quality, which directly affects its energy efficiency [10], may vary in different environments. For a lower link quality, the transmission time for the same amount of data is longer than that for a higher link quality. Thus, the energy consumption will be higher for the lower link quality. By quantitative tests in a $2000m^2$ office building with multiple APs as shown in Figure 1, we find that in 45.6% download cases and 20.1% upload cases, the energy efficiency of WiFi is even worse than 4G HSPA+ on the same platform. Under those cases, WiFi offloading brings no energy benefit and even increases the energy consumption. This tells us the WiFi link quality, which is previously overlooked, should be carefully considered. Accordingly, the data traffic for offloading should be scheduled according to WiFi link dynamics. Driven by the observation, we further find that for most data transfer, the real-time requirement is not tight, which spares adequate space for us to reschedule the transmission. This finding tells us that we can exploit links with high quality to conduct data offloading while avoiding using links with low quality, so that the overall energy efficiency of WiFi offloading could be greatly improved.

Under link dynamics, designing a scheme for energy efficient WiFi offloading is challenging. The main reason is that link quality is dynamic. This feature makes it hard to determine a global optimal offloading scheme. As a result, the effectiveness of a scheme is often restricted in small time scales. In addition, the real-time requirement of data transmission also suffers a risk of being violated under this condition. To address those challenges, in this paper, we propose Q-offload, a quality aware offloading scheme to make the offloading decision towards optimal energy efficiency without affecting user experience. First, we propose a general iterative link quality prediction method to continuously update the offloading decision by exploiting available good links for data offloading. Thus the energy consumption can be significantly reduced, especially with link dynamics. Second, we also design a deadline guarantee mechanism to estimate the data finish time in order to satisfy the transmission deadline. Accordingly, user experience will not be affected in Q-offload. We have implemented Q-offload on the Android platform and evaluated its performance in both trace-driven analysis and real-world experiments. To summarize, our contributions are as follows:

- We introduce the idea of quality aware WiFi offloading through empirical quantification on the impact of

1052-8725/15 $31.00 © 2015 IEEE
DOI 10.1109/RTSS.2015.30

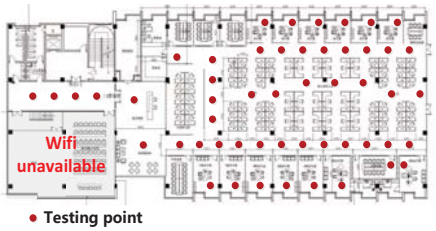


Figure 1: Experimental floorplan

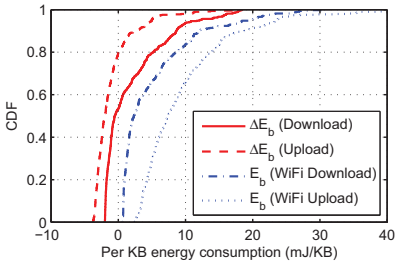


Figure 2: E_b analysis.

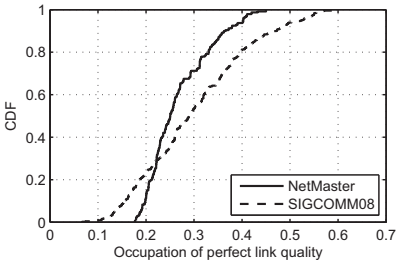


Figure 3: Occupation of slots with good link quality, i.e. top 20%.

WiFi link dynamics. We demonstrate that the energy efficiency of WiFi offloading can be greatly improved by exploiting feasible good link quality.

- We present the analytical formulation of the problem and demonstrate its hardness. Following the analysis, we propose the design of Q-offload, the first attempt towards energy efficient WiFi offloading with link dynamics. By leveraging different strategies to make real-time offloading decisions on the link dynamics, Q-offload can achieve energy efficient WiFi offloading while also preserving user experience.
- We conduct extensive evaluation of Q-offload in both trace-driven analysis and real-world experiments. The results show that Q-offload can achieve 33.5%∼55.7% energy efficiency improvement, compared with state-of-the-arts under different conditions.

The rest of this paper is organized as follows: Section II presents in detail the background and our motivation. Section III presents the formulation of the problem. Section IV introduces the design details of Q-offload. Section V presents the implementation, followed by the performance evaluation in Section VI. Section VII presents the related work and we conclude this work in Section VIII.

II. Background and Motivation

A. Background

To handle the explosive and ubiquitous data requirement, cellular networks, e.g., LTE, HSPA+, are widely used nowadays, which provide fast network access with up to 100Mbps bandwidth [11]. However, cellular networks also introduce a relatively high energy consumption [2], [10], [12]. As measured in [2], the average power consumption of LTE is around 1600mW (Download) and 2000mW (Upload). Considering the limited battery life on smartphones, e.g., around 3000mAh, accessing the Internet only through cellular networks would severely narrow down the usage time, and thus affect the quality of user experience.

To save the energy consumed in data communication, offloading data traffic through WiFi is recognized as a more energy efficient approach to reduce communication overhead [3]–[5], [13]. For example, the default setting of modern smartphones is offloading traffic to WiFi when it is available.

B. Inefficiency of WiFi offloading

However, offloading traffic through WiFi is not always energy efficient. The main reason is that WiFi link quality varies in different environments. When link quality is poor, it requires more time to transmit the same amount of data [10], [14]. Accordingly, the energy consumption in data transmission increases. Therefore, WiFi offloading, which is traditionally presumed to be energy efficient, may not be beneficial in poor link quality.

To quantify the impact of link quality variation on the energy efficiency of WiFi offloading, we conduct field tests in a $2000m^2$ office building with multiple APs. The floor plan is shown in Fig.1. We use *average energy consumption per KB* E_b to estimate the energy efficiency as in [2]. E_b can be calculated as:

$$E_b = \frac{\beta}{b} + \alpha \tag{1}$$

in which b is the throughput and α, β are two power parameters. The power parameters of each network type is measured by FLUKE 120 scopemeter. We use Sumsang Galaxy Note3 as the testing platform and examine E_b of WiFi at 52 different locations. The WiFi type is 802.11n and we choose 4G HSPA+ as the benchmark. At each testing point, we send/receive TCP packets to/from a remote server located in Shanghai for 10 minutes.

We measure $E_b(WiFi)$ for the average energy consumption per KB of WiFi and $E_b(HSPA+)$ for the average energy consumption per KB of cellular network. We plot the CDF of $\Delta E_b = E_b(WiFi) - E_b(HSPA+)$ and $E_b(WiFi)$ for each position in Fig.2. First, we observe that the link quality variation, which is estimated by $E_b(WiFi)$, is significant. For transmitting the same amount of data, the E_b has a variation up to 29.8mJ/KB for download and 37.6mJ/KB for upload. Moreover, if we perform best-effort WiFi offloading as in most of existing works [3], [5], 45.6% downloaded traffic and 20.1% uploaded traffic through WiFi cost even more energy than HSPA+. On the other hand, we also notice that if we can reschedule the data transmission from poor link quality to good link quality, e.g., 1MB data from E_b(WiFi download) $= 30.43mJ/KB$ to E_b(WiFi download) $= 0.63mJ/KB$, about 29.8J energy could be saved, which is significant for the limited battery

capacity. Therefore, this finding motivates us by exploring good link quality, the energy efficiency of mobile data offloading through WiFi can be significantly improved.

C. Link quality aware offloading scheme: feasibility and challenges

In order to schedule traffic for energy efficient WiFi offloading, we should answer two questions: (1) whether the real-time requirement of mobile applications allows such scheduling, and (2) whether those good links can accommodate the scheduled network requests?

For the first question, many measurement studies have been done on analyzing the deadlines of different mobile applications. For example, the studies in [15], [16] suggest more than 50% of the interviewed users would wait up to 10 minutes to stream YouTube videos and 3-5 hours for file downloads. In addition, for delay-sensitive applications, e.g., browser, not all objects are directly displayed on the screen. Accordingly, the off-screen data can also be delayed up to tens of seconds [17]. The results indicate that for many applications, the real-time requirement is not tight, and thus we can schedule the offloading time while also guaranteeing the corresponding deadline.

To answer the second question, we conduct offline analysis on two datasets, NetMaster [18] and SIGCOMM08 [19]. The former was collected with 8 users for three weeks. The latter was collected at a conference that has a peak of 31 clients. We define the good link quality as the top 20%. We plot the cdf of occupation of network capacity in Fig.3 and find the average occupation ratios are low for both datasets, i.e. 26.73% of network capacity are used for NetMaster and 30.06% for SIGCOMM08. As a result, the analysis demonstrates the link quality aware offloading scheme is also feasible in practice.

D. Summary

In the section, we investigate the impact of link quality variation on the energy efficiency of WiFi offloading and show the inefficiency of existing WiFi offloading methods due to link dynamics. We also illustrate that the deadline requirement for typical data traffic and link utilization allows scheduling of data to reduce energy consumption while not affecting user experience. In the following sections, we will show how to schedule the traffic based on link dynamics while meeting users' deadline requirement under different conditions.

III. Problem formulation

We consider a mobile user who has data of size D to transmit within a WiFi coverage. The data can be interpreted as a game to be downloaded from Google play, a video to be watched on YouTube App, a webpage to be loaded in a browser or etc. For each type, it has a corresponding deadline, e.g., the time when the user plays the game. Without loss of generality, we denote the deadline as n and the start of the network request as 1. The time period between $[1, n]$ forms the offloading decision window $\mathcal{T}_{1,n}$. For window $\mathcal{T}_{1,n}$, we use a discrete time-slotted model to denote it, i.e. $\mathcal{T}_{1,n}$ is divided into n discrete equally-sized time slots $\mathcal{T}_{1,n} = \{1, 2, ..., n\}$. For each slot, we define its length as Δt. By default, we set $\Delta t = 1s$.

Let $b(t)$ denote the WiFi link quality experienced by the user in slot t. Without loss of generality, we specify $b(t)$ as the achievable link bandwidth (upload/download), as bandwidth is the most critical factor in the energy consumption of data transmission [2]. To model the varying pattern of $b(t)$, we denote $P_{i,j}$ as the probability that $b(t)$ transits from state i to state j [20], [21]. Assume there are M states for $b(t)$, the transition matrix is given by:

$$\mathbb{P} = \begin{bmatrix} P_{1,1} & P_{2,1} & ... & P_{M,1} \\ & \cdots & & \\ & & \cdots & \\ P_{M,1} & P_{M,1} & ... & P_{M,M} \end{bmatrix} \tag{2}$$

With link dynamics, the matrix may not be fixed, i.e., the entry $P_{i,j}$ is a time-varying variable. Other channel models, like Gilbert-Elliott (GE) channel model [20], can also be adopted in our scenario under the above settings. In addition, we also notice that within a WiFi coverage, there may exist multiple APs offering different $b(t)$. Without loss of generality, we denote $b(t)$ as the best one.

Based on current bandwidth $b(t)$ of a mobile device, the decision set a contains two operations: "choose t for transmission" and "remain idle". For simplification, we denote $a(t)$ as follows:

$$a(t) = \begin{cases} 1, & \text{choose } t \text{ for transmission} \\ 0, & \text{remain idle} \end{cases}$$

According to the offloading decision, let $Z(t)$ denote the amount of data transmitted in slot t, which can be calculated as $Z(t) = b(t) \cdot a(t) \cdot \Delta t$.

For energy consumption, we assume WiFi PSM is enabled by default and a preferable AP has already been selected. Accordingly, the idle listening, scanning and initial energy consumption are excluded in our model and the transmission energy is proportional to the transmission time. We adopt a consistent power consumption model in [22] and denote the power rate of WiFi as P_{wifi}. Then we can calculate the energy consumption of a slot $E(t)$ as follows:

$$E(t) = P_{wifi} \cdot a(t) \cdot \Delta t$$

Based on the definition, for network request D with finite window $\mathcal{T}_{1,n}$, we aim to find an offloading scheme $\mathcal{S}^* = \{a(1), ..., a(n)\}$ such that the total energy consumption is

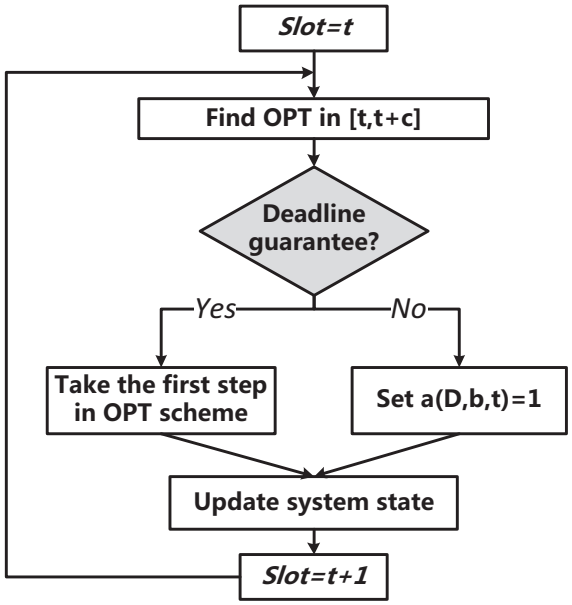


Figure 4: The workflow of Q-offload

minimized, i.e.

$$\begin{aligned} \min \quad & \sum_{i=1}^{n} a(i) \cdot E(i) \\ s.t. \quad & \sum_{i=1}^{n} Z(t) \geq D \\ & a(i) \in \{0,1\}, \quad \forall i \end{aligned} \tag{3}$$

Notice the optimization problem (Eq.3) is in fact an online knapsack problem with unknown optimal number of knapsacks [23]. Even if the optimal number is known, the problem is still NP-hard to solve and does not admit any competitive deterministic online algorithm [23]. Given this fact, we resort to Model Predictive Control (MPC) method [24] and propose Q-offload, a heuristic scheme to solve this problem in the following section.

IV. Design of Q-offload

In the section, we propose the design of Q-offload, a heuristic scheme to achieve energy efficient WiFi data offloading. Since long-term link quality is hard to be accurately predicted, the basic idea of Q-offload is continuously exploring local energy efficient offloading opportunities within the predictable window, and keeping shifting the window to update the offloading decision. Therefore, the good link quality in the deadline can be efficiently utilized.

In the section, we first present an overview of Q-offload, and then discuss three key components of Q-offload: dynamic window length control, local offloading scheme determination and deadline guarantee mechanism.

A. Overview

Figure.4 depicts the workflow of Q-offload. Assuming current slot is t and the system state is (D, b, t) with $D > 0$, the process of Q-offload can be stated as follows:

1) At time t and for the current system state (D, b, t), determining the local offloading scheme over a fixed future interval, say $[t, t + c]$ under two conditions:
   1. $t + c \geq n$. The scenario indicates we can obtain a global information over the window $\mathcal{T}_{t,n}$. Then we can directly solve Eq.3 to determine the optimal offloading scheme. This may be possible if the link follows some link models, e.g., link bursty model [20], [25], link temporal correlation model [21], [26], or the deadline is near.
   2. $t + c < n$. It indicates we cannot have the total global information. Accordingly, we use "transmit D in the slot $i \in [t, t+c]$ with largest $b(i)$" as the decision criterion.
2) Checking whether the deadline constraint can be guaranteed at current system state (D, b, t), i.e. whether the expected remaining capacity is sufficient to transmit D. If yes, applying only the first step in the resulting control sequences. Otherwise, performing the transmission at t and set $a(D, b, t) = 1$.
3) Updating system parameters based on the scheme $a(D, b, t)$.
4) Measuring the system state reached at slot $t + 1$ and repeating the fixed window optimization at time $t + 1$ over the future interval $[t + 1, t + 1 + c]$, starting from the current $(D, b, t + 1)$.

B. Dynamic window length control

To realize Q-offload, it is important to determine an appropriate window length within which the prediction accuracy can be guaranteed, so that the local offloading scheme is likely to be most energy efficient. In this subsection, we discuss how to determine the length of decision window under different conditions to guarantee the prediction accuracy.

First, we define the prediction accuracy as:

Definition IV.1. *The prediction accuracy in $[t, t + c]$ is the mean gap between the predicted values and actual values, i.e.,*

$$Accuracy = 1 - \frac{1}{c} \sum_{i=t}^{t+c} \frac{|\hat{b(i)} - b(i)|}{b(i)} \tag{4}$$

where $\hat{b(i)}$ is the predicted value at i-th slot based on transition matrix $\mathbb{P}$ and $b(i)$ is the true value.

Clearly, if $Accuracy$ is higher, the determined offloading scheme is more likely to be effective in the decision window. On the other hand, we should also notice that high $Accuracy$ is hard to be achieved with a fixed length c under different conditions. For example, the best window length between a static user and a mobile user is often different, since the

Algorithm 1: Pseudo code for Q-offload

Input: Window $\mathcal{T}_{1,n}$, prediction length c, untransmitted data D, transition matrix $\mathbb{P}$, current slot t;
Output: Offloading decision;

```
//Dynamic prediction length control
Accuracy = 1 - (1/c) * sum_{i=t-c+1}^{t} |b^(i) - b(i)| / b(i);
if Accuracy < γ then
    c = max{c_0, (1/2)c};
else
    c = c + 1;
Predict {b(t^+1), b(t^+2), ..., b(t^+c)} based on ℙ;
//Deadline guarantee mechanism
if t + c ≥ n then
    D_safe = sum_{i=t}^{n} b^(i) · Δt;
else
    D_safe = ((n-t+1)/(t-1)) sum_{i=1}^{t-1} b(i) · Δt;
if D > D_safe then
    Exec_Trans(D, t);
//Local offloading scheme determination
if t + c ≥ n then
    S_L(t) = Solve Eq.3;
    if b(t) ≥ min{b(i)|i ∈ S_L(t)} then
        Exec_Trans(D, t);
else
    if b(t) ≥ max{b(t^+1), b(t^+2), ..., b(t^+c)} then
        Exec_Trans(D, i);
        D = max{0, D(k) - Z(i_k)};
if D equals to 0 then
    Return;
else
    Continue Algorithm.1 with t + 1;
```

variation of link quality is usually more significant in the latter scenario. As a result, we should accordingly adjust c to ensure the prediction accuracy.

To achieve this goal, we propose a dynamic control scheme to determine the appropriate c under different conditions. Without loss of generality, we assume the minimum prediction length is $c_0 > 1$ and the accuracy threshold is γ. Basically, we consider two cases:

1) $Accuracy < \gamma$. It indicates under current condition, the prediction length is too long. This may be caused by the change of user mobility, e.g., from static to mobile or meeting new impact factors. Accordingly, the effectiveness of offloading decision would be affected, and thus we should decrease the prediction length. By default, we set $c = \max\{c_0, \frac{1}{2}c\}$ to guarantee quick reaction.
2) $Accuracy \geq \gamma$. It indicates the prediction is accurate in the current condition. Accordingly, we can extend the prediction length to improve the decision by considering more candidates. To avoid making wrong extensions, we set $c = c + 1$ under this condition.

By default, we set $c_0 = 10$ and $\gamma = 0.8$. In the evaluation part, we validate that this setting achieves a good performance in practice.

C. Local offloading scheme determination

In order to determine the local offloading scheme, we should consider whether current prediction window can cover the remaining slots. If yes, we can use a deterministic method to make the offloading decision since we can predict the remaining link quality variation. Otherwise, we introduce a greedy method to handle the condition. More specifically, our scheme is stated as follows:

Case 1: $t + c \geq n$. This condition indicates that we can obtain a global information of the link quality variation for the remaining slots. Accordingly, the optimal scheme should be selecting L largest slots in $\mathcal{T}_{t,n}$ such that their aggregated capacity just holds D. Without loss of generality, we denote the scheme as $S_L(t)$. Then the offloading decision should be transmitting D in the slot $i \in [t, n]$ whose bandwidth is no smaller than the minimum bandwidth in $S_L(t)$, i.e.

$$a(D, b, i) = \begin{cases} 1, & b(i) \geq \min\{b(i) | i \in S_L(t)\} \\ 0, & Otherwise \end{cases} \tag{5}$$

Case 2: $t+c < n$. In this case, we only have limited information about link quality variation over $\mathcal{T}_{t,n}$. To minimize the energy consumption, we should use the slot with the best link quality in the $[t, t + c]$ for data offloading. Therefore, we use "transmit D in the slot $i \in [t, t+c]$ with largest $b(i)$" as the decision criterion. If the most probable evolving path from t to $t + c$ is $L([t, t + c]|\mathbb{P})$. Then the transmission scheme for slot $i \in [t, n]$ under this condition is:

$$a(D, b, i) = \begin{cases} 1, & b(i) \geq \max\{b(i) | b(i) \in L([t, t + c]|\mathbb{P})\} \\ 0, & Otherwise \end{cases} \tag{6}$$

It should be noted for a time window $[t, t + c]$, the method of finding the best link quality is related to the link model. The calculation of $\max\{b(i)|b(i) \in L([t, t + c]|\mathbb{P})\}$ may be different if other models are applied [27], [30].

D. Deadline guarantee mechanism

We propose a deadline guarantee mechanism in Q-offload. The basic idea of the mechanism is to check whether the expected remaining capacity is enough to hold D before conducting the corresponding local scheme. Without loss of generality, we denote the expected remaining capacity at slot t as $D_{safe}(t)$:

$$D_{safe}(t) = \begin{cases} \sum_{i=t}^{n} \hat{b(i)} \cdot \Delta t, & t + c \geq n \\ \frac{n - t + 1}{t - 1} \sum_{i=1}^{t-1} b(i) \cdot \Delta t, & Otherwise \end{cases} \tag{7}$$

If $D \leq D_{safe}(t)$, it indicates there is enough space for scheduling. Then Q-offload follows the resulting offloading scheme. Otherwise, if $D > D_{safe}(t)$, the deadline constraint

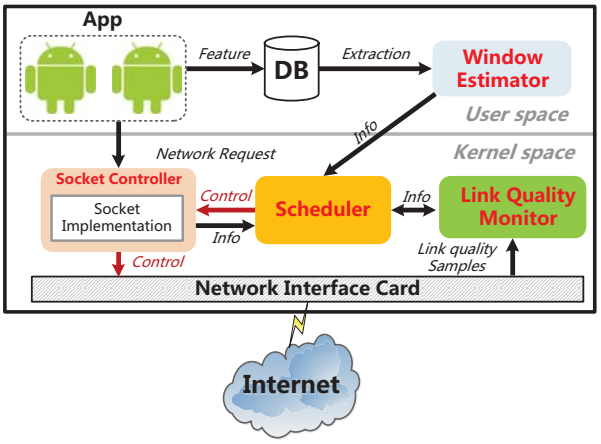


Figure 5: The system overview of Q-offload

of D may be violated if $a(D, b, t) = 0$. To avoid this phenomenon, Q-offload will perform the transmission of D in the current slot.

E. Summary

Based on the descriptions mentioned above, we summarize Q-offload in Algorithm.1. According to our design, Q-offload is a moderate offloading scheme whose first concern is satisfying users' requirement. When the offloading pressure is light, it will exploit the good link quality for offloading and achieve energy efficiency improvement. On the other hand, when the offloading pressure is high, e.g., there is less space for scheduling data offloading within the deadline, Q-offload will offload as much data as possible to to satisfy users' requirement. In addition, if multiple D coexist with different deadlines, we can also apply Q-offload in the scenario by offloading D sequentially according to their deadline urgency.

V. Implementation

Based on the design, we have implemented Q-offload as a cross layer approach on Android system for energy efficient offloading.

A. System overview

The system architecture of Q-offload is shown in Fig.5. First, for each App, it tracks the usage feature in the database, i.e. the average interval between the start of the network request and usage. When the App has a network request, the window estimator will extract the corresponding deadline and pass it to the scheduler. Meanwhile, the link quality monitor will also pass current link quality and transition matrix to the scheduler. Following Algorithm.1, the scheduler will determine the offloading scheme for each network request.

B. Window Estimator

This part is responsible for deriving the window $\mathcal{T}$ for each offloading request. To obtain the value, we query the database and extract the mean window for the corresponding App. If there is no information about the App, e.g., newly installed, we also allow the part to obtain the information from user's input before making offloading decisions. Since user only needs to enter one number, the extra overhead is negligible and will not affect normal user experience.

C. Link Quality Monitor

As aforementioned, this part tracks the link quality values from network interface card and derives the transition matrix for the scheduler. Based on the link quality values, it maintains a transition matrix $\mathbb{P}$. For each transition, e.g., $b(i)$ to $b(j)$, the link quality monitor updates the entry from $b(i)$ to $b(j)$ as follows:

$$P_{b(i)\to b(j)} = \frac{\delta \cdot N_{b(i)\to b(j)} + 1}{\delta \cdot \sum_{\forall k} N_{b(i)\to b(k)} + 1} \tag{8}$$

where $N_{b(i)\to b(k)}$ denotes the number of transitions from $b(i)$ to $b(k)$ and δ is a loss ratio. We set $\delta = 0.5$ to make in-time update. Moreover, we set $b(i) = 50KB/s \cdot \lfloor \frac{b(i)}{50KB/s} \rfloor$ to control the number of entries in the matrix.

D. Scheduler

The scheduler makes offloading decisions based on the information from the window estimator, socket controller and link quality monitor. For each iteration in Algorithm.1, the scheduler will pass the offloading decision to the socket controller. Accordingly, socket controller will implement the decision and return the size of untransmitted data to the scheduler to make further decisions. This process will continue until data D is fully offloaded. After that, c will be set as 10 again.

E. Socket Controller

The socket controller is implemented as a firewall of the standard socket implementation based on the kernel function *iptables*, which manages the real-time rules of data streams. When the offloading decision is "transmitting data in the current slot", the module makes no change. On the other hand, when the offloading decision is "remain idle", then the module adds the rule *–limit 1/minute* for the corresponding port to restrict the speed for 1 packet per minute for the data stream. By doing this, we maintain the activeness of the stream while also limiting the transmission in undesired slots. Accordingly, we can reschedule data transmission to our offloading decisions.

VI. Evaluation

In this section, we demonstrate the evaluation of Q-offload scheme through both trace-driven analysis and real-world experiments.

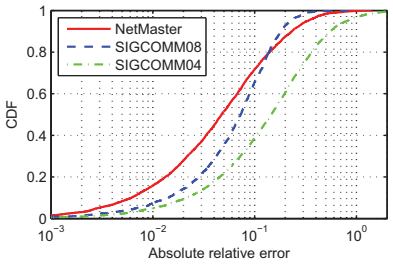


Figure 6: Prediction error on initial c_0.

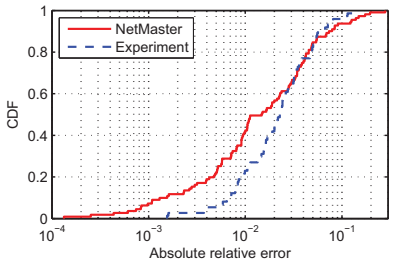


Figure 7: Prediction error when user is static.

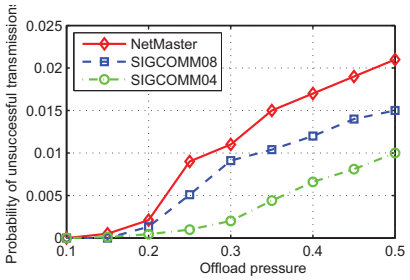
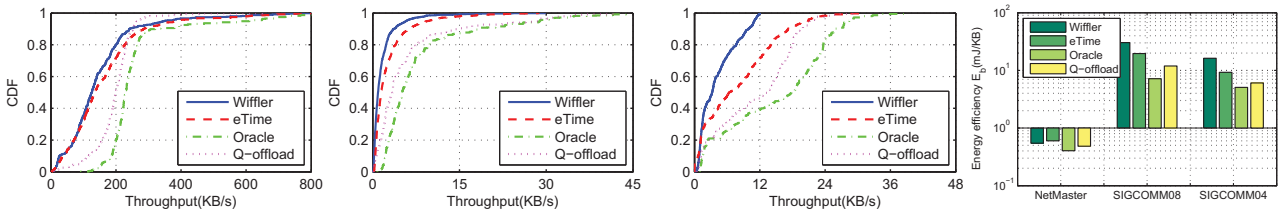


Figure 8: Robustness of deadline guarantee mechanism.



(a) On NetMaster (b) On SIGCOMM08 (c) On SIGCOMM04 (d) Energy efficiency comparison

Figure 9: Trace driven analysis of the performance of Wiffler, eTime, Q-offload and the oracle.

A. Experimental setup

In order to evaluate the performance of Q-offload scheme comprehensively, we conduct both trace-driven analysis and real world experiments detailed as follows:

Trace-driven analysis: We use three datasets for testing: SIGCOMM08, NetMaster and SIGCOMM04. The number of tested sessions are $\approx 10^5$, which contain over 10^9 packet transmissions over WiFi. In each test, we vary the deadline length from 20 to 500 seconds to simulate different usage patterns and user requirements. In addition, we set the offloaded data size as 20% of total capacity within the deadline by default. The target platform is also Samsung Galaxy Note3.

Real-world experiments: We also implement a prototype of Q-offload on Samsung Galaxy Note3 to evaluate its performance in practice. To test the performance of Q-offload in different scenarios, we develop a file download application with about 500 lines of java code. It downloads files with sizes varying from 50KB to 10MB from a remote server in Wuxi. The file download application can receive user specified deadline for the corresponding download, which is specified in the following subsection. To evaluate the performance of different schemes, for each file size, we input different feasible deadlines and repeat the download for 10 times in a walking environment with different trajectories. To derive accurate E_b for the download operation, we enable Power-saving Mode (PSM) to eliminate energy consumed in idle listening and include the CPU energy consumption of Algorithm.1 by offline analysis. Moreover, E_b is still calculated from Eq.1.

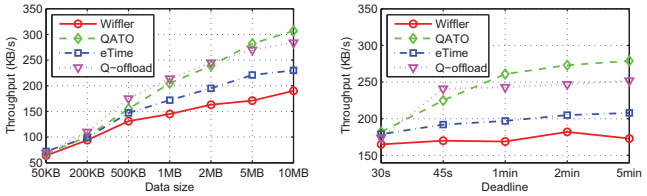
State-of-the-arts: We compare Q-offload against the following state-of-the-arts: (1) Wiffler [5], which is a best-effort offloading scheme widely adopted in many existing works, (2) eTime [22], which utilizes a local greedy framework to derive the efficient offloading decision, and (3) QATO [28], which alternates between different network access for data offloading. In our experiments, we use cellular network (HSPA+) as the alternative if WiFi link quality is poor. We implement QATO in the real-world experiments. We also compare different approaches to the (4) Oracle, the optimal scheme by offline analysis.

B. Component analysis

We first investigate whether the components of Q-offload are robust under different conditions. Basically, we focus on three components: the setting of c_0, the setting of γ and the deadline guarantee mechanism.

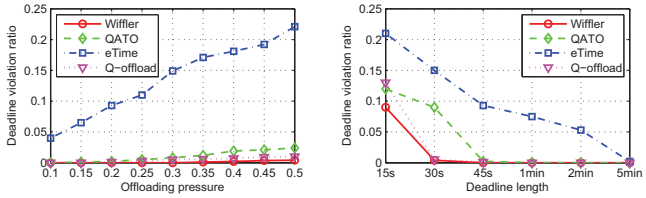
1) The setting of c_0: First, we examine whether the initial state of $c_0 = 10$ can guarantee a high prediction accuracy. By conducting trace-driven analysis, we plot the cdf of prediction error in Fig.6. As shown in the figure, we observe that under the prediction length of 10 seconds, the mean prediction error in over 60% of traces in less than 20%. The results demonstrate $c_0 = 10$ is in fact an appropriate initial state for Q-offload to start with.

2) The setting of γ: The value of γ impacts the decision horizon evolution. Clearly, an appropriate γ should satisfy: (1) increasing the horizon length when link quality is stable and (2) restricting the horizon length to around c_0 when link quality varies. For the first requirement, we should notice that even when the user remains static, there still exists link quality variation. If γ is too small, the system will not perform well under this scenario. To address this issue, we conduct both trace-driven analysis and real-world



(a) Under different data sizes (b) Under different deadlines

Figure 10: Throughput analysis in real-world experiments



(a) Under different offloading pressure (b) Under different deadlines

Figure 11: Deadline violation ratio analysis in real-world experiments

evaluations with different WiFi APs when the user remains static. We plot the prediction error with c_0 under this scenario in Fig.7. We find that for over 95% traces, the prediction error is less that 20%. Accordingly, the threshold $\gamma = 0.8$ is proper to adapt Q-offload to different conditions.

3) The robustness of deadline guarantee mechanism: A key requirement of Q-offload is guaranteeing the deadline requirement. To analyze the robustness of deadline guarantee mechanism in Q-offload, we make trace-driven analysis by varying the offloading pressure from 0.1 to 0.5 of the total capacity within the deadline. In addition, for each size, we vary its deadline from 20 to 500 seconds to obtain a comprehensive analysis. We plot the results in Fig.8 and find that the largest unsuccessful rate is only 2%. The results demonstrate that Q-offload is robust in guaranteeing deadlines for various offloading pressure under different conditions.

C. Trace-driven analysis

1) Throughput comparison: For throughput comparison, we observe that Q-offload significantly improves the throughput in data offloading on all datasets, compared with Wiffler and eTime. In average, the throughput in transmitting the same amount of data increases by 2.6x (0.38 to 6.5x) compared with Wiffler, while 1.7x (0.25-2.1x) compared with eTime, which are shown in Fig.9a, Fig.9b and Fig.9c. Further investigation shows that this is mainly because Q-offload can efficiently exploit good link quality within the corresponding deadlines. By scheduling the transmission of D to those slots with good link quality, Q-offload can decrease the time in transmitting D, and thus improve the overall throughput. Since the deadline is fixed in our scenario, we also notice that eTime tends to transmit D until the deadline is near. Accordingly, lots of slots with good link quality are wasted, which explains why there is a gap between Q-offload and itself. Furthermore, we also notice that the approximation ratio between Q-offload and the oracle scheme is not significant, i.e. 0.775 (0.51 to 0.91). Accordingly, this observation demonstrates our heuristic scheme is efficient in achieving energy efficient offloading performance.

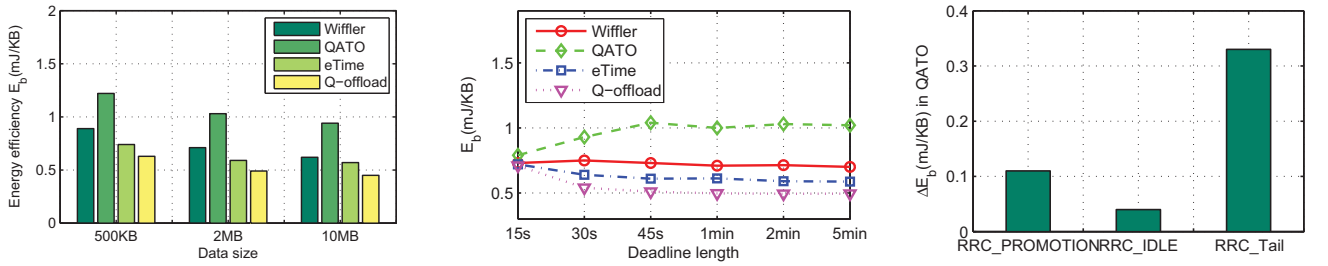
2) Energy efficiency comparison: Following the throughput analysis, we also examine the improvement of energy efficiency E_b after using Q-offload on trace-driven analysis. We plot the results in Fig.9d. In consistence with the former results, we also find that Q-offload decreases the average E_b 55.7%(23.4%∼88.7%) in average. Compared with Wiffler, Q-offload decreases E_b by 52.5% while for eTime, Q-offload saves 58.8% E_b in transmitting the same D. The results show that Q-offload can spare significant portion of energy in WiFi offloading by exploiting good in-deadline link quality.

D. Real-world experiments

In real-world experiments, we also compare the performance Q-offload with the selected benchmarks under different data sizes and deadline lengths. Basically, we evaluate their performance in three aspects: throughput, deadline violation ratio and energy efficiency.

1) Throughput comparison: For throughput comparison, we first analyze their performance under different data sizes. We choose the data sizes varying from 50KB to 10MB for testing and associate feasible deadlines for them, i.e. 30 seconds for the data under 500KB, 1min for data size between 1MB and 2MB and 3min for data size over 5MB. We plot the results in Fig.10a and find that compared with eTime and Wiffler, Q-offload improves the throughput in transmitting the same amount of data from 7.8% to 49% under different data sizes. Then, for a given data size 2MB, we also analyze the performance of different schemes under different deadline lengths in Fig.10b. We find that the convergence to the highest throughput of Q-offload is also the fastest among all benchmarks. On the other hand, we also notice that the throughput improve of Q-offload is not obvious compared with QATO. This is because in a mobile environment, link quality variation is significant, which has been discussed in Section.II. When meeting poor link quality, reusing cellular network (HSPA+) would be a better choice. However, in the following subsection, we will point out that the switch operation in QATO is not energy efficient in the current communication system.

2) Deadline analysis: An important requirement of the offloading schemes is guaranteeing the deadline constraints. In Fig.11, we plot the deadline violation ratios for all tested schemes. First, we analyze the ratio when the offloading pressure increases from 0.1 to 0.5. We observe that for QATO, Wiffler and Q-offload, the deadline is well guaranteed for almost 100 percent. However, for eTime, we find the



(a) E_b comparison for different sizes. (b) E_b evolution on different deadlines. (c) The decomposition of extra E_b in QATO.

Figure 12: Energy efficiency analysis in real-world experiments

ratio is high. This is because when the deadline is fixed, it tends to enable the data transmission when the deadline is near. In some cases, the remaining capacity is insufficient to offload D which leads relatively higher violation ratio. In addition, we also examine the deadline violation ratio for a fixed data size, i.e. 2MB, under different deadline lengths in Fig.11b. As shown in the figure, when the deadline increases from 15s and 30s, Q-offload can also efficiently decrease the violation ratio to near zero as the real-time offloading scheme Wiffler, owing to its deadline guarantee mechanism.

3) Energy efficiency comparison: Then we investigate the energy efficiency of different schemes. In Fig.12a, we plot the per KB energy consumption E_b for all schemes with data size 500KB, 2MB and 10MB. Compared with the benchmarks, Q-offload improves the E_b for the same amount of data from 15% to 52.1%. In addition, we also analyze E_b evolution of all schemes for a fixed data size, i.e., 2MB under different deadlines. As shown in Fig.12b, we observe that for each deadline, the E_b of Q-offload is smaller than other schemes, which demonstrates the benefit of exploiting good link quality in WiFi offloading. Interestingly, we also observe that the E_b of QATO is the highest among four schemes, which does not match with its throughput performance. By further analysis, we find this is due to the extra energy cost of data transmission in the transmission unrelated power states of cellular network, e.g., *RRC_PROMOTION* and *RRC_Tail*. This is because in QATO, it has to switch frequently between WiFi and cellular based on link quality variation. For each switch from WiFi to cellular, the radio has to be promoted from sleep to DCH for data transmission. When the data transmission completes, a tail will also occur and waste significant portion of energy. For each switch, lots of energy will be wasted in power state transitions, e.g., RRC_PROMOTION and RRC_IDLE, and RRC_tail. By decomposing the extra energy cost of transmitting 2MB data in Fig.12c, we find ΔE_b is over 0.5mJ/KB, which is significant compared with the E_b of Q-offload. Although some solutions have been proposed, they all require fundamental modifications in Android system or transmitting protocol stack, which limits their applications [12]. In conclusion, Q-offload can achieve more energy efficient WiFi offloading compared with state-of-the-arts.

VII. Related Work

Optimizing the energy efficiency of data transmission on smartphones has been a hot spot in recent years. In this section, we group the existing works related to Q-offload design into the following three categories.

Measurement The works in [2], [10], [14], [29]–[31] conduct extensive experiments to analyze network usage on smartphones. Huang et al. [2] collect 118GB user traces to examine the performance of different network types that can be used on smartphones. The results illustrate that modern cellular network, e.g., 3G/4G is less energy efficient than WiFi in data transmission. Balasubramanian et al. [31] find that 3G incurs a high tail energy overhead for lingering in the high-power state after the completion of a transfer. Some fine-grained measurements are also made to analyze the factors that affect network energy efficiency [14], [27], [30]. The results from those measurements imply that there exists a great potential in optimizing network energy consumption.

Cellular network optimization To optimize the energy consumption of cellular networks, researchers have proposed various approaches [7], [12], [27], [28], [30], [32]. Athivarapu et al. [12] utilizes a decision tree based method to reduce energy cost in radio tail of cellular data transmission. Schulman et al. [30] predict the signal strength variation for achieving energy-aware cellular data scheduling. Those works can be recognized as supplements to our Q-offload scheme when WiFi is unavailable.

Data offloading through WiFi Offloading traffic to WiFi is recognized as a viable solution in many existing works [3]–[5] and also implemented as the default setting on modern smartphones. Ding et al. [3] propose a collaborative offloading scheme among cellular operators, WiFi service providers and end-users. Balasubramanian et al. [5] design a real-time WiFi offloading strategy, i.e. Wiffler, by predicting WiFi access availability. Shu et al. [22] propose an offloading method eTime based on Lyapunov stability. By determining the tradeoff between delay cost and offloading opportunities, they can guarantee a feasible strategy on $t \rightarrow \infty$. However, in their framework, the deadline constraint is hard to be guaranteed due to the difficulty in determining

proper tradeoff function. In a word, most existing works ignore the fact that WiFi link quality varies significantly under different scenarios [10], [14]. The variation of link quality may significantly affect the energy efficiency of WiFi offloading. Our work aims to address this fundamental issue for achieving energy efficient WiFi offloading.

VIII. Conclusion

WiFi offloading provides great potential in satisfying ubiquitous applications on smartphones. However, existing WiFi offloading approaches, presumed to be energy efficient, ignore the impact of link dynamics and may even increase the energy consumption. To address such a problem, we propose Q-offload, a quality aware scheme to achieve energy efficient WiFi offloading that can work with link dynamics. Compared with existing works, the main advantage of Q-offload is reducing offloading energy consumption while not affecting user experience. We have implemented Q-offload on the Android platform and conducted extensive evaluations in both trace-driven analysis and real-world experiments. The evaluation results show that Q-offload can reduce in-average energy consumption by 55.7% in trace-driven analysis while 33.5% in real-world experiments, compared with state-of-the-arts.

Acknowledgements

This work was supported in part by NSFC Major Program under grant No. 61190110, a grant from RGC under the contracts 615613 and 16211715, NSFC under Grants 61202359, 61373166, 61170213, a grant from China Cache Int. Corp. under the contract CCNT12EG01, a grant from NSFC Guangdong under the contract U1301253, National Science Fund for Excellent Young Scientist No. 61422207 and Huawei Innovation Research Program (HIRP).

References

[1] J. Ericsson, "Ericsson mobility report," 2014.

[2] 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 *Proceedings of ACM MobiSys*, 2012, pp. 225–238.

[3] A. Y. Ding, B. Han, Y. Xiao, P. Hui, A. Srinivasan, M. Kojo, and S. Tarkoma, "Enabling energy-aware collaborative mobile data offloading for smartphones," in *Proceedings of IEEE SECON*, 2013, pp. 487–495.

[4] K. Lee, J. Lee, Y. Yi, I. Rhee, and S. Chong, "Mobile data offloading: how much can wifi deliver?" in *Proceedings of ACM CoNEXT*, 2010, p. 26.

[5] A. Balasubramanian, R. Mahajan, and A. Venkataramani, "Augmenting mobile 3g using wifi," in *Proceedings of ACM MobiSys*, 2010, pp. 209–222.

[6] M. V. Barbera, S. Kosta, A. Mei, and J. Stefa, "To offload or not to offload? the bandwidth and energy costs of mobile cloud computing," in *Proceedings of IEEE INFOCOM*, 2013, pp. 1285–1293.

[7] F. Qian, Z. Wang, Y. Gao, J. Huang, A. Gerber, Z. Mao, S. Sen, and O. Spatscheck, "Periodic transfers in mobile applications: network-wide origin, impact, and optimization," in *Proceedings of ACM WWW*, 2012, pp. 51–60.

[8] J. Huang, F. Qian, Z. M. Mao, S. Sen, and O. Spatscheck, "Screen-off traffic characterization and optimization in 3g/4g networks," in *Proceedings of ACM IMC*, 2012, pp. 357–364.

[9] J. Manweiler and R. Roy Choudhury, "Avoiding the rush hours: Wifi energy management via traffic isolation," in *Proceedings of ACM MobiSys*, 2011, pp. 253–266.

[10] N. Ding, D. Wagner, X. Chen, Y. C. Hu, and A. Rice, "Characterizing and modeling the impact of wireless signal strength on smartphone battery drain," in *Proceedings of ACM SIGMETRICS*, 2013, pp. 29–40.

[11] J. Huang, F. Qian, Y. Guo, Y. Zhou, Q. Xu, Z. M. Mao, S. Sen, and O. Spatscheck, "An in-depth study of lte: effect of network protocol and application behavior on performance," in *Proceedings of ACM SIGCOMM*, 2013, pp. 363–374.

[12] P. K. Athivarapu, R. Bhagwan, S. Guha, V. Navda, R. Ramjee, D. Arora, V. N. Padmanabhan, and G. Varghese, "Radiojockey: mining program execution to optimize cellular radio usage," in *Proceedings of ACM MobiCom*, 2012, pp. 101–112.

[13] K. Fahad R. Dogar, Peter Steenkiste, "Catnap: Exploiting high bandwidth wireless interfaces to save energy for mobile devices," in *Proceedings of ACM MobiSys*, 2010.

[14] A. Patro, S. Govindan, and S. Banerjee, "Observing home wireless experience through wifi aps," in *Proceedings of ACM MobiCom*, 2013, pp. 339–350.

[15] F. Mehmeti and T. Spyropoulos, "Is it worth to be patient? analysis and optimization of delayed mobile data offloading," in *Proceedings of IEEE INFOCOM*, 2014, pp. 2364–2372.

[16] C. Joe-Wong, S. Ha, and J. Bawa, "When the price is right: Enabling timedependent pricing for mobile data," *ACM SIGCHI 2013*, 2013.

[17] C. Shi, K. Joshi, R. K. Panta, M. H. Ammar, and E. W. Zegura, "Coast: collaborative application-aware scheduling of last-mile cellular traffic," in *Proceedings of ACM MobiSys*, 2014, pp. 245–258.

[18] Y. Zhang, Y. He, X. Wu, Y. Liu, and W. He, "Netmaster: Taming energy devourers on smartphones," in *Proceedings of IEEE ICPP*, 2014, pp. 301–310.

[19] "CRAWDAD," Downloaded from http://crawdad.org/, 2015.

[20] A. Kamthe, M. A. Carreira-Perpinán, and A. E. Cerpa, "M&m: multi-level markov model for wireless link simulations," in *Proceedings of ACM Sensys*, 2009, pp. 57–70.

[21] K. Srinivasan, M. A. Kazandjieva, S. Agarwal, and P. Levis, "The β-factor: measuring wireless link burstiness," in *Proceedings of ACM Sensys*. ACM, 2008, pp. 29–42.

[22] P. Shu, F. Liu, H. Jin, M. Chen, F. Wen, and Y. Qu, "etime: energy-efficient transmission between cloud and mobile devices," in *Proceedings of IEEE INFOCOM*, 2013, pp. 195–199.

[23] H.-J. Böckenhauer, D. Komm, R. Královič, and P. Rossmanith, *On the advice complexity of the knapsack problem*. Springer, 2012.

[24] E. F. Camacho and C. B. Alba, *Model predictive control*. Springer Science & Business Media, 2013.

[25] M. H. Alizai, O. Landsiedel, J. Á. B. Link, S. Götz, and K. Wehrle, "Bursty traffic over bursty links," in *Proceedings of ACM Sensys*, 2009, pp. 71–84.

[26] Z. Zhao, W. Dong, G. Guan, J. Bu, T. Gu, and C. Chen, "Modeling link correlation in low-power wireless networks," in *Proceedings of IEEE INFOCOM*, 2015.

[27] A. Rahmati and L. Zhong, "Context-for-wireless: context-sensitive energy-efficient wireless data transfer," in *Proceedings of ACM MobiSys*, 2007, pp. 165–178.

[28] W. Hu and G. Cao, "Quality-aware traffic offloading in wireless networks," in *Proceedings of ACM MobiHoc*, 2014, pp. 277–286.

[29] P. Deshpande, X. Hou, and S. R. Das, "Performance comparison of 3g and metro-scale wifi for vehicular network access," in *Proceedings of ACM IMC*, 2010, pp. 301–307.

[30] A. Schulman, V. Navda, R. Ramjee, N. Spring, P. Deshpande, C. Grunewald, K. Jain, and V. N. Padmanabhan, "Bartendr: a practical approach to energy-aware cellular data scheduling," in *Proceedings of ACM MobiCom*, 2010, pp. 85–96.

[31] N. Balasubramanian, A. Balasubramanian, and A. Venkataramani, "Energy consumption in mobile phones: a measurement study and implications for network applications," in *Proceedings of ACM IMC*, 2009, pp. 280–293.

[32] A. Chakraborty, V. Navda, V. N. Padmanabhan, and R. Ramjee, "Coordinating cellular background transfers using loadsense," in *Proceedings of ACM MobiCom*, 2013, pp. 63–74.