Using Taint Tracking to Improve Energy Efficiency of Always-on Smartphone Apps

Haichen Shen (Student author), Aruna Balasubramanian, Anthony LaMarca, and David Wetherall
haichen@cs.washington.edu, arunab@cs.washington.edu, anthony.lamarca@intel.com, djw@cs.washington.edu

1. Motivation

Sensor monitoring apps on smartphones provide a variety of services, ranging from muting the ringer in certain locations, to evaluating Parkinson patients, to encouraging people to avoid junk food. Unfortunately, sensor apps often require continuous monitoring and are heavy drain on the mobile's battery. Since sensors are on the same controller as the CPU, taking a sensor reading for a few microseconds results in the CPU being awake for hundreds of milliseconds.

To address this issue, researchers have proposed a *sensor hub* to collect and process sensor data in an always-on micro-controller, distinct from the main processor. While these systems have been shown to be effective for simple applications such as a pedestrian step counter, it is unclear how much savings a sensor hub offers a complex application. The issue is how the applications uses sensor data. If applications buffer sensor readings and process them periodically, they can easily be partitioned to the sensor hub. However, if they process sensor data as they are read, a sensor hub may not improve performance. Without understanding sensor usage, it is difficult to design a general sensor hub architecture.

We propose that *taint tracking* is a useful tool in investigating this issue. By tainting sensor data as it is read, we can track it through a running system and determine when sensed data produces user-perceivable events. We then can determine which sensor data contributed and how old it was. We can also look at the algorithmic transformation the data underwent. In this way, we can, without performing a static code analysis, determine how applications make use of sensors and whether they would be good candidates for optimization via a sensor hub.

2. Taint Tracking

We use *taint tracking* to systematically track data from the sensor to a user-perceivable action. We adapt a system called TaintDroid [1] to perform this tracking. TaintDroid is a privacy monitoring system that watches for private information leaks over a network. Our goal is different, and we make several modifications to TaintDroid accordingly. To track timeliness of data, our modified TaintDroid assigns a unique tag to each tainted variable for each sensor data; for example, the 2nd accelerometer reading has a different tag than the 3rd. We also log when the tainted variable is used in a user-perceivable manner. In our implementation, we say a tainted variable is user-visible if it is sent over the network, written to a file, or displayed on the screen. As a result, our system can track when sensor data is collected (based on tainting a variable), and when sensor data contributes to a visible side-effect (based on the log). Most importantly, we add the ability to track taints through control flow; TaintDroid only tracks taints through data flow. To add control flow tainting, we identifying potential taint blocks, for example, an *if* statement conditioned on a tainted value. We then track the flow of sensor data through the taint block. Tainting control flow is essential for sensing applications, because these applications often perform computation based on the value of the sensor reading; for example, based on whether the sensor reading is greater than a threshold value. Since these computation do not retain the actual sensor value, data flow taints alone is insufficient.

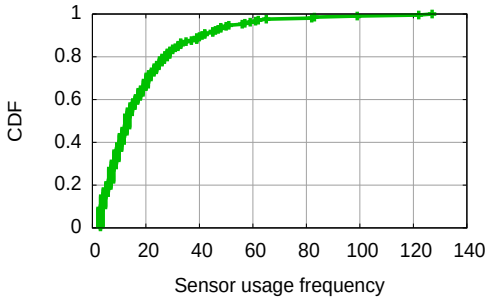

Figure 1: The sensor usage of a pedometer app.

Figure 1 show the results of our taint analysis on an off-the-shelf pedometer app. As an example reading, the point (30, 0.8) shows that 20% of the time, the app only updates once in a user-perceivable manner after 30 contiguous sensor readings. If the 30 contiguous readings were buffered for those 20% cases, the application could have reduced energy consumption by 95% for those cases.

3. Next Steps

We are currently analyzing many popular off-the-shelf sensor applications using our tool. The goal of our analysis is to understand the energy implications of different sensor hub tasks, for example, sensor data buffering, simple sensor thresholding, etc, for a given application, under varying scenarios. We believe that taint tracking is a powerful tool for not only understanding sensor usage in applications, but can also inform the design of system architectures that leverage sensor hubs to improve energy efficiency.

4. References

[1] W. Enck, P. Gilbert, B.-G. Chun, L. P. Cox, J. Jung, P. McDaniel, and A. N. Sheth. Taintdroid: an information-flow tracking system for realtime privacy monitoring on smartphones. In *OSDI*, 2010.