The Model Is Not Enough: Understanding Energy Consumption in Mobile Devices

James Bornholt, Australian National University, u4842199@anu.edu.au
Todd Mytkowicz, Microsoft Research, toddm@microsoft.com
Kathryn S. McKinley, Microsoft Research, mckinley@microsoft.com

Why understand energy?

Smart phones and tablets force software developers to focus on battery life.

Developers already optimise performance using profilers. Our goal is to build an energy profiler.

Why is energy profiling difficult?

The two ways to calculate energy—hardware power meters and software modelling—have drawbacks.

Attribution of energy to code is made difficult by tail power states, which shift the blame.

Tail power states: who's to blame?

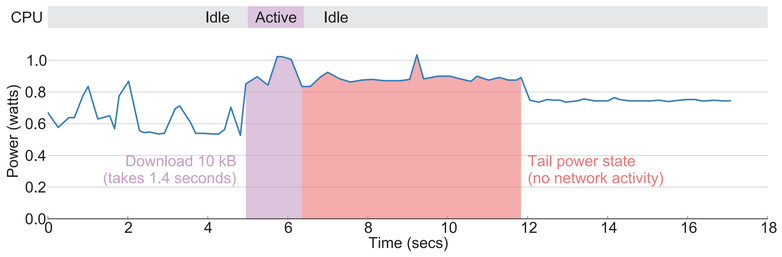

Figure 1 The 3G radio's tail power state consumes energy while the CPU is idle.

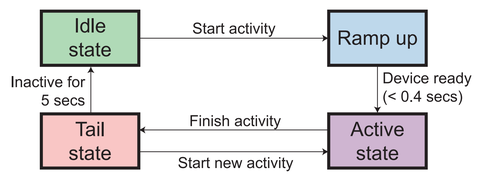

Figure 2 Tail power states avoid returning to the ramp up state, therefore reducing latency.

A deeper look: energy profiling

Why understand energy?

- Software and energy interact in subtle and non-trivial ways – for example, grouping of network requests to avoid thrashing the network hardware
- Energy bugs present serious usability problems, because users cannot easily identify them until it is too late (i.e. the battery is empty)

Why is energy profiling difficult?

- Hardware meters are often expensive and bulky
- Models are specific to their training environment
- Tail power states reduce ramp-up latency by allowing components to remain powered on after last use
- So it's wrong to attribute current power draw to the currently executing code – the real culprit may be long gone!

Power modelling

Modelling uses metrics such as CPU and network usage to extrapolate power draw, but this has issues for profiling use.

Recent work uses system call tracing to handle tail power states, but other issues remain.

Power measurement

Hardware power meters are common for power measurement, but these are impractical for most developers.

Our work shows the onboard "fuel gauge" battery sensor is accurate to within 2% of external meters, but measurement alone cannot address tail power states.

Modelling misses important variation

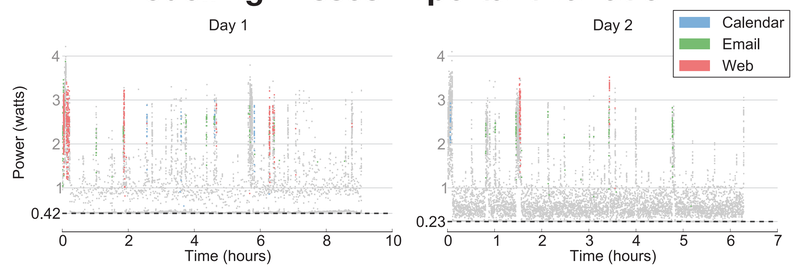

Figure 3 Modelling misses a base power variation, worth 30% of battery life over 8 hours.

A deeper look: modelling and measurement

Power modelling

- The raw numbers are often quite good: errors of below 10%
- But the models:
  1. Do not address tail power states
  2. Are specific to the lab environment they are trained in
  3. Cannot detect the unexplained power variation in Figure 3
- In system call tracing, finite state machines model each component's power states, using system calls to transition the machine
- This addresses the tail power state problem, but suffers the other drawbacks of modelling, and also cannot yet capture important components like the screen

Power measurement

- External meters are expensive and bulky, making them inaccessible and difficult to use
- The fuel gauge typically measures battery capacity, but modern sensors provide instantaneous power draw
- When testing an HTC Windows Phone, the onboard sensor was accurate to within 2% of total energy compared to the power meter
- The overhead of sampling the fuel gauge at 5 Hz was less than 4%.
- This means the fuel gauge is accurate enough to replace the external power meter for many uses, but may not be appropriate for very high frequency sampling
  - The power meter we used samples at 5000 Hz

A hybrid approach

Based on our results, we advocate a hybrid approach to energy profiling.

This approach makes energy profiling more accurate, more actionable, and more accessible.

What do we get from this data?

Energy profiles let developers isolate poor energy usage in their code.

Energy profiles open a new field of potential online OS-level energy optimisations.

Energy profiles identify real bugs

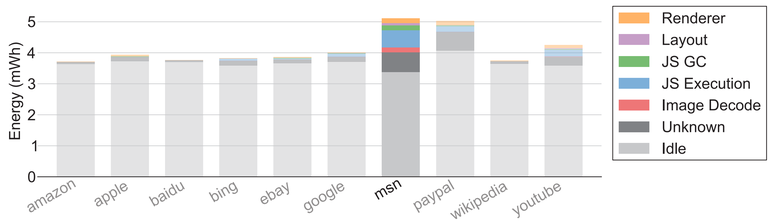

Figure 4 An energy bug in the MSN website, seen and diagnosed by an energy profiler.

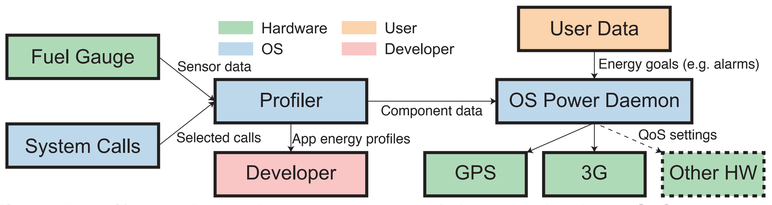

Figure 5 Profiles can be used to meet energy goals by tuning hardware QoS settings.

A deeper look: hybrid energy profiles

A hybrid approach

- Accurate onboard sensors for online, whole-system power readings, and system call modelling to handle tail power states
- Energy profiles will always show a complete picture of system energy use, rather than omitting unmodelled components, making profiles more accurate
- Tail power states are handled, making profiles more actionable
- By removing the need for hardware, even for training, profiles are more accessible

What do we get from this data?

- In Figure 4 we see MSN as a clear outlier in energy use
- The profile identifies excessive image preloading as the cause of this inefficiency
- The OS can use online profiles to achieve battery life goals; for example, tuning energy use to ensure a scheduled alarm goes off
- Guided by component attribution, this tuning can target specific energy users like the GPS, and trade accuracy or speed for battery life

Australian National University

Microsoft Research