Aggregating Inter-App Traffic to Optimize Cellular Radio Energy Consumption on Smartphones

Swadhin Pradhan*, Sourav Kumar Dandapat†, Niloy Ganguly†, Bivas Mitra† and Pradipta De‡,
*CS Department, The University of Texas at Austin, USA
†CSE Department, Indian Institute of Technology Kharagpur, India
‡CS Department, SUNY Korea

***Abstract*—Cellular radio interfaces on smartphones consume a significant amount of battery power, specially with growing number of network centric applications. With high bandwidth cellular access links pushing the bottleneck to the network core, the risk of poor bandwidth utilization of the access link increases, which leads to energy wastage as the radio interface must stay active longer. In addition, small sized packet transmissions from applications wake up the interface frequently, but do not require the entire bandwidth capacity. In this work, we improve the radio usage by aggregating packet transmission from multiple applications. We introduce different time delays while transmitting packets from foreground and background applications such that user experience is minimally impacted. Through empirical observations, we determine the impact of different types of traffic on bandwidth utilization. Naive attempts to improve bandwidth utilization lead to increase in the number of packets missing the deadline for dispatch. With these observations, we propose a technique that balances the bandwidth utilization and deadline misses. Simulation driven experiments using synthetic traces and real trace based on application usage on Android based smartphones show energy gain of around 10% over other competing techniques.**

I. Introduction

Several network-centric applications that use the cellular radio interface of smartphones consume high energy [3]. In order to optimize the energy consumption, various energy saving techniques are implemented. For instance, in absence of communication activity, the network card switches to an intermediate low energy state, (instead of switching to idle state), to save the ramp up energy. However, presence of intermittent network request with low bandwidth demands, such as periodic synchronization with a server, leads to high number of state switches and thus resulting in significant amount of energy wastage [9]. In addition, it is expected that mobile broadband connectivity will continue to improve allowing higher last mile bandwidth [7]. If applications cannot fully leverage the high bandwidth access link to the base station due to round trip delay in the end-to-end path, then it also leads to under-utilization of the available bandwidth, leading to wasted energy. The CatNap system by Dogan et al. batches packets with tiny time gaps among them on the access router to allow the end device to sleep longer, and also better utilize the access bandwidth [8]. Decoupling the access links (connecting end devices to edge servers or base stations) from bottlenecked backbone links (connecting base stations to servers), can improve network throughput. Closer inspection of the network architecture reveals that (a) reducing the frequency of transitions between the low and high energy state and (2) full utilization of the available bandwidth, opens up the potential to prevent energy wastage. This paper proposes to leverage on this opportunity.

978-1-4244-8953-4/11/$26.00

In this paper, we aim to maximize the radio resource utilization by aggregating the packets across multiple applications. Higher packet aggregation requires that requests from different applications may need to be delayed to synchronize their transmissions. Each application is assigned a delay budget per network request within which a packet transmission from that app must be serviced. In general, background applications are more delay tolerant than foreground apps due to lack of user interaction [10]. The key contribution of the paper is to show that improving bandwidth utilization by coalescing packet transmission across applications at the cost of delaying the transmission of packets from selected applications can lead to significant energy gains. We draw the motivation based on empirical measurements on different packet traces that show the inter relationship between bandwidth utilization and packets missing their deadlines (as a result of delaying packets). We use the observation to derive a function that optimizes the bandwidth utilization (energy efficiency) without adversely affecting the deadline misses (user experience).

Our experimental results show that significant energy gains can be achieved by efficient utilization of the available bandwidth in high power mode by batching packets across foreground and background applications without impacting the user experience. We use the standard energy model of 3G network card, as described in [2], to compare the energy gains among competing scheduling strategies while operating in two different modes - (a) Fast Dormancy, and (b) Fast Dormancy with 5 sec tail timer. Simulation experiments using synthetic traces, as well as, real trace (packet trace collected on Android based smartphones) show that up to 10% network card energy can be saved compared to other competing techniques.

The rest of the paper is organized as follows. Section II presents the prior work. Section III discusses necessary models and assumptions. Section IV presents the technique for aggregating packets for transmission. Section V shows the evaluation of the algorithm based on simulation experiments using traffic traces generated based on the traffic models, while in Section VI we use traffic traces from normal usage on a

smartphone. We conclude in Section VII.

II. Related Work

Energy efficiency techniques for 3G radio primarily explored two directions - tail time adaptation, and traffic aggregation.

The design of Tail Optimization Protocol (TOP) by Qian et al. leverages the Fast Dormancy feature and usage pattern to predict long idle periods when the card can be immediately switched off, thereby eliminating any tail time wastage [15]. A further generalized approach for predicting network idle time is proposed by Kim et al. [11]. RadioJockey analyses program behavior to determine communication spurts to enable faster switch off [1]. Further improvement is achieved by better utilization of cellular bandwidth in Bartendr [16] and LoadSense [6], where free channel time and gateway server load is used to trigger transmissions. Cui et al. has proposed an online scheduling algorithm, called PerES [19], which is closest to our goals. PerES attempts to conserve energy while ensuring good transmission experience for users. In this work, the problem is formulated as an energy delay optimization problem and solved using Lyapunov optimization framework. We take the path of empirical observations to design our solution.

A series of work, starting with Tailender, has observed that utilizing the low bandwidth channel during tail time for transmission can significantly save energy. Tailender showed that by clubbing traffic one can reduce energy by 35% for email applications, 52% for news feeds and 40% for web search [3]. Xu et al. focuses on the behavior of email applications on smartphones, and proposes techniques to reduce energy cost of email sync by 50% [18]. Backgrounds jobs on smartphones lead to unnecessary wakeup of 3G radio, and often these jobs are periodic in nature [13]. Techniques for batching periodic jobs are proposed in [5]. TailTheft uses the tail time for pre-fetching and delayed transfers, showing the benefits on per application behavior [12]. Going beyond single app scheduling, in this work we show that significant energy savings can be derived by scheduling packets across apps. For a multi-tasking smartphone user this is a natural usage behavior.

Catnap system is proposed as a solution for better utilization of high speed access link in the context of high speed Wi-Fi, and slow broadband Cable/DSL [8]. We adopt the network design principle of CatNap. Although Catnap concludes that efficacy of their approach is limited to applications with large packets, we show in this work that even in presence of small packet sizes, significant energy savings can be gained. Our work also takes into account the typical multi-tasking behavior of smartphone users, and considers how to ensure user satisfaction while minimally disrupting foreground job.

III. Models and Assumptions

This section presents the models and assumptions used in formulating the problem and evaluating the proposed solution.

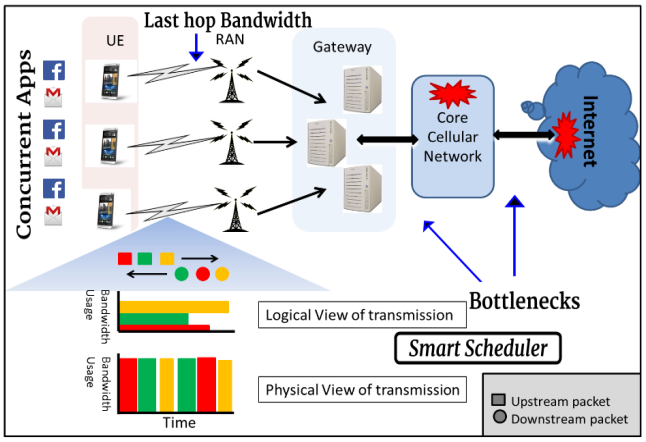

Fig. 1. Simplified topology illustrating packet transfer over high bandwidth access link in cellular network. Multiple small packets can be aggregated during upload, which can reduce the need to switch off the 3G network card.

A. Network Model

We assume a cellular network model, similar to 3G UMTS or 4G LTE, with high bandwidth access links. Fig. 1 illustrates the network model. With fast access links, the bottleneck in the network can shift towards the core cellular network, and the Internet backbone. As a result, although an upstream request packet from the smartphone to the Radio Access Network (RAN) may have low latency, the response packet corresponding to the request may take long to come back. If the gap between the request and response packets are longer than a few seconds, the radio network card will be put to sleep to save energy wastage on the smartphone. However, a state transition also incurs a significant energy wastage, if the card wakes up frequently [14].

The orchestration of packet transmission, as shown in Fig. 1, prevents a state transition by scheduling other request packets during the interval between a request-response sequence. For a single app's request-response sequence, without card state transition, the goodput of the access link is the total bytes exchanged over the request-response completion time. Thus the goodput is much lower than the throughput of the access link. However, if packets from other apps are scheduled for transmission in the gap, then the goodput increases, as illustrated in the logical view of transmission. The physical view of transmission shows the packet exchange at the physical layer of the card, while the logical view illustrates the idea of aggregating the packets from different applications.

An alternative way to improve the utilization of the channel is to delay packets from different applications and synchronize their transmission. The available bandwidth can be more effectively utilized instead of waking up the network interface frequently for sending individual packets which may not utilize the available bandwidth entirely. In essence, the bandwidth wastage is minimized by introducing an application specific delay before a packet is transmitted.

B. Application Model

Mobile applications can be categorized as,

Background Services (A_B) : These services run as background processes and generate traffic periodically to Internet hosts. Such services include news service or software updater. Data consumption and network request pattern of background services (A_B) generally follow a specific pattern, e.g. CNN news app pulls news snippets periodically.

Foreground Applications (A_F) : These are the applications with which the user directly interacts with. Some of the popular examples are streaming application like *Youtube*, gaming application like *Free Online Games*, browser application like the *Dolphin* browser. It is required to model the network traffic generated by these applications, which are often running as foreground processes on a smartphone. URL visits or revisits in mobile browser follow varied pattern depending upon different types of users [17], whereas inter-arrival time of sending or receiving emails follows a power-law with $\alpha = 1$ [4]. These two applications represent more general category of foreground applications where the distribution of network requests are either random or follow a well-known distribution. Such foreground applications are termed as (a) $A_F - Random$, where the distribution of requests is random and (b) $A_F - Normal$, where the distribution follows a Power law.

C. Traffic Model

Traffic model simulates network workload of mobile applications, and is represented using the following parameters.

Application Sync Timing (Ψ) : This parameter decides the data sync interval for background services. For background services, we consider a constant value of Ψ, configured by a user or system default. Values of 15 mins and 30 mins are assumed. For foreground process, the sync timing is not applicable since we assume that user interaction drives the communication to the servers.

User Interaction Timing (Υ) : This parameter emulates user interaction pattern with the foreground application. As [4] suggests that human decision mainly follows the pattern of high activity for a small duration followed by inactivity for a longer duration, viz. a power-law distribution, therefore we emulate foreground application activity by choosing Υ as a power-law distribution with α = 1. For background service A_B, Υ is constant since there is no user interaction.

However, some interaction like browsing behavior on smartphones has been reported recently to follow no specific pattern [17]. To mimic this behavior, Υ value is selected randomly within a range. For background services A_B, we choose a constant Υ value.

Data Transmission Size (Λ) : The request packet size varies across applications [8]. This parameter models the network request size to reflect application characteristics. For example, gaming applications are likely to have smaller request packets than that of streaming applications.

Bandwidth Demand (Δ) : The bandwidth demand is considered different for different application types. For instance streaming has the highest bandwidth demand while gaming has the lowest. The bandwidth demand for background applications is considered low.

Slack duration (Φ) : Slack duration captures the maximum allowable delay before a request packet from an application must be transmitted to maintain quality of service. Foreground applications will typically have low Φ value compared to background applications since the user is directly interacting with the foreground application.

D. Energy Model

We model the energy consumption of two different operating modes of 3G interface card - *Fast Dormancy (FD)* and *Fast Dormancy with 5 second Tail Timer (TT)* [15]. A 3G interface state machine has 3 states - $IDLE$ where the radio is switched off and no energy is consumed, $CELL\_DCH$ (or DCH), which is the high throughput state and the radio consumes high energy, and $CELL\_FACH$ (or $FACH$), which is an intermediate low power state allowing low bandwidth communications [2]. In TT mode, the network card transitions into $FACH$ state and stays there for 5 seconds before transitioning into $IDLE$ state if a packet is not scheduled for transmission. In FD mode, the card can transition directly into $IDLE$ state. Due to difference in state transitions the energy consumptions are different for the two modes.

The total energy consumption of the 3G network card can be computed using the formula, Total Energy Consumption = $C_R + C_M + C_D$, where C_R is the ramp up energy to switch from idle state to high energy state, C_M is the energy consumed to keep the card powered on in the high or low energy state, C_D is the data transmission energy. In Fast Dormancy (FD) mode, there is no tail energy required, however, if the packet transmissions are frequent then the ramp up energy cost is high to switch from $IDLE$ to DCH. C_M component is high in FD mode, while it is half in Tail Timer (TT) mode. But, if we increase the tail timer, the total energy consumed will be higher in TT mode due to tail energy wastage. However, in both modes, the data transmission energy, C_D, is proportional to the data size transmitted.

The values for each parameter is based on values reported in [3], [14], and are as follows. In FD mode, $C_R = 3.5$ J, $C_M = 0.8$ J, and $C_D = 0.25x$ J. While in TT mode, C_R is negligible if the transition is from $FACH$ to DCH, otherwise it is 3.5J, $C_M = 0.62t$ J, and $C_D = 0.25x$ J, where x denotes the amount of KiloByte transferred and t is tail time duration spent in $FACH$.

IV. Scheduling Protocol

In this section, we formally define the problem and present our scheduling protocol. We assume a set of concurrent applications request network resource intermittently. To maintain quality of service, the network requests from each application should be serviced within a fixed time duration, which we defined earlier as *slack duration*. The slack duration varies depending on whether the application is a foreground or a background process. The scheduler must intelligently schedule the network requests such that more energy is saved while

transmitting packets within the slack duration. Two constraints cannot be violated: (a) requests from the same application cannot be triggered simultaneously (b) total bandwidth consumption by all the scheduled requests should be less than the available channel bandwidth.

Formally, we denote the j^{th} network request from the application A_i as P_{ij}. The arrival time of a request P_{ij} is r_{ij} and the request is serviced at x_{ij}. Each request P_{ij} has a slack duration, f_{ij}, which means that request P_{ij} can be scheduled latest by r_{ij} + f_{ij} to prevent missing the deadline. Service duration of request P_{ij} is denoted by d_{ij} and bandwidth required for the network request is represented as b_{ij}. The communication channel has a finite bandwidth B which implies that at any time instant total bandwidth consumption of all the scheduled requests must not exceed B. All the scheduled requests from same application must be sequential in nature; that means until a request P_{ij} is served, P_{ik} (where $k > j$) can not be scheduled, i.e. $x_{ik} \geq x_{ij}$ + d_{ij}.

The network is designed using two queues to hold request packets, *wait queue*, and *run queue*. Since a network request may not be scheduled for transmission immediately on arrival, such packets are queued in *wait queue*. Packets ready for transmission are taken out of *wait queue* and inserted into *run queue*. A request in *wait queue* is marked for insertion into *run queue* when the slack duration is expiring. The scheduler checks the *wait queue* at the next scheduling point for marked requests, dequeues the request and inserts into the *run queue*, from where it is selected for transmission. The scheduler executes every time the *run queue* changes state due to insertion/deletion of requests.

A. Scheduling Algorithm

The scheduler uses a formula, F, to decide whether to move a request from *wait queue* to the *run queue*. Defining the value of F is one of the main research questions. This section explains the selection of F. In order to define a principled F, we have to take into account both optimized energy usage and reasonable system performance. Hence, our approach towards designing the scheduling strategy is to first study empirically the variations of two key metrics - (i) bandwidth utilization or wastage, and (ii) deadline miss for packets.

B. Insights from Preliminary Experiments

We use three synthetic traces with different traffic characteristics to observe the behavior of the two metrics. The three traffic traces are generated as: (a) Type-1: a collection of network requests from foreground applications but with low bandwidth demand for each packet, (b) Type-2: a collection of low bandwidth background traffic coupled with intermittent (high bandwidth) foreground network requests, (c) Mixed: combines the characteristics of Type-1 and Type-2 traces.

In the simplest case, we assume that no intelligent scheduling is applied, and a network request is serviced (send to run queue) as soon as it arrives. Fig. 2 shows the variations of bandwidth wastage and deadline miss for the three traces. Note that for Trace-1 since the bandwidth requirement per packet is low, therefore, the bandwidth wastage is higher, but fewer deadline miss occur. It is reversed for Trace-2, while it lies in between for the Mixed trace. The key observation is that there are two competing factors bandwidth wastage and deadline miss which should be accommodated while defining F. More formally, we can combine them additively to reach a balance between the two factors. Therefore F can be defined as follows.

$$F = \beta \cdot Bandwidth\_wastage + (1 - \beta) \cdot Experience\_user \quad (1)$$

where β is a normalizing constant.

Data: Invoke scheduler on *run queue* state change
if *Trigger received* **then**
  for *each marked requests in* wait queue **do**
    if *request is compatible with all requests in* run queue **then**
      Compute F;
      if $F > 0$ **then**
        Move request from *wait queue* to *run queue*;
      else
        Remove mark;
      end
    end
  end
end

Algorithm: Balanced Scheduling: Defining F using Eqn. 5 leads to balanced scheduling

.

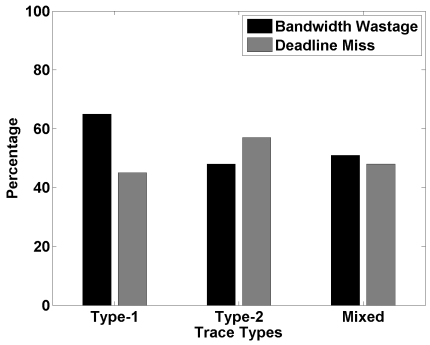

Fig. 2. The variation of bandwidth wastage and deadline misses when packets are scheduled as soon as they arrive for three traces with different characteristics.

C. Bandwidth Wastage

Let the maximum available bandwidth be B and let there be n requests denoted by P_i, $i = 1 \cdots n$, each starting at x_i time, running for a duration of d_i and requiring a bandwidth b_i [1]. Then Bandwidth Wastage (BW) is the total available bandwidth during the active period of at least one of the requests minus the bandwidth utilized by these requests. It

[1]without loss of generality we are dropping the j subscript for ease of understanding

can be defined as,

$$BW = B \times T - \sum_{i=1}^{n} b_i \times d_i, \;\; T = Max(x_i + d_i) - Min(x_j), \;\; \forall (i,j) ... \tag{2}$$

Assume a packet transmission is delayed using the available slack duration, which allows the scheduler to aggregate multiple packets together for transmission. If delaying makes better utilization of bandwidth compared to sending it as soon as it arrives, then the term $BW_1 - BW_2$ gives the bandwidth utilization efficiency, where BW_1 and BW_2 denotes bandwidth wastage by delaying and not delaying packets respectively. By normalizing the term, we get the term,

$$Bandwidth\_wastage = \frac{BW_1 - BW_2}{Max(BW_1, BW_2)} \tag{3}$$

D. User Experience

Three factors need to be considered while optimizing user experience - (a) a deadline (r_{ij} + f_{ij}) should not be missed and (b) process the request not too late but (c) delay the request if it helps in aggregating a number of application requests together. Let us say the request P_{ij} appears at time instant r_{ij} and has slack duration f_{ij}. Let us also assume that all the network requests presently in the run queue will be served completely by E (finish time). We assume that if E is very close to r_{ij}, it is prudent to delay the request and wait for other requests to appear for better parallelization - the probability of which decreases as the value of E - r_{ij} increases. Consequently we formalize through the following formula,

$$Experience\_user = \frac{E - \frac{r_{ij} + (r_{ij} + f_{ij})}{2}}{Max(E, \frac{r_{ij} + (r_{ij} + f_{ij})}{2})} \tag{4}$$

E. Normalization factor (β)

In the next set of empirical measurements, we carefully looked at the impact of each term in Eqn. 1 in determining the value of F such that we can select a value for the normalization parameter, β. The value of the normalization parameter, β, should be such that across all traces the impact of each term is roughly the same. With that aim we check the average number of times each term contributes in making the value of F positive. Since at β = 0.9, the contributions of the two terms are roughly equal, therefore, β value is set to 0.9.

F. Balanced Scheduling Protocol

Finally, for different traffic characteristics, one may need to give priority to either bandwidth utilization or deadline miss. This is incorporated by introducing a scaling parameter, α, to skew the importance of the two terms, as follows.

$$F = \alpha \cdot \beta \cdot Bandwidth\_wastage + (1-\alpha) \cdot (1-\beta) \cdot Experience\_user \tag{5}$$

where α varies between 0 and 1.

The α value should be ideally chosen such that the bandwidth wastage and deadline can be optimally traded off. As shown in Fig. 3, the optimal point where the bandwidth wastage and deadline misses are optimally balanced occurs under different α values.

V. Synthetic Trace Based Evaluation

In this section, we compare the performance of Balanced Scheduling technique against similar methods for improving energy efficiency of 3G radio interface using model based synthetic traces.

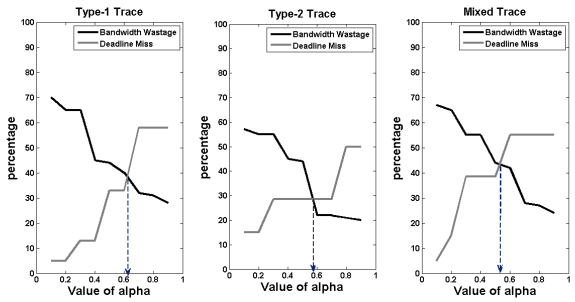

Fig. 3. The graph shows that for different traffic types the optimal trade off between bandwidth wastage and deadline miss occurs at different points.

A. Experimental Setup

The simulation experiments are driven by traffic workloads generated using the models defined in Section III. We model network traffic in three scenarios, where the foreground application is a gaming (Quizup), streaming (YouTube) or a browsing application. In each scenario there are several background applications, like news feeds or software updaters, which generate traffic.

TABLE I
Foreground and Background App Parameters used in Synthetic Trace Generation

App Type	Sync Time (s) Ψ	UI Time (s) Υ	Data Tx Size (KB) Λ	Bandwidth Demand (KBps) Δ	Slack Duration (s) Φ
A_B	900,1800	5,10,15	3,5 [9]	10	5-7
A_F-Normal	NA	Power-law [4]	3,5 [8]	5,15,40	2
A_F-Random	NA	2-20 [17]	3-50 [8]	20	2

1) Parameter Values: Choice of parameter values determines representative workloads. Values of different parameters are as shown in Table I. The capacity of the access link is set to 50 KBps in DCH state.

According to [8], three different packet sizes characterize all network transmissions from smartphone applications. Application-wise we can map this to gaming (small sized packet transmission), browsing (medium sized packet transmission) and streaming (large sized packet transmission), leading to different bandwidth requirement. We model these applications according to the application model in Section III-B. Bandwidth demand is set to 5 KBps for gaming app, 15 KBps for browsing app , and 40 KBps for streaming app. $A_F - Normal$ traffic models the gaming and streaming foreground apps, while $A_F - Random$ traffic models browsing. **Gaming** and **Streaming** scenarios comprise of one foreground app, modeled as $A_F - Normal$ traffic and bandwidth demand from Table I, and three background apps. **Browsing** scenario comprises of one foreground app, modeled as $A_F - Random$ traffic with 15 KBps bandwidth requirement, and three background apps. There are 3 background application adding traffic in all scenarios. Trace duration is 1 hr.

2) Evaluation Metrics: Performance evaluation tracks two key metrics - energy saved, and number of packets missing deadlines. Specific metrics are,

Energy Consumption per KB: This metric captures total energy spent to transmit one KiloByte of data over the network interface. Energy consumed is calculated using energy model in Section III-D. Alternately, we also report *Energy Saved* in comparison to a naive FIFO packet scheduling.

Deadline Miss: Deadline miss is captured as proportion of requests which have missed their deadline. It is expressed as, $\frac{Total\_Number\_of\_Deadline\_Miss}{Total\_Number\_of\_Requests} * 100$. This metric reflects the user experience.

The results related to energy consumption are further dissected using the following metrics,

Radio On Time: It captures the radio on time as a fraction of total data transmission duration.

State Switch Rate: The state switch rate counts the number of times per unit time the radio changes state - from $IDLE \rightarrow DCH$, and $DCH/FACH \rightarrow IDLE$.

3) Competing Scheduling Techniques: We compare the Balanced Scheduling technique against three different techniques - TailEnder, Tail Optimization Protocol (TOP), and Performance-aware Energy Scheduler (PerES). We present implementation details of these techniques.

TailEnder: $TailEnder$ [3] uses threshold based tail time prediction by considering deadlines of packets of an application. In principle, it delays packets as-long-as-possible without affecting user experience.

We extend $TailEnder$ [3] to prevent tail energy wastage across multiple applications, instead of original design aimed at single application. This naive implementation introduces per application separate queues which are serviced according to the $TailEnder$ heuristics using a simple round robin order.

PerES: Performance-aware Energy Scheduler or PerES models cross application energy-delay tradeoff as an optimization problem and applies Lyapunov optimization framework. It assumes that the wireless channel bandwidth is variable.

The implementation depends on choice of several parameters. The chosen parameter values, based on [19], are as follows. δ value of SVA is 0.001, application preference weights are 1/10 (for foreground application) and 1/100 (for background application), θ value is taken as 10, wireless signal as taken uniform randomly from the range of $-50\ dBm$ to $-110\ dBm$, and application details are according to Table I.

TOP: Tail Optimization Protocol (TOP) reduces tail energy wastage by predicting the application behavior [15]. Since this paper claims that tail time can be predicted with 60% accuracy, therefore, for 60% of the uniformly randomly chosen application traces, we assume that TOP is aware of the packets apriori. The remaining parameters are based on Table I.

B. Experimental Results

1) Choice of α value: The goal of this experiment is to select a value of α that is suitable for a specific trace. We plot the variation of three metrics - Energy Saved compared to FIFO packet scheduling, Deadline Miss, and Bandwidth Wastage - for the three scenarios. Fig. 4 shows the results for the three cases with varying α values.

Note that with increasing α value energy saved increases since the bandwidth wastage reduces. But deadline miss increases with increasing α since importance of deadline miss reduces according to Eqn. 5. Since the two factors are conflicting, α value can be learned by collecting statistics on more traces. But in this work, we do not focus on the optimal selection of α. Instead we show that even a suitable choice of α can lead to noticeable gains. Based on Fig. 4, we choose the α value to be 0.3, 0.4 and 0.5 for gaming, streaming and browsing scenarios respectively. In the following experiment results, we have used these α values for each setting.

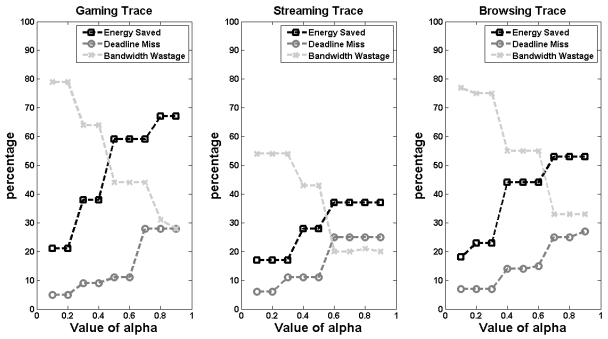

Fig. 4. The impact of variation of α on 3 traces with different characteristics - gaming, streaming and browsing, are shown.

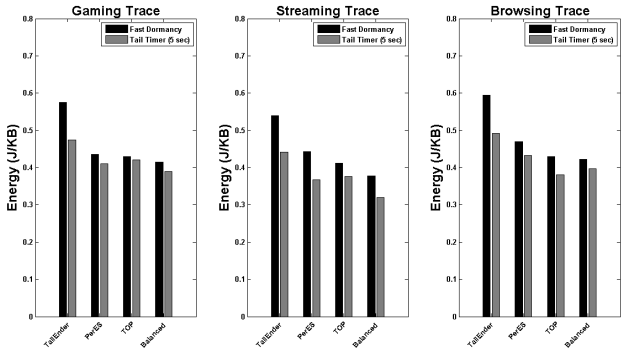

Fig. 5. Comparing energy usage of competing scheduling strategies in Fast Dormancy and Tail Timer, with 5 sec tail time, modes. We compare three scenarios - gaming, streaming and browsing for the chosen α values.

2) Energy Gain: In this experiment, we focus on how much energy is saved using our proposed balanced scheduling. We compare energy consumption of balanced scheduling against competing techniques, described in Section V-A3 in both Fast Dormancy (FD) and Tail Timer (TT) mode of operation. Fig. 5 shows the results for three different traffic traces.

For gaming and browsing, Balanced scheduling performs similar to PerES and TOP, but better than TailEnder. These two scenarios represent interactive foreground applications with relatively smaller request size, and reduced opportunity for delaying a request. Hence the gains are marginal compared to the other techniques. Note that in streaming, as the packets are larger in size, and higher delay in transmission can be tolerated, there is more opportunity for aggregation. Therefore, Balanced Scheduling performs much better than the other techniques.

The energy consumption in TT mode is less than that of the FD mode. When network traffic is sufficiently high, then in TT mode the card transitions from $FACH$ to DCH state, thereby expending less energy compared to a transition from $IDLE$ to DCH in FD mode. The lower energy spent in each transition, and often the low number of transitions, as verified with more experimental results later, leads to better energy savings in TT mode.

3) Deadline Miss: In our technique we introduce an application specific variable slack duration to each request. Ideally a packet should be transmitted within this slack such that user experience is not impacted. In this experiment, our goal is to study the impact of increasing number of background processes on the number of packets that miss the deadline. Lower deadline miss implies better

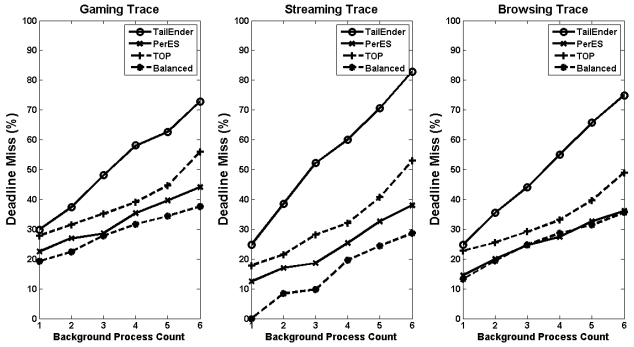

Fig. 6. Comparison of percentage of deadline miss for competing scheduling strategies. For Balanced scheduling we use the chosen α values.

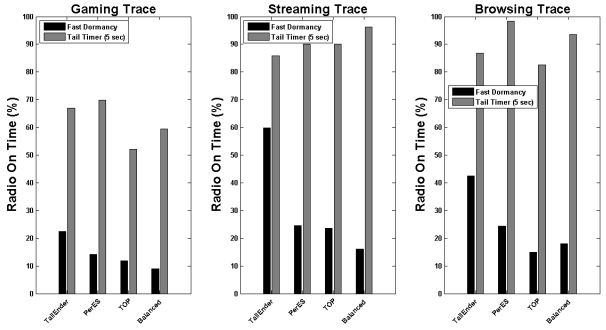

Fig. 7. Comparing the radio ON time for each competing scheduling strategies while operating in Fast Dormancy and Tail Timer, with 5 sec tail time, modes. We compare three scenarios - gaming, streaming and browsing for the chosen α value.

user experience. The results record the deadlines missed by packets from foreground process since this number directly impacts the user experience. Fig. 6 shows the results of deadline miss percentage for all the competing techniques, and for three different scenarios, where the α values are same as the ones to obtain the energy consumption results.

There are several observations based on the results. First, Balanced scheduling performs better than the other schemes in terms of missed deadlines. PerES and TOP perform better than TailEnder since these two techniques are designed for cross application scheduling, unlike TailEnder which have been extended for cross application scenario. Second, with increasing background process count, deadline miss increases. The background processes start consuming the bandwidth although their slack duration is higher than foreground processes. This leads to more foreground requests missing their deadline. Third, in Browsing scenario PerES and Balanced scheduling has similar deadline miss, however, in other two scenarios, Balanced does better.

4) Insight into Gains: This section looks deeper into the reasons for the energy gains. There are two key factors - (i) longer the radio stays on, more energy is consumed, and (ii) increasing the number of state transitions leads to more energy consumption due to ramp up energy. These two factors conflict with each other. Hence in the following experiments, we study the contribution of these two factors in energy consumption.

Fig. 7 shows the percentage of radio on time in two operating modes, FD and TT, for three scenarios of gaming, streaming and browsing, and for all the competing techniques. From the figure we observe that in FD mode, the radio on time is proportional to the request size. In gaming scenario, since the request sizes are small, therefore, DCH mode is less utilized giving lower radio on time. Radio on time increases for Streaming scenario. Similar argument holds for the TT mode, but the additional time spent in $FACH$ mode gets added to the radio on time.

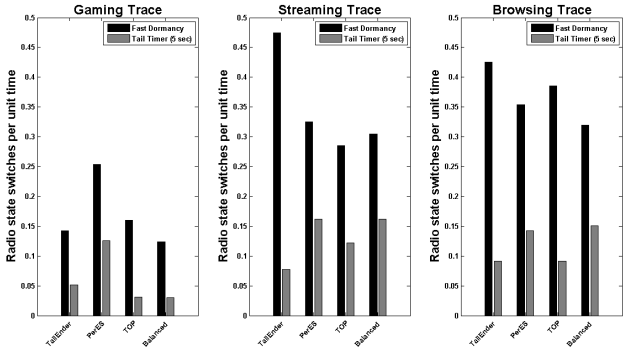

Fig. 8. Comparing the number of times per unit time the radio interface switches state. We compare the metric for the competing techniques and on traces from three scenarios - gaming, streaming and browsing, where the α values for each setting is pre-determined.

Fig. 8 shows the number of times the radio switches state per unit time. Higher number of switches implies that ramp up energy must be spent to switch state. According to the energy model (Section III-D), in FD mode ramp up energy is much higher than that in TT mode, if the card switches from $FACH$ to DCH state. From the results, we can observe that FD mode leads to higher number of switches. Since in FD mode, the state transition is triggered within 2 seconds of the end of transmission, therefore, more switches occur than in TT mode. Lower number of transitions in TT mode contributes to lower energy consumption.

Based on the results from Fig. 7 and Fig. 8, and comparing the energy consumption results in Fig. 5, we can infer that lower number of state switches contributes more significantly towards energy savings than radio on time. Hence TT mode saves more energy than FD mode, as shown in Fig. 5. The result can be further explained by observing that in FD mode radio on time is much lower than TT mode. The higher radio on time in TT mode is due to additional time spent in $FACH$ state. However, the energy spent in $FACH$ state is half of that spent in DCH state.

VI. Real Trace Based Evaluation

This section shows the results of evaluating Balanced Scheduling on real traces from a rooted Android based Samsung Galaxy S3 GT19300 smartphone. We gather the network traffic trace for 1 hour duration where the user is using the browser. We collected the packet trace using tcpdump, and map the ports to process identifiers in order to differentiate among traffic from each application. Wherever necessary, we used parameter values as shown in Table I.

Fig. 9 shows the energy usage of Balanced scheduling compared to the other techniques. Energy used by Balanced Scheduling is about 10% less than the competing techniques. The performance gain is lower compared to that of simulation results due to the nature of the real trace. The network traffic is grouped in nature since a foreground app spawns multiple background threads which trigger network requests at the same time. Such grouping limits the scope of aggregating traffic and reduce state transitions. Fig. 10 shows the corresponding deadline miss. Balanced Scheduling performs better than other techniques in this aspect.

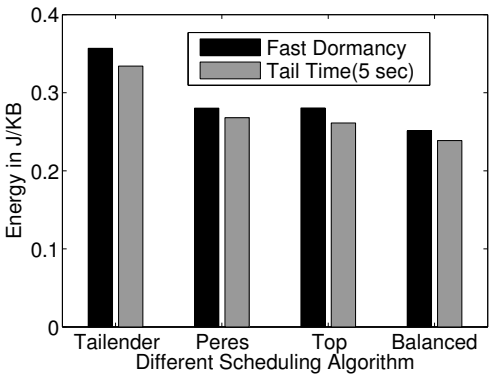

Fig. 9. Showing energy required to transfer one KB data across different scheduling algorithms on collected real trace (browsing). α for *Balanced* scheduling is chosen as 0.5.

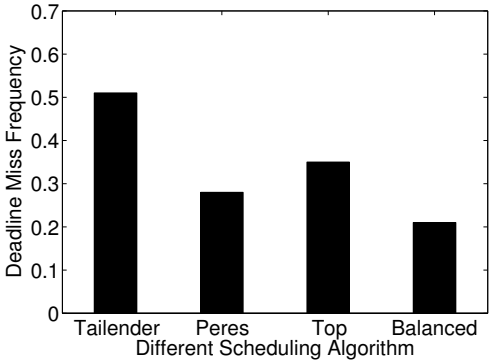

Fig. 10. Showing deadline miss frequency across different scheduling algorithms on collected real trace (browsing). α for *Balanced* scheduling is chosen as 0.5.

VII. Conclusion

Optimizing energy consumption of cellular radio interface is important in presence of growing number of network centric applications on smartphones. In this work, we show that aggregation of packets *across applications* leads to better utilization of available bandwidth during the high power state of the 3G network card. In order to aggregate packets, we introduce the idea of variable delay for network requests from foreground and background applications, which ensures better user experience. Simulation results show that close to 10% energy can be saved using intelligent scheduling without adversely affecting user experience. An important take-away from this work is that reducing number of state transitions of the network interface can save more energy than optimizing utilization of the tail period of the card.

VIII. Acknowledgment

This work is partially supported by the grant given by Vodafone IIT KGP Centre of Excellence in Telecommunications (VICET) and ITRA (Media Lab Asia). This research is supported by the MSIP (Ministry of Science, ICT and Future Planning), Korea, under the "IT Consilience Creative Program" (NIPA-2013-H0203-13-1001) supervised by the NIPA (National IT Industry Promotion Agency)

References

[1] 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 *MobiCom 2012*.

[2] Comparing lte and 3g energy consumption. https://developer.att.com/developer/forward.jsp?passedItemId=11900006.

[3] N. Balasubramanian, A. Balasubramanian, and A. Venkataramani. Energy consumption in mobile phones: A measurement study and implications for network applications. In *IMC 2009*.

[4] A.-L. Barabasi. The origin of bursts and heavy tails in human dynamics. *Nature*, 435:207, 2005.

[5] M. Calder and M. K. Marina. Batch scheduling of recurrent applications for energy savings on mobile phones. In *SECON*. IEEE, 2010.

[6] A. Chakraborty, V. Navda, V. N. Padmanabhan, and R. Ramjee. Coordinating cellular background transfers using loadsense. In *Proceedings of the 19th Annual International Conference on Mobile Computing & Networking*, MobiCom, 2013.

[7] Cisco visual networking index: Global mobile data traffic forecast update 20132018. http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white_paper_c11-520862.html.

[8] F. R. Dogar, P. Steenkiste, and K. Papagiannaki. Catnap: Exploiting high bandwidth wireless interfaces to save energy for mobile devices. In *MobiSys 2010*.

[9] H. Falaki, D. Lymberopoulos, R. Mahajan, S. Kandula, and D. Estrin. A first look at traffic on smartphones. In *IMC 2010*.

[10] J. Huang, F. Qian, Z. M. Mao, S. Sen, and O. Spatscheck. Screen-off traffic characterization and optimization in 3g/4g networks. In *IMC 2012*.

[11] Y. Kim and J. Kim. Personalized diapause: Reducing radio energy consumption of smartphones by network-context aware dormancy predictions. In *HotPower 2012*.

[12] H. Liu, Y. Zhang, and Y. Zhou. Tailtheft: Leveraging the wasted time for saving energy in cellular communications. In *MobiArch 2011*.

[13] 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 the 21st International Conference on World Wide Web*, WWW, 2012.

[14] F. Qian, Z. Wang, A. Gerber, Z. M. Mao, S. Sen, and O. Spatscheck. Characterizing radio resource allocation for 3g networks. In *IMC 2010*.

[15] F. Qian, Z. Wang, A. Gerber, Z. M. Mao, S. Sen, and O. Spatscheck. Top: Tail optimization protocol for cellular radio resource allocation. *ICNP 2010*.

[16] 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 the 2010 International Conference on Mobile Computing and Networking*, 2010.

[17] C. Tossell, P. Kortum, A. Rahmati, C. Shepard, and L. Zhong. Characterizing web use on smartphones. In *SIGCHI 2012*.

[18] F. Xu, Y. Liu, T. Moscibroda, R. Chandra, L. Jin, Y. Zhang, and Q. Li. Optimizing background email sync on smartphones. In *MobiSys 2013*.

[19] C. Yong, S. Xiao, X. Wang, M. Li, H. Wang, and Z. Lai. Performance-aware energy optimization on mobile devices. In *INFOCOM 2014*.