MoSen: A Middleware for Mobile Sensor Programming

Kshitiz Bakhshi
Indraprastha Institute of Information Technology
New Delhi, India
kshitiz10040@iiitd.ac.in

Nishant Jain
Indraprastha Institute of Information Technology
New Delhi, India
nishant10058@iiitd.ac.in

Arjun Ahuja
Indraprastha Institute of Information Technology
New Delhi, India
arjun10021@iiitd.ac.in

Pushpendra Singh
Indraprastha Institute of Information Technology
New Delhi, India
psingh@iiitd.ac.in

ABSTRACT

Variety of mobile phone applications now seek access to the sensors in-built into the phone. Such applications require access to sensors for either data collection or for responding to events generated by sensors. Different mobile operating system provide different sets of APIs for accessing the sensors thus requiring rewriting of code to make it portable.

In this work, we are presenting MoSen, a mobile sensing middleware that presents a common interface for accessing in-built sensors in different mobile OS's thus saving considerable effort in mobile sensor programming while not affecting the performance of the device.

Categories and Subject Descriptors

D.2.13 [**Software Engineering**]: Reusable Software

Keywords

Middleware, mobile, sensors

1. INTRODUCTION

Mobile phones have now become the pervasive computing environment with much higher penetration than personal computers. Majority of these phones are smart-phones that are capable of running high-end applications. Today's mobile platform is dominated by iOS and Android where each of them have billions of applications available in their respective app stores. Most of these applications are common and have been ported to support the other platform. For an application to run on multiple mobile OS platforms, a considerable amount of redevelopment effort is required - mostly duplicating the same functionality. Most of these applications make use of sensors in-built into the mobile phones, e.g., accelerometer, GPS, proximity sensors etc. Since different OS's provide different sets of APIs to access their sensors; hence, the code to access sensors needs to be rewritten with the new ported application. While GUI programming is very much tied with a specific platform, we believe that we can provide an abstraction for sensor programming. This will help a developer in keeping a uniform interface of sensor related code, and directly reuse the common code across platform while porting an application. Existing approaches require each application to develop its own framework for sensor data processing. Most of the sensor application simple collect data, especially in research community, which is then used for analyzing. Such application end up collecting same data repeatedly resulting in unnecessary energy consumption and data redundancy. Recently, researchers have been working to overcome the heterogeneity barrier in different contexts of mobile phone applications [1] [2]. In their recent work, Chaudhri et al. [3], aim to provide a unified interface for external sensors for Android plat-form. On similar lines, our middleware enables uniform interface for in-built sensors across platforms thus overcoming heterogeneity at the mobile OS level while providing uniformity in accessing sensors.

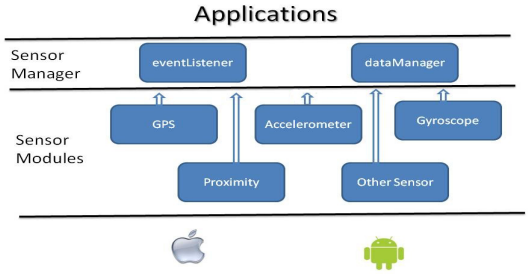

Figure 1: MoSen Architecture

2. DESIGN AND IMPLEMENTATION

While implementation is ongoing, MoSen follows a modular design, see Figure 1. MoSen is exposed to native application as a library. The application interacts with MoSen via the sensor manager interface which exposes available sensor

DEV '13 January 11-12, 2013 Bangalore India

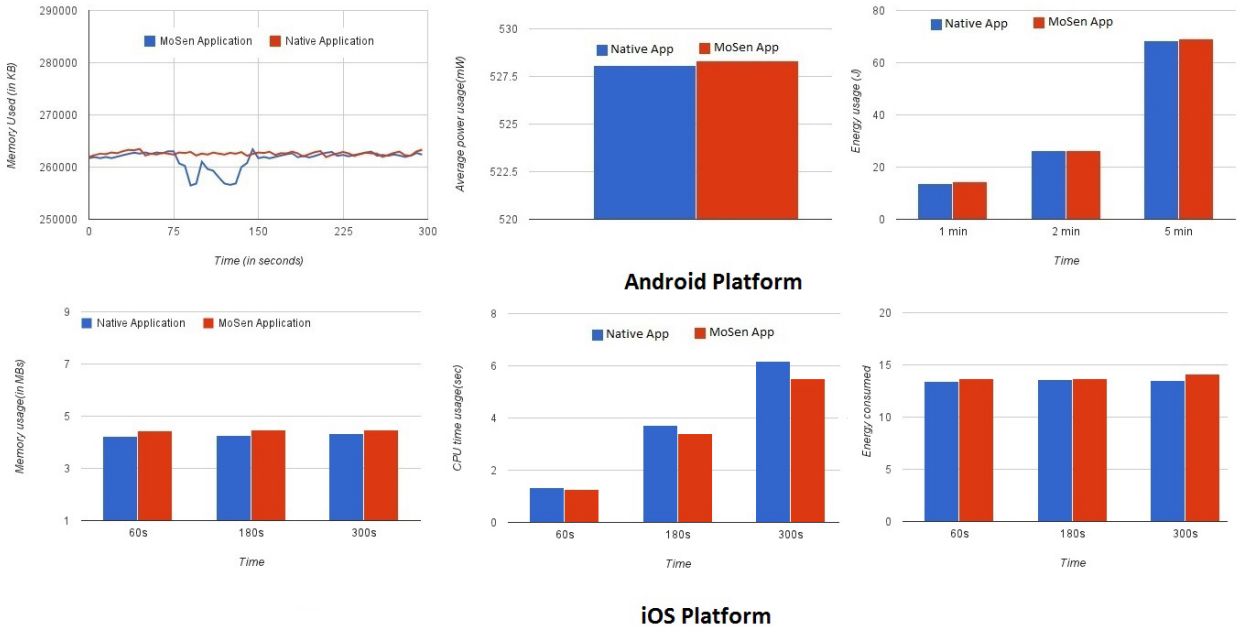

Figure 2: Performance data of tests conducted on iOS and Android platforms

Table 1: Line of code advantage

	Native App.	MoSen App
Android	133	18
iOS	147	17

modules. To support a new sensor, a new module for that sensor is added. In current implementation, MoSen supports two modes of accessing sensor data: 1)event-based notification: an application can register itself to receive notification from a sensor 2) logged data: applications can also request to log data in a file and get access to logged sensor data. The logged data can then be used by other applications as well. We believe that most of the mobile applications fall in above two categories. The MoSen platform provides a common API interface to variety of in-built sensors of mobile phones running on either iOS or Android. A reusable code base further reduces effort at the developer side for all sensor related programming. Essentially, MoSen can become part of any application that interfaces with sensors on Android or iOS platforms (later to Windows Phones as well). MoSen currently supports accelerometer, GPS, gyroscope, and proximity sensors on both iOS and Android platforms and additional light sensor on Android. The code base of MoSen is currently 832 lines for Android and 625 lines for iOS. Support for other sensors, e.g., sound, video, virtual sensors (e.g. phone orientation) etc. is ongoing. MoSen is implemented in Java for Android and in Objective C for iOS. Application developer can include MoSen as binaries in their programs. The native implementation of MoSen does not adversely affects the performance of the application. The source code will also be made available as open source.

3. PRELIMINARY RESULTS & CONCLUSION

To test MoSen, we developed a simple native application for iOS and Android to access sensors and collect sensor data in a file, such applications have been widely used by research community. We also developed another set of applications, of same functionality, using MoSen. The Table 1, shows the reduced line of codes. We then tested the performance of applications in terms of memory, CPU time, and energy consumption. We found that there is almost negligible effect of using MoSen on these vital parameters see figure 2. The advantages of reduced code base, unified design, and simple interface to access sensors provide strong reasons to use MoSen. In future, MoSen will support other sensors also making it a robust sensor programming platform. MoSen will also be exported to Windows Phone in future.

4. REFERENCES

[1] E. Andriescu, R. Speicys Cardoso, and V. Issarny. Ambistream: a middleware for multimedia streaming on heterogeneous mobile devices. *Middleware 2011*, pages 249–268, 2011.

[2] A. Bennaceur, P. Singh, P. Raverdy, and V. Issarny. The ibicoop middleware: Enablers and services for emerging pervasive computing environments. In *Pervasive Computing and Communications, 2009. PerCom 2009. IEEE International Conference on*, pages 1–6. IEEE, 2009.

[3] R. Chaudhri, W. Brunette, M. Goel, R. Sodt, J. VanOrden, M. Falcone, and G. Borriello. Open data kit sensors: mobile data collection with wired and wireless sensors. In *Proceedings of the 2nd ACM Symposium on Computing for Development*, page 9. ACM, 2012.