2012 9th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)

MobiCon: Mobile Context Monitoring Platform

Incorporating Context-awareness to Smartphone-centric Personal Sensor Networks

Youngki Lee, Younghyun Ju, Chulhong Min, Jihyun Yu, Junehwa Song
KAIST, Daejeon, Korea
{youngki, yhju, chulhong, jihyun, junesong}@nclab.kaist.ac.kr

***Abstract*— In this demonstration, we will show *MobiCon*, a context monitoring platform; it runs over smartphones and sensor OSs, and facilitates development and deployment of everyday context-aware applications. For many years, lots of research efforts have been made in building low-cost, yet effective sensor networks for various application domains such as structural health monitoring of bridges, disaster recovery, automated ventilation of buildings. Integration of sensors into smartphones and the advent of wearable devices open a new opportunity for mobile applications to leverage in-situ user contexts such as his/her location, activity, social relationship, health status. In recent studies of mobile and pervasive computing, a number of useful mobile context-aware applications have been proposed, but their actual deployment is slow due to complexity of context processing and heavy resource and battery usage. To address such challenges, we have been building MobiCon for many years, upon which diverse context-aware applications are developed and deployed without concerns about complexity of context processing and resource optimization.**

Keywords-Context, Mobile, Platform, Sensing, Resource, Energy

I. INTRODUCTION

Smartphones are close to their owners almost 24/7, longer than close friends or family. They have inherent opportunities to catch users' situations and contexts better than anyone else. Such opportunities are increasingly realized with recent integration of sensors into smartphones. Today's smartphones embed diverse sensing modules such as camera, microphone, accelerometer, gyroscope, and GPS. Sensing capabilities will be further enriched with the aid of a variety of wearable sensors and nearby space-embedded ones. With such sensors, smartphones will be aware of in-situ user contexts in real-time, such as activity, location, emotion, health, and surroundings.

Current mobile applications, however, are not fully taking advantage of the new opportunities. Most applications are simple versions of traditional desktop applications, adapted to mobile environments, e.g., small touch screens and battery-operated devices. Example applications include web browsers, e-mail, games, and twitter. Some applications provide mobile-specific features by utilizing phone-embedded sensors, but they still remain in a basic level. The leveraged user contexts are limited to primitive ones that can be directly extracted from sensing data or through simple processing. Also, users should explicitly trigger the applications to make use of their services. For example, Google Maps present a user's current location on the map by using GPS when the user launches the application.

We envision that future mobile applications will evolve to leverage in-situ user contexts more actively. The contexts they utilize will extend to the ones that are obtainable by applying complex steps of processing and advanced machine learning techniques. Also, they will adopt additional peripheral sensors to identify more diverse contexts. They will continuously understand user contexts in real-time, providing automatic and user-unobtrusive services. For example, an exercising application can identify a user's activities by analyzing sensing data from multiple on-body inertial sensors, and continuously notify the user of her calorie expenditure while she is jogging.

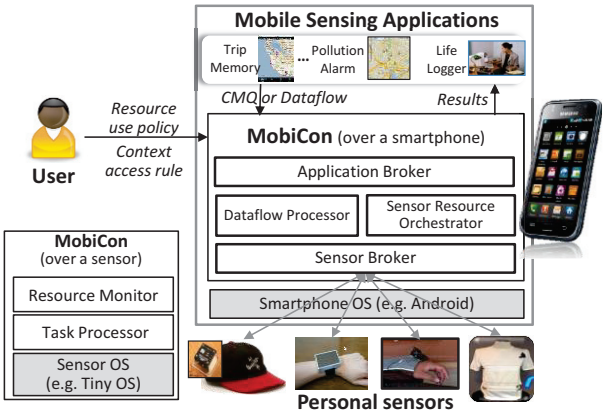


Figure 1. MobiCon architecture overview

Recently, plenty of advanced mobile sensing applications have been proposed in research communities. However, applications in practice only use sensing capabilities just in a primitive level. A major cause stems from the lack of support of current mobile OSs such as iOS and Android; they provide applications only with low-level interfaces for simple sensor data reading over phone-embedded sensors. Accordingly, developers should address many new challenges by themselves. First, they should devise sophisticated logics to extract high-level context from raw sensing data. Also, they should deal with heterogeneous and distributed sensing devices, which have different computing capabilities and programming environments. More difficult, they should make enormous effort for optimization for continuous sensing and processing; the optimization is even more difficult since resource availability of sensor devices and demands from other concurrent applications change dynamically.

To address the challenges and accelerate the proliferation of new mobile sensing applications, we have been developing a novel mobile platform, named MobiCon since 2007. In this demonstration, we will show several key functionalities of the platform, which will be briefly introduced in Section II.

II. MOBICON ARCHITECTURE OVERVIEW

MobiCon is a middleware system mediating the mobile and sensor OSs and mobile context-aware applications. Fig. 1 shows the overall architecture of MobiCon system and its environments. Three major entities interact as follows:

978-1-4673-1905-8/12/$31.00 ©2012 IEEE

Applications that need to sense and extract contextual information register their context monitoring requests to MobiCon through the MobiCon APIs.

MobiCon runtime processes concurrent application requests effectively while coordinating and adapting the resource use of applications for sensing and processing.

Mobile user specifies his preference about resource utilization and privacy. MobiCon applies such user policies to coordinate its resource use and control the usage of contexts.

In the rest of this section, we briefly introduce MobiCon APIs, and several key functionalities of the runtime.

A. *Application Interfaces*

MobiCon incorporates two types of APIs to support, (1) applications that require common user contexts (location, activity, environment, etc.), and (2) applications that require highly customized processing over sensor data. For the former, MobiCon provides a declarative query interface, called *context monitoring query* (CMQ) [2][3][4]. Through CMQ specification, applications easily obtain contexts of interests by delegating complications in context monitoring to the platform. They do not need to concern about which feature extraction and classification modules to apply, how much resources (e.g., CPU and battery) are available, etc. For the latter, MobiCon provides a dataflow programming interface that allows the flexible specification of customized context monitoring logic as a dataflow graph of operators [1]. Developers can easily build a dataflow graph by composing MobiCon-built-in operators and application-specific operators.

B. *Sensor Resource Orchestration*

A key feature of MobiCon runtime is to actively coordinate resource use of concurrent applications over distributed personal sensing devices [3]. Once applications turn in CMQ specifications, the system finds the best combination of sensor resources to process the contexts on-the-fly considering current resource status of sensor devices and applications. Specifically, MobiCon first prepares multiple alternative processing plans to monitor a high-level context; each plan utilizes different combination of sensor devices and their resources, providing opportunities for MobiCon to flexibly coordinate the resource use of applications. To prepare alternatives, it leverages the diversity of semantic translation: a context can be often derived from different sensing modalities, feature sets, and classifiers. For instance, a user's '*walking*' is monitored by the frequency domain analysis from acceleration data or statistical processing of GPS data. Among such alternative plans, MobiCon executes the best combination for concurrent requests, holistically considering diverse system inputs; 1) the resource demand of concurrent applications, 2) resource availability of devices, and 3) system-level policies. Through active orchestration, it helps applications share in resources and processing with a holistic view on the applications and resources. Also, MobiCon resolves resource contention between applications. Moreover, it helps the applications adapt to dynamic sensor membership and their resource availability applying alternative plans.

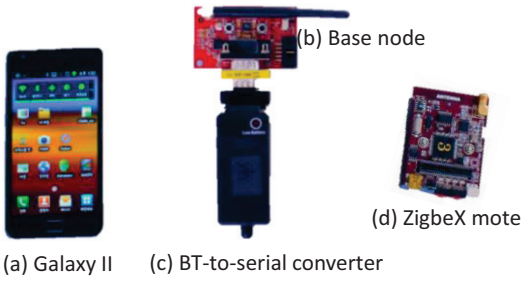


Figure 2. Hardware setup

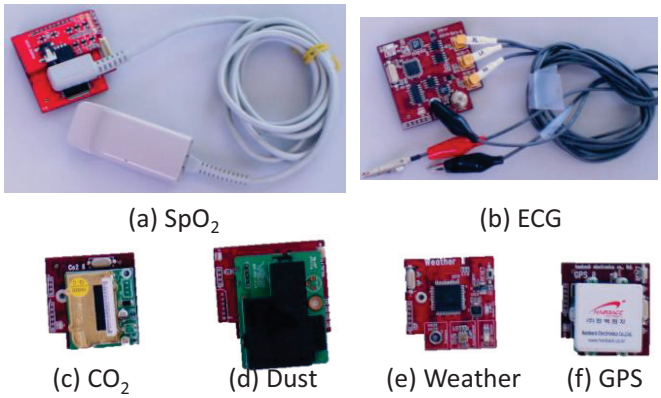


Figure 3. Sensing modules

C. *Sensor Broker for Personal Sensing Devices*

In MobiCon, a sensor broker takes an important role for close cooperation between the mobile device and sensor devices. For effective cooperation, we first develop a range of protocols to exchange data and control messages, e.g., *sensor detection protocol*, *sensor control protocol*, and a *data reporting protocol*. Based on the protocols, sensors communicate with a mobile device. In addition, we further devise an energy optimization method for the protocols since communication is the most energy-consuming tasks of sensor devices. Finally, to incorporate diverse sensor devices using heterogeneous communication protocols and message formats, we encapsulate such sensor-dependent parts by defining common interfaces and a message protocol. This enables other components such as the dataflow processor to operate independently of sensor-specific details.

D. *Energy- and Processing-Efficienct Context Processor*

Another key component of MobiCon is an efficient context processor, which continually processes high-rate sensor data to extract high-level context information. Energy and processing efficiency of the context processor is critical as the resources of mobile and sensor devices are significantly limited while the context processing involves continuous execution of energy- and processing-intensive operators. If the context processor continuously imposes high CPU overhead on smartphones, the performance of other user-interactive applications such as web browsing could be significantly degraded. Also, its continuous energy use decreases the battery lifetime of the smartphones considerably, disrupting other common use of smartphones. To achieve processing- and energy- efficiency, we have developed diverse techniques e.g., incremental and shared processing of sensor data [2] and query-structure-aware sensor control

TABLE I CONTEXTS AND SENSORS THAT MOBICON SUPPORTS

Context type	Context value examples	Sensors
Physical activity	Standing, sitting, walking, running	3-axis accelerometers
Heart rate	Dangerous, rapid, normal, slow, too slow	SpO_2 or ECG sensor
Temperature	Hot, warm, cool, cold	Weather sensor
Dust	Danger, terrible, bad, sensitive, normal, good	Dust sensor
CO_2	Worst, pretty bad, a little bad, allowed, normal	CO_2 sensor
Outdoor location	Logitude, latigude	GPS
Surrounding activity	Chatting, music, reading	Microphone

methods [2], minimizing dataflow execution overhead [1], and incorporated the techniques into MobiCon runtime system.

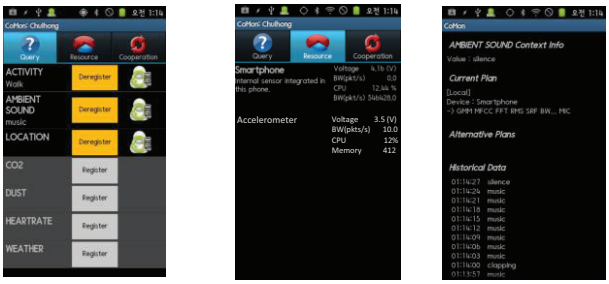
III. DEMONSTRATION

In the demonstration, we will show key functionalities and representative operation cases of MobiCon with our prototype system. We have developed the MobiCon system on Android phones and various types of external sensor devices. Fig. 2 shows the hardware setup for the demonstration. For the sensor devices, we will use ZigbeX II sensor motes (MicaZ clone, Fig. 2 (d)). Each mote incorporates Atmega 128L MCU with 4KB data EEPROM, CC2420 RF transceiver supporting 2.4GHz band ZigBee protocol, 1Mb external flash, 3 LEDs, and additional extension board for sensing modules. With the extension board, diverse sensing modules are selectively attached (see Fig. 3 for the modules). To connect between smartphones and sensor nodes, we used a base node with a Bluetooth-to-serial converter as shown in Fig. 2 (b) and (c).

Our demonstration consists of three major parts: (1) basic context monitoring capabilities of MobiCon, (2) resource orchestration to support concurrent requests, and (3) highly optimized context processing (see Fig. 4 for screenshots).

First, we plan to demonstrate basic context monitoring functionalities of MobiCon. As an underlying middleware, MobiCon monitors various types of user contexts on behalf of applications. Table I shows the context and sensor types that MobiCon supports. To show the interaction between MobiCon and the applications, we have implemented a simple demo application that allows users to directly register and deregister context monitoring requests as shown in Fig. 4 (a). During the demonstration, participants can easily manipulate MobiCon and identify in-situ user contexts through an intuitive interface. The application also shows the supportability of registered queries according to the join and leave of corresponding sensor devices. For instance, the query regarding CO2 contexts would be deactivated when CO2 sensor is turned off.

Second, we will show resource coordination and adaptation capabilities of MobiCon to support concurrent applications



(a) Query registration (b) Resource monitoring (c) Processing plan

Figure 4. Screenshots

over dynamic mobile sensing environments. To support dynamic sensor membership caused by user mobility and wearable devices, MobiCon continuously detects the changes in available sensor devices and their resources. With available sensors, it actively finds the best combination of sensor devices to process concurrent requests. These are shown in the demonstration with two major parts; (1) real-time resource monitoring and (2) resource coordination for concurrent applications. Fig. 4 (b) shows the screenshot of current sensor resource availability and (c) shows the plans that MobiCon selects for registered requests, respectively. In demonstration, we will dynamically change the sensor membership and application requests and show how MobiCon changes the processing plans for concurrent requests and adapts to dynamic sensor membership.

Lastly, we plan to show the performance benefit of the context processing methods of MobiCon. To show the benefit, we compare the processing cost of MobiCon with that of the alternative system without applying our method (See [1] for details on the alternative). For MobiCon and the alternative, we will display real-time resource usage consumed to process application requests, in terms of CPU utilization, memory usage, battery voltage, and bandwidth (see Fig. 4 (b)).

REFERENCES

[1] Ju, Y., Min, C., Lee, Y., Yu, J., and Song, J. An Efficient Dataflow Execution Method for Mobile Context Monitoring Applications, in PerCom, 2012.

[2] Kang, S., Lee, J., Jang, H., Lee, H., Lee, Y., Park, S., Park, T., and Song, J. SeeMon: Scalable and Energy-efficient Context Monitoring Framework for Sensor-rich Mobile Environments. In MobiSys, 2008.

[3] Kang, S., Lee, Y., Min, C., Ju, Y., Park, T., Lee, J., Rhee, Y., and Song, J. Orchestrator: An Active Resource Orchestration Framework for Mobile Context Monitoring in Sensor-rich Mobile Environments. In PerCom, 2010.

[4] Lee, Y., Iyengar, S. S., Min, C., Ju, Y., Kang, S., Park, T., Lee, J., Rhee, Y., and Song, J. MobiCon: Mobile Context-Monitoring Platform. Communications of the ACM, 2012.

[5] Lee, Y, Ju, Y, Min, C, Kang, S, Hwang, I, Song, J, CoMon: Cooperative Ambience Monitoring Platform with Benefir and Continuity Awareness, In MobiSys, 2012.