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

Swadhin Pradhan\textsuperscript{1}, Sourav Kumar Dandapat\textsuperscript{2}, Bivas Mitra\textsuperscript{2}, Niloy Ganguly\textsuperscript{2} and Pradipta De\textsuperscript{3}.

\textsuperscript{1}UTAustin, USA  \quad \textsuperscript{2}IIT Kharagpur, India  \quad \textsuperscript{3}SUNY, Korea
Outline

- Introduction
- Balanced Scheduling Protocol
- Evaluation Setup and Traces
- Results and Takeaways
- Conclusion and Future Works
Introduction

- Display, Network, and CPU are main components of Energy Drain. [Mittal et. al., MobiCom ‘13]

- Poorly written apps can sap 30% to 40% of a phone’s battery. [Mahajan et. al., IMC ’09]

- Network intensive applications are increasing (~69% of the apps are cloud based).

- Different background services running intermittently and waking up the network card for a small duration. [Qian et. al., WWW ’12]
Cellular Radio Energy Model

Total Energy Consumption = CR + CD + CT, where CR is the ramp up energy (IDLE to CELL DCH), CD is the data transmission energy, and CT is the tail energy (in CELL FACH).
Total View of App Connectivity

Last hop Bandwidth

Bottlenecks

Smart Scheduler

Logical View of transmission

Physical View of transmission
Idea: Cross Application Traffic Aggregation

Network Usage Intervals

App1

App2

App2 with Stalling

Effectively Network Radio is On for this entire Interval

Effectively Network Radio is On with our Strategy

Slack that does not hurt user’s experience
Problem Objective

- Scheduling all network requests using minimum energy without hurting user’s experience.

- Multi-objective optimization problem
  - Minimum Energy $\Rightarrow$ **Best utilization of bandwidth** (Side-effect: Lower Switching Frequency).
  - User’s Experience $\Rightarrow$ Request should be served within deadline.
Problem Constraints

- Depending on expected response time of application, a \textbf{flexibility or slack time} is allowed to schedule each packet.

- Requests from the \textbf{same application cannot be triggered simultaneously}.

- \textbf{Total bandwidth consumption} by all the scheduled requests should be \textbf{less than the available channel bandwidth} (We consider \textit{constant last hop bandwidth}).
Approach Intuition: Deciding Function

- If a request is delayed then there is potentially more opportunity of batching.

- If a request is delayed much, it may miss deadline.

- So, we need to develop a function to decide at certain time if a request should be scheduled or should be delayed further.
Terminology

- $A_i$: $i^{th}$ application
- $A_{ij}$: $j^{th}$ request of $i^{th}$ application
- $r_{ij}$: Arrival time of $A_{ij}$
- $x_{ij}$: Scheduling time of $A_{ij}$
- $f_{ij}$: Slack time of $A_{ij}$
- $d_{ij}$: Service duration of $A_{ij}$
- $ft$: Finish time of all requests in run queue
Deciding Function (F)

\[ F = \beta \cdot \text{Bandwidth\_wastage} + (1 - \beta) \cdot \text{Experience\_user} \]

Where \( \beta \) is normalizing constant.

\[ F = \alpha \cdot \beta \cdot \text{Bandwidth\_wastage} + (1 - \alpha)(1 - \beta) \cdot \text{Experience\_user} \]

Where \( \alpha \) is factor to give priority over other
F : Bandwidth Wastage Component

Bandwidth_wastage = \frac{BW1 - BW2}{\text{Max}(BW1, BW2)}

BW1 : Bandwidth Wastage Before Scheduling
BW2 : Bandwidth Wastage After Scheduling
F : User Experience Component

Do not schedule if $ft$ lies here Wait for more requests

Schedule if $ft$ lies here

Experience$_{user} = \frac{ft - r_{ij} + (r_{ij} + f_{ij})}{2} \frac{Max(ft, \frac{r_{ij} + (r_{ij} + f_{ij})}{2})}{2}$
Balanced Scheduling Protocol

- There are two queues:
  - *Running Queue* has all the running requests served by Cellular Radio.
  - *Waiting Queue* has all the pending requests.

- Requests are put into wait queue as soon as they arrive.

- Pushed to run queue when Deciding Function (F) is positive.
Experimental Setup

- Application Types (MobiSys '12)
  - Gaming (Short Bursts)
  - Browsing (Medium Bursts)
  - Streaming (Large Bursts)
Experimental Setup

- Synthetic Trace Generation tuning parameters
  - User Interaction Timing (Power Law)
  - Data Transmission Size (Power Law with set of sizes)
  - Bandwidth Demand (Fixed Demand per App)
  - Slack Duration (Fixed per application type)
Experimental Setup

Switching Strategies

- Fast Dormancy.

- Fast Dormancy with Fixed Tail Timer.

![Diagram](attachment:image.png)

**Ramp Up Energy (CR)**

**Total Energy**

\[ \text{Total Energy} = \text{CR} + \text{CD} \]

**Data Transmission Energy (CD)**

**Tail Energy (CT)**

Instantly

**IDLE**

No Power
No Bandwidth

**CELL_DCH**

High Power
High Bandwidth

**CELL_FACH**

Low Power
Low Bandwidth

**Tail Time**

**Tail Energy (CT)**

**Data Transmission Energy (CD)**

**Ramp Up Energy (CR)**

**Total Energy**

\[ \text{Total Energy} = \text{CR} + \text{CD} + \text{CT} \]

**IDLE**

No Power
No Bandwidth

**CELL_DCH**

High Power
High Bandwidth

**CELL_FACH**

Low Power
Low Bandwidth
Real Trace Collection

- Collected using ARO (AT&T) tool, tcpdump, and ps.
- Samsung Galaxy S3 GTI9300 (Rooted).
- One hour Browsing Trace from a user.
- Applications are differentiated through port mapping.
Evaluation Metrics

- **Energy Consumption per KB**: Total energy spent to transmit one KiloByte of data.

- **Deadline Miss**: Proportion of requests which have missed their transmission deadline.

- **State Switch Rate**: Number of times per unit time the radio changes state - from IDLE to DCH, and DCH/FACH to IDLE.

- **Radio On Time**: Radio on time as a fraction of total data transmission duration.
Alpha Value Tuning

Good Trade-off between Energy and User Experience
Competing Scheduling Techniques

- **TailEnder**: Uses threshold based tail time prediction by considering deadlines of packets of an application.

- **PerES**: Performance-aware Energy Scheduler or PerES models cross application energy-delay tradeoff as an optimization problem and applies Lyapunov optimization framework.

- **TOP**: Tail Optimization Protocol reduces tail energy wastage by predicting the application behavior.
Energy Consumption per KB (~10%)
Deadline Miss

Gaming Trace

Streaming Trace

Browsing Trace

Deadline Miss (%) vs. Background Process Count for different traces and policies.
Switching

**Gaming Trace**
- Fast Dormancy
- Tail Timer (5 sec)

**Streaming Trace**
- Fast Dormancy
- Tail Timer (5 sec)

**Browsing Trace**
- Fast Dormancy
- Tail Timer (5 sec)
Radio On Time

**Gaming Trace**
- Fast Dormancy
- Tail Timer (5 sec)

**Streaming Trace**
- Fast Dormancy
- Tail Timer (5 sec)

**Browsing Trace**
- Fast Dormancy
- Tail Timer (5 sec)
Energy Consumption in Real Trace
Deadline Miss in Real Trace

![Bar graph showing deadline miss frequency across different scheduling algorithms: Tailender, Peres, Top, Balanced.](chart.png)
Takeaways

- Around 10% better than PerEs and TOP in Energy Gain wise, but far better from TailEnder.

- Percentage of Deadline Misses for Foreground App remains satisfactory.

- Reducing number of state transitions of the network interface can save more energy than optimizing utilization of the tail period of the card.
Future Works

- Extensive and large scale real trace based evaluation to validate the simulation based results.

- Building middleware which will run our aggregation strategy across applications.

- Extension and implementation of in other elements like sensors, GPS etc.

- Building a Application network activity recorder tool which can be installed without rooting.
Thank you