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

Swadhin Pradhan*, Sourav Kumar Dandapat*, Niloy Ganguly*, Bivas Mitra* and Pradipta De†,
*CSE Department, IIT Kharagpur, India
†CS Department, SUNY Korea.

***Abstract*—Surging popularity of network centric apps for smartphones is driven by cloud computing. Many of these apps run as background services, waking up intermittently to synchronize with the servers. This triggers frequent small sized request packets that activate the radio interface. It has been noted that frequent card activation wastes battery due to high switching energy. Hence the challenge is to maximize the radio resource utilization on each card activation. Two factors contribute to low utilization. First, apps are not synchronized to wake up simultaneously. Second, high speed cellular access links push the bottleneck to the network core leading to low bandwidth utilization of the access link.**

I. Introduction

As the landscape of smartphone based apps are evolving, several studies have investigated the traffic on smartphones [1]. One of the observations form the work is that most smartphone data transfers are small, with median size of 3KB only. Typically these small transfers are non-overlapping in time, and wake up the radio resource, like 3G network card, for every communication leading to high energy usage. To reduce switching tail time is used, however tail time also consumes significant energy. Reducing tail time energy wastage has been addressed primarily by (a) dynamic adjustment of the tail time timer by observing traffic patterns [2], and (b) using the tail time for transmissions [3]. In order to fill the tail time with transmissions, the classical approach is to aggregate packets from a single application either by delaying packets, or reorganizing computation and communication leading to higher batching efficiency. However, our observation is that aggregating packets from multiple applications can leverage even higher benefits. Second, the packets are typically of small size, specially for the background services. Hence batching packets from a single app may be insufficient to utilize the high bandwidth access links in emerging cellular networks, like LTE. Interleaving packets from multiple apps may lead to better utilization of the radio resource, specially in high bandwidth $CELL\_DCH$ state.

In this paper, our goal is to maximize the radio resource utilization in $CELL\_DCH$ state by aggregating packets across multiple apps. Higher packet aggregation requires that requests from different apps may need to be delayed to synchronize the transmissions. Each app is assigned a delay limit within which a packet transmission request from that app must be serviced.

In summary, the key contributions of this work are,

- Proposes a practical solution to enable batching packets across multiple apps by defining app-specific variable delay limits.

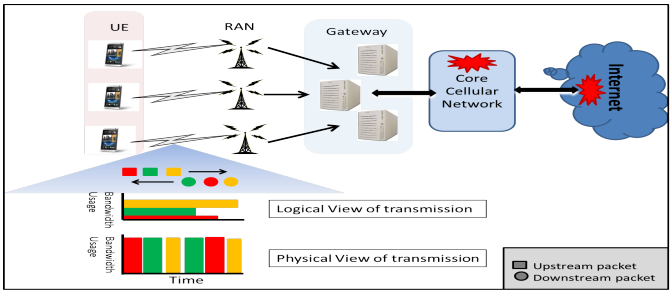


Fig. 1. Simplified topology illustrating packet transfer over high bandwidth access link in cellular network. Multiple small packets can be aggregated during upload, which can reduce the need to switch off the 3G network card.

- Shows the benefits of inter application batch scheduling in Fast Dormancy, and Fast Dormancy with a tail time operating mode.

II. Models and Assumptions

A. Network Model

We assume a cellular network model, similar to 3G UMTS or 4G LTE, where the access links are supposed to be of high bandwidth. Fig 1 illustrates the network model through a simplified diagram. As the access links are becoming faster, the bottleneck in the network is shifting 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. This gap can be utilized if enough requests are batched from applications running in the system.

B. Application Model

Mobile applications can be broadly categorized as background services and foreground application. Background services (A_B) run in the background on smart phones and sync with server periodically -news feeder, weather update etc. Foreground application does have a front end and it is normally interactive. Network usage of foreground application (A_F) heavily depends on application type and user interaction pattern.

C. Traffic Model

Network activity of a mobile application depends upon the sync timing, user interaction pattern, and data consumption during syncing or interaction. According to [4], users' interaction pattern (Υ) follow a power law. We choose higher value of

bandwidth demand (Δ) for streaming apps where lower value is considered for background services. We have considered another parameter namely flexibility (Φ) which announces allowed time that a network request from that app can wait.

D. Energy Model

We model the energy consumption using the conventional three-state ($CELL\_DCH$, $CELL\_FACH$, $IDLE$) 3G network card operating model and corresponding energy values for state transitions [5]. For calculating energy usage in 3G card, we consider the equation, Total Energy Consumption = $C_R + C_D + C_T$, where C_R is the ramp up energy ($IDLE$ to $CELL\_DCH$), C_D is the data transmission energy, and C_T is the tail energy (in $CELL\_FACH$) [5]. Moreover, we have adopted two different strategies [2] in 3G data transmission, namely *Fast Dormancy (FD)* and *Fast Dormancy with 5 second Tail Timer (TT)*. In FD, tail-time (t) is considered zero and for TT, tail-time (t) is considered as five seconds. We have assumed that, in FD, the demotion from $CELL\_DCH$ to $IDLE$ state happens instantly bypassing $CELL\_FACH$ state without losing any energy.

III. Scheduling Strategies

In this section, we define the problem and propose different versions of online algorithms and one offline algorithm.

A. Problem Description

Key concept: Let us assume that a set of applications, running in parallel, request for network resource intermittently. Moreover, depending on application (background/foreground), a flexibility or slack time is allowed to schedule each packet. The task of the scheduler is to correctly schedule each application request, exploiting the flexibility provided by the slack time, such that the maximum number of requests can be served simultaneously; this will eventually maximize the network card usage.

B. Complexity of the Problem

A simplified case of the above problem arises under infinite bandwidth assumption. An optimal solution to this problem can be derived, as shown later. However, imposing the bandwidth constraint turns the problem into a variant of bin packing optimization which is NP-hard.
The reduction is achieved through the following transformation of NP-hard version of bin-packing [6] optimization problem to this problem. Let us partition the entire traffic duration into unit sized time intervals and also modify the bandwidth demand b_{ij} of request P_{ij} to the fraction b_{ij}/B (= b'_{ij}), where B is the bandwidth limit of the device. Now, we can map the items of bin-packing problem to the network requests and unit sized bins to the unit bandwidth demand. Hence, the goal of bin packing to find the minimum number of bins leads to our aim to achieve minimum duration of network usage.

C. Scheduling Algorithms

In this section, we present three scheduling algorithms - (a) *Lazy Scheduling*, (b) *Early Scheduling* (c) *Balanced Scheduling*, each of which can be selectively triggered by an implementation of traffic aware scheduler, as explained before. Implementation of traffic aware scheduling is beyond the scope of this work.

1) Lazy Scheduling: The wait queue sends a trigger to the scheduler when it has at least one request (say P_{ij} from application A_i) reaching its deadline

On receiving the trigger, scheduler checks if the run queue is empty (i.e. network card is not in $CELL\_DCH$ mode). If empty, it removes the request P_{ij} from the wait queue, along with the sequence of successive compatible requests appearing behind P_{ij} and places all these requests in the run queue. On the other hand, if the run queue is not empty, it waits till it empties.

The two other scheduling algorithms function similar to *lazy scheduler*, except when the run queue is not empty.

2) Early Scheduling: If the run queue is not empty, scheduler checks wait queue to find any compatible request which can be scheduled. If it finds any such request in the wait queue, scheduler removes it from wait queue and places it in the run queue.

3) Balanced Scheduling: If there are some requests in the run queue, the scheduler first checks their compatibility in a way similar to *Early Scheduler*. If a request is present, *Balanced Scheduling* strategy computes a benefit function to decide placement of the compatible requests in the run queue. This benefit function essentially captures the following two criteria (a) amount of bandwidth wastage and (b) deadline miss probability.

D. Offline Scheduler

In this algorithm, it is assumed that the global information about the applications (request arrival time, service time, bandwidth requirement) are available a priori; in that aspect, this scheduling approach is offline. This algorithm is split into two components; first part drops the bandwidth constraint and only aims to maximize the number of applications which can be served in parallel adhering to the allowed slack time assigned to each application. Here, the only constraint we have is that two requests from the same application can not be scheduled at the same time. Subsequently, we incorporate the bandwidth constraint and finalize the solution.

A schedule of a set of network requests is best when collective time of all scheduled requests is minimum. Let us assume that there are two network requests R_A and R_B from two different applications A nad B respectively. Without loss of generality, let's assume that R_A arrives before R_B. If we can schedule both the requests at the same time then that would be the best scheduling for these two requests where the collective time taken would be max of total execution time(R_A, R_B). However, if R_A can be delayed and scheduled with R_B, depends on allowed flexibility of application A and

gap of arrival time of R_A and R_B. With dynamic programming approach we can find out the best schedule for a set of requests. Let us assume that the formulation described above provides us an optimal schedule for a set of requests S_i at time t_i. Next, we select a (possibly proper) subset of requests S_u^i from this set S_i respecting the bandwidth constraint, which essentially means that the total bandwidth consumption of the requests in set S_u^i is limited to the total available bandwidth B. The leftover requests $(S_i - S_u^i)$, if any, will be suitably shifted (forward/backward) to a point where bandwidth violation can be avoided.

IV. Evaluation

In this section, we evaluate the performance of batch scheduling techniques using simulation experiments.

A. Experimental Setup

The simulation experiments are driven by synthetic traces representing smartphone traffic. We generate traffic from one foreground app which the user is using, and multiple background services using parameters as mentioned in Table I. In order to mimic different usage scenarios in the foreground, such as interactive app (like gaming where bandwidth demand is assumed 5 KBps), streaming app (like YouTube where bandwidth demand is assumed as 40 Kbps), and browsing (bandwidth demand is 15 KBps), we generate traffic from three different combinations of apps. We set the total capacity of the access link to 50 KBps in $CELL\_DCH$ state.

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	10	5-7
A_F-Normal	NA	Power-law	3,5	5,15,40	2
A_F-Random	NA	2-20	3-50	20	2

B. Evaluation Metric

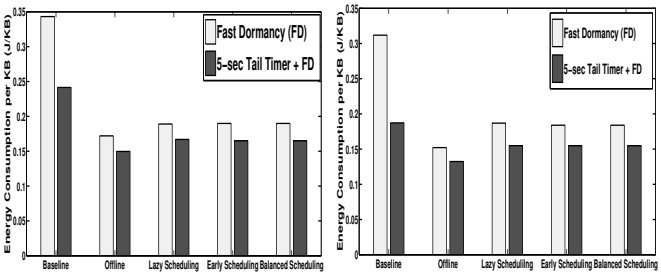
Energy Consumption per KB ***(E_{KB}):*** This metric captures energy spent to transmit one KiloByte of data over the network interface. It considers ramp up energy, transmission energy and tail energy.

We also define a ***Baseline Scheme*** for data transmission, where a network packet is serviced immediately as it arrives.

C. Energy Gains

Fig. 2(a) and Fig. 2(b) show the energy consumed to transmit per KiloByte of data across all the entire foreground and background traffic for Gaming and Streaming scenarios respectively.

In FD mode, online schemes consumes $\frac{1}{2}$ - $\frac{2}{3}$ energy compared to baseline scheme where offline algorithms consume half enegy compared to baseline scheme. In TT mode, energy consumption reduces significantly for baseline scheme as due to tail time it reduces switching. Due to batchinh of network requests in online and offile schemes, network bursts are well separated and could not not reduce much switching from tail time. However, online algorithms still perform better in TT mode compared to baseline scheme.



(a) Gaming Scenario (b) Streaming Scenario

Fig. 2. Comparison of Energy Consumption per KB of data transmission for different schemes in Gaming and Streaming scenario.

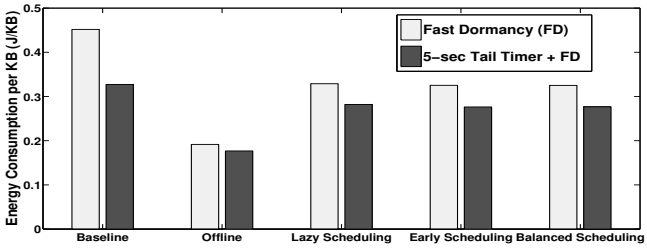


Fig. 3. Comparison of Energy Consumption for different schemes in Browsing scenario.

Fig. 3 shows the results for energy consumption per KB of data transmission compared across all the scheduling schemes in browsing scenario. In FD and TT mode, online algorithms save 25% and 10% energy respectively compared to baseline scheme where offline algorithm saves more than 50% and 30% energy respectively in FD and TT mode.

V. Conclusion

An important observation from this work is that better utilization of available bandwidth in $CELL\_DCH$ state leads to significant energy gains, both in Fast Dormancy and Fast Dormancy with Tail Time operating modes. The insight can augment other approaches in 3G energy optimization.

References

[1] H. Falaki, D. Lymberopoulos, R. Mahajan, S. Kandula, and D. Estrin, "A first look at traffic on smartphones," in *Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement*, IMC, 2010.

[2] F. Qian, Z. Wang, A. Gerber, Z. M. Mao, S. Sen, and O. Spatscheck, "Top: Tail optimization protocol for cellular radio resource allocation," *20th IEEE International Conference on Network Protocols (ICNP)*, 2010.

[3] N. Balasubramanian, A. Balasubramanian, and A. Venkataramani, "Energy consumption in mobile phones: A measurement study and implications for network applications," in *Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement Conference*, 2009.

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

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

[6] M. R. Garey and D. S. Johnson, *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., 1990.