Lecture 19: Energy efficiency in mobile applications

Mythili Vutukuru

CS 653 Spring 2014

April 3, Monday

Energy savings at various layers

- PHY / MAC – put radio to sleep to avoid idle listening (e.g., 802.11 power save mode)
- Routing layer – routing and forwarding protocols can be made energy aware and pick paths / forwarding nodes that save energy
- TCP layer – TCP interactions with 802.11 power save mode can lead to increased delay in getting ACKs (due to radio sleeping) in the slow start phase, and negatively impact energy savings
- This lecture – energy consumption at the application layer, and ideas to save power.

Profiling energy usage of apps

- How to profile energy usage?
  - Log app activity at process, thread, routine, system call granularity.
  - Log energy usage with a power meter.
  - Correlate the two to get an energy profile of an app.
- Many complications: for example, some components (3G, GPS) exhibit tail energy behavior. When one app turns on 3G radio, the radio stays in the ON state for a certain “radio tail” duration after the app finishes using it. It is not clear whom to assign this energy usage to.
  - One way is to attribute it to the last system call that triggered the tail.
- Such challenges make energy profiling non-trivial.
- See the paper “Fine grained energy accounting on smartphones with Eprof” for more details.

Energy usage in apps

- After profiling apps for energy usage, what are the main insights?
- Energy usage can be classified into two types
  - Energy spent on I/O – radio, GPS, WiFi, screen
  - Energy spent on computation – rendering web pages, computing future moves in games, speech recognition
- Some examples of high power consumption in apps
  - Google search or news apps spend a lot of time in data transfer and hence in the 3G radio tail
  - Browsers spend energy in parsing and rendering Javascript, CSS, decompressing JPEG images
  - Angrybirds or other gaming apps spend energy on computing gaming moves
  - Lot of free apps that depend on ads spend energy on GPS (and GPS tail)
  - The code that closes idle TCP connections incurs the penalty of starting the radio for a short period of time (to send FIN) and incurs the radio tail energy again.

How to save I/O energy usage

- I/O activities happens in “bundles”, with wait time between bundles. Why the gap between bundles?
  - User think time
  - App spacing out I/O requests to rate limit bandwidth
  - Closing TCP connections after a timeout
- The gap between bundles incurs wasteful radio tail energy
- Network optimizations like TCP connection resue and cleanup after a timeout hurt energy usage
- Energy saving by grouping together I/O activity is a possible idea.

How to save energy spent on computation?

- Browsers spend energy on
  - Parsing and render Javascript and CSS
  - Decompress images (e.g., JPEG) to BMP for display
- Other apps spend energy in computations such as speech recognition or computing future moves of games
- Two ways to reduce computation in a browser
  - Front-end proxy that optimizes the web page for mobile rendering (e.g., low resolution images). Suitable for browsers.
  - Offloading some computation to a remote server. More suitable for voice recognition kind of apps.
- See the reference “Who Killed by Battery: Analyzing Mobile Browser Energy Consumption” for more details on the two approaches.

Code offloading or remote computation

- Example: Voice recognition feature “Siri” in the iPhone. Voice sample from phone is analyzed and parsed on remote Apple servers, in order not to consume resources on phone.
- Tradeoff
  - Remote computation will save CPU cycles (hence energy) and processing time
  - Network transfer will consume energy (depends on how much code to exchange) and may add to latency
- Also, only some code can be offloaded. Cannot offload code that depends on I/O, camera etc.
- Code partitioning can be manual or automatic.
- See the reference “MAUI: Making smartphones last longer with code offload” for more details. MAUI asks programmers to mark code that can be remotely executed. At runtime, MAUI automatically makes decisions on what to offload based on the energy saved by offloading computation vs. the energy spent to transfer state to remote server.