Energy-Delay Tradeoff for Request Bundling on Smartphones

Ali Sehati and Majid Ghaderi
Department of Computer Science, University of Calgary
{asehati, mghaderi}@ucalgary.ca

***Abstract*—To reduce the energy consumption of a smartphone, multiple data transfer requests from applications can be bundled together and granted at once in order to reduce the time the radio interface is on. The side effect of bundling is the increased delay experienced by mobile applications. While several bundling algorithms have been proposed in the literature, a general and systematic solution to balance the energy-delay tradeoff is missing. In this paper, we formulate bundling as a cost minimization problem, in which the tradeoff between energy and delay is captured by a cost function. We then propose an online algorithm for minimizing the bundling cost and show that the algorithm is 4-competitive with respect to the optimal offline algorithm that knows the entire sequence of data transfer requests a priori. We evaluate the performance of the proposed algorithm and the accuracy of our results in a range of realistic scenarios using both model-driven simulations and real experiments on a smartphone. Our results show that depending on the delay tolerance level of a user, energy savings ranging from zero (delay intolerant) to about 100% (delay tolerant) can be achieved using our algorithm.**

I. Introduction

A. Motivation

A major contributor to the total energy consumption of a smartphone is the energy consumed by its radio interface [1]. Recent studies show that a significant portion of the radio energy is spent when there is no active data transfer [2], [3]. The reason is that once a data transfer is completed, the radio interface of the device lingers in the (high power) "on" state for some *tail time* before switching to the (low power) "off" state. The reason for this behavior is to avoid switching radio states in case any data transfer request arrives during the tail time, which in turn reduces network signaling overhead. In 3G/LTE networks, typical values for the tail time are around several seconds depending on the carrier configuration [4], [5].

The implication of tail time for a smartphone is that periodic and intermittent data transfer requests can keep the radio on for a long period of time, leading to rapid depletion of its battery. Such a traffic pattern is indeed the characteristic of many mobile applications due to data syncs and status updates [6], keep-alive messages [7], code offloading [8] and even web browsing [9], [10]. A well-know technique to alleviate the effect of tail time is request *bundling* [2], [3], [8], [11]–[13]. With bundling, rather than granting individual data transfer requests as they arrive, multiple requests are bundled together and granted at once. Thus, the tail energy is amortized over multiple transfers in a bundle, which results in reduced radio energy consumption.

A critical question when implementing request bundling is *how long to wait to create a bundle*. By waiting too long, potentially more requests can be bundled together leading to more energy savings. This comes, however, at the expense of increased delay, which may negatively affect the performance of mobile applications, and consequently the user experience. The difficulty in designing an optimal bundling algorithm is that the bundling decisions have to be made online *without knowing the timing of future data transfer requests*.

B. Related Work

The following is a brief review of several works on request bundling that are more relevant to this paper.

1) Packet Bundling: Packet bundling can be implemented at the link layer irrespective of the applications running on the device. Deng *et al.* [3] proposed a learning algorithm to predict the traffic pattern in order to decide when a packet bundle should be granted. In another work, Dogar *et al.* [14] showed that the bandwidth discrepancy between wired and wireless segments of a connection can create small idle gaps between successive packets. They then designed a proxy-based system that bundles multiple packets together in order to maximize the idle time between bundle grants. Alonso *et al.* [11] proposed bundling packets in order to improve the energy efficiency of the DRX mechanism in LTE networks. Assuming a Poisson packet arrival process, they computed the optimal size of bundles that minimizes the radio energy consumption, while ensuring a bounded average packet delay.

2) Mobile Web Browsing: Hoque *et al.* [15] showed that when viewing web pages, there are idle gaps between consecutive web object downloads. This is due to the fact that inter-object dependencies in web pages require some processing, for example to evaluate Java scripts, which can create delays between object downloads [9]. To reduce the radio energy consumption of mobile web browsing, a proxy-based architecture was proposed in [12], [13], in which a network-hosted proxy fetches web objects from remote servers and then sends them to the client device in bundles. In particular, using the heuristic assumption of equal-sized bundles, Sivakumar *et al.* [12] computed the optimal number of bundles that minimizes the radio energy usage of visiting a web page.

3) Delayed Data Offloading: While not directly related to minimizing the radio on time on smartphones, the problem in delayed data offloading has a similar structure to the bundling problem considered in this paper, and hence any solution for the bundling problem will be of benefit to the data offloading

problem as well. In delayed data offloading, the problem is to decide how long to wait until a WiFi network becomes available to offload data transfer requests from the cellular network to the WiFi network [16], [17].

4) Delay-Tolerant Applications: Balasubramanian *et al.* [2] proposed a threshold-based bundling algorithm called TailEnder which achieves a competitive ratio of 2. The idea is that data transfer requests from delay-tolerant applications can be postponed up to a deadline without any penalty. The TailEnder requires modification of applications so that they can inform the bundling algorithm of their deadlines.

5) Mobile Code Offloading: In practice, it is difficult to know the deadlines for data transfer requests particularly for non-delay-tolerant applications. Instead, Xiang *et al.* [8] formulated the bundling problem in the context of mobile code offloading as a cost minimization problem, where the cost of bundling is given as a function of both energy and delay. Thus, a bundling algorithm may delay bundles arbitrarily in order to reduce its energy cost as long as it is willing to accept the increased delay cost. Then, based on [18], they devised an online algorithm to minimize the cost of bundling.

Clearly, with bundling, there is a tradeoff between energy saving and increased delay. In general, the energy-delay tradeoff depends on various contextual and technological factors [19]. For example, a user possessing a smartphone with full battery may prefer earlier task completion over energy saving. With the exception of [8], the aforementioned works do not have the flexibility to achieve different energy-delay tradeoffs. Indeed, the problem setting considered in our work is similar to the one considered in [8]. There are, however, significant differences between the two works, as discussed next.

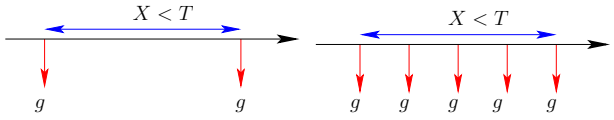
C. Our Work

For a bundling algorithm A, define the cost of the algorithm as a weighted combination of its energy and delay cost, denoted by E_A and D_A, respectively. That is,

$$C_A = E_A + \alpha D_A, \tag{1}$$

where, C_A denotes the total cost of algorithm A. The energy cost E_A is defined as the time the radio spends in the on state under algorithm A. The exact definition of the delay cost D_A is clarified later. The weight factor $\alpha \geq 0$ is used to achieve a desired tradeoff between energy and delay. A smaller value of α indicates the willingness of the user to tolerate a higher delay for the sake of reducing the radio energy consumption. By controlling α, different energy-delay tradeoffs can be achieved ranging from maximum energy reduction (with $\alpha = 0$) to zero energy saving (with $\alpha \gg 1$).

The problem considered in this paper is how to minimize the cost C_A without knowing the future data transfer requests a priori. We call this problem the *Energy-Aware Bundling* problem (EnerB) in the rest of the paper. Inspired by the classic Ski Rental problem [20], we design a deterministic online algorithm for EnerB called the *Break-Even* algorithm (BE). In BE, a grant is made when the energy cost and weighted delay cost of a bundle become equal. Our algorithm and its analysis are considerably general and can accommodate different definitions of the delay cost. We prove that the competitive ratio of BE with respect to the optimal offline algorithm that knows the future data transfer requests in advance is 4. Moreover, we show that the ratio 4 is tight. Extensive simulation and experimental results are also provided to study the behavior of BE in a range of scenarios. Our results show that in realistic scenarios, the performance of BE is significantly better than what is implied by the worst-case competitive ratio 4.



(a) Two grants. (b) Multiple grants.

Fig. 1: Energy cost is variable: Scenarios (a) and (b) have the same energy cost even though they have different number of grants.

One would appreciate the resemblance between our problem and the *dynamic TCP acknowledgment* problem (DynAck) [21]. In DynAck, the goal is to minimize the number of acks sent by bundling multiple acks together and sending a single cumulative acknowledgment. The problem can then be cast as a variation of EnerB, in which the energy cost E_A is replaced by the acknowledgment cost, *i.e.*, the number of acks sent. Indeed, this relation was exploited in [8] to design a bundling algorithm for mobile code offloading. There is, however, a subtle difference between problems EnerB and DynAck, which completely changes the problem. *Whereas the cost of sending an ack is constant, the energy cost of making a grant is variable*. Specifically, the cost of sending an ack is always 1 (no matter how long the algorithm waits to send the ack), and thus, the acknowledgment cost incurred by the algorithm is simply given by the total number of acks sent.

In contrast, in EnerB, the energy cost of making a grant is *variable* as it represents the amount of time the radio has been on. In this case, the energy cost E_A is the summation of radio on times over the duration of the algorithm, and hence the number of grants may not have any relation to E_A. Consider the scenarios depicted in Fig. 1, where T denotes the tail time. Grants are specified by g on the figure. In Fig. 1(a), there is one grant at the beginning and one grant at the end of an interval of length X. In Fig. 1(b), in addition to the grants at the beginning and end of the interval of length X, there are three other grants during the interval. Since $X < T$, in both scenarios, the radio is on for the entire duration X. Thus, the energy cost of the bundling algorithm in both scenarios is equal to X, even though there are different number of grants during the same interval. Thus, as opposed to the analysis presented in [8], [18], [21], if algorithm A makes more grants than algorithm B, we cannot conclude that $E_A > E_B$, as shown in Fig. 1. This relation is at the heart of the analysis presented for the performance of the online algorithms proposed in [8], [18], [21]. As shown later, we design a completely new approach to analyze the performance of our algorithm as the approach

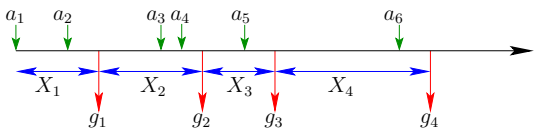


Fig. 2: Relation between arrivals, grants and intervals.

taken in the above mentioned works does not apply.

D. Paper Organization

We start by formally describing the problem in Section II. In Section III we present an optimal offline algorithm. Our proposed online algorithm is presented in Section IV, which is then followed by a detailed analysis of the algorithm in Section V. Performance evaluation results are discussed in Section VI. Section VII concludes the paper.

II. Problem Statement

Consider a sequence of data transfer request arrivals $\mathcal{A} = \langle a_1, \ldots, a_n \rangle$, where a_i denotes the arrival time of request i. *The sequence $\mathcal{A}$ is not known in advance*. Without loss of generality, we assume that the radio is off when the first request arrives. The goal is to design an online algorithm to bundle multiple requests together and grant them at once as opposed to individually granting each request. Depending on the application context, a data transfer request may involve uploading and/or downloading data over the radio interface.

Let $\mathcal{G}_A = \langle g_1, \ldots, g_k \rangle$ denote the sequence of grants made by some algorithm A, for the arrival sequence $\mathcal{A}$, where g_i denotes the time of grant i. Let $\mathcal{X}_A = \{X_1, \ldots, X_k\}$ denote the set of all grant intervals of algorithm A, where $X_1 = [a_1, g_1]$ and $X_i = (g_{i-1}, g_i]$, for $i \geq 2$. All requests that arrive during the interval X_i are bundled together and granted at time g_i. Throughout the paper, we use the notation X_i to refer to the i-th grant interval as well as the length of that interval, when there is no ambiguity.

Fig. 2 shows the relation between arrivals and grants. The objective of the bundling algorithm is to determine the grant times g_i that minimize the cost $C_A = E_A + \alpha D_A$.

A. Energy Cost

The energy cost E_A is defined as the time the radio spends in the on state under algorithm A. Let T denote the tail time. Then, the energy cost of grant interval X_i is given by $E_A(X_i) = \min\{X_i, T\}$. It then follows that,

$$E_A = \sum_{X_i \in \mathcal{X}_A} E_A(X_i) + T = \sum_{X_i \in \mathcal{X}_A} \min\{X_i, T\} + T, \quad (2)$$

where the additional term T is added to account for a tail time after the last grant. To simplify the analysis, similar to [8], [18], [21], we have ignored the transfer time of bundles as this time is the same for every bundling algorithm.

B. Delay Cost

The delay cost is defined as the sum of delays of all requests handled by the algorithm. We use the notation $D_A(X_i)$ to denote the delay cost of bundle i, which includes all requests that arrive during interval X_i. Consider a request $a_j \in X_i$. The delay cost of request a_j is given by $(g_i - a_j)$. The delay cost of bundle i is then expressed as $D_A(X_i) = \sum_{a_j \in X_i} (g_i - a_j)$. Thus, it follows that,

$$D_A = \sum_{X_i \in \mathcal{X}_A} D_A(X_i) = \sum_{X_i \in \mathcal{X}_A} \sum_{a_j \in X_i} (g_i - a_j). \quad (3)$$

It will become clear later, in Section IV, that our online algorithm as well as its upper bound analysis are independent of the specific delay cost function considered and can be applied to a variety of other delay cost definitions.

III. Optimal Offline Algorithm

The optimal offline algorithm, referred to as OPT, knows the entire request arrival sequence a priori. While unrealistic, OPT can be used as a benchmark when evaluating the performance of our online algorithm. We design OPT based on the observation that every grant of the optimal algorithm happens right at the time of some request arrival, *i.e.*, for all $g_i \in \mathcal{G}_{\text{OPT}}$, we have $g_i = a_k$, for some $a_k \in \mathcal{A}$. That is, the optimal algorithm never makes a grant in-between two arrivals because doing so would only increase its cost. As a result, we can consider the problem of finding the optimal grant sequence as a discrete time optimization problem. Thus, upon each request arrival, the optimal algorithm should decide whether to make a grant for the current bundle or wait for the next request arrival.

Algorithm 1 OPT: Optimal Offline Algorithm

Input: $\mathcal{A} = \langle a_1, a_2, \ldots, a_n \rangle$
Output: C_{min}, Seq
1: **Initialize:** $C_{min}[1] = 0$
2: **Initialize:** $Seq[1] = \langle 1 \rangle$
3: **for** $i \in [2, n]$ **do**
4: $\quad C_{min}[i] = \alpha f_{delay}(1, i)$
5: $\quad Seq[i] = \mathcal{I}\langle i \rangle$
6: $\quad$ **for** $j \in [1, i-1]$ **do**
7: $\quad\quad \mathcal{C} = C_{min}[i-j] + \alpha f_{delay}(i-j+1, i)$
8: $\quad\quad\quad + \min\{a_i - a_{i-j}, T\}$
9: $\quad\quad$ **if** $\mathcal{C} < C_{min}[i]$ **then**
10: $\quad\quad\quad C_{min}[i] = \mathcal{C}$
11: $\quad\quad\quad Seq[i] = \langle Seq[i-j], \mathcal{I}\langle j \rangle \rangle$
12: $\quad\quad$ **end if**
13: $\quad$ **end for**
14: **end for**

Specifically, we present a dynamic programming solution with the runtime of $O(n^2)$, where n is the length of the arrival sequence $\mathcal{A}$. The input to the algorithm is the arrival sequence $\mathcal{A}$. The output of the algorithm are arrays C_{min} and Seq. Here, $C_{min}[i]$ represents the optimal cost of the subsequence $\mathcal{A}_i = \langle a_1, \ldots, a_i \rangle$, and $Seq[i]$ is a binary sequence representing the decisions of OPT for the subsequence $\mathcal{A}_i$. If $Seq[i] = 1$, then OPT makes a grant at a_i, otherwise it waits for the next arrival. Also, C_{OPT} is given by $C_{min}[n] + T$.

Algorithm 1 shows the steps in OPT. As can be seen, it constructs the optimal grant sequence in a bottom-up fashion.

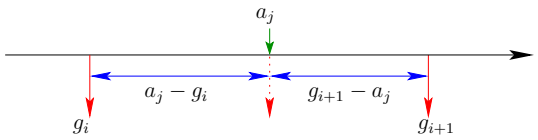


Fig. 3: Modified grant sequence in Lemma 1.

The main idea of the algorithm is that, for any arrival sequence, there is a grant right when the last request arrives, and the second to last grant can happen when any of the previous requests arrives. Specifically, OPT consists of a nested loop where each iteration of the outer loop finds the optimal solution for the smaller subsequence $\mathcal{A}_i$ and stores it in $C_{min}[i]$ and $Seq[i]$. Finally, the last iteration of the outer loop computes the optimal grant decisions for the arrival sequence $\mathcal{A} = \mathcal{A}_n$ using the results achieved in previous iterations. The notation $f_{delay}(j,i)$, for $j < i$, denotes the delay cost incurred by requests $\langle a_j, \ldots, a_i \rangle$ when they are granted at time a_i, which is given by $f_{delay}(j,i) = \sum_{k=j}^{i}(a_i - a_k)$. Also, $\mathcal{I}\langle l \rangle$ denotes a binary sequence of length l in which all elements are 0 except the last one, which is set to 1.

Lemma 1. *When $\alpha > 1$, OPT makes a grant for every request arrival.*

Proof. Assume that there exists an arrival at time a_j for which OPT does not make a grant and instead decides to wait for future request arrivals. Let g_i denote the last grant of OPT before a_j. Also, let g_{i+1} denote the first grant of OPT after a_j (see Fig. 3). If we modify the optimal grant sequence by adding a grant at a_j, the latency cost of the new sequence decreases by at least $\alpha(g_{i+1} - a_j)$. However, adding a grant at a_j changes the energy cost contribution of the interval $(g_i, g_{i+1}]$ from $\min\{g_{i+1} - g_i, T\}$ to $\min\{g_{i+1} - a_j, T\} + \min\{a_j - g_i, T\}$. As a result, the net increase in the energy cost can be characterized as follows:

$$\begin{aligned}\min\{g_{i+1} - a_j, T\} + \min\{a_j - g_i, T\} - \min\{g_{i+1} - g_i, T\} \\ \leq \min\{g_{i+1} - a_j, T\} \leq g_{i+1} - a_j\,.\end{aligned}$$

The first inequality holds simply because $a_j < g_{i+1}$. Given that $\alpha > 1$, the increase in energy cost is less than the decrease in latency cost, *i.e.*, $\alpha(g_{i+1} - a_j)$. Thus, the cost of the modified grant sequence is less than the cost of the optimal grant sequence, which is a contradiction. □

Interestingly, using a similar argument, it can be established that when $\alpha = 1$, there exist multiple optimal grant sequences and that granting at each arrival is one of them.

IV. Online Break-Even Algorithm

The Break-Even algorithm (BE) works as follows. Assume that the last grant was at time g_i and the algorithm has to decide when to make its next grant g_{i+1}. Let $E_{\text{BE}}(t)$ and $D_{\text{BE}}(t)$ denote the energy and delay cost incurred since the previous grant g_i up to time t. Then, BE makes a grant at time t for which $E_{\text{BE}}(t) = \alpha D_{\text{BE}}(t)$. Fig. 4 depicts an example illustrating the behavior of BE. The staircase curve shows the arrival of requests over time. It is easy to see that for any choice of t, the area of the resulted shaded region will be equal to the latency cost associated with a grant at time t. Let D denote the numerical value of this area. BE will choose the grant time t that satisfies the following equation:

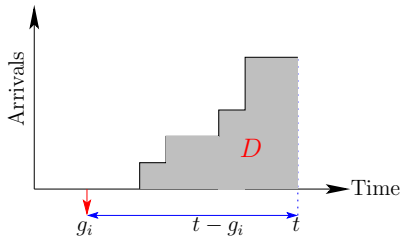


Fig. 4: An example depicting the delay cost of BE.

$$\min\{t - g_i, T\} = \alpha D\,. \tag{4}$$

The first grant is treated differently, *i.e.*, when there is no g_i. The algorithm makes its first grant at some time t that satisfies the equation $T = \alpha D_{\text{BE}}(t)$.

The algorithm BE can be implemented using timers. Specifically, upon the arrival of the l-th request at time a_l, a timer is set to time out at time $a_l + t$, where the value of t is computed based on (4). If the next request arrives before the expiry of this timer, the timer value will reset to a new value t by solving (4) with a new value of D. Otherwise, a time out at time $g_{i+1} = a_l + t$ will provoke a grant event in which case all the pending requests will be granted.

V. Analysis of the Break-Even Algorithm

As mentioned earlier, the main difficulty in determining the competitive ratio of BE is that the algorithm can make several grants in a short period of time and incur only a small energy cost. This implies that, an algorithm can make many grants and incur smaller energy cost compared to another algorithm that makes only a few grants. This is completely different from the problem setting considered in [8], [18], [21], where it is assumed that if an algorithm makes more grants then it incurs a higher energy cost.

In the reminder of this section, we focus on proving the following theorems.

Theorem 1. *The Break-Even algorithm achieves a competitive ratio of 4, that is, $C_{\text{BE}} \leq 4\,C_{\text{OPT}}$.*

Theorem 2. *The competitive ratio of 4 is tight, that is, there is an arrival sequence for which $C_{\text{BE}} = 4\,C_{\text{OPT}}$.*

A. Preliminaries

The main idea is to lower-bound the cost of OPT with respect to the cost of BE.

Lemma 2. *The cost of grant interval X_i is given by $C_{\text{BE}}(X_i) = 2\min\{X_i, T\}$.*

Proof. Recall that BE makes a grant when $E_{\text{BE}}(X_i) = \alpha D_{\text{BE}}(X_i)$. Thus, the cost of interval i is given by $C_{\text{BE}}(X_i) =$

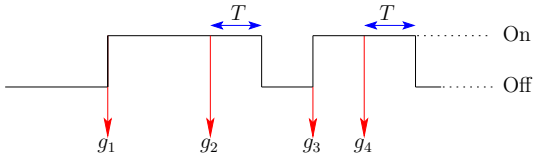


Fig. 5: Grants and radio state transitions of OPT.

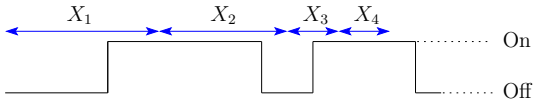


Fig. 6: Grant intervals of BE overlaid on radio states of OPT.

$2E_{\mathrm{BE}}(X_i)$. Based on the definition of the energy cost, we have $E_{\mathrm{BE}}(X_i) = \min\{X_i, T\}$. This completes the proof. □

Observation 1. *At least one request arrives during an interval $X_i = (g_{i-1}, g_i]$. If there is no arrival, then the algorithm does not make any grant at g_i as there is no request to grant.*

Observation 2. *Using Lemma 2, the cost of interval X_i satisfies the relations $C_{\mathrm{BE}}(X_i) \leq 2X_i$ and $C_{\mathrm{BE}}(X_i) \leq 2T$.*

B. Radio State Transitions

Let $\mathcal{G}_{OPT} = \langle g_1, \ldots, g_l \rangle$ denote the sequence of grants made by OPT. Recall that OPT knows the sequence $\mathcal{A}$ in advance, and hence can optimally space its grants in order to minimize its cost. *We make no assumption about the relation between the number of grants of* BE *and* OPT.

Fig. 5 depicts the grants of OPT and the corresponding state of the radio interface under this algorithm. At the beginning, the radio is off. During the execution of the algorithm, the radio may switch between the on and off states several times. Once the last grant is made, which happens when the last request arrives at time a_n, after a tail time T, the radio goes to the off state again.

Fig. 6 depicts the grant intervals of BE overlaid on the radio states under OPT. Two types of intervals can be identified:

1) Intervals that do not include any radio state transitions.
2) Intervals that include one or more radio state transitions.

The following subsections, analyze these two types of interval.

C. Intervals with No Radio Transition

Let X denote such an interval. There are two types of such intervals:

1) *Radio is on during interval X:* In this case, no matter how many grants OPT makes, it incurs at least the energy cost as its radio is on. Its delay cost could be zero, but it keeps the radio on for at least X amount of time. Thus,

$$C_{\mathrm{OPT}}(X) \geq X\,. \tag{5}$$

Based on Observation 2, it is obtained that,

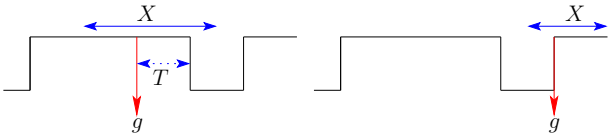
$$C_{\mathrm{BE}}(X) = 2\min\{X, T\} \leq 2X \leq 2C_{\mathrm{OPT}}(X)\,. \tag{6}$$

2) *Radio is off during interval X:* Following Observation 1, since BE makes a grant at the end of interval X, it means that at least one request has arrived during X. As the radio is off for OPT, we conclude that OPT incurs at least a delay cost equal to $D_{\mathrm{BE}}(X)$, as does BE. Therefore,

$$C_{\mathrm{OPT}}(X) \geq \alpha D_{\mathrm{BE}}(X)\,. \tag{7}$$

As discussed in the proof of Lemma 2, we have $C_{\mathrm{BE}}(X) = 2E_{\mathrm{BE}}(X) = 2\alpha D_{\mathrm{BE}}(X)$. Thus, the following relation is obtained,

$$C_{\mathrm{BE}}(X) = 2\alpha D_{\mathrm{BE}}(X) \leq 2C_{\mathrm{OPT}}(X)\,. \tag{8}$$



(a) First grant of OPT happens when the radio is on.

(b) First grant of OPT happens when the radio is off.

Fig. 7: Intervals with one or more radio transitions.

D. Intervals with One or More Radio Transitions

Let X denote such an interval. Consider the grants of OPT during the interval X. If there are no grants, then clearly $C_{\mathrm{BE}}(X) \leq 2C_{\mathrm{OPT}}(X)$ as OPT suffers from at least a delay cost equal to $D_{\mathrm{BE}}(X)$ because it does not make any grants during X. Thus, in the remainder of this subsection, we consider the case that OPT makes at least one grant during interval X.

Consider the first grant of OPT in X. As depicted in Fig. 7, there are two cases to be considered:

1) *The first grant happens when the radio is already on:* This case is depicted in Fig. 7(a). Since OPT has a grant when the radio is on, the radio remains on for at least a tail time T. Thus, $C_{\mathrm{OPT}}(X) \geq T$. It then follows that,

$$C_{\mathrm{BE}}(X) \leq 2\min\{X, T\} \leq 2T \leq 2C_{\mathrm{OPT}}(X)\,. \tag{9}$$

2) *The first grant happens when the radio is off:* This case is depicted in Fig. 7(b). In this case, the cost of OPT could be as low as zero if its first grant (denoted by g on the figure) happens right at the same time as the grant of X. Thus, the only relation that can be established in this case is,

$$C_{\mathrm{BE}}(X) \leq 2\min\{X, T\} \leq 2T, \tag{10}$$

and the ratio $C_{\mathrm{BE}}(X)/C_{\mathrm{OPT}}(X)$ is indeed unbounded. Let $\mathcal{X}^0_{\mathrm{BE}} \subseteq \mathcal{X}_{\mathrm{BE}}$ denote the set of all such intervals of BE.

The key idea is then to bound the cost of OPT over the interval set $\mathcal{X}^0_{\mathrm{BE}}$ rather than bounding its cost over individual intervals (where its cost could be zero). This is established during the proof of Theorem 1 presented next.

E. Proof of Theorem 1

Proof. We have the following relations for the cost of BE and OPT with respect to grant intervals $\mathcal{X}_{\mathrm{BE}}$,

$$C_{\mathrm{OPT}} = \sum_{X_i \in \mathcal{X}_{\mathrm{BE}}} C_{\mathrm{OPT}}(X_i) + T, \tag{11}$$

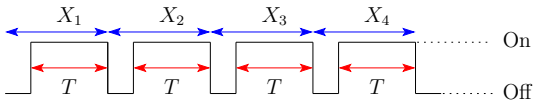


Fig. 8: Upper bound for $|\mathcal{X}^0_{\rm BE}|$.

$$C_{\rm BE} = \sum_{X_i \in \mathcal{X}_{\rm BE}} C_{\rm BE}(X_i) + T\,. \tag{12}$$

It was shown in the previous subsections that for every interval $X_i \in \mathcal{X}_{\rm BE}/\mathcal{X}^0_{\rm BE}$, we have, $C_{\rm BE}(X_i) \le 2C_{\rm OPT}(X_i)$. Also, for every interval $X_i \in \mathcal{X}^0_{\rm BE}$, we have, $C_{\rm BE}(X_i) \le 2T$. Therefore, it is obtained that,

$$\begin{aligned} C_{\rm BE} &= \sum_{X_i \in \mathcal{X}_{\rm BE}} C_{\rm BE}(X_i) + T \\ &= \sum_{X_i \in \mathcal{X}_{\rm BE} \setminus \mathcal{X}^0_{\rm BE}} C_{\rm BE}(X_i) + \sum_{X_i \in \mathcal{X}^0_{\rm BE}} C_{\rm BE}(X_i) + T \\ &\le 2 \sum_{X_i \in \mathcal{X}_{\rm BE} \setminus \mathcal{X}^0_{\rm BE}} C_{\rm OPT}(X_i) + \sum_{X_i \in \mathcal{X}^0_{\rm BE}} (2T) + T \\ &\le 2C_{\rm OPT} + 2T\,|\mathcal{X}^0_{\rm BE}|, \end{aligned} \tag{13}$$

where $|\mathcal{X}^0_{\rm BE}|$ denotes the cardinality of set $\mathcal{X}^0_{\rm BE}$. Thus, it is left to compute an upper bound for the term $|\mathcal{X}^0_{\rm BE}|$. Clearly, in the worst case, this term is upper bounded by $|\mathcal{X}_{\rm BE}|$ as depicted in Fig. 8. In this case, for every radio transition from the off to on state, there is an interval X_i. In this worst-case scenario, for every interval $X_i \in \mathcal{X}^0_{\rm BE}$, OPT incurs at least an energy cost equal to one tail time as it turns on the radio during the interval. It then follows that,

$$C_{\rm OPT} \ge |\mathcal{X}^0_{\rm BE}|\,T, \tag{14}$$

and, consequently,

$$|\mathcal{X}^0_{\rm BE}| \le \frac{C_{\rm OPT}}{T}\,. \tag{15}$$

Substituting the above inequality in (13) yields the following result,

$$\begin{aligned} C_{\rm BE} &\le 2C_{\rm OPT} + 2T|\mathcal{X}^0_{\rm BE}| \\ &\le 2C_{\rm OPT} + 2C_{\rm OPT} \\ &\le 4C_{\rm OPT}\,. \end{aligned} \tag{16}$$

□

F. Proof of Theorem 2

Proof. To show that the competitive ratio 4 is tight, it is sufficient to provide an example that attains this ratio. To this end, consider the scenario depicted in Fig. 9. Assume that $\alpha < 1$, *i.e.*, energy is more important than delay.

In this example, every time the radio is off, a large number of requests (depicted by a thick arrow) arrive in a short period of time, *i.e.*, a batch arrival, so that the online algorithm makes a grant immediately after the batch arrival. Recall that individual request delays are added together. Therefore, even though individual delays are small, a large number of them are added together to become equal to T, at which point BE makes its first grant. Since the radio is off, the cost of the first grant is given by its delay cost, which is equal to T.

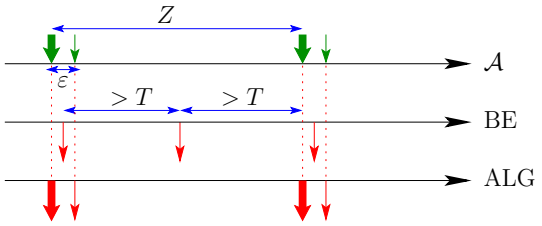


Fig. 9: Example for the lower bound.

A short time ε after this grant, another request arrives (depicted by a thin arrow) and then nothing arrives for a while. Algorithm BE waits for some time before making a grant for the second arrival. Since $\alpha < 1$, the wait time will be longer than T in order to satisfy the equation $E_{\rm BE}(\varepsilon + t) = \min\{\varepsilon + t, T\} = \alpha t$. For the second grant, the cost is the summation of the energy cost T and its weighted delay cost, which is also equal to T, for a total cost of $2T$. Once time T has passed after the second grant, the radio goes to the off state, which incurs the energy cost T due to the tail time. At this time, the same scenario is repeated again (*i.e.*, a large batch arrival immediately followed by a single arrival).

Next, consider the cost of OPT for the same scenario. We do not know how OPT behaves in this case, but we do know that *its cost is less than or equal to the cost of the algorithm that makes a grant as soon as any request arrives*. This algorithm is denoted by ALG on Fig.9. For this algorithm, its delay cost will be zero, and thus its total cost is given by the radio on time ε plus a tail time.

Let Z denote the time interval from when the first request in a batch arrives until the arrival of the first request of the next batch as depicted in Fig.9. Based on the above argument, it is obtained that,

$$C_{\rm BE}(Z) = T + 2T + T, \tag{17}$$

$$C_{\rm OPT}(Z) \le C_{\rm ALG}(Z) = \varepsilon + T\,. \tag{18}$$

The proof is established by noting that,

$$\lim_{\varepsilon \to 0} \frac{C_{\rm BE}}{C_{\rm OPT}} = 4\,. \tag{19}$$

□

G. Remarks On the Competitive Ratio

Notice that the competitive ratio of 4 for BE is in the *worst case*. On average, the algorithm performs much better as shown in Section VI. In practice, it is unlikely that a large number of requests arrive in a short period of time. For example, let us assume that the minimum inter-arrival time is τ. Consider the lower bound scenario described in the previous subsection. Let n denote the number of back-to-back request arrivals in a batch until BE makes a grant. Thus, the i-th request waits for time $(n-i)\tau$ until BE makes a grant. At the time of the first grant at time t, we have $T = \alpha D_{\rm BE}(t)$, which yields,

$$T = \alpha(0 + 1 + 2 + \cdots + n - 1)\tau, \tag{20}$$

and, consequently, $n \approx \sqrt{\frac{2T}{\alpha\tau}}$. In this scenario, the algorithm OPT incurs either a delay cost equal to T (if it does not make a grant), or an energy cost equal to $n\tau$ (if it grants every request), as there is some time between successive arrivals. As $T \geq n\tau$, we obtain that, $C_{\mathrm{OPT}} \geq n\tau + T$. It then follows that, $\lim_{\tau \to T/(2\alpha)} \frac{C_{\mathrm{BE}}}{C_{\mathrm{OPT}}} = 2$.

VI. Performance Evaluation

In this section, we first present model-driven simulation results to verify the accuracy of our results. Then, we present experimental results based on measurements on a smartphone to demonstrate the utility and performance of the proposed algorithm in realistic scenarios. In addition to BE and OPT, we have also implemented the *Default* algorithm, in which requests are granted as soon as they arrive.

A. Model-Driven Evaluation

In this part, we use a custom-developed discrete-event simulator to compute energy and delay costs under different algorithms. The input to the simulator consists of the weight factor α, the value of the tail time and the arrival sequence. Unless otherwise stated, the tail time is set to 200 ms, which is the value for the tail time of the Continuous Reception substate in LTE's Radio Resource Control (RRC) state machine [5].

1) Exploring Energy-Delay Tradeoff: The first experiment is performed using a sequence of 100 request arrivals, where the inter-arrivals are sampled from a normal distribution with mean 200 ms and standard deviation 80 ms. In this sequence, about 41% (59%) of the inter-arrival times are less (greater) than the tail time.

Figs. 10(a) and 10(b) represent the delay and energy cost for different values of α, respectively. It can be seen that by exploring the large parameter space of α, BE is able to provide different levels of energy savings depending on the user preference. Specifically, by increasing the weight α, BE achieves lower delay values at the expense of higher energy consumption. In our experiments, $\alpha = 10^{-4}$ and $\alpha = 10^{3}$ mark two ends of the spectrum where maximum energy saving (and delay reduction) are achieved at the expense of increased delay (and energy consumption). For example, $\alpha = 10^{3}$ results in 100% delay reduction compared to $\alpha = 10^{-4}$, while $\alpha = 10^{-4}$ brings about 98.9% energy saving compared to $\alpha = 10^{3}$.

Fig. 10(c) combines the previous two plots by showing the pairwise values of energy and delay along with their corresponding weight factors. We see a negligible influence of the weight change on the energy cost in the regimes where delay is more important than energy ($\alpha > 1$). Fig. 10(d) shows the average size of the bundles created by BE. It is observed that BE ia able to adjust its behavior based on the weight given to the delay cost. Specifically, the average bundle size decreases from 100 to 1 by increasing the delay weight from 10^{-4} to 10^{3}. For lower values of α, greater energy savings require a more aggressive aggregation policy, thus creating larger bundles. On the other hand, in scenarios where delay has higher weight, BE tends to avoid bundling and grants requests as soon as they arrive.

Table I compares the performance of BE and OPT in terms of the empirical competitive ratio achieved for different values of α. It can be seen that in this experiment, BE performs considerably better than the predicted worst-case competitive ratio of 4. Also notice that for $\alpha = 10^{3}$, the total cost of BE becomes equal to the cost of OPT. The reason is that in the extreme case of maximum delay reduction, both BE and OPT have identical behaviors as they grant requests as they arrive.

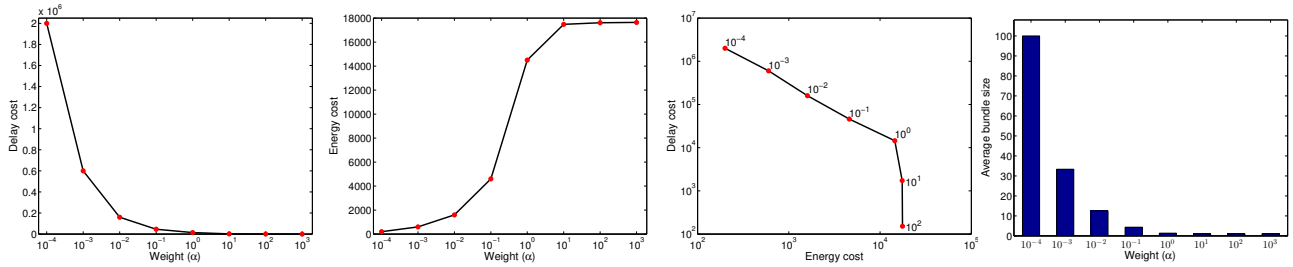
TABLE I: Empirical competitive ratio of BE.

α	$C_{\mathrm{BE}}/C_{\mathrm{OPT}}$	α	$C_{\mathrm{BE}}/C_{\mathrm{OPT}}$
10^{-4}	1.32	1	1.64
10^{-3}	1.30	10^{1}	1.97
10^{-2}	1.18	10^{2}	1.86
10^{-1}	1.17	10^{3}	1

2) Performance under Different Arrival Patterns: To study the behavior of BE under different arrival patterns, we consider the fluctuation level of the inter-arrival times. Similar to [22], we use the coefficient of variation (CV) to classify sequences of arrival times into groups of *low*, *medium* and *high* fluctuations. We conduct simulations with sequences characterized by CV $= 0.5$ (low fluctuation), CV $= 1.5$ (medium fluctuation) and CV $= 5$ (high fluctuation). In all sequences, inter-arrival times are sampled from a normal distribution with mean 200 ms.

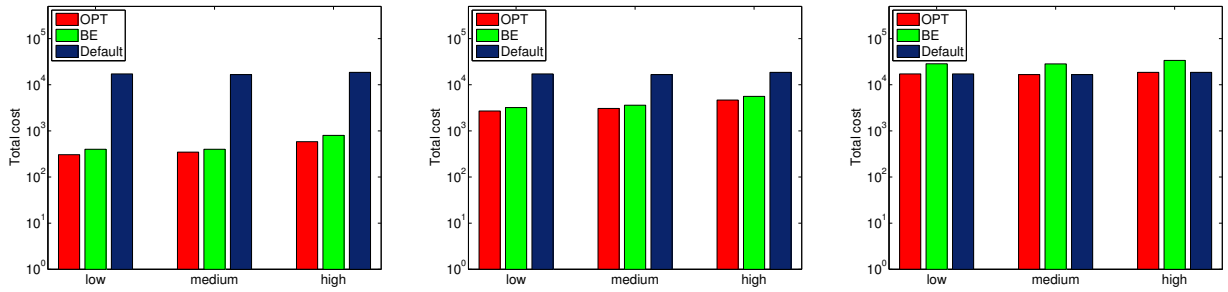
Fig. 11 compares the total cost of the three algorithms under varying fluctuation levels and weight values. It is observed that for a specific delay weight, the performance of BE changes depending on the characteristics of the arrival sequence. For example in Fig. 11(a) ($\alpha = 10^{-4}$), the cost of BE is 1.15 and 1.37 times the cost of OPT for sequences with medium and high fluctuation, respectively. Among all the considered scenarios, the ratio $C_{\mathrm{BE}}/C_{\mathrm{OPT}}$ ranges from 1.15 to 1.82, which is consistent with our analysis. We can also see that, in scenarios with higher weight for energy ($\alpha = 10^{-2}, 10^{-4}$), BE outperforms the Default algorithm. For example, in a setting with $\alpha = 10^{-2}$ and high fluctuation, BE results in 69.8% reduction in the total cost compared to Default. As Fig. 11 illustrates, across all α values, the lowest performance of BE is achieved when delay and energy have an equal weight ($\alpha = 1$). This conforms with Lemma 1 that in those regimes, it is optimal to schedule requests immediately on their arrival. As a result, deferring a request will only contribute to both energy and delay cost, and thus leading to a higher competitive ratio. It is, however, straightforward to modify BE to follow the behavior of OPT when $\alpha \geq 1$.

3) A Bursty Arrival Pattern: In this section, we verify that the competitive ratio of BE is indeed greater than 2. To this end, we generate an arrival sequence inspired by the example described in Subsection V-F. As Fig. 12 shows, this sequence is a repetition of a specific pattern, where a burst of requests (marked by a thick arrow) arrives and then after a short period of time, denoted by t_s, a single request arrives (marked by a thin arrow). Thereafter, a long interval, denoted by t_l, passes



(a) Delay cost (in ms). (b) Energy cost (in ms). (c) Energy-delay tradeoff. (d) Average bundle size.

Fig. 10: Performance of BE: By controlling α, different energy-delay tradeoffs can be achieved.



(a) $\alpha = 10^{-4}$ (b) $\alpha = 10^{-2}$ (c) $\alpha = 1$

Fig. 11: Comparing the performance of BE with OPT and Default under different fluctuation levels of request inter-arrival times.

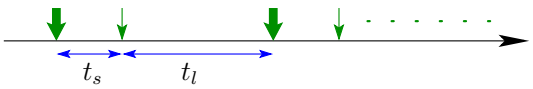


Fig. 12: A bursty arrival sequence: The thick and thin arrows indicate burst and single arrivals, respectively

before the arrival of the next burst. By following this pattern, we construct a sequence of 500 requests, where the size of each burst is uniformly distributed between 1 and 14. The short and long time intervals (t_s and t_l) are exponentially distributed with means 40 ms and 400 ms, respectively.

TABLE II: Empirical competitive ratio with bursty arrival pattern.

α	$C_{\text{BE}}/C_{\text{OPT}}$	α	$C_{\text{BE}}/C_{\text{OPT}}$
0.7	2.24	1.1	2.45
0.8	2.36	1.2	2.34
0.9	2.51	1.3	2.24
1	2.63	1.4	2.19

Table II tabulates the empirical competitive ratio of BE for values of α that result in a cost ratio greater than 2. We note that, while the cost ratio is greater than 2 in these scenarios, it is still consistent with the ratio 4. Interestingly, in our simulations with $\alpha = 1$, the cost ratio increases from 1.82 to 2.63 by changing the arrival pattern from the high fluctuation (described earlier) to the bursty pattern.

4) Effect of Tail Time: To study the effect of the tail time on the performance of BE, we use a sequence of 100 requests, where the inter-arrival times are sampled from a normal distribution with mean 300 ms and standard deviation 80 ms. We perform a set of experiments with three different values for the tail time (100, 200 and 300 ms). Fig. 13 depicts

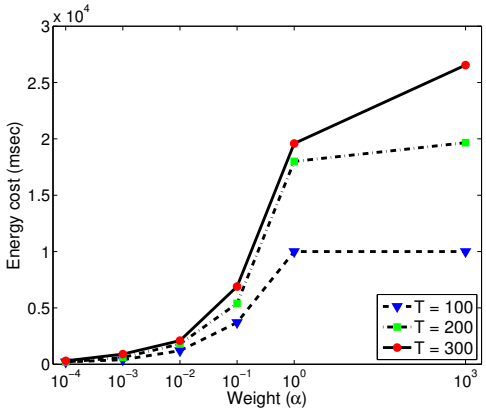
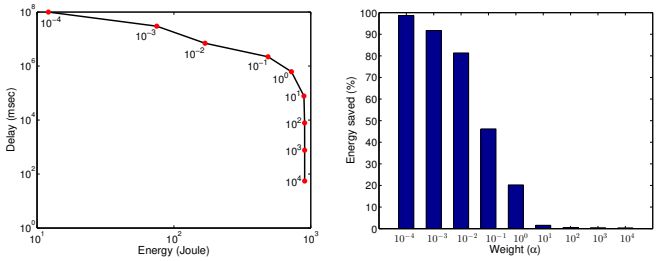


Fig. 13: Energy cost of BE with different tail times.

the energy cost for different values of the tail time as a function of the delay weight (α). It is observed that the energy-delay tradeoff and hence the energy cost is dependent on the value of the tail time. Specifically, decreasing the tail time leads to a lower energy cost in almost all cases (α values). For example, in the case of $\alpha = 10^3$, an energy cost improvement of about 62% can be achieved when changing the tail time from 300 ms to 100 ms. This is due to the fact that larger values of tail time tend to keep the network interface on for longer, which can cause higher energy consumption.

Fig. 13 also shows that the benefit of shorter tail times grows with increasing the delay weight. The reason is that, to reduce delay, a higher number of grants are needed, which in turn generates a higher number of inter-grant idle gaps. Therefore, the role played by the tail time becomes more significant when



(a) Energy-delay tradeoff. (b) Energy savings.

Fig. 14: LTE experiments on a Nexus smartphone.

delay has higher importance to the user. A similar argument can be made to justify the marginal benefits of shortening the tail time in settings with higher energy weight ($\alpha < 1$).

B. Smartphone Experiments on LTE

To assess the performance of BE under more realistic conditions, we also performed experiments on a Nexus smartphone. To measure the energy consumption of the radio interface, we used the AT&T ARO tool [1], which is configured with AT&T LTE network parameters. To conduct experiments on a smartphone, we developed an Android app that turns off the screen and performs HTTP transfers at user-specified times. Each *run* of this app uses a grant sequence file and a URL as input. It then repeatedly downloads the object referred to by the URL at the times specified in the grant file. We also installed a firewall app (AFWall+) on the phone to block all background traffic from OS services and other apps.

For the input sequence, we created a sequence of 100 requests with normal inter-arrival times of mean 10 seconds and standard deviation 5 seconds, based on the measurement results reported in [2] for popular news feed updates. Also, given that the operation of BE depends on the value of the tail time, we used 10 seconds as the tail value which is the default value in ARO's AT&T LTE profile for the inactivity timer of the RRC_CONNECTED state. Using the input sequence, and for different values of α, we run BE and Default and record their resulting grant times in separate files. We then feed those files to our Android app and measure the radio energy consumption of the device during each run.

Fig. 14(a) plots the pairwise energy-delay values of BE along with their corresponding α values. Notice that both Fig. 14(a) and Fig. 10(c) in previous section use the same delay measure which is the cumulative delay (in ms) incurred by all requests in the sequence. However, while Fig. 10(c) expresses energy in terms of the milliseconds spent in the high power state, here we present energy in terms of Joules. Fig. 14(a) illustrates a consistent behavior between simulation and real-world experiments as BE spans the broad spectrum of the energy-delay tradeoff. Also, our experiment with the Default algorithm (which only achieves a fixed tradeoff point) resulted in zero delay and an energy expenditure of 905.7 Joules. Fig. 14(b) plots the energy savings of BE compared to the Default algorithm for different values of α. We can see growing energy savings by increasing the relative importance of energy. Across all the values of α, the energy savings of BE can range between 0.4% and 98%. As an example, BE results in 20.3% energy reduction when delay and energy are equally important to the user.

VII. Conclusion

In this paper, we studied the problem of energy-aware request bundling on smartphones. We proposed an online algorithm for the problem and proved that it is 4-competitive. Our algorithm does not make any assumption about the traffic pattern or nature of applications. We then evaluated our algorithm using simulations and live experiments, which showed that, in realistic scenarios, the performance of the proposed algorithm is close to that of the optimal offline algorithm that knows the arrival sequence in advance. A possible extension of this work is to design a randomized version of our algorithm.

References

[1] F. Qian *et al.*, "Profiling resource usage for mobile applications: a cross-layer approach," in *Proc. ACM Mobisys*, 2011.

[2] N. Balasubramanian *et al.*, "Energy consumption in mobile phones: a measurement study and implications for network applications," in *Proc. ACM IMC*, 2009.

[3] S. Deng and H. Balakrishnan, "Traffic-aware techniques to reduce 3G/LTE wireless energy consumption," in *Proc. ACM CoNEXT*, 2012.

[4] J. Huang *et al.*, "A close examination of performance and power characteristics of 4G LTE networks," in *Proc. ACM MobiSys*, 2012.

[5] A. Nika *et al.*, "Energy and performance of smartphone radio bundling in outdoor environments," in *Proc. WWW*, 2015.

[6] M. Gupta *et al.*, "Energy impact of emerging mobile internet applications on LTE networks: issues and solutions," *IEEE Commun. Mag.*, vol. 51, no. 2, pp. 90–97, 2013.

[7] F. Qian *et al.*, "Periodic transfers in mobile applications: network-wide origin, impact, and optimization," in *Proc. WWW*, 2012.

[8] L. Xiang *et al.*, "Ready, set, go: Coalesced offloading from mobile devices to the cloud," in *Proc. IEEE INFOCOM*, 2014.

[9] X. S. Wang *et al.*, "Demystifying page load performance with wprof," in *Proc. USENIX NSDI*, 2013.

[10] F. Qian *et al.*, "Characterizing resource usage for mobile web browsing," in *Proc. ACM MobiSys*, 2014.

[11] Herrería-Alonso *et al.*, "Adaptive DRX scheme to improve energy efficiency in LTE networks with bounded delay," *IEEE J. Sel. Areas Commun.*, vol. 33, no. 12, pp. 2963–2973, 2015.

[12] A. Sivakumar *et al.*, "PARCEL: Proxy assisted browsing in cellular networks for energy and latency reduction," in *Proc. ACM CoNEXT*, 2014.

[13] L. Wang and J. Manner, "Energy-efficient mobile web in a bundle," *Computer Networks*, vol. 57, no. 17, pp. 3581–3600, 2013.

[14] F. R. Dogar *et al.*, "Catnap: exploiting high bandwidth wireless interfaces to save energy for mobile devices," in *Proc. ACM MobiSys*, 2010.

[15] M. A. Hoque *et al.*, "Poster: Extremely parallel resource pre-fetching for energy optimized mobile web browsing," in *Proc. ACM MobiCom*, 2015.

[16] M.-R. Ra *et al.*, "Energy-delay tradeoffs in smartphone applications," in *Proc. ACM MobiSys*, 2010.

[17] F. Mehmeti and T. Spyropoulos, "Is it worth to be patient? analysis and optimization of delayed mobile data offloading," in *Proc. IEEE INFOCOM*, 2014.

[18] A. R. Karlin *et al.*, "Dynamic TCP acknowledgement and other stories about e/(e-1)," in *Proc. ACM STOC*, 2001.

[19] S.-T. Hong and H. Kim, "Qoe-aware computation offloading scheduling to capture energy-latency tradeoff in mobile clouds," in *Proc. IEEE SECON*, 2016.

[20] A. R. Karlin *et al.*, "Competitive snoopy caching," *Algorithmica*, vol. 3, no. 1-4, pp. 79–119, 1988.

[21] D. R. Dooly *et al.*, "On-line analysis of the TCP acknowledgment delay problem," *Journal of the ACM*, vol. 48, no. 2, pp. 243–273, 2001.

[22] W. Wang *et al.*, "Dynamic cloud resource reservation via cloud brokerage," in *Proc. IEEE ICDCS*, 2013.