ModeSens: An Approach for Multi-modal Mobile Sensing

Ahmed Abdel Moamen and Nadeem Jamali

Department of Computer Science, University of Saskatchewan, Saskatoon, SK, Canada

ama883@mail.usask.ca, jamali@cs.usask.ca

Abstract

The growing ubiquity of personal connected devices has created the opportunity for a range of applications which tap into their sensors. The sensing requirements of applications often dynamically evolve over time depending on contextual factors, evolving interest in different types of data, or simply to economize resource consumption. The code implementing this evolution is typically mixed with that of the application's functionality. Here we separate the two concerns by modeling the evolution of sensing requirements as transitions between modes. The paper describes ModeSens, an approach to modeling and programming `multi-modal` sensing requirements of applications. The approach improves programmability by enhancing modularity. Our experimental evaluation measures the performance and energy costs of using ModeSens.

Categories and Subject Descriptors D.1.3 [*Software*]: Programming Techniques – Concurrent Programming

Keywords Multi-modal sensing, ModeSens, Actors, FSM

1. Introduction

An important opportunity for conserving energy for sensor-based applications lies in optimizing sensing for the evolving context that a device is in. Consider, for example, a health-monitoring application which monitors a device owner's heart beat at a low frequency, but switches mode to monitor several sensors when an unexpected pattern is detected. We call this multi-modal sensing.

Multi-modal behavior has previously been examined in various mobile device contexts: to filter actions to be carried out [9], to predict user behavior based on their mode [4], to recommend optimal routes based on traffic flow mode [8], or selecting promising device in a "sensing as a service" context [7]. In contrast, we are specifically interested in *sensing modes*, which tell us the sensing requirements of an application's environmental, interest or resource context.

2. Model and Implementation

Multi-modal sensing is naturally modeled using a finite state machine $M = \langle S, \Sigma, \delta, s_0 \rangle$, where S and Σ are non-empty finite sets of states and inputs, $s_0 \in S$ is the initial state, and δ: $S \times \Sigma \rightarrow S$ is the state transition function. Σ contains triggers for mode change, defined in terms of recently sensed data. Sensing of specific data can fire a trigger $t \in \Sigma$ which leads to state transition from s_i to s_j.

We have prototyped such a finite-state machine for multi-modal sensing as a self-contained mobile application implemented over the CyberOrgs [5] extension of Actor Architecture (AA) [6] ported to Android. AA is a platform for implementing systems of actors [3], which are concurrent objects which communicate using asynchronous messages; CyberOrgs help coordinate actors' resources.

Data from several sensors – GPS, accelerometer, microphone, magnetometer, gyroscope, pressure, humidity, temperature and light sensors – can be collected. Higher-level trigger events have been pre-implemented in terms of these (low-level) sensor events – as executable specifications of 7 to 18 lines of code – which the developer can draw from and customize by providing parameters.

3. Evaluation

We evaluate ModeSens in two ways. First, we state the programmability benefits of separating sensing mode transition concerns from functional concerns of applications. Second, we present experimental results on performance and energy costs of using ModeSens. Note that the latter are intended more to document these costs than to compare with the alternative of mixing mode transition concerns with functional concerns, which would obviously still have the same monitoring, triggering and mode transition costs.

Programmability. Our approach to multi-modal sensing offers programmability advantages by separating concerns of the sensing mode transition from the application's functional code, which would otherwise be (and are) mixed. To simplify the specification of the multi-modal system, we have developed a simple graphical user interface to allow programmers to specify the finite state machine for the mode transitions, by defining new modes, transitions between modes, and any outputs to the application.

Experimental Evaluation. Our experimental evaluation used a specific case study involving human activity recognition as a sensing application, which ran on a Samsung Note II

(1.6 GHz quad-core, 2 GB RAM, Android 5.1). We defined four activity modes: stilling (i.e., being still), walking, bicycling and driving. The stilling mode required accelerometer data, walking mode required accelerometer and gyroscope data, and the bicycling and driving modes required accelerometer and GPS data. In addition, the mode transition logic also relied on sensed data. Although not required in general, we assumed that all sensors were to be sampled to detect mode transition triggers at a sampling rate of $1Hz$. In practice, these samples could already be available because of sensing for other applications as shown in [1].

Performance. We separately measured the ongoing cost of monitoring for detecting mode transition triggers, and the sensing and processing delays in carrying out the triggered mode transitions.

The trigger detection mechanism checked for a trigger on arrival of every new feed set, and examined a window of recently sensed data to detect a trigger. We found the trigger detection cost to depend primarily on the size of the window of recently sensed data considered. For our case study, we used a window of size 12, which seemed sufficient for detecting mode transition triggers.

Table 1 shows the measured ongoing processing cost of detecting mode transition triggers for each mode. The time (in *ms*) is for both acquiring and processing the set of feeds (one feed from each sensor) to check for a transition trigger. *Old* applies when data already being collected for the mode's function can be utilized; *New* applies when fresh sensing is required. Because the sensing for detecting triggers was at the rate of $1Hz$ in our case study, this cost is also in *ms/s*. The delays in transitioning individual sensors from a current sampling rate to a different one were measured to be $6.21ms$ (*sd:* 1.21), $10.71ms$ (*sd:* 1.76) and $17.39ms$ (*sd:* 2.36) for the accelerometer, gyroscope and GPS sensors, respectively. These led to (symmetric) delays of $12.32ms$ for Walking-Stilling transition, $19ms$ for Stilling-Bicyclying and Stilling-Driving, $31.32ms$ for Walking-Stilling and Walking-Driving, and $1.61ms$ for Driving-Bicycling. These costs involved the sampling rate changes for the sensors involved plus a small amount of processing cost required to call for the changes. The sampling rate change cost was incurred only if a previously unsampled sensor was to be sampled in the new mode, or vice versa.

Energy Consumption. Experiments were carried out to measure the ongoing energy cost incurred by the sensors for additional sensing for detecting mode transition triggers, as well as the cost of changing the sensors' sampling rates to carry out the mode change. We used PowerTutor [10] for energy measurements. The ongoing costs were measured to be $0.63m\mathcal{J}$, $1.24m\mathcal{J}$ and $1.93m\mathcal{J}$ for the accelerometer, gyroscope and GPS sensors, respectively. These per feed set costs amounted to mJ per second, because the sensing for detecting triggers was at the rate of $1Hz$. The cost of changing a sensor's sampling rate was measured to be $2.42m\mathcal{J}$, $3.18m\mathcal{J}$ and $4.05m\mathcal{J}$ for the accelerometer, gyroscope and GPS sensors, respectively.

Table 1. Cost of Detecting Mode Transition Triggers (*ms/s*)

Mode	Accel.	Gyro.	GPS	**Total**
Stilling	Old: 0.39	New: 4.50	New: 7.31	12.2
Walking	Old: 0.39	Old: 0.39	New: 7.31	8.09
Bicycling	Old: 0.39	New: 4.50	Old: 0.39	5.28
Driving	Old: 0.39	New: 4.50	Old: 0.39	5.28

4. Conclusions

We presented ModeSens, an approach to programming mode transition concerns of multi-modal sensing applications separately from their functional concerns. The mode transition logic can be easily specified using an appropriate finite state machine, for which we have implemented a simple GUI. Our evaluation was two-fold. First, we stated the obvious programmability benefits of using ModeSens. Second, we presented experimental results documenting the processing overhead, sensing delays and energy costs involved in using ModeSens for achieving mode transitions. In on-going work, we are examining the composition of ModeSens with ShareSens [1] to support opportunistic sharing of dynamically evolving sensing requirements, particularly in the context of crowd-sourced services [2].

Acknowledgments

NSERC and CFI support is gratefully acknowledged.

References

[1] A. Abdel Moamen and N. Jamali. ShareSens: An approach to optimizing energy consumption of continuous mobile sensing workloads. In *Proc. of Mobile Services*, pages 89–96, NY, USA, 2015.

[2] A. Abdel Moamen and N. Jamali. CSSWare: A middleware for scalable mobile crowd-sourced services. In *Proc. of MobiCASE*, pages 1–18, Berlin, Germany, 2015.

[3] G. Agha. Actors: A model of concurrent computation in distributed systems. *MIT Press, Cambridge, MA, USA*, 1986.

[4] D. Chiu, O. Lee, H. fung Leung, E. Au, and M. Wong. A multi-modal agent based mobile route advisory system for public transport network. In *Proc. of HICSS*, pages 92–99, Hawaii, 2005.

[5] N. Jamali and X. Zhao. Hierarchical resource usage coordination for large-scale multi-agent systems. In *Massively Multi-Agent Systems I*, pages 40–54. Springer, 2005.

[6] M. Jang, A. Momen, and G. Agha. Efficient agent communication in multi-agent systems. In *Proc. of SELMAS*, pages 236–253, UK, 2004.

[7] P. Jayaraman, C. Perera, D. Georgakopoulos, and A. Zaslavsky. Efficient opportunistic sensing using mobile collaborative platform mosden. In *Proc. of CollaborateCom*, pages 77–86, USA, 2013.

[8] A. Matic, V. Osmani, A. Maxhuni, and O. Mayora. Multi-modal mobile sensing of social interactions. In *Proc. of PervasiveHealth*, pages 105–114, San Diego, USA, 2012.

[9] Y. Wang. An FSM model for situation-aware mobile application software systems. In *Proc. of ACM-SE*, pages 52–57, Alabama, 2004.

[10] L. Zhang, B. Tiwana, Z. Qian, Z. Wang, R. P. Dick, Z. M. Mao, and L. Yang. Accurate online power estimation and automatic battery behavior based power model generation for smartphones. In *Proc. of CODES+ISSS*, pages 105–114, USA, 2010.