SmartSensors: Energy-efficient sensing on mobile devices

1. BACKGROUND

Mobile application developers can choose between several approaches when attempting to perform continuous sensing. The availability of some of these approaches is dependent upon the mobile device's hardware and overlaying operating system.

The most basic approach that can be used on any sensor-enabled mobile device is to have the mobile device always on. On the Android operating system, the application would acquire a wake lock to prevent the device from sleeping and register a listener for a specific sensor. Whenever the sensor produces a new reading, the operating system passes the reading to the application via a callback. This will continue to happen until the listener is unregistered or until the wake lock is released.

To avoid keeping the device awake for extended periods of time, a duty cycling approach can be used. In such an approach, the device is allowed to sleep for fixed, usually regular, periods of time. An application can implement duty cycling by scheduling and alarm using the system's alarm service and ensuring it does not have have a wake lock acquired. The application is then notified by the operating system when the sleep period ends. At this time, the application can collect sensor readings for a period of time, then reschedule another alarm. This approach has a couple of apparent drawbacks. First off, events of interest will not be detected if they occur while the device is sleeping. Secondly, the device might wake-up to discover that the event of interest is not actually occurring. This approach may be slightly improved by dynamically adjusting the length of time readings are collected for. If after a wake-up, the application detects an event of interest, it can continue to collect sensor readings until events of interest are not being detected anymore, at which time it can return to its regular duty cycling schedule.

The newly released Google Nexus 5, running Android 4.4, introduces an extension to duty cycling referred to as sensor batching. This approach requires hardware support and is currently not available on any devices other than the Nexus 5. In this approach, the device hardware can collect sensor readings while the main processor is sleeping. At wake-up, the entire batch of sensor data is delivered to the application(s). Batching has the great advantage that the applications receive all the sensor data, meaning that no events of interest will be missed. However, this comes at the cost of detection timeliness. Applications may detect events of interest many seconds after they actually occurred. Also, as with the duty cycling approach, the device may wake up to find out that no events of interest have occurred in the current batch.

Multi-processor architectures allow computation offloading and wake-ups via interrupts. A particular architecture may limit the degree of programmability of the peripheral processor. For example, in Refex[1], the authors show an architecture where the application developers can program their own sensor data analysis algorithms that are to be executed on the peripheral processor. On the other hand, recent mobile devices limit computational offloading to a small, pre-defines set of activities. For example, the Motorola Moto X uses two low-power peripheral processor to wake-up the device when certain events occur. A dedicated natural language processor is used to wake-up the device when the user says the phrase "OK Google Now" and a contextual processor is used to turn on part of the sceen when the device is taken out of a pocket or turn on the camera application when two twists of the wrist are detected. Additionally, Android 4.4 allows future devices (if they contain the appropriate sensors) to wake-up on trigger events such as "significant motion" or a step. However, the application developers have no control over the underlying algorithms or customization of parameters.

2. REFERENCES

[1] X. Lin, Z. Wang, R. LiKamWa, and L. Zhong. Reflex: using low-power processors in smartphones without knowing them.