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

Swadhin Pradhan[1], Sourav Kumar Dandapat[2], **Bivas Mitra**[2], Niloy Ganguly[2] and Pradipta De[3].

[1]UTAustin, USA **[2]IIT Kharagpur, India** **[3]SUNY, Korea**

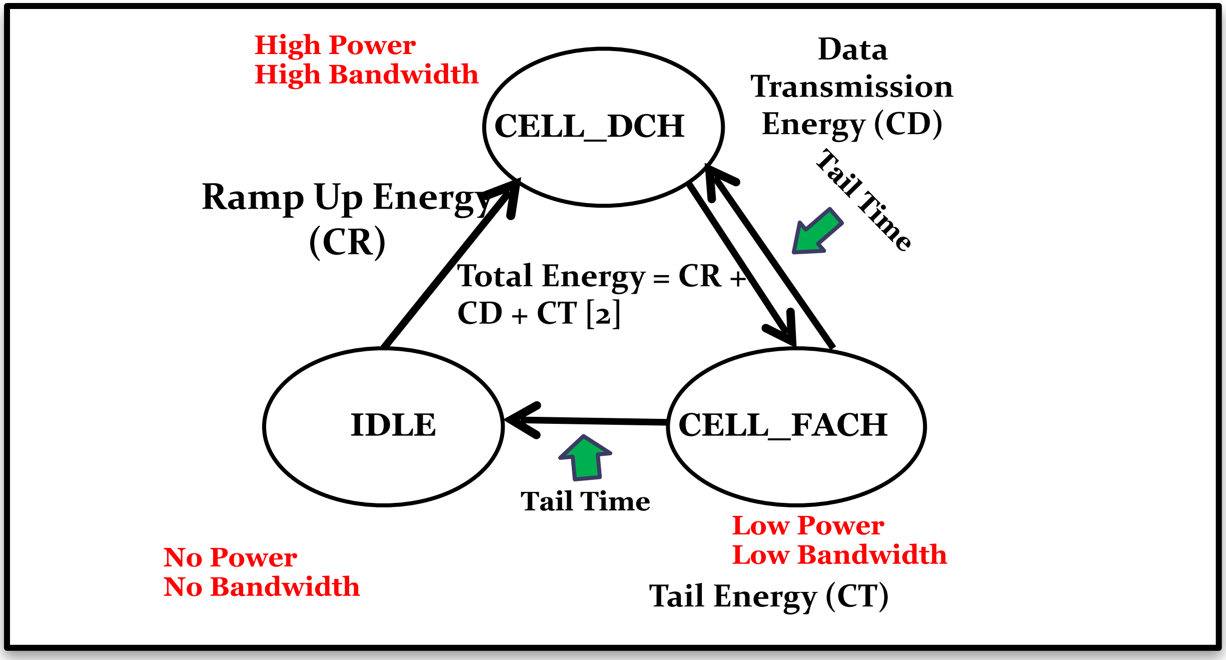
Outline

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

Introduction

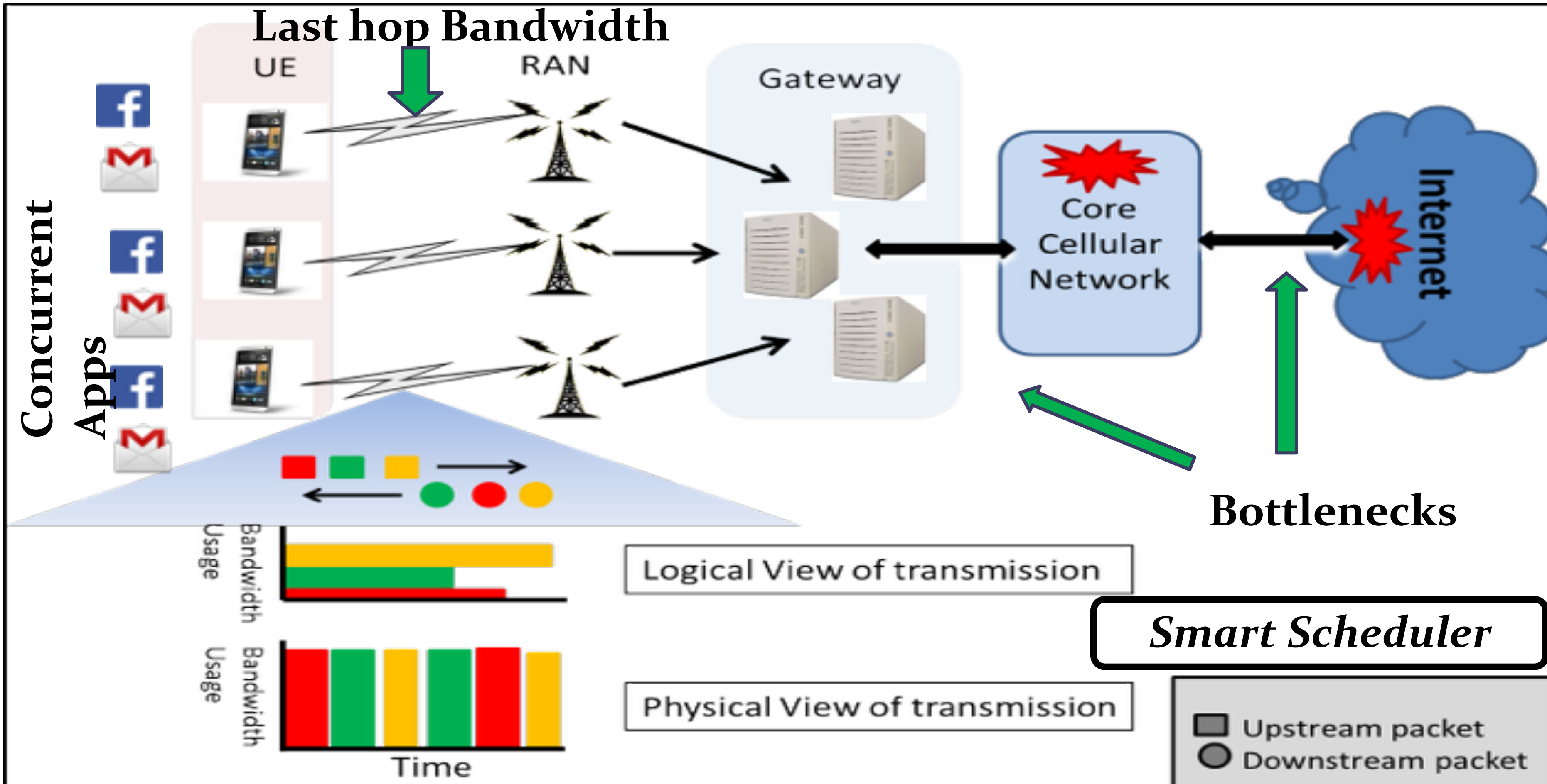
- Display, Network, and CPU are main components of Energy Drain.[**Mittal et. al., MobiCom '13**]
- Poorly written apps can sap 30% to 40% of a phone's battery. [**Mahajan et. al., IMC '09**]
- Network intensive applications are increasing (~69% of the apps are cloud based).
- Different background services running intermittently and waking up the network card for a small duration. **[Qian et. al., WWW '12]**

Cellular Radio Energy Model

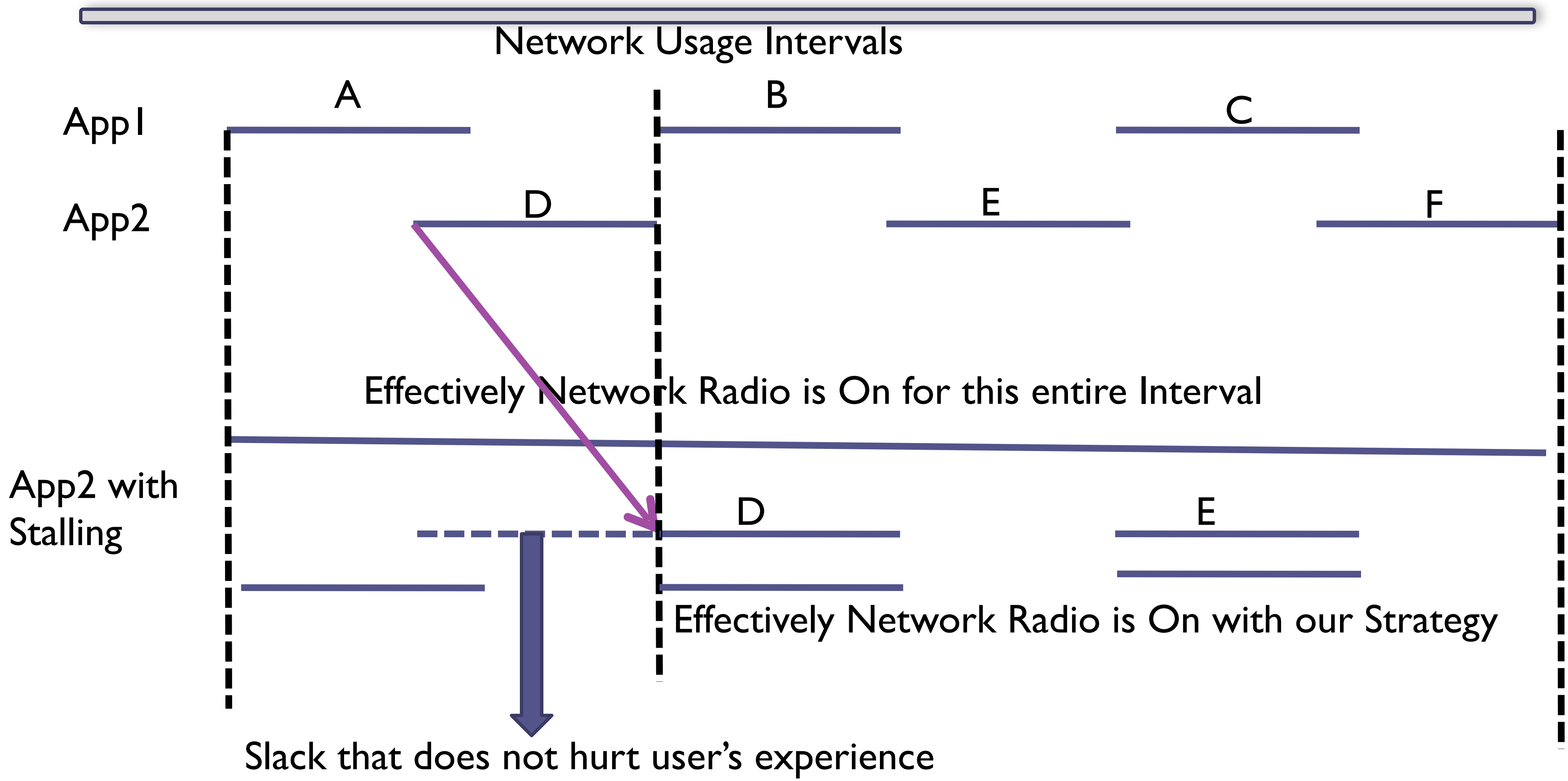


Total Energy Consumption = CR + CD + CT ,
where CR is the ramp up energy (**IDLE to CELL DCH**),
CD is the data transmission energy,
and CT is the tail energy (in CELL FACH).

Total View of App Connectivity



Idea : Cross Application Traffic Aggregation



Problem Objective

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

- Multi-objective optimization problem
  - Minimum Energy => **Best utilization of bandwidth** (Side-effect : Lower Switching Frequency).
  - User's Experience => Request should be served within deadline.

Problem Constraints

- Depending on expected response time of application, a **flexibility or slack time** is allowed to schedule each packet.
- Requests from the **same application cannot be triggered simultaneously**.
- **Total bandwidth consumption** by all the scheduled requests should be **less than the available channel bandwidth** (We consider ***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 time of A_{ij}
- d_{ij} = service duration of A_{ij}
- ft = finish time of all requests in run queue

Deciding Function (F)

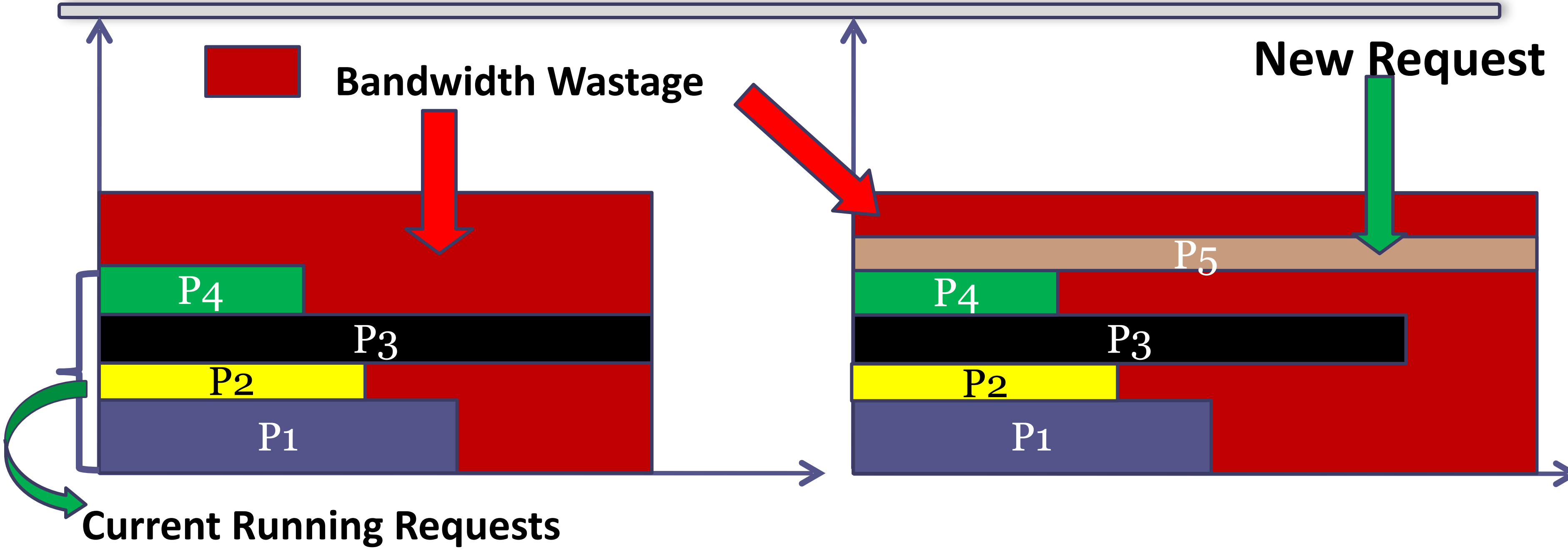
$F=\beta.\ \text{Bandwidth\_wastage} + (1-\beta).\text{Experience\_user}$

Where β is normalizing constant.

$F=\alpha.\beta\ .\ \text{Bandwidth\_wastage} + (1-\alpha)(1-\beta).\text{Experience\_user}$

Where α is factor to give priority over other

F : Bandwidth Wastage Component

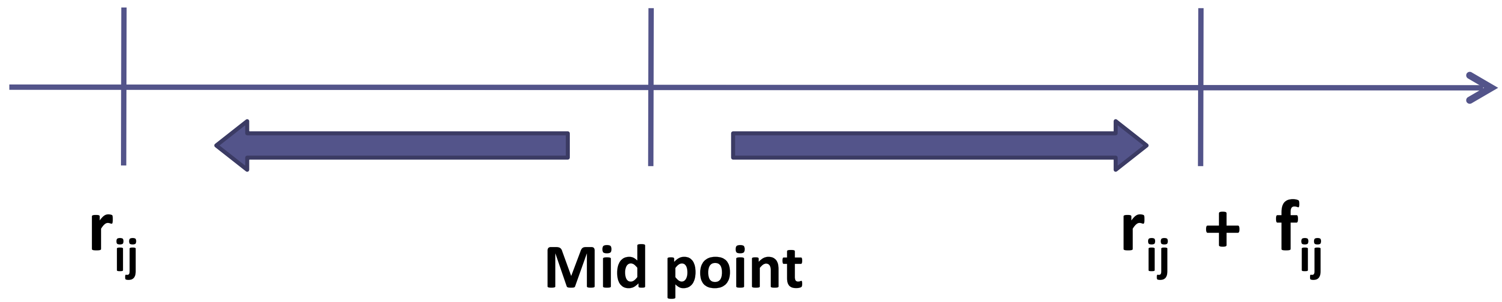


Bandwidth_wastage = $\dfrac{BW1 - BW2}{Max(BW1, BW2)}$

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

$$\text{Experience\_user} = \frac{ft - \dfrac{r_{ij} + (r_{ij} + f_{ij})}{2}}{Max\left(ft, \dfrac{rij + (rij + fij)}{2}\right)}$$

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

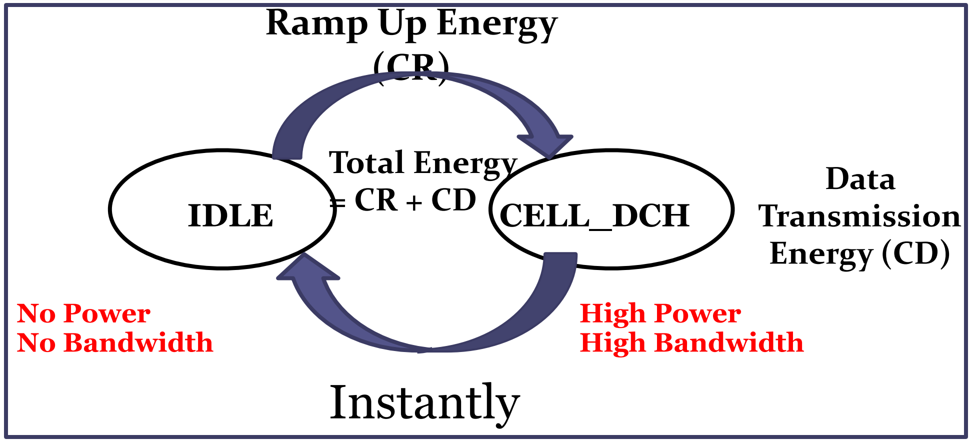
- Application Types (MobiSys '12)
  - Gaming (Short Bursts)
  - Browsing (Medium Bursts)
  - Streaming (Large Bursts)

Experimental Setup

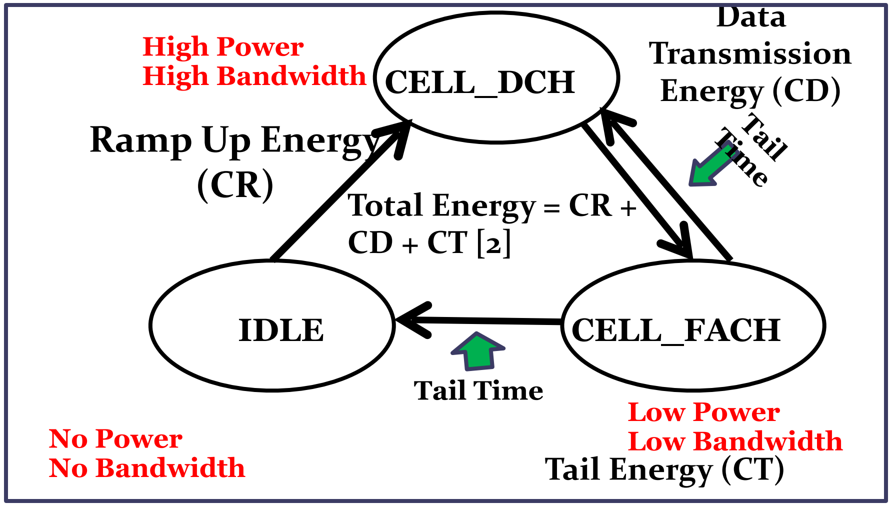
- Synthetic Trace Generation tuning parameters
  - User Interaction Timing (Power Law)
  - Data Transmission Size (Power Law with set of sizes)
  - Bandwidth Demand (Fixed Demand per App)
  - Slack Duration (Fixed per application type)

Experimental Setup

- Switching Strategies
  - ➢ Fast Dormancy.



  - ➢ Fast Dormancy with Fixed Tail Timer.



Real Trace Collection

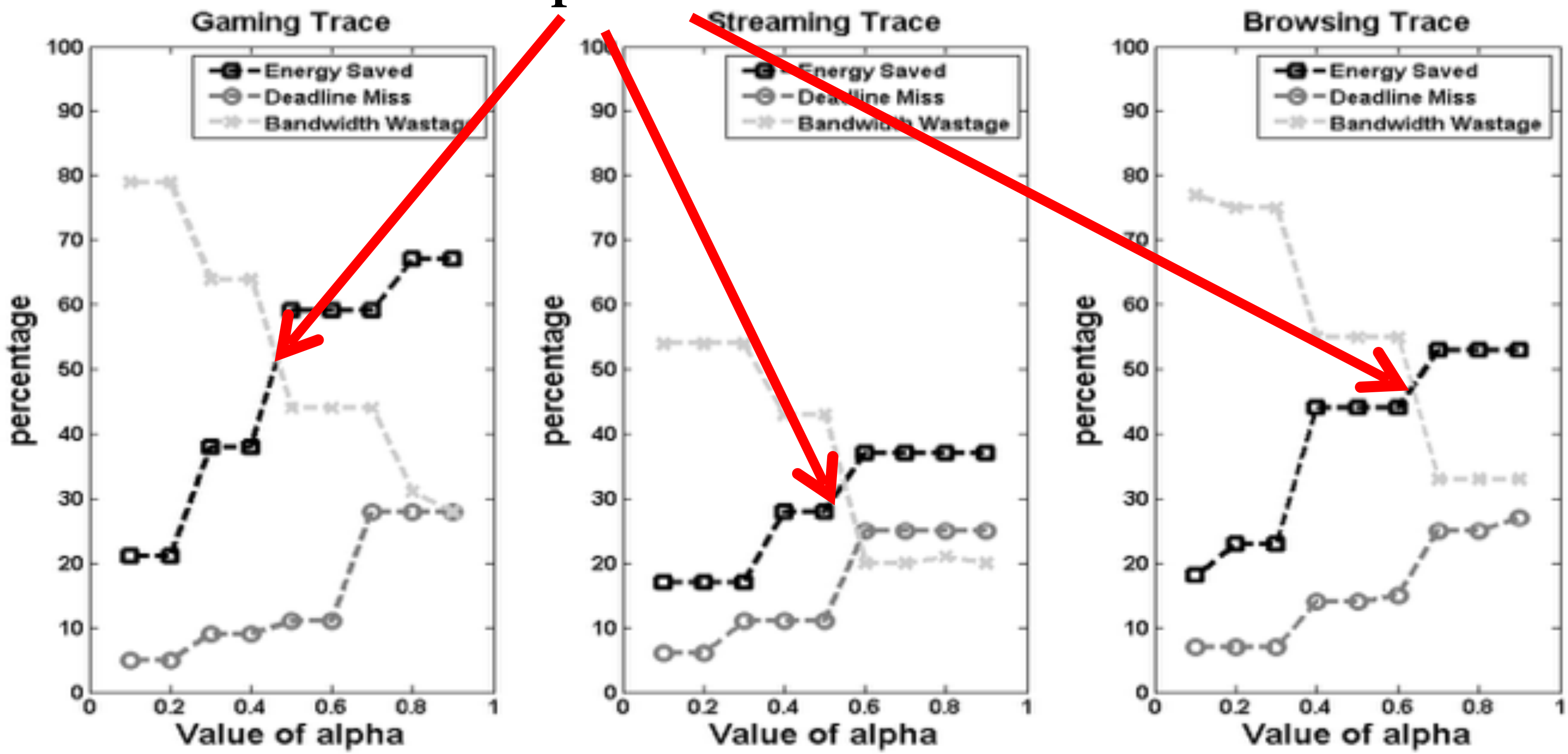
- Collected using ARO (AT&T) tool, tcpdump, and ps.
- Samsung Galaxy S3 GTI9300 (Rooted).
- One hour Browsing Trace from a user.
- Applications are differentiated through port mapping.

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

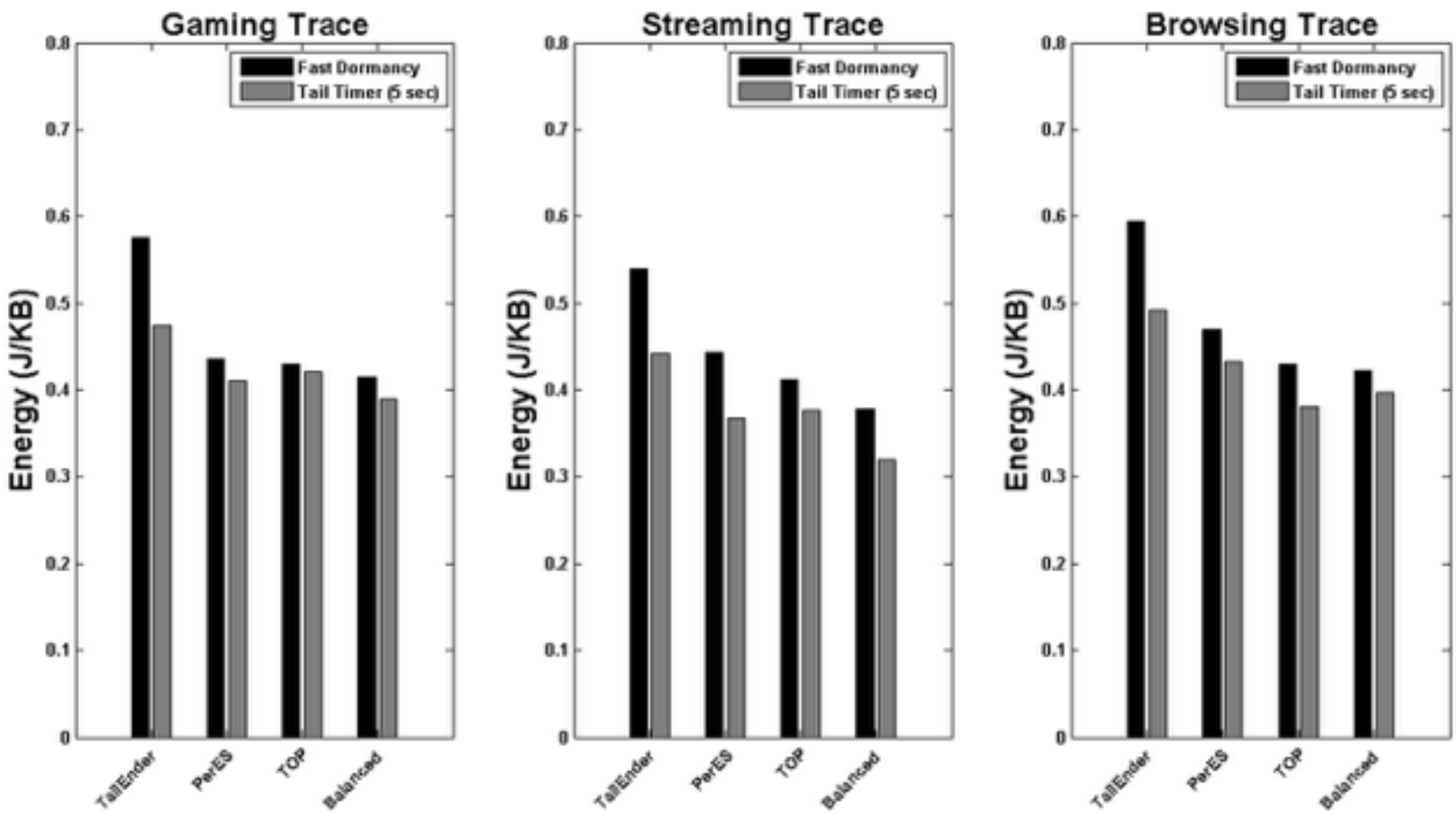
Good Trade-off between Energy and User Experience



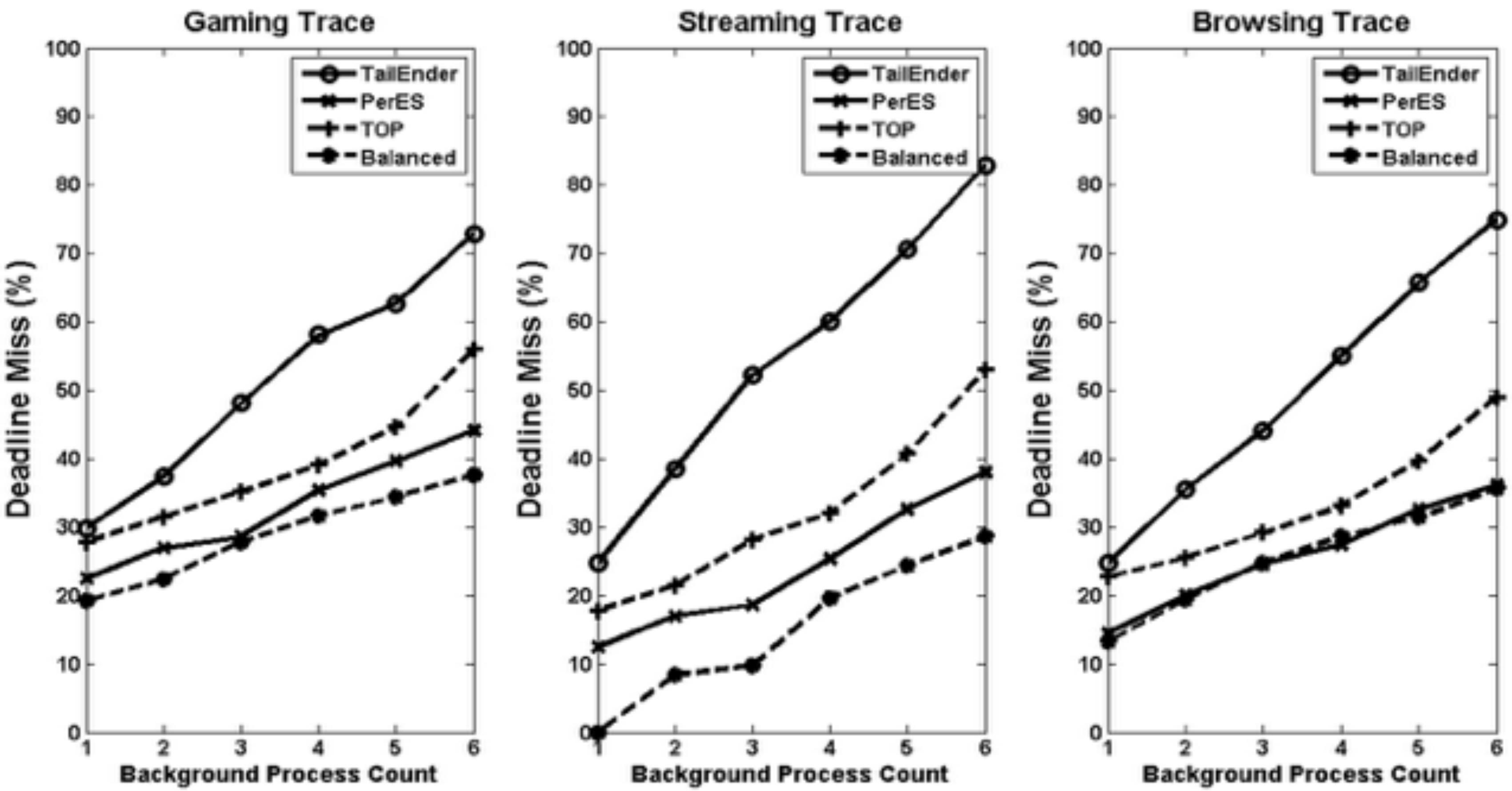
Competing Scheduling Techniques

- **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.

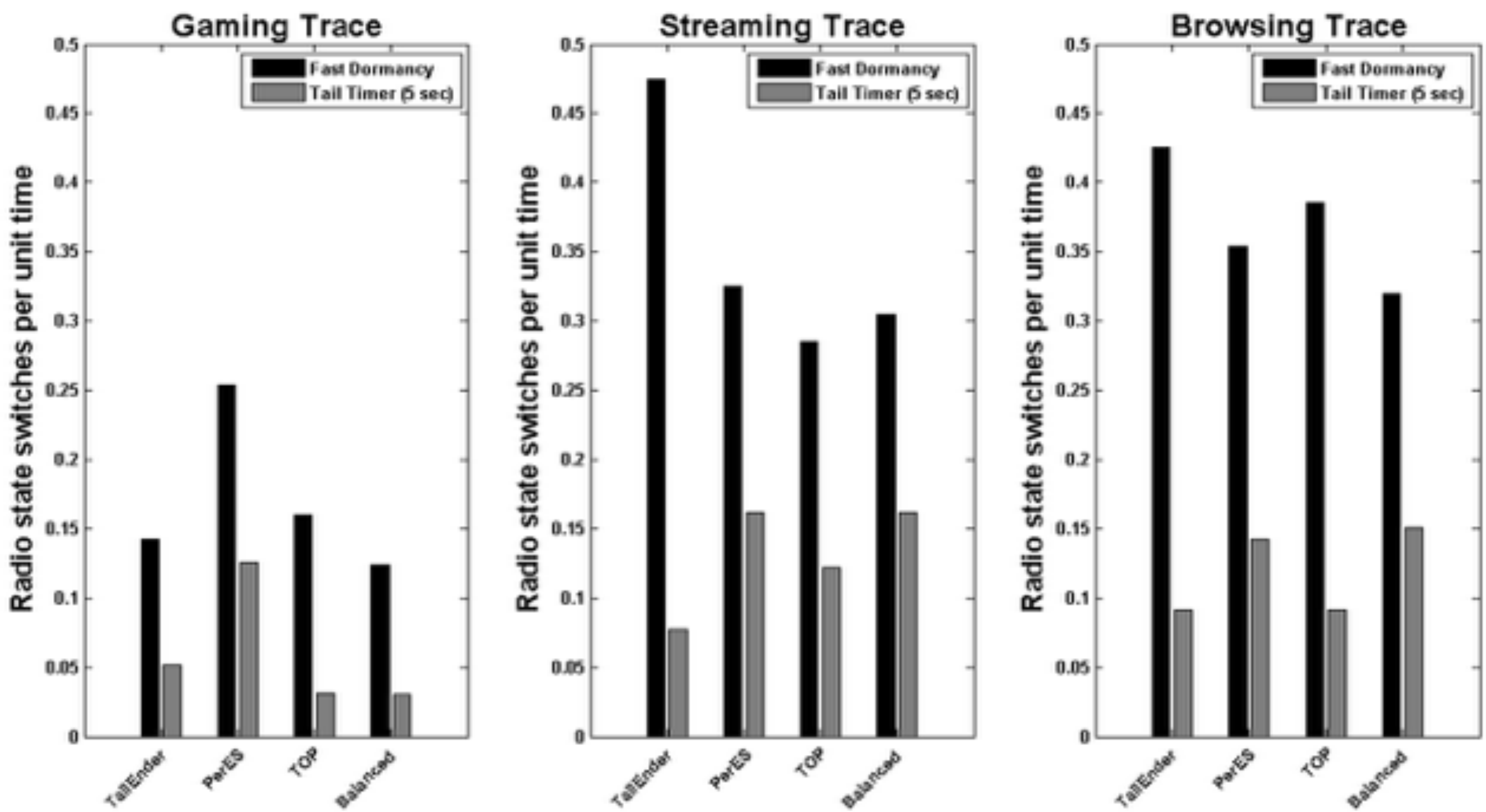
Energy Consumption per KB (~10%)



Deadline Miss

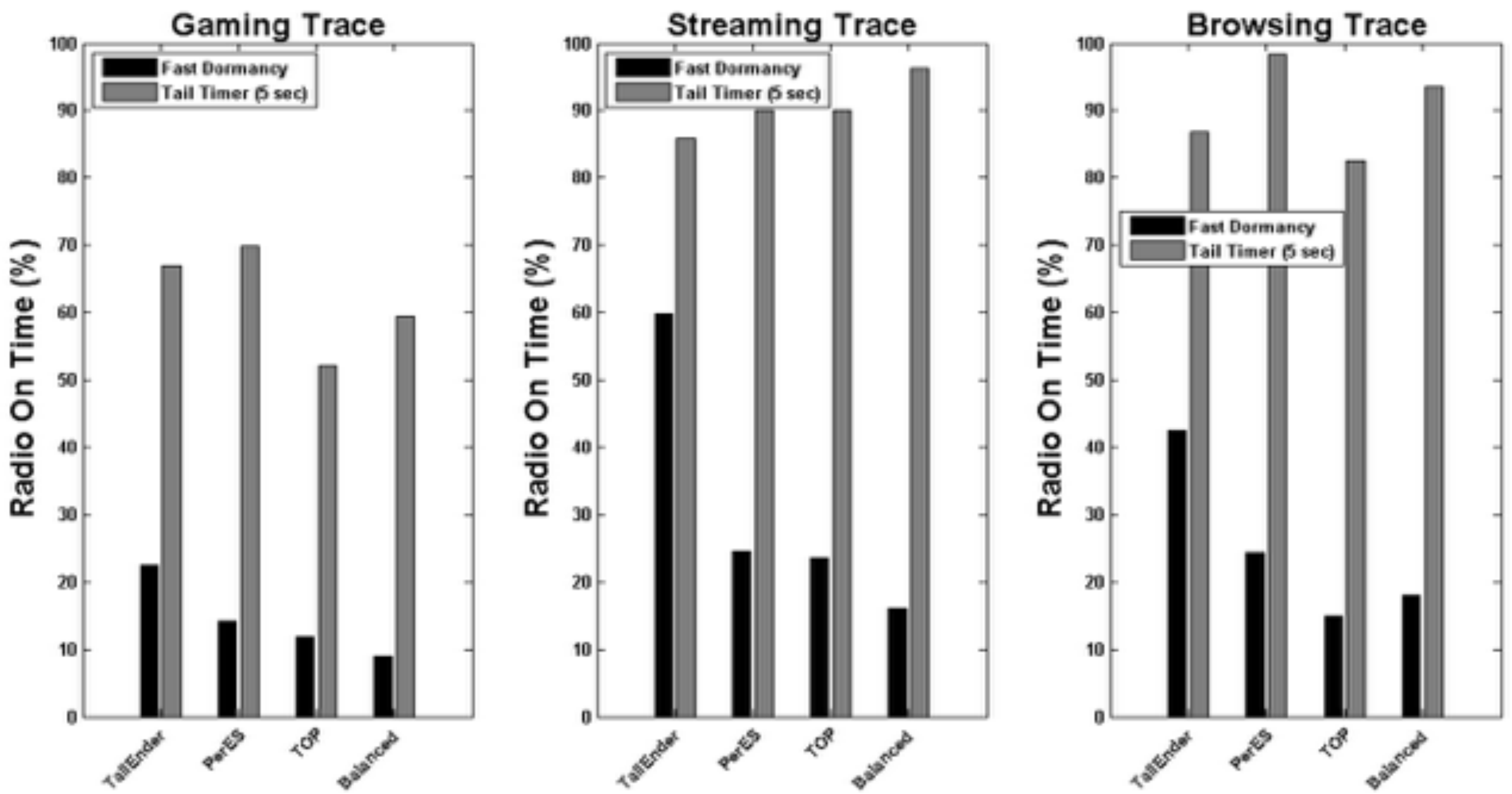


Switching

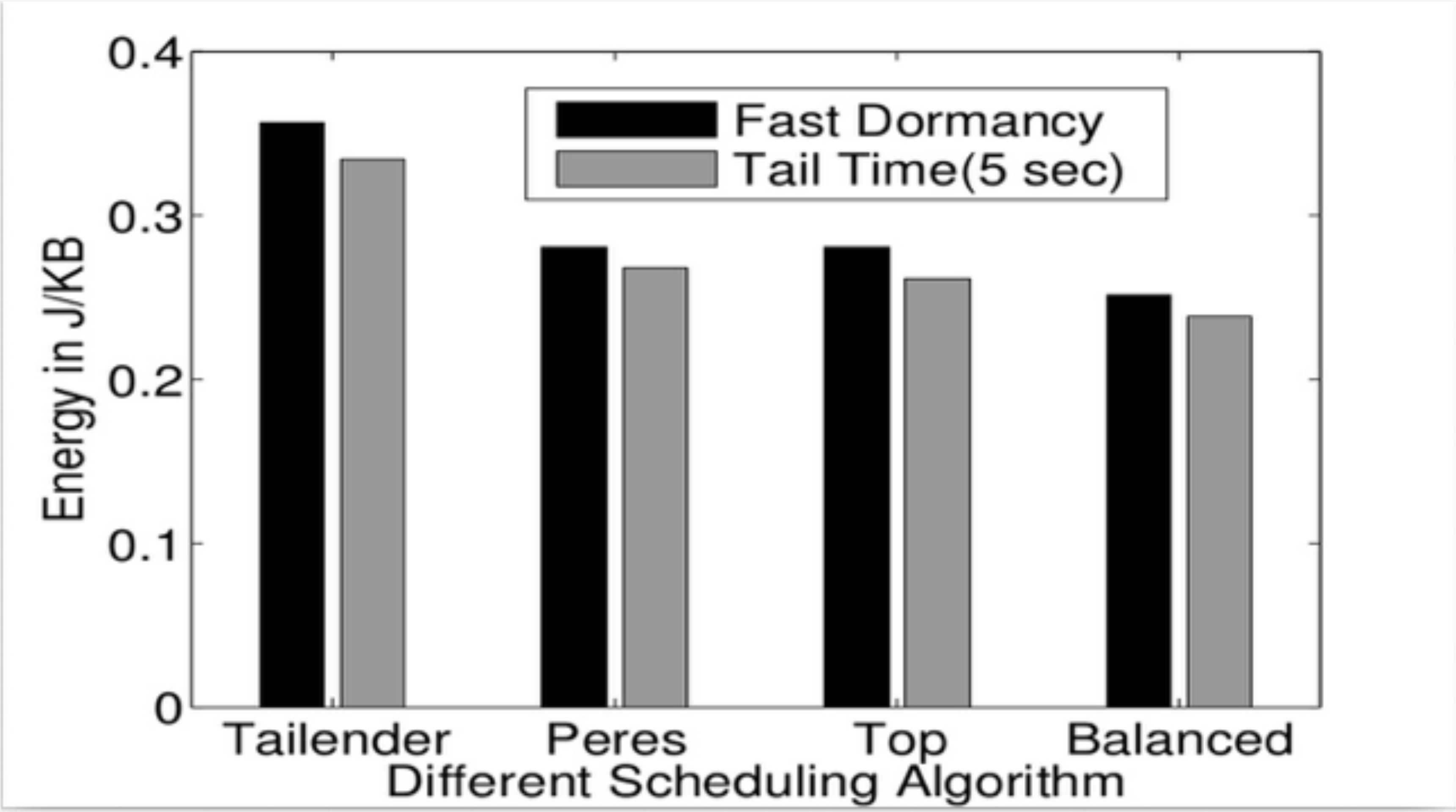


Gaming Trace
Streaming Trace
Browsing Trace
Fast Dormancy
Tail Timer (5 sec)
Radio state switches per unit time
0
0.05
0.1
0.15
0.2
0.25
0.3
0.35
0.4
0.45
0.5
TailEnder
PerES
TOP
Balanced

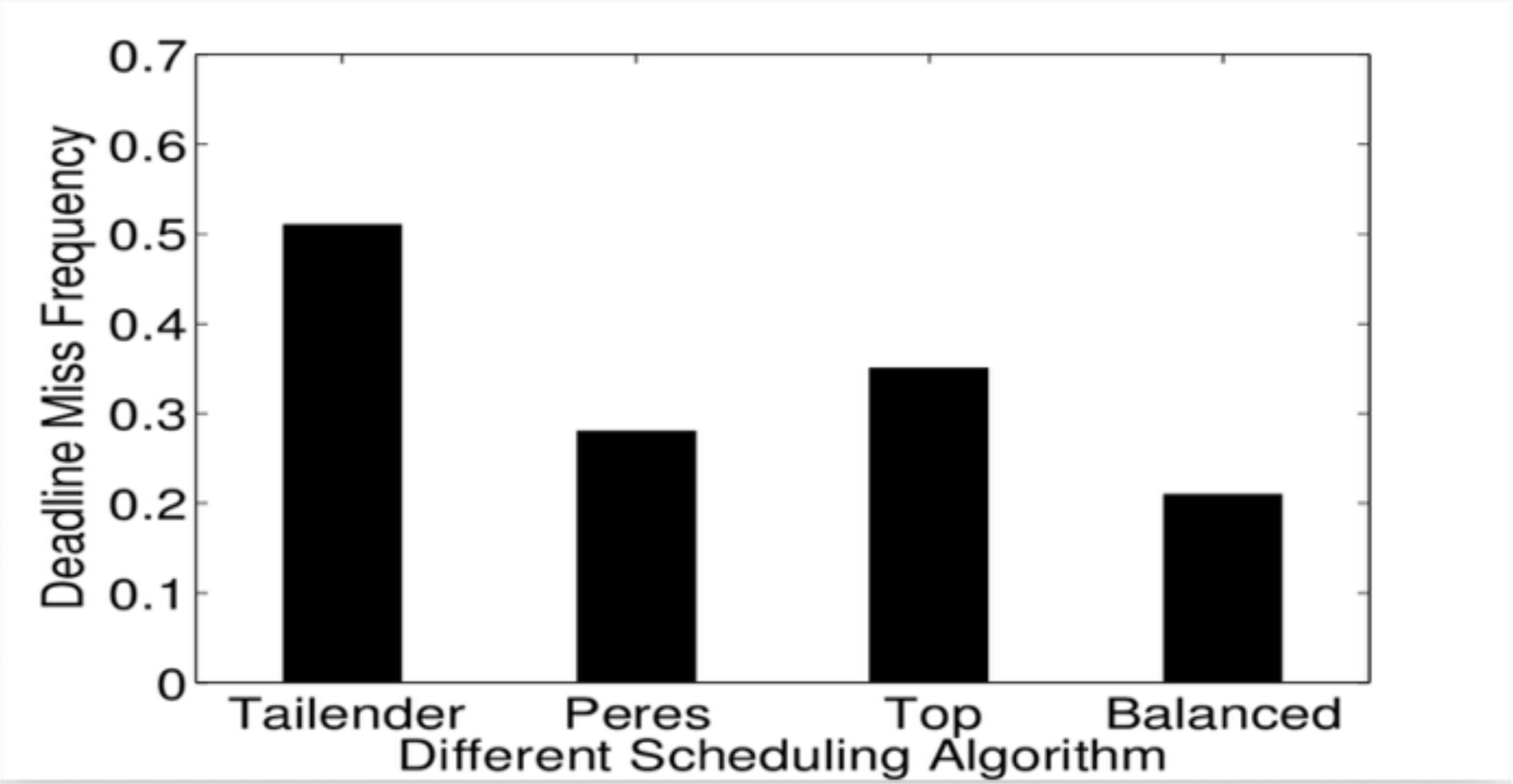
Radio On Time



Energy Consumption in Real Trace



Deadline Miss in Real Trace



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