Proactive Energy-Aware Adaptive Video Streaming on Mobile Devices

Jiayi Meng, Qiang Xu, Y. Charlie Hu
Purdue University
Modern Mobile Apps are Power Hungry
Case Study: 360° Video Streaming on YouTube

Methodology:
- Streamed 6 Youtube videos with different resolutions and frame rates on Pixel 2 over 802.11ac
- Measured power using Monsoon power monitor

Streaming videos @4K/60 draws 580 mA
Energy-aware App Adaption

• Definition: App dynamically adjusts data fidelity to meet a user-specified goal for battery duration [SOSP’99]

• Example scenarios
  • Video streaming apps: adapt video quality to support a 4-hour plane ride with 60% battery level drop
  • Navigation apps: adapt filtering level of a map to support a 2-hour drive with 40% battery level drop

References:
Outline

• Limitations of classic energy-aware adaptation
• Key observation
• Proactive energy-aware adaptation
• Case study: 360° video streaming
Classic Energy-aware App Adaptation: System-level

Energy Control
- Halting or throttling app threads, processes or resource containers [Ecosystem’02, Cinder’11]
- Pre-trained per-hardware-component power modeling [Ecosystem’02, Cinder’11]

Energy Accounting
- External power measurement [Powerscope’99, Quanto’08]

OS

App
- App 1
- App 2
- App 3
- App 4

Hardware
- CPU
- GPU
- HW Decoder
- Network
- Screen

References:
[2] Powerscope: A tool for profiling the energy usage of mobile applications [WMCSA’99]
[3] Energy is just another resource: Energy accounting and energy pricing in the nemesis os [HotOS’01]
[4] Ecosystem: Managing energy as a first class operating system resource [ASPLOS’02]
[5] Quanto: Tracking energy in networked embedded systems [OSDI’08]
Characteristics of Classic Energy-aware App Adaptation

- Reactive
  - OS treats app as black-box and informs it to adapt after energy deviation from the pre-specified budget happens

- Disintegrated
  - OS monitors the app energy drain, while app performs adaptation

- Implication
  - The app does not know how much app fidelity it should adapt in the next time interval
Reactive Adaptation Causes Oscillation

Power

$t_0$  $t_1$  $t_2$  $t_3$  $t_4$  $t_5$  $t_6$  $t_7$  $t_8$  $t_9$

Power Budget

OS upcalls
Key Observation: Modern Apps Have Proactive Built-in Adaptation

• Built-in adaptation: Apps proactively adapt data fidelity to network dynamics or other system constraints to optimize QoE

• Examples
  • Adaptive bitrate (ABR) in video streaming systems: DASH
  • Adaptive offloading computation to edge servers for deep learning enhanced tasks, such as video analytics: Sysmac [1]

References:
Key Idea: Proactive Energy-aware Adaptation

• The energy-drain budget can be seamlessly integrated into the built-in proactive QoE adaptation of the app

• Advantage
  • App energy drain adaptation is no longer an “after-effect” and hence likely to reduce the oscillation in app adaptation and improve the app QoE
Outline

• Limitations of classic energy-aware adaptation
• Key observation
• Proactive energy-aware adaptation
• Case study: 360° video streaming
Background of ABR Video Streaming

Video Client

Request:

Output

Video Server

1 sec/s

Input

1 sec

video content

bitrate

video content

ABR Algorithm

Throughput

Time

bitrate

Animation borrowed from Te-Yuan Huang (SIGCOMM ’14) http://conferences.sigcomm.org/sigcomm/2014/doc/slides/38.pdf
ABR Problem Formulation [Sigcomm’15]

\[
\text{maximize } \sum_{k} QoE_i \\
\text{subject to buffer and network dynamics}
\]

\[
QoE_k = \text{Video Quality}_k - \text{Quality Switching}_{(k-1, k)} - \text{Rebuffering}_k
\]

References:
Model Predictive Control (MPC) Algorithm [Sigcomm’15]

• Goal: decide the video chunk quality to be fetched next $F_k$ by predicting QoE of next $N$ chunks

$\max_{F_k, \ldots, F_{k+N-1}} \sum_{k} QoE_i$

How to integrate energy budget into the built-in app adaptation logic?

References:
Energy-aware QoE Maximization Problem for ABR

User-specified energy budget: total energy budget $E_b$ over a fixed amount of time $T_d \rightarrow$ power budget $P_b = E_b / T_d$

E.g. $E_b$ : 50% battery level drop; $T_d$ : 4-hour plane ride

$$\max \sum QoE_i$$

subject to buffer and network dynamics

and total energy constraint
Proactive Energy-aware ABR

Energy-aware ABR Controller

Feedback Control Loop

Energy Profiler (of past intervals)

Next Video Chunk

Power Budget

Video Chunks

Playback Buffer

Power Predictor

X-put Predictor

Video Chunks
Challenges of Proactive Energy-aware ABR

• How to predict power consumption for each adaptation candidate?

• How to incorporate energy budget into its QoE optimization logic?
Proactive Energy-aware ABR

\[
\max \sum QoE_i
\]

subject to buffer and network dynamics

and \( E_k + \cdots + E_{k+N-1} < N \cdot P_b \cdot \delta t \)

- \( E_k \) predicted energy for chunk \( k \)
- \( N \) number of chunks to predict
- \( P_b \) power budget
- \( \delta t \) per-chunk interval duration
Exploiting Energy Surplus in Proactive Energy-aware ABR

• App energy drain is cumulative and elastic over time and thus energy deficit/surplus ($E_s$) is accumulated

$$\max \sum QoE_i$$

subject to buffer and network dynamics

and $E_k + \cdots + E_{k+N-1} < N \cdot P_b \cdot \delta t + E_s$

---

$E_k$ predicted energy for chunk $k$

$N$ number of chunks to predict

$P_b$ power budget

$\delta t$ per-chunk interval duration

$E_s$ energy surplus so far
<table>
<thead>
<tr>
<th>LA(1):</th>
<th>LA(1)+LB:</th>
<th>LA(N)+LB:</th>
</tr>
</thead>
<tbody>
<tr>
<td>look ahead 1</td>
<td>look ahead 1 and look back</td>
<td>look ahead N and look back</td>
</tr>
<tr>
<td>max $\sum QoE_i$</td>
<td>max $\sum QoE_i$</td>
<td>max $\sum QoE_i$</td>
</tr>
<tr>
<td>subject to buffer and network dynamics and $E_k &lt; 1 \cdot P_b \cdot \delta t$</td>
<td>subject to buffer and network dynamics and $E_k &lt; 1 \cdot P_b \cdot \delta t + E_s$</td>
<td>subject to buffer and network dynamics and $E_k + \cdots + E_{k+N-1} &lt; N \cdot P_b \cdot \delta t + E_s$</td>
</tr>
</tbody>
</table>
Trace-driven Evaluation

- Network-trace datasets: Ytrace and FCC
- Devices: Pixel 2 and Moto Z3
- Two types of power budgets:
  - Low power budget: 20\textsuperscript{th}-percentile per-interval power draw
  - High power budget: average power draw over the streaming session
LA(N)+LB saves 29.10% power than Default and achieves the highest QoE among the three proactive designs.
Performance Comparison between Reactive and Proactive Approaches

RA: reactive approach
LA(N)+LB: proactively looking ahead for N chunks and looking back
LA(N)+LB+S has 44.8% higher QoE than RA+S

Low Power Budget on Pixel 2

RA+6
LA(1)+LB
LA(1)+LB+6

RA
RA+S
LA(N)+LB
LA(N)+LB+S

Percent of Power Diff (%)
QoE Breakdown Comparison between Reactive and Proactive Approaches

LA(N)+LB+S shows significant benefits over RA+S because of reduced quality switching

Low Power Budget on Pixel 2
Generalization

• Supporting multiple apps competing for the energy budget
  • User provides input on how the total energy budget should be split
  • Or a global energy-aware controller jointly optimizes QoE of concurrently running apps
Summary

• Classic reactive energy-aware app adaptation can lead to app fidelity oscillation which can negatively affect user-perceived QoE.

• We observe the built-in QoE optimization frameworks of modern mobile apps naturally lend themselves to proactive energy-aware app adaptation.

• We showcase how to integrate user-specified energy budget with the built-in app adaptation logic of MPC-based ABR system, which has been open-sourced.

• Proactive energy-aware video streaming improves QoE by 44.8% (Pixel 2) and 19.2% (Moto Z3) over the reactive approach under low power budget.
Thanks!

Please feel free to contact us (meng72@purdue.edu), if you have further questions. 😊
Backup
Challenges of Proactive Energy-aware ABR

• How to predict power consumption for each adaptation candidate?

• How to incorporate energy budget into its QoE optimization logic?
Asynchronous Component Behavior
Function-wise Power Prediction

• Key idea: cluster hardware components processing the common video chunk at each time interval
  → each cluster corresponds to one high-level app function

• Functions for 360° video streaming:
  • Video decoding and displaying function
  • Network transmission function
Challenges of Proactive Energy-aware ABR

• How to predict power consumption for each adaptation candidate?

• How to incorporate energy budget into its QoE optimization logic?
Function-wise power predictor achieves low mean per-interval energy prediction error of 4.87% (Pixel 2) and 5.86% (Moto Z3).
Performance of Proactive Approaches under Low Power Budget on Pixel 2

LA(N)+LB saves 29.10% power than Default and achieves the highest QoE among the three proactive designs.
QoE Breakdown of Proactive Approaches under Low Power Budget on Pixel 2

- **Video Quality**: LA(1) and LA(1)+LB offer better Video Quality compared to Default.
- **Smoothness**: LA(1)+LB and LA(N)+LB show improved Smoothness compared to LA(1) and LA(1)+LB.
- **Rebuffering**: LA(1) and LA(1)+LB experience lower Rebuffering compared to LA(N)+LB.
Performance of Proactive Approaches under High Power Budget on Pixel 2

The penalty of proactive energy-aware adaptation is really small, compared to the energy-oblivious default ABR.