Energy-Efficiency in Mobile Software

Julia Rubin
(jointly with Duling Lai, Jürgen Cito, Phillip Stanley-Marbell)

January 31, 2017
Mobile Applications

- More than 2 billion smartphones in use
  - More than desktop computers
  - Plus smart watches, glasses, cameras, TVs, in-car navigation, and messaging

- More than 3 million mobile apps in official app stores
  - ... and much more apps in hundreds of alternative stores

- ~2.3 million active mobile developers worldwide
  - 760,000 people in Asia
  - 680,000 people in North America
  - 680,000 people in Europe

[https://blog.newrelic.com/2014/06/13/mobile-app-development-trends-worldwide-need-know/]
[Developer Economics Q3 2014]
Mobile Application Developers

Lack knowledge, tools, and incentives to deal with:

– Security
– Privacy
– Energy-efficiency
What Can Go Wrong?
What Can Go Wrong?

Major consumers of energy:
Screen, WiFi, GPS, Sensors, Camera, CPU

[Pathak et al., EuroSys’12, Banerjee et al., FSE’14]
Example App

GPS location used
Example App

1. GPS location update started
2. GPS location used
Example App

GPS location update started

1

GPS location used

2
Example App

1. GPS location update started
2. GPS location used

47% more energy
Energy Bugs

An error in the system that causes an unexpected amount of high energy consumption

[Pathak et al., HotNets’11]
Energy Bugs and Hotspots

An application consumes an abnormally high amount of battery power ...

- ... even after it has completed execution → Bug

- ... even though the utilization of its hardware resources is low → Hotspot

[Banerjee et al., FSE’14]
Taxonomy of Energy Bugs and Hotspots

- **Hardware Resources**
  - Resource Leak
  - Suboptimal Resource Binding

- **Sleep-state Transition Heuristics**
  - Wakelock Bug
  - Tail Energy Hotspot

- **Background Services**
  - Vacuous Background Services
  - Expensive Background Services

- **Defective Functionality**
  - Immortality Bug
  - Loop Bug

[Banerjee et al., FSE’14]
Resources (e.g., WiFi) that are not released or the device component (e.g., screen, CPU) is left in a high-power state after the application has finished execution

[Banerjee et al., FSE’14]
A failure to remove a service (e.g., location or sensor updates), which will keep on reporting data even though no application needs it

[Banerjee et al., FSE'14]
Taxonomy of Energy Bugs and Hotspots

[Banerjee et al., FSE’14]
Taxonomy of Energy Bugs and Hotspots

Hardware Resources

- Resource Leak
- Suboptimal Resource Binding

Sleep-state Transition Heuristics

- Wakelock Bug

Background Services

- Vacuous Background Services

Defective Functionality

- Immortality Bug
- Loop Bug

Binding resources too early or releasing them too late

[Banerjee et al., FSE’14]
Taxonomy of Energy Bugs and Hotspots

Hardware Resources
- Resource Leak
- Suboptimal Resource Binding

Sleep-state Transition Heuristics
- Wakelock Bug
- Tail Energy Hotspot

Background Services
- Vacuous Background Services

Defective Functionality
- Immortality Bug

Scattered usage of components (e.g., screen, WiFi), which causes tail energy consumption when component is switched off

[Banerjee et al., FSE’14]
Taxonomy of Energy Bugs and Hotspots

Hardware Resources
- Resource Leak

Sleep-state Transition Heuristics
- Wakelock Bug

Background Services
- Vacuous Background Services
- Expensive Background Services

Defective Functionality
- Immortality Bug
- Loop Bug

Fine-grained instead of coarse-grain updates (e.g., GPS vs. WiFi based location); unnecessarily high sampling rate for a service

[Banerjee et al., FSE’14]
Taxonomy of Energy Bugs and Hotspots

[Banerjee et al., FSE’14]
Taxonomy of Energy Bugs and Hotspots

- Hardware Resources
  - Resource Leak
  - Suboptimal Resource Binding
- Sleep-state Transition Heuristics
  - WakeLock Bug
- Background Services
  - Vacuous Background Services
- Defective Functionality
  - Immortality Bug

- **Pathak et al., MobiSys’12:** static analysis to find definitions (resource acquisition) that are not killed (released)
- **Guo et al., ASE’13 (Relda):** builds a Function Call Graph (callbacks and resource-related operations) to check which resources are not released
**Taxonomy of Energy Bugs and Hotspots**

- **Hardware Resources**
  - Resource Leak
  - Suboptimal Resource Binding

- **Sleep-state Transition Heuristics**
  - Wakelock Bug

- **Background Services**
  - Vacuous Background Services

- **Defective Functionality**
  - Immortality Bug

- **Background Services**
  - Expensive Background Services

- **Functionality**
  - Defective Functionality

- **Heuristics**
  - Heuristics

- **Services**
  - Services

- **Resource**
  - Resource

- **Leak**
  - Leak

- **Subop=mal**
  - Suboptimal

- **Resource**
  - Resource

- **Binding**
  - Binding

- **Sleep-state**
  - Sleep-state

- **Transition**
  - Transition

- **Heuristics**
  - Heuristics

- **Wakelock**
  - Wakelock

- **Bug**
  - Bug

- **Tail**
  - Tail

- **Energy**
  - Energy

- **Hotspot**
  - Hotspot

- **Liu et al., PerCom’13 (GreenDroid):** calculates utilization ratio by using dynamic taint analysis
- **Banerjee et al., MobileSoft’16:** improves binding location by matching defect and design expressions
Example App

1. GPS location update started
2. GPS location used

47% more energy
Example App

1. **47% more energy**

2. **visibly slower**

GPS location update started

GPS location used
QoS Considerations

“For best location accuracy, you might choose to start listening for location updates when users begin creating the content or even when the application starts, <...>”

“You might need to consider how long a typical task of creating the content takes and judge if this duration allows for efficient collection of a location estimate.”

[https://developer.android.com/guide/topics/location/strategies.html]
Taxonomy of Energy Bugs and Hotspots

Hardware Resources
- Resource Leak
- Suboptimal Resource Binding

Tail Energy Hotspot

- Chen et al., HotPower’13: prefetching ads; upper-bound is 3.2%, hard to achieve
- Li et al., ICSE’16: bundles sequential HTTP requests, 15% reduction

Defective Functionality
Expensive Background Services
Loop Bug
Taxonomy of Energy Bugs and Hotspots

- Hardware Resources
  - Resource Leak
  - Suboptimal Resource Binding
- Sleep-state Transition Heuristics
  - Wakelock Bug
  - Tail Energy Hotspot
- Background Services
  - Vacuous Background Services
  - Expensive Background Services
- Defective Functionality
  - Immortality Bug
  - Loop Bug
Taxonomy of Energy Bugs and Hotspots

Hardware Resources
- Resource Leak
- Suboptimal Resource Binding

Sleep-state Transition Heuristics
- Tail Energy Hotspot

Background Services
- Expensive Background Services

Defective Functionality
- Immortality Bug
- Loop Bug

• Cito et al., ASE’16
Advertising and Analytics (A&A)

- At least 3 third-party libs in an app [Rubin at al., ASE’15]
- Consume 65%-75% of energy [Pathak et al., EuroSys’12]
- More than 40% of HTTP connections do not contribute to user-observable behavior [Rubin at al., ASE’15] – Analytics
- Advertising consumes 16% of energy (plus 48% more CPU time and 79% more network data) [Gui et al., ICSE’15]
Mobile Users

**Goal**
Maximize Battery Life
Minimize Energy Consumption

Mobile Developers

**Goal**
Maximize Revenue (Ads)
Maximize App Insight (Analytics)
Main Idea

Automatically identify *recurrent A&A requests* and *adapt their frequency* to the current battery state.


Battery Aware Transformation

Based on

- $\rho$ – period of the recurrent requests
- $b$ – battery status

add a delay before recurrent A&A requests

Linear adaptation:

$$f_{\text{linear}}(b, \rho) = \frac{\rho \times c}{b}$$

Low power mode (at 20% battery status):

$$f_{\text{LowPowerMode}}(b, \rho) = \begin{cases} 
\rho \times c & \text{if } b \leq 0.2 \\
0 & \text{otherwise}
\end{cases}$$
Savings – an Example

- VLC Direct: video stream player
- One recurrent request – every 30 sec
- Introduced 100% delay – to 60 sec
- Run for 30 mins
- 5.8% reduction in energy consumption (16% upper bound)
Providing Incentives

• Problem: applications are “greedy”
  – Do not consider other apps
  – Do not consider battery status

• In plan: game-theoretical approach for mobile energy marketplace
  – price energy and bill applications for the energy they use
Summary

Mobile Application Developers

Lack knowledge, tools, and incentives to deal with:

- Security
- Privacy
- Energy-efficiency

Taxonomy of Energy Bugs and Hotspots

- Hardware Resources
  - Resource Leak
  - Suboptimal Resource Binding
- Sleep-state Transition Heuristics
  - Wakelock Bug
  - Tail Energy Hotspot
- Background Services
  - Vacuous Background Services
  - Expensive Background Services
- Defective Functionality
  - Immortality Bug
  - Loop Bug

Providing Incentives

- Problem: applications are “greedy”
  - Do not consider other apps
  - Do not consider battery status
- In plan: game-theoretical approach for mobile energy marketplace
  - price energy and bill applications for the energy they use

[Banerjee et al., FSE’14]